ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΠΜΣ ΣΤΗΝ ΕΠΙΣΤΗΜΗ ΥΠΟΛΟΓΙΣΤΩΝ

«Ανάπτυξη κινητής εφαρμογής για σύστημα διαχείρισης εγγράφων»

Διπλωματική Εργασία

Μετατροπή - Migration React Application σε Ionic Application

ΤΖΙΒΡΑΣ ΓΕΡΑΣΙΜΟΣ
Α.Μ.: 2022202202018
ΧΑΥΛΗΣ ΕΥΑΓΓΕΛΟΣ
Α.Μ.: 2022202202021

Τρίπολη | Φεβρουάριος 2024

Περιεχόμενα

Ευρετήριο σχημάτων

Περίληψη

Η παρούσα διπλωματική εργασία αναφέρεται στην ανάπτυξη μιας κινητής εφαρμογής που αφορά ένα σύστημα ηλεκτρονικής διαχείρισης εγγράφων. Πιο συγκεκριμένα, μέσω της εφαρμογής θα μπορεί ο κάθε χρήστης να εκτελέσει κάποιες ενέργειες πάνω στα έγγραφα που επιθυμεί, όπως είναι σύνταξη, επεξεργασία, υπογραφή, πρωτοκόλληση και διανομή αυτών. Καθ' όλη τη διάρκεια ζωής του εγγράφου, θα μπορεί ο χρήστης να ενημερώνεται για την πορεία του, ενώ παράλληλα διατηρείται η ιστορικότητα. Επιπλέον θα μπορεί να αρχειοθετήσει τα έγγραφα του και να τα οργανώσει όπως ο ίδιος επιθυμεί μέσω των εκάστοτε ετικετών. Ακόμα μέσω της εφαρμογής θα μπορεί να δημιουργήσει, να επεξεργαστεί, να δεχθεί και να ολοκληρώσει εργασίες, διευκολύνοντας έτσι το επιτελικό έργο του εκάστοτε οργανισμού. Αξίζει να σημειωθεί ότι η εφαρμογή διαθέτει και ξεχωριστή ενότητα ημερολογίου, με σκοπό να ενημερώνεται ο χρήστης για τη διάρκεια που έχουν οριστεί οι εργασίες που του έχουν ανατεθεί. Στην ενότητα σημαντικών θεμάτων μπορούν να ομαδοποιηθούν έγγραφα και εργασίες που αφορούν ένα συγκεκριμένο θέμα, ενώ μέσω της ενότητας της στοχοθεσίας, γίνεται διαχείριση και παρακολούθηση του εκάστοτε φορέα, οργανισμού, διεύθυνσης, τμήματος. Επιπροσθέτως στην ενότητα του Πρωτοκόλλου μέσω κατάλληλων φίλτρων μπορεί ο κάθε χρήστης να βρει όποιο έγγραφο επιθυμεί και συγκεκριμένες πληροφορίες για αυτό, είτε αφορά εισερχόμενο, είτε εξερχόμενο πρωτόκολλο. Οι ενότητες των αδειών αλλά και αυτή των αντικαταστάσεων αφορά την ύπαρξη συνέχεια στην ομαλή λειτουργία του εκάστοτε φορέα, σε περίπτωση απουσίας του προσωπικού. Μέσω αυτών μπορεί να γίνει μεταβίβαση καθηκόντων σε άλλους υπαλλήλους, αλλά και να ενημερωθεί και να διαχειριστεί ο εκάστοτε προϊστάμενος την απουσία του προσωπικού. Ενώ, η ενότητα των επαφών του οργανισμού, αλλά και επαφών εκτός αυτού, βοηθά στη διατήρηση της καλύτερης επικοινωνίας του οργανισμού τόσο εντός αυτού, αλλά και προς τους έξω από αυτόν. Τέλος, η ενότητα της βιβλιοθήκης λειτουργεί ως ένα μέρος όπου ο χρήστης θα μπορεί να βρει σημαντικά έγγραφα (Νομοθεσίες, Εγκύκλιοι κ.λ.π.).

Η παραπάνω εφαρμογή έχει υλοποιηθεί ως μια web εφαρμογή, με το όνομα «Σύστημα Ηλεκτρονικής Διαχείρισης Εγγράφων ΙΡΙΔΑ», και χρησιμοποιείται από πολλούς φορείς του Δημοσίου. Η ανάπτυξη μιας ξεχωριστής εφαρμογής θα μπορέσει να αποφέρει πολλά θετικά όπως είναι:

- Ευκολία πρόσβασης: Οι χρήστες προτιμούν συχνά να έχουν πρόσβαση σε υπηρεσίες και πληροφορίες από τα κινητά τους, καθώς αυτό επιτρέπει τη χρήση τους από παντού.
- Αλληλεπίδραση με συσκευές: Οι mobile εφαρμογές μπορούν να εκμεταλλευτούν τις δυνατότητες των κινητών συσκευών για να προσφέρουν μια βελτιωμένη εμπειρία χρήστη.
- Ειδικές ειδοποιήσεις και ενημερώσεις: Με τις mobile εφαρμογές, μπορούν να αποστέλλονται ειδοποιήσεις στους χρήστες για νέα περιεχόμενα, ενημερώσεις ή ειδικές προσφορές.
- Βελτιωμένη απόκριση: Οι mobile εφαρμογές συχνά προσφέρουν γρηγορότερη και πιο απευθείας πρόσβαση στην εφαρμογή και τα δεδομένα, χωρίς την ανάγκη να φορτώνουν μια ιστοσελίδα.

- Εξατομίκευση: Οι mobile εφαρμογές επιτρέπουν την εξατομίκευση της εμπειρίας χρήστη, προσφέροντας λειτουργίες και περιεχόμενο που προσαρμόζονται στις ανάγκες κάθε χρήστη.
- Αυξημένη προβολή και αναγνωρισιμότητα: Η διάθεση μιας mobile εφαρμογής μπορεί να βελτιώσει την προβολή και την αναγνωρισιμότητα της εφαρμογής.

Abstract

This thesis refers to the development of a mobile application that implements an electronic document management system. More specifically, using the application a user can perform actions on documents, including composition, editing, signing, protocol number assignment and distribution. During the life span of the document, the user can be notified for the document's current status, while in parallel the history of the document is maintained. Furthermore, the user can file his/her documents and organize them as desired, using tags. Moreover, through the application the user can create, process, accept and fulfill tasks, facilitating the strategic goals of the respective organization. It is worth noting that the application includes a separate 'calendar' section, aiming to keep the user informed regarding the duration of the tasks that have been assigned to him. In the 'Important subjects' section documents and tasks concerning a specific subject can be grouped, while in the 'goal setting' section an organization, directorate or department can be managed and monitored. Additionally, in the 'protocol' section, the user may employ filters to find documents and information on them, for both incoming and outgoing documents. The sections on 'leaves' and 'replacements' concern the operational contingency of the organization when personnel is absent. Using the functionality of these sections, tasks can be transferred to other employees, whereas the head of the department is also notified about the absence of personnel. The 'contacts' section concerns both the employees of the organization and contacts outside its scope, and provides tools for better managing the inter-person communication and interaction. Finally, the 'library' section provides functionality for storing and locating important documents (legislation, circulars, etc).

The abovementioned application had been implemented as a web application, under the name "IRIDA Electronic Document Management System", and is used by many public bodies. The development of a separate application is expected to have numerous benefits including:

- Ease of access: Users often prefer to access services and information from their mobiles, since this mode overcomes time and space barriers.
- Interaction with devices: Mobile applications can take advantage of the capabilities of mobile devices, to provide enhanced user experience.
- Special notifications and updates: With mobile applications, notifications can be sent to users about new content, updates or special offers.
- Improved responsiveness: Mobile applications often offer faster and more direct access to the application and data, without the need to load a web page.
- Personalization: Mobile applications allow the personalization of the user experience, offering functions and content that are adapted to the needs of each user.
- Increased visibility and awareness: Making a mobile application available can improve the visibility and awareness of the application.

1 Εισαγωγή

Καθώς ο κόσμος γίνεται όλο και πιο διασυνδεδεμένος και εξαρτάται από την ψηφιακή επικοινωνία, η ζήτηση για εφαρμογές κινητών τηλεφώνων που διευκολύνουν την κοινή χρήση και τη διαχείριση εγγράφων έχει αυξηθεί εκθετικά. Οι εφαρμογές για κινητά προσφέρουν απαράμιλλη ευκολία και ευελιξία, επιτρέποντας στους χρήστες να στέλνουν, να λαμβάνουν και να διαχειρίζονται έγγραφα εν κινήσει. Το Σύστημα Ηλεκτρονικής Διαχείρισης Εγγράφων "Ίριδα" επιδιώκει να αξιοποιήσει αυτή την τάση και να παρέχει μια απρόσκοπτη λύση για οργανισμούς.

Ο κεντρικός στόχος του "Ίριδα" είναι να απλοποιήσει τη συχνά πολύπλοκη και χρονοβόρα διαδικασία ανταλλαγής εγγράφων. Είτε πρόκειται για την αποστολή σημαντικών αρχείων σε συναδέλφους, είτε για τη συνεργασία σε έργα με μέλη της ομάδας, είτε για την ασφαλή κοινή χρήση ευαίσθητων πληροφοριών, η εφαρμογή για κινητά έχει ως στόχο να αποτελέσει την ιδανική πλατφόρμα για όλες τις ανάγκες που σχετίζονται με έγγραφα.

Ένα από τα βασικά χαρακτηριστικά του "Ίριδα" είναι η συμβατότητα πολλαπλών πλατφορμών. Αναγνωρίζοντας την ποικιλομορφία των συσκευών και των λειτουργικών συστημάτων στο σημερινό τοπίο των κινητών τηλεφώνων και πιστεύουμε ότι η κοινή χρήση εγγράφων δεν πρέπει να περιορίζεται από περιορισμούς πλατφόρμας. Η εφαρμογή έχει σχεδιαστεί για να λειτουργεί απρόσκοπτα τόσο σε συσκευές iOS όσο και σε συσκευές Android, διασφαλίζοντας ότι οι χρήστες μπορούν να έχουν πρόσβαση στα έγγραφά τους από οποιοδήποτε smartphone ή tablet.

Η επιλογή του Ionic Framework ως πλατφόρμα ανάπτυξής είναι μια κρίσιμη απόφαση για την επίτευξη αυτής της συμβατότητας πολλαπλών πλατφορμών. Το Ionic προσφέρει ένα ισχυρό σύνολο εργαλείων και στοιχείων που διευκολύνουν τους προγραμματιστές να δημιουργούν εφαρμογές για κινητά που λειτουργούν ομαλά σε πολλές πλατφόρμες. Στις επόμενες ενότητες διερευνώνται οι θεμελιώδεις πτυχές του Ionic Framework και τον τρόπο με τον οποίο ευθυγραμμίζεται με τους στόχους του έργου μας.

Επιπλέον, η παρούσα εργασία αναφέρει τις λειτουργίες του "Ίριδα" για την κοινή χρήση εγγράφων. Αναφέρονται ακόμα οι τρόποι με τον οποίο η εφαρμογή ενισχύει την αποδοτικότητα με τον εξορθολογισμό των ροών εργασίας εγγράφων, τη μείωση του κινδύνου σφαλμάτων και την ελαχιστοποίηση του χρόνου και της προσπάθειας που απαιτείται για τη διαχείριση των εγγράφων. Διερευνώνται επίσης διάφορες περιπτώσεις χρήσης, από μεμονωμένους χρήστες που στέλνουν προσωπικά έγγραφα έως οργανισμούς που εφαρμόζουν το "Ίριδα" ως μέρος της στρατηγικής τους για τη διαχείριση εγγράφων.

Επιπλέον, το παρόν έγγραφο θα συγκρίνει το Ionic Framework με το React Native, ένα άλλο δημοφιλές πλαίσιο για την ανάπτυξη εφαρμογών για κινητά. Αναλύονται οι διαφορές στις προσεγγίσεις τους, τις διαδικασίες ανάπτυξης και την υποστήριξη του οικοσυστήματος για να παρέχουμε μια ολοκληρωμένη κατανόηση του λόγου για τον οποίο επιλέξαμε το Ionic για το έργο μας.

Στις ενότητες που ακολουθούν, παρουσιάζεται μια λεπτομερή ανάλυση των φάσεων ανάπτυξης που εμπλέκονται στην υλοποίηση του "Ίριδα". Η φάση Pre-Migration θα δώσει έμφαση στη σημασία του προσεκτικού σχεδιασμού, του σχεδιασμού της εμπειρίας χρήστη και των προσδοκιών των χρηστών. Περιγράφεται η δομή των φακέλων και των αρχείων της εφαρμογής, αποδεικνύοντας πώς η αρχιτεκτονική διευκολύνει την αποτελεσματική ανάπτυξη και συντήρηση.

Η φάση του Migration θα εμβαθύνει στις στρατηγικές και τις εκτιμήσεις που εμπλέκονται στη μετατροπή μιας εφαρμογής React σε Ionic. Αναφέρονται το full migration και το gradual migration, για την επιλογή της καταλληλότερης στρατηγικής για το έργο. Θα παρουσιαστούν επίσης πρακτικά βήματα και παραδείγματα μετατροπής κώδικα.

Τέλος, η φάση Post - Migration θα αναδείξει τη σημασία των δοκιμών και της διασφάλισης ποιότητας στη διασφάλιση μιας ομαλής και αξιόπιστης εμπειρίας χρήστη. Παρατίθεται η σημασία των συνεπών στυλ κωδικοποίησης στις εφαρμογές React και παρουσιάζονται παραδείγματα διαδικασιών ελέγχου.

Τέλος, στις επόμενες ενότητες του παρόντος εγγράφου, διερευνώνται πραγματικές μελέτες περιπτώσεων και βέλτιστες πρακτικές για την επιτυχή μετατροπή εφαρμογών React σε Ionic. Παρουσιάζονται επίσης στρατηγικές για τη διατήρηση και την ενημέρωση της εφαρμογής "Ίριδας" ώστε να προσαρμόζεται στις εξελισσόμενες ανάγκες των χρηστών και στις τεχνολογικές εξελίξεις. Το "Ίριδα" αντιπροσωπεύει ένα πολλά υποσχόμενο άλμα προς τα εμπρός στην ανταλλαγή εγγράφων και το παρόν έγγραφο έχει ως στόχο να παρέχει έναν ολοκληρωμένο οδικό χάρτη για την ανάπτυξη και τη συνεχή επιτυχία του.

1.1 Τι είναι η Ίριδα

Το Σύστημα Ηλεκτρονικής Διακίνησης Εγγράφων ΙΡΙΔΑ είναι ένα πληροφοριακό σύστημα που χρησιμοποιείται για τη ηλεκτρονική διαχείριση εγγράφων και εργασιών.

Σκοπός της είναι η αντικατάσταση της έντυπης αλληλογραφίας και της διαδικασίας λήψης υπογραφών σε έντυπο, όπως επίσης και η επέκταση της λειτουργικότητάς της προσφέροντας ένα κεντρικό και ολοκληρωμένο σύστημα διαχείρισης αλληλογραφίας, λήψης υπογραφών και απόδοσης αριθμού πρωτοκόλλου.

Οι κύριοι λόγοι για την οποία χρησιμοποιείται είναι για:

- τη σύνταξη, έγκριση, πρωτοκόλληση και διανομή των εγγράφων, την λεπτομερή καταγραφή των ενεργειών των οργάνων της αλυσίδας των υπογραφών και των σχολίων που διατυπώνονται,
- την πρωτοκόλληση των εισερχόμενων εγγράφων
- την εισαγωγή ψηφιοποιημένων εγγράφων (εκτός ΙΡΙΔΑΣ), μέσω σάρωσης αυτών, για περαιτέρω χειρισμό.
- την ανάθεση εργασιών προς υλοποίηση ενεργειών, που απορρέουν από τα εισερχόμενα έγγραφα
- την παρακολούθηση και συντονισμό του επιτελικού έργου
- την παρακολούθηση σημαντικών θεμάτων.
- το ημερολόγιο με ειδοποιήσεις.
- τον κατάλογο επαφών.
- την στοχοθεσία.
- την διαχείριση αδειών του προσωπικού
- την βιβλιοθήκη σημαντικών εγγράφων (Νομοθεσία, Εγκύκλιοι κ.λ.π.).
- την παραγωγή γραφημάτων εξέλιξης εργασιών.

- την οργάνωση των ηλεκτρονικών εγγράφων του προσωπικού.

Η «ΙΡΙΔΑ» αναπτύχθηκε εξ ολοκλήρου από τα στελέχη του Κέντρου Μηχανογράφησης του Γενικού Επιτελείου Αεροπορίας. Αρχικά εφαρμόστηκε στις τάξεις της Πολεμικής Αεροπορίας και έπειτα προσαρμόστηκε πλήρως στις ανάγκες του Υπουργείου Εσωτερικών και των φορέων Τοπικής Αυτοδιοίκησης Α΄ και Β΄ Βαθμού, στο πλαίσιο της συνεργασίας τους.

Η «ΙΡΙΔΑ» αποτελεί ένα σύγχρονο, ισχυρό, αποτελεσματικό και ασφαλές πληροφοριακό σύστημα – το οποίο φιλοξενείται στην κυβερνητική υποδομή G-Cloud του Υπουργείου Ψηφιακής Διακυβέρνησης – που ενισχύει την αποδοτικότητα, τη διαφάνεια, τον ιεραρχικό έλεγχο και τον συντονισμό των υπηρεσιών και ως εκ τούτου τη συνολική διοικητική ικανότητα του να ανταποκρίνεται και ταχύτητα, ακρίβεια και διαφάνεια στις λειτουργικές απαιτήσεις ενός σύγχρονου οργανωτικού περιβάλλοντος, ενώ παράλληλα επιτυγχάνει σημαντική εξοικονόμηση πόρων.

Με τη χρήση του συστήματος εξασφαλίζεται η εμπιστευτικότητα, η ακεραιότητα και διαθεσιμότητα της πληροφορίας που αφορά στην αλληλογραφία. Η ένταξη του συστήματος «ΙΡΙΔΑ» στη λειτουργία των φορέων, οδηγεί στην αναβάθμιση της κανονιστικής συμμόρφωσης που αφορά στα ανάλογα θέματα, ενώ αυτονόητα οδηγεί και στην αύξηση της ταχύτητας διεκπεραίωσης των θεμάτων. Παράλληλα, σημαντικό πλεονέκτημα είναι η εξοικονόμηση πόρων αλλά και χώρου. Επιπλέον, παρέχει ευελιξία στο φορέα καθώς η χρήση του από τους υπαλλήλους είναι ανεξάρτητη της θέσης εργασίας ή της γεωγραφικής τους θέσης, δίνοντας τη δυνατότητα για εργασία εξ αποστάσεως, με βάση ανάλογο διοικητικό πλαίσιο και εφόσον το επιβάλλουν οι συνθήκες. Τέλος, με την αξιοποίηση του συστήματος, προστατεύεται το περιβάλλον καθώς ο φορέας περιορίζει δραστικά την κατανάλωση χαρτιού (λιγότερες εκτυπώσεις), ενώ επιπλέον μειώνει και τις μετακινήσεις που αφορούν στη διακίνηση εγγράφων.

1.2 Λόγοι δημιουργίας εφαρμογής κινητού

Σε μια ταχέως εξελισσόμενη ψηφιακή εποχή, οι μη κερδοσκοπικοί οργανισμοί αντιμετωπίζουν την πρόκληση της προσαρμογής στις μεταβαλλόμενες προτιμήσεις επικοινωνίας και δέσμευσης του κοινού-στόχου τους. Ενώ οι διαδικτυακές εφαρμογές αποτελούν βασικό στοιχείο για τους οργανισμούς αυτούς, η σημασία της συμπλήρωσης της διαδικτυακής τους παρουσίας με μια εφαρμογή για κινητά γίνεται ολοένα και πιο εμφανής. Παρακάτω παρουσιάζονται οι λόγοι για τους οποίους οι μη κερδοσκοπικοί οργανισμοί θα πρέπει να εξετάσουν το ενδεχόμενο επένδυσης σε μια εφαρμογή για κινητά, ακόμη και όταν οι πόροι είναι περιορισμένοι.

Οι μη κερδοσκοπικοί οργανισμοί διαδραματίζουν ζωτικό ρόλο στην αντιμετώπιση κοινωνικών ζητημάτων, βασιζόμενοι σε μεγάλο βαθμό στην ικανότητά τους να συνδέονται και να εμπλέκουν τους υποστηρικτές και τους δικαιούχους τους. Στον σημερινό κόσμο, όπου οι κινητές συσκευές έχουν γίνει αναπόσπαστο μέρος της καθημερινής ζωής, η αξιοποίηση των δυνατοτήτων των εφαρμογών κινητής τηλεφωνίας μπορεί να αποτελέσει στρατηγική κίνηση. Κάποια οφέλη και ευκαιρίες που προσφέρουν οι κινητές εφαρμογές στους μη κερδοσκοπικούς οργανισμούς είναι:

- **Επέκταση της εμβέλειας και της προσβασιμότητας:** Οι εφαρμογές για κινητά προσφέρουν στους μη κερδοσκοπικούς οργανισμούς μια ανεκτίμητη ευκαιρία να επεκτείνουν την εμβέλειά τους και να ενισχύσουν την προσβασιμότητα. Στη σημερινή

ψηφιακή εποχή, πολλά άτομα βασίζονται κυρίως σε κινητές συσκευές για πληροφορίες και υπηρεσίες. Με την ανάπτυξη μιας εφαρμογής για κινητά, οι μη κερδοσκοπικοί οργανισμοί μπορούν να αξιοποιήσουν αυτή τη βάση χρηστών και να συνδεθούν με όσους δεν έχουν εύκολη πρόσβαση σε υπολογιστή, αλλά διαθέτουν smartphone ή tablet. Επιπλέον, οι εφαρμογές για κινητά μπορούν να σχεδιαστούν με χαρακτηριστικά προσβασιμότητας, όπως φωνητικές εντολές και προγράμματα ανάγνωσης οθόνης, καθιστώντας τις πιο περιεκτικές και εξασφαλίζοντας ότι τα άτομα με αναπηρία μπορούν να συμμετέχουν στο περιεχόμενο και τις υπηρεσίες του οργανισμού επί ίσοις όροις με τους άλλους.

- **Βελτιωμένη εμπειρία χρήστη:** Όταν πρόκειται για την εμπειρία χρήστη, οι εφαρμογές για κινητά υπερτερούν έναντι των ιστότοπων για κινητά. Αυτές οι εφαρμογές είναι προσαρμοσμένες για συγκεκριμένες πλατφόρμες κινητών, όπως το iOS και το Android, επιτρέποντας βελτιστοποιημένες εμπειρίες σε κάθε πλατφόρμα. Προσφέρουν ταχύτερους χρόνους φόρτωσης, ομαλότερη πλοήγηση και πιο διαισθητική διεπαφή, οδηγώντας σε μεγαλύτερη ικανοποίηση των χρηστών. Οι μη κερδοσκοπικοί οργανισμοί μπορούν να επωφεληθούν σε μεγάλο βαθμό από αυτή τη βελτιωμένη εμπειρία χρήστη, καθώς προωθεί τη δέσμευση και ενθαρρύνει τους χρήστες να αλληλεπιδρούν συχνότερα και αποτελεσματικότερα με τις πρωτοβουλίες και τους πόρους του οργανισμού.
- **Πρόσβαση χωρίς σύνδεση:** Η πρόσβαση εκτός σύνδεσης είναι ένα κρίσιμο πλεονέκτημα που προσφέρουν οι εφαρμογές για κινητά στους μη κερδοσκοπικούς οργανισμούς. Αποθηκεύοντας βασικά δεδομένα τοπικά, όπως άρθρα, πόρους ή φόρμες, οι εφαρμογές για κινητά δίνουν τη δυνατότητα στους χρήστες να έχουν πρόσβαση σε κρίσιμες πληροφορίες ακόμη και όταν βρίσκονται σε περιοχές με κακή ή καθόλου συνδεσιμότητα στο διαδίκτυο. Αυτό το χαρακτηριστικό είναι ιδιαίτερα επωφελές για μη κερδοσκοπικούς οργανισμούς που εξυπηρετούν απομακρυσμένες ή υποεξυπηρετούμενες κοινότητες και οργανισμούς, διασφαλίζοντας ότι οι δικαιούχοι τους μπορούν να έχουν πρόσβαση σε ζωτικούς πόρους και υποστήριξη όταν το χρειάζονται περισσότερο, ανεξάρτητα από την τοποθεσία ή την πρόσβασή τους στο διαδίκτυο.
- **Αξιοποίηση των ειδοποιήσεων τύπου ώθησης (Push):** Οι ειδοποιήσεις τύπου ώθησης Push αποτελούν ένα ισχυρό εργαλείο στα χέρια των μη κερδοσκοπικών οργανισμών για να κρατούν τους υποστηρικτές τους δεσμευμένους και ενημερωμένους. Αυτές οι ειδοποιήσεις μπορούν να χρησιμοποιηθούν για την παροχή ενημερώσεων σε πραγματικό χρόνο σχετικά με διαφορετικές εκδηλώσεις, επείγοντα περιστατικά ή διάφορα συμβάντα απευθείας στις οθόνες των συσκευών των χρηστών. Παρακάμπτοντας τα γεμάτα εισερχόμενα emails και φτάνοντας στους χρήστες απευθείας στις κινητές συσκευές τους, οι ειδοποιήσεις push εξασφαλίζουν ότι οι σημαντικές πληροφορίες τραβούν την προσοχή των χρηστών αμέσως, προωθώντας μια πιο αφοσιωμένη και ενημερωμένη βάση υποστηρικτών.
- **Monetization:** Ενώ οι μη κερδοσκοπικοί οργανισμοί καθοδηγούνται από την αποστολή τους και όχι από τα κέρδη, οι εφαρμογές για κινητά μπορούν να προσφέρουν πρόσθετες πηγές εσόδων για τη στήριξη του ζωτικού τους έργου. Οι επιλογές δωρεών εντός της εφαρμογής, οι πωλήσεις εμπορευμάτων ή οι συνεργασίες με εταιρικούς χορηγούς μπορούν να διευκολυνθούν μέσω μιας εφαρμογής για κινητά. Αυτές οι ευκαιρίες εσόδων μπορούν να βοηθήσουν τους μη κερδοσκοπικούς οργανισμούς να διατηρήσουν τις δραστηριότητές τους και να προωθήσουν την

αποστολή τους, εξασφαλίζοντας έναν πιο ισχυρό και διαρκή αντίκτυπο στις κοινότητές τους.

- **Βελτιωμένη δέσμευση:** Οι εφαρμογές για κινητά έχουν τη δυνατότητα να βελτιώσουν σημαντικά τη δέσμευση των χρηστών μέσω διαφόρων χαρακτηριστικών και λειτουργιών. Τα στοιχεία παιχνιδοποίησης, όπως οι προκλήσεις, οι πίνακες κατάταξης και τα εμβλήματα, μπορούν να παρακινήσουν τους χρήστες να συμμετέχουν ενεργά στις δραστηριότητες του οργανισμού, καλλιεργώντας την αίσθηση της επίτευξης και της κοινότητας. Επιπλέον, οι επιλογές κοινωνικής κοινοποίησης που ενσωματώνονται στις εφαρμογές για κινητά δίνουν τη δυνατότητα στους υποστηρικτές να διαδώσουν τις πρωτοβουλίες του μη κερδοσκοπικού οργανισμού, επεκτείνοντας οργανικά την εμβέλειά του και ενισχύοντας τον αντίκτυπό του.
- **Πρόσβαση στις λειτουργίες της συσκευής:** Οι εφαρμογές για κινητά μπορούν να αξιοποιήσουν τα μοναδικά χαρακτηριστικά των κινητών συσκευών, οδηγώντας σε καινοτόμες λύσεις που μπορούν να ωφελήσουν μη κερδοσκοπικούς οργανισμούς και τους δικαιούχους τους. Για παράδειγμα, η ενσωμάτωση κάμερας μπορεί να χρησιμοποιηθεί για τη σάρωση εγγράφων, τη λήψη φωτογραφιών που σχετίζονται με τον σκοπό της οργάνωσης ή ακόμη και τη διευκόλυνση εικονικών περιηγήσεων σε έργα ή πρωτοβουλίες, παρέχοντας μια πιο πλούσια και διαδραστική εμπειρία για τους χρήστες.
- **Ορατότητα μάρκας:** Μια εφαρμογή για κινητά που εμφανίζει σε περίοπτη θέση το εμπορικό σήμα του μη κερδοσκοπικού οργανισμού χρησιμεύει ως συνεχής υπενθύμιση της αποστολής και της ταυτότητας του οργανισμού. Ενισχύει την παρουσία του οργανισμού στις συσκευές των χρηστών, αυξάνοντας την πιθανότητα να ασχοληθούν με το περιεχόμενο και τις πρωτοβουλίες του οργανισμού. Η εφαρμογή συμβολίζει την τεχνολογική συνάφεια και υπογραμμίζει τη δέσμευση του οργανισμού να παρέχει στους υποστηρικτές του ένα σύγχρονο και βολικό μέσο εμπλοκής, ενισχύοντας περαιτέρω το εμπορικό σήμα του οργανισμού και τον αντίκτυπό του.
- **Ανταγωνιστικό πλεονέκτημα:** Σε ένα τοπίο όπου πολλοί μη κερδοσκοπικοί οργανισμοί αγκαλιάζουν τις εφαρμογές για κινητά, η μη ύπαρξη μιας τέτοιας εφαρμογής μπορεί να θέσει έναν οργανισμό σε ανταγωνιστικό μειονέκτημα. Οι χρήστες συχνά προτιμούν τους οργανισμούς που προσφέρουν εφαρμογές για κινητά για την ευκολία και τη σύγχρονη ελκυστικότητά τους. Παρέχοντας μια εφαρμογή για κινητά, οι μη κερδοσκοπικοί οργανισμοί μπορούν να ξεχωρίσουν και να ανταποκριθούν στις προτιμήσεις ενός τεχνολογικά ενημερωμένου κοινού, προσελκύοντας ενδεχομένως περισσότερους υποστηρικτές και παραμένοντας ανταγωνιστικοί στον τομέα τους.
- **Λήψη αποφάσεων βάσει δεδομένων:** Οι εφαρμογές για κινητά παρέχουν στους μη κερδοσκοπικούς οργανισμούς πληθώρα πολύτιμων δεδομένων και πληροφοριών. Τα εργαλεία ανάλυσης που είναι ενσωματωμένα σε αυτές τις εφαρμογές προσφέρουν λεπτομερείς πληροφορίες σχετικά με τη συμπεριφορά των χρηστών, δείχνοντας ποιες λειτουργίες είναι δημοφιλείς, πού οι χρήστες πέφτουν και πόσο συχνά συμμετέχουν. Τα δεδομένα αυτά δίνουν τη δυνατότητα στους μη κερδοσκοπικούς οργανισμούς να λαμβάνουν τεκμηριωμένες αποφάσεις, να βελτιώνουν τις στρατηγικές τους και να βελτιώνουν συνεχώς τις εμπειρίες των χρηστών. Αξιοποιώντας τις πληροφορίες που βασίζονται σε δεδομένα, οι οργανισμοί μπορούν να κατανέμουν τους πόρους πιο αποτελεσματικά και να προσαρμόζουν τις πρωτοβουλίες τους ώστε να εξυπηρετούν καλύτερα τις κοινότητές τους, μεγιστοποιώντας τελικά τον αντίκτυπό τους και επιτυγχάνοντας πιο αποτελεσματικά την αποστολή τους.

Συμπερασματικά, η υιοθέτηση εφαρμογών για κινητά από μη κερδοσκοπικούς οργανισμούς, ακόμη και από εκείνους με περιορισμένους πόρους, μπορεί να ενισχύσει σημαντικά την εμβέλεια, τη δέσμευση και τον αντίκτυπό τους. Τα πλεονεκτήματα μιας εφαρμογής για κινητά, συμπεριλαμβανομένης της προσβασιμότητας, της εμπειρίας του χρήστη και της δέσμευσης, υπερτερούν του αρχικού κόστους ανάπτυξης και συντήρησης. Καθώς το ψηφιακό τοπίο συνεχίζει να εξελίσσεται, οι μη κερδοσκοπικοί οργανισμοί θα πρέπει να εξετάσουν το ενδεχόμενο να επενδύσουν σε εφαρμογές για κινητά για να παραμείνουν αποτελεσματικοί και συνδεδεμένοι με τις κοινότητές τους. Έτσι, αγκαλιάζοντας την τεχνολογία των κινητών τηλεφώνων, οι μη κερδοσκοπικοί οργανισμοί μπορούν να γεφυρώσουν το χάσμα μεταξύ των αποστολών τους και της ψηφιακά ενημερωμένης κοινωνίας που στοχεύουν να εξυπηρετήσουν.

2 Επιλογή εργαλείων

Η ομάδα εργασίας βρέθηκε σε ένα δίλημμα κατά την μετάβαση μιας εφαρμογής React JS σε κινητές συσκευές. Αυτό το δίλημμα περιλάμβανε δύο κύριες επιλογές: το Ionic Framework και το React Native. Και τα δύο είναι πλατφόρμες για την ανάπτυξη κινητών εφαρμογών, αλλά έχουν διαφορετικές προσεγγίσεις και χαρακτηριστικά.

Η ομάδα αναγνώρισε ότι και τα δύο πλαίσια είχαν τα πλεονεκτήματά τους, αλλά τελικά επέλεξε το Ionic Framework. Ο λόγος που προτιμήθηκε το Ionic ήταν λόγω της ευελιξίας και της ταχύτητάς του στην ανάπτυξη εφαρμογών που μπορούν να λειτουργούν σε διάφορες πλατφόρμες, όπως iOS, Android, και web. Αυτό κατέστησε δυνατή την δημιουργία μιας εφαρμογής που ήταν εξίσου λειτουργική σε διάφορες κινητές συσκευές, χωρίς την ανάγκη για πολλαπλή ανάπτυξη για κάθε πλατφόρμα ξεχωριστά.

Η απόφαση αυτή κρίθηκε ως επιτυχημένη, καθώς επέτρεψε στην ομάδα να μετατρέψει την αρχική React JS εφαρμογή σε μια πολύ ευέλικτη και λειτουργική εφαρμογή που προσφέρεται σε διάφορες πλατφόρμες, εξασφαλίζοντας έτσι την καλύτερη δυνατή εμπειρία για τους χρήστες τους.

2.1 Ionic Framework

Το Ionic Framework είναι ένα ευρέως διαδεδομένο front-end framework ανοικτού κώδικα για την ανάπτυξη εφαρμογών κινητών και διαδικτυακών εφαρμογών πολλαπλών πλατφορμών. Δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν υψηλής ποιότητας, διαδραστικές και αποδοτικές εφαρμογές χρησιμοποιώντας τυποποιημένες τεχνολογίες ιστού, όπως HTML, CSS και JavaScript. Το Ionic φημίζεται ιδιαίτερα για την ικανότητά του να διευκολύνει την ανάπτυξη εφαρμογών για κινητά που μπορούν να τρέξουν σε διάφορες πλατφόρμες με μια ενιαία βάση κώδικα.

2.1.1 Βασικά χαρακτηριστικά και στοιχεία:

- *UI Components:* Το Ionic προσφέρει μια ολοκληρωμένη βιβλιοθήκη προ-σχεδιασμένων στοιχείων UI που περιλαμβάνουν κουμπιά, φόρμες, μενού πλοήγησης, modals, καρτέλες, κάρτες και πολλά άλλα. Αυτά τα στοιχεία δεν είναι μόνο πλούσια σε χαρακτηριστικά, αλλά και εξαιρετικά προσαρμόσιμα, επιτρέποντας στους προγραμματιστές να τα προσαρμόσουν στις συγκεκριμένες απαιτήσεις της εφαρμογής τους.
- *Ενσωμάτωση Cordova/PhoneGap: ́*Ενα από τα χαρακτηριστικά που ξεχωρίζει το Ionic είναι η απρόσκοπτη ενσωμάτωσή του με το Apache Cordova (πρώην γνωστό ως PhoneGap). Το Cordova επιτρέπει στους προγραμματιστές να έχουν πρόσβαση σε δυνατότητες και API εγγενών συσκευών χρησιμοποιώντας JavaScript, γεφυρώνοντας το χάσμα μεταξύ της ανάπτυξης εφαρμογών web και εγγενών εφαρμογών. Αυτό καθιστά το Ionic μια εξαιρετική επιλογή για τη δημιουργία υβριδικών εφαρμογών για κινητά που μπορούν να αναπτυχθούν σε πολλαπλές πλατφόρμες, συμπεριλαμβανομένων των iOS, Android και άλλων.
- *Θεματοποίηση και στυλ:* Το Ionic παρέχει ένα ισχυρό σύστημα θεματοποίησης (theming) και μια σειρά από προ-δημιουργημένα θέματα που απλοποιούν τη διαδικασία δημιουργίας οπτικά ελκυστικών και συνεπών σχεδίων εφαρμογών. Οι

προγραμματιστές μπορούν επίσης να προσαρμόσουν αυτά τα θέματα ώστε να ευθυγραμμιστούν με την ταυτότητα της μάρκας τους ή τις προτιμήσεις σχεδιασμού τους.

- *Ενσωμάτωση JS Frameworks:* Το Ionic στηρίχθηκε αρχικά σε μεγάλο βαθμό στο AngularJS, αλλά έκτοτε έχει εξελιχθεί ώστε να υποστηρίζει το σύγχρονο Angular Framework. Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να επιλέγουν το JavaScript Framework που προτιμούν για την κατασκευή εφαρμογών Ionic, απευθυνόμενοι σε ένα ευρύτερο κοινό προγραμματιστών. Εκτός από το Angular, το Ionic προσφέρει επίσημες δεσμεύσεις και υποστήριξη για το React και το Vue.js, παρέχοντας στους προγραμματιστές εναλλακτικές επιλογές για τη δημιουργία εφαρμογών Ionic με τη χρήση αυτών των δημοφιλών βιβλιοθηκών JavaScript.
- *Διεπαφή γραμμής εντολών (CLI):* Το Ionic CLI είναι ένα ισχυρό σύνολο εργαλείων γραμμής εντολών που απλοποιούν διάφορες πτυχές της ανάπτυξης εφαρμογών, συμπεριλαμβανομένης της ρύθμισης έργου, της δημιουργίας κώδικα, των δοκιμών και της ανάπτυξης. Αυτό το CLI ενισχύει την παραγωγικότητα και βοηθά τους προγραμματιστές να διατηρούν τη συνοχή του έργου.
- *Προοδευτικές εφαρμογές ιστού - Progressive Web Apps (PWAs):* Το Ionic είναι καλά εξοπλισμένο για να διευκολύνει την ανάπτυξη εφαρμογών προοδευτικού ιστού (Progressive Web Apps - PWAs). Οι PWAs προσφέρουν εμπειρίες που μοιάζουν με native μέσα σε προγράμματα περιήγησης ιστού, με χαρακτηριστικά όπως η λειτουργία χωρίς σύνδεση, οι γρήγοροι χρόνοι φόρτωσης και η συμβατότητα πολλαπλών πλατφορμών.
- *Κοινότητα και Plugins:* Η κοινότητα Ionic είναι ενεργή και έχει δημιουργήσει πολυάριθμα plugins και επεκτάσεις. Αυτά τα πρόσθετα επεκτείνουν τις δυνατότητες του πλαισίου και επιτρέπουν στους προγραμματιστές να ενσωματώνουν υπηρεσίες, βιβλιοθήκες και λειτουργίες τρίτων κατασκευαστών στις εφαρμογές Ionic.

2.1.2 Πλεονεκτήματα χρήσης

2.1.2.1 Επαναχρησιμοποίηση κώδικα

Η έννοια της επαναχρησιμοποίησης του κώδικα στο Ionic Framework ξεπερνά την απλή ευκολία - μεταμορφώνει ριζικά τον τρόπο με τον οποίο οι προγραμματιστές προσεγγίζουν την ανάπτυξη εφαρμογών κινητών και διαδικτυακών εφαρμογών πολλαπλών πλατφορμών. Στο επίκεντρο αυτής της αρχής βρίσκεται η δυνατότητα να γράφουμε κώδικα μία φορά και να τον χρησιμοποιούμε σε πολλαπλές πλατφόρμες, όπως iOS, Android και προγράμματα περιήγησης ιστού. Αυτό επιτυγχάνεται μέσω της εξάρτησης του Ionic από τεχνολογίες ιστού, όπως η HTML, η CSS και η JavaScript, οι οποίες υποστηρίζονται καθολικά και είναι κατανοητές από τους προγραμματιστές.

Το Ionic επιτυγχάνει την επαναχρησιμοποίηση του κώδικα μέσω της αρχιτεκτονικής του που βασίζεται σε συστατικά. Οι προγραμματιστές δημιουργούν στοιχεία UI, όπως κουμπιά, φόρμες, μενού πλοήγησης και άλλα, χρησιμοποιώντας την πλούσια βιβλιοθήκη προ-σχεδιασμένων στοιχείων του Ionic. Αυτά τα στοιχεία μπορούν να ενσωματωθούν απρόσκοπτα στην κωδικοποιημένη βάση της εφαρμογής και η εμφάνιση και η συμπεριφορά τους μπορούν να προσαρμοστούν ανάλογα με τις ανάγκες.

Ένα από τα κύρια πλεονεκτήματα αυτής της προσέγγισης είναι η σημαντική μείωση του χρόνου και της προσπάθειας ανάπτυξης. Αντί να διατηρούν ξεχωριστές βάσεις κώδικα για κάθε πλατφόρμα, οι προγραμματιστές μπορούν να επικεντρώσουν τις προσπάθειές τους σε μια ενιαία βάση κώδικα, εξοικονομώντας τόσο χρόνο όσο και πόρους. Αυτό όχι μόνο επιταχύνει την αρχική φάση ανάπτυξης, αλλά απλοποιεί επίσης τη συνεχή συντήρηση και τις ενημερώσεις χαρακτηριστικών, καθώς οι αλλαγές μπορούν να γίνουν καθολικά, χωρίς την ανάγκη για τροποποιήσεις ειδικά για κάθε πλατφόρμα.

Επιπλέον, η υποστήριξη του Ionic για το Cordova/PhoneGap επιτρέπει στους προγραμματιστές να έχουν πρόσβαση σε εγγενείς λειτουργίες και API συσκευών χρησιμοποιώντας JavaScript. Αυτό σημαίνει ότι ενώ η βασική λογική και η διεπαφή χρήστη της εφαρμογής παραμένουν συνεπείς σε όλες τις πλατφόρμες, οι προγραμματιστές μπορούν να αξιοποιήσουν πλήρως τις δυνατότητες της συσκευής. Για παράδειγμα, μπορούν να χρησιμοποιούν απρόσκοπτα την κάμερα, το GPS, τους αισθητήρες και τις ειδοποιήσεις push της συσκευής.

Συνολικά, η επαναχρησιμοποίηση του κώδικα στο Ionic Framework αλλάζει τα δεδομένα τόσο για τους προγραμματιστές όσο και για τις επιχειρήσεις. Εξασφαλίζει μια συνεπή εμπειρία χρήστη σε όλες τις πλατφόρμες, μειώνει το κόστος ανάπτυξης και τον χρόνο διάθεσης στην αγορά και απλοποιεί τη μακροπρόθεσμη συντήρηση και τις ενημερώσεις, καθιστώντας το μια συναρπαστική επιλογή για την ανάπτυξη σύγχρονων εφαρμογών.

2.1.2.2 Οικονομία-Αποτελεσματικότητα

Προσφέρει μια πειστική λύση για τις επιχειρήσεις που στοχεύουν στη βελτιστοποίηση του προϋπολογισμού τους, παρέχοντας παράλληλα εφαρμογές υψηλής ποιότητας σε πολλαπλές πλατφόρμες.

Ο πυρήνας της σχέσης κόστους-αποτελεσματικότητας του Ionic έγκειται στην ικανότητά του να αξιοποιεί μια ενιαία βάση κώδικα για την ανάπτυξη εφαρμογών που εκτελούνται απρόσκοπτα σε iOS, Android και διαδίκτυο. Αυτή η προσέγγιση μειώνει δραματικά το κόστος ανάπτυξης και συντήρησης με διάφορους τρόπους:

- *Χρόνος ανάπτυξης και πόροι:* Με το Ionic, οι προγραμματιστές δεν χρειάζεται να δημιουργούν και να συντηρούν ξεχωριστές βάσεις κώδικα για κάθε πλατφόρμα. Αντ' αυτού, μπορούν να κατευθύνουν τις προσπάθειές τους σε μια ενιαία βάση κώδικα χρησιμοποιώντας τεχνολογίες ιστού όπως HTML, CSS και JavaScript. Αυτή η ενοποιημένη διαδικασία ανάπτυξης είναι σημαντικά ταχύτερη και απαιτεί λιγότερους πόρους από ό,τι η μεμονωμένη κατασκευή εγγενών εφαρμογών.
- *Ταλέντο ανάπτυξης:* Η διατήρηση μιας ενιαίας βάσης κώδικα μειώνει τη ζήτηση για εξειδικευμένη τεχνογνωσία ανάπτυξης για κάθε πλατφόρμα. Οι επιχειρήσεις μπορούν να βασίζονται σε μια ευρύτερη δεξαμενή προγραμματιστών ιστού που είναι ήδη εξειδικευμένοι στις τεχνολογίες ιστού, καθιστώντας την πρόσληψη πιο προσιτή και οικονομικά αποδοτική.
- *Διασφάλιση ποιότητας:* Οι προσπάθειες διασφάλισης ποιότητας εξορθολογίζονται με την επαναχρησιμοποίηση του κώδικα του Ionic. Οι δοκιμές μπορούν να επικεντρωθούν σε μία βάση κώδικα, εξασφαλίζοντας συνεπή λειτουργικότητα και συμπεριφορά σε όλες τις πλατφόρμες. Αυτό μειώνει το χρόνο και το κόστος που σχετίζονται με εκτεταμένες δοκιμές σε πολλαπλές εγγενείς πλατφόρμες.

- *Ενημερώσεις και συντήρηση:* Η μακροπρόθεσμη εξοικονόμηση κόστους είναι ιδιαίτερα αξιοσημείωτη όταν πρόκειται για ενημερώσεις και συντήρηση εφαρμογών. Το Ionic επιτρέπει καθολικές ενημερώσεις, πράγμα που σημαίνει ότι οι αλλαγές, οι βελτιώσεις και οι διορθώσεις σφαλμάτων μπορούν να εφαρμοστούν ομοιόμορφα σε όλες τις πλατφόρμες. Αυτό ελαχιστοποιεί την ανάγκη για προσπάθειες ανάπτυξης συγκεκριμένων πλατφορμών, μειώνοντας το συνεχές κόστος συντήρησης.
- *Κλιμάκωση και επέκταση:* Καθώς οι επιχειρήσεις αναπτύσσονται και επεκτείνουν τις προσφορές των εφαρμογών τους ή εισέρχονται σε νέες αγορές, η αποδοτικότητα κόστους του Ionic γίνεται ακόμη πιο έντονη. Η δυνατότητα κλιμάκωσης με την ανάπτυξη νέων χαρακτηριστικών ή την επέκταση σε νέες πλατφόρμες χωρίς σημαντική αύξηση του κόστους ανάπτυξης αποτελεί σημαντικό πλεονέκτημα.

Συνολικά, η διαχείριση μιας ενιαίας βάσης κώδικα με το Ionic είναι μια εξαιρετικά οικονομική προσέγγιση σε σύγκριση με τη διατήρηση ξεχωριστών βάσεων κώδικα για iOS και Android. Αυτό το οικονομικό μοντέλο δεν ωφελεί μόνο την τελική γραμμή, αλλά δίνει επίσης τη δυνατότητα στις επιχειρήσεις να κατανέμουν στρατηγικά τους πόρους, να ενισχύουν την ταχύτητα ανάπτυξης και να επιτυγχάνουν ταχύτερη απόδοση της επένδυσης (ROI) για τα έργα εφαρμογών τους.

2.1.2.3 Γρήγορη ανάπτυξη

Η ταχεία ανάπτυξη είναι μια κρίσιμη πτυχή του Ionic Framework, καθιστώντας το μια προτιμώμενη επιλογή για τις επιχειρήσεις και τους προγραμματιστές που επιθυμούν να φέρουν τις εφαρμογές τους στην αγορά γρήγορα, διατηρώντας παράλληλα πρότυπα υψηλής ποιότητας.

Στο επίκεντρο των δυνατοτήτων ταχείας ανάπτυξης του Ionic βρίσκεται η εκτεταμένη βιβλιοθήκη προ-σχεδιασμένων στοιχείων και προτύπων UI. Αυτά τα στοιχεία περιλαμβάνουν ένα ευρύ φάσμα στοιχείων διεπαφής χρήστη, όπως κουμπιά, φόρμες, μενού πλοήγησης, modals, καρτέλες, κάρτες και πολλά άλλα. Αυτά τα έτοιμα δομικά στοιχεία επιτρέπουν στους προγραμματιστές να επιταχύνουν τη δημιουργία οπτικά ελκυστικών και διαδραστικών εφαρμογών.

Τα στοιχεία UI του Ionic δεν εξοικονομούν απλώς χρόνο, αλλά προωθούν επίσης τη συνοχή στο σχεδιασμό εφαρμογών. Οι προγραμματιστές μπορούν να αξιοποιήσουν αυτά τα στοιχεία για να διασφαλίσουν ότι οι εφαρμογές τους τηρούν τις βέλτιστες πρακτικές όσον αφορά την εμπειρία χρήστη (UX) και το σχεδιασμό διεπαφής χρήστη (UI). Αυτή η συνέπεια όχι μόνο βελτιώνει την εμπειρία του χρήστη, αλλά μειώνει επίσης την ανάγκη για εκτεταμένες προσπάθειες σχεδιασμού και δημιουργίας πρωτοτύπων, επιταχύνοντας περαιτέρω την ανάπτυξη.

Επιπλέον, τα στοιχεία του UI του Ionic είναι εξαιρετικά προσαρμόσιμα. Οι προγραμματιστές μπορούν να προσαρμόσουν την εμφάνιση και τη συμπεριφορά αυτών των στοιχείων ώστε να ανταποκρίνονται ακριβώς στις μοναδικές απαιτήσεις σχεδιασμού και λειτουργικότητας της εφαρμογής τους. Αυτό το επίπεδο ευελιξίας σημαίνει ότι, ενώ το Ionic προσφέρει μια γρήγορη εκκίνηση, δεν συμβιβάζεται με τη δυνατότητα δημιουργίας μιας μοναδικής και προσαρμοσμένης στο εμπορικό σήμα εφαρμογής.

Επιπλέον, η υποστήριξη του Ionic για δημοφιλή JavaScript Frameworks όπως το Angular, το React και το Vue.js συμβάλλει στην ταχεία ανάπτυξη. Οι προγραμματιστές μπορούν να

επιλέξουν το framework με το οποίο αισθάνονται πιο άνετα, απλοποιώντας την ανάπτυξη με το να εργάζονται μέσα στο οικοσύστημα που προτιμούν.

Ο συνδυασμός προ-σχεδιασμένων στοιχείων UI, προτύπων και η ευελιξία προσαρμογής τους, μαζί με την υποστήριξη διαφορετικών πλαισίων JavaScript, δίνει τη δυνατότητα στους προγραμματιστές να επιταχύνουν σημαντικά τη διαδικασία ανάπτυξης εφαρμογών. Αυτή η αποδοτικότητα είναι ανεκτίμητη στις ανταγωνιστικές αγορές, επιτρέποντας στις επιχειρήσεις να διαθέτουν τα προϊόντα τους ταχύτερα στην αγορά και να ανταποκρίνονται γρήγορα στα σχόλια των χρηστών και στις μεταβαλλόμενες απαιτήσεις της αγοράς. Στην ουσία, η έμφαση του Ionic στην ταχεία ανάπτυξη ευθυγραμμίζεται με την ανάγκη για ευελιξία και ταχύτητα στο σύγχρονο τοπίο ανάπτυξης λογισμικού.

2.1.2.4 Πρόσβαση σε native χαρακτηριστικά με το Ionic

Ένα από τα χαρακτηριστικά που ξεχωρίζουν στο Ionic Framework είναι η απρόσκοπτη ενσωμάτωσή του με το Apache Cordova, ένα ισχυρό εργαλείο που γεφυρώνει το χάσμα μεταξύ των τεχνολογιών ιστού και των εγγενών δυνατοτήτων (native features) των κινητών συσκευών. Αυτή η ενσωμάτωση δίνει τη δυνατότητα στους προγραμματιστές να αξιοποιούν ένα ευρύ φάσμα εγγενών χαρακτηριστικών και API συσκευών χρησιμοποιώντας τη γνώριμη γλώσσα JavaScript, βελτιώνοντας έτσι την εμπειρία του χρήστη με πλήθος λειτουργιών που αφορούν συγκεκριμένες συσκευές.

Η υποστήριξη του Ionic για το Cordova επιτρέπει στους προγραμματιστές να ενσωματώνουν αβίαστα στις εφαρμογές τους εγγενή χαρακτηριστικά, όπως πρόσβαση στην κάμερα, υπηρεσίες εντοπισμού θέσης GPS, αισθητήρες (π.χ. επιταχυνσιόμετρο, γυροσκόπιο) και ειδοποιήσεις push. Αυτή η πρόσβαση σε εγγενείς λειτουργίες ξεπερνά την απλή ευκολία-επιτρέπει στους προγραμματιστές να παρέχουν στους χρήστες μια πραγματικά καθηλωτική και εγγενή εμπειρία.

Για παράδειγμα, αξιοποιώντας το Ionic και το Cordova, οι προγραμματιστές μπορούν να ενσωματώσουν την κάμερα της συσκευής για να ενεργοποιήσουν λειτουργίες όπως η λήψη φωτογραφιών, η σάρωση barcode ή η σάρωση εγγράφων μέσα στην εφαρμογή. Μπορούν να αξιοποιήσουν υπηρεσίες GPS και γεωγραφικού εντοπισμού για να δημιουργήσουν εφαρμογές με επίγνωση της τοποθεσίας για υπηρεσίες όπως χάρτες, πλοήγηση και παροχή περιεχομένου με βάση την τοποθεσία. Η πρόσβαση σε αισθητήρες ανοίγει ευκαιρίες για την ανάπτυξη εφαρμογών γυμναστικής και ευεξίας, εμπειριών παιχνιδιών και εφαρμογών που ανταποκρίνονται στον προσανατολισμό και την κίνηση της συσκευής.

Οι ειδοποιήσεις push, μια κρίσιμη πτυχή της εμπλοκής των χρηστών, είναι προσβάσιμες μέσω του Ionic και του Cordova. Οι προγραμματιστές μπορούν να εφαρμόζουν ειδοποιήσεις push για να ενημερώνουν και να δεσμεύουν τους χρήστες, οδηγώντας σε βελτιωμένη διατήρηση και αλληλεπίδραση των χρηστών με την εφαρμογή.

Επιπλέον, η προσέγγιση του Ionic που δεν επηρεάζει τις πλατφόρμες διασφαλίζει ότι η ενσωμάτωση αυτών των εγγενών χαρακτηριστικών παραμένει συνεπής σε διάφορες πλατφόρμες, όπως το iOS, το Android και ο ιστός. Αυτό όχι μόνο απλοποιεί την ανάπτυξη, αλλά εγγυάται επίσης μια ομοιόμορφη εμπειρία για τους χρήστες, ανεξάρτητα από τη συσκευή ή το λειτουργικό σύστημα που χρησιμοποιούν.

Εν κατακλείδι, η ικανότητα του Ionic να έχει απρόσκοπτη πρόσβαση σε εγγενή χαρακτηριστικά συσκευών μέσω του Cordova δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν εφαρμογές που προσφέρουν μια πραγματικά εγγενή εμπειρία. Αυτή η πλούσια ενσωμάτωση

ενισχύει τη δέσμευση των χρηστών, διευρύνει το φάσμα των πιθανών εφαρμογών και προσθέτει βάθος στη λειτουργικότητα των εφαρμογών που υποστηρίζονται από το Ionic, καθιστώντας το μια ελκυστική επιλογή για τους προγραμματιστές που στοχεύουν στην παροχή πλούσιων σε χαρακτηριστικά και ιδιαίτερα διαδραστικών εμπειριών για τους χρήστες.

2.1.2.5 Συμβατότητα πολλαπλών πλατφορμών

Η συμβατότητα πολλαπλών πλατφορμών αποτελεί χαρακτηριστικό γνώρισμα του Ionic Framework και αντιπροσωπεύει ένα σημαντικό πλεονέκτημα για τις επιχειρήσεις και τους προγραμματιστές που επιθυμούν να προσεγγίσουν ένα ευρύ και ποικίλο κοινό χωρίς την πολυπλοκότητα της διατήρησης ξεχωριστών βάσεων κώδικα για κάθε πλατφόρμα.

Το Ionic επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που εκτελούνται απρόσκοπτα σε πολλαπλές πλατφόρμες, όπως iOS, Android και προγράμματα περιήγησης στο διαδίκτυο, χρησιμοποιώντας μια ενιαία βάση κώδικα. Αυτή η προσέγγιση όχι μόνο απλοποιεί τη διαδικασία ανάπτυξης, αλλά διασφαλίζει επίσης ότι οι χρήστες σε διαφορετικές συσκευές μπορούν να έχουν πρόσβαση και να επωφελούνται από τα ίδια χαρακτηριστικά και τη λειτουργικότητα της εφαρμογής.

Η ομορφιά της συμβατότητας πολλαπλών πλατφορμών με το Ionic έγκειται στην ικανότητά του να μειώνει την ανάγκη για προσπάθειες ανάπτυξης συγκεκριμένων πλατφορμών. Παραδοσιακά, η δημιουργία εγγενών εφαρμογών για iOS και Android θα περιλάμβανε διαφορετικές ομάδες ανάπτυξης, τεχνολογίες και χρονοδιαγράμματα. Αυτή η προσέγγιση όχι μόνο κλιμακώνει το κόστος ανάπτυξης, αλλά εισάγει επίσης προκλήσεις στον συγχρονισμό των ενημερώσεων και τη διασφάλιση συνεπών εμπειριών χρήστη.

Η ανεξαρτησία της Ionic από την πλατφόρμα εξαλείφει αυτά τα εμπόδια. Οι προγραμματιστές μπορούν να επικεντρωθούν σε μια ενιαία βάση κώδικα γραμμένη σε οικείες τεχνολογίες ιστού, όπως HTML, CSS και JavaScript, ενώ το Ionic φροντίζει για τις διαφορές των υποκείμενων πλατφορμών. Αυτό επιταχύνει σημαντικά τους κύκλους ανάπτυξης, μειώνει το κόστος και απλοποιεί τις προσπάθειες συντήρησης.

Επιπλέον, η δέσμευση του Ionic για τη διατήρηση μιας συνεπούς εμπειρίας χρήστη σε όλες τις πλατφόρμες διασφαλίζει ότι οι χρήστες λαμβάνουν μια συνεκτική και οικεία αλληλεπίδραση με την εφαρμογή, ανεξάρτητα από τη συσκευή που χρησιμοποιούν. Αυτή η συνοχή στο σχεδιασμό και τη λειτουργικότητα ενισχύει την ικανοποίηση των χρηστών και ελαχιστοποιεί την πιθανή σύγχυση των χρηστών κατά την εναλλαγή μεταξύ πλατφορμών.

Το Ionic προσαρμόζεται επίσης σε διάφορα μεγέθη οθόνης και αναλύσεις, καθιστώντας το κατάλληλο για διαφορετικές συσκευές, από smartphones και tablets μέχρι προγράμματα περιήγησης σε επιτραπέζιους υπολογιστές. Οι αρχές του Responsive Design διασφαλίζουν ότι η διάταξη της εφαρμογής και τα στοιχεία διεπαφής χρήστη προσαρμόζονται με χάρη ώστε να παρέχουν τη βέλτιστη εμπειρία σε κάθε τύπο συσκευής.

Εν κατακλείδι, η συμβατότητα του Ionic με διαφορετικές πλατφόρμες αποτελεί στρατηγικό πλεονέκτημα για τις επιχειρήσεις που στοχεύουν στη μεγιστοποίηση της εμβέλειάς τους και στην ελαχιστοποίηση του κόστους ανάπτυξης. Επιτρέποντας τη δημιουργία εφαρμογών που τρέχουν απρόσκοπτα σε πλατφόρμες iOS, Android και web από μία ενιαία βάση κώδικα, το Ionic προσφέρει μια συναρπαστική λύση για τη σύγχρονη ανάπτυξη εφαρμογών, προωθώντας μια ευρύτερη και πιο ποικιλόμορφη βάση χρηστών, διατηρώντας παράλληλα την αποτελεσματικότητα και τη συνοχή.

2.1.2.6 Τεχνολογίες ιστού

Η αξιοποίηση των τεχνολογιών ιστού από το Ionic αποτελεί ακρογωνιαίο λίθο της ελκυστικότητάς του, προσφέροντας μια σειρά πλεονεκτημάτων που το καθιστούν ελκυστική επιλογή τόσο για τους προγραμματιστές όσο και για τις επιχειρήσεις.

- *Οικείες τεχνολογίες ιστού:* Το Ionic αγκαλιάζει τεχνολογίες ιστού όπως η HTML, η CSS και η JavaScript ως θεμέλιο για την κατασκευή εφαρμογών. Αυτή η επιλογή είναι σημαντική, επειδή αξιοποιεί τις δεξιότητες και τις γνώσεις μιας τεράστιας δεξαμενής προγραμματιστών που είναι ήδη εξοικειωμένοι με την ανάπτυξη ιστού. Ως αποτέλεσμα, οι προγραμματιστές μπορούν να μεταβούν απρόσκοπτα στο Ionic χωρίς απότομη καμπύλη εκμάθησης, μειώνοντας το χρόνο που απαιτείται για να γίνουν ικανοί στο πλαίσιο.
- *Ευρεία κοινότητα προγραμματιστών:* Η υιοθέτηση τεχνολογιών ιστού εξασφαλίζει ότι το Ionic διαθέτει μια ευρεία και ενεργή κοινότητα προγραμματιστών. Αυτό το ζωντανό οικοσύστημα παρέχει πολλά πλεονεκτήματα, όπως εκτεταμένη τεκμηρίωση, σεμινάρια, φόρουμ και plugins και επεκτάσεις τρίτων κατασκευαστών. Οι προγραμματιστές μπορούν να αξιοποιήσουν αυτή την κοινότητα για υποστήριξη, να μοιραστούν γνώσεις και να βρουν λύσεις σε κοινές προκλήσεις, επιταχύνοντας την ανάπτυξη και την αντιμετώπιση προβλημάτων.
- *Επαναχρησιμοποίηση κώδικα:* Η κατασκευή εφαρμογών με τεχνολογίες ιστού και το Ionic επιτρέπει την επαναχρησιμοποίηση του κώδικα όχι μόνο σε διάφορες πλατφόρμες αλλά και σε διάφορα έργα που σχετίζονται με τον ιστό. Οι προγραμματιστές μπορούν να αξιοποιήσουν τις υπάρχουσες δεξιότητες ανάπτυξης ιστού και τις βιβλιοθήκες κώδικα, μειώνοντας περαιτέρω τον χρόνο και την προσπάθεια ανάπτυξης.
- *Ολοκλήρωση του οικοσυστήματος:* Το Ionic ενσωματώνεται απρόσκοπτα με δημοφιλή εργαλεία και βιβλιοθήκες ανάπτυξης ιστού, επιτρέποντας στους προγραμματιστές να χρησιμοποιούν τα εργαλεία που προτιμούν για εργασίες όπως η επεξεργασία κώδικα, ο έλεγχος έκδοσης και οι δοκιμές. Αυτή η ενσωμάτωση προάγει ένα άνετο και αποτελεσματικό περιβάλλον ανάπτυξης για όσους είναι ήδη εξοικειωμένοι με τις ροές εργασίας ανάπτυξης ιστού.
- *Παραδείγματα ανάπτυξης ιστού:* Το Ionic ενθαρρύνει τη χρήση παραδειγμάτων ανάπτυξης ιστού, όπως ο ανταποκρινόμενος σχεδιασμός και η αρχιτεκτονική βασισμένη σε στοιχεία. Αυτό σημαίνει ότι οι προγραμματιστές μπορούν να εφαρμόζουν βέλτιστες πρακτικές από την ανάπτυξη ιστού για τη δημιουργία ανταποκρινόμενων και κλιμακούμενων διεπαφών χρήστη, διασφαλίζοντας ότι οι εφαρμογές φαίνονται και αποδίδουν καλά σε μια ποικιλία συσκευών και μεγεθών οθόνης.
- *Ευκαιρίες διασταύρωσης δεξιοτήτων:* Η έμφαση που δίνει το Ionic στις τεχνολογίες ιστού προσφέρει στους προγραμματιστές την ευκαιρία να διαφοροποιήσουν τις δεξιότητές τους. Οι προγραμματιστές ιστού μπορούν να επεκτείνουν την τεχνογνωσία τους στην ανάπτυξη εφαρμογών για κινητά, ανοίγοντας νέες ευκαιρίες καριέρας και επιτρέποντάς τους να εργαστούν σε ένα ευρύτερο φάσμα έργων.

Εν κατακλείδι, ο εναγκαλισμός του Ionic με τις τεχνολογίες ιστού όχι μόνο μειώνει την καμπύλη εκμάθησης για τους προγραμματιστές, αλλά και αξιοποιεί μια τεράστια κοινότητα επαγγελματιών ανάπτυξης ιστού. Ενισχύει την επαναχρησιμοποίηση του κώδικα,

ενσωματώνεται με οικεία εργαλεία ανάπτυξης και επιτρέπει στους προγραμματιστές να εφαρμόζουν τις βέλτιστες πρακτικές ανάπτυξης ιστού για τη δημιουργία εφαρμογών υψηλής ποιότητας σε διαφορετικές πλατφόρμες. Αυτό καθιστά το Ionic μια προσιτή και ευέλικτη επιλογή για τις επιχειρήσεις και τους προγραμματιστές που αναζητούν αποτελεσματικές και οικονομικά αποδοτικές λύσεις για την ανάπτυξη σύγχρονων εφαρμογών.

2.1.2.7 Ενεργή κοινότητα και οικοσύστημα

Μια ενεργή και ζωντανή κοινότητα αποτελεί ακρογωνιαίο λίθο της επιτυχίας του Ionic Framework και διαδραματίζει καθοριστικό ρόλο στην υποστήριξη των προγραμματιστών και των επιχειρήσεων καθ' όλη τη διάρκεια του ταξιδιού ανάπτυξης εφαρμογών.

Το Ionic έχει καλλιεργήσει μια μεγάλη και ακμάζουσα κοινότητα προγραμματιστών, συνεργατών και ενθουσιωδών. Η ενεργή συμμετοχή αυτής της κοινότητας προάγει ένα συνεργατικό περιβάλλον όπου ευδοκιμούν η ανταλλαγή γνώσεων, η επίλυση προβλημάτων και η καινοτομία. Οι προγραμματιστές από όλα τα επίπεδα εμπειρίας μπορούν να επωφεληθούν από αυτή τη συλλογική σοφία, καθιστώντας το Ionic μια προσιτή επιλογή για όσους είναι νέοι στο πλαίσιο και μια πλούσια σε πόρους πλατφόρμα για τους έμπειρους επαγγελματίες.

Ένας από τους βασικούς πυλώνες της υποστήριξης της κοινότητας του Ionic είναι η εκτεταμένη τεκμηρίωσή του. Το Ionic παρέχει ολοκληρωμένη και καλά δομημένη τεκμηρίωση που καλύπτει κάθε πτυχή του πλαισίου. Αυτή η τεκμηρίωση χρησιμεύει ως πολύτιμη αναφορά για τους προγραμματιστές, προσφέροντας λεπτομερείς εξηγήσεις, δείγματα κώδικα και βέλτιστες πρακτικές. Είτε πρόκειται για την έναρξη χρήσης του Ionic, είτε για την εξερεύνηση των δυνατοτήτων του, είτε για την αντιμετώπιση προβλημάτων, η τεκμηρίωση παρέχει την καθοδήγηση που απαιτείται για την αποτελεσματική πλοήγηση στο πλαίσιο.

Εκτός από την τεκμηρίωση, το Ionic προσφέρει μια πληθώρα σεμιναρίων, οδηγών και πηγών εκμάθησης. Αυτό το εκπαιδευτικό υλικό απευθύνεται σε προγραμματιστές που επιθυμούν να βελτιώσουν τις δεξιότητές τους, να μάθουν τις βέλτιστες πρακτικές και να ανακαλύψουν νέους τρόπους αξιοποίησης των δυνατοτήτων του Ionic. Τα σεμινάρια κυμαίνονται από φιλικές προς τους αρχάριους εισαγωγές έως προηγμένες τεχνικές, διασφαλίζοντας ότι προγραμματιστές όλων των επιπέδων δεξιοτήτων θα βρουν σχετικό περιεχόμενο.

Το οικοσύστημα του Ionic επεκτείνεται πέρα από την τεκμηρίωση και τα σεμινάρια και περιλαμβάνει μια τεράστια συλλογή από πρόσθετα και ενσωματώσεις τρίτων. Αυτά τα πρόσθετα καλύπτουν ένα ευρύ φάσμα λειτουργιών, από την ενσωμάτωση με δημοφιλείς υπηρεσίες backend έως την προσθήκη συγκεκριμένων χαρακτηριστικών όπως ο γεωγραφικός εντοπισμός, ο έλεγχος ταυτότητας και η κοινή χρήση κοινωνικών μέσων. Η διαθεσιμότητα αυτών των plugins απλοποιεί τη διαδικασία ανάπτυξης, εξοικονομώντας χρόνο και προσπάθεια, παρέχοντας προκατασκευασμένες λύσεις σε κοινές προκλήσεις.

Η ενεργή κοινότητα συμβάλλει περαιτέρω στο οικοσύστημα δημιουργώντας και συντηρώντας αυτά τα πρόσθετα, διασφαλίζοντας ότι παραμένουν ενημερωμένα και συμβατά με τις τελευταίες εκδόσεις του Ionic. Οι προγραμματιστές μπορούν να αξιοποιήσουν αυτό το πλούσιο αποθετήριο πόρων για να βελτιώσουν τις εφαρμογές τους με ποικίλα χαρακτηριστικά και λειτουργίες.

Συμπερασματικά, η ενεργή κοινότητα και το οικοσύστημα του Ionic αποτελούν απόδειξη της δύναμης και της ανθεκτικότητας του πλαισίου. Αυτή η υποστηρικτική κοινότητα παρέχει στους προγραμματιστές τις γνώσεις, τους πόρους και τα εργαλεία που χρειάζονται για να επιτύχουν

στις προσπάθειες ανάπτυξης εφαρμογών. Είτε μέσω της τεκμηρίωσης, είτε μέσω σεμιναρίων είτε μέσω πρόσθετων προγραμμάτων, η δέσμευση του Ionic για την υποστήριξη της κοινότητας εμπλουτίζει την εμπειρία ανάπτυξης και επιταχύνει τη δημιουργία εφαρμογών υψηλής ποιότητας σε πολλαπλές πλατφόρμες.

2.1.2.8 Progressive Web Apps (PWAs)

Το Ionic ξεχωρίζει ως ένα εξέχον framework για τη δημιουργία Progressive Web Apps (PWAs), μια σύγχρονη προσέγγιση στην ανάπτυξη ιστοσελίδων που συνδυάζει τα καλύτερα των εμπειριών εφαρμογών ιστού και κινητών συσκευών. Οι PWAs έχουν σχεδιαστεί για να παρέχουν στους χρήστες μια απρόσκοπτη και ελκυστική αλληλεπίδραση και το Ionic προσφέρει τα εργαλεία και τα χαρακτηριστικά που είναι απαραίτητα για την πλήρη αξιοποίηση των δυνατοτήτων αυτής της τεχνολογίας.

Τα PWA που αναπτύσσονται με το Ionic διαθέτουν αρκετά διακριτικά χαρακτηριστικά που βελτιώνουν την εμπειρία του χρήστη:

- *Offline λειτουργικότητα:* Ένα από τα καθοριστικά χαρακτηριστικά των PWAs είναι η ικανότητά τους να λειτουργούν εκτός σύνδεσης ή σε συνθήκες χαμηλού δικτύου. Το Ionic δίνει τη δυνατότητα στους προγραμματιστές να εφαρμόζουν service workers και στρατηγικές προσωρινής αποθήκευσης, επιτρέποντας στα PWAs να αποθηκεύουν τοπικά βασικά στοιχεία. Αυτό σημαίνει ότι οι χρήστες μπορούν να έχουν πρόσβαση και να αλληλεπιδρούν με την εφαρμογή ακόμη και όταν βρίσκονται εκτός σύνδεσης, εξασφαλίζοντας συνεχή δέσμευση και παραγωγικότητα.
- *Γρήγοροι χρόνοι φόρτωσης:* Τα εργαλεία βελτιστοποίησης επιδόσεων του Ionic συμβάλλουν σε γρήγορους χρόνους φόρτωσης, έναν κρίσιμο παράγοντα για την ικανοποίηση των χρηστών. Τα PWA που έχουν κατασκευαστεί με το Ionic είναι σχεδιασμένα για να φορτώνουν γρήγορα, παρέχοντας περιεχόμενο και λειτουργικότητα χωρίς απογοητευτικές καθυστερήσεις. Αυτή η ταχύτατη απόδοση ενισχύει τη διατήρηση και τη δέσμευση των χρηστών.
- *Ανταπόκριση σε διάφορες συσκευές:* Οι αρχές responsive design του Ionic διασφαλίζουν ότι τα PWA προσαρμόζονται απρόσκοπτα σε διάφορες συσκευές, μεγέθη οθόνης και αναλύσεις. Αυτή η ανταποκρισιμότητα είναι ιδιαίτερα πολύτιμη καθώς οι χρήστες έχουν πρόσβαση στα PWAs σε smartphones, tablets, επιτραπέζιους υπολογιστές και άλλες συσκευές. Ανεξάρτητα από την πλατφόρμα, οι χρήστες λαμβάνουν μια βελτιστοποιημένη και οπτικά ελκυστική εμπειρία.
- *Προσβασιμότητα πολλαπλών πλατφορμών:* Η συμβατότητα πολλαπλών πλατφορμών του Ionic επεκτείνεται και στα PWAs. Οι προγραμματιστές μπορούν να δημιουργήσουν PWAs που λειτουργούν άψογα σε διαφορετικά προγράμματα περιήγησης ιστού, εξασφαλίζοντας μια συνεπή και αξιόπιστη εμπειρία για τους χρήστες, ανεξάρτητα από το πρόγραμμα περιήγησης ή τη συσκευή που προτιμούν.
- *Εμπλοκή και επαναπροσέγγιση:* Τα PWAs προσφέρουν χαρακτηριστικά όπως οι ειδοποιήσεις push, οι οποίες επιτρέπουν στους προγραμματιστές να επαναπροσέγγισουν τους χρήστες και να τους κρατούν ενήμερους για ενημερώσεις και σχετικό περιεχόμενο. Αυτό το χαρακτηριστικό είναι ανεκτίμητο για τη διατήρηση του ενδιαφέροντος των χρηστών και τη διασφάλιση της επιστροφής τους στο PWA με την πάροδο του χρόνου.

- *Εγκατάσταση και πρόσβαση στην αρχική οθόνη:* Τα PWA που έχουν αναπτυχθεί με το Ionic μπορούν να εγκατασταθούν στις συσκευές των χρηστών απευθείας από το πρόγραμμα περιήγησης, επιτρέποντάς τους να προσθέσουν την εφαρμογή στην αρχική τους οθόνη. Αυτό παρέχει μια εμπειρία για εύκολη και βολική πρόσβαση για τους χρήστες.

Συνοψίζοντας, η υποστήριξη του Ionic για PWAs ευθυγραμμίζεται απόλυτα με τις απαιτήσεις της σύγχρονης ανάπτυξης ιστοσελίδων. Επιτρέπει στους προγραμματιστές να δημιουργούν εφαρμογές που δεν είναι μόνο ευέλικτες και γρήγορες, αλλά και ικανές να παρέχουν εξαιρετική εμπειρία χρήστη, λειτουργικότητα εκτός σύνδεσης και δυνατότητα αποτελεσματικής εμπλοκής και επαναπροσέγγισης των χρηστών. Η δέσμευση του Ionic στα PWAs το καθιστά μια ελκυστική επιλογή για τις επιχειρήσεις που επιθυμούν να παρέχουν πρωτοποριακές εφαρμογές ιστού με τις επιδόσεις και τα χαρακτηριστικά που αναμένουν οι χρήστες στο σημερινό ψηφιακό τοπίο.

2.1.2.9 Θεματοποίηση και διαμόρφωση

Η θεματοποίηση και το styling είναι κρίσιμα στοιχεία κάθε επιτυχημένης εφαρμογής και το Ionic Framework δίνει μεγάλη έμφαση σε αυτές τις πτυχές, προσφέροντας στους προγραμματιστές ένα ισχυρό σύστημα για τη δημιουργία οπτικά ελκυστικών και συνεπών διεπαφών χρήστη.

Το σύστημα θεματοποίησης του Ionic είναι ένα χαρακτηριστικό που ξεχωρίζει και απλοποιεί τη διαδικασία ορισμού και διατήρησης μιας συνεκτικής γλώσσας σχεδιασμού για την εφαρμογή. Παρέχει στους προγραμματιστές προκατασκευασμένα θέματα και στυλ που χρησιμεύουν ως σταθερά θεμέλια για την οπτική ταυτότητα της εφαρμογής. Αυτά τα θέματα περιλαμβάνουν την τυπογραφία, τους χρωματικούς συνδυασμούς, την εικονογραφία και διάφορα στοιχεία του UI, εξασφαλίζοντας μια αρμονική και επαγγελματική εμφάνιση και αίσθηση.

Ένα από τα αξιοσημείωτα πλεονεκτήματα του συστήματος θεματοποίησης του Ionic είναι η δυνατότητα προσαρμογής του. Οι προγραμματιστές έχουν την ευελιξία να προσαρμόζουν αυτά τα προκατασκευασμένα θέματα ώστε να ταιριάζουν ακριβώς με την επωνυμία και τις σχεδιαστικές απαιτήσεις της εφαρμογής τους. Αυτή η προσαρμογή επεκτείνεται σε πτυχές όπως η τοποθέτηση του λογότυπου, οι προσαρμογές της χρωματικής παλέτας, οι επιλογές γραμματοσειράς και ο συνολικός σχεδιασμός της διεπαφής χρήστη. Αυτό το επίπεδο ελέγχου επιτρέπει στις επιχειρήσεις να δημιουργούν εφαρμογές που αντικατοπτρίζουν τη μοναδική ταυτότητα του εμπορικού τους σήματος και να έχουν απήχηση στο κοινό-στόχο τους.

Επιπλέον, το σύστημα θεματοποίησης του Ionic εξασφαλίζει μια συνεπή διεπαφή χρήστη σε ολόκληρη την εφαρμογή. Η συνέπεια είναι απαραίτητη για την παροχή στους χρήστες μιας συνεκτικής και προβλέψιμης εμπειρίας καθώς περιηγούνται στην εφαρμογή. Με την τήρηση μιας ενιαίας γλώσσας σχεδιασμού, οι εφαρμογές που υποστηρίζονται από το Ionic διατηρούν μια επαγγελματική και όμορφη εμφάνιση, ενισχύοντας την εμπιστοσύνη και την ικανοποίηση των χρηστών.

Οι δυνατότητες θεματοποίησης και διαμόρφωσης του Ionic είναι ιδιαίτερα πολύτιμες για τις επιχειρήσεις που επιδιώκουν να δημιουργήσουν μια ισχυρή παρουσία της μάρκας μέσω των εφαρμογών τους. Το σύστημα αυτό απλοποιεί τη διαδικασία διατήρησης μιας συνεπούς εμφάνισης και αισθητικής, ακόμη και όταν η εφαρμογή εξελίσσεται με την πάροδο του χρόνου με βελτιώσεις και ενημερώσεις λειτουργιών.

Συμπερασματικά, οι δυνατότητες θεματοποίησης και διαμόρφωσης του Ionic δίνουν τη δυνατότητα στους προγραμματιστές να δημιουργούν εφαρμογές που δεν είναι μόνο οπτικά ελκυστικές αλλά και διατηρούν υψηλό επίπεδο συνοχής. Αυτό συμβάλλει στη θετική εμπειρία του χρήστη, ενισχύει την ταυτότητα της μάρκας και βελτιώνει τον συνολικό επαγγελματισμό της εφαρμογής, καθιστώντας το Ionic ιδανική επιλογή για επιχειρήσεις και προγραμματιστές που δίνουν προτεραιότητα στο σχεδιασμό και την αισθητική στα προϊόντα λογισμικού τους.

2.1.2.10 Platform Agnostic

Ο αγνωστικισμός πλατφόρμας (platform agnostic) είναι ένα βασικό χαρακτηριστικό του Ionic Framework και αποτελεί στρατηγικό πλεονέκτημα για τις επιχειρήσεις και τους προγραμματιστές που αναζητούν ευελιξία στις προσπάθειες ανάπτυξης εφαρμογών. Η προσέγγιση του Ionic ως προς την ανεξαρτησία πλατφόρμας σημαίνει ότι δεν είναι προσδεδεμένο σε ένα συγκεκριμένο λειτουργικό σύστημα ή μια συγκεκριμένη τεχνολογία, προσφέροντας στις επιχειρήσεις την ελευθερία να λαμβάνουν τεκμηριωμένες αποφάσεις σχετικά με τις πλατφόρμες στις οποίες θα στοχεύσουν, χωρίς να δεσμεύονται από περιοριστικούς περιορισμούς.

Ένα από τα κύρια οφέλη της ανεξαρτησίας από πλατφόρμες είναι η δυνατότητα προσαρμογής στις μεταβαλλόμενες δυναμικές της αγοράς και στις προτιμήσεις των χρηστών. Οι επιχειρήσεις μπορούν να αξιολογούν το τοπίο και να κατανέμουν τους πόρους με βάση τις τάσεις της αγοράς και τη ζήτηση των χρηστών αντί να δεσμεύονται σε μια μοναδική τεχνολογική επιλογή. Είτε πρόκειται για εφαρμογές iOS, Android ή web, η Ionic παρέχει μια ευέλικτη λύση που επιτρέπει στις επιχειρήσεις να αναπτύσσουν εφαρμογές εκεί όπου είναι πιο πιθανό να επιτύχουν.

Επιπλέον, η ανεξαρτησία του Ionic από την πλατφόρμα εξασφαλίζει ότι οι ομάδες ανάπτυξης δεν χρειάζεται να διαθέτουν τεχνογνωσία για συγκεκριμένες πλατφόρμες ή να συμμετέχουν σε ξεχωριστές προσπάθειες ανάπτυξης για διαφορετικά λειτουργικά συστήματα. Αυτό μεταφράζεται σε εξοικονόμηση κόστους, καθώς οι επιχειρήσεις μπορούν να αξιοποιήσουν μια ενιαία, ενοποιημένη διαδικασία ανάπτυξης που καλύπτει πολλαπλές πλατφόρμες. Απλοποιεί επίσης την απόκτηση ταλέντων, καθώς οι επιχειρήσεις μπορούν να αξιοποιήσουν μια μεγαλύτερη δεξαμενή προγραμματιστών με δεξιότητες ανάπτυξης ιστού, αντί να απαιτούν τεχνογνωσία σε συγκεκριμένες εγγενείς τεχνολογίες.

Η ανεξαρτησία από πλατφόρμες ευθυγραμμίζεται με μια προοδευτική προσέγγιση στην ανάπτυξη εφαρμογών, όπου οι επιχειρήσεις μπορούν να πειραματιστούν, να επαναλάβουν και να προσαρμόσουν τις στρατηγικές τους με βάση την ανατροφοδότηση του πραγματικού κόσμου και τη δυναμική της αγοράς. Αυτή η προσαρμοστικότητα είναι απαραίτητη στο σημερινό ταχέως εξελισσόμενο ψηφιακό τοπίο, όπου η τεχνολογία και οι προτιμήσεις των χρηστών εξελίσσονται συνεχώς.

Εν κατακλείδι, η φιλοσοφία της Ionic που δεν βασίζεται σε πλατφόρμες δίνει στις επιχειρήσεις την ελευθερία να λαμβάνουν στρατηγικές αποφάσεις σχετικά με τη στόχευση σε πλατφόρμες, αξιοποιώντας τη δύναμη της ανάπτυξης πολλαπλών πλατφορμών χωρίς να περιορίζονται από το τεχνολογικό κλείδωμα. Αυτή η ευελιξία, σε συνδυασμό με την οικονομική αποδοτικότητα και την ευκολία ανάπτυξης, τοποθετεί το Ionic ως ιδανική επιλογή για οργανισμούς που αναζητούν ευελιξία και προσαρμοστικότητα στις προσπάθειες ανάπτυξης εφαρμογών.

2.1.3 Περιπτώσεις χρήσης

Το Ionic Framework βρίσκει ευέλικτες εφαρμογές σε ένα ευρύ φάσμα βιομηχανιών και διαφορετικών περιπτώσεων χρήσης. Ξεχωρίζει στην ανάπτυξη εφαρμογών για κινητά που απευθύνονται σε καταναλωτές, παρέχοντας στις επιχειρήσεις μια οικονομικά αποδοτική λύση για να προσεγγίσουν το κοινό-στόχο τους σε πολλαπλές πλατφόρμες ταυτόχρονα. Στο πεδίο των εταιρικών εφαρμογών για κινητά, το Ionic προσφέρει ένα πολύτιμο εργαλείο για τους οργανισμούς για τον εξορθολογισμό των εσωτερικών διαδικασιών, την ενίσχυση της παραγωγικότητας των εργαζομένων και τη βελτιστοποίηση της επικοινωνίας. Οι εφαρμογές ηλεκτρονικού εμπορίου αξιοποιούν τη συμβατότητα του Ionic σε πολλαπλές πλατφόρμες για να προσφέρουν απρόσκοπτη εμπειρία αγορών στους πελάτες, είτε χρησιμοποιούν iOS, Android ή τον ιστό. Επιπλέον, οι εφαρμογές μέσων κοινωνικής δικτύωσης επωφελούνται από τις δυνατότητες ταχείας ανάπτυξης του Ionic και την πρόσβαση σε εγγενή χαρακτηριστικά συσκευών, επιτρέποντας στους χρήστες να συμμετέχουν με πλούσιο, διαδραστικό περιεχόμενο σε διάφορες συσκευές.

Οι εκπαιδευτικές εφαρμογές και οι πλατφόρμες ηλεκτρονικής μάθησης βασίζονται στο Ionic για τη δημιουργία προσβάσιμων και ελκυστικών μαθησιακών εμπειριών, ενώ οι εφαρμογές υγείας και γυμναστικής αξιοποιούν την ευελιξία του για την παρακολούθηση δεδομένων υγείας και την ενθάρρυνση του ενεργού τρόπου ζωής. Οι εφαρμογές κρατήσεων και κρατήσεων, που απευθύνονται σε κλάδους όπως τα ταξίδια και η φιλοξενία, χρησιμοποιούν το Ionic για να παρέχουν μια συνεπή και φιλική προς το χρήστη εμπειρία κρατήσεων. Συνοψίζοντας, το Ionic Framework έχει αποδείξει την προσαρμοστικότητα και την αποτελεσματικότητά του σε μια πληθώρα τομέων, καθιστώντας το μια πολύτιμη επιλογή για ποικίλες ανάγκες ανάπτυξης εφαρμογών.

Το Ionic Framework είναι ένα ισχυρό εργαλείο στον κόσμο της ανάπτυξης εφαρμογών πολλαπλών πλατφορμών. Δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν πλούσιες σε χαρακτηριστικά, οπτικά ελκυστικές και αποδοτικές εφαρμογές που λειτουργούν απρόσκοπτα σε πολλαπλές πλατφόρμες, προσφέροντας μια οικονομικά αποδοτική και αποτελεσματική λύση για σύγχρονα έργα ανάπτυξης εφαρμογών. Η ενσωμάτωσή του με τεχνολογίες ιστού, η πρόσβαση σε δυνατότητες εγγενών συσκευών και η ενεργή υποστήριξη της κοινότητας το καθιστούν πολύτιμη επιλογή για τους προγραμματιστές που στοχεύουν να προσεγγίσουν ένα ευρύ κοινό με τις εφαρμογές τους.

2.2 Διαφορές Ionic Framework με React Native

Παρακάτω συγκρίνονται δύο διαφορετικά frameworks ανάπτυξης εφαρμογών, το Ionic και η React Native. Μέσω αυτής της σύγκρισης και ανάλυσης θα φανούν οι λόγοι επιλογής του Ionic Framework από την ομάδα εργασίας, σε σχέση με τη React Native. Η ανάλυση υπογραμμίζει ότι και τα δύο frameworks έχουν τα δυνατά τους σημεία, καθιστώντας δύσκολη την οριστική ανακήρυξη του ενός ως ανώτερου του άλλου. Η επιλογή μεταξύ του Ionic και της React Native εξαρτάται από τις συγκεκριμένες απαιτήσεις του έργου, τις λειτουργίες και το κοινό-στόχο. Ενώ το Ionic διακρίνεται για την ταχεία ανάπτυξη και τη διαθεσιμότητα του κώδικα, το React Native προσφέρει υψηλές επιδόσεις, πλούσιο UI και εξαιρετική εμπειρία χρήστη. Η σαφής κατανόηση των στόχων του έργου είναι ζωτικής σημασίας για την αποτελεσματική αξιοποίηση των πλεονεκτημάτων αυτών των framework.

Η επιλογή του καταλληλότερου πλαισίου είναι μια κρίσιμη απόφαση που μπορεί να επηρεάσει σημαντικά την επιτυχία ενός έργου. Καθώς η ζήτηση για εφαρμογές για κινητά συνεχίζει να

αυξάνεται σε διάφορους κλάδους, οι προγραμματιστές, οι επιχειρήσεις και οι υπεύθυνοι λήψης αποφάσεων βρίσκονται αντιμέτωποι με ένα δύσκολο δίλημμα - ποιο framework να επιλέξουν.

Το Ionic και το React Native έχουν αναδειχθεί κύριες επιλογές σε αυτόν τον τομέα, προσφέροντας το καθένα το δικό του μοναδικό σύνολο πλεονεκτημάτων και δυνατοτήτων. Ωστόσο, το έργο της απόφασης μεταξύ αυτών των δύο framework δεν είναι καθόλου απλό, καθώς τα πλεονεκτήματα και οι αδυναμίες τους ανταποκρίνονται σε διαφορετικές απαιτήσεις και στόχους έργων.

2.2.1 Διαφορά στην προσέγγιση: Web vs Native

Το Ionic Framework με χρήση της React και το React Native ακολουθούν αποκλίνουσες πορείες όσον αφορά την υλοποίηση των διεπαφών χρήστη των εφαρμογών για κινητά, γεγονός που επηρεάζει σημαντικά τις διαδικασίες και τις δυνατότητές τους για την ανάπτυξη.

2.2.1.1 Η προσέγγιση Web-First του Ionic React:

Το Ionic React έχει σχεδιαστεί με μια φιλοσοφία web-first, που σημαίνει ότι αξιοποιεί τεχνολογίες ιστού βασισμένες σε πρότυπα διαπλατφορμών για να μιμηθεί τα εγγενή μοτίβα UI iOS και Android. Αντί να έχει άμεση πρόσβαση σε στοιχεία ελέγχου UI συγκεκριμένων πλατφορμών, το Ionic React βασίζεται στην τεχνολογία ιστού για τη δημιουργία μιας ενοποιημένης διεπαφής χρήστη σε πολλαπλές πλατφόρμες. Αυτή η προσέγγιση προσφέρει πολλά πλεονεκτήματα:

- *Απόδοση και συμβατότητα:* Χρησιμοποιώντας κατά κύριο λόγο τυποποιημένους ελέγχους iOS και Android κάτω από την επιφάνεια, το Ionic React μπορεί, σε ορισμένες περιπτώσεις, να προσφέρει ανώτερες επιδόσεις. Οι εφαρμογές που δημιουργούνται με το Ionic React μοιάζουν ιδιαίτερα με τις εφαρμογές της πλατφόρμας react, εξασφαλίζοντας συμβατότητα και εξοικείωση για τους χρήστες.
- *Ευκολία ενσωμάτωσης:* Η διασύνδεση με τους υπάρχοντες ελέγχους UI εγγενών εφαρμογών είναι σχετικά απλή στο Ionic React. Αυτή η ευκολία ενσωμάτωσης απλοποιεί τη διαδικασία ενσωμάτωσης εγγενών στοιχείων στο σχεδιασμό της εφαρμογής σας.
- *Εμπειρία ανάπτυξης:* Το Ionic React αντικατοπτρίζει στενά την παραδοσιακή ανάπτυξη εφαρμογών ιστού React. Οι προγραμματιστές μπορούν να αξιοποιήσουν την οικειότητα που έχουν με τα εργαλεία που βασίζονται στον ιστό, απολαμβάνοντας μια γρήγορη και οικεία εμπειρία ανάπτυξης απευθείας μέσα σε προγράμματα περιήγησης ιστού, όπως το Chrome (καθώς και τα React Developer Tools που το συνοδεύουν). Αυτή η ομοιότητα επιτρέπει την κατασκευή σημαντικών τμημάτων της λειτουργικότητας μιας εφαρμογής απευθείας στο πρόγραμμα περιήγησης. Επιπλέον, το Ionic React μπορεί να ενσωματωθεί απρόσκοπτα σε υπάρχουσες διαδικτυακές εφαρμογές React.

2.2.1.2 Η προσέγγιση της React Native που βασίζεται στην πλατφόρμα:

Αντίθετα, το React Native υιοθετεί μια φιλοσοφία που ξεκινάει από την πλατφόρμα. Παρέχει ένα επίπεδο αφαίρεσης μέσω της React που επιτρέπει τον άμεσο έλεγχο των στοιχείων ελέγχου του UI που αφορούν συγκεκριμένες πλατφόρμες. Αυτή η προσέγγιση έχει τα πλεονεκτήματά της, αλλά συνοδεύεται και από ορισμένες προκλήσεις:

- *Απόδοση και προσαρμογή:* Η χρήση των εγγενών στοιχείων ελέγχου της React Native μπορεί να οδηγήσει σε βέλτιστες επιδόσεις σε ορισμένα σενάρια. Ωστόσο, το μειονέκτημα είναι ότι αυτά τα εγγενή στοιχεία ελέγχου είναι συχνά λιγότερο προσαρμόσιμα σε σύγκριση με τις τεχνολογίες ιστού. Αυτός ο περιορισμός μπορεί να δημιουργήσει προκλήσεις για τους προγραμματιστές και τους σχεδιαστές που επιθυμούν να δημιουργήσουν ιδιαίτερα προσαρμοσμένες εμπειρίες UI.
- *Συμβατότητα τεχνολογίας ιστού:* Το React Native δεν είναι ένα πραγματικό περιβάλλον προγράμματος περιήγησης και, ως εκ τούτου, δεν μπορεί να χρησιμοποιήσει τις τυπικές τεχνικές CSS, HTML και DOM. Παρόλο που προσφέρει μια χρησιμότητα που μοιάζει με CSS, οι προγραμματιστές και οι σχεδιαστές ιστού μπορεί να αντιμετωπίσουν μια δύσκολη καμπύλη εκμάθησης όταν εργάζονται με το React Native.
- *Προκλήσεις φορητότητας:* Ο υφιστάμενος κώδικας και οι βιβλιοθήκες της React web ενδέχεται να μην μεταφερθούν απρόσκοπτα στο React Native. Η προσαρμογή των στοιχείων που βασίζονται στον ιστό για το React Native απαιτεί συνήθως σημαντική προσαρμογή.
- *Διαδικασία ανάπτυξης:* Η διαδικασία ανάπτυξης της React Native διαφέρει σημαντικά από την παραδοσιακή ανάπτυξη ιστού. Προσφέρει περιορισμένη υποστήριξη για ενσωμάτωση με έναν αποσφαλματωτή - debugger του Chrome, αλλά απαιτεί την εκτέλεση σε εξομοιωτή ή σε φυσική συσκευή, σε αντίθεση με το Ionic React. Αυτή η διάκριση σημαίνει ότι η αποσφαλμάτωση στο React Native δεν παρέχει μια πραγματική εμπειρία βασισμένη στο πρόγραμμα περιήγησης, ενώ είναι περισσότερο χρονοβόρα.

2.2.1.3 Οικοσύστημα πλατφόρμας και επενδύσεις:

Μια κρίσιμη διάκριση έγκειται στα οικοσυστήματα αυτών των πλαισίων. Το Ionic React είναι βαθιά ριζωμένο στην πλατφόρμα ιστού, καθιστώντας το μια φυσική επιλογή για ομάδες που έχουν ήδη επενδύσει σε τεχνολογίες ιστού. Αντίθετα, το React Native είναι μια αυτοτελής πλατφόρμα και οικοσύστημα, που απαιτεί αυτάρκεια για μακροπρόθεσμη επιτυχία. Η επιλογή της React Native συνεπάγεται μια αφοσιωμένη δέσμευση στο μοναδικό οικοσύστημά του.

Συνοπτικά, η επιλογή μεταξύ του Ionic React και της React Native εξαρτάται από το αν δίνεται προτεραιότητα στην εξοικείωση και την ευελιξία των τεχνολογιών που βασίζονται στον ιστό ή στις επιδόσεις και τον έλεγχο που παρέχουν τα εγγενή στοιχεία για συγκεκριμένες πλατφόρμες. Η κατανόηση των θεμελιωδών διαφορών στις προσεγγίσεις τους είναι καθοριστικής σημασίας για την επιλογή του καταλληλότερου πλαισίου για το έργο ανάπτυξης εφαρμογών για κινητά τηλέφωνα.

2.2.2 Διαφορά στην υποστήριξη Progressive Web App

Οι προοδευτικές εφαρμογές ιστού (Progressive Web Apps, PWA) αποτελούν ένα σύγχρονο πρότυπο για την κατασκευή εφαρμογών για κινητά που διαθέτουν τα χαρακτηριστικά των εγγενών εφαρμογών, ενώ διανέμονται μέσω ενός προγράμματος περιήγησης ιστού και όχι αποκλειστικά μέσω των παραδοσιακών καταστημάτων εφαρμογών. Οι PWAs προσφέρουν μια σειρά πλεονεκτημάτων, συμπεριλαμβανομένης της ενισχυμένης δέσμευσης των χρηστών, της βελτιωμένης βελτιστοποίησης για τις μηχανές αναζήτησης και της αυξημένης δυνατότητας κοινής χρήσης. Κατά συνέπεια, οι PWAs έχουν κερδίσει σημαντική θέσεις τόσο στους τομείς των καταναλωτικών όσο και των επιχειρηματικών εφαρμογών.

Μια αξιοσημείωτη διαφορά μεταξύ του Ionic React και της React Native είναι η προσέγγισή τους στην υποστήριξη Progressive Web App:

2.2.2.1 Η υποστήριξη PWA του Ionic React:

Το Ionic React υπερέχει από αυτή την άποψη, παρέχοντας πρώτης τάξεως υποστήριξη για Progressive Web Apps. Αυτό σημαίνει ότι το Ionic React δίνει τη δυνατότητα στις ομάδες ανάπτυξης να δημιουργούν και να διανέμουν απρόσκοπτα εφαρμογές τόσο σε καταστήματα εφαρμογών iOS και Android όσο και ως Progressive Web Apps στο διαδίκτυο, όλα από μία ενιαία βάση κώδικα. Αυτή η προσέγγιση προσφέρει πολλά αξιοσημείωτα πλεονεκτήματα:

- *Ενοποιημένη ανάπτυξη:* Οι ομάδες μπορούν να εργάζονται σε μια ενιαία βάση κώδικα για τις εφαρμογές τους για κινητά, απλοποιώντας την ανάπτυξη και μειώνοντας την ανάγκη για προσαρμογές που αφορούν συγκεκριμένες πλατφόρμες.
- *Μεγαλύτερη εμβέλεια:* Αξιοποιώντας τη δύναμη των Progressive Web Apps, το Ionic React δίνει τη δυνατότητα στην εφαρμογή μας να προσεγγίσει ένα ευρύτερο κοινό, καθώς μπορεί να προσεγγιστεί μέσω προγραμμάτων περιήγησης ιστού σε διάφορες συσκευές χωρίς την ανάγκη εγκατάστασης από καταστήματα εφαρμογών.
- *Αποδοτικότητα κόστους:* Η δυνατότητα ταυτόχρονης στόχευσης πολλαπλών πλατφορμών με την ίδια βάση κώδικα έχει ως αποτέλεσμα την εξοικονόμηση κόστους, καθιστώντας το Ionic React μια ελκυστική επιλογή για οργανισμούς που επιθυμούν να μεγιστοποιήσουν τους πόρους ανάπτυξής τους.
- *Βελτιωμένη εμπειρία χρήστη:* Οι PWA που δημιουργούνται με το Ionic React προσφέρουν στους χρήστες μια απρόσκοπτη και ευέλικτη εμπειρία, είτε η πρόσβαση γίνεται μέσω ενός προγράμματος περιήγησης είτε εγκαθίσταται ως εγγενής εφαρμογή.

2.2.2.2 Η περιορισμένη υποστήριξη PWA της React Native:

Αντίθετα, το React Native δεν υποστηρίζει επίσημα τις Progressive Web Apps στον ίδιο βαθμό. Ενώ το React Native είναι γνωστό για τις δυνατότητές του στην ανάπτυξη εγγενών εφαρμογών για iOS και Android, η κύρια εστίασή του παραμένει στην παροχή εμπειριών που μοιάζουν με εγγενείς εφαρμογές. Αυτό σημαίνει ότι το React Native δεν είναι εξοπλισμένο με εγγενή εργαλεία και διαμορφώσεις για την κατασκευή PWAs.

2.2.2.3 Σκέψεις για μελλοντικά σχέδια:

Εάν η ομάδα μας φιλοδοξεί να ενσωματώσει τις εφαρμογές προοδευτικού ιστού στη στρατηγική διανομής εφαρμογών, το Ionic React αποτελεί μια επιτακτική λύση. Επιλέγοντας το Ionic React, η ομάδα μας μπορεί να κατασκευάσει και να διανείμει εφαρμογές σε μεγάλα καταστήματα εφαρμογών και στον ιστό ταυτόχρονα, απλοποιώντας τη διαδικασία ανάπτυξης και μεγιστοποιώντας την εμβέλεια της εφαρμογής σας.

Συνοψίζοντας, κατά την αξιολόγηση του Ionic React και της React Native για το έργο μας, είναι σημαντικό να εξετάζουμε τα μελλοντικά μας σχέδια, ειδικά αν οι Progressive Web Apps αποτελούν μέρος της στρατηγικής διανομής μας. Η ισχυρή υποστήριξη του Ionic React για PWAs το καθιστά κατάλληλη επιλογή για ομάδες που αναζητούν μια ενιαία προσέγγιση στην ανάπτυξη εφαρμογών που εκτείνεται σε πλατφόρμες ιστού και κινητών.

2.2.3 Διαφορά στην υποστήριξη πολλαπλών πλατφορμών

Η προσέγγιση της υποστήριξης πολλαπλών πλατφορμών είναι ένας κρίσιμος παράγοντας που διακρίνει το React Native και το Ionic React, με κάθε framework να υιοθετεί μια ξεχωριστή στρατηγική για τη στόχευση πολλαπλών πλατφορμών.

2.2.3.1 Η εστίαση της React Native σε συγκεκριμένες πλατφόρμες:

Το React Native, στον πυρήνα του, είναι μια βιβλιοθήκη UI που προσφέρει επίσημη υποστήριξη πρωτίστως για την ανάπτυξη εφαρμογών iOS και Android στο κατάστημα εφαρμογών. Ενώ υπάρχουν ανεπίσημα έργα και προσπάθειες της κοινότητας για την επέκταση των δυνατοτήτων της React Native σε πλατφόρμες desktop και web, η πρωταρχική και επίσημη εστίασή του παραμένει στην ανάπτυξη εφαρμογών για κινητά τηλέφωνα για iOS και Android. Το React Native ακολουθεί τη φιλοσοφία μάθε μία φορά, γράψε οπουδήποτε", που σημαίνει ότι οι προγραμματιστές αποκτούν ένα κοινό σύνολο εννοιών (με επίκεντρο το React με προσαρμογές της React Native) και στη συνέχεια δημιουργούν οθόνες διεπαφής χρήστη που είναι συγκεκριμένες για κάθε πλατφόρμα. Αυτή η προσέγγιση καθιστά αναγκαία τη δημιουργία οθονών UI προσαρμοσμένων ξεχωριστά για iOS και Android.

2.2.3.2 Η υιοθέτηση πολλαπλών πλατφορμών από το Ionic React:

Το Ionic React, από την άλλη πλευρά, υιοθετεί μια ευρύτερη προσέγγιση "γράψτε μία φορά, εκτελέστε οπουδήποτε". Υποστηρίζει επίσημα ένα πιο εκτεταμένο φάσμα πλατφορμών, συμπεριλαμβανομένων των iOS, Android, Electron (για εφαρμογές γραφείου) και τον ιστό, χρησιμοποιώντας την τεχνολογία Progressive Web App (PWA). Αυτή η προσέγγιση προσφέρει πολλά πλεονεκτήματα:

- *Συνέπεια σε όλες τις πλατφόρμες:* Το Ionic React επιτρέπει την απρόσκοπτη λειτουργία της ίδιας διεπαφής χρήστη σε πολλαπλές πλατφόρμες, χάρη στις αρχές του responsive web design, το CSS και τα βοηθητικά προγράμματα εντοπισμού πλατφόρμας. Οι προγραμματιστές μπορούν να δημιουργήσουν ένα ενιαίο UI που προσαρμόζεται έξυπνα σε διαφορετικές συσκευές και μεγέθη οθόνης.
- *Αποδοτικότητα και επαναχρησιμοποίηση κώδικα:* Η προσέγγιση "write once, run anywhere" απλοποιεί την ανάπτυξη, επιτρέποντας στους προγραμματιστές να διατηρούν μια ενιαία βάση κώδικα για την εφαρμογή τους, ελαχιστοποιώντας τον πλεονασμό και προωθώντας την επαναχρησιμοποίηση του κώδικα.
- *Ευελιξία προσαρμογής:* Ενώ το Ionic React παρέχει μια συνεπή εμπειρία UI σε όλες τις πλατφόρμες, προσφέρει επίσης την ευελιξία στους προγραμματιστές να προσαρμόσουν την εφαρμογή για συγκεκριμένες πλατφόρμες, αν το επιθυμούν. Αυτή η δυνατότητα προσαρμογής επιτυγχάνεται μέσω των αρχών του responsive design και του styling για συγκεκριμένες πλατφόρμες.
- *Προσαρμοστικό styling:* Το Ionic React χρησιμοποιεί μια έννοια γνωστή ως Adaptive Styling, όπου τα βασικά στοιχεία του UI, όπως η πλοήγηση, οι καρτέλες, οι γραμμές εργαλείων και τα κουμπιά, αντιστοιχίζονται έξυπνα στις προσδοκίες της πλατφόρμας (π.χ. Material Design), διατηρώντας παράλληλα τη δυνατότητα των σχεδιαστών να προσαρμόζουν την εμφάνιση και την αίσθηση.

Σε αντίθεση με το React Native, όπου οι προγραμματιστές πρέπει να δημιουργήσουν ξεχωριστές οθόνες για iOS και Android, το Ionic React βελτιώνει τη διαδικασία ανάπτυξης

προωθώντας την επαναχρησιμοποίηση του κώδικα και τη συνοχή σε όλες τις πλατφόρμες. Αυτή η προσέγγιση ευθυγραμμίζεται με τη φιλοσοφία "γράψτε μία φορά, εκτελέστε οπουδήποτε", επιτρέποντας στις ομάδες ανάπτυξης να στοχεύουν αποτελεσματικά σε ένα ευρύτερο φάσμα πλατφορμών με μια ενοποιημένη βάση κώδικα.

2.2.4 Ομοιότητα στην πρόσβαση και τη λειτουργικότητα

Παρά τις διαφορετικές τους προσεγγίσεις, το Ionic React και το React Native μοιράζονται μια κρίσιμη ομοιότητα στην παροχή πλήρους εγγενούς πρόσβασης και δυνατοτήτων, επιτρέποντας στους προγραμματιστές να αξιοποιήσουν τη δύναμη των εγγενών χαρακτηριστικών και APIs για τις εφαρμογές τους.

Πλήρης εγγενής πρόσβαση και στα δύο frameworks:

Τόσο το Ionic React όσο και το React Native υπερέχουν στο να προσφέρουν στους προγραμματιστές πρόσβαση σε εγγενείς δυνατότητες και API συσκευών. Αυτή η πρόσβαση αποτελεί ακρογωνιαίο λίθο του σχεδιασμού τους, διασφαλίζοντας ότι οι εφαρμογές που κατασκευάζονται με αυτά τα frameworks μπορούν να παρέχουν μια πλούσια και καθηλωτική εγγενή εμπειρία. Ακολουθούν ορισμένα βασικά σημεία που αναδεικνύουν τις εγγενείς δυνατότητές τους:

- *Μητρική ενσωμάτωση:* Τόσο το Ionic React όσο και το React Native επιτρέπουν τη δημιουργία γνήσιων εγγενών έργων και αρχείων εφαρμογών. Αυτό σημαίνει ότι οι εφαρμογές που προκύπτουν είναι πραγματικές εγγενείς εφαρμογές και όχι απλές διαδικτυακές ή υβριδικές λύσεις.
- *Πρόσβαση σε εγγενή APIs:* Οι προγραμματιστές που χρησιμοποιούν οποιοδήποτε από τα δύο framework έχουν τη δυνατότητα να αλληλεπιδρούν με τα εγγενή χαρακτηριστικά και τις λειτουργίες των συσκευών. Αυτό περιλαμβάνει την πρόσβαση σε αισθητήρες της συσκευής, κάμερες, γεωεντοπισμό και πολλά άλλα. Τα frameworks παρέχουν ένα επίπεδο που επιτρέπει στον κώδικα JavaScript να επικοινωνεί με τον υποκείμενο εγγενή κώδικα, καθιστώντας δυνατή την απρόσκοπτη αξιοποίηση των εγγενών δυνατοτήτων.
- *Εκτέλεση εγγενούς κώδικα:* Τόσο το Ionic React όσο και το React Native εκτελούν JavaScript κατά την εκτέλεση, σε αντίθεση με την προ-μεταγλώττιση σε εγγενή κώδικα. Αυτή η εκτέλεση κατά τον χρόνο εκτέλεσης επιτρέπει στους προγραμματιστές να ενσωματώνουν προσαρμοσμένο εγγενή κώδικα όταν είναι απαραίτητο για να επεκτείνουν τη λειτουργικότητα των εφαρμογών τους.
- *Ενσωμάτωση εγγενούς UI:* Παρόλο που και τα δύο πλαίσια διαθέτουν στοιχεία βασισμένα στον ιστό, επιτρέπουν την ενσωμάτωση στοιχείων και στοιχείων ελέγχου εγγενούς περιβάλλοντος εργασίας όταν χρειάζεται. Αυτά τα στοιχεία συνδυάζονται άψογα με την ιεραρχία της διεπαφής χρήστη της εφαρμογής, εξασφαλίζοντας μια συνεκτική εμπειρία χρήστη.

2.2.4.1 Αποσαφήνιση παρανοήσεων:

Είναι σημαντικό να αντιμετωπιστούν οι συνήθεις παρανοήσεις γύρω από τις εφαρμογές για κινητά που βασίζονται στον ιστό και τη διαδικασία σύνταξης:

- *Εφαρμογές κινητών τηλεφώνων με βάση τον ιστό:* Οι εφαρμογές κινητών τηλεφώνων που βασίζονται στον ιστό και αναπτύσσονται με το Ionic React αξιοποιούν μεν τα

στοιχεία ελέγχου WebView για την απόδοση, αλλά αποτελούν αναπόσπαστο μέρος της ιεραρχίας του περιβάλλοντος εργασίας χρήστη της πραγματικής εγγενούς εφαρμογής. Αυτή η διάκριση τους επιτρέπει να έχουν πρόσβαση σε όλα τα εγγενή API και να εκτελούν εγγενή κώδικα, αν και όχι απευθείας μέσα στο WebView.

- *Μεταγλώττιση React Native:* Σε αντίθεση με τη δημοφιλή πεποίθηση, οι εφαρμογές React Native δεν "μεταγλωττίζονται σε native". Τόσο το Ionic React (μέσω Capacitor ή Cordova) όσο και το React Native εκτελούν JavaScript κατά την εκτέλεση. Ωστόσο, το Ionic React αξιοποιεί την προηγμένη μεταγλώττιση Just In Time (JIT) για τη JavaScript στα περισσότερα περιβάλλοντα WebView, προσφέροντας εγγενή πλεονεκτήματα απόδοσης που δεν είναι διαθέσιμα στο React Native, το οποίο δεν υποστηρίζει JIT στο JavaScriptCore.

2.2.4.2 Διαφορετικές προσεγγίσεις για την εγγενή λειτουργικότητα:

Το Ionic React χρησιμοποιεί το Capacitor, ένα έργο που αναπτύχθηκε από την Ionic, για να διευκολύνει την κλήση εγγενούς κώδικα απευθείας από την JavaScript. Το Capacitor απλοποιεί τη διαδικασία έκθεσης νέων εγγενών λειτουργιών και ενσωμάτωσης εγγενών στοιχείων UI. Ειδικότερα, η πρώτης τάξεως υποστήριξη του Capacitor για το Swift ενισχύει την ευκολία δημιουργίας εγγενών λειτουργιών για εφαρμογές iOS.

Το React Native, από την άλλη πλευρά, χρησιμοποιεί τη δική του προσέγγιση για την ενσωμάτωση εγγενών λειτουργιών, εστιάζοντας στη γεφύρωση του κώδικα JavaScript με εγγενείς ενότητες για την πρόσβαση σε εγγενή λειτουργικότητα. Αν και η αρχιτεκτονική διαφέρει από το Capacitor, δίνει εξίσου τη δυνατότητα στους προγραμματιστές να επεκτείνουν απρόσκοπτα τις εγγενείς δυνατότητες μιας εφαρμογής.

Συνοψίζοντας, το Ionic React και το React Native προσφέρουν σχεδόν την ίδια υποστήριξη για την εκτέλεση εγγενή κώδικα και την πρόσβαση σε εγγενή API, παρά τις διαφορετικές μεθοδολογίες τους. Οι προγραμματιστές έχουν την ευελιξία να επιλέξουν το framework που ευθυγραμμίζεται καλύτερα με τις προτιμήσεις ανάπτυξης και τις απαιτήσεις του έργου τους, γνωρίζοντας ότι και τα δύο frameworks είναι ικανά να παρέχουν μια ισχυρή native εμπειρία.

2.2.5 Διαφορά στην υποστήριξη και στη λειτουργικότητα για οργανισμούς

Το Ionic React και το React Native αποκλίνουν σημαντικά όσον αφορά την υποστήριξη επιχειρήσεων και το επίπεδο δέσμευσης για την εξυπηρέτηση των αναγκών των επιχειρήσεων και των οργανισμών.

2.2.5.1 Εστίαση στις επιχειρήσεις του Ionic React:

Το Ionic React ξεχωρίζει στο οικοσύστημα ανάπτυξης εφαρμογών κινητής τηλεφωνίας, καθώς υποστηρίζεται ισχυρά από μια ειδική επιχειρηματική οντότητα με κεντρική εστίαση στην παροχή βοήθειας σε ομάδες για την κατασκευή και κλιμάκωση κρίσιμων εφαρμογών. Η Ionic έχει δεσμευτεί να εξυπηρετεί τις επιχειρήσεις και έχει δημιουργήσει τη φήμη ενός αξιόπιστου συνεργάτη για τις επιχειρήσεις. Ορισμένες βασικές πτυχές που αναδεικνύουν τον επιχειρηματικό προσανατολισμό του Ionic React περιλαμβάνουν:

- *Εμπορική υποστήριξη:* Το Ionic δεν είναι απλώς ένα έργο ανοικτού κώδικα- υποστηρίζεται από μια επιχειρηματική οντότητα που έχει ως στόχο την εξυπηρέτηση της αγοράς των επιχειρήσεων. Αυτή η εμπορική υποστήριξη είναι καθοριστική για την παροχή ολοκληρωμένης υποστήριξης και πόρων για τους πελάτες επιχειρήσεων και οργανισμών.
- *Εκτεταμένη υιοθέτηση από επιχειρήσεις:* Η Ionic συνεργάζεται ενεργά με πολυάριθμες μεσαίες και μεγάλες επιχειρήσεις και οργανισμούς , βοηθώντας τους στη δημιουργία εφαρμογών κινητών και διαδικτυακών εφαρμογών που απευθύνονται τόσο στον καταναλωτή όσο και στους εργαζόμενους. Αυτές οι εφαρμογές συχνά φτάνουν σε εκατομμύρια χρήστες και παράγουν σημαντικά έσοδα. Η σημαντική εμπειρία της Ionic στη συνεργασία με επιχειρήσεις υπογραμμίζει την καταλληλότητά της για έργα κρίσιμης σημασίας.
- *Αφιερωμένοι εμπειρογνώμονες εφαρμογών:* Το Ionic προσφέρει έναν πολύτιμο πόρο με τη μορφή εξειδικευμένων εμπειρογνωμόνων εφαρμογών, οι οποίοι είναι διαθέσιμοι για να βοηθήσουν τις ομάδες των επιχειρήσεων καθ' όλη τη διάρκεια της διαδικασίας ανάπτυξης. Αυτός ο πόρος διασφαλίζει ότι οι επιχειρήσεις έχουν πρόσβαση σε έγκαιρη καθοδήγηση και υποστήριξη κατά την πλοήγηση στις πολύπλοκες προκλήσεις ανάπτυξης εφαρμογών.
- *Επιχειρησιακή εγγενής λειτουργικότητα:* Το Ionic δίνει προτεραιότητα στις απαιτήσεις των επιχειρήσεων και προσφέρει βασική εγγενή λειτουργικότητα προσαρμοσμένη στις ανάγκες των εταιρικών πελατών. Αυτή η λειτουργικότητα περιλαμβάνει ισχυρή υποστήριξη για ασφαλή έλεγχο ταυτότητας και διαχείριση ταυτότητας, αξιοποιώντας προηγμένα API κρυπτογράφησης που είναι διαθέσιμα τόσο στο iOS όσο και στο Android. Επιπλέον, το Ionic παρέχει υψηλής απόδοσης κρυπτογραφημένη αποθήκευση δεδομένων χωρίς σύνδεση, καλύπτοντας τις ανάγκες ασφάλειας δεδομένων των επιχειρήσεων.
- *Δυνατότητες DevOps για κινητά:* Το προϊόν Appflow της Ionic προσφέρει μια σειρά χαρακτηριστικών Mobile DevOps, δίνοντας τη δυνατότητα στις ομάδες να βελτιστοποιήσουν τις διαδικασίες ανάπτυξης και ανάπτυξης εφαρμογών. Αυτό περιλαμβάνει τη δυνατότητα να αναπτύσσουν ενημερώσεις εφαρμογών σε πραγματικό χρόνο και να ενσωματώνουν απρόσκοπτα τη δυαδική κατασκευή εγγενών εφαρμογών στις ροές εργασίας τους για συνεχή ολοκλήρωση και συνεχή ανάπτυξη (CI/CD). Στο εγγύς μέλλον, το Ionic σχεδιάζει να αυτοματοποιήσει τις υποβολές σε καταστήματα εφαρμογών ως μέρος της ροής εργασιών ανάπτυξης, ενισχύοντας περαιτέρω την αποτελεσματικότητα της ανάπτυξης εφαρμογών σε επιχειρήσεις.

2.2.5.2 Προέλευση και εστίαση της React Native:

Το React Native ξεκίνησε ως ένα εσωτερικό έργο στο Facebook, με κύριο στόχο τη βελτίωση των διαδικασιών ανάπτυξης εφαρμογών του ίδιου του Facebook. Ενώ έχει εξελιχθεί σε ένα ευρέως χρησιμοποιούμενο πλαίσιο ανοικτού κώδικα με μια ακμάζουσα κοινότητα, το React Native δεν έχει μια εμπορική ή επιχειρηματική επιχειρηματική οντότητα άμεσα πίσω του. Οι προτεραιότητες ανάπτυξής του ενδέχεται να μην ευθυγραμμίζονται τόσο στενά με τις συγκεκριμένες ανάγκες και προκλήσεις που αντιμετωπίζουν οι επιχειρήσεις και οι οργανισμοί.

Συνοψίζοντας, η μοναδική τοποθέτηση του Ionic React ως πλαίσιο με επίκεντρο τις επιχειρήσεις, που υποστηρίζεται από μια εξειδικευμένη επιχείρηση, το διακρίνει από το React Native όσον αφορά τη δέσμευσή του να εξυπηρετεί την αγορά των επιχειρήσεων. Η ολοκληρωμένη υποστήριξη του Ionic, η προσαρμοσμένη εγγενής λειτουργικότητα και τα χαρακτηριστικά Mobile DevOps το τοποθετούν ως μια επιτακτική επιλογή για οργανισμούς που επιδιώκουν να αναπτύξουν σημαντικές εφαρμογές που απαιτούν υψηλό επίπεδο υποστήριξης και λειτουργικότητας για επιχειρήσεις και οργανισμούς.

2.2.6 Συμπεράσματα

Στην απόφαση μεταξύ Ionic React και React Native για την ανάπτυξη εφαρμογών για κινητά, είναι σημαντικό να λάβουμε υπόψη τις συγκεκριμένες ανάγκες και τους στόχους του έργου μας, καθώς και τα δυνατά σημεία και τα χαρακτηριστικά του κάθε framework. Τόσο το Ionic React όσο και το React Native προσφέρουν αξιόλογες λύσεις, αλλά απευθύνονται σε διαφορετικά σενάρια ανάπτυξης και ομάδες με διαφορετικές δεξιότητες και προτεραιότητες.

Επιλογή React Native όταν:

- Η πρωταρχική εστίαση είναι η αποκλειστική στόχευση στις πλατφόρμες iOS και Android.
- Η ομάδα αποτελείται από παραδοσιακούς προγραμματιστές native ή έμπειρους προγραμματιστές JavaScript οι οποίοι είναι άνετοι με τις επιπλοκές της React Native.
- Υπάρχει ένα υπάρχον αποθετήριο εγγενών στοιχείων ελέγχου και συστατικών που σκοπεύουμε να αξιοποιήσουμε στην εφαρμογή μας.
- Το έργο είναι προσανατολισμένο στον καταναλωτή και ευθυγραμμίζεται με την κουλτούρα των startups για την ανάπτυξη εγγενών εφαρμογών.

Επιλογή Ionic React όταν:

- Διαθέτουμε παραδοσιακές δεξιότητες και βιβλιοθήκες ανάπτυξης ιστοσελίδων και στοχεύουμε τόσο σε πλατφόρμες κινητών τηλεφώνων όσο και σε πλατφόρμες ιστού (συμπεριλαμβανομένων των Progressive Web Apps).
- Η ομάδα εργασίας εκτιμά την επαναχρησιμοποίηση του κώδικα και τη συνοχή σε όλες τις πλατφόρμες, δίνοντας έμφαση στην προσέγγιση "write once, run anywhere".
- Υπάρχει ιστορικό ανάπτυξης ιστοσελίδων και εφαρμογών ιστού και επιδιώκουμε να επεκτείνουμε την τεχνογνωσία μας στην ανάπτυξη εφαρμογών για κινητά.
- Το έργο μας περιλαμβάνει τόσο εφαρμογές που απευθύνονται σε καταναλωτές όσο και σε εργαζόμενους.

Επιχειρήσεις και εφαρμογές κρίσιμης σημασίας:

Το Ionic React ξεχωρίζει ως μια ισχυρή επιλογή για επιχειρήσεις και εφαρμογές κρίσιμης σημασίας λόγω της εμπορικής του υποστήριξης, της λειτουργικότητας ειδικά για επιχειρήσεις και της εξειδικευμένης υποστήριξης. Εάν η εγγυημένη υποστήριξη και η διαθεσιμότητα λειτουργιών επιχειρηματικού επιπέδου είναι υψίστης σημασίας για το έργο μας, το Ionic React μπορεί να είναι η καταλληλότερη επιλογή. Η εστίαση της Ionic στην επιχειρηματική τεχνολογία ανάπτυξης εφαρμογών τη διακρίνει ως μία από τις λίγες εταιρείες ανάπτυξης εφαρμογών πλατφόρμας που είναι αφιερωμένες στην εξυπηρέτηση της αγοράς των επιχειρήσεων.

3 Φάση Pre-Migration

3.1 Η σημασία της Pre - Migration φάσης

Η μετατροπή μιας εφαρμογής ιστού React στο Ionic framework είναι μια πολύπλοκη διαδικασία που απαιτεί σχολαστικό σχεδιασμό και εκτέλεση. Η συγκεκριμένη φάση παίζει καθοριστικό ρόλο στη διασφάλιση της επιτυχίας αυτής της μετάβασης.

Η φάση προ-μετάβασης χρησιμεύει ως το αρχικό βήμα στο ταξίδι της μετάβασης μιας διαδικτυακής εφαρμογής React στο Ionic framework. Η φάση αυτή περιλαμβάνει ενδελεχή αξιολόγηση, σχεδιασμό και προετοιμασία για τη διευκόλυνση μιας απρόσκοπτης μετάβασης. Ακολουθούν οι ιδιαιτερότητες που συναντώνται σε αυτή τη φάσης της προ-μετάβασης, τονίζοντας τον κρίσιμο ρόλο της στην επίτευξη μιας επιτυχημένης μετάβασης.

Η φάση προ-μετάβασης χαρακτηρίζεται από διάφορα βασικά στοιχεία που συμβάλλουν συλλογικά στη σημασία της στη διαδικασία αυτής της μετάβασης. Αυτές οι συνιστώσες περιλαμβάνουν την αξιολόγηση της βάσης κώδικα (codebase), την αξιολόγηση της λειτουργικότητας και του UI, την αξιολόγηση των βιβλιοθηκών και των εξαρτήσεων της εφαρμογής με τρίτους, αλλά και στη συγκέντρωση πόρων.

3.1.1 Στοιχεία της φάσης προ-μετάβασης:

- **Αξιολόγηση της βάσης κώδικα:** Σε αυτό το στοιχείο, η ομάδα ανάπτυξης αναλύει την υπάρχουσα βάση κώδικα της εφαρμογής React. Στόχος είναι να αποκτήσει μια ολοκληρωμένη κατανόηση της δομής, της οργάνωσης και των εξαρτήσεων της βάσης κώδικα. Αυτή η αξιολόγηση είναι ζωτικής σημασίας για τον εντοπισμό πιθανών προκλήσεων που μπορεί να προκύψουν κατά τη διαδικασία μετάβασης, όπως ασύμβατα στοιχεία React, βιβλιοθήκες ή αρχιτεκτονικά ζητήματα.
- **Αξιολόγηση της λειτουργικότητας και της διεπαφής χρήστη:** Η αξιολόγηση της λειτουργικότητας και των στοιχείων του UI επικεντρώνεται στην αξιολόγηση του τρόπου με τον οποίο τα χαρακτηριστικά της εφαρμογής και τα στοιχεία διεπαφής χρήστη θα προσαρμοστούν στο περιβάλλον του πλαισίου Ionic που είναι προσανατολισμένο στις κινητές συσκευές. Αυτή η φάση περιλαμβάνει μια προσεκτική εξέταση της εμπειρίας χρήστη της εφαρμογής και τον προσδιορισμό του κατά πόσον ορισμένα χαρακτηριστικά πρέπει να επανασχεδιαστούν ή να υλοποιηθούν εκ νέου για τη βελτιστοποίηση της εμπειρίας χρήστη για κινητά τηλέφωνα.
- **Βιβλιοθήκες, dependencies και πακέτα τρίτων:** Κατά τη διάρκεια αυτής της φάσης, η ομάδα ανάπτυξης εξετάζει εξονυχιστικά τη συμβατότητα των βιβλιοθηκών και των εξαρτήσεων τρίτων που χρησιμοποιούνται στην εφαρμογή React με το Ionic framework. Λαμβάνονται αποφάσεις σχετικά με το αν αυτές οι εξαρτήσεις πρέπει να αντικατασταθούν, να αναδιαμορφωθούν ή να προσαρμοστούν ώστε να ευθυγραμμιστούν με τις απαιτήσεις του Ionic. Τα ζητήματα συμβατότητας αντιμετωπίζονται προληπτικά για την αποφυγή επιπλοκών κατά τη διάρκεια της μετάβασης.
- **Συγκέντρωση πόρων:** Η συγκέντρωση πόρων εξασφαλίζει ότι η ομάδα ανάπτυξης είναι καλά εξοπλισμένη για να ξεκινήσει αποτελεσματικά τη διαδικασία μετάβασης. Περιλαμβάνει την απόκτηση βασικών εργαλείων και πόρων, όπως το Ionic CLI, τα plugins και τη σχετική τεκμηρίωση. Αυτό το βήμα είναι ζωτικής σημασίας για τον

εξορθολογισμό της διαδικασίας μετάβασης και τη διασφάλιση ότι η ομάδα έχει πρόσβαση στους απαραίτητους πόρους για την υποστήριξη της μετάβασης.

3.1.2 Σημασία της φάσης προ-μετάβασης:

- **Μετριασμός των κινδύνων:** Ένα από τα κύρια πλεονεκτήματα της φάσης πριν από τη μετεγκατάσταση είναι ο ρόλος της στον μετριασμό των κινδύνων. Με τον εντοπισμό πιθανών προκλήσεων και εμποδίων σε πρώιμο στάδιο της διαδικασίας, η ομάδα ανάπτυξης μπορεί να σχεδιάσει στρατηγικές για την επίλυση αυτών των ζητημάτων. Αυτή η προληπτική προσέγγιση μειώνει τον χρόνο διακοπής λειτουργίας και ελαχιστοποιεί τις απρόβλεπτες επιπλοκές κατά τη διάρκεια της μετάβασης, συμβάλλοντας τελικά σε μια ομαλότερη μετάβαση.
- **Αποδοτικότητα ως προς το κόστος και τον χρόνο:** Η αποδοτικότητα τόσο από άποψη κόστους όσο και από άποψη χρόνου αποτελεί βασικό πλεονέκτημα της φάσης πριν από τη μετάβαση. Ο κατάλληλος σχεδιασμός και η προετοιμασία οδηγούν σε σημαντική εξοικονόμηση κόστους και χρόνου κατά τη διαδικασία μετάβασης. Η εκ των προτέρων αντιμετώπιση θεμάτων συμβατότητας και αρχιτεκτονικών προβληματισμών μειώνει την ανάγκη για εκτεταμένη ανακατασκευή κατά τη διάρκεια της μετάβασης, ελαχιστοποιώντας τις διακοπές λειτουργίες και τα έξοδα που ενδέχεται να προκύψουν..
- **Διασφάλιση ποιότητας:** Η φάση προ-μετάβασης διασφαλίζει ότι η προκύπτουσα εφαρμογή Ionic διατηρεί τη βασική λειτουργικότητα και την εμπειρία χρήστη της αρχικής εφαρμογής ιστού React. Επιτρέπει στην ομάδα ανάπτυξης να δημιουργήσει μια πιο ποιοτική εφαρμογή Ionic για κινητά που ανταποκρίνεται ή υπερβαίνει τις προσδοκίες των χρηστών. Η διασφάλιση της ποιότητας βρίσκεται στο επίκεντρο αυτής της φάσης, δίνοντας έμφαση στη σημασία της παροχής μιας απρόσκοπτης και φιλικής προς τον χρήστη εμπειρίας για κινητά.

Συμπερασματικά, η φάση προ-μετάβασης αποτελεί απαραίτητο συστατικό της διαδικασίας μετάβασης από μια εφαρμογή React σε Ionic. Δίνει τη δυνατότητα στις ομάδες ανάπτυξης να αξιολογούν, να σχεδιάζουν και να προετοιμάζονται για μια επιτυχημένη μετάβαση, εντοπίζοντας πιθανές προκλήσεις, μειώνοντας τους κινδύνους και βελτιστοποιώντας την αποτελεσματικότητα της διαδικασίας μετάβασης. Αναδεικνύεται ο κρίσιμος ρόλος της φάσης της προ-μετάβασης στη διασφάλιση της ομαλής μετάβασης μιας διαδικτυακής εφαρμογής React στο Ionic framework, που τελικά οδηγεί στην ανάπτυξη μιας υψηλής ποιότητας εφαρμογής για κινητά.

3.1.3 Ζητήματα σχεδιασμού και εμπειρίας χρήστη (UX)

Κατά τη φάση πριν από τη μετάβαση, κατά τη μετάβαση μιας διαδικτυακής εφαρμογής React στο Ionic framework, είναι σημαντικό να ληφθούν υπόψη ζητήματα σχεδιασμού και εμπειρίας χρήστη (UX). Ο σχεδιασμός και η UX παίζουν καθοριστικό ρόλο στην επιτυχία κάθε εφαρμογής για κινητά και η ευθυγράμμιση με τις προσδοκίες των χρηστών σε αυτούς τους τομείς είναι υψίστης σημασίας. Ακολουθούν διάφοροι βασικοί παράγοντες που πρέπει να ληφθούν υπόψη:

- **Σχεδιασμός πρώτα για κινητά τηλέφωνα:** Οι χρήστες κινητών τηλεφώνων αναμένουν οι εφαρμογές να σχεδιάζονται με γνώμονα τις συγκεκριμένες ανάγκες τους. Στη φάση πριν από τη μετάβαση, είναι ζωτικής σημασίας η υιοθέτηση μιας προσέγγισης σχεδιασμού πρώτα για κινητά. Αυτό σημαίνει ότι δίνεται προτεραιότητα

στο σχεδιασμό για μικρότερες οθόνες και αλληλεπιδράσεις αφής, διασφαλίζοντας ότι τα στοιχεία του UI έχουν το κατάλληλο μέγεθος, απόσταση και είναι φιλικά προς την αφή. Το framework του Ionic είναι εγγενώς προσανατολισμένο προς τον σχεδιασμό mobile-first, καθιστώντας το ιδανική επιλογή για την ικανοποίηση αυτής της προσδοκίας.

- **Συνέπεια στον οπτικό σχεδιασμό:**Οι χρήστες εκτιμούν έναν συνεπή και οπτικά ελκυστικό σχεδιασμό σε όλη την εμπειρία της εφαρμογής τους. Κατά τη φάση πριν από τη μετάβαση, αξιολογούμε τον οπτικό σχεδιασμό και τα στοιχεία branding της υπάρχουσας εφαρμογής ιστού React. Βεβαιωνόμαστε ότι αυτά τα στοιχεία ενσωματώνονται απρόσκοπτα στην εφαρμογή Ionic. Η συνέπεια στα χρώματα, την τυπογραφία, τα εικονίδια και το συνολικό οπτικό στυλ συμβάλλει στη διατήρηση της εξοικείωσης των χρηστών και ενισχύει την ταυτότητα της μάρκας.
- **Responsive Layouts:**Οι ανταποκρινόμενες διατάξεις είναι μια θεμελιώδης θεώρηση UX. Οι χρήστες κινητών τηλεφώνων εναλλάσσονται συχνά μεταξύ κατακόρυφου και οριζόντιου προσανατολισμού και η εφαρμογή θα πρέπει να προσαρμόζεται ανάλογα. Στη φάση πριν από τη μετάβαση, επανεξετάζουμε και τροποποιούμε τα στοιχεία UI για να διασφαλίσουμε ότι προσαρμόζονται με επιτυχία σε διαφορετικά μεγέθη και προσανατολισμούς οθόνης. Το σύστημα πλέγματος (grid layout) του Ionic και οι αρχές responsive design μπορούν να βοηθήσουν στην επίτευξη responsive διατάξεων.
- **Αλληλεπιδράσεις που βασίζονται σε χειρονομίες:**Οι κινητές συσκευές βασίζονται σε μεγάλο βαθμό στις χειρονομίες για την πλοήγηση και την αλληλεπίδραση. Οι χρήστες αναμένουν διαισθητικές χειρονομίες swipe, pinch-to-zoom και άλλες αλληλεπιδράσεις αφής. Κατά τη φάση πριν από τη μετάβαση, αξιολογούμε τα μοτίβα αλληλεπίδρασης της εφαρμογής React και ενσωματώνουμε αλληλεπιδράσεις που βασίζονται σε χειρονομίες, όπου είναι απαραίτητο. Το Ionic παρέχει ενσωματωμένη υποστήριξη για χειρονομίες και συμβάντα αφής, απλοποιώντας την υλοποίηση αυτών των χαρακτηριστικών.
- **Κινούμενα σχέδια και ανατροφοδότηση:** Οι καλά εκτελεσμένες κινούμενες εικόνες και οι μηχανισμοί ανατροφοδότησης βελτιώνουν τη συνολική εμπειρία του χρήστη. Οι χρήστες εκτιμούν τις λεπτές κινούμενες εικόνες για τις μεταβάσεις μεταξύ των οθονών, την οπτική ανατροφοδότηση για το πάτημα κουμπιών και τους δείκτες φόρτωσης. Στη φάση πριν από τη μετάβαση, εξετάζουμε πώς μπορούν να ενσωματωθούν οι κινούμενες εικόνες και η ανατροφοδότηση στην εφαρμογή Ionic για να γίνουν οι αλληλεπιδράσεις πιο ελκυστικές και ενημερωτικές.
- **Προσβασιμότητα:** Η προσβασιμότητα είναι μια κρίσιμη πτυχή του UX, διασφαλίζοντας ότι η εφαρμογή είναι χρησιμοποιήσιμη από όλα τα άτομα, συμπεριλαμβανομένων των ατόμων με αναπηρίες. Κατά τη φάση πριν από τη μετάβαση, αξιολογούμε τα χαρακτηριστικά προσβασιμότητας της εφαρμογής React και επιδιώκουμε να τα διατηρήσουμε ή να τα βελτιώσετε στην εφαρμογή Ionic. Το Ionic παρέχει φιλικά προς την προσβασιμότητα στοιχεία και κατευθυντήριες γραμμές που βοηθούν στη δημιουργία εμπειριών χωρίς αποκλεισμούς για όλους τους χρήστες.
- **Περιεχόμενο με επίκεντρο τον χρήστη:** Η προσαρμογή του περιεχομένου στις ανάγκες των κινητών χρηστών είναι απαραίτητη. Στη φάση πριν από τη μετάβαση, επανεξετάζουμε το περιεχόμενο και τη διάταξη της εφαρμογής React και δίνουμε προτεραιότητα σε περιεχόμενο με επίκεντρο το κινητό. Το περιεχόμενο θα πρέπει να είναι συνοπτικό, σχετικό και να παρουσιάζεται με τρόπο φιλικό προς τον χρήστη, ώστε να προσαρμόζεται σε μικρότερες οθόνες. Εξετάζουμε τον τρόπο ροής και οργάνωσης

του περιεχομένου εντός της εφαρμογής Ionic για να ενισχύσουμε την αναγνωσιμότητα και τη δέσμευση.

- **Απόδοση και χρόνοι φόρτωσης:**Οι χρόνοι φόρτωσης επηρεάζουν σημαντικά την ικανοποίηση των χρηστών. Οι χρήστες αναμένουν γρήγορη πρόσβαση στο περιεχόμενο και τις λειτουργίες της εφαρμογής. Στη φάση πριν από τη μετάβαση, βελτιστοποιούνται στοιχεία όπως εικόνες και βίντεο για κινητές συσκευές, ελαχιστοποιούνται οι περιττές μεταφορές δεδομένων και χρησιμοποιούμε τεχνικές lazy loading. Τα χαρακτηριστικά βελτιστοποίησης επιδόσεων του Ionic μπορούν να βοηθήσουν στην επίτευξη γρήγορων χρόνων φόρτωσης.
- **Κατευθυντήριες γραμμές πλατφόρμας:**Το iOS και το Android έχουν ξεχωριστές κατευθυντήριες γραμμές σχεδιασμού και συμβάσεις διεπαφής χρήστη. Για να ικανοποιήσουμε τις προσδοκίες των χρηστών, τηρούνται αυτές οι ειδικές για την πλατφόρμα κατευθυντήριες γραμμές κατά τη φάση πριν από τη μετάβαση. Το Ionic παρέχει εργαλεία και στοιχεία που ευθυγραμμίζονται με τις συμβάσεις της πλατφόρμας, διευκολύνοντας τη δημιουργία μιας εγγενούς εμφάνισης για τους χρήστες τόσο του iOS όσο και του Android.

Συμπερασματικά, η εστίαση σε ζητήματα σχεδιασμού και εμπειρίας χρήστη κατά τη φάση πριν από τη μετάβαση είναι απαραίτητη για να διασφαλιστεί ότι η εφαρμογή Ionic ανταποκρίνεται και υπερβαίνει τις προσδοκίες των χρηστών. Με την υιοθέτηση μιας προσέγγισης με γνώμονα το κινητό πρώτα, τη διατήρηση της συνοχής του σχεδιασμού, τη δημιουργία ευέλικτων διατάξεων, την ενσωμάτωση αλληλεπιδράσεων που βασίζονται σε χειρονομίες, την παροχή ουσιαστικών κινούμενων εικόνων και ανατροφοδότησης, την ιεράρχηση της προσβασιμότητας, τη βελτιστοποίηση των επιδόσεων, την τελειοποίηση του περιεχομένου, τη διεξαγωγή δοκιμών ευχρηστίας και την τήρηση των κατευθυντήριων γραμμών της πλατφόρμας, οι ομάδες ανάπτυξης μπορούν να παραδώσουν μια εφαρμογή για κινητά που προσφέρει εξαιρετική εμπειρία χρήσης και έχει απήχηση στο κοινό των κινητών συσκευών.

3.1.4 Προσδοκίες χρηστών

Κατά τη φάση πριν από τη μετάβαση μιας διαδικτυακής εφαρμογής React στο Ionic Framework, είναι υψίστης σημασίας να ληφθούν υπόψη και να ευθυγραμμιστούν με τις προσδοκίες των χρηστών για τη λειτουργικότητα και την απόδοση των κινητών συσκευών. Οι χρήστες κινητών τηλεφώνων έχουν συγκεκριμένες απαιτήσεις και πρότυπα όσον αφορά τις εφαρμογές με τις οποίες αλληλεπιδρούν στα smartphones και τα tablet τους. Η αναγνώριση και η αντιμετώπιση αυτών των προσδοκιών από νωρίς στη διαδικασία μετάβασης είναι απαραίτητη για τη δημιουργία μιας επιτυχημένης εφαρμογής Ionic για κινητά. Ακολουθούν διάφοροι βασικοί παράγοντες που πρέπει να λάβουμε υπόψη:

- **Ανταπόκριση:** Οι χρήστες κινητών τηλεφώνων αναμένουν οι εφαρμογές να ανταποκρίνονται και να είναι ‘ρευστές’. Αναμένουν ομαλές μεταβάσεις μεταξύ των οθονών, γρήγορους χρόνους φόρτωσης και ελάχιστη καθυστέρηση κατά την αλληλεπίδραση με την εφαρμογή. Στη φάση πριν από τη μετεγκατάσταση, είναι επιτακτική ανάγκη να αξιολογηθεί η ανταπόκριση της εφαρμογής ιστού React και να εντοπιστούν οι περιοχές που ενδέχεται να χρειάζονται βελτιστοποίηση για ένα περιβάλλον κινητής τηλεφωνίας. Το Ionic Framework παρέχει εργαλεία και κατευθυντήριες γραμμές για την επίτευξη responsive design, και αυτά θα πρέπει να αξιοποιηθούν για την ικανοποίηση των προσδοκιών των χρηστών.

- **UI φιλικό προς την αφή:**Οι κινητές συσκευές βασίζονται σε μεγάλο βαθμό στις αλληλεπιδράσεις αφής. Οι χρήστες αναμένουν ότι τα κουμπιά, τα μενού και τα διαδραστικά στοιχεία πρέπει να σχεδιάζονται με γνώμονα την αφή. Κατά τη φάση πριν από τη μετάβαση, τα στοιχεία του UI της εφαρμογής React θα πρέπει να αξιολογηθούν και, εάν είναι απαραίτητο, να επανασχεδιαστούν ή να προσαρμοστούν ώστε να προσαρμόζονται στις εισόδους αφής. Η διασφάλιση ότι τα στοιχεία έχουν το κατάλληλο μέγεθος και απόσταση για χειρονομίες αφής θα βελτιώσει τη συνολική εμπειρία του χρήστη.
- **Βελτιστοποίηση επιδόσεων:**Η απόδοση είναι μια κρίσιμη πτυχή της ανάπτυξης εφαρμογών για κινητά. Οι χρήστες αναμένουν γρήγορες εκκινήσεις εφαρμογών, ελάχιστους χρόνους φόρτωσης και ομαλές κινούμενες εικόνες. Στη φάση πριν από τη μετάβαση, θα πρέπει να εντοπιστούν και να αντιμετωπιστούν τα σημεία συμφόρησης των επιδόσεων. Αυτό μπορεί να περιλαμβάνει τη βελτιστοποίηση του κώδικα JavaScript, τη μείωση των περιττών αιτημάτων δικτύου και τη χρήση τεχνικών όπως η τεμπέλικη φόρτωση (lazy loading) των στοιχείων για τη βελτίωση των χρόνων φόρτωσης. Οι βελτιστοποιήσεις επιδόσεων και οι βέλτιστες πρακτικές του Ionic θα πρέπει να υιοθετηθούν για να ανταποκριθούν σε αυτές τις προσδοκίες.
- **Λειτουργικότητα εκτός σύνδεσης:**Οι χρήστες κινητών συσκευών αντιμετωπίζουν συχνά καταστάσεις με περιορισμένη ή καθόλου συνδεσιμότητα στο διαδίκτυο. Ως εκ τούτου, αναμένουν από τις εφαρμογές να προσφέρουν τουλάχιστον κάποιο βαθμό λειτουργικότητας εκτός σύνδεσης. Κατά τη φάση πριν από τη μετάβαση, εξετάζουμε τον τρόπο με τον οποίο η προσωρινή αποθήκευση δεδομένων και η υποστήριξη εκτός σύνδεσης μπορούν να ενσωματωθούν στην εφαρμογή Ionic. Αυτό διασφαλίζει ότι οι χρήστες μπορούν να έχουν πρόσβαση σε βασικό περιεχόμενο ή λειτουργίες ακόμη και όταν βρίσκονται εκτός σύνδεσης, ενισχύοντας τη χρησιμότητα και την αξιοπιστία της εφαρμογής.
- **Αποδοτικότητα της μπαταρίας:**Οι χρήστες κινητών τηλεφώνων κάνουν έντονη χρήση της μπαταρίας. Η υπερβολική κατανάλωση ενέργειας μπορεί να οδηγήσει σε απογοήτευση και να επηρεάσει αρνητικά τη φήμη της εφαρμογής. Στη φάση πριν από τη μετάβαση, αξιολογούμε την κατανάλωση πόρων της εφαρμογής React και κάνουμε τις απαραίτητες προσαρμογές. Θα πρέπει να αξιοποιηθούν τα στοιχεία και οι βελτιστοποιήσεις του Ionic που είναι αποδοτικά για την μπαταρία, ώστε να μειωθεί η κατανάλωση της εφαρμογής από τη συσκευή του χρήστη.
- **Συνέπεια σε όλες τις πλατφόρμες:**Οι χρήστες κινητών συσκευών συχνά εναλλάσσονται μεταξύ διαφορετικών συσκευών και πλατφορμών. Αναμένουν μια συνεπή εμπειρία ανεξάρτητα από το αν χρησιμοποιούν iOS ή Android. Στη φάση πριν από τη μετάβαση, βεβαιωνόμαστε ότι ο σχεδιασμός και η λειτουργικότητα της εφαρμογής Ionic παραμένουν συνεπείς σε όλες τις πλατφόρμες. Αυτό περιλαμβάνει την τήρηση των κατευθυντήριων γραμμών σχεδιασμού για συγκεκριμένες πλατφόρμες και τη διασφάλιση ότι τα βασικά χαρακτηριστικά συμπεριφέρονται με παρόμοιο τρόπο τόσο στις συσκευές iOS όσο και στις συσκευές Android.
- **Ασφάλεια και απόρρητο:** Η ασφάλεια και η ιδιωτικότητα των δεδομένων των χρηστών είναι αδιαπραγμάτευτες προσδοκίες στο σημερινό τοπίο των κινητών συσκευών. Κατά τη φάση πριν από τη μετάβαση, αξιολογούμε τα μέτρα ασφαλείας και τις πρακτικές απορρήτου της υπάρχουσας εφαρμογής React. Εφαρμόζουμε ισχυρά χαρακτηριστικά ασφαλείας και κρυπτογράφηση δεδομένων για την προστασία των πληροφοριών των χρηστών. Η τήρηση των προτύπων του κλάδου και η συμμόρφωση

με τους κανονισμούς προστασίας δεδομένων είναι απαραίτητες για την ικανοποίηση των προσδοκιών των χρηστών και την οικοδόμηση εμπιστοσύνης.

- **Κανάλια ανατροφοδότησης των χρηστών:** Καθιερώνουμε σαφή κανάλια για την ανατροφοδότηση των χρηστών και την αναφορά σφαλμάτων. Οι χρήστες εκτιμούν τις εφαρμογές που αναζητούν ενεργά και ανταποκρίνονται στις πληροφορίες τους. Συμπεριλαμβάνουμε μηχανισμούς ανατροφοδότησης εντός της εφαρμογής Ionic, όπως φόρμες εντός της εφαρμογής ή συνδέσμους προς κανάλια υποστήριξης, ώστε οι χρήστες να μπορούν να αναφέρουν προβλήματα ή να προτείνουν βελτιώσεις. Αυτό προάγει μια σχέση συνεργασίας με τους χρήστες και αποδεικνύει τη δέσμευση για την ικανοποίηση των προσδοκιών τους.

Συμπερασματικά, η κατανόηση και η ευθυγράμμιση με τις προσδοκίες των χρηστών για τη λειτουργικότητα και την απόδοση των κινητών συσκευών κατά τη φάση πριν από τη μετάβαση είναι ζωτικής σημασίας για την επιτυχία μιας μετάβασης από React σε Ionic. Δίνοντας προτεραιότητα στην απόκριση, τη φιλικότητα προς την αφή, τη βελτιστοποίηση των επιδόσεων, τη λειτουργικότητα εκτός σύνδεσης, την αποδοτικότητα της μπαταρίας, τη συνέπεια της πλατφόρμας, την ασφάλεια και τα κανάλια ανατροφοδότησης των χρηστών, οι ομάδες ανάπτυξης μπορούν να διασφαλίσουν ότι η προκύπτουσα εφαρμογή Ionic για κινητά όχι μόνο ανταποκρίνεται αλλά και υπερβαίνει τις προσδοκίες των χρηστών, οδηγώντας τελικά σε θετική εμπειρία των χρηστών και επιτυχία της εφαρμογής.

3.2 Γενική δομή φακέλων και αρχείων

Μια οργανωμένη δομή φακέλων για εφαρμογές React, δίνει έμφαση στην αρθρωτότητα, την επεκτασιμότητα και τη συντηρησιμότητα. Η δομή αυτή, παρέχει σαφή διαχωρισμό των αρμοδιοτήτων, λειτουργιών και προάγει την αποτελεσματική συνεργασία μεταξύ των προγραμματιστών.

Το React, ως μια ευρέως χρησιμοποιούμενη βιβλιοθήκη JavaScript για τη δημιουργία διεπαφών χρήστη, απαιτεί μια καλά δομημένη ιεραρχία φακέλων για τη διαχείριση σύνθετων εφαρμογών. Ακολουθεί μια παρουσίαση μιας δομημένη προσέγγιση για την οργάνωση του κώδικα σε μια εφαρμογή React, εστιάζοντας στην αρθρωτότητα και τη σαφήνεια, και είναι αυτή που χρησιμοποιείται και στο project μας.

3.2.1 Δομή φακέλων

3.2.1.1 Φάκελος root

- *assets*: Ο φάκελος "assets" χρησιμεύει ως κεντρική τοποθεσία για την αποθήκευση στατικών πόρων που ενισχύουν τις οπτικές πτυχές της εφαρμογής. Αυτό περιλαμβάνει εικόνες, γραμματοσειρές και άλλα στοιχεία που δεν αποτελούν κώδικα και είναι ζωτικής σημασίας για τη διεπαφή χρήστη. Η τοποθέτηση αυτών των στοιχείων εδώ, απλοποιεί τη διαχείρισή τους και διασφαλίζει ότι είναι εύκολα προσβάσιμα σε διάφορα μέρη της εφαρμογής χωρίς να επιβαρύνουν το codebase. Προωθείται η συνοχή του σχεδιασμού και της παρουσίασης σε ολόκληρη την εφαρμογή.
- *modules*: Ο φάκελος "modules" βρίσκεται στον πυρήνα της δομής των φακέλων και αντιπροσωπεύει μια βασική αρχή οργάνωσης. Κάθε ενότητα ενθυλακώνει ένα

συγκεκριμένο μέρος της λειτουργικότητας της εφαρμογής, όπως ένα χαρακτηριστικό ή ένα τμήμα της διεπαφής χρήστη. Χωρίζοντας την εφαρμογή σε ενότητες, οι προγραμματιστές μπορούν να απομονώσουν σχετικά στοιχεία, σελίδες και αλληλεπιδράσεις API. Αυτή η αρθρωτή προσέγγιση διευκολύνει την ευκολότερη συντήρηση, την επαναχρησιμοποίηση του κώδικα και τη συνεργασία μεταξύ των μελών της ομάδας, καθώς κάθε ενότητα μπορεί να αναπτυχθεί και να δοκιμαστεί ανεξάρτητα.

- *styles*: Ο φάκελος "styles" διαδραματίζει κρίσιμο ρόλο στη διατήρηση μιας συνεκτικής και συνεπούς διεπαφής χρήστη. Τα global ή κοινά στυλ συγκεντρώνονται εδώ για να διασφαλιστεί ότι η εφαρμογή ακολουθεί μια ενιαία γλώσσα σχεδίασης. Αυτός ο διαχωρισμός των στυλ από το CSS ή το styling των συγκεκριμένων συστατικών μειώνει τον πλεονασμό, ελαχιστοποιεί τις συγκρούσεις και απλοποιεί το έργο ενός συνεχούς και συνεπούς styling σε ολόκληρη την εφαρμογή. Προωθείται έτσι, μια εξορθολογισμένη και αποτελεσματική προσέγγιση για τη διαχείριση της οπτικής αισθητικής της εφαρμογής.

3.2.1.2 Φάκελοι ενοτήτων

Για κάθε ενότητα δημιουργείται και συντηρείται η ακόλουθη δομή φακέλων:

- *api*: Ο φάκελος "api" μέσα σε κάθε ενότητα είναι υπεύθυνος για τη διαχείριση των αλληλεπιδράσεων δεδομένων με εξωτερικές υπηρεσίες ή διακομιστές. Αποτελείται συνήθως από τρία βασικά αρχεία:
    - *api.js*: Αυτό το αρχείο χρησιμεύει ως κεντρικός κόμβος για τον ορισμό τελικών σημείων API και ρυθμίσεων ειδικά για την ενότητα. Εξασφαλίζει ότι όλος ο κώδικας που σχετίζεται με το API είναι οργανωμένος σε ένα μέρος, διευκολύνοντας την ενημέρωση και τη συντήρησή του καθώς εξελίσσεται η εφαρμογή.
    - *commands.js*: Το αρχείο "commands.js" επικεντρώνεται στην ενθυλάκωση ειδικών για την ενότητα ενεργειών ή εντολών που προκαλούν αλλαγές στην κατάσταση ή τα δεδομένα της εφαρμογής. Ο διαχωρισμός των εντολών από άλλο κώδικα που σχετίζεται με το API ενισχύει τη σαφήνεια και απλοποιεί την αποσφαλμάτωση και τον έλεγχο.
    - *queries.js*: Στο αρχείο "queries.js" ορίζονται τα ειδικά για την ενότητα ερωτήματα ή οι λειτουργίες ανάκτησης δεδομένων. Αυτός ο διαχωρισμός διασφαλίζει ότι ο κώδικας ανάκτησης δεδομένων είναι διακριτός από άλλη λογική που σχετίζεται με το API, διευκολύνοντας την αποτελεσματική διαχείριση δεδομένων εντός της ενότητας.
- *components*: Ο φάκελος "components" περιέχει επαναχρησιμοποιήσιμα στοιχεία UI και το κύριο component της ενότητας:
    - *OtherComponent1.jsx* και *OtherComponent2.jsx*: Αυτά τα αρχεία φιλοξενούν επαναχρησιμοποιήσιμα στοιχεία που είναι προσαρμοσμένα στις συγκεκριμένες ανάγκες της ενότητας. Με την απομόνωση αυτών των στοιχείων εντός της ενότητας, οι προγραμματιστές μπορούν εύκολα να τα επαναχρησιμοποιούν εντός της ενότητας και να τα συντηρούν με τρόπο που δεν επηρεάζει άλλα τμήματα της εφαρμογής.

    - *MainModuleComponent.jsx*: Το αρχείο "MainModuleComponent.jsx" είναι το πιο σημαντικό component για τη διεπαφή χρήστη της ενότητας. Δεν αποδίδει μόνο το κύριο περιεχόμενο της ενότητας, αλλά μπορεί επίσης να χειριστεί τη δρομολόγηση και την πλοήγηση ειδικά για την ενότητα. Αυτό το στοιχείο παρέχει μια καθαρή διεπαφή για την αλληλεπίδραση με την ενότητα, προωθώντας την ενθυλάκωση και την επαναχρησιμοποίηση.

- *pages*: Ο κατάλογος "pages" μέσα σε κάθε ενότητα οργανώνει τη διεπαφή χρήστη σε μεμονωμένες σελίδες. Κάθε σελίδα βρίσκεται στον ειδικό της φάκελο και ακολουθεί μια συνεπή δομή:
    - *index.jsx*: Το αρχείο "index.jsx" λειτουργεί ως σημείο εισόδου για τη σελίδα, συνδυάζοντας τα στοιχεία που είναι υπεύθυνα για την απόδοση του περιεχομένου και της λειτουργικότητας της σελίδας. Αυτή η ενθυλάκωση εξασφαλίζει ότι κάθε σελίδα είναι αυτοτελής και απλοποιεί τη διαδικασία προσθήκης ή τροποποίησης σελίδων εντός της ενότητας.
    - *usePage1.jsx*: Το αρχείο "usePage1.jsx" διαχειρίζεται τις κλήσεις API και τις εντολές που αφορούν ειδικά τη σελίδα. Απομονώνει τις λειτουργίες ανάκτησης δεδομένων και αλλαγής δεδομένων, διασφαλίζοντας ότι η λογική της σελίδας είναι καλά διαχωρισμένη από άλλα component που είναι υπεύθυνα μόνο για την απόδοση της διεπαφής χρήστη.
    - *Page1.jsx*: Το αρχείο "Page1.jsx" είναι αφιερωμένο στη λογική προβολής της σελίδας. Επικεντρώνεται στα στοιχεία διεπαφής χρήστη, τη διάταξη και τη διαδραστικότητα που αφορούν ειδικά τη σελίδα. Ο διαχωρισμός της λογικής προβολής από άλλες πτυχές της σελίδας βελτιώνει τη συντηρησιμότητα και τη δυνατότητα ελέγχου.
    - *Page1Toolbar.jsx*: Το αρχείο "Page1Toolbar.jsx" αντιπροσωπεύει τη γραμμή εργαλείων ή την κεφαλίδα που σχετίζεται με τη σελίδα. Παρέχει μια ειδική θέση για τη διαχείριση στοιχείων UI που σχετίζονται με την πλοήγηση ή τις ενέργειες σε επίπεδο σελίδας, εξασφαλίζοντας έναν καθαρό διαχωρισμό των ανησυχιών εντός της σελίδας.

Η δομή φακέλων που περιγράφεται σε αυτή την εργασία προσφέρει μια σπονδυλωτή και δομημένη προσέγγιση στην ανάπτυξη εφαρμογών React. Με την οργάνωση του κώδικα σε διακριτές ενότητες και τον περαιτέρω διαχωρισμό των αρμοδιοτήτων εντός κάθε ενότητας, οι προγραμματιστές μπορούν να διατηρήσουν υψηλά επίπεδα σαφήνειας, διαχειρισιμότητας του κώδικα και επεκτασιμότητας. Αυτή η δομή ευνοεί τη συνεργατική ανάπτυξη, εξορθολογίζει τη συντήρηση του κώδικα και παρέχει μια ισχυρή βάση για σύνθετες εφαρμογές React.

3.2.2 Ανάλυση της δομής φακέλων και αρχείων

Παρακάτω αναλύεται η δομή αρχείων και φακέλων της εφαρμογής. Η δομή της εφαρμογής είναι πολύ λεπτομερής και καλά οργανωμένη, αποτελώντας τη βάση του συνολικού μας κώδικα. Από τον φάκελο "src" ξεκινάμε με τον εντοπισμό των κύριων τμημάτων της εφαρμογής.

Ο φάκελος "assets" περιέχει τα πολυμέσα όπως εικόνες και γραφικά, που χρησιμοποιούνται στην εφαρμογή, ενώ το "modules" αποτελεί μια καλή πρακτική για τον ορισμό ανεξάρτητων

τμημάτων της εφαρμογής, όπως η πιστοποίηση του χρήστη, οι εργασίες, τα έγγραφα, οι επαφές και πολλά άλλα. Αυτό επιτρέπει να οργανώσουμε καλύτερα τον κώδικα και να διατηρήσουμε την ευκολία στη διαχείριση της εφαρμογής.

Στον φάκελο "auth" βρίσκεται η λογική που σχετίζεται με την πιστοποίηση του χρήστη, ενώ στο φάκελο που περιέχει "components" υπάρχουν τα components που χρησιμοποιούνται σε διάφορα μέρη της εφαρμογής, όπως τα "SupportButton" και "UserAvatar". Ακόμα ο φάκελος "pages" του auth περιέχει τις σελίδες της εφαρμογής που είναι ορατές από τον χρήστη, όπως η σελίδα "Login". Στο "utils" βρίσκουμε βοηθητικά αρχεία και εργαλεία που χρησιμοποιούνται σε διάφορα μέρη της εφαρμογής.

Ο φάκελος "calendar", για παράδειγμα, περιλαμβάνει τα components και τις σελίδες που σχετίζονται με το ημερολόγιο, ενώ ο φάκελος "contacts" περιέχει τα στοιχεία που αφορούν τις επαφές.

Ο φάκελος "coordinations" να αντιστοιχεί σε λειτουργίες που σχετίζονται με τον συντονισμό, ενώ ο φάκελος "customLists" φαίνεται να περιέχει προσαρμοσμένες λίστες και τον σχετικό κώδικα.

Ο φάκελος "documents" αφορά όλα τα σχετικά με τα έγγραφα κομμάτια της εφαρμογής, όπως η αναζήτηση, η προβολή και η επεξεργασία των εγγράφων. Ο φάκελος "goals" περιλαμβάνει τον κώδικα που αφορά τους στόχους.

Το "issues" αντιστοιχεί σε λειτουργίες που σχετίζονται με τη διαχείριση των σημαντικών θεμάτων. Ο φάκελος "leaves" περιλαμβάνει τα στοιχεία που σχετίζονται με τη διαχείριση των αδειών, ενώ ο φάκελος "library" περιλαμβάνει τη διαχείριση μιας βιβλιοθήκης.

Ο φάκελος "registry" σχετίζεται με τη διαχείριση του μητρώου και της καταγραφής πρωτοκόλλου, ενώ ο φάκελος "support" περιλαμβάνει τα στοιχεία που αφορούν την υποστήριξη της εφαρμογής μας.

Ο φάκελος "tasks" περιλαμβάνει λειτουργίες που σχετίζονται με τις εργασίες και τη διαχείρισή τους. Το "tools" περιέχει εργαλεία ή λειτουργίες που δεν ανήκουν σε καμία άλλη κατηγορία.

Τέλος, ο φάκελος "welcome" αναφέρεται στο αρχικό καλωσόρισμα του χρήστη

Αυτή η δομή βοηθά στην ευκολία της διαχείρισης και ανάπτυξης της εφαρμογής, ενώ οι "api" φάκελοι αναλαμβάνουν την επικοινωνία με εξωτερικούς πόρους και διακομιστές.

Συνολικά, η δομή αυτή παρέχει μια σαφή επισκόπηση του κώδικα και των διαφόρων τμημάτων της εφαρμογής, καθιστώντας τη διαδικασία μεταφοράς στην Ionic εφαρμογή μας προσιτή και οργανωμένη.

```
public
src
   • assets
   • modules
         o auth
               ■ assets
               ■ components
                     • AuthContainer
                     • ChangePassword
                     • SupportButton
                     • UserAvatar
                     • UserDrawer
```

        - pages
            - Login
        - utils
        - index.jsx
    - calendar
        - api
        - components
            - Calendar
        - pages
            - Calendar
        - index.jsx
    - contacts
        - api
        - components
            - Contacts
        - pages
            - ContactsExternal
            - ContactsHaf
            - ContactsRoots
            - ContactsRootsOrg
            - Create
            - EditExternalContact
            - ViewContactHaf
            - ViewExternalContact
        - index.jsx
    - coordinations
        - api
        - components
            - Coordinations
            - Coordinator
        - pages
            - Create
            - Edit
            - Provided
        - index.jsx
    - customLists
        - api
        - components
            - Approval
            - CustomLists
            - Distribution
            - Recipients
            - ViewCard
        - pages
            - CreateCustomApprovalTable
            - CreateCustomDistributionTable
            - CreateCustomRecipientTable
            - CustomApprovalTables
            - CustomDistributionTables
            - CustomRecipientTable
            - EditCustomApprovalTable
            - EditCustomDistributionTable
            - EditCustomRecipientTable
            - ViewCustomApprovalTable
            - ViewCustomDistributionTable
            - ViewCustomRecipientTable
        - index.jsx
    - documents
        - api
        - components
            - ChildApproval

        - ChildrenApproval
        - DocumentApproval
        - DocumentApprovalMasterView
        - DocumentApprovalView
        - DocumentAttachments
        - DocumentChargeAction
        - DocumentChargeInfo
        - DocumentChargeRecipients
        - DocumentChildren
        - DocumentChildrenView
        - DocumentDetails
        - DocumentDistribution
        - DocumentForward
        - DocumentForwardDistribution
        - DocumentImportRecipients
        - DocumentImportSender
        - DocumentImportStatus
        - DocumentPriority
        - DocumentReferences
        - Documents
        - DocumentSearch
        - DocumentSearchLibrary
        - DocumentSearchProtocol
        - DocumentStatus
        - DocumentTags
        - DocumentTasksStatus
        - DocumentVersions
        - DocumentViewComponents
        - DocumentViewLink
        - EditDocumentAttachments
        - FolderSelect
        - HistoryDetailsDrawer
    - hooks
    - pages
        - Action
        - Archived
        - ArchivedMaster
        - Clone
        - CloneMaster
        - ContinueMaster
        - CorrectRepetition
        - CorrectRepetitionImported
        - Create
        - CreateChild
        - CreateFromEndpoint
        - CreateImport
        - CreateMaster
        - Distributed
        - Distribution
        - DocumentTasks
        - Edit
        - EditChild
        - EditImport
        - EditMaster
        - External
        - Imported
        - Info
        - InProgress
        - Landing
        - Pending
        - RegisteredEdit

            - Rejected
            - Rejection
            - Reply
            - Tagged
            - View
            - ViewChild
            - ViewMaster
            - ViewMasterVersion
            - ViewReference
            - ViewReferenceMaster
            - ViewVersion
        - index.jsx
    - goals
        - api
        - components
            - DocumentSearch
            - DocumentSearchLibrary
            - GoalInvolved
            - GoalRecipients
            - GoalReferences
            - Goals
            - GoalsTasks
        - pages
            - Create
            - Edit
            - GoalsAll
            - View
        - index.jsx
    - issues
        - api
        - components
            - DocumentSearch
            - DocumentSearchLibrary
            - IssueInvolved
            - IssueRecipients
            - IssuesReferences
            - Issues
            - IssueTasks
        - pages
            - Create
            - CreateFromDoc
            - CreateFromTask
            - Edit
            - IssuesAll
            - View
        - index.jsx
    - leaves
        - api
        - components
            - History
            - LeaveApproval
            - LeaveDestinations
            - LeaveRecipient
            - Leaves
            - LeaveSubstitute
        - pages
            - Approved
            - Clone
            - Create
            - Edit
            - LeavesAll

            - Pending
            - View
        - index.jsx
    - library
        - api
        - components
            - Library
            - LibraryAttachments
        - pages
            - Create
            - Edit
            - Library
            - View
        - index.jsx
    - registry
        - api
        - components
            - DocumentRecipients
            - FolderSelect
            - Registry
            - RegistryDetails
        - hooks
        - pages
            - InsertInbound
            - InsertOutbound
            - RegistryIn
            - RegistryOut
        - index.jsx
    - shared
        - api
        - assets
        - components
            - Accordion
            - Alert
            - AnimatedBackground
            - AnimatedLandingBackground
            - AnimatedText
            - AppLayout
            - AppTour
            - Badge
            - Button
            - Calendar
            - Callout
            - Carousel
            - Display
            - Drawer
            - ErrorFallback
            - Field
            - Filter
            - FloatingMenu
            - Footer
            - FormControl
            - Header
            - Icon
            - Image
            - LandingCard
            - LandingNavCard
            - Loading
            - Notes
            - Page
            - Pagination

            - PdfViewer
            - Popup
            - ProgressBar
            - Sidebar
            - SortableList
            - Spinner
            - Table
            - Tabs
            - Timeline
            - Toasts
            - index.jsx
        - hooks
        - providers
        - utils
        - index.jsx
    - substitutions
        - api
        - components
            - Substitute
            - Substitutions
        - pages
            - Create
            - Edit
            - FromSubstitution
            - Substituted
        - index.jsx
    - support
        - api
        - assets
        - components
            - AnnouncementsButton
            - AnnouncementsDrawer
            - BugReportButton
            - HelpButton
            - Support
            - SupportButton
            - SupportDrawer
            - SupportMobileButton
            - SupportMobileDrawer
        - pages
            - ChangeLog
            - Faqs
            - Landing
            - Settings
            - Team
        - utils
        - index.jsx
    - tasks
        - api
        - components
            - DocumentSearch
            - DocumentSearchLibrary
            - FinishGroupTasks
            - Percentage
            - TaskDocuments
            - TaskGoals
            - TaskIndicator
            - TaskPercentage
            - TaskPriority
            - TaskRecipient
            - TaskRecipients

```
                    ● Tasks
                    ● TasksSignificantIssues
                    ● TaskStatus
                ■ pages
                    ● Create
                    ● CreateChild
                    ● CreateFromDoc
                    ● Edit
                    ● InboxAll
                    ● InboxComplete
                    ● InboxDelegates
                    ● InboxPending
                    ● Landing
                    ● OutboxAll
                    ● OutboxComplete
                    ● OutboxDelegates
                    ● OutboxPending
                    ● View
                ■ index.jsx
        ○ tools
            ■ api
            ■ components
                ● Tools
                ● ViewCard
            ■ pages
                ● Landing
            ■ index.jsx
        ○ welcome
            ■ pages
                ● index.jsx
                ● Welcome.jsx
        ○ App.jsx
    ● styles
        ○ App.css
        ○ fontSize.module.css
        ○ index.scss
        ○ theme.scss
        ○ useFontSizeToggle.jsx
        ○ useThemeStyles.module.css
index.jsx
```

3.3 Ανάλυση της αρχιτεκτονικής που ακολουθείται

Ως μια διάσημη βιβλιοθήκη JavaScript για τη δημιουργία διεπαφών χρήστη και στο πλαίσιο αυτής της αρχιτεκτονικής, τα στοιχεία React αναλαμβάνουν κεντρικό ρόλο ως τα θεμελιώδη δομικά στοιχεία για την κατασκευή της διεπαφής χρήστη. Αυτή η προσέγγιση με βάση τα components διευκολύνει την αρθρωτότητα (modularity) στην εφαρμογή, καθώς και τη συντηρησιμότητά της.

Η αρχιτεκτονική που ακολουθείται σε όλη την εφαρμογή παραμένει η ίδια σε όλα τα modules και τα components που υπάρχουν. Επειδή λόγω έκτασης της εργασίας δεν μπορούν να αναλυθούν όλα, από εδώ και πέρα θα αναλυθεί η ενότητα της εφαρμογής που αφορά τη βιβλιοθήκη. Αυτή η ενότητα περιέχει τις σελίδες - οθόνες: Create (Δημιουργία ανάρτησης) , Edit (Επεξεργασία ανάρτησης) , View (Προβολή ανάρτησης) και Table (Πίνακας αναρτήσεων). Όμως και πάλι η αρχιτεκτονική παραμένει ίδια ανάμεσα στις εκάστοτε σελίδες. Για τον λόγο αυτό, θα αναλυθεί παρακάτω η αρχιτεκτονική που ακολουθείται στη σελίδα της Προβολής ανάρτησης (view) με url: /library/view/{docId}.

Παρακάτω θα αναλύεται η αρχιτεκτονική που ακολουθείται σε μία συγκεκριμένη σελίδα της εφαρμογής που είναι η ViewLibrary. Αυτή είναι υπεύθυνη για την προβολή ενός εγγράφου από την βιβλιοθήκη. Χάρη στο συνεπές στυλ κωδικοποίησης και κώδικα που ακολουθείται σε όλη την εφαρμογή, η συγκεκριμένη σελίδα είναι δομημένη όπως όλες τις άλλες.

Ένα αξιοσημείωτο χαρακτηριστικό αυτής της αρχιτεκτονικής είναι η ενσωμάτωση προσαρμοσμένων hooks, όπως τα useQuery, useCommand, useTitle, buttonOrdering και useAuth. Αυτά τα προσαρμοσμένα hooks ενθυλακώνουν διακριτές πτυχές της λογικής της εφαρμογής, συμπεριλαμβανομένης της ανάκτησης δεδομένων, της διαχείρισης κατάστασης, του ελέγχου ταυτότητας και της συμπεριφοράς του περιβάλλοντος εργασίας. Με την ενθυλάκωση αυτών των λειτουργιών σε hook, ο κώδικας τηρεί τις βέλτιστες πρακτικές διαχωρισμού των λειτουργιών και προωθεί την επαναχρησιμοποίηση του κώδικα.

Η δρομολόγηση εντός της εφαρμογής επιτυγχάνεται απρόσκοπτα μέσω του πακέτου React Router, μιας βιβλιοθήκης που έχει σχεδιαστεί για την πλοήγηση στην πλευρά του client. Hooks όπως το useParams και το useNavigate επιτρέπουν στους χρήστες να μετακινούνται μεταξύ διαφόρων προβολών - σελίδων, βελτιώνοντας τη συνολική εμπειρία του χρήστη.

Η εφαρμογή διασυνδέεται με το API χρησιμοποιώντας την κλάση LibraryApi. Αυτή η ενθυλάκωση της λειτουργικότητας που σχετίζεται με το API βελτιώνει τις εργασίες ανάκτησης και διαχείρισης δεδομένων, όπως η ανάκτηση εγγράφων βιβλιοθήκης και η εκτέλεση εντολών διαγραφής.

Η διαμόρφωση των components και το στυλ τους, μια βασική πτυχή της ανάπτυξης διεπαφής χρήστη, αντιμετωπίζεται με την εφαρμογή κλάσεων και στυλ CSS. Βιβλιοθήκες που χρησιμοποιούνται στην εφαρμογή όπως η clsx που χρησιμοποιείται στην προκειμένη περίπτωση, μπορούν να χρησιμοποιηθούν για τη δημιουργία classNames, εξασφαλίζοντας συνεπές και οπτικά ελκυστικό UI. Επιπλέον, οι τεχνικές CSS-in-JS μπορούν να χρησιμοποιηθούν για δυναμικό στυλ, συμβάλλοντας σε έναν συνεκτικό οπτικό σχεδιασμό.

Η διαχείριση καταστάσεων είναι μια ζωτικής σημασίας πτυχή της αρχιτεκτονικής, όπου χρησιμοποιούνται hooks της React όπως τα useState, useMemo και προσαρμοσμένες λύσεις διαχείρισης καταστάσεων. Η εφαρμογή διαχειρίζεται αποτελεσματικά την κατάσταση της φόρμας, μέσω του hook useFormValues και της ανάλογης προσαρμοσμένης λογικής, εξασφαλίζοντας συγχρονισμό και απόκριση στις αλληλεπιδράσεις της φόρμας.

Οι ειδοποιήσεις χρηστών αποτελούν κρίσιμο μέρος της αρχιτεκτονικής, με παράδειγμα τη συνάρτηση toastSuccess. Αυτές οι ειδοποιήσεις ενημερώνουν άμεσα τους χρήστες για επιτυχημένες ενέργειες, ενισχύοντας τη φιλικότητα προς το χρήστη και την ανατροφοδότηση.

Το κύριο component που είναι αρμόδιο για τις λειτουργίες είναι το useViewLibrary. Αυτό το περιλαμβάνει μια σειρά αρμοδιοτήτων ζωτικής σημασίας για τη λειτουργικότητα της εφαρμογής.

Το κύριο στοιχείο για την προβολή της σελίδας είναι το ViewLibrary, και στην ουσία σε αυτό το component δεν υπάρχει καθόλου business logic, παρά μόνο στοιχεία τα οποία προβάλλονται στον χρήστη.

Η άντληση δεδομένων αποτελεί βασική ευθύνη της useViewLibrary. Χρησιμοποιείται το hook useQuery για την ανάκτηση δεδομένων εγγράφων της βιβλιοθήκης βάσει του παρεχόμενου id. Τα αποκτηθέντα δεδομένα αποθηκεύονται στην ιδιότητα query.data, καθιστώντας τα προσβάσιμα για την απόδοση τους προς τον χρήστη.

Ο χειρισμός του ελέγχου ταυτότητας ενσωματώνεται στο component, χρησιμοποιώντας το hook useAuth με σκοπό την πρόσβαση σε πληροφορίες που σχετίζονται με τον έλεγχο ταυτότητας, συμπεριλαμβανομένου του διακριτικού - token- και του προφίλ του χρήστη. Αυτό διασφαλίζει ότι τα διαπιστευτήρια και οι ρόλοι των χρηστών (προφίλ) είναι άμεσα διαθέσιμα για εξουσιοδοτημένες ενέργειες.

Η εκτέλεση εντολών, πραγματοποιείται μέσω του hook useCommand. Η συνάρτηση πχ . handleDelete ενορχηστρώνει την εκτέλεση εντολών, παρέχοντας στους χρήστες μια ομαλή εμπειρία. Μετά την επιτυχή διαγραφή, πλοηγείται στην προβολή της βιβλιοθήκης και εμφανίζει μια ειδοποίηση επιτυχίας, ενισχύοντας την ικανοποίηση των χρηστών.

Η δημιουργία κουμπιών είναι μια άλλη κρίσιμη πτυχή της useViewLibrary. Αξιοποιώντας το hook useMemo, το component παράγει μια σειρά από κουμπιά, συμπεριλαμβανομένων των επιλογών "Edit" και "Delete". Αυτά τα κουμπιά απλοποιούν τις αλληλεπιδράσεις του χρήστη για εργασίες όπως η επεξεργασία και η αφαίρεση εγγράφων της βιβλιοθήκης. Οι ιδιότητες των κουμπιών, όπως τα εικονίδια, οι συμβουλές εργαλείων, τα χρώματα και οι καταστάσεις φόρτωσης, διαμορφώνονται ξεχωριστά για βέλτιστη χρηστικότητα και παραμετροποίηση.

Η ενσωμάτωση της δρομολόγησης επιτυγχάνεται απρόσκοπτα μέσω της αξιοποίησης της λειτουργίας useNavigate από το React Router. Αυτή η λειτουργία επιτρέπει την ομαλή πλοήγηση σε διαφορετικές προβολές εντός της εφαρμογής, ενισχύοντας την εξερεύνηση και την αλληλεπίδραση του χρήστη. Επιπλέον, το hook useTitle ορίζει δυναμικά τον τίτλο της σελίδας ώστε να αντικατοπτρίζει τον τίτλο του εγγράφου της βιβλιοθήκης που προβάλλεται, ενισχύοντας την κατανόηση και την προσβασιμότητα του χρήστη.

3.4 Παράδειγμα φακέλων και αρχείων με κώδικα

Παρακάτω παρουσιάζεται μια κλασική δομή αρχείων και φακέλων μαζί με τον κώδικα τους που μπορεί να έχει το κάθε module της εφαρμογής. Λόγω της έκτασης της εφαρμογής, δεν μπορεί να παρουσιαστεί όλος ο κώδικας που χρησιμοποιήθηκε. Για τον λόγο αυτό, παρακάτω διατίθεται ο κώδικας ο οποίος είναι υπεύθυνος για την ενότητα της βιβλιοθήκης. Κάθε ενότητα περιλαμβάνει τους φακέλους api, components και pages.

- **Φάκελος `api`:** Σε αυτόν τον φάκελο συνήθως τοποθετούνται τα αρχεία που σχετίζονται με την επικοινωνία του module με άλλες υπηρεσίες ή πηγές δεδομένων. Μπορεί να περιλαμβάνει τα αρχεία που αντιπροσωπεύουν τα API endpoints, αρχεία που διαχειρίζονται αιτήσεις και απαντήσεις, καθώς και αρχεία που παρέχουν συναρτήσεις για την επικοινωνία με τις υπηρεσίες.
- **Φάκελος `components`:** Σε αυτόν τον φάκελο τοποθετούνται τα αρχεία που αντιπροσωπεύουν τα components που χρησιμοποιούνται στο module. Αυτά τα components μπορεί να είναι είτε γενικής χρήσης και να χρησιμοποιούνται από πολλές σελίδες του module, είτε ειδικά για συγκεκριμένες σελίδες. Αυτά τα αρχεία περιλαμβάνουν τον κώδικα HTML, CSS και JavaScript που απαιτείται για τη δημιουργία των UI στοιχείων και τη λογική που αφορά τα components.
- **Φάκελος `pages`:** Σε αυτόν τον φάκελο τοποθετούνται τα αρχεία που αντιπροσωπεύουν τις σελίδες του module. Κάθε σελίδα αντιστοιχεί σε ένα διαφορετικό route του module και αποτελεί την βασική μονάδα πλοήγησης στο πλαίσιο του module. Συνήθως, αυτά τα αρχεία περιλαμβάνουν τη δομή της σελίδας και καλούν τα αντίστοιχα components για να δημιουργήσουν τη σελίδα και να εφαρμόσουν τη λογική της.

Αυτή η δομή βοηθά στην οργάνωση και τη συντήρηση του κώδικα, καθώς επιτρέπει την διαχωρισμένη διαχείριση των διαφορετικών πτυχών του module. Κάθε φάκελος έχει τον δικό του σκοπό και αρμοδιότητες, ενώ ο κώδικας παραμένει οργανωμένος και ευανάγνωστος, επιτρέποντας την ευκολότερη συντήρηση και επέκταση του module.

3.4.1 api

Ο συγκεκριμένος φάκελος περιέχει τρία κύρια αρχεία που είναι απαραίτητα για τη λειτουργία ενός συγκεκριμένου module. Αυτά τα αρχεία διαδραματίζουν σημαντικό ρόλο στην επικοινωνία του module με άλλα τμήματα του προγράμματος ή της εφαρμογής.

- **Αρχείο Κεντρικού API:** Αυτό το αρχείο είναι το κεντρικό API του module και περιλαμβάνει συναρτήσεις και κλάσεις που παρέχουν πρόσβαση σε συγκεκριμένες υπηρεσίες ή δυνατότητες που προσφέρει το module.
- **Αρχείο Ερωτημάτων (Queries):** Το αρχείο αυτό περιέχει τις συναρτήσεις ή τα endpoints που χρησιμοποιούνται για να διαβάσουν πληροφορίες από το module. Αυτά τα ερωτήματα μπορεί να αφορούν την ανάκτηση δεδομένων από τη βάση δεδομένων ή την επιστροφή πληροφοριών προς άλλα τμήματα του προγράμματος.
- **Αρχείο Εντολών (Commands):** Το αρχείο αυτό περιέχει συναρτήσεις ή endpoints που χρησιμοποιούνται για να εκτελέσουν ενέργειες ή να πραγματοποιήσουν μεταβολές στο module. Μπορεί να περιλαμβάνει λειτουργίες που επιτρέπουν την ενημέρωση της βάσης δεδομένων, την δημιουργία νέων εγγραφών ή άλλες αλλαγές στην κατάσταση του module.

Η διάκριση ανάμεσα σε αρχεία ερωτημάτων και εντολών βοηθά στην οργάνωση του κώδικα και την διαχείριση των λειτουργιών του module. Επίσης, το κεντρικό API παρέχει μια διεπαφή για την επικοινωνία με το module, ενώ τα αρχεία ερωτημάτων και εντολών παρέχουν τις συγκεκριμένες λειτουργίες που μπορούν να εκτελεστούν. Με αυτόν τον τρόπο, ο κώδικας παραμένει οργανωμένος και ευανάγνωστος, ενώ η διαχείριση των ενεργειών και των ερωτημάτων που αφορούν το module γίνεται πιο αποτελεσματική και διαφανής.

Στη συνέχεια παρατίθενται παραδείγματα για τα αρχεία του φακέλου API:

- **api / api.tsx**

```
import { ApiClient } from 'modules/shared' // Εισαγωγή της κλάσης
'ApiClient' από το 'modules/shared'.
import { LibraryQueries } from './queries' // Εισαγωγή της κλάσης
'LibraryQueries' από το τρέχον τοπικό αρχείο 'queries'.
import { LibraryCommands } from './commands' // Εισαγωγή της κλάσης
'LibraryCommands' από το τρέχον τοπικό αρχείο 'commands'.

export class LibraryApi { // Δήλωση της κλάσης 'LibraryApi'.
  private readonly client: ApiClient // Ιδιωτικό πεδίο για τον client της
βιβλιοθήκης.

  readonly queries: LibraryQueries // Δημόσιο πεδίο για τις ερωτήσεις της
βιβλιοθήκης.
  readonly commands: LibraryCommands // Δημόσιο πεδίο για τις εντολές της
βιβλιοθήκης.
```

```
  constructor(token: string, department: number, duty: number) { //
Κατασκευαστής - constructor της κλάσης με παραμέτρους.
    this.client = new ApiClient(token, department, duty) // Δημιουργία ενός
αντικειμένου 'ApiClient'.
    this.queries = new LibraryQueries(this.client) // Δημιουργία ενός
αντικειμένου 'LibraryQueries' με τον πελάτη - client.
    this.commands = new LibraryCommands(this.client) // Δημιουργία ενός
αντικειμένου 'LibraryCommands' με τον πελάτη - client.
  }
}

export default LibraryApi // Εξαγωγή της κλάσης 'LibraryApi' ως προεπιλογή.
```

- **api / commands.tsx**

```
import { ApiClient } from 'modules/shared' // Εισαγωγή της κλάσης
'ApiClient' από το 'modules/shared'.

export class LibraryCommands { // Δήλωση της κλάσης 'LibraryCommands'.
  private readonly client: ApiClient // Ιδιωτικό πεδίο για τον client των
εντολών της βιβλιοθήκης.

  constructor(client: ApiClient) { // Κατασκευαστής της κλάσης με παράμετρο
τον πελάτη (client).
    this.client = client // Ανάθεση του πελάτη στο ιδιωτικό πεδίο.
  }

  public async createLibraryDocument(libraryDocument: any): Promise<any> {
// Δήλωση της δημόσιας μεθόδου για τη δημιουργία εγγράφου βιβλιοθήκης.
    return await this.client.post('/library', libraryDocument) // Αποστολή
αιτήματος POST στον εξυπηρετητή για δημιουργία εγγράφου βιβλιοθήκης.
  }

  public async updateLibraryDocument(id: string, document: any):
Promise<any> { // Δήλωση της δημόσιας μεθόδου για την ενημέρωση εγγράφου
βιβλιοθήκης.
    return await this.client.put(`/library/${id}`, document) // Αποστολή
αιτήματος PUT στον εξυπηρετητή για ενημέρωση εγγράφου βιβλιοθήκης με το
συγκεκριμένο id.
  }

  public async deleteLibraryDocument(id: string): Promise<any> { // Δήλωση
της δημόσιας μεθόδου για τη διαγραφή εγγράφου βιβλιοθήκης.
    return await this.client.delete(`/library/${id}`) // Αποστολή αιτήματος
DELETE στον εξυπηρετητή για διαγραφή εγγράφου βιβλιοθήκης με το
συγκεκριμένο id.
  }
}
```

- **api / index.jsx**

```
export { LibraryApi as LibraryApi } from './api' // Εξαγωγή της κλάσης
'LibraryApi' από το τρέχον τοπικό αρχείο 'api' με το όνομα 'LibraryApi'.
```

- **api / queries.tsx**

```
import { ApiClient, QueryResponse } from 'modules/shared' // Εισαγωγή των
κλάσεων 'ApiClient' και 'QueryResponse' από το 'modules/shared'.

export class LibraryQueries { // Δήλωση της κλάσης 'LibraryQueries'.
  private readonly client: ApiClient // Ιδιωτικό πεδίο για τον client των
queries της βιβλιοθήκης.

  constructor(client: ApiClient) { // Κατασκευαστής της κλάσης με παράμετρο
τον πελάτη (client).
    this.client = client // Ανάθεση του πελάτη στο ιδιωτικό πεδίο.
  }

  public async getLibraryDocuments(params: any): Promise<any> { // Δήλωση
της δημόσιας μεθόδου για τη λήψη εγγράφων βιβλιοθήκης.
    const searchParams = new URLSearchParams(params).toString() //
Δημιουργία και μετατροπή των παραμέτρων αναζήτησης σε συμβολοσειρά.
    return await this.client.get(`/library?${searchParams}`) // Αποστολή
αιτήματος GET στον εξυπηρετητή για λήψη εγγράφων βιβλιοθήκης με τις
συγκεκριμένες παραμέτρους αναζήτησης.
  }

  public async getViewLibraryDocumentById(id: string): Promise<any> { //
Δήλωση της δημόσιας μεθόδου για τη λήψη ενός εγγράφου βιβλιοθήκης με βάση
το ID για προβολή.
    return await this.client.get(`/library/view/${id}`) // Αποστολή
αιτήματος GET στον εξυπηρετητή για λήψη εγγράφου βιβλιοθήκης με το
συγκεκριμένο ID για προβολή.
  }

  public async getLibraryDocumentById(id: string): Promise<any> { // Δήλωση
της δημόσιας μεθόδου για τη λήψη ενός εγγράφου βιβλιοθήκης με βάση το ID.
    return await this.client.get(`/library/${id}`) // Αποστολή αιτήματος
GET στον εξυπηρετητή για λήψη εγγράφου βιβλιοθήκης με το συγκεκριμένο ID.
  }

  public async getNewLibrary(): Promise<QueryResponse<any>> { // Δήλωση της
δημόσιας μεθόδου για τη λήψη νέου εγγράφου βιβλιοθήκης.
    return await this.client.get<any>('/library/new') // Αποστολή αιτήματος
GET στον εξυπηρετητή για λήψη νέου εγγράφου βιβλιοθήκης.
  }
}
```

3.4.2 components

Αυτός ο φάκελος του module περιέχει όλα τα components που χρησιμοποιούνται στις οθόνες της συγκεκριμένης ενότητας, καθώς και ένα ειδικό component που είναι κοινό σε όλα τα modules. Ας εξηγήσουμε τον ρόλο και το περιεχόμενο αυτού του φακέλου:

- **Κοινό Component (Ίδιο όνομα με το module):**Το κοινό component που έχει το ίδιο όνομα με το module είναι υπεύθυνο για την καθολική διάταξη της σελίδας του module. Εδώ καθορίζεται η βασική δομή και ο σχεδιασμός της σελίδας, καθώς και τα routes και οι ανακατευθύνσεις που σχετίζονται με αυτό το module. Επίσης, μπορεί να περιλαμβάνει τα αριστερό και δεξί πλαίσιο της εφαρμογής, που είναι κοινά για το module.

- **Υπόλοιπα Components:** Στον ίδιο φάκελο μπορεί να υπάρχουν και άλλα components που χρησιμοποιούνται αποκλειστικά στις οθόνες του συγκεκριμένου module. Αυτά τα components μπορεί να αφορούν συγκεκριμένες λειτουργίες ή επιδείξεις πληροφοριών που αφορούν μόνο αυτό το module.

Με αυτόν τον τρόπο, ο κοινός κώδικας για τη διάταξη της σελίδας και τις ανακατευθύνσεις διαχειρίζεται μόνο μια φορά στο κοινό component, ενώ τα υπόλοιπα components είναι ειδικά για το συγκεκριμένο module και περιέχουν λειτουργίες και παρουσιάσεις πληροφοριών που αφορούν αποκλειστικά την εν λόγω ενότητα της εφαρμογής. Αυτή η δομή επιτρέπει την καλύτερη διαχείριση και συντήρηση του κώδικα, καθώς κάθε module έχει το δικό του ανεξάρτητο σύνολο components που αντικατοπτρίζει τις συγκεκριμένες ανάγκες του.

Στη συνέχεια παρατίθενται παραδείγματα των περιεχομένων των αρχείων του φακέλου *components*.

- components / Page
    - **components / Page / index.jsx**

```
export { default as Library } from './Library'
// Εξαγωγή του κεντρικού component του συγκεκριμένου module
```

    - **components / Page / Page.jsx**

```
import { lazy, Suspense } from 'react'
// Εισαγωγή των συναρτήσεων 'lazy' και 'Suspense' από το 'react'.
import { AppLayout, Loading } from 'modules/shared'
// Εισαγωγή των component 'AppLayout' και 'Loading' από το
'modules/shared'.
import LibraryLeftSidebar from './LibrarySidebar'
// Εισαγωγή του στοιχείου 'LibraryLeftSidebar' από το τρέχον τοπικό αρχείο
'LibrarySidebar'.
import LibraryRightSidebar from './LibraryNotebar'
// Εισαγωγή του στοιχείου 'LibraryRightSidebar' από το τρέχον τοπικό αρχείο
'LibraryNotebar'.

const LibraryRoutes = lazy(() => import('./LibraryRoutes')) // Δημιουργία
αργά φορτωμένου στοιχείου 'LibraryRoutes' από το τρέχον τοπικό αρχείο
'LibraryRoutes'.

const { Feature } = AppLayout // Ανάθεση της συνιστώσας 'Feature' από την
'AppLayout' στη μεταβλητή 'Feature'.
const { Left, Main, Right } = Feature // Ανάθεση των συνιστωσών 'Left',
'Main' και 'Right' από τη μεταβλητή 'Feature'.

function Library() { // Δήλωση της συνάρτησης 'Library'.
  return (
    <>
      <Left>
        <LibraryLeftSidebar /> {/* Εμφάνιση του στοιχείου
'LibraryLeftSidebar' στο αριστερό πλευρικό πλαίσιο. */}
      </Left>
      <Main>
```

```
        <Suspense fallback={<Loading />}> {/* Χρήση της 'Suspense' με
αναμονή και εμφάνιση του στοιχείου 'Loading' ως αντικατάσταση κατά τη
φόρτωση. */}
          <LibraryRoutes /> {/* Εμφάνιση του αργά φορτωμένου στοιχείου
'LibraryRoutes'. */}
        </Suspense>
      </Main>

      <Right>
        <LibraryRightSidebar /> {/* Εμφάνιση του στοιχείου
'LibraryRightSidebar' στο δεξί πλευρικό πλαίσιο. */}
      </Right>
    </>
  )
}

export default Library // Εξαγωγή της συνάρτησης 'Library' ως προεπιλογή.
```

- **components / Page / PageNotebar.jsx**

```
// Εισαγωγή του στοιχείο Notebar από τον φάκελο
'modules/shared/components/Notes'.
import { Notebar } from 'modules/shared/components/Notes'

// Δημιουργία της συνάρτησης LibraryRightSidebar.
function LibraryRightSidebar() {
  // Επιστροφή του στοιχείου Notebar ως περιεχόμενο της συνάρτησης.
  return <Notebar />
}

// Εξαγωγή της συνάρτησης LibraryRightSidebar ως προεπιλεγμένη εξαγωγή
αυτού του αρχείου.
export default LibraryRightSidebar
```

- **components / Page / PageRoutes.jsx**

```
import { lazy } from 'react' // Εισαγωγή της συνάρτησης 'lazy' από το
'react'.
import { Routes, Route, Navigate } from 'react-router-dom' // Εισαγωγή των
συνιστωσών 'Routes', 'Route' και 'Navigate' από το 'react-router-dom'.

const Library = lazy(() => import('modules/library/pages/Library')) //
Δημιουργία αργά φορτωμένου στοιχείου 'Library'.
const View = lazy(() => import('modules/library/pages/View')) // Δημιουργία
αργά φορτωμένου στοιχείου 'View'.
const Edit = lazy(() => import('modules/library/pages/Edit')) // Δημιουργία
αργά φορτωμένου στοιχείου 'Edit'.
const Create = lazy(() => import('modules/library/pages/Create')) //
Δημιουργία αργά φορτωμένου στοιχείου 'Create'.

function LibraryRoutes() { // Δήλωση της συνάρτησης 'LibraryRoutes'.
  return (
    <Routes> // Σύνθεση των δρομολογητών (routes) με την χρήση του
στοιχείου 'Routes'.
      <Route path="/" element={<Library />} /> // Δρομολόγηση στο στοιχείο
'Library' όταν η διεύθυνση URL είναι '/'.
```

```
      <Route path="/create" element={<Create />} /> // Δρομολόγηση στο
στοιχείο 'Create' όταν η διεύθυνση URL είναι '/create'.
      <Route path="/view/:id" element={<View />} /> // Δρομολόγηση στο
στοιχείο 'View' όταν η διεύθυνση URL περιέχει '/view/:id'.
      <Route path="/edit/:id" element={<Edit />} /> // Δρομολόγηση στο
στοιχείο 'Edit' όταν η διεύθυνση URL περιέχει '/edit/:id'.
      <Route path="*" element={<Navigate to="/library" replace />} /> //
Στην περίπτωση που η διεύθυνση URL δεν ταιριάζει σε κανένα από τα παραπάνω,
δρομολόγηση μέσω του 'Navigate' για την ανακατεύθυνση στη διεύθυνση
'/library'.
    </Routes>
  )
}

export default LibraryRoutes // Εξαγωγή της συνάρτησης 'LibraryRoutes' ως
προεπιλογή.
```

- **components / Page / PageSidebar.jsx**

```
import { useAuth } from 'modules/auth' // Εισαγωγή της συνάρτησης 'useAuth'
από το 'modules/auth'.
import { Sidebar, useBadges } from 'modules/shared' // Εισαγωγή των
στοιχείων 'Sidebar' και 'useBadges' από το 'modules/shared'.

function LibraryLeftSidebar() { // Δήλωση της συνάρτησης
'LibraryLeftSidebar'.
  const { profile, isHaf } = useAuth() // Χρήση της συνάρτησης 'useAuth'
και ανάθεση των τιμών 'profile' και 'isHaf'.
  const badges = useBadges() // Χρήση της συνάρτησης 'useBadges' και
ανάθεση των τιμών 'badges'.

  const sectionsHaf = [ // Δήλωση του πίνακα 'sectionsHaf' που περιέχει τα
στοιχεία του πλαισίου αριστερής πλευράς για τους χρήστες HAF (χρήστες
Πολεμικής Αεροπορίας).
    {
      icon: 'book-fill',
      label: 'Πρωτόκολλο',
      action: true,
      items: [
        {
          icon: 'arrow-down-left-square',
          label: 'Εισερχομένων',
          action: true,
          path: '/registry/inbound'
        },
        {
          icon: 'arrow-up-right-square',
          label: 'Εξερχομένων',
          action: true,
          path: '/registry/outbound'
        }
      ]
    },
    // Άλλα στοιχεία που προσαρμόζονται ανάλογα με το προφίλ του χρήστη
HAF.
  ]

  const sectionsGov = [ // Δήλωση του πίνακα 'sectionsGov' που περιέχει τα
στοιχεία του πλαισίου αριστερής πλευράς για τους χρήστες του δημοσίου.
```

```
    {
      icon: 'book-fill',
      label: 'Πρωτόκολλο',
      action: true,
      items: [
        {
          icon: 'arrow-down-left-square',
          label: 'Εισερχομένων',
          action: true,
          path: '/registry/inbound'
        },
        {
          icon: 'arrow-up-right-square',
          label: 'Εξερχομένων',
          action: true,
          path: '/registry/outbound'
        }
      ]
    },
    // Άλλα στοιχεία που προσαρμόζονται ανάλογα με το προφίλ του χρήστη του
δημοσίου.
  ]

  return (
    <Sidebar
      sections={
        import.meta.env.VITE_IS_HAF === 'true' ? sectionsHaf : sectionsGov
      } // Ανάθεση του πίνακα 'sectionsHaf' ή 'sectionsGov' ανάλογα με την
τιμή της μεταβλητής 'VITE_IS_HAF'.
      badges={badges} // Περάτωση των σημάτων στο στοιχείο 'Sidebar'.
    />
  )
}

export default LibraryLeftSidebar // Εξαγωγή της συνάρτησης
'LibraryLeftSidebar' ως προεπιλογή.
```

3.4.3 pages

Στον φάκελο του συγκεκριμένου module, οργανώνονται όλες οι σελίδες (οθόνες) που ανήκουν σε αυτό το module. Κάθε σελίδα αντιστοιχεί σε ένα διαφορετικό route του module και αποτελείται από τουλάχιστον δύο βασικά αρχεία: ένα αρχείο λειτουργικότητας (το οποίο περιέχει τη λογική της σελίδας) και ένα αρχείο που περιέχει τη διάταξη και την προβολή της σελίδας (χωρίς λογική).

Διευκρινίζοντας τον ρόλο και το περιεχόμενο αυτών των δύο αρχείων:

- **Αρχείο Λειτουργικότητας:**Αυτό το αρχείο περιέχει όλη τη λογική και τη λειτουργικότητα της αντίστοιχης σελίδας. Εδώ ορίζονται οι λειτουργίες που θα εκτελούνται όταν ο χρήστης αλληλεπιδρά με τη σελίδα, όπως αποθήκευση δεδομένων, επεξεργασία πληροφοριών ή προβολή δεδομένων από τη βάση δεδομένων. Αυτό το αρχείο περιλαμβάνει την "εγκέφαλο" της σελίδας.
- **Αρχείο Διάταξης και Προβολής:**Το δεύτερο αρχείο αναλαμβάνει την διάταξη και την προβολή της σελίδας, αλλά δεν περιλαμβάνει λογική. Συνήθως περιλαμβάνει το HTML - JSX και το CSS που απαιτούνται για την απεικόνιση των στοιχείων στην οθόνη,

καθώς και τυχόν σχεδιαστικά στοιχεία. Αυτό το αρχείο προσφέρει την "εμφάνιση" της σελίδας.

Τα πιθανά είδη σελίδων που μπορεί να περιλαμβάνει κάθε module (Create, Edit, View, Πίνακας) αναπτύσσουν τις διάφορες λειτουργίες που μπορούν να εκτελεστούν στο εκάστοτε module. Οργανώνονται με τη σκοπιά του να προσφέρουν στους χρήστες τη δυνατότητα δημιουργίας, επεξεργασίας, προβολής και επισκόπησης δεδομένων ανάλογα με τις ανάγκες του συγκεκριμένου module.

- pages / Create
  - pages / Create / assets
    - **pages / Create / assets / tourCreate.jsx**

```
const tourLibraryCreate = [
  {
    selector: '[data-tut="title"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'title'.
    content: 'Εδώ συμπληρώνετε τον οργανισμό της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="folder"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'folder'.
    content: 'Εδώ συμπληρώνετε το φάκελο της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="subject"]', // Ορισμός του επιλογέα (selector)
για το πεδίο 'subject'.
    content: 'Εδώ συμπληρώνετε το θέμα της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="classification"]', // Ορισμός του επιλογέα
(selector) για το πεδίο 'classification'.
    content: 'Εδώ συμπληρώνετε τη διαβάθμιση της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="type"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'type'.
    content: 'Εδώ συμπληρώνετε τον τύπο της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="remarks"]', // Ορισμός του επιλογέα (selector)
για το πεδίο 'remarks'.
    content: 'Εδώ συμπληρώνετε τις παρατηρήσεις σας' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="attachments"]', // Ορισμός του επιλογέα
(selector) για το πεδίο 'attachments'.
    content: 'Εδώ επισυνάπτετε αρχεία' // Εξήγηση του περιεχομένου της
οδηγίας.
  }
]
```

```
export default tourLibraryCreate // Εξαγωγή της σταθεράς
'tourLibraryCreate'.
```

- **pages / Create / Create.jsx**

```
import Tour from 'reactour'
// Εισαγωγή του component 'Tour' από τη βιβλιοθήκη 'reactour'.
import React from 'react'
// Εισαγωγή του 'React'.
import { Page, Tabs, Field, useFormValues, useTheme } from 'modules/shared'
// Εισαγωγή διαφόρων στοιχείων και hooks από το 'modules/shared'.
import CreateToolbar from './CreateToolbar'
// Εισαγωγή του στοιχείου 'CreateToolbar' από ένα τοπικό αρχείο.
import tourLibraryCreate from './assets/tourLibraryCreate'
// Εισαγωγή μιας διαμόρφωσης περιήγησης από ένα τοπικό αρχείο.
import LibraryAttachments from
'modules/library/components/LibraryAttachments'
// Εισαγωγή του στοιχείου 'LibraryAttachments' από το
'modules/library/components/LibraryAttachments'.
import { FolderSelect } from 'modules/documents/components/FolderSelect'
// Εισαγωγή του στοιχείου 'FolderSelect' από
'modules/documents/components/FolderSelect'.
import { useFolders } from
'modules/documents/components/FolderSelect/useFolders'
// Εισαγωγή του hook 'useFolders' από
'modules/documents/components/FolderSelect/useFolders'.
import clsx from 'clsx'
// Εισαγωγή της υπηρεσίας 'clsx' για τη δημιουργία ονομάτων κλάσεων CSS.

function Create({
  libraryDoc, // Ορισμός της ιδιότητας 'libraryDoc'.
  handleCreate, // Ορισμός της ιδιότητας 'handleCreate'.
  tourOpen, // Ορισμός της ιδιότητας 'tourOpen'.
  setTourOpen, // Ορισμός της ιδιότητας 'setTourOpen'.
  fname, // Ορισμός της ιδιότητας 'fname'.
  loading // Ορισμός της ιδιότητας 'loading'.
}) {
  const theme = useTheme('library.form')
// Χρήση του hook 'useTheme' για την απόκτηση ενός αντικειμένου θέματος.
  const libValues = useFormValues(fname)
 // Χρήση του hook 'useFormValues' για την ανάκτηση των τιμών φόρμας με
βάση το 'fname'.
  var libFolderSubject =
    useFolders().find((x) => x.id === libValues.folder) || ''
 // Εύρεση αντικειμένου φακέλου με βάση το 'libValues.folder'
χρησιμοποιώντας το hook 'useFolders'.
  // Ορισμός μεταβλητής ως πίνακα με τις καρτέλες και τα χαρακτηριστικά
τους.
  const tabs = [
    {
      name: 'details', // Όνομα καρτέλας 'details'.
      icon: 'file-text-fill', // Εικονίδιο για την καρτέλα.
      caption: 'Πληροφορίες', // Λεζάντα για την καρτέλα.
      fields: [
        'title',
        'folder',
        'subject',
        'classification',
```

```
        'type',
        'remarks',
        'attachments'
      ], // Λίστα πεδίων που σχετίζονται με αυτήν την καρτέλα.
      content: ( // JSX περιεχόμενο για την καρτέλα.
        <div className={clsx(...theme.classes)}> {/* Δημιουργία ενός div με
τα ονόματα κλάσεων CSS που δημιουργούνται από το 'clsx' */}
          <Field
            tour="title" // Παροχή ιδιότητας 'tour' για αυτό το πεδίο.
            form={fname} // Ορισμός ιδιότητας 'form' για αυτό το πεδίο.
            name="title" // Όνομα πεδίου 'title'.
            maxLength={250} // Μέγιστο μήκος για την είσοδο.
            type="text" // Τύπος εισόδου 'text'.
            labelCol={2} // Στήλη ετικέτας.
            label="Τίτλος" // Ετικέτα πεδίου.
            initialValue={libraryDoc.library.title} // Αρχική τιμή για την
είσοδο.
          />
          {/* Παρόμοια πεδία τύπου Field με διάφορες ιδιότητες */}
          {/* ... */}
        </div>
      )
    }
  ]
// Η επιστροφή της συνάρτησης που εμφανίζεται στον χρήστη
  return (
    <React.Fragment>
      {/* Απεικόνιση του στοιχείου 'Page', το οποίο έχει 'Page.Header',
'Page.Content', 'Page.Footer'  */}
      <Page>
        <Page.Header>
          {/* Στοιχεία κεφαλίδας σελίδας */}
          {/* ... */}
        </Page.Header>
        <Page.Content>
          {/* Απεικόνιση ενός στοιχείου Tabs με τις καρτέλες που έχουν
καθοριστεί */}
          <Tabs tabs={tabs} centered={true} formKey={fname} />
        </Page.Content>
        <Page.Footer></Page.Footer>
      </Page>
      {/* Απεικόνιση του στοιχείου 'Tour' */}
      <Tour
        isOpen={tourOpen} // Η ιδιότητα 'tourOpen' για τον έλεγχο του αν
είναι ανοικτή η περιήγηση - βοήθεια.
        rounded={5} // Ορισμός της τιμής για το στρογγυλότητα της
περιήγησης.
        steps={tourLibraryCreate} // Τα βήματα για την περιήγηση.
        onRequestClose={() => setTourOpen((prevState) => !prevState)} //
Χειρισμός του αιτήματος κλεισίματος της περιήγησης.
      />
    </React.Fragment>
  )
}
export default Create
// Εξαγωγή του στοιχείου 'Create'.
```

- **pages / Create / CreateToolbar.jsx**

```
import { useMemo } from 'react' // Εισαγωγή της συνάρτησης 'useMemo' από το
'react'.
import { Page, buttonOrdering, useForm } from 'modules/shared' // Εισαγωγή
διάφορων στοιχείων από 'modules/shared'.
import { schema } from './validation' // Εισαγωγή του σχήματος από το
'validation'.

function CreateToolbar({
  fname, // Ορισμός της ιδιότητας 'fname'.
  libraryDoc, // Ορισμός της ιδιότητας 'libraryDoc'.
  handleCreate, // Ορισμός της ιδιότητας 'handleCreate'.
  setTourOpen, // Ορισμός της ιδιότητας 'setTourOpen'.
  loading // Ορισμός της ιδιότητας 'loading'.
}) {
  const { submit } = useForm(fname, schema) // Ανάκτηση της συνάρτησης
'submit' και χρήση του 'useForm' με 'fname' και 'schema'.

  function handleSave() { // Ορισμός της συνάρτησης 'handleSave'.
    const values = submit() // Ανάκτηση τιμών μέσω της συνάρτησης 'submit'.
    if (values.errors) { // Αν υπάρχουν σφάλματα στις τιμές.
      return // Επιστροφή χωρίς εκτέλεση περαιτέρω ενεργειών.
    }
    handleCreate({ ...values }) // Κλήση της συνάρτησης 'handleCreate' με
τις τιμές ως παραμέτρους.
    console.debug('submit', values) // Εκτύπωση των τιμών στον πίνακα
αποσφαλμάτωσης.
  }

  const buttons = useMemo(() => { // Χρήση της συνάρτησης 'useMemo' για τη
δημιουργία των κουμπιών.
    const items = [ // Δημιουργία λίστας αντικειμένων κουμπιών.
      {
        key: 'save', // Κλειδί κουμπιού 'save'.
        icon: 'check2', // Εικονίδιο κουμπιού.
        title: 'Αποθήκευση', // Τίτλος κουμπιού.
        onClick: handleSave, // Συνάρτηση που εκτελείται κατά το κλικ.
        loading: loading // Κατάσταση φόρτωσης κουμπιού.
      },
      {
        key: 'tour', // Κλειδί κουμπιού 'tour'.
        icon: 'info-circle', // Εικονίδιο κουμπιού.
        title: 'Βοήθεια', // Τίτλος κουμπιού.
        onClick: () => setTourOpen((prevState) => !prevState) // Συνάρτηση
που εκτελείται κατά το κλικ.
      }
    ]
    return buttonOrdering({ items }) // Επιστροφή των κουμπιών με βάση τη
σειρά που καθορίζεται από το 'buttonOrdering'.
  }, [handleSave, loading, setTourOpen]) // Εξαρτήσεις που παρέχονται στη
συνάρτηση 'useMemo'.

  return (
    <Page.Header.Toolbar.Group> {/* Ομάδα εργαλείων στην κεφαλίδα σελίδας
*/}
      {buttons.map((btn) => ( // Χαρτογράφηση των κουμπιών και απεικόνισή
τους.
        <Page.Header.Toolbar.Button {...btn} /> // Απεικόνιση κουμπιού με
τις ιδιότητες από το αντίστοιχο αντικείμενο.
      ))}
    </Page.Header.Toolbar.Group>
  )
```

```
}

export default CreateToolbar
// Εξαγωγή του στοιχείου 'CreateToolbar'.
```

- **pages / Create / index.jsx**

```
import { withContainer, withErrorBoundary } from 'modules/shared'
// Εισαγωγή των στοιχείων 'withContainer' και 'withErrorBoundary' από το
'modules/shared'.
import useCreate from './useCreate'
// Εισαγωγή του 'useCreate' από το τοπικό αρχείο useCreate.
import Create from './Create'
// Εισαγωγή του 'Create' από το τοπικό αρχείο Create.

// Εξαγωγή του αποτελέσματος της συνάρτησης 'withErrorBoundary' με το
αποτέλεσμα της συνάρτησης 'withContainer'.
export default withErrorBoundary(withContainer(Create, useCreate))
```

- **page / Create / useCreate.jsx**

```
import { useState } from 'react' // Εισαγωγή της συνάρτησης 'useState' από
το 'react'.
import { useAuth } from 'modules/auth' // Εισαγωγή του 'useAuth' από το
'modules/auth'.
import { useNavigate } from 'react-router-dom' // Εισαγωγή της συνάρτησης
'useNavigate' από το 'react-router-dom'.
import { toastSuccess, useQuery, useCommand, useTitle } from
'modules/shared' // Εισαγωγή διάφορων στοιχείων από το 'modules/shared'.
import LibraryApi from 'modules/library/api/api' // Εισαγωγή της κλάσης
'LibraryApi' από 'modules/library/api/api'.

function useCreate() { // Δήλωση της συνάρτησης 'useCreate'.
  let navigate = useNavigate() // Χρήση της συνάρτησης 'useNavigate' για να
αποκτήσουμε την συνάρτηση 'navigate'.
  const [fname] = useState(() => 'library/create/' + Date.now().toString())
// Δήλωση του state 'fname' με αρχική τιμή.
  const [tourOpen, setTourOpen] = useState(false) // Δήλωση του state
'tourOpen' με αρχική τιμή false.
  const { token, profile } = useAuth() // Χρήση του hook 'useAuth' για
αποκτήσουμε τα δεδομένα του χρήστη.
  const query = useQuery( // Χρήση της συνάρτησης 'useQuery' με ορίσματα.
    ['library-new'], // Λίστα εξαρτήσεων για το query.
    async () =>
      await new LibraryApi(
        token,
        profile.department.id,
        profile.title.id
      ).queries.getNewLibrary() // Εκτέλεση του query για την ανάκτηση νέων
στοιχείων βιβλιοθήκης.
  )
  const command = useCommand( // Χρήση της συνάρτησης 'useCommand' με
ορίσματα.
    ['library-all'], // Λίστα εξαρτήσεων για το command.
    async (libraryDocument) =>
      await new LibraryApi(
```

```
        token,
        profile.department.id,
        profile.title.id
      ).commands.createLibraryDocument(libraryDocument) // Εκτέλεση της
εντολής για τη δημιουργία νέου εγγράφου βιβλιοθήκης.
  )

  function handleCreate(libraryDocument) { // Δήλωση της συνάρτησης
'handleCreate'.
    command.execute(libraryDocument, { // Εκτέλεση της εντολής με τον
σχετικό χειρισμό.
      onSuccess: () => {
        navigate('/library') // Πλοήγηση στη σελίδα '/library' μετά την
επιτυχή δημιουργία.
        toastSuccess('Η ανάρτηση δημιουργήθηκε') // Εμφάνιση επιτυχούς
μηνύματος.
      }
    })
  }

  useTitle('Βιβλιοθήκη - Δημιουργία ανάρτησης') // Χρήση της συνάρτησης
'useTitle' για τον τίτλο της σελίδας.

  return { // Επιστροφή αντικειμένου με τις σχετικές ιδιότητες.
    libraryDoc: query, // Τα δεδομένα που παρέχονται από το query.
    tourOpen, // Η κατάσταση του tour.
    setTourOpen, // Η συνάρτηση για τον έλεγχο της κατάστασης του tour.
    handleCreate, // Η συνάρτηση για τη δημιουργία ανάρτησης.
    fname, // Το όνομα φόρμας.
    loading: command.isLoading // Η κατάσταση φόρτωσης.
  }
}

export default useCreate // Εξαγωγή της συνάρτησης 'useCreate'.
```

- **pages / Create / validation.jsx**

```
import * as yup from 'yup' // Εισαγωγή όλων των εξαγόμενων μεταβλητών από
τη βιβλιοθήκη 'yup'.

// Δημιουργία του σχήματος επικύρωσης (validation schema) με χρήση της
βιβλιοθήκης 'yup'.
export const schema = yup.object().shape({
  title: yup
    .string() // Ορισμός του πεδίου 'title' ως string.
    .max(250, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 250') // Έλεγχος
μέγιστου μήκους.
    .required('Το πεδίο είναι υποχρεωτικό'), // Έλεγχος υποχρεωτικότητας.

  subject: yup
    .string() // Ορισμός του πεδίου 'subject' ως string.
    .max(250, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 250') // Έλεγχος
μέγιστου μήκους.
    .required('Το πεδίο είναι υποχρεωτικό') // Έλεγχος υποχρεωτικότητας.
    .min(5, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 5'), // Έλεγχος
ελάχιστου μήκους.

  attachments: yup
    .array() // Ορισμός του πεδίου 'attachments' ως πίνακας.
```

```
    .min(1, 'Η λίστα αρχείων δεν μπορεί να είναι κενή') // Έλεγχος
ελάχιστου μεγέθους πίνακα.
    .nullable() // Επιτρέπει την τιμή 'null'.
    .required('Η ύπαρξη τουλάχιστον ενός αρχείου είναι υποχρεωτικό'), //
Έλεγχος υποχρεωτικότητας.

  remarks: yup
    .string() // Ορισμός του πεδίου 'remarks' ως string.
    .max(500, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 500') // Έλεγχος
μέγιστου μήκους.
})
```

- pages / Edit
  - pages / Edit / assets
    - **pages / Edit / assets / tourEdit.jsx**

```
const tourLibraryEdit = [
  {
    selector: '[data-tut="title"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'title'.
    content: 'Εδώ συμπληρώνετε τον οργανισμό της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="folder"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'folder'.
    content: 'Εδώ συμπληρώνετε το φάκελο της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="subject"]', // Ορισμός του επιλογέα (selector)
για το πεδίο 'subject'.
    content: 'Εδώ συμπληρώνετε το θέμα της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="classification"]', // Ορισμός του επιλογέα
(selector) για το πεδίο 'classification'.
    content: 'Εδώ συμπληρώνετε τη διαβάθμιση της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="type"]', // Ορισμός του επιλογέα (selector) για
το πεδίο 'type'.
    content: 'Εδώ συμπληρώνετε τον τύπο της ανάρτησης' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="remarks"]', // Ορισμός του επιλογέα (selector)
για το πεδίο 'remarks'.
    content: 'Εδώ συμπληρώνετε τις παρατηρήσεις σας' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="attachments"]', // Ορισμός του επιλογέα
(selector) για το πεδίο 'attachments'.
    content: 'Εδώ επισυνάπτετε αρχεία' // Εξήγηση του περιεχομένου της
οδηγίας.
  }
```

```
]

export default tourLibraryEdit // Εξαγωγή της σταθεράς 'tourLibraryEdit'.
```

- **pages / Edit / Edit.jsx**

```
import Tour from 'reactour'; // Εισαγωγή του Tour από το πακέτο 'reactour'
import React from 'react'; // Εισαγωγή της React

import tourLibraryEdit from
'modules/library/pages/Edit/assets/tourLibraryEdit'; // Εισαγωγή του
tourLibraryEdit από την συγκεκριμένη διαδρομή
import { Page, Tabs, Field, useFormValues, useTheme } from
'modules/shared'; // Εισαγωγή πολλών στοιχείων από το 'modules/shared'
import EditToolbar from './EditToolbar'; // Εισαγωγή του EditToolbar από το
τρέχον φάκελο που περιέχει το αρχείο 'EditToolbar.jsx'
import LibraryAttachments from
'modules/library/components/LibraryAttachments'; // Εισαγωγή του
LibraryAttachments από το 'modules/library/components/LibraryAttachments'
import { FolderSelect } from 'modules/documents/components/FolderSelect';
// Εισαγωγή του FolderSelect από το
'modules/documents/components/FolderSelect'
import { useFolders } from
'modules/documents/components/FolderSelect/useFolders'; // Εισαγωγή της
useFolders από το 'modules/documents/components/FolderSelect/useFolders'
import clsx from 'clsx'; // Εισαγωγή της clsx

function Edit({ fname, libraryDoc, handleUpdate, tourOpen, setTourOpen,
loading }) {
  const theme = useTheme('library.form'); // Χρήση της useTheme για την
θέσπιση του θέματος
  const libValues = useFormValues(fname); // Χρήση της useFormValues για
την ανάγνωση των τιμών της φόρμας
  var libFolderSubject = useFolders().find((x) => x.id ===
libValues.folder) || ''; // Βρίσκει το φάκελο της βιβλιοθήκης

  const tabs = [
    {
      name: 'details', // Όνομα καρτέλας
      icon: 'file-text-fill', // Εικονίδιο καρτέλας
      caption: 'Πληροφορίες', // Ετικέτα καρτέλας
      fields: [
        'title',
        'folder',
        'subject',
        'classification',
        'type',
        'remarks',
        'attachments'
      ], // Πεδία καρτέλας
      content: (
        <div className={clsx(...theme.classes)}>
          <Field
            tour="title"
            form={fname}
            name="title"
            type="text"
            maxLength={250}
            labelCol={2}
```

```
            label="Τίτλος"
            initialValue={libraryDoc.title}
          />
          {/* Εμφάνιση πεδίου τίτλου */}
          <Field
            tour="folder"
            form={fname}
            name="folder"
            labelCol={2}
            label="Φάκελος"
            initialValue={libraryDoc.folder}
            Component={FolderSelect}
          />
          {/* Εμφάνιση πεδίου φακέλου */}
          <Field
            tour="subject"
            form={fname}
            name="subject"
            type="textarea"
            labelCol={2}
            maxLength={250}
            label="Θέμα"
            initialValue={
              libValues.folder !== libraryDoc.folder
                ? libFolderSubject.description
                : libraryDoc.subject
            }
          />
          {/* Εμφάνιση πεδίου θέματος */}
          <Field
            tour="classification"
            form={fname}
            name="classification"
            type="select"
            options={libraryDoc.classifications}
            labelCol={2}
            label="Διαβάθμιση"
            initialValue={libraryDoc.classification}
          />
          {/* Εμφάνιση πεδίου διαβάθμισης */}
          <Field
            tour="type"
            form={fname}
            name="type"
            type="select"
            options={libraryDoc.types}
            labelCol={2}
            label="Τύπος"
            initialValue={libraryDoc.type}
          />
          {/* Εμφάνιση πεδίου τύπου */}
          <Field
            tour="remarks"
            form={fname}
            name="remarks"
            type="textarea"
            maxLength={500}
            labelCol={2}
            label="Σημειώσεις"
            initialValue={libraryDoc.remarks}
          />
```

```
          {/* Εμφάνιση πεδίου σημειώσεων */}
          <Field
            form={fname}
            name="attachments"
            labelCol={2}
            initialValue={libraryDoc.attachments}
            Component={LibraryAttachments}
          />
          {/* Εμφάνιση πεδίου συνημμένων */}
        </div>
      )
    }
  ];

  return (
    <React.Fragment>
      <Page>
        <Page.Header>
          <Page.Header.Title
            folder="Βιβλιοθήκη"
            path="/library"
            title="Επεξεργασία ανάρτησης"
          />
          {/* Επικεφαλίδα σελίδας */}
          <Page.Header.Toolbar>
            <EditToolbar
              libraryDoc={libraryDoc}
              handleUpdate={handleUpdate}
              setTourOpen={setTourOpen}
              fname={fname}
              loading={loading}
            />
            {/* Στοιχείο Toolbar για την επεξεργασία */}
          </Page.Header.Toolbar>
        </Page.Header>
        <Page.Content>
          <Tabs tabs={tabs} centered={true} formKey={fname} />
          {/* Καρτέλες σελίδας */}
        </Page.Content>
        <Page.Footer></Page.Footer>
      </Page>
      <Tour
        isOpen={tourOpen}
        rounded={5}
        steps={tourLibraryEdit}
        onRequestClose={() => setTourOpen((prevState) => !prevState)}
      />
      {/* Τουρ περιήγησης */}
    </React.Fragment>
  );
}

export default Edit;
```

- **pages / Edit / EditToolbar.jsx**

```
import { useMemo } from 'react'; // Εισαγωγή της useMemo από το React
import { Page, useForm, buttonOrdering } from 'modules/shared'; // Εισαγωγή
πολλών στοιχείων από το 'modules/shared'
```

```
import { schema } from './validation'; // Εισαγωγή του schema από τον τρέχον φάκελο

function EditToolbar({
  fname,
  libraryDoc,
  handleUpdate,
  setTourOpen,
  loading
}) {
  const { submit } = useForm(fname, schema); // Χρήση της useForm για τη διαχείριση της φόρμας και του schema για τον έλεγχο της εγκυρότητας των δεδομένων

  function handleSave() {
    const values = submit(); // Κλήση της συνάρτησης submit για την υποβολή των δεδομένων
    if (values.errors) {
      return;
    }
    handleUpdate({ ...libraryDoc, ...values }); // Κλήση της handleUpdate για την αποθήκευση των αλλαγών
    console.debug('submit', values);
  }

  const buttons = useMemo(() => {
    const items = [
      {
        key: 'save', // Κλειδί για το κουμπί αποθήκευσης
        icon: 'check2', // Εικονίδιο
        title: 'Αποθήκευση', // Τίτλος
        onClick: handleSave, // Συνάρτηση onClick
        loading: loading // Φόρτωση κατά την αποθήκευση
      },
      {
        key: 'tour', // Κλειδί κουμπιού 'tour'.
        icon: 'info-circle', // Εικονίδιο κουμπιού.
        title: 'Βοήθεια', // Τίτλος κουμπιού.
        onClick: () => setTourOpen((prevState) => !prevState) // Συνάρτηση που εκτελείται κατά το κλικ.
      }
    ]
    return buttonOrdering({ items }) // Επιστροφή των κουμπιών με βάση τη σειρά που καθορίζεται από το 'buttonOrdering'.
  }, [handleSave, loading, setTourOpen]) // Εξαρτήσεις που παρέχονται στη συνάρτηση 'useMemo'.

  return (
    <Page.Header.Toolbar.Group> {/* Ομάδα εργαλείων στην κεφαλίδα σελίδας */}
      {buttons.map((btn) => ( // Χαρτογράφηση των κουμπιών και απεικόνισή τους.
        <Page.Header.Toolbar.Button {...btn} /> // Απεικόνιση κουμπιού με τις ιδιότητες από το αντίστοιχο αντικείμενο.
      ))}
    </Page.Header.Toolbar.Group>
  )
}

export default EditToolbar
// Εξαγωγή του στοιχείου 'EditToolbar'.
```

- **pages / Edit / index.jsx**

```
import { withContainer, withErrorBoundary } from 'modules/shared'; //
Εισαγωγή πολλών στοιχείων από το 'modules/shared'
import useEdit from './useEdit'; // Εισαγωγή της useEdit από τον τρέχον
φάκελο
import Edit from './Edit'; // Εισαγωγή του Edit από τον τρέχον φάκελο

export default withErrorBoundary(withContainer(Edit, useEdit));
```

- **pages / Edit / useEdit.jsx**

```
import { useState } from 'react'; // Εισαγωγή της useState από το React
import { useAuth } from 'modules/auth'; // Εισαγωγή της useAuth από το
'modules/auth'
import { useParams, useNavigate } from 'react-router-dom'; // Εισαγωγή των
useParams και useNavigate από το 'react-router-dom'

import { toastSuccess, useQuery, useCommand, useTitle } from
'modules/shared'; // Εισαγωγή πολλών στοιχείων από το 'modules/shared'
import LibraryApi from 'modules/library/api/api'; // Εισαγωγή του
LibraryApi από το 'modules/library/api/api'

function useEdit() {
  let navigate = useNavigate(); // Χρήση της useNavigate για την πλοήγηση
σε διαφορετικές διαδρομές
  const { id } = useParams(); // Λήψη της παραμέτρου `id` από το URL
  const [fname] = useState(() => `library/edit/${id}` +
Date.now().toString()); // Δημιουργία μοναδικού ονόματος φόρμας με το `id`
  const [tourOpen, setTourOpen] = useState(false); // Αρχικοποίηση
μεταβλητής για την κατάσταση της περιήγησης (περιοδείας)
  const { token, profile } = useAuth(); // Χρήση της useAuth για τη λήψη
του τόκεν και του προφίλ χρήστη
  const query = useQuery(
    ['library-id', id],
    async () =>
      await new LibraryApi(
        token,
        profile.department.id,
        profile.title.id
      ).queries.getLibraryDocumentById(id)
  ); // Χρήση της useQuery για την ανάκτηση των δεδομένων της βιβλιοθήκης
βάσει του `id`

  const command = useCommand(
    ['library-all', 'library-id', 'library-view'],
    async (libraryDocument) =>
      await new LibraryApi(
        token,
        profile.department.id,
        profile.title.id
      ).commands.updateLibraryDocument(id, libraryDocument)
  ); // Χρήση της useCommand για την αποθήκευση των αλλαγών στη βιβλιοθήκη

  function handleUpdate(libraryDocument) {
```

```
    command.execute(libraryDocument, {
      onSuccess: () => {
        navigate(`/library/view/${id}`); // Πλοήγηση στη σελίδα προβολής
της βιβλιοθήκης μετά την ενημέρωση
        toastSuccess('Η ανάρτηση ενημερώθηκε'); // Εμφάνιση επιτυχούς
μηνύματος
      }
    });
  }

  useTitle('Βιβλιοθήκη - Επεξεργασία ανάρτησης'); // Αλλαγή του τίτλου της
σελίδας

  return {
    libraryDoc: query.data, // Δεδομένα της βιβλιοθήκης
    tourOpen, // Κατάσταση περιήγησης
    setTourOpen, // Συνάρτηση για την αλλαγή της κατάστασης περιήγησης
    handleUpdate, // Συνάρτηση για την ενημέρωση της βιβλιοθήκης
    fname, // Όνομα φόρμας
    loading: command.isLoading // Κατάσταση φόρτωσης
  };
}

export default useEdit;
// Εξαγωγή του useEdit
```

- **pages / Edit / validation.jsx**

```
import * as yup from 'yup'; // Εισαγωγή όλων των εξαγωγέων από το yup

export const schema = yup.object().shape({
  title: yup
    .string()
    .max(250, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 250')
    .required('Το πεδίο είναι υποχρεωτικό'), // Καθορισμός του σχήματος του
πεδίου "title"
  subject: yup
    .string()
    .max(250, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 250')
    .required('Το πεδίο είναι υποχρεωτικό')
    .min(5, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 5'), // Καθορισμός
του σχήματος του πεδίου "subject"
  attachments: yup
    .array()
    .min(1, 'Η λίστα αρχείων δεν μπορεί να είναι κενή')
    .nullable()
    .required('Η ύπαρξη τουλάχιστον ενός αρχείου είναι υποχρεωτικό'), //
Καθορισμός του σχήματος του πεδίου "attachments"
  remarks: yup
    .string()
    .max(500, 'Το πεδίο έχει μέγιστο όριο χαρακτήρων των 500') //
Καθορισμός του σχήματος του πεδίου "remarks"
});
```

- pages / View
  - **pages / View / components**
  - **pages / View / index.jsx**

```
import View from './View'; // Εισαγωγή του component View από τον τρέχον
φάκελο
import { withContainer, withErrorBoundary } from 'modules/shared'; //
Εισαγωγή των withContainer και withErrorBoundary από το 'modules/shared'
import useViewLibrary from './useViewLibrary'; // Εισαγωγή της συνάρτησης
useViewLibrary από τον τρέχον φάκελο

export default withErrorBoundary(withContainer(View, useViewLibrary)); //
Εξαγωγή του component View με τη χρήση των withContainer και
withErrorBoundary
```

- **pages / View / useViewLibrary.jsx**

```
import { useMemo } from 'react'; // Εισαγωγή της useMemo από το React
import { useParams } from 'react-router-dom'; // Εισαγωγή της useParams από
το 'react-router-dom'
import { useNavigate } from 'react-router-dom'; // Εισαγωγή της useNavigate
από το 'react-router-dom'
import { useAuth } from 'modules/auth'; // Εισαγωγή της useAuth από το
'modules/auth'
import {
  toastSuccess,
  useQuery,
  useCommand,
  useTitle,
  buttonOrdering
} from 'modules/shared'; // Εισαγωγή πολλών στοιχείων από το
'modules/shared'
import LibraryApi from 'modules/library/api/api'; // Εισαγωγή του
LibraryApi από το 'modules/library/api/api'

function useViewLibrary() {
  let navigate = useNavigate(); // Χρήση της useNavigate για την πλοήγηση
σε διαφορετικές διαδρομές
  const { id } = useParams(); // Λήψη της παραμέτρου `id` από το URL
  const { token, profile } = useAuth(); // Χρήση της useAuth για τη λήψη
του τόκεν - αναγνωριστικού και του προφίλ χρήστη

  const query = useQuery(
    ['library-view', id],
    async () =>
      await new LibraryApi(
        token,
        profile.department.id,
        profile.title.id
      ).queries.getViewLibraryDocumentById(id)
  ); // Χρήση της useQuery για την ανάκτηση των δεδομένων της βιβλιοθήκης
βάσει του `id`

  const command = useCommand(
    ['library-all'],
    async (id) =>
      await new LibraryApi(
        token,
        profile.department.id,
        profile.title.id
      ).commands.deleteLibraryDocument(id)
```

```
  ); // Χρήση της useCommand για τη διαγραφή της βιβλιοθήκης βάσει του `id`

  function handleDelete(id) {
    command.execute(id, {
      onSuccess: () => {
        navigate('/library'); // Πλοήγηση στην αρχική σελίδα της
βιβλιοθήκης μετά τη διαγραφή
        toastSuccess('Η ανάρτηση διαγράφηκε'); // Εμφάνιση επιτυχούς
μηνύματος
      }
    });
  }

  const buttons = useMemo(() => { //Ορισμός των κουμπιών που θα
χρησιμοποιηθούν στο toolbar
    const items = [
      {
        id: 'editLibraryDoc',
        icon: 'pencil-square',
        onClick: () => navigate(`/library/edit/${id}`),
        tooltip: 'Επεξεργασία Εγγράφου Βιβλιοθήκης'
      },
      {
        id: 'deleteLibraryDoc',
        icon: 'trash',
        onClick: () => handleDelete(id),
        tooltip: 'Διαγραφή Εγγράφου Βιβλιοθήκης',
        color: 'danger',
        loading: command.isLoading
      }
    ];
    return buttonOrdering({ items });
  }, [navigate, handleDelete, command]);

  useTitle(`Βιβλιοθήκη -` + query.data.title); // Αλλαγή του τίτλου της
σελίδας

  return {
    libraryDoc: query.data, // Δεδομένα της βιβλιοθήκης
    buttons: buttons // Κουμπιά ενέργειας
  };
}

export default useViewLibrary; //Εξαγωγή του hook useViewLibrary
```

- **pages / View / View.jsx**

```
import { Page, Tabs } from 'modules/shared'; // Εισαγωγή των Page και Tabs
από το 'modules/shared'
import { LibraryInfo } from './components'; // Εισαγωγή του LibraryInfo από
τον τρέχον φάκελο

function View({ buttons, libraryDoc }) {
  const tabs = [
    {
      name: 'details',
      icon: 'info-circle',
      caption: 'Πληροφορίες',
```

```
      content: <LibraryInfo data={libraryDoc} /> // Καθορισμός του
περιεχομένου της καρτέλας "Πληροφορίες"
    }
  ];
  return (
    <Page>
      <Page.Header>
        <Page.Header.Title
          folder="Βιβλιοθήκη"
          path="/library"
          title="Προβολή Εγγράφου Βιβλιοθήκης"
        /> {/* Καθορισμός του τίτλου της σελίδας */}
        <Page.Header.Toolbar>
          <Page.Header.Toolbar.Group>
            {buttons.map((b, idx) => {
              return <Page.Header.Toolbar.Button {...b} key={idx} />; //
Δημιουργία και απεικόνιση των κουμπιών ενέργειας
            })}
          </Page.Header.Toolbar.Group>
        </Page.Header.Toolbar>
      </Page.Header>
      <Page.Content>
        <Tabs tabs={tabs} centered={true} border="" /> {/* Απεικόνιση των
καρτελών με το περιεχόμενό τους */}
      </Page.Content>
    </Page>
  );
}

export default View; //Εξαγωγή του στοιχείου View
```

- PageTable
  - pages / pageTable / assets
    - **pages / pageTable / tourPageTable.jsx**

```
const tourLibrary = [
  {
    selector: '[data-tut="classification"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει την διαβάθμιση της ανάρτησης. Κάντε κλικ πάνω
στην κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="type"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει τον τύπο της ανάρτησης. Κάντε κλικ πάνω στην
κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.'// Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="title"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει τον τίτλο της ανάρτησης. Κάντε κλικ πάνω στην
κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="subject"]', //Ορισμός του επιλογέα
```

```
    content:
      'Η στήλη περιγράφει το θέμα της ανάρτησης. Κάντε κλικ πάνω στην
κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="publisher"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει τον συντάκτη της ανάρτησης. Κάντε κλικ πάνω στην
κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="folder"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει το φάκελο της ανάρτησης. Κάντε κλικ πάνω στην
κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.' // Εξήγηση του
περιεχομένου της οδηγίας.
  },
  {
    selector: '[data-tut="publicationDate"]', //Ορισμός του επιλογέα
    content:
      'Η στήλη περιγράφει την Ημ. Έκδοσης της ανάρτησης. Κάντε κλικ πάνω
στην κεφαλίδα της στήλης για να ταξινομήσετε την στήλη.'
  } // Εξήγηση του περιεχομένου της οδηγίας.
];

export default tourLibrary;
```

- **pages / PageTable / index.jsx**

```
import { withContainer, withErrorBoundary } from 'modules/shared'; //
Εισαγωγή των συναρτήσεων 'withContainer' και 'withErrorBoundary' από το
'modules/shared'.
import useLibrary from './useLibrary'; // Εισαγωγή του custom hook
'useLibrary' από τον τρέχοντα κατάλογο.
import Library from './Library'; // Εισαγωγή της συνιστώσας 'Library' από
τον τρέχοντα κατάλογο.

export default withErrorBoundary(withContainer(Library, useLibrary)); //
Εξαγωγή του στοιχείου 'Library' με την εφαρμογή των συναρτήσεων
'withContainer' και 'withErrorBoundary' πάνω σε αυτό, χρησιμοποιώντας το
custom hook 'useLibrary'.
```

- **pages / Page / PageTable.jsx**

```
import React from 'react' // Εισαγωγή του πακέτου 'react'.
import Tour from 'reactour' // Εισαγωγή του πακέτου 'reactour'.
import { Page, Pagination, Filter } from 'modules/shared' // Εισαγωγή των
στοιχείων 'Page', 'Pagination', και 'Filter' από το 'modules/shared'.
import LibraryTable from './LibraryTable' // Εισαγωγή του στοιχείου
'LibraryTable' από τον τρέχοντα κατάλογο.
import tourLibrary from './assets/tourLibrary' // Εισαγωγή του στοιχείου
Tour της βιβλιοθήκης από τον τρέχοντα κατάλογο.
```

```
function Library({ // Δήλωση της συνάρτησης 'Library' με τις παραμέτρους
που λαμβάνει.
  libraryDocuments, // Τα έγγραφα της βιβλιοθήκης.
  pagination, // Πληροφορίες σχετικά με τη σελιδοποίηση.
  buttons, // Τα κουμπιά της σελίδας.
  setTourOpen, // Συνάρτηση για τον έλεγχο της μεταβλητής 'tourOpen'.
  tourOpen // Η τιμή της μεταβλητής 'tourOpen'.
}) {
  return (
    <React.Fragment> {/* Ένα άδειο στοιχείο 'React.Fragment' για να
ενώσουμε πολλά στοιχεία χωρίς να δημιουργούμε ένα περιττό DOM στοιχείο. */}
      <Page> {/* Στοιχείο 'Page' για τη διαχείριση της σελίδας. */}
        <Page.Header> {/* Κεφαλίδα της σελίδας. */}
          <Page.Header.Title
            folder="Εργαλεία"
            path="/tools"
            title="Βιβλιοθήκη" // Ο τίτλος της σελίδας.
          />
          <Page.Header.Toolbar> {/* Εργαλεία στην κεφαλίδα της σελίδας. */}
            <Page.Header.Toolbar.Group> {/* Ομάδα εργαλείων. */}
              {buttons.map((b, idx) => { // Τα κουμπιά από τον πίνακα
κουμπιών.
                return <Page.Header.Toolbar.Button {...b} key={idx} />
              })}
            </Page.Header.Toolbar.Group>
          </Page.Header.Toolbar>
          <Filter /> {/* Στοιχείο 'Filter' για το φίλτρο. */}
        </Page.Header>
        <Page.Content> {/* Περιεχόμενο της σελίδας. */}
          <LibraryTable libraryDocuments={libraryDocuments} /> {/* Στοιχείο
'LibraryTable' για την προβολή των εγγράφων της βιβλιοθήκης. */}
        </Page.Content>
        <Page.Footer> {/* Υποσέλιδο της σελίδας. */}
          <Pagination // Στοιχείο 'Pagination' για την διαχείριση της
σελιδοποίησης.
            total={pagination.totalRecords} // Συνολικός αριθμός εγγραφών.
            pages={pagination.totalPages} // Συνολικές σελίδες.
          />
        </Page.Footer>
      </Page>
      <Tour // Συνιστώσα 'Tour' για την ξενάγηση.
        isOpen={tourOpen} // Εμφάνιση ή απόκρυψη της ξενάγησης με βάση την
τιμή της μεταβλητής 'tourOpen'.
        rounded={5} // Στυλ του παραθύρου ξενάγησης.
        steps={tourLibrary} // Τα βήματα της ξενάγησης που ορίζονται στον
πίνακα 'tourLibrary'.
        onRequestClose={() => setTourOpen((prevState) => !prevState)} //
Συνάρτηση για το κλείσιμο της ξενάγησης.
      />
    </React.Fragment>
  )
}

export default Library // Εξαγωγή του στοιχείου 'Library' ως προεπιλογή.
```

- **pages / Page / PageTableTable.jsx**

```
import { useNavigate } from 'react-router-dom' // Εισαγωγή της συνάρτησης
'useNavigate' από το 'react-router-dom'.
```

```
import { Table, TableMobile, useIsMobile } from 'modules/shared' //
Εισαγωγή των συνιστωσών 'Table', 'TableMobile', και 'useIsMobile' από το
'modules/shared'.

function LibraryTable({ libraryDocuments }) { // Δήλωση της συνάρτησης
'LibraryTable' που δέχεται την παράμετρο 'libraryDocuments'.
  const isMobile = useIsMobile() // Χρήση της συνάρτησης 'useIsMobile' και
ανάθεση της τιμής στη μεταβλητή 'isMobile'.
  const { Head, HeadRow, HeadCell, Body, Row, Cell } = isMobile // Ανάθεση
των στοιχείων του πίνακα (Head, HeadRow, HeadCell, Body, Row, Cell) ανάλογα
με το αν είναι κινητή συσκευή (isMobile).
    ? TableMobile
    : Table

  const navigate = useNavigate() // Χρήση της συνάρτησης 'useNavigate' και
ανάθεση της τιμής στη μεταβλητή 'navigate'.
  return (
    <Table> {/* Ρυθμίσεις πίνακα */}
      <Head> {/* Περιοχή κεφαλίδας πίνακα */}
        <HeadRow> {/* Σειρά κεφαλίδας πίνακα */}
          <HeadCell
            tour="classification"
            sortable="Classification"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Διαβάθμιση">
            Δ {/* Κείμενο κεφαλίδας */}
          </HeadCell>
          <HeadCell
            tour="type"
            sortable="Type"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Τύπος">
            Τύπος
          </HeadCell>
          <HeadCell
            tour="title"
            sortable="Title"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Τίτλος">
            Τίτλος
          </HeadCell>
          <HeadCell
            tour="subject"
            sortable="Subject"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Θέμα"
            minimumWidth={true}>
            Θέμα
          </HeadCell>
          <HeadCell
            tour="publisher"
            sortable="Publisher"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Συντάκτης">
            Συντάκτης
          </HeadCell>
          <HeadCell
            tour="folder"
            sortable="Folder"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Φάκελος">
            Φάκελος
          </HeadCell>
          <HeadCell
            tour="publicationDate"
            sortable="PublicationDate"
            title="Κάντε κλικ για ταξινόμηση με τη στήλη Ημ. Έκδοσης">
```

```
            Ημ. Έκδοσης
          </HeadCell>
        </HeadRow>
      </Head>
      <Body> {/* Περιοχή σώματος πίνακα */}
        {libraryDocuments.map((item) => ( // Αντιστοίχισης για κάθε εγγραφή
της βιβλιοθήκης.
          <Row
            key={item.id} // Κλειδί για την αναγνώριση κάθε εγγραφής.
            navigator={true} // Επιτρέπει την πλοήγηση προς την εγγραφή
όταν γίνει κλικ.
            onClick={() => navigate(`/library/view/${item.id}`)}> {/*
Πλοήγηση προς τη σελίδα της εγγραφής όταν γίνει κλικ. */}
            <Cell>{item.classification}</Cell> {/* Κελί με την ταξινόμηση.
*/}
            <Cell>{item.type}</Cell> {/* Κελί με τον τύπο. */}
            <Cell>{item.title}</Cell> {/* Κελί με τον τίτλο. */}
            <Cell>{item.subject}</Cell> {/* Κελί με το θέμα. */}
            <Cell>{item.publisher}</Cell> {/* Κελί με τον συντάκτη. */}
            <Cell>{item.folder}</Cell> {/* Κελί με το φάκελο. */}
            <Cell>{item.publicationDate}</Cell> {/* Κελί με την ημερομηνία
έκδοσης. */}
          </Row>
        ))}
      </Body>
    </Table>
  )
}

export default LibraryTable // Εξαγωγή της συνάρτησης 'LibraryTable' ως
προεπιλογή.
```

- **pages / PageTable / usePageTable.jsx**

```
import { useNavigate } from 'react-router-dom' // Εισαγωγή της συνάρτησης
'useNavigate' από το 'react-router-dom'.
import { useState, useMemo } from 'react' // Εισαγωγή των συναρτήσεων
'useState' και 'useMemo' από το 'react'.

import { useAuth } from 'modules/auth' // Εισαγωγή της συνάρτησης 'useAuth'
από το 'modules/auth'.
import {
  useSearchParams,
  useQuery,
  useTitle,
  buttonOrdering
} from 'modules/shared' // Εισαγωγή των συναρτήσεων 'useSearchParams',
'useQuery', 'useTitle', και 'buttonOrdering' από το 'modules/shared'.
import LibraryApi from 'modules/library/api/api' // Εισαγωγή της κλάσης
'LibraryApi' από το 'modules/library/api/api'.

function useLibrary() { // Δήλωση της συνάρτησης 'useLibrary'.
  const navigate = useNavigate() // Χρήση της συνάρτησης 'useNavigate' και
ανάθεση της τιμής στη μεταβλητή 'navigate'.
  const [tourOpen, setTourOpen] = useState(false) // Αρχικοποίηση της
μεταβλητής 'tourOpen' με την τιμή 'false'.

  const { token, profile } = useAuth() // Χρήση της συνάρτησης 'useAuth'
και ανάθεση των τιμών 'token' και 'profile'.
```

```
  const [params] = useSearchParams() // Χρήση της συνάρτησης
'useSearchParams' και ανάθεση της τιμής 'params'.
  const searchParams = new URLSearchParams(params).toString() // Μετατροπή
των παραμέτρων αναζήτησης σε συμβολοσειρά.
  const query = useQuery( // Χρήση της συνάρτησης 'useQuery'.
    ['library-all', searchParams], // Παράμετροι της αναζήτησης (cacheKey,
searchParams).
    async () =>
      await new LibraryApi( // Δημιουργία αντικειμένου 'LibraryApi'.
        token,
        profile.department.id,
        profile.title.id
      ).queries.getLibraryDocuments(searchParams) // Κλήση της μεθόδου
'getLibraryDocuments' της 'LibraryApi'.
  )
  const buttons = useMemo(() => { // Χρήση της συνάρτησης 'useMemo'.
    const items = [ // Πίνακας με τα κουμπιά.
      {
        id: 'new',
        disabled: false,
        icon: 'plus-square',
        onClick: () => navigate('/library/create'), // Συνάρτηση κλικ που
πλοηγεί τον χρήστη στη σελίδα δημιουργίας νέας ανάρτησης.
        title: 'Δημιουργία Νέας ανάρτησης'
      },
      {
        id: 'help',
        disabled: false,
        icon: 'info-circle',
        title: 'Βοήθεια',
        onClick: () => setTourOpen((prevState) => !prevState) // Συνάρτηση
κλικ που εναλλάσσει την τιμή της μεταβλητής 'tourOpen'.
      }
    ]
    return buttonOrdering({ items }) // Επιστροφή του πίνακα κουμπιών με τη
σωστή σειρά.
  }, [navigate, setTourOpen]) // Το 'useMemo' θα επαναλάβει τον υπολογισμό
όταν αλλάξει η τιμή των εξαρτώμενων μεταβλητών.

  useTitle('Βιβλιοθήκη') // Ορισμός του τίτλου της σελίδας χρησιμοποιώντας
τη συνάρτηση 'useTitle'.

  return { // Επιστροφή ενός αντικειμένου με τις απαραίτητες πληροφορίες.
    buttons, // Τα κουμπιά της σελίδας.
    libraryDocuments: query.data, // Τα έγγραφα της βιβλιοθήκης που
προέρχονται από το αποτέλεσμα της αναζήτησης.
    tourOpen, // Η τιμή της μεταβλητής 'tourOpen'.
    setTourOpen, // Η συνάρτηση για τον έλεγχο της μεταβλητής 'tourOpen'.
    pagination: query.pagination // Πληροφορίες σχετικά με την σελιδοποίηση
των αποτελεσμάτων της αναζήτησης.
  }
}

export default useLibrary // Εξαγωγή της συνάρτησης 'useLibrary' ως
προεπιλογή.
```

3.4.4 index.jsx

Το συγκεκριμένο αρχείο χρησιμοποιείται ώστε να μπορεί να εξάγεται ως ενα το συγκεκριμένο module. Έτσι αυτό το αρχείο μπορεί να εξάγει το module με έναν τρόπο που το καθιστά διαθέσιμο για χρήση ως μονάδα.

```
import { Library } from './components/Library'
// Εισαγωγή του κεντρικού στοιχείου Library από το φάκελο
components/Library

export default Library
// Εξαγωγή όλου του συγκεκριμένου module
```

3.5 Ανάλυση των dependencies της React App

Στη συνέχεια παρέχεται μια ολοκληρωμένη ανάλυση των πακέτων που χρησιμοποιούνται σε μια εφαρμογή React. Οι εξεταζόμενες εξαρτήσεις - dependencies καλύπτουν ένα ευρύ φάσμα λειτουργιών, από τον σχεδιασμό του UI και τις βελτιώσεις της εμπειρίας χρήστη έως τις δοκιμές και τη διαχείριση καταστάσεων. Κάθε εξάρτηση διαδραματίζει κρίσιμο ρόλο στη διαμόρφωση της αρχιτεκτονικής και της λειτουργικότητας της εφαρμογής. Η παρούσα ανάλυση έχει ως στόχο να ρίξει φως στη σημασία αυτών των βιβλιοθηκών και στον τρόπο με τον οποίο συμβάλλουν στην ανάπτυξη μιας συνεκτικής και πλούσιας σε χαρακτηριστικά εφαρμογής React. Τα συγκεκριμένα πακέτα χρησιμοποιούνται από την εφαρμογή React αλλά και έπειτα από το migration , και από την εκάστοτε κινητή συσκευή που θα παραχθεί με το Ionic Framework.

3.5.1 @dnd-kit/core (Έκδοση 6.0.5)

Η *@dnd-kit/core* είναι μια θεμελιώδης βιβλιοθήκη για την ενσωμάτωση της λειτουργικότητας drag-and-drop στις εφαρμογές React. Προσφέρει ένα ισχυρό σύνολο εργαλείων και στοιχείων που διευκολύνουν τη δημιουργία διαδραστικών και διαισθητικών διεπαφών χρήστη. Με αυτή τη βιβλιοθήκη, οι προγραμματιστές μπορούν εύκολα να υλοποιήσουν αλληλεπιδράσεις drag-and-drop, καθιστώντας την κατάλληλη για εργασίες όπως η αναδιάταξη λιστών, η δημιουργία πινάκων Kanban ή ο σχεδιασμός προσαρμοσμένων στοιχείων UI με δυνατότητες drag-and-drop.

3.5.2 @dnd-kit/modifiers (Έκδοση 6.0.0)

Συμπληρώνοντας τη βασική βιβλιοθήκη, το *@dnd-kit/modifiers* επεκτείνει τη λειτουργικότητα των λειτουργιών drag-and-drop. Δίνει τη δυνατότητα στους προγραμματιστές να προσαρμόζουν και να τελειοποιούν τη συμπεριφορά των draggable στοιχείων, παρέχοντας μεγαλύτερο έλεγχο στην εμπειρία του χρήστη. Αυτή η ενότητα είναι ιδιαίτερα πολύτιμη κατά το χειρισμό σύνθετων περιπτώσεων χρήσης, όπως ο περιορισμός της κίνησης, η χρήση σε καταστάσεις grid layout ή η ενεργοποίηση ενεργειών βάσει συγκεκριμένων συνθηκών κατά τη διάρκεια μιας λειτουργίας drag-and-drop.

3.5.3 @dnd-kit/sortable (Έκδοση 7.0.1)

@dnd-kit/sortable βασίζεται στη βασική βιβλιοθήκη για να επιτρέπει τη δημιουργία λιστών με δυνατότητα ταξινόμησης. Η ταξινόμηση στοιχείων αποτελεί κοινή απαίτηση σε διάφορες εφαρμογές, όπως εργαλεία διαχείρισης εργασιών και πλατφόρμες ηλεκτρονικού εμπορίου. Αυτή η ενότητα απλοποιεί την υλοποίηση ταξινομήσιμων διεπαφών, επιτρέποντας στους χρήστες να αναδιατάσσουν τα στοιχεία χωρίς κόπο με μια διεπαφή drag-and-drop.

3.5.4 @microsoft/signalr (Έκδοση 6.0.8)

Το *@microsoft/signalr* είναι μια βιβλιοθήκη που διευκολύνει την επικοινωνία σε πραγματικό χρόνο μεταξύ του διακομιστή και του πελάτη σε μια εφαρμογή React. Χρησιμοποιεί WebSockets και άλλους μηχανισμούς μεταφοράς για να επιτρέπει στιγμιαίες ενημερώσεις, καθιστώντας το ιδανικό για λειτουργίες όπως εφαρμογές συνομιλίας, ζωντανές ειδοποιήσεις και συνεργατικά εργαλεία. Αυτή η βιβλιοθήκη διασφαλίζει ότι οι χρήστες λαμβάνουν έγκαιρες πληροφορίες και αλληλεπιδράσεις.

3.5.5 @popperjs/core (Έκδοση 2.11.8)

Το *@popperjs/core* είναι μια μηχανή τοποθέτησης (positioning engine) που δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν δυναμικά στοιχεία UI, όπως tooltips, popovers και dropdown menus. Προσφέρει ακριβή έλεγχο της τοποθέτησης και της ευθυγράμμισης των στοιχείων, βελτιώνοντας τη συνολική εμπειρία του χρήστη. Όταν συνδυάζεται με πλαίσια CSS όπως το Bootstrap, βοηθά στη δημιουργία ευέλικτων διεπαφών χρήστη.

3.5.6 @testing-library/jest-dom (Έκδοση 5.16.5), @testing-library/react (Έκδοση 13.4.0), @testing-library/user-event (Έκδοση 14.4.3)

Αυτές οι βιβλιοθήκες δοκιμών είναι απαραίτητες για τη διασφάλιση της αξιοπιστίας και της λειτουργικότητας των εφαρμογών React.Το *@testing-library/jest-dom* παρέχει προσαρμοσμένους matchers για το Jest, επιτρέποντας στους προγραμματιστές να γράφουν ευκόλως εννοούμενες και ακριβείς δοκιμές. Το *@testing-library/react* προσφέρει μια σουίτα βοηθητικών προγραμμάτων για την απόδοση και την αλληλεπίδραση με τα στοιχεία της React κατά τη διάρκεια των δοκιμών. Το *@testing-library/user-event* προσομοιώνει τις αλληλεπιδράσεις του χρήστη για τον έλεγχο των διεπαφών χρήστη, βοηθώντας στην επικύρωση της συμπεριφοράς των στοιχείων UI.

3.5.7 axios (Έκδοση 0.27.2)

Το *axios* είναι ένας ευρέως χρησιμοποιούμενος HTTP πελάτης (client) για την πραγματοποίηση αιτημάτων API σε εφαρμογές React. Απλοποιεί τη διαδικασία αποστολής και χειρισμού αιτημάτων HTTP, προσφέροντας χαρακτηριστικά όπως η αναχαίτιση αιτημάτων/απαντήσεων, η ακύρωση αιτημάτων και η αυτόματη σειριοποίηση αιτημάτων.

3.5.8 bootstrap (έκδοση 5.2.0), bootstrap-icons (έκδοση 1.9.1)

Το *bootstrap* είναι ένα δημοφιλές CSS framework που παρέχει responsive και βοηθητικά προγράμματα σχεδιασμού που ανταποκρίνονται στις ανάγκες του χρήστη. Το *bootstrap-

icons* επεκτείνει το Bootstrap προσφέροντας μια ολοκληρωμένη συλλογή εικονιδίων για χρήση στην εφαρμογή, ενισχύοντας την οπτική της ελκυστικότητα και την εμπειρία του χρήστη.

3.5.9 clsx (Έκδοση 1.2.1)

Το *clsx* είναι μια βοηθητική βιβλιοθήκη που απλοποιεί τη διαχείριση των υπό συνθήκη ονομάτων κλάσεων στο JSX. Είναι ιδιαίτερα χρήσιμη για τη δυναμική εφαρμογή κλάσεων CSS με βάση την κατάσταση του στοιχείου ή τις αλληλεπιδράσεις του χρήστη, βελτιώνοντας την ευελιξία του styling σε εφαρμογές React.

3.5.10 date-fns (Έκδοση 2.29.2)

Το *date-fns* είναι μια βοηθητική βιβλιοθήκη ημερομηνίας που απλοποιεί την εργασία με ημερομηνίες και ώρες σε JavaScript. Προσφέρει ένα ευρύ φάσμα συναρτήσεων για τη μορφοποίηση, την ανάλυση και τον χειρισμό ημερομηνιών, εξασφαλίζοντας ακριβή και συνεπή χειρισμό ημερομηνιών στην εφαρμογή σας.

3.5.11 query-string (Έκδοση 7.1.1)

Η *query-string* είναι μια χρήσιμη βιβλιοθήκη για την ανάλυση και μορφοποίηση των URL query strings. Είναι ιδιαίτερα πολύτιμη όταν ασχολείται με τις παραμέτρους δρομολόγησης και URL σε εφαρμογές React, επιτρέποντας στους προγραμματιστές να εξάγουν και να χειρίζονται εύκολα τις παραμέτρους ερωτήματος.

3.5.12 react (Έκδοση 18.2.0), react-dom (Έκδοση 18.2.0)

Οι *react* και *react-dom* είναι οι βασικές βιβλιοθήκες για τη δημιουργία εφαρμογών React. Το *react* διαχειρίζεται τη δημιουργία και την απόδοση των στοιχείων UI, ενώ το *react-dom* διαχειρίζεται την αλληλεπίδραση μεταξύ της React και του Document Object Model (DOM) του προγράμματος περιήγησης ιστού. Αυτές οι βιβλιοθήκες αποτελούν το θεμέλιο των εφαρμογών που βασίζονται στο React.

3.5.13 react-error-boundary (Έκδοση 3.1.4)

Το *react-error-boundary* παρέχει όρια σφαλμάτων για τα στοιχεία της React. Αυτή η βιβλιοθήκη βοηθά τους προγραμματιστές να χειρίζονται τα σφάλματα που μπορεί να προκύψουν μέσα σε ένα component, αποτρέποντάς τα από το να επηρεάσουν ολόκληρη την εφαρμογή και προσφέροντας μια ομαλή εμπειρία χρήσης.

3.5.14 react-hot-toast (Έκδοση 2.3.0)

Το *react-hot-toast* απλοποιεί την εμφάνιση προσαρμοσμένων ειδοποιήσεων τύπου toast σε εφαρμογές React. Παρέχει έναν φιλικό προς το χρήστη τρόπο για τη μετάδοση σημαντικών πληροφοριών, όπως μηνύματα επιτυχίας ή ειδοποιήσεις σφαλμάτων, βελτιώνοντας τη συνολική εμπειρία του χρήστη.

3.5.15 react-query (Έκδοση 3.39.2)

Το *react-query* είναι μια ισχυρή βιβλιοθήκη για τη διαχείριση ασύγχρονων δεδομένων και αλληλεπιδράσεων με τον διακομιστή σε εφαρμογές React. Εισάγει hooks και βοηθητικά

προγράμματα που βελτιστοποιούν τη λήψη δεδομένων, την προσωρινή αποθήκευση και τη διαχείριση του state, διευκολύνοντας τον χειρισμό σύνθετων σεναρίων που βασίζονται σε δεδομένα.

3.5.16 react-router-dom (Έκδοση 6.3.0)

Η *react-router-dom* είναι μια ευρέως υιοθετημένη βιβλιοθήκη δρομολόγησης (routing) για εφαρμογές React. Διευκολύνει την πλοήγηση και το χειρισμό URL, επιτρέποντας στους προγραμματιστές να δημιουργούν εφαρμογές μιας σελίδας με πολλαπλές προβολές και δυναμική δρομολόγηση.

3.5.17 reactour (Έκδοση 1.18.7)

Το *reactour* είναι μια βιβλιοθήκη για τη δημιουργία ξεναγήσεων και περιηγήσεων μέσα σε εφαρμογές React. Ενισχύει την εισαγωγή των χρηστών και την ανακάλυψη των χαρακτηριστικών, παρέχοντας έναν δομημένο και διαδραστικό τρόπο για την εισαγωγή των χρηστών σε βασικά χαρακτηριστικά της εφαρμογής.

3.5.18 recoil (Έκδοση 0.7.5)

Το *recoil* είναι μια βιβλιοθήκη διαχείρισης καταστάσεων (state management) ειδικά σχεδιασμένη για εφαρμογές React. Απλοποιεί τη διαχείριση του state της εφαρμογής, διευκολύνοντας τον χειρισμό σύνθετων απαιτήσεων, την κοινή χρήση state μεταξύ των components και τη βελτιστοποίηση της απόδοσής τους.

3.5.19 sass (Έκδοση 1.54.7)

Το *sass* είναι ένας δημοφιλής προεπεξεργαστής (preprocessor) CSS που δίνει τη δυνατότητα στους προγραμματιστές να γράφουν καθαρότερα και πιο συντηρήσιμα stylesheets. Εισάγει χαρακτηριστικά όπως οι μεταβλητές, η ένθεση και τα mixins, διευκολύνοντας τη διαχείριση και την οργάνωση των στυλ για τα components.

3.5.20 styled-components (Έκδοση 6.1.0)

Το *styled-components* είναι μια βιβλιοθήκη CSS-in-JS που επιτρέπει στους προγραμματιστές να γράφουν CSS απευθείας μέσα στα React components. Αυτή η προσέγγιση ενισχύει την ενθυλάκωση των component, την επαναχρησιμοποίηση και τη συντηρησιμότητα, ενώ επιτρέπει τη δυναμική μορφοποίηση με βάση τα props και το state των components.

3.5.21 typescript (έκδοση 4.8.2)

Η *typescript* είναι ένα στατικά τυποποιημένο υπερσύνολο (statically typed superset) της JavaScript, που χρησιμοποιείται συχνά σε React projects για τη βελτίωση της ποιότητας του κώδικα και την παροχή καλύτερων εργαλείων και ελέγχου σφαλμάτων. Βελτιώνει την εμπειρία ανάπτυξης προσφέροντας ισχυρό έλεγχο τύπων και αυτόματη συμπλήρωση.

3.5.22 uuid (Έκδοση 8.3.2)

Το *uuid* είναι μια βιβλιοθήκη που απλοποιεί την παραγωγή και τον χειρισμό των καθολικά μοναδικών αναγνωριστικών (UUIDs). Αυτά τα αναγνωριστικά είναι χρήσιμα για διάφορους σκοπούς σε εφαρμογές React, όπως η διάκριση μεταξύ καταχωρήσεων δεδομένων και η διασφάλιση της ακεραιότητας των δεδομένων.

3.5.23 vite-plugin-svgr (Έκδοση 3.2.0)

Το *vite-plugin-svgr* είναι ένα πρόσθετο του Vite που διευκολύνει την εισαγωγή αρχείων SVG ως component της React. Αυτό απλοποιεί την ενσωμάτωση εικονιδίων και γραφικών SVG σε εφαρμογές React, καθιστώντας τις εύχρηστες και προσαρμόσιμες.

3.5.24 yup (Έκδοση 0.32.11)

Το *yup* είναι μια βιβλιοθήκη επικύρωσης σχημάτων που χρησιμοποιείται συνήθως για την επικύρωση φορμών σε εφαρμογές React. Εξασφαλίζει ότι η είσοδος του χρήστη στην εκάστοτε φόρμα πληροί προκαθορισμένα κριτήρια, βελτιώνοντας την ποιότητα και την αξιοπιστία των δεδομένων στην εφαρμογή.

Συμπερασματικά, οι εξαρτήσεις που παρατίθενται σε αυτή την εφαρμογή React είναι κρίσιμα δομικά στοιχεία που εξυπηρετούν διάφορους σκοπούς, από την ενίσχυση των διεπαφών χρήστη και τη διαχείριση της κατάστασης μέχρι τη βελτίωση των δοκιμών και του χειρισμού σφαλμάτων. Η κατανόηση και η αποτελεσματική χρήση αυτών των βιβλιοθηκών είναι απαραίτητη για τη δημιουργία πλούσιων σε χαρακτηριστικά και συντηρήσιμων εφαρμογών React που παρέχουν εξαιρετική εμπειρία χρήσης.

3.6 Ανάλυση της αρχιτεκτονικής που ακολουθείται στο API

Η εφαρμογή χωρίζεται σε τρία επίπεδα, εκείνο της βάσης δεδομένων (database) που συντηρεί τα δεδομένα, του server που παρουσιάζει και επεξεργάζεται τα δεδομένα, και του client που προβάλλει τα δεδομένα. Με τη σειρά του ο server χωρίζεται στα υποεπίπεδα του Domain που περιέχει τα μοντέλα της εφαρμογής, του Persistence που σχηματοποιεί τη βάση δεδομένων σύμφωνα με τα μοντέλα, του Application στο οποίο υλοποιούνται οι επιχειρησιακές λειτουργίες της εφαρμογής με τη χρήση των μοντέλων και της βάσης δεδομένων. Η αλληλεπίδραση του επιπέδου server με διάφορους πελάτες (clients) γίνεται μέσω του υποεπιπέδου WebApi στο οποίο τοποθετούνται τα σημεία επικοινωνίας (endpoints). Συνεπώς, για τη χρήση της οποιαδήποτε επιχειρησιακής λειτουργίας γίνεται κλήση και του αντίστοιχου endpoint. Συνεπώς η αρχική οργάνωση της εφαρμογής έχει ως εξής:

database ⇔ server ⇔ client

3.6.1 Επίπεδο βάσης δεδομένων (database)

Η βάση δεδομένων που χρησιμοποιεί η εφαρμογή είναι η MongoDb, όπου είναι η πιο δημοφιλής βάση δεδομένων NoSQL, είναι μια βάση δεδομένων ανοιχτού κώδικα προσανατολισμένη στα έγγραφα. Ο όρος «NoSQL» σημαίνει «μη σχεσιακός». Σημαίνει ότι η MongoDB δεν βασίζεται στη δομή της σχεσιακής βάσης δεδομένων που μοιάζει με πίνακα,

αλλά παρέχει έναν εντελώς διαφορετικό μηχανισμό για την αποθήκευση και την ανάκτηση δεδομένων. Αυτή η μορφή αποθήκευσης ονομάζεται BSON (παρόμοια με τη μορφή JSON). Συνεπώς τα αρχεία BSON που ονομάζονται εγγραφές ή και έγγραφα, αποθηκεύονται σε συλλογές (collections). Οι συλλογές που υπάρχουν στη βάση δεδομένων είναι οι εξής:

- Announcements (Ανακοινώσεις)
- Contacts (Επαφές)
- Coordinations (Διεκπαιρεώσεις)
- Documents (Έγγραφα)
- FAQs (Ερωτήσεις)
- fs.files (Αρχεία)
- Goals (Στόχοι)
- Leaves (Άδειες)
- Library (Βιβλιοθήκη)
- Registry (Πρωτόκολλο)
- SignificantIssues (Σημαντικά θέματα)
- Substitutions (Αντικαταστάσεις)
- Tasks (Εργασίες)
- UserNotes (Σημειώσεις χρήστη)

3.6.2 Επίπεδο Server

Το επίπεδο του server είναι οργανωμένο σύμφωνα με το μοτίβο αρχιτεκτονικής του clean architecture καθώς τα επιμέρους υποεπίπεδα είναι διακριτά μεταξύ τους και συσχετισμένα. Δηλαδή, το υποεπίδο Domain συσχετίζεται με εκείνο του Persistence, το τελευταίο με το Application και το WebApi με το Application. Η οργάνωση αυτή του κώδικα έχει ως στόχο την κατά το μέγιστο αποτελεσματική συντήρηση και περαιτέρω ανάπτυξή του. Συνεπώς έχουμε:

➢ Domain → Persistence

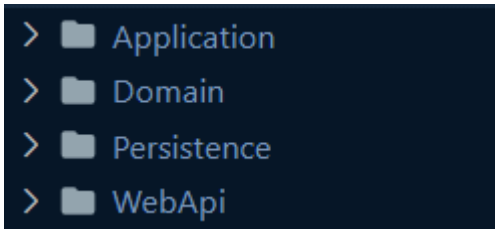
```
<ProjectReference Include="..\Domain\Domain.csproj" />
```

➢ Persistence → Application

```
<ProjectReference Include="..\Persistence\Persistence.csproj" />
```

➢ Application → API

```
<ProjectReference Include="..\Application\Application.csproj" />
```



3.6.3 Υποεπίπεδο Domain (layer)

Συγκεκριμένα το υποεπίπεδο Domain περιέχει το φάκελο Models στον οποίο τοποθετούνται όλες εκείνες οι οντότητες και οι συσχετίσεις τους που συγκροτούν το μοντέλο της εφαρμογής. Ακόμα, στο ίδιο υποεπίπεδο περιέχεται ο φάκελος Dtos (Data Transfer Objects) στον οποίο τοποθετούνται όλες εκείνες οι οντότητες με τις οποίες αλληλεπιδρούν οι διάφοροι clients της εφαρμογής. Δηλαδή, καθορίζεται το σχήμα του εκάστοτε JSON που λαμβάνει ή στέλνει ο κάθε client. Και στις δύο περιπτώσεις υποφακέλων οι επιμέρους κλάσεις άρα και οντότητες τοποθετούνται σε κατηγορίες (modules). Για παράδειγμα το μοντέλο που θα αντιπροσωπεύει την εργασία ενός υπαλλήλου θα είχε την εξής μορφή ως κλάση:

```
public class EmployeeTask
    {
        public RecipientProfile From { get; set; }
        public RecipientProfile To { get; set; }
        public List<Reference> Documents { get; set; }
        public string Text { get; set; }
        public TaskStatus Status { get; set; }
        public TaskType Type { get; set; }
        public Priority Priority { get; set; }
        public int Percentage { get; set; }
        public Nullable<DateTime> AckDate { get; set; }
        public Nullable<DateTime> LastUpdate { get; set; }
        public string LastUpdatedBy { get; set; }
        public Lookup ParentTask { get; set; }
        public IList<Lookup> Attachments { get; set; }
        public IList<string> OtherRecipients { get; set; }
        public string TaskId { get; set; }
        public bool IsPersonal { get; set; }
    }
```

Ομοίως και η μορφή ενός Dto που αντιπροσωπεύει τη λίστα εργασιών για ενημέρωση ενός υπαλλήλου θα είχε την εξής μορφή:

```
public record InboxInfoDto(
    string Id,
    int Priority,
    string Classification,
    string Type,
    string Folder,
    string RegNo,
    string OutboundNo,
    string Subject,
    string Publisher,
    string Author,
    string Date,
    string Labels,
    bool IsRead,
```

```
    TasksStatus TasksStatus,
    bool IsMaster
);
```

3.6.4 Υποεπίπεδο Persistence (layer)

Στο υποεπίπεδο Persistence καθορίζεται το σχήμα που θα λάβει η βάση δεδομένων σύμφωνα με τις συσχετίσεις του μοντέλου των οντοτήτων που καθορίστηκαν στο Domain. Συνεπώς, απαιτείται η δημιουργία αναφοράς από το υποεπίπεδο Domain σε εκείνο του Persistence, έτσι ώστε να δημιουργηθούν και οι απαραίτητες συσχετίσεις των οντοτήτων σε επίπεδο βάσης. Δηλαδή, σε αυτό το υποεπίπεδο ορίζεται η μορφή της βάσης καθώς και οι επιμέρους ρυθμίσεις όσον αφορά τον τύπο δεδομένων του κάθε πεδίου της βάσης δεδομένων. Όπως αναφέρθηκε η βάση που χρησιμοποιείται είναι μη σχεσιακή (NoSQL Database) και στο συγκεκριμένο υποεπίπεδο καθορίζονται οι συλλογές (collections) σε σχέση με το κάθε μοντέλο που αναφέρθηκε στο Domain. Επίσης καθορίζεται και η μορφή με την οποία αποθηκεύονται στη βάση καθώς στο φάκελο ClassMaps δηλώνεται η κάθε οντότητα σε συνάρτηση με το αντίστοιχο collection. Συνεπώς για τον ορισμό των collections γίνεται η χρήση της κλάσης DataContext ως εξής:

```
public class DataContext
{
    IMongoCollection<Document> Documents { get; }
    IMongoCollection<EmployeeTask> Tasks { get; }
    IMongoCollection<Leave> Leaves { get; }
    IMongoCollection<SignificantIssue> SignificantIssues { get; }
    IMongoCollection<Substitution> Substitutions { get; }
    IMongoCollection<Coordination> Coordinations { get; }
    IMongoCollection<Goal> Goals { get; }
}
```

Αντίστοιχα, μια κλάση ClassMap για την οντότητα του εγγράφου (Document) θα είχε την εξής μορφή:

```
public class DocumentClassMap : BsonClassMap<Document>
{
    public DocumentClassMap()
    {
        this.AutoMap();
        this.UnmapProperty(c => c.View);
        this.UnmapProperty(c => c.Title);
    }
}
```

3.6.5 Υποεπίπεδο Application (layer)

Η επιχειρησιακή λογική της εφαρμογής υλοποιείται στο υποεπίπεδο του Application καθώς σε αυτό εντάσσονται όλες εκείνες οι λειτουργίες που θα εκτελέσει η εφαρμογή. Συγκεκριμένα για κάθε κατηγορία (module) που περιγράφεται ως οντότητα στο υποεπίπεδο Domain, περιέχονται όλες εκείνες οι λειτουργίες που προκαλούν κάποια μεταβολή σε αυτή. Δηλαδή για την κατηγορία των εγγράφων (documents) υπάρχουν όλες εκείνες οι λειτουργίες που δημιουργούν, ενημερώνουν, κάνουν ανάγνωση και διαγράφουν κάποιο έγγραφο (CRUD operations). Συνεπώς, μέσα στο υποεπίπεδο υπάρχουν οι εξής κατηγορίες (modules):

- Ανακοινώσεις (Announcements)

- Επαφές (Contacts)
- Διεκπαιρεώσεις (Coordinations)
- Έγγραφα (Documents)
- Εξωτερική σύνδεση (External)
- Ερωτήσεις (FAQs)
- Αρχεία (Files)
- Στόχοι (Goals)
- Έγγραφα από εισαγωγή (ImportedDocuments)
- Άδειες (Leaves)
- Βιβλιοθήκη (Library)
- Ενημερωτικά έγγραφα (MasterDocuments)
- Θέσεις υπαλλήλων (Positions)
- Πρωτόκολλο (Registry)
- Σημαντικά θέματα (Significant Issues)
- Στατιστικά (Stats)
- Αντικαταστάσεις (Substitutions)
- Εργασίες (Tasks)
- Σημειώσεις χρήστη (UserNotes)

Στο συγκεκριμένο υποεπίπεδο η οργάνωση του κώδικα γίνεται με την αρχιτεκτονική του clean architecture και χρησιμοποιείται το σχέδιο ανάπτυξης (design patterns) CQRS pattern. Το CQRS (Command Query Responsibility Segregation) είναι ένα πρότυπο σχεδίασης (design pattern) που χρησιμοποιείται στον τομέα της αρχιτεκτονικής λογισμικού για να χωρίσει τις εντολές (commands) και τα ερωτήματα (queries) που χειρίζεται ένα σύστημα. Βασικά χαρακτηριστικά του CQRS είναι τα ακόλουθα:

- Εντολές (Commands): Οι εντολές αντιπροσωπεύουν τις αλλαγές που πρέπει να εφαρμοστούν σε ένα σύστημα. Αυτές οι εντολές μπορεί να περιλαμβάνουν εισαγωγή, ενημέρωση ή διαγραφή δεδομένων. Οι εντολές αυτές εκτελούνται μόνο στο επίπεδο εγγραφής (write model) του συστήματος.
- Ερωτήματα (Queries): Τα ερωτήματα αντιπροσωπεύουν τις λειτουργίες ανάγνωσης των δεδομένων. Αυτά τα ερωτήματα διαχειρίζονται μόνο από το επίπεδο ανάγνωσης (read model) του συστήματος.

Ο σκοπός του CQRS είναι να επιτρέψει την ανεξάρτητη εξέλιξη και τη βελτιστοποίηση των δύο διαφορετικών μοντέλων (εγγραφής και ανάγνωσης). Αυτή η διαχωριστική προσέγγιση επιτρέπει στους προγραμματιστές να βελτιστοποιούν κάθε μοντέλο για τις ανάγκες του κατάλληλου τύπου λειτουργίας, χωρίς να επηρεάζει το άλλο. Η χρήση του CQRS μπορεί να είναι χρήσιμη όταν ένα σύστημα είναι υποχρεωμένο να υποστηρίξει διάφορες προβολές και ερωτήματα, ενώ παράλληλα να επιτρέπει πολύπλοκες εντολές γραφής. Συνεπώς στην περίπτωση του υποεπιπέδου Application για κάθε κατηγορία που αναφέρθηκε έχουμε έχουμε τους υποφακέλους Commands και Queries όπου εκτελούνται και οι αντίστοιχες λειτουργίες.

Ακόμα στο υποεπίπεδο Application γίνεται του σχεδίου ανάπτυξης Dependency Injection pattern. Όπου το Dependency Injection (DI) είναι ένα πρότυπο σχεδίασης που χρησιμοποιείται στον τομέα της λογισμικής αρχιτεκτονικής για τη διαχείριση των εξαρτήσεων μεταξύ των κλάσεων. Στην ουσία, το Dependency Injection βοηθά στη μείωση του συσχετισμού μεταξύ κλάσεων, καθιστώντας τον κώδικα πιο ευανάγνωστο, ευέλικτο και ευκολονόητο. Οι βασικές έννοιες του Dependency Injection είναι:

- Dependency (Εξάρτηση): Μια κλάση έχει μια εξάρτηση όταν χρησιμοποιεί ένα αντικείμενο άλλης κλάσης για να εκτελέσει μια λειτουργία. Οι εξαρτήσεις μπορεί να είναι αντικείμενα άλλων κλάσεων, υπηρεσίες, ή οτιδήποτε άλλο.
- Injection (Εισαγωγή): Η διαδικασία της εισαγωγής σημαίνει ότι οι εξαρτήσεις δεν δημιουργούνται εντός της κλάσης που τις χρησιμοποιεί, αλλά παρέχονται σε αυτήν από έξω. Αυτό γίνεται για να επιτραπεί η ευκολότερη ανταλλαγή, δοκιμή, και επαναχρησιμοποίηση των εξαρτήσεων.

Η εισαγωγή εξαρτήσεων μπορεί να γίνει με διάφορους τρόπους, όπως:

- Constructor Injection (Εισαγωγή μέσω κατασκευαστή): Οι εξαρτήσεις περνούν μέσω του κατασκευαστή της κλάσης.
- Setter Injection (Εισαγωγή μέσω μεθόδου setter): Οι εξαρτήσεις περνούν μέσω μεθόδου setter της κλάσης.
- Method Injection (Εισαγωγή μέσω μεθόδου): Οι εξαρτήσεις περνούν ως παράμετροι σε μεθόδους.

Επίσης, το Dependency Injection βελτιώνει την αναγνωσιμότητα του κώδικα, διευκολύνει τη δοκιμή και επιτρέπει την επαναχρησιμοποίηση των κλάσεων. Επιπλέον, βοηθά στη διαχείριση των εξαρτήσεων και στη μείωση του συσχετισμού μεταξύ των διάφορων συστατικών του συστήματος. Αρα και στην περίπτωση του Application layer περιέχεται ο φάκελος Common στον οποίο υπάρχουν οι υποφάκελοι Interfaces και Services. Στον υποφάκελο Interfaces περιέχονται όλες εκείνες οι διεπαφές στις οποίες αναφέρονται οι μέθοδοι που θα υλοποιηθούν στον υποφάκελο Services και θα χρησιμοποιηθούν από κάποιο Command ή Query. Ένα ερώτημα που θα άνηκε στην κατηγορία ενός Query για την αναζήτηση της λίστας όλων των εισερχόμενων εργασιών ενός υπαλλήλου θα είχε την εξής μορφή:

```
#region Query
public record InboxAllQuery(UrlQuery UrlQuery) :
IRequest<Response<List<InboxAllDto>>>;
#endregion

#region Handler
public class InboxAllHandler : IRequestHandler<InboxAllQuery,
Response<List<InboxAllDto>>>
{
    private readonly IDataContext _context;
    private readonly ICurrentUserService _currentUserService;

    public InboxAllHandler(IDataContext context,
         ICurrentUserService currentUserService)
    {
        _context = context;
        _currentUserService = currentUserService;
    }
```

```
    public async Task<Response<List<InboxAllDto>>> Handle(InboxAllQuery
request, CancellationToken token)
    {
        // Active Profile Management
        var user = _currentUserService.UserAccessor();

        // Default Sorting
        if (!request.UrlQuery.HasSortBy)
        {
            request.UrlQuery.SortBy = TasksRepository.StartDateSorting;
            request.UrlQuery.SortDescending = true;
        }

        // Filtering Inbox All Tasks
        var filter = TasksFilters.InboxAllFilter(user);

        // Filtering Table Columns Tasks
     if (request.UrlQuery.HasFilter)
  filter&=TasksFilters.AssignedTasksFilter(request.UrlQuery.Filter);

     // Paginating Tasks and Mapping to InboxAllDto
   var pagedTasks = await _context.GetEnvelopeAsync(_context.Tasks,
filter, request.UrlQuery);
     var dto = pagedTasks.Rows.Select(t => t.ToInboxAllDto()).ToList();

        // Returning Initialized Response
        return new Response<List<InboxAllDto>>()
        {
            Data = dto,
            Pagination = new Pagination
            {
                TotalRecords = pagedTasks.UrlQuery.TotalRecords,
                PageSize = pagedTasks.UrlQuery.PageSize,
                PageNumber = pagedTasks.UrlQuery.PageNumber ?? 1,
            }
        };
    }
}
#endregion
```

Ένα αντίστοιχο Command για τη δημιουργία ενός σημαντικού θέματος θα είχε την εξής μορφή:

```
#region Command
public record CreateSignificantIssueCommand(CreateSignificantIssueDto
IssueDto) : IRequest<CommandResponse<string>>;
#endregion

#region Handler
public class CreateSignificantIssueHandler :
IRequestHandler<CreateSignificantIssueCommand, CommandResponse<string>>
{
    private readonly IDataContext _context;
    private readonly Site _site;
    private readonly ICurrentUserService _currentUserService;
    private readonly IBadgesManager _badgesManager;
    private readonly IIssuesHelper _issuesHelper;
    private readonly IProfileHelper _profileHelper;
```

```
    public CreateSignificantIssueHandler(
        IDataContext context,
        Site site,
        ICurrentUserService currentUserService,
        IBadgesManager badgesManager,
        IIssuesHelper issuesHelper,
        IProfileHelper profileHelper)
    {
        _context = context;
        _site = site;
        _currentUserService = currentUserService;
        _badgesManager = badgesManager;
        _issuesHelper = issuesHelper;
        _profileHelper = profileHelper;
    }

     public async Task<CommandResponse<string>> Handle
     (CreateSignificantIssueCommand request, CancellationToken token)
    {
        // Active Profile Management
        var user = _currentUserService.UserAccessor();

        var dto = request.IssueDto;

        var messageRecipients = new List<Tuple<int, int>>
        { new Tuple<int, int>(user.PositionId, user.DutyId) };

        var from = await _profileHelper.CreateParticipantProfileAsync(
         user.PositionId, user.DutyId,
         user.Title, user.Employee);

        var maxSN = await _issuesHelper.GetSignificantIssuesNextSN(
         user.PositionId, dto.StartDate.Year, token);
        var groupInfo = await _issuesHelper.GenerateGroupInfoAsync(
         user.PositionId, 6);

        var newIssue = dto.ToCreateSignificantIssue(
            from,
            maxSN,
            groupInfo,
            _site.SiteId
        );

        foreach (var recipient in dto.Recipients)
        {
            var participant = await _profileHelper
            .CreateParticipantProfileAsync(
                     recipient.PositionId, recipient.DutyId,
                     null, new Tuple<int, int>(-1, -1));

            newIssue.To.Add(participant);
            messageRecipients.Add(
                new Tuple<int, int>(
                recipient.PositionId, recipient.DutyId));
        }

        await _context.SignificantIssues.InsertOneAsync(newIssue);
```

```
        foreach (var recipient in messageRecipients)
        {
            var message = newIssue.CreateSignificantIssueBadgeMessage(
               recipient);
            await _badgesManager.SendMessage(message);
        }

        return new CommandResponse<string>()
                .WithData($"Το σημαντικό θέμα με id {newIssue.Id}
δημιουργήθηκε επιτυχώς");
    }
}
#endregion
```

Επιπλέον στο συγκεκριμένο υποεπίπεδο υπάρχει ο φάκελος Filters όπου αναφέρονται τα κυριότερα φίλτρα αναζήτησης. Ένα φίλτρο αναζήτησης χρησιμοποιείται κατά κύριο λόγο σε στα ερωτήματα αναζήτησης, εντολές ενημέρωσης και διαγραφής. Τα φίλτρα κατηγοριοποιούνται ανά τις κύριες κατηγορίες (modules) με σκοπό την επαναχρησιμοποίηση του κώδικα, στις εξής :

- DocumentsFilters (φίλτρα που αντιστοιχούν στα έγγραφα)
- TasksFilters (φίλτρα που αντιστοιχούν στις εργασίες)
- GoalsFilters (φίλτρα που αντιστοιχούν στους στόχους)
- SignificantIssuesFilters (φίλτρα που αντιστοιχούν στα σημαντικά θέματα)
- LeavesFilters (φίλτρα που αντιστοιχούν στις άδειες)
- ContactsFilters (φίλτρα που αντιστοιχούν στις επαφές)
- CoordinationsFilters (φίλτρα που αντιστοιχούν στις διεκπαιρεώσεις)
- LibraryFilters (φίλτρα που αντιστοιχούν στη βιβλιοθήκη)
- RegistryFilters (φίλτρα που αντιστοιχούν στο πρωτόκολλο)

Όπως αναφέρθηκε στο ερώτημα που αντιστοιχεί στη λίστα όλων των εισερχόμενων εργασιών ενός υπαλλήλου InboxAllQuery, γίνεται χρήση των φίλτρων InboxAllFilter και AssignedTasksFilters. Συνεπώς για παράδειγμα ενός φίλτρου για το InboxAllFilter θα είχαμε το εξής:

```
public static FilterDefinition<EmployeeTask> InboxAllFilter(UserProfile
user)
        {
            var filter = Builders<EmployeeTask>.Filter.Eq(x =>
               x.To.Department.Id, user.PositionId) &
              Builders<EmployeeTask>.Filter.Eq(x =>
               x.To.Title.Id, user.DutyId) &
                (
                    Builders<EmployeeTask>.Filter.Eq(x =>
                      x.IsPersonal, false) |
                    (
                           Builders<EmployeeTask>.Filter.Eq(
                            "To.Name._id.0",
                            user.EmployeeUser.EmployeeId) &
                           Builders<EmployeeTask>.Filter.Eq(
                            "To.Name._id.1",
```

```
                            user.EmployeeUser.EmployeeCategory)
                    )
                );

            return filter;
        }
```

Ενώ για το AssignedTasksFilters θα είχαμε το εξής:

```
public static FilterDefinition<EmployeeTask> AssignedTasksFilter(string
input)
        {
            var filter = Regex.Escape(input);
            var query = Builders<EmployeeTask>.Filter.Regex(x =>
             x.From.Department.Description,
                   new BsonRegularExpression(filter, "i")) |
                Builders<EmployeeTask>.Filter.Regex(x =>
                  x.From.Name.Description,
                  new BsonRegularExpression(filter, "i")) |
                Builders<EmployeeTask>.Filter.Regex(x =>
                 x.Subject, new BsonRegularExpression(filter, "i")) |
                Builders<EmployeeTask>.Filter.Regex(x =>
                 x.Text, new BsonRegularExpression(filter, "i")) |
                Builders<EmployeeTask>.Filter.ElemMatch(x =>
                 x.Documents, doc => doc.Description.Contains(filter));

            var isDate = DateTime.TryParse(filter, out DateTime dateF);

            if (isDate)
            {
                query = query |
                    (Builders<EmployeeTask>.Filter.Gte(x =>
               x.StartDate, dateF) &
         Builders<EmployeeTask>.Filter.Lt(x => x.StartDate,
dateF.AddDays(1))) |

            (Builders<EmployeeTask>.Filter.Gte(x =>
             x.DueDate, dateF) &
            Builders<EmployeeTask>.Filter.Lt(x =>
            x.DueDate, dateF.AddDays(1)));
             }
             return query;
        }
```

Το υποεπίπεδο Application έχει ακόμα το φάκελο Mapping, στο οποίο γίνεται χρήση των μοντέλων και των DTOs που αναφέρθηκαν στο υποεπίπεδο Domain. Συνεπώς, σε ένα ερώτημα αναζήτησης στον client θα επιστραφεί ένα JSON αρχείο το οποίο θα έχει τη μορφή κάποιου DTO. Η πληροφορία όμως ανακτάται από τη βάση δεδομένων σε μορφή κάποιου μοντέλου, όποτε με σκοπό να παρουσιαστεί στον client σε κάποια μορφή DTO , απαιτείται η μορφοποίηση και αντιστοίχιση σε αυτό. Το ίδιο συμβαίνει και όταν ο client αποστέλλει κάποιο JSON σε μορφή κάποιου DTO, απαιτείται η αντιστοίχιση αυτού σε κάποιο μοντέλο. Συνεπώς για κάθε κατηγορία μοντέλου υπάρχει και η αντίστοιχη που κάνει μορφοποίηση και αντιστοίχιση. Για παράδειγμα για την αντιστοίχιση του μοντέλου σε κάποιο DTO στο ερώτημα για την ανάκτηση της λίστας όλων των εισερχόμενων εργασιών ενός υπαλλήλου (InboxAllQuery) θα είχαμε το εξής:

```
var dto = pagedTasks.Rows.Select(t => t.ToInboxAllDto()).ToList();
```

```
public static class InboxAllDtoMapping
{
    public static InboxAllDto ToInboxAllDto(this EmployeeTask model) =>
        new InboxAllDto(
            Id: model.Id,
            Priority: model.Priority.GetDescription(),
            Status: model.Status.GetDescription(),
            Subject: model.Subject,
            From: model.From.Abbreviation,
            StartDate: DateTimeExtensions.GetLocalDateString(
        model.StartDate),
            DueDate: DateTimeExtensions.GetLocalDateString(
              model.DueDate),
            Percentage: model.Percentage,
            IsAcknowledged: model.IsAcknowledged,
            Description: model.Text,
            Actions: model.Remarks,
            ChildrenTasksCount: model.ChildrenTasksCount,
            Documents: model.Documents
                .Where(d => d.ReferenceType == ReferenceType.Document)
                .Select(d =>
                  new DocumentRefsDto(
                     d.Id,
                     d.Description,
 Enumerations.GetItemByValue(EnumerationType.DocumentType,
d.DocumentType).Category == EnumerationCategory.InfoDocument))
                .ToList()
        );
}
```

Ενώ για παράδειγμα για την αντιστοίχιση ενός DTO σε μοντέλο κατά την αποθήκευση ενός σημαντικού θέματος θα είχαμε το εξής:

```
public static class CreateSignificantIssueDtoMapping
{
    public static SignificantIssue ToCreateSignificantIssue(
        this CreateSignificantIssueDto dto,
        ParticipantProfile from,
        int sn,
        GroupInfo groupInfo,
        string siteId
    ) => new SignificantIssue()
    {
        From = from,
        SN = sn,
        GroupInfo = groupInfo,
        LastUpdatedBy = from,
        LastUpdatedDate = DateTime.Now,
        DueDateHistory = new List<DateHistory>()
        {
            new DateHistory
            {
                Title = $"{from.Abbreviation} - {from.InfoTitle}",
                Remarks = "Εναρξη Θέματος",
                UpdateDate = DateTime.Now,
                PreviousDate = dto.DueDate,
                NewDate = dto.DueDate
            }
```

```
        },
        Subject = dto.Subject,
        Text = dto.Text,
        Remarks = dto.Remarks,
        Priority = dto.Priority,
        Status = dto.Status,
        Documents = dto.References.Select(d =>
       d.ToReferenceModel()).ToList(),
        Tasks = new List<LookupExt<int>>(),
        Involved = dto.Involved,
        IsAnnotated = dto.IsAnnotated,
        IsExternalInfo = dto.IsExternalInfo,
        SiteId = siteId
    };
}
```

3.6.6 Υποεπίπεδο WebApi (layer)

Το συγκεκριμένο υποεπίπεδο λειτουργεί ως ένα API (Application Programming Interface) σύμφωνα με το αρχιτεκτονικό μοντέλο των RESTful (Representational State Transfer), όπου καθορίζονται τα σημεία επικοινωνίας του επιπέδου server με έναν ή και περισσότερους client. Τα σημεία επικοινωνίας του WebApi καλούνται και ως endpoints και στη συγκεκριμένη περίπτωση αναφέρονται ως Routes σε κάποιον Controller, όπου εκτελούν κάποια ενέργεια GET, POST, PUT ή DELETE σύμφωνα με το HTTPS πρωτόκολλο. Οι ενέργειες των GET εκτελούνται σε κάποιον Controller που εκτελεί ερωτήματα-Queries ενώ οι ενέργειες των POST, PUT και DELETE αντιστοιχούν σε κάποιον Controller που εκτελεί εντολές-Commands. Για παράδειγμα όσον αφορά τα σημεία επικοινωνίας που αφορούν τις λίστες όλων των εξερχομένων και εισερχομένων εργασιών ενός υπαλλήλου θα είχαμε τον εξής Controller με τα ανάλογα endpoints:

```
[Route("api/v2/tasks")]
[ApiController]
public class TasksQueriesController : BaseController
{
      [Route("inbox/all")]
      [ApiExplorerSettings(GroupName = "v2")]
      [HttpGet]
    [SwaggerOperation(Summary = "ROUTE-CLIENT: tasks/inbox, MENU:
Εισερχόμενα/Όλες", Description = "Επιστροφή Λίστας με Όλες τις
Εργασίες(Task)")]
    [SwaggerResponse(200, "Επιστρέφει αντικείμενο τύπου Response με
δεδομένα λίστα από αντικείμενα InboxAllDto.",
typeof(Response<List<InboxAllDto>>))]
    public async Task<IActionResult> GetAllInboxTasks([FromQuery] UrlQuery
urlQuery, CancellationToken token)
        => Ok(await _mediator.Send(new InboxAllQuery(urlQuery), token));

      [Route("outbox/all")]
      [ApiExplorerSettings(GroupName = "v2")]
      [HttpGet]
   [SwaggerOperation(Summary = "ROUTE-CLIENT: tasks/outbox, MENU:
Εξερχόμενες/Όλες", Description = "Επιστροφή Ολων των Εξερχόμενων
Εργασιών(Task)")]
    [SwaggerResponse(200, "Επιστρέφει αντικείμενο τύπου Response με
δεδομένα λίστα από αντικείμενα OutboxAllDto",
typeof(Response<List<OutboxAllDto>>))]
    public async Task<IActionResult> GetAllOutboxTasks([FromQuery] UrlQuery
urlQuery, CancellationToken token)
```

```
        => Ok(await _mediator.Send(new OutboxAllQuery(urlQuery), token));
}
```

Στο παραπάνω παράδειγμα παρατηρούμε ότι για το σημείο εξόδου api/v2/tasks/inbox/all γίνεται χρήση του InboxAllQuery όπου λαμβάνει ως είσοδο ένα αντικείμενο UrlQuery (η χρήση του γίνεται με σκοπό να επιστρέφονται τα αποτελέσματα σελιδοποιημένα και φιλτραρισμένα) και επιστρέφει ένα JSON με λίστα αντικειμένων τύπου InboxAllDto. Όμοια λειτουργεί και το endpoint api/v2/tasks/outbox/all. Συνεπώς για το endpoint api/v2/tasks/inbox/all θα είχαμε το εξής αποτέλεσμα:


```
{
  "data": [
    {
      "id": "659503caf66ed65af7b8d453",
      "priority": "ΚΟΙΝΟ",
      "status": "Δεν Εεκίνησε",
      "subject": "erfef",
      "from": "ΔΙΕΥΘΥΝΤΗΣ",
      "startDate": "03/01/2024",
      "dueDate": "04/01/2024",
      "percentage": 0,
      "isAcknowledged": false,
      "description": "",
      "actions": null,
      "childrenTasksCount": 0,
      "documents": []
    },
    {
      "id": "65492950d74e78fd3f0345ec",
      "priority": "ΚΟΙΝΟ",
      "status": "Δεν Εεκίνησε",
      "subject": "Test task 2!!!Νομοθεσία Οργάνωσης και Λειτουργίας Ενόπλων
Δυνάμεων (ΕΔ) - ΟΡΘΗ ΕΠΑΝΑΛΗΨΗ",
      "from": "ΤΜΗΜΑΤΑΡΧΗΣ",
      "startDate": "06/11/2023",
      "dueDate": "07/11/2023",
      "percentage": 0,
      "isAcknowledged": true,
      "description": "Test task 2!!!",
      "actions": null,
      "childrenTasksCount": 0,
      "documents": [
        {
          "id": "653766b13e77892d86ef3819",
          "description": "Νομοθεσία Οργάνωσης και Λειτουργίας Ενόπλων
Δυνάμεων (ΕΔ) - ΟΡΘΗ ΕΠΑΝΑΛΗΨΗ",
          "isMaster": false
        }
      ]
    }
  ],
  "pagination": {
    "pageNumber": 1,
    "pageSize": 10,
    "totalRecords": 243,
    "totalPages": 25
  },
  "actions": []
}
```

Όμοια για το endpoint api/v2/tasks/outbox/al θα είχαμε το εξής αποτέλεσμα:


```
{
  "data": [
    {
      "id": "6544dd64d57c53ddc750d2c0",
      "priority": "ΚΟΙΝΟ",
      "status": "Δεν Ξεκίνησε",
      "subject": "Υποεργασία 1: Υποεργασία 1: Νομοθεσία Οργάνωσης και
Λειτουργίας Ενόπλων Δυνάμεων (ΕΔ)",
      "recipient": "ΟΔΗΓΟΣ ΟΧΗΜΑΤΩΝ",
      "startDate": "03/11/2023",
      "dueDate": "04/11/2023",
      "percentage": 0,
      "lastUpdate": "03/11/2023",
      "description": "",
      "actions": null,
      "documents": [
        {
          "id": "6544bb3c555b008c2f068683",
          "description": "Νομοθεσία Οργάνωσης και Λειτουργίας Ενόπλων
Δυνάμεων (ΕΔ)",
          "isMaster": false
        }
      ]
    },
    {
      "id": "58d4fdfe44b95544601e977f",
      "priority": "ΚΟΙΝΟ",
      "status": "Ολοκληρώθηκε",
      "subject": "ΑΔ.Φ.000/795865/Σ.190/03-08-16/ΓΕΑ/Δ1",
      "recipient": "ΤΜΗΜΑΤΑΡΧΗΣ",
      "startDate": "24/03/2017",
      "dueDate": "24/03/2017",
      "percentage": 100,
      "lastUpdate": "06/11/2017",
      "description": "Επικαιροποίηση ιστορικών στοιχείων Δεικτών",
      "actions": null,
      "documents": [
        {
          "id": "57a1d1eb44b9552b00ca717f",
          "description": "ΑΔ.Φ.000/795865/Σ.190/03-08-16/ΓΕΑ/Δ1",
          "isMaster": false
        }
      ]
    }
  ],
  "pagination": {
    "pageNumber": 1,
    "pageSize": 10,
    "totalRecords": 3,
    "totalPages": 1
  },
  "actions": []
}
```


Ενώ ένας Controller που εκτελεί εντολές επί των εργασιών όπως η δημιουργία, ενημέρωση και διαγραφή θα ονομαζόταν TasksCommandController και θα είχε την εξής μορφή:

```
[Route("api/v2/tasks")]
[ApiController]
public class TasksCommandsController : BaseController
{
      [ApiExplorerSettings(GroupName = "v2")]
      [HttpPost]
      [SwaggerOperation(Summary = "Δημιουργία Εργασίας", Description =
"Δημιουργία Εργασίας")]
    [SwaggerResponse(200, "Επιστρέφει αντικείμενο τύπου CommandResponse με
δεδομένα λίστα από string", typeof(CommandResponse<List<string>>))]
    public async Task<IActionResult> CreateTask([FromBody] CreateTaskDto
dto, CancellationToken token)
        => Ok(await _mediator.Send(new CreateTaskCommand(dto), token));


      [Route("{id}")]
      [ApiExplorerSettings(GroupName = "v2")]
      [HttpPut]
      [SwaggerOperation(Summary = "Ενημέρωση Εργασίας με {id}", Description
= "Ενημέρωση Εργασίας με {id}")]
    [SwaggerResponse(200, "Επιστρέφει αντικείμενο τύπου CommandResponse με
δεδομένα string", typeof(CommandResponse<string>))]
    public async Task<IActionResult> UpdateTask(string id, [FromBody]
UpdateTaskDto dto, CancellationToken token)
        => Ok(await _mediator.Send(new UpdateTaskCommand(id, dto), token));


      [Route("{id}")]
      [ApiExplorerSettings(GroupName = "v2")]
      [HttpDelete]
      [SwaggerOperation(Summary = "Διαγραφή Εργασίας", Description = "")]
      [SwaggerResponse(200, "Επιστρέφει αντικείμενο τύπου CommandResponse
με δεδομένα string", typeof(CommandResponse<string>))]
    public async Task<IActionResult> DeleteTask([FromRoute]
DeleteTaskCommand command, CancellationToken token)
        => Ok(await _mediator.Send(command, token));
}
```

Στο παραπάνω παράδειγμα παρατηρούμε ότι για το σημείο εξόδου POST api/v2/tasks/inbox γίνεται χρήση του CreateTaskCommand όπου λαμβάνει ως είσοδο ένα αντικείμενο CreateTaskDto και επιστρέφει ένα JSON με ένα μήνυμα επιβεβαίωσης για την εκτέλεση της ενέργειας. Όμοια λειτουργεί και το PUT endpoint api/v2/tasks/{id}. Συνεπώς για το endpoint POST api/v2/tasks ο client της εφαρμογής θα έστελνε το εξής αντικείμενο με όνομα, περιγραφή, ημερομηνίες, προτεραιότητα, αποδέκτες και σχετικά για τη νέα εργασία:

```
{
    "subject": "Νέα Εργασία",
    "text": "Κάποια περιγραφή για την εργασία",
    "startDate": "2024-01-22T10:16:00.000Z",
    "dueDate": "2024-01-23T21:59:00.000Z",
    "priority": 0,
    "recipients": [
        {
            "employeeName": "Α' (ΔΕ ΠΡΟΣΩΠΙΚΟΥ Η/Υ ) ΔΕΣΠΟΙΝΑ",
        }
    ],
    "references": [
        {
            "id": "6542361b8f20832dc2672ab8",
```

```
            "name": "ΑΔ.Φ.001/96/Σ.48/23-11-23/ΓΕΑ/Γ5",
            "documentType": 1,
            "referenceType": 0
        },
        {
            "id": null,
            "name": "σχετικό",
            "referenceType": 2
        },
        {
            "id": "65ae40e02356e0353e09b420",
            "referenceType": 2,
            "name": "Εγγραφο.docx"
        },
        {
            "id": "5a60752a44b95519dc841e49",
            "name": "ΠαΔ 6-17/2018/ΓΕΑ",
            "referenceType": 1
        }
    ]
}
```


Η εκτέλεση της εντολής CreateTaskCommand έχει ως αποτέλεσμα ο client της εφαρμογής να λαμβάνει το εξής μήνυμα επιβεβαίωσης:


```
{
  "data": "Η εργασία με id 65ae41012356e0353e09b448 δημιουργήθηκε επιτυχώς"
}
```

4 Φάση Migration

4.1 Η σημασία της Migration φάσης

Η φάση του migration είναι το κομβικό στάδιο της διαδικασίας μετάβασης μιας εφαρμογής ιστού React σε μια εφαρμογή για κινητά, χρησιμοποιώντας το Ionic Framework, όπου οι προγραμματιστές θέτουν σε εφαρμογή τα σχέδια και τις γνώσεις που συγκεντρώθηκαν κατά τη φάση της προ - μετάβασης (Φάση Pre-Migration). Η φάση αυτή περιλαμβάνει μια συστηματική, βήμα προς βήμα διαδικασία μετατροπής των στοιχείων, της λογικής και των χαρακτηριστικών της React στο Ionic Framework. Οι προγραμματιστές εμβαθύνουν στο codebase της React, εντοπίζοντας και προσαρμόζοντας στοιχεία και βιβλιοθήκες στα αντίστοιχα του Ionic. Βαρύτητα δίνεται στους τομείς της δρομολόγησης και της πλοήγησης, οι οποίες μπορεί να διαφέρουν σημαντικά μεταξύ των δύο αυτών πλαισίων (frameworks). Μια πρωταρχική πρόκληση κατά τη διάρκεια της μετάβασης αυτής, είναι η διασφάλιση της απρόσκοπτης μετάβασης της διαχείρισης κατάστασης, ειδικά εάν η εφαρμογή React βασίζεται σε συγκεκριμένες βιβλιοθήκες διαχείρισης κατάστασης, όπως είναι η Redux ή η Mobx. Οι προγραμματιστές πρέπει να επινοήσουν και να ακολουθήσουν στρατηγικές για την αποτελεσματική μεταφορά και προσαρμογή της λογικής διαχείρισης καταστάσεων εντός του Ionic Framework.

Επιπλέον, η φάση αυτή απαιτεί την αντιμετώπιση πιθανών προβλημάτων συμβατότητας μεταξύ των στοιχείων React και Ionic και τη λεπτομερή ρύθμιση της διεπαφής χρήστη ώστε να τηρούνται οι οδηγίες σχεδιασμού και οι αρχές responsive του Ionic. Καθ' όλη τη διάρκεια αυτής της διαδικασίας, οι ολοκληρωμένες δοκιμές και η αποσφαλμάτωση είναι ζωτικής σημασίας για να εντοπιστούν τυχόν προβλήματα και να διασφαλιστεί η άψογη λειτουργία της εφαρμογής στις κινητές συσκευές. Η πρόκληση της διατήρησης μιας καθαρής και οργανωμένης δομής έργου δεν μπορεί να υποτιμηθεί, καθώς μια καλά δομημένη βάση κώδικα διευκολύνει σημαντικά τη διαδικασία μετάβασης και τη μελλοντική συντήρηση της εφαρμογής.

Ακόμα, η φάση της μετάβασης είναι μια κρίσιμη διαδικασία όπου οι προγραμματιστές πρέπει να περιηγηθούν σχολαστικά στις ενδεχόμενες επιπλοκές αυτής της μετάβασης από το React στο Ionic, να αντιμετωπίσουν ζητήματα συμβατότητας και να βελτιστοποιήσουν την βάση κώδικα για να δημιουργήσουν μια πλήρως λειτουργική, αποδοτική και ανταποκρινόμενη εφαρμογή για κινητά. Απαιτείται ισορροπία μεταξύ της διατήρησης της βασικής λειτουργικότητας της αρχικής εφαρμογής React και της υιοθέτησης των αρχών mobile-first του πλαισίου Ionic. Η επιτυχής πλοήγηση σε αυτές τις προκλήσεις έχει ως αποτέλεσμα μια εφαρμογή για κινητά που αξιοποιεί τα πλεονεκτήματα του Ionic, διατηρώντας παράλληλα την ουσία της αρχικής εφαρμογής React.

Αναλυτικότερα, η φάση αυτή χαρακτηρίζεται από μια σειρά σαφώς καθορισμένων βημάτων και προκλήσεων που πρέπει να αντιμετωπιστούν για μια επιτυχημένη μετάβαση.

4.1.1 Μετατροπή στοιχείων:

Σε αυτή τη φάση, οι προγραμματιστές πραγματοποιούν λεπτομερή ανάλυση της βάσης κώδικα React, εντοπίζοντας τα στοιχεία και τα χαρακτηριστικά που πρέπει να προσαρμοστούν στο Ionic. Τα συστατικά React, τα οποία αποτελούν τα δομικά στοιχεία της διαδικτυακής εφαρμογής, πρέπει να μετασχηματιστούν συστηματικά στα αντίστοιχα Ionic. Αυτό

περιλαμβάνει την επανεγγραφή και την αναδιαμόρφωση του κώδικα για να εξασφαλιστεί η συμβατότητα με τη δομή και τα πρότυπα σχεδίασης του Ionic.

4.1.2 Δρομολόγηση και πλοήγηση:

Μία από τις πρωταρχικές προκλήσεις κατά τη μετάβαση είναι ο χειρισμός της δρομολόγησης και της πλοήγησης, καθώς πρόκειται για θεμελιώδεις πτυχές κάθε εφαρμογής ιστού ή κινητής τηλεφωνίας. Το React και το Ionic έχουν διαφορετικά παραδείγματα πλοήγησης, τα οποία απαιτούν από τους προγραμματιστές να επαναρυθμίσουν προσεκτικά τη λογική δρομολόγησης της εφαρμογής. Ακόμα, θα πρέπει οι προγραμματιστές να λάβουν πολύ σοβαρά υπόψη ότι η πλοήγηση θα γίνεται σε κινητές συσκευές καθώς και με ότι αυτό συνοδεύεται. Π.χ. οι χρήστες των κινητών συσκευών χρησιμοποιούν πολύ συχνότερα κινήσεις του χεριού τύπου swipe, ενώ χρησιμοποιούν περισσότερο το “πίσω”, απ’ ότι οι χρήστες του υπολογιστή.

4.1.3 Μετατροπή στη διαχείριση καταστάσεων:

Η διαχείριση της κατάστασης της εφαρμογής είναι κρίσιμη για τη διατήρηση της ακεραιότητας της λειτουργικότητας της εφαρμογής. Πολλές εφαρμογές React βασίζονται σε βιβλιοθήκες διαχείρισης κατάστασης όπως η Redux, η Recoil ή η Mobx. Η μετάβαση της διαχείρισης κατάστασης από το React στο Ionic απαιτεί στρατηγικό σχεδιασμό. Οι προγραμματιστές πρέπει να καθορίσουν αν θα συνεχίσουν να χρησιμοποιούν αυτές τις βιβλιοθήκες ή θα αξιοποιήσουν τις λύσεις διαχείρισης κατάστασης του Ionic. Αυτό περιλαμβάνει την προσαρμογή των action creators, reducers και stores ώστε να ταιριάζουν στις απαιτήσεις του Ionic framework.

4.1.4 Προκλήσεις συμβατότητας:

Η μετάβαση μιας εφαρμογής React στο Ionic συχνά συνοδεύεται από προκλήσεις συμβατότητας. Τα στοιχεία React και Ionic ενδέχεται να μην ενσωματώνονται απρόσκοπτα, απαιτώντας από τους προγραμματιστές να αντιμετωπίσουν ζητήματα που σχετίζονται με τις μεθόδους του κύκλου ζωής των στοιχείων (component lifecycle), το χειρισμό συμβάντων και τη δέσμευση δεδομένων. Η αντιμετώπιση αυτών των προκλήσεων συμβατότητας είναι μια κρίσιμη πτυχή της φάσης της μετάβασης.

4.1.5 Διεπαφή χρήστη και μορφοποίηση:

Για να διασφαλιστεί μια συνεκτική εμπειρία χρήστη, οι προγραμματιστές πρέπει να προσαρμόσουν τα στοιχεία της διεπαφής χρήστη (UI) και το styling ώστε να συμμορφώνονται με τις αρχές σχεδιασμού του Ionic και τις κατευθυντήριες γραμμές σχεδιασμού responsive design. Αυτό περιλαμβάνει την τροποποίηση των κλάσεων CSS, την προσαρμογή των δομών διάταξης και την ενσωμάτωση του εκτεταμένου συνόλου στοιχείων UI του Ionic.

4.1.6 Δοκιμές και αποσφαλμάτωση:

Οι ολοκληρωμένες δοκιμές και η αποσφαλμάτωση είναι απαραίτητες κατά τη φάση της μετάβασης. Οι προγραμματιστές θα πρέπει να χρησιμοποιούν εργαλεία εντοπισμού σφαλμάτων ειδικά για την ανάπτυξη Ionic, όπως το Ionic DevApp, και να εκτελούν ενδελεχείς

δοκιμές σε διάφορες κινητές συσκευές και μεγέθη οθόνης. Αυτή η φάση διασφαλίζει ότι η μεταφερόμενη εφαρμογή λειτουργεί σωστά και αποδίδει βέλτιστα στις κινητές πλατφόρμες.

4.1.7 Δομή και οργάνωση του κώδικα:

Η διατήρηση μιας καθαρής και οργανωμένης δομής του έργου είναι ζωτικής σημασίας για την αναγνωσιμότητα, τη συντηρησιμότητα και την επεκτασιμότητα του κώδικα. Η διασφάλιση της καλής δομής της βάσης κώδικα απλοποιεί τη διαδικασία μετάβασης και διευκολύνει τη μελλοντική συντήρηση των εφαρμογών.

Συνοψίζοντας, η φάση της μετάβασης είναι ένα σύνθετο και περίπλοκο στάδιο που απαιτεί από τους προγραμματιστές να συνδυάζουν τεχνικές δεξιότητες με βαθιά κατανόηση τόσο της React όσο και των πλαισίων Ionic. Περιλαμβάνει σχολαστική μετάβαση συστατικών, προσαρμογή της δρομολόγησης και της πλοήγησης, προσεκτική μετάβαση της διαχείρισης καταστάσεων, αντιμετώπιση των προκλήσεων συμβατότητας, ευθυγράμμιση του UI και του styling με τις κατευθυντήριες γραμμές του Ionic, αυστηρές δοκιμές και αποσφαλμάτωση και διατήρηση μιας οργανωμένης δομής κώδικα. Η επιτυχής πλοήγηση σε αυτές τις προκλήσεις έχει ως αποτέλεσμα μια πλήρως λειτουργική εφαρμογή Ionic για κινητά που αξιοποιεί τα πλεονεκτήματα και των δύο πλαισίων, παρέχοντας παράλληλα μια απρόσκοπτη εμπειρία χρήστη σε κινητές συσκευές.

4.2 Διαφορετικές στρατηγικές migration: Full migration vs Gradual migration

Κατά τη διάρκεια της μετάβασης μιας εφαρμογής ιστού React σε μια εφαρμογή Ionic για κινητά, μία από τις καίριες αποφάσεις που πρέπει να λάβουν οι προγραμματιστές είναι η επιλογή της καταλληλότερης στρατηγικής μετάβασης. Η απόφαση αυτή διαμορφώνει θεμελιωδώς τη διαδικασία μετάβασης και έχει σημαντικό αντίκτυπο στο χρονοδιάγραμμα του έργου, στην κατανομή των πόρων και στη συνολική επιτυχία.

4.2.1 Full Migration

Η στρατηγική πλήρους μετάβασης περιλαμβάνει την ταυτόχρονη και ολοκληρωμένη μετάβαση ολόκληρης της εφαρμογής React στο Ionic framework. Αυτή η προσέγγιση χαρακτηρίζεται από μια πλήρη μετάβαση από το React στο Ionic, όπου όλα τα στοιχεία, τα χαρακτηριστικά και οι λειτουργίες μεταφέρονται ταυτόχρονα.

4.2.1.1 Πλεονεκτήματα

- Εξορθολογισμένη ανάπτυξη: Η πλήρης μετάβαση οδηγεί συχνά σε ταχύτερη πρόοδο της ανάπτυξης, δεδομένου ότι όλες οι προσπάθειες ανάπτυξης συγκεντρώνονται σε μία μόνο πλατφόρμα, την Ionic.
- Συνέπεια: Η προκύπτουσα εφαρμογή Ionic είναι πιο πιθανό να διατηρήσει μια συνεπή εμπειρία χρήστη και μια συνεπή βάση κώδικα, καθώς χρησιμοποιεί εξ ολοκλήρου την αρχιτεκτονική και τα πρότυπα σχεδίασης της Ionic.

- Άμεση παρουσία σε κινητά τηλέφωνα: Εάν ο χρόνος είναι σημαντικός, μια πλήρης μετάβαση μπορεί να παραδώσει γρήγορα μια εφαρμογή για κινητά, επιτρέποντας στις επιχειρήσεις να αξιοποιήσουν άμεσα την αγορά κινητών τηλεφώνων.

4.2.1.2 Εκτιμήσεις

- Ένταση πόρων: Η πλήρης μετάβαση μπορεί να είναι εντατική σε πόρους, απαιτώντας σημαντική επένδυση χρόνου και προσπάθειας για να διασφαλιστεί μια απρόσκοπτη μετάβαση χωρίς διακοπές.
- Πολυπλοκότητα δοκιμών: Οι αυστηρές δοκιμές είναι ζωτικής σημασίας μετά την πλήρη μετάβαση για τον εντοπισμό και τη διόρθωση τυχόν προβλημάτων ή ασυμφωνιών που προκύπτουν από την πλήρη μετάβαση.
- Χρόνος διακοπής λειτουργίας: Ανάλογα με την πολυπλοκότητα της εφαρμογής, ενδέχεται να υπάρξει μια περίοδος διακοπής λειτουργίας κατά την οποία η εφαρμογή δεν θα είναι διαθέσιμη κατά τη διάρκεια της μετάβασης.

4.2.2 Gradual Migration

Μια στρατηγική σταδιακής μετάβασης, αντίθετα, περιλαμβάνει μια σταδιακή προσέγγιση όπου συγκεκριμένα τμήματα, χαρακτηριστικά ή ενότητες της εφαρμογής React μεταφέρονται σταδιακά στο Ionic με την πάροδο του χρόνου. Αυτή η προσέγγιση επιτρέπει στους προγραμματιστές να δώσουν προτεραιότητα σε ποια τμήματα της εφαρμογής θα μετατραπούν πρώτα, διατηρώντας παράλληλα την υπάρχουσα εφαρμογή React λειτουργική κατά τη διάρκεια της μετάβασης.

4.2.2.1 Πλεονεκτήματα

- Ευελιξία: Η σταδιακή μετάβαση παρέχει μεγαλύτερη ευελιξία όσον αφορά τη διαχείριση του έργου και την κατανομή των πόρων. Επιτρέπει στις ομάδες να αντιμετωπίσουν τη μετάβαση σε μικρότερα, διαχειρίσιμα κομμάτια.
- Ελάχιστη διακοπή: Δεδομένου ότι η εφαρμογή React παραμένει άθικτη κατά τη διάρκεια της διαδικασίας μετάβασης, υπάρχει ελάχιστη διαταραχή στη συνεχιζόμενη ανάπτυξη και στην πρόσβαση των χρηστών.
- Μειωμένος κίνδυνος: Οι προγραμματιστές μπορούν να δοκιμάσουν διεξοδικά κάθε μεταφερόμενο τμήμα πριν προχωρήσουν στο επόμενο, μειώνοντας τον κίνδυνο εισαγωγής κρίσιμων ζητημάτων στην εφαρμογή.

4.2.2.2 Εκτιμήσεις

- Μεγαλύτερο χρονοδιάγραμμα: Η σταδιακή μετάβαση μπορεί να παρατείνει το συνολικό χρονοδιάγραμμα του έργου, καθώς η μετάβαση πραγματοποιείται σε φάσεις, οδηγώντας ενδεχομένως σε μεγαλύτερο χρονικό διάστημα μέχρι να είναι διαθέσιμη η πλήρης εφαρμογή Ionic.
- Προκλήσεις ενσωμάτωσης: Η διασφάλιση της απρόσκοπτης ενσωμάτωσης μεταξύ της υπάρχουσας εφαρμογής React και των νέων μεταφερόμενων τμημάτων Ionic μπορεί να δημιουργήσει προκλήσεις, απαιτώντας προσεκτικό συντονισμό.

- Διαχείριση πολυπλοκότητας: Η διαχείριση των εξαρτήσεων και η διατήρηση μιας συνεκτικής εμπειρίας χρήστη και στα δύο πλαίσια απαιτεί σχολαστικό σχεδιασμό.

Η επιλογή μεταξύ πλήρους και σταδιακής μετάβασης θα πρέπει να καθοδηγείται από συγκεκριμένους παράγοντες του έργου, συμπεριλαμβανομένου του μεγέθους και της πολυπλοκότητας της εφαρμογής React, των διαθέσιμων πόρων ανάπτυξης, των χρονικών περιορισμών και της ανάγκης για συνεχή διαδικτυακή εφαρμογή κατά τη διάρκεια της μετάβασης. Τελικά, η επιλεγμένη στρατηγική θα πρέπει να ευθυγραμμιστεί με τους στόχους του έργου και τη διαθεσιμότητα των πόρων, ώστε να διασφαλιστεί η επιτυχής μετάβαση από το React στο Ionic, είτε μέσω μιας ταχείας πλήρους μετάβασης είτε μέσω μιας σταδιακής, ευέλικτης προσέγγισης που ελαχιστοποιεί τη διακοπή και τον κίνδυνο.

Συμπερασματικά, η στρατηγική μετάβασης που επιλέγεται είναι μια καίρια απόφαση που διαμορφώνει ολόκληρη τη διαδικασία μετάβασης και θα πρέπει να λαμβάνεται με σύνεση βάσει των μοναδικών απαιτήσεων και περιορισμών κάθε έργου.

4.3 Επιλογή της κατάλληλης στρατηγικής migration

Η απόφαση της ομάδας να επιλέξει τη στρατηγική Gradual Migration ανάμεσα στις δύο προτεινόμενες στρατηγικές για το Σύστημα Ηλεκτρονικής Διαχείρισης Εγγράφων "Ίριδα" αντικατοπτρίζει την κατανόηση της ιδιαίτερης φύσης της εφαρμογής. Το "Ίριδα" αναπτύσσεται για να υποστηρίξει διάφορες πτυχές της γραφειοκρατίας διαφόρων οργανισμών, και αυτό σημαίνει ότι πρέπει να προσαρμόζεται συνεχώς στις διαφορετικές απαιτήσεις που εμφανίζονται.

Η επιλογή του Gradual Migration επιτρέπει στην εφαρμογή να εξελίσσεται σταδιακά χωρίς να υστερεί στις τρέχουσες αλλαγές που χρειάζεται να πραγματοποιηθούν. Αυτό είναι σημαντικό, διότι οι ανάγκες μπορεί να αλλάζουν ταχέως, είτε πρόκειται για διορθώσεις σφαλμάτων (bugs) είτε για νέα χαρακτηριστικά (features), και αυτή η ευελιξία είναι κρίσιμη για να διατηρηθεί η αποτελεσματικότητα και η αποδοτικότητα του συστήματος.

Συνολικά, η επιλογή αυτή αντανακλά τη στρατηγική της συνεχούς βελτίωσης και προσαρμογής της εφαρμογής "Ίριδα" στις ανάγκες των οργανισμών που τη χρησιμοποιούν, ενώ ταυτόχρονα εξασφαλίζει τη σταθερότητα και την απρόσκοπτη λειτουργία του συστήματος.

4.4 Εφαρμογή του σχεδίου Migration

4.4.1 Μετατροπή των στοιχείων της εφαρμογής σε Mobile Friendly

Η μετάβαση μιας React εφαρμογής στο Ionic Framework ενώ διασφαλίζεται η ανταποκριτικότητα στις κινητές συσκευές είναι μια κρίσιμη διαδικασία, και επιτυγχάνεται αποτελεσματικά μέσω του συνδυασμού CSS, clsx, Bootstrap και προσαρμοσμένων components που είναι σχεδιασμένα ειδικά για κινητές συσκευές. Λόγω τους μεγέθους της βάσης κώδικα, αλλά και του πλήθους των στοιχείων που χρειάστηκε να προγραμματιστούν, δεν μπορούν να αναφερθούν εδώ. Για τον λόγο αυτό, αναφέρονται παρακάτω οι στρατηγικές που ακολουθήθηκαν.

4.4.1.1 Responsive Σχεδιασμός με CSS

Χρησιμοποιώντας CSS ορίστηκαν διάφορα στυλ για διάφορα μεγέθη οθονών. Αυτό επιτρέπει να προσαρμόζουμε τη διάταξη και το περιεχόμενο της εφαρμογής μας ώστε να ταιριάζουν στον διαθέσιμο χώρο οθόνης της εκάστοτε συσκευής. Ξεκινώντας με μια προσέγγιση προσανατολισμένη στα κινητά, σχεδιάζοντας έτσι την εφαρμογή αρχικά για μικρές οθόνες και πηγαίνοντας σταδιακά για μεγαλύτερες, καλύπτοντας έτσι ένα μεγάλο μερίδιο χρηστών.

4.4.1.2 Bootstrap για Responsive Grid

Οι κλάσεις πλέγματος του Bootstrap όπως `container`, `row` και `col` βοηθούν στο να δομήσουμε τη διάταξη της εφαρμογής σας ώστε να προσαρμόζεται αυτόματα στο μέγεθος της οθόνης. Ακόμα, ιδιαίτερα συχνές είναι και οι εντολές flexbox, όπως ‘d-flex’.

4.4.1.3 clsx για Conditional Styling:

Χρησιμοποιώντας τη βιβλιοθήκη cslx μπορούμε να επιτύχουμε μια δυναμική προσθήκη ή αφαίρεση κλάσεων με βάση το μέγεθος της οθόνης ή άλλες συνθήκες. Για παράδειγμα, μπορούμε να εφαρμόζουμε διαφορετικές κλάσεις για κινητά και επιτραπέζιες διατάξεις χρησιμοποιώντας το clsx για να ελέγχουμε το στυλ στην εκάστοτε περίπτωση.

4.4.1.4 Προσαρμοσμένα Components για Κινητές Συσκευές

Δεδομένου ότι οι κινητές συσκευές έχουν μοναδικές διεπαφές χρήστη και περιορισμούς, η δημιουργία προσαρμοσμένων στοιχείων που είναι σχεδιασμένα ειδικά για αυτές τις συσκευές χρησιμοποιείται σε πολλές περιπτώσεις. Αυτά τα στοιχεία μπορούν να βελτιστοποιήσουν την εμπειρία του χρήστη προσφέροντας μια πιο εύκολη και αποτελεσματική διεπαφή για τις μικρότερες οθόνες αφής.

4.4.1.5 Δοκιμές και Αποσφαλμάτωση

Δοκιμάζοντας τακτικά την εφαρμογή μας σε διάφορες συσκευές και μεγέθη οθονών για να εξασφαλίσετε ότι φαίνεται και λειτουργεί όπως αναμένεται. Τα εργαλεία προγραμματιστή του προγράμματος περιήγησης (browser), οι προσομοιωτές συσκευών και πραγματικές συσκευές είναι πολύ χρήσιμα για την αποσφαλμάτωση και τον εκσυγχρονισμό του ανταποκρινούμενου σχεδιασμού που ακολουθούμε.

4.4.1.6 Σκέψεις για την Εμπειρία Χρήστη (UX)

Η ανταποκριτικότητα (responsiveness) της εφαρμογής δεν αφορά μόνο τη διάταξη, αλλά και την εξασφάλιση μιας ασφαλούς εμπειρίας χρήστη. Χρησιμοποιήθηκαν επιπλέον κάποια στοιχεία που είναι εύκολα προσβάσιμα με την αφή, την πλοήγηση, το μέγεθος γραμματοσειράς και τις κινητικές χειρονομίες (swipes και gestures) για να βελτιώσουμε την χρηστικότητα της εφαρμογής μας στις κινητές συσκευές.

4.4.1.7 Βελτιστοποίηση της Απόδοσης

Η ανταποκριτικότητα πρέπει να λαμβάνει υπόψη και την απόδοση. Ελαχιστοποιώντας τις περιττές κινήσεις και βελτιστοποιώντας το μέγεθος των σελίδων για γρήγορους χρόνους φόρτωσης σε κινητές συνδέσεις.

Με την ενσωμάτωση αυτών των στρατηγικών, μπορούμε να κάνουμε με επιτυχία το migration της εφαρμογής React στο Ionic Framework και να δημιουργήσουμε έναν ανταποκρινόμενο σχεδιασμό που παρέχει μια ομαλή εμπειρία χρήστη σε διάφορες κινητές συσκευές και μεγέθη οθονών.

4.4.2 Αντιμετώπιση των διαφορών σε συγκεκριμένες πλατφόρμες

Παρακάτω παρουσιάζονται δύο hooks, το useIsMobile και το useWindowSize τα οποία χρησιμοποιούνται σε διάφορα σημεία στην εφαρμογή και τα οποία είναι ιδιαίτερα χρήσιμα καθώς βοηθάνε στο να παίρνουμε πληροφορίες για τις διαστάσεις οθόνης του χρήστη, αλλά και για τον browser που χρησιμοποιεί ο χρήστης. Αυτό οδηγεί ώστε να προσαρμόζεται καλύτερα η εκάστοτε οθόνη - σελίδα της εφαρμογής στην οθόνη του χρήστη, είτε κινητής συσκευής, είτε επιτραπέζιας.

- **useIsMobile()**: Το παρακάτω React hook, useIsMobile, χρησιμοποιείται για να καθορίσει εάν το τρέχον περιβάλλον είναι μία κινητή συσκευή ή όχι. Αυτό το hook χρησιμοποιεί τα hooks useState και useEffect για να διατηρεί και να ενημερώνει την κατάσταση που υποδεικνύει εάν ο χρήστης βρίσκεται σε κινητή συσκευή ή όχι. Παρακάτω παρέχεται μια λεπτομερής εξήγηση γραμμή προς γραμμή του κώδικα μαζί με επιπλέον πληροφορίες:

```
// DevTip: When you change the browser / mobile view from developers tools
it is not working. You have to reload with the new view, so you will inform
it if the new browser will be mobile or desktop.

Αυτό το σχόλιο δίνει μια συμβουλή προγραμματιστή και αναφέρει ότι η
λειτουργία του κώδικα μπορεί να επηρεαστεί από τις αλλαγές προβολής στα
εργαλεία προγραμματιστή στον browser. Για να δουλέψει σωστά, πρέπει να
ανανεώσετε τη σελίδα με τη νέα προβολή για να ενημερώσει αν το νέο
πρόγραμμα περιήγησης είναι κινητό ή επιτραπέζιο.

import { useState, useEffect } from 'react';
//Αυτή η γραμμή εισάγει τις συναρτήσεις useState και useEffect από τη
βιβλιοθήκη 'react'.

export function useIsMobile() {
  // Αυτή η γραμμή ορίζει ένα προσαρμοσμένο React hook με το όνομα
useIsMobile. Αυτό το hook χρησιμοποιείται για να αναγνωρίζει αν ο
περιηγητής είναι σε κινητή συσκευή (mobile) ή όχι.

  const [isMobile, setIsMobile] = useState(false);
// Αυτή η γραμμή αρχικοποιεί μια μεταβλητή κατάστασης με το όνομα isMobile
χρησιμοποιώντας το hook useState. Αρχικά, η τιμή της isMobile ορίζεται ως
false

  useEffect(() => {
// Αυτή η γραμμή ξεκινά ένα μπλοκ useEffect, το οποίο εκτελείται όταν το
στοιχείο αναρτάται στο component tree.
       const userAgent =
      typeof window.navigator === 'undefined' ? '' : navigator.userAgent;
//Εδώ, δημιουργείται μια μεταβλητή userAgent που περιέχει το user agent
string του περιηγητή. Αν το window.navigator είναι undefined, η μεταβλητή
παίρνει την τιμή κενού, αλλιώς παίρνει τον user agent string.
    const mobile =
```

```
      /Android|webOS|iPhone|iPad|iPod|BlackBerry|IEMobile|Opera
Mini/i.test(
        userAgent
      );
//Εδώ, δημιουργείται μια μεταβλητή mobile, η οποία ελέγχει αν το user agent
string περιέχει οποιαδήποτε από τις συμβατές με κινητά λογικές συσκευές. Αν
ναι, η mobile γίνεται true, αλλιώς παραμένει false.

        setIsMobile(mobile);
//Αυτή η γραμμή ενημερώνει την κατάσταση isMobile με την τιμή της mobile,
που δηλώνει αν ο περιηγητής είναι κινητού.
  }, []);
//Αυτή η γραμμή καθορίζει έναν κενό πίνακα εξαρτήσεων, πράγμα που σημαίνει
ότι το useEffect θα εκτελεστεί μόνο μία φορά, ακριβώς μετά την ανάρτηση του
στοιχείου.

  return isMobile;
//Τέλος, αυτή η γραμμή επιστρέφει την τιμή της isMobile, η οποία
αντιπροσωπεύει αν ο περιηγητής θεωρείται mobile (κινητού) ή όχι.
}
```

- **useWindowSize():** Παρακάτω ορίζεται ένα προσαρμοσμένο hook React, το useWindowSize, που παρακολουθεί το πλάτος και το ύψος του παραθύρου χρησιμοποιώντας τα hooks useState και useEffect. Χρησιμοποιεί window.listener για να “ακούει” για γεγονότα αλλαγής μεγέθους του παραθύρου, ενημερώνοντας την κατάσταση αντίστοιχα και καθαρίζοντας τον event listener όταν το στοιχείο καταργείται. Αυτό το hook μπορεί να χρησιμοποιηθεί σε functional components για την πρόσβαση και την ανταπόκριση σε αλλαγές στις διαστάσεις του παραθύρου.

```
import { useEffect, useState } from 'react'
//Αυτή η γραμμή εισάγει τις συναρτήσεις useEffect και useState από τη
βιβλιοθήκη 'react', οι οποίες χρησιμοποιούνται για τη δημιουργία
προσαρμοσμένων συναρτήσεων (hooks) για τη διαχείριση των επιπτώσεων και της
κατάστασης των στοιχείων.

export function useWindowSize() {
//Αυτή η γραμμή ορίζει μια προσαρμοσμένη συνάρτηση React με το όνομα
useWindowSize. Οι προσαρμοσμένες συναρτήσεις (hooks) είναι συναρτήσεις που
ενσωματώνουν επαναχρησιμοποιήσιμη λογική και κατάσταση για τα στοιχεία.
  const [windowSize, setWindowSize] = useState({
    width: undefined,
    height: undefined
  })
//Αυτή η γραμμή αρχικοποιεί μια μεταβλητή κατάστασης με το όνομα windowSize
χρησιμοποιώντας το hook useState. Έχει μια αρχική τιμή της μορφής { width:
undefined, height: undefined }, υποδηλώνοντας ότι οι διαστάσεις είναι
αρχικά άγνωστες.

  useEffect(() => {
//Αυτή η γραμμή ξεκινά ένα μπλοκ useEffect, το οποίο επιτρέπει την εκτέλεση
side effects σε functional components.
    function handleResize() {
//Αυτή η γραμμή δηλώνει μια συνάρτηση με το όνομα handleResize, η οποία θα
κληθεί όταν αλλάξει το μέγεθος του παραθύρου.
      setWindowSize({ width: window.innerWidth, height: window.innerHeight
})
    }
```

```
//Εντός της συνάρτησης handleResize, αυτή η γραμμή ενημερώνει την κατάσταση
windowSize με το τρέχον πλάτος και ύψος του παραθύρου.

    window.addEventListener('resize', handleResize)
//Αυτή η γραμμή προσθέτει έναν ακροατή γεγονότων στο γεγονός αλλαγής
μεγέθους του παραθύρου, που θα καλέσει τη συνάρτηση handleResize όταν το
παράθυρο αλλάξει μέγεθος.

    handleResize()
//Αυτή η γραμμή καλεί αμέσως τη συνάρτηση handleResize για να ορίσει το
αρχικό windowSize με βάση τις τρέχουσες διαστάσεις του παραθύρου.

    return () => window.removeEventListener('resize', handleResize)
//Αυτή η γραμμή ορίζει μια συνάρτηση καθαρισμού που αφαιρεί τον ακροατή
γεγονότων για το γεγονός αλλαγής μεγέθους του παραθύρου όταν το στοιχείο
καταργείται.
  }, [])
//Αυτή η γραμμή καθορίζει έναν κενό πίνακα εξαρτήσεων, πράγμα που σημαίνει
ότι το useEffect θα εκτελεστεί μόνο μία φορά, ακριβώς μετά την ανάρτηση του
στοιχείου.

  return windowSize
//Τέλος, αυτή η γραμμή επιστρέφει το αντικείμενο windowSize, το οποίο
αντιπροσωπεύει τις τρέχουσες διαστάσεις του παραθύρου.
}
```

4.4.3 Κύριοι άξονες αλλαγών

4.4.3.1 Πίνακες

Οι σελίδες της εφαρμογής που περιέχουν πίνακες αποτελούν ένα μεγάλο μέρος της εφαρμογής. Για το συγκεκριμένο λόγο, όσον αφορά τους πίνακες, ακολουθήθηκε μια προσέγγιση responsive design όπου, στην προβολή της εφαρμογής για κινητά, εμφανίζεται ένας οριζόντιος πίνακας ως κατακόρυφη λίστα. Αυτό είναι ένα κοινό πρότυπο σχεδίασης που χρησιμοποιείται για τη βελτίωση της εμπειρίας του χρήστη σε μικρότερες οθόνες, όπως αυτές των κινητών συσκευών. Έτσι, όταν μετατρέπουμε έναν οριζόντιο πίνακα σε κατακόρυφη λίστα για προβολές σε κινητά, κάθε γραμμή του οριζόντιου πίνακα γίνεται ένα ξεχωριστό στοιχείο στην κατακόρυφη λίστα.

Αυτή η προσέγγιση έχει πολλά πλεονεκτήματα:

- Βελτιωμένη αναγνωσιμότητα: Σε μικρές οθόνες, οι οριζόντιοι πίνακες μπορεί να γίνουν δύσκολοι στην ανάγνωση και την πλοήγηση, επειδή οι στήλες μπορεί να μην χωράνε άνετα μέσα στο παράθυρο προβολής. Με τη μετατροπή τους σε κατακόρυφη λίστα, οι χρήστες μπορούν εύκολα να μετακινηθούν στα στοιχεία χωρίς την ανάγκη οριζόντιας κύλισης.
- Αξιοποίηση χώρου: Οι κατακόρυφες λίστες χρησιμοποιούν τον διαθέσιμο κατακόρυφο χώρο πιο αποτελεσματικά στις οθόνες κινητών τηλεφώνων, επιτρέποντας την εμφάνιση περισσότερο περιεχομένου χωρίς να κατακλύζεται ο χρήστης με πάρα πολλές πληροφορίες ταυτόχρονα.
- Φιλική προς την αφή: Οι κινητές συσκευές χρησιμοποιούν κυρίως διεπαφές αφής και οι κάθετες λίστες είναι πιο φιλικές προς την αφή από τους πίνακες με οριζόντιο

προσανατολισμό. Οι χρήστες μπορούν εύκολα να αγγίξουν μεμονωμένα στοιχεία για να αποκτήσουν πρόσβαση σε περισσότερες πληροφορίες ή να αναλάβουν δράση.

- Συνέπεια: Προσαρμόζοντας τη διάταξη του πίνακα στο μέγεθος της οθόνης, δημιουργείται μια πιο συνεπή εμπειρία χρήστη σε διαφορετικές συσκευές, διασφαλίζοντας ότι το περιεχόμενό \ παραμένει προσβάσιμο και χρησιμοποιήσιμο τόσο στην επιφάνεια εργασίας όσο και στο κινητό.

4.4.3.2 Μετακίνηση μπάρας ενεργειών

Όλες οι σελίδες της εφαρμογής περιέχουν την μπάρα ενεργειών (toolbar), όπου μέσω αυτής μπορεί να εκτελέσει τις διάφορες ενέργειες που απαιτούνται ο χρήστης. Για τον λόγο αυτό, κρίθηκε σκόπιμη η μετακίνηση της γραμμής εργαλείων στο κάτω μέρος της σελίδας και η ουσιαστική μετατροπή της σε υποσέλιδο στην προβολή μέσω κινητού.

Αυτό αποτελεί και ένα κοινό μοτίβο σχεδιασμού που εξυπηρετεί διάφορους σκοπούς και οφέλη όπως είναι:

- Βελτιωμένη εμπειρία χρήστη: Η τοποθέτηση της γραμμής εργαλείων στο κάτω μέρος της προβολής για κινητά καθιστά εύκολα προσβάσιμη την πλοήγηση με το ένα χέρι, κάτι που είναι πιο άνετο για τους χρήστες κινητών τηλεφώνων. Αυτό είναι ιδιαίτερα σημαντικό καθώς οι χρήστες συνήθως αλληλεπιδρούν με τις κινητές συσκευές χρησιμοποιώντας τους αντίχειρές τους.
- Ορατότητα περιεχομένου: Μετακινώντας τη γραμμή εργαλείων στο κάτω μέρος, εξασφαλίζεται ότι το κύριο περιεχόμενο της σελίδας είναι πιο ορατό και καταλαμβάνει το μεγαλύτερο μέρος της οθόνης. Αυτό είναι απαραίτητο για κινητές συσκευές με περιορισμένη επιφάνεια οθόνης.
- Συνέπεια: Η διατήρηση της συνέπειας στην τοποθέτηση των στοιχείων πλοήγησης βοηθά τους χρήστες να κατανοήσουν πώς να αλληλεπιδρούν με την εφαρμογή σε διαφορετικά μεγέθη οθόνης. Όταν οι χρήστες μεταβαίνουν από μια μεγαλύτερη οθόνη επιτραπέζιου υπολογιστή σε μια μικρότερη οθόνη κινητού τηλεφώνου, εξακολουθούν να βρίσκουν εύκολα τη γραμμή εργαλείων, αν και σε διαφορετική θέση.
- Φιλική προς την αφή: Η τοποθέτηση της γραμμής εργαλείων στο κάτω μέρος την καθιστά πιο φιλική προς την αφή, καθώς οι χρήστες μπορούν εύκολα να έχουν πρόσβαση σε αυτή χωρίς να τεντώνουν τα δάχτυλά τους ή να προσαρμόζουν τη λαβή τους στη συσκευή. Αυτό μπορεί να οδηγήσει σε μια πιο άνετη και αποτελεσματική εμπειρία χρήσης.
- Responsive Design: Πρόκειται για μια βασική πτυχή του responsive web design. Η προσαρμογή της διάταξης της γραμμής εργαλείων με βάση το μέγεθος της οθόνης διασφαλίζει ότι η εφαρμογή παραμένει εύχρηστη και οπτικά ελκυστική σε ένα ευρύ φάσμα συσκευών, από επιτραπέζιους υπολογιστές έως smartphones

4.4.3.3 Πρόσβαση στο μενού της εφαρμογής

Η προσθήκη μιας χειρονομίας σάρωσης για την αποκάλυψη του μενού είναι ένα συνηθισμένο και φιλικό προς το χρήστη πρότυπο αλληλεπίδρασης που μπορεί να βελτιώσει την εμπειρία του χρήστη κινητού στην εφαρμογή.

Ακολουθεί το τι επιτυγχάνεται με αυτή τη λειτουργία:

- Αποτελεσματική πλοήγηση: Τα μενού που αποκαλύπτονται με σάρωση παρέχουν στους χρήστες έναν αποτελεσματικό τρόπο πρόσβασης σε επιλογές πλοήγησης, ρυθμίσεις ή πρόσθετο περιεχόμενο χωρίς να γεμίζουν την κύρια οθόνη. Εξαλείφει την ανάγκη για ένα μόνιμο κουμπί ή εικονίδιο μενού στην οθόνη, κάτι που είναι ιδιαίτερα πολύτιμο σε μικρές οθόνες κινητών τηλεφώνων.
- Διατήρηση του χώρου της οθόνης: Κρύβοντας το μενού εκτός οθόνης μέχρι να το χρειαστεί ξανά ο χρήστης, αξιοποιείται καλύτερα ο περιορισμένος χώρο της οθόνης στις κινητές συσκευές. Οι χρήστες μπορούν να έχουν πρόσβαση στο μενού όταν το επιθυμούν και δεν θα καταλαμβάνει πολύτιμο χώρο όταν δεν το χρειάζονται.
- Αλληλεπίδραση φιλική προς το χρήστη: Το σύρσιμο είναι μια διαισθητική χειρονομία με την οποία οι χρήστες είναι εξοικειωμένοι από διάφορες εφαρμογές κινητών τηλεφώνων. Φαίνεται φυσική και μπορεί να βελτιώσει τη συνολική χρηστικότητα της εφαρμογής σας.
- Responsive Design: Το μενού swipe-to-reveal είναι ένα μοτίβο σχεδιασμού που ανταποκρίνεται και προσαρμόζεται σε διαφορετικά μεγέθη οθόνης. Λειτουργεί εξίσου καλά τόσο σε μικρές οθόνες κινητών τηλεφώνων όσο και σε μεγαλύτερες οθόνες tablet ή επιτραπέζιων υπολογιστών.
- Καθαρό και μινιμαλιστικό UI: Αυτό το μοτίβο αλληλεπίδρασης προωθεί μια πιο καθαρή και μινιμαλιστική διεπαφή χρήστη. Η διεπαφή της εφαρμογής σας παραμένει αμιγής, συμβάλλοντας σε έναν οπτικά ελκυστικό σχεδιασμό.

4.4.3.4 Συνδυασμός επιλογών στη γραμμή πλοήγησης

Ο συνδυασμός όλων των επιλογών από τη γραμμή πλοήγησης σε ένα μόνο κουμπί στην προβολή μέσω κινητού είναι ένα κοινό πρότυπο σχεδιασμού γνωστό ως "μενού hamburger" ή "πτυσσόμενο μενού".

Ακολουθεί το τι επιτυγχάνεται με αυτή την προσέγγιση:

- Αποδοτικότητα χώρου: Στις οθόνες κινητών τηλεφώνων, τα οριζόντια μενού πλοήγησης μπορεί να καταλαμβάνουν σημαντική ποσότητα χώρου. Με την ενοποίηση όλων των επιλογών του μενού σε ένα μόνο κουμπί, εξοικονομείται πολύτιμος χώρος στην οθόνη για το κύριο περιεχόμενο, διευκολύνοντας τους χρήστες να επικεντρωθούν στα βασικά χαρακτηριστικά της εφαρμογής.
- Μείωση της ακαταστασίας: Η απλοποίηση του μενού πλοήγησης για κινητά βοηθά στη μείωση της οπτικής ακαταστασίας και παρέχει μια πιο καθαρή και πιο οργανωμένη διεπαφή χρήστη. Οι χρήστες δεν θα κατακλύζονται από έναν μακρύ κατάλογο επιλογών μενού, ειδικά σε μικρότερες οθόνες.
- Βελτιωμένη εμπειρία χρήστη: Ένα καλά υλοποιημένο μενού “hamburger” μπορεί να βελτιώσει την εμπειρία του χρήστη κινητής συσκευής, καθιστώντας την πλοήγηση πιο προσιτή και διαισθητική. Όταν οι χρήστες πατούν το κουμπί μενού, μπορούν να έχουν πρόσβαση στις πλήρεις επιλογές του μενού χωρίς να πλοηγηθούν σε ξεχωριστή σελίδα.
- Responsive Design: Το μενού “hamburger” είναι ένα ευέλικτο μοτίβο σχεδιασμού που προσαρμόζεται σε διαφορετικά μεγέθη οθόνης, διασφαλίζοντας ότι η πλοήγηση στην εφαρμογή παραμένει λειτουργική και φιλική προς τον χρήστη σε διάφορες συσκευές, συμπεριλαμβανομένων των smartphones και των tablet.

- Συνέπεια: Διατηρεί τη συνέπεια στην εμπειρία πλοήγησης σε διάφορες συσκευές. Οι χρήστες που αλλάζουν από επιτραπέζιο σε κινητό θα βρουν το ίδιο σύνολο επιλογών, παρόλο που η διάταξη έχει αλλάξει.

4.4.3.5 Κάθετα προσανατολισμένα components

Σε όλη την εφαρμογή τα στοιχεία σχεδιάστηκαν, έτσι ώστε να είναι κάθετα προσανατολισμένα στην προβολή από κινητά, επιτυγχάνοντας πολλά οφέλη για την εμπειρία του χρήστη:

- Βελτιωμένη αναγνωσιμότητα: Οι κατακόρυφες διατάξεις είναι συνήθως ευκολότερες στην ανάγνωση και τη σάρωση σε φορητές συσκευές, επειδή οι χρήστες μπορούν να μετακινηθούν κατακόρυφα, πράγμα που είναι μια πιο φυσική και συνηθισμένη χειρονομία από την οριζόντια κύλιση. Αυτό βελτιώνει τη συνολική αναγνωσιμότητα του περιεχομένου σας.
- Βελτιστοποίηση για κύλιση: Οι χρήστες κινητών τηλεφώνων έχουν συνηθίσει να κάνουν κάθετη κύλιση για να περιηγηθούν στο περιεχόμενο. Οργανώνοντας τα στοιχεία κάθετα, ευθυγραμμίζεται η εφαρμογή με αυτή την προσδοκία των χρηστών και κάνουμε πιο διαισθητική την αλληλεπίδραση των χρηστών με την εφαρμογή μας.
- Συνεπής αλληλεπίδραση: Η συνέπεια στη διάταξη των στοιχείων σε διαφορετικές συσκευές (επιτραπέζιες και κινητές) είναι απαραίτητη για μια απρόσκοπτη εμπειρία χρήστη. Ο κάθετος προσανατολισμός εξασφαλίζει ότι το περιεχόμενο ρέει με συνέπεια από το ένα στοιχείο στο επόμενο, ανεξάρτητα από το μέγεθος της οθόνης.
- Αξιοποίηση του χώρου: Οι κάθετες διατάξεις κάνουν αποτελεσματική χρήση του διαθέσιμου χώρου της οθόνης στις κινητές συσκευές. Το περιεχόμενο μπορεί να εμφανίζεται με τρόπο που ελαχιστοποιεί την ανάγκη για οριζόντια κύλιση ή μεγέθυνση, διασφαλίζοντας ότι οι χρήστες μπορούν να έχουν πρόσβαση σε όλες τις πληροφορίες χωρίς απογοήτευση.
- Φιλική προς την αφή: Οι κατακόρυφες διατάξεις είναι πιο φιλικές προς την αφή, επειδή οι χρήστες μπορούν εύκολα να αγγίξουν και να αλληλεπιδράσουν με τα στοιχεία σε μία μόνο κατακόρυφη στήλη. Αυτό είναι ιδιαίτερα σημαντικό για τις κινητές συσκευές, όπου η είσοδος μέσω αφής είναι ο κύριος τρόπος αλληλεπίδρασης.
- Responsive Design: Ο κάθετος προσανατολισμός αποτελεί βασική πτυχή του responsive web design. Εξασφαλίζει ότι τα στοιχεία σας προσαρμόζονται με χάρη σε διάφορα μεγέθη και προσανατολισμούς οθόνης, από μικρές οθόνες smartphone έως μεγαλύτερες ταμπλέτες.

4.5 Βήματα μετατροπής react app σε ionic app

4.5.1 Δημιουργία του αρχείου capacitor.config.json

Σκοπός:

Το αρχείο capacitor.config.json χρησιμεύει ως το κύριο αρχείο ρυθμίσεων για το Capacitor, καθορίζοντας βασικές ρυθμίσεις για την εφαρμογή σας.

Λεπτομέρειες:

Βεβαιωθείτε ότι τα πεδία "appId" και "appName" είναι μοναδικά και έχουν το κατάλληλο όνομα για την εφαρμογή σας. Το "webDir" θα πρέπει να δείχνει στον κατάλογο όπου βρίσκονται τα αρχεία κατασκευής της εφαρμογής σας React.js. Για παράδειγμα, αν χρησιμοποιείτε τη λειτουργία Create React App, το "webDir" μπορεί να είναι το "build". Βεβαιωθείτε ότι έχετε ενημερώσει το πεδίο "name" ώστε να ταιριάζει με το όνομα της εφαρμογής σας.

```
{
  "appId": "io.ionic.nameofyourapp",
  "appName": "nameofyourapp",
  "bundledWebRuntime": false,
  "npmClient": "npm",
  "webDir": "build",
  "cordova": {}
}
```

4.5.2 Δημιουργία του αρχείο ionic.config.json

Σκοπός:

Το αρχείο ionic.config.json χρησιμοποιείται για να ρυθμίσει την εφαρμογή σε ένα Ionic framework και να καθορίσει τον τύπο της εφαρμογής που κατασκευάζεται.

Λεπτομέρειες:

Το πεδίο "name" θα πρέπει να αντικατοπτρίζει το όνομα της εφαρμογής. Το πεδίο "type" σε ανάθεση ως "react" για να δηλωθεί ότι χρησιμοποιείται η React.js. Η ενότητα "integrations" μπορεί να παραμείνει κενή όταν χρησιμοποιείται το Capacitor.

```
{
  "name": "nameofyourapp",
  "integrations": {
    "capacitor": {}
  },
  "type": "react"
}
```

4.5.3 Ενσωμάτωση του build σε React Project

Σκοπός:

Η κατασκευή της εφαρμογής React.js προετοιμάζει την εφαρμογή σας για ανάπτυξη και ενσωμάτωση με το Capacitor.

Λεπτομέρειες:

Πρέπει να γίνει χρήση της εντολής npm run build στον ριζικό κατάλογο του project για να δημιουργηθούν τα αρχεία κατασκευής. Αυτό το βήμα είναι ζωτικής σημασίας, καθώς το Capacitor θα χρησιμοποιήσει αυτά τα αρχεία για την εφαρμογή για κινητά. Το "webDir" που καθορίζεται στο capacitor.config.json θα πρέπει να ταιριάζει με τον κατάλογο εξόδου της κατασκευής.

```
npm run build
```

4.5.4 Εγκατάσταση το Ionic globally

Σκοπός:

Το Ionic CLI παρέχει βασικές εντολές για τη διαχείριση του έργου σας Ionic και Capacitor.

Λεπτομέρειες:

Για να εγκατασταθεί το Ionic CLI σε global επίπεδο, θα πρέπει να εκτελεστεί η ακόλουθη εντολή. Αυτό το βήμα διασφαλίζει ότι μπορείτε να έχετε πρόσβαση στις εντολές Ionic από οπουδήποτε στον υπολογιστή σας.

```
npm install -g @ionic/cli
```

4.5.5 Εγκατάσταση Capacitor Core

Σκοπός:

Το Capacitor Core είναι ένα θεμελιώδες πακέτο για την ενσωμάτωση του Capacitor στο project.

Λεπτομέρειες:

Εγκατάσταση το Capacitor Core ως εξάρτηση του project χρησιμοποιώντας την ακόλουθη εντολή. Αυτό το πακέτο επιτρέπει στο έργο σας να αλληλεπιδρά με τις λειτουργίες του Capacitor.

```
npm install @capacitor/core --save
```

4.5.6 Εγκατάσταση του Capacitor CLI

Σκοπός:

Το Capacitor CLI παρέχει βοηθητικά προγράμματα γραμμής εντολών για τη διαχείριση έργων και πλατφορμών Capacitor.

Λεπτομέρειες:

Προσθήκη του Capacitor CLI ως εξάρτηση ανάπτυξης με την ακόλουθη εντολή. Αυτό το CLI θα είναι απαραίτητο για την προσθήκη και τη διαχείριση πλατφορμών, πρόσθετων προγραμμάτων και άλλων.

```
npm i -D @capacitor/cli
```

4.5.7 Προσθήκη πλατφόρμας Android

Σκοπός:

Αυτό το βήμα ρυθμίζει την πλατφόρμα Android στο έργο σας Capacitor.

Λεπτομέρειες:

Πρέπει να εκτελεστεί η ακόλουθη εντολή για να προστεθεί η πλατφόρμα Android στο project. Αυτή η εντολή θα αρχικοποιήσει το έργο Android, θα εγκαταστήσει τις απαιτούμενες εξαρτήσεις και θα ρυθμίσει τις απαραίτητες διαμορφώσεις.

```
ionic capacitor add android
```

4.5.8 Άνοιγμα Android project στο Android Studio

Σκοπός:

Το άνοιγμα του έργου Android στο Android Studio επιτρέπει την εκτέλεση εργασιών στα εγγενή στοιχεία και τις διαμορφώσεις της εφαρμογής.

Λεπτομέρειες:

Για να ξεκινήσει το Android Studio με το project Capacitor γίνεται εκτελώντας την ακόλουθη εντολή. Εντός του Android Studio, ενδέχεται να χρειαστεί να ενημερωθεί το Gradle στην τελευταία έκδοση για να διασφαλίσετε τη συμβατότητα με το project.

```
npx cap open android
```

4.5.9 Δημιουργία του αρχείο APK

Σκοπός:

Η δημιουργία του αρχείου APK είναι το τελικό βήμα για τη δημιουργία της εφαρμογής Android.

Λεπτομέρειες:

Από το Android Studio, γίνεται μετάβαση στο μενού "Build" και γίνεται επιλογή του "Build Bundle(s) / APK(s)". Ακολουθώντας τις οδηγίες για τη διαμόρφωση των ρυθμίσεις της εφαρμογής, να επιλύονται οι εξαρτήσεις και δημιουργείται το αρχείο APK για διανομή ή δοκιμή.

5 Φάση Post - Migration

5.1 Η σημασία της Post - Migration Φασης

Η φάση μετά-μετανάστευσης είναι ένα κρίσιμο βήμα στη διαδικασία μετάβασης από μια διαδικτυακή εφαρμογή που βασίζεται στο React σε μια εφαρμογή για κινητά που βασίζεται στο Ionic Framework. Οι πρωταρχικοί της στόχοι είναι η διασφάλιση μιας απρόσκοπτης εμπειρίας χρήστη, η βελτιστοποίηση των επιδόσεων της εφαρμογής, η διασφάλιση της συμβατότητας πολλαπλών πλατφορμών και η διατήρηση ισχυρής ασφάλειας.

5.1.1 Βελτιστοποίηση επιδόσεων:

- Βελτίωση της ταχύτητας και της απόκρισης της εφαρμογής:
- Στη φάση μετά τη μετάβαση, ένας βασικός στόχος είναι η βελτίωση της ταχύτητας και της απόκρισης της εφαρμογής. Αυτό περιλαμβάνει τη βελτιστοποίηση του κώδικα, τη μείωση των περιττών κύκλων απόδοσης και τη χρήση εργαλείων παρακολούθησης επιδόσεων για τον εντοπισμό σημείων συμφόρησης. Οι προγραμματιστές θα πρέπει να επικεντρωθούν στην αποδοτική απόδοση και να ελαχιστοποιήσουν τη χρήση ‘βαριών’ λειτουργιών JavaScript που θα μπορούσαν να επιβραδύνουν τη διεπαφή της εφαρμογής.
- *Μείωση της χρήσης μνήμης*: Η μείωση της χρήσης της μνήμης είναι ζωτικής σημασίας για μια ομαλή εμπειρία κινητής τηλεφωνίας. Οι προγραμματιστές θα πρέπει να αναδιαμορφώνουν την εφαρμογή ώστε να χειρίζονται τη μνήμη πιο αποτελεσματικά, συμπεριλαμβανομένης της διαχείρισης των μεταβλητών, των πόρων και των κύκλων ζωής των συστατικών. Η χρήση της τεμπέλικης φόρτωσης (lazy loading) και των τεχνικών ανάκτησης δεδομένων με αποδοτική μνήμη μπορεί επίσης να συμβάλει σε ένα πιο λιτό αποτύπωμα μνήμης.

5.1.2 Βελτιώσεις της εμπειρίας χρήστη

- *Βελτιώσεις σχεδιασμού ειδικά για κινητά τηλέφωνα*: Για την παροχή άριστης εμπειρίας χρήστη σε φορητές συσκευές, ο σχεδιασμός της διεπαφής χρήστη (UI) και της εμπειρίας χρήστη (UX) θα πρέπει να είναι προσαρμοσμένος για μικρότερες οθόνες και αλληλεπιδράσεις αφής. Αυτό περιλαμβάνει τον επανασχεδιασμό των σελίδων - οθονών, τη βελτιστοποίηση των μεγεθών γραμματοσειράς και τη διασφάλιση στοιχείων φιλικών προς την αφή, όπως κουμπιά και χειριστήρια πλοήγησης.
- *Χειρονομίες αφής και πλοήγηση:* Το Ionic Framework παρέχει ενσωματωμένη υποστήριξη για χειρονομίες αφής και μοτίβα πλοήγησης ειδικά για εφαρμογές κινητών τηλεφώνων. Κατά τη φάση μετά τη μετάβαση, οι προγραμματιστές θα πρέπει να αξιοποιήσουν αυτά τα χαρακτηριστικά για να βελτιώσουν τη ροή πλοήγησης της εφαρμογής και να εφαρμόσουν κοινές χειρονομίες κινητών συσκευών, όπως σάρωση, πάτημα και pinch-to-zoom, για μια πιο αισθητική εμπειρία χρήστη.

5.1.3 Συμβατότητα πολλαπλών πλατφορμών

- *Εξασφάλιση της λειτουργικότητας της εφαρμογής σε διάφορες συσκευές*: Η συμβατότητα πολλαπλών πλατφορμών είναι απαραίτητη για να γίνει προσέγγιση σε ένα ευρύ κοινό. Οι προγραμματιστές θα πρέπει να δοκιμάζουν διεξοδικά την εφαρμογή σε διάφορες συσκευές, συμπεριλαμβανομένων διαφορετικών μαρκών και μοντέλων smartphones και tablets, καθώς και διαφορετικών λειτουργικών συστημάτων (iOS και Android). Οι δοκιμές συμβατότητας βοηθούν στον εντοπισμό και την επίλυση τυχόν προβλημάτων που αφορούν συγκεκριμένες πλατφόρμες.
- *Χειρισμός διαφορετικών μεγεθών και αναλύσεων οθόνης:* Το Ionic Framework προσφέρει αρχές responsive design που επιτρέπουν στους προγραμματιστές να προσαρμόζουν τη διάταξη και το περιεχόμενο της εφαρμογής σε διαφορετικά μεγέθη οθόνης και αναλύσεις. Κατά τη φάση μετά τη μετάβαση, οι προγραμματιστές θα πρέπει να αξιοποιήσουν τεχνικές responsive design και να χρησιμοποιήσουν queries μέσων CSS για να διασφαλίσουν ότι η εφαρμογή φαίνεται και λειτουργεί καλά σε διάφορες διαστάσεις οθόνης.

5.1.4 Ασφάλεια

- *Κρυπτογράφηση δεδομένων και ασφαλής επικοινωνία*: Η ασφάλεια είναι υψίστης σημασίας, ιδίως όταν πρόκειται για ευαίσθητα δεδομένα. Οι προγραμματιστές θα πρέπει να εφαρμόζουν μηχανισμούς κρυπτογράφησης δεδομένων, όπως το HTTPS για ασφαλή επικοινωνία, για την προστασία των δεδομένων που μεταδίδονται μεταξύ της εφαρμογής και των διακομιστών. Επιπλέον, θα πρέπει να ακολουθούν βέλτιστες πρακτικές για τον έλεγχο ταυτότητας και την εξουσιοδότηση, ώστε να διασφαλίζουν ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε ευαίσθητες πληροφορίες.
- *Ασφαλής αποθήκευση ευαίσθητων πληροφοριών:* Αφού γίνεται χρήση ευαίσθητων δεδομένων, η ασφαλής αποθήκευση τους είναι απαραίτητη. Οι προγραμματιστές θα πρέπει να χρησιμοποιούν τους μηχανισμούς ασφαλούς αποθήκευσης που παρέχει το Ionic Framework για την προστασία των ευαίσθητων πληροφοριών στη συσκευή. Αυτό περιλαμβάνει την κρυπτογράφηση της τοπικής αποθήκευσης, τη χρήση ασφαλών tokens και την εφαρμογή ασφαλών πρακτικών διαχείρισης κλειδιών για την αποτροπή παραβιάσεων ή διαρροών δεδομένων.

Συμπερασματικά, η φάση μετά τη μετάβαση από μια εφαρμογή React σε μια εφαρμογή Ionic Framework περιλαμβάνει τη βελτιστοποίηση της απόδοσης, τη βελτίωση της εμπειρίας χρήστη για κινητά, τη διασφάλιση της συμβατότητας μεταξύ πλατφορμών και την αντιμετώπιση των ζητημάτων ασφάλειας. Με την προσεκτική αντιμετώπιση αυτών των στόχων, οι προγραμματιστές μπορούν να εξασφαλίσουν μια επιτυχημένη μετάβαση και να παραδώσουν μια εφαρμογή υψηλής ποιότητας για κινητά που ανταποκρίνεται στις προσδοκίες των χρηστών και διατηρεί την ασφάλεια των δεδομένων.

5.2 Κοινά προβλήματα που σχετίζονται με τη μετάβαση

Η αντιμετώπιση κοινών προβλημάτων που σχετίζονται με τη μετάβαση είναι μια κρίσιμη πτυχή της επιτυχημένης μετάβασης από μια διαδικτυακή εφαρμογή React σε μια εφαρμογή Ionic για

κινητά. Σε αυτή την ενότητα, θα εμβαθύνουμε σε διάφορα κοινά ζητήματα που μπορεί να αντιμετωπίσουν οι προγραμματιστές κατά τη διαδικασία μετάβασης.

5.2.1 Προκλήσεις συμβατότητας

- *Πρόβλημα*: Το React και το Ionic έχουν διαφορετικούς κύκλους ζωής components και μηχανισμούς απόδοσης, γεγονός που οδηγεί σε πιθανές προκλήσεις συμβατότητας κατά τη μετάβαση στοιχείων του React στο Ionic.
- *Λύση*: Προσεκτική αναθεώρηση και αναδιαμόρφωση των στοιχείων React για να ευθυγραμμιστούν με τον κύκλο ζωής των στοιχείων του Ionic. Δίνουμε ιδιαίτερη προσοχή στις μεθόδους του κύκλου ζωής και στο χειρισμό συμβάντων, προσαρμόζοντάς τες στις συμβάσεις του Ionic.

5.2.2 Μετάβαση στη διαχείριση καταστάσεων

- *Πρόβλημα*: Η μετάβαση της διαχείρισης κατάστασης από το React (π.χ. Redux ή Mobx) στο Ionic απαιτεί προσεκτικό σχεδιασμό και προσαρμογή.
- *Λύση*: Εξετάζεται το ενδεχόμενο χρήσης των ενσωματωμένων λύσεων διαχείρισης καταστάσεων του Ionic ή βιβλιοθηκών για τη διαχείριση καταστάσεων στο Ionic. Επανασχεδιάζεται η λογική διαχείρισης καταστάσεων που βασίζεται στο Redux, στο Recoil ή στο Mobx ώστε να ταιριάζει στην αρχιτεκτονική του Ionic.

5.2.3 Styling και συνοχή του UI

- *Πρόβλημα*: Η διασφάλιση ότι η μεταφερόμενη εφαρμογή Ionic διατηρεί ένα συνεπές και οπτικά ελκυστικό UI μπορεί να είναι πρόκληση.
- *Λύση*: Αξιοποιείται το εκτεταμένο σύνολο στοιχείων UI του Ionic και τηρούμε τις οδηγίες σχεδιασμού του Ionic για να εξασφαλίσουμε μια συνεπή εμφάνιση και αίσθηση. Μεταφέρουμε τα στυλ CSS και τα προσαρμόζεται όπως απαιτείται για να ευθυγραμμιστούν με το στυλ του Ionic.

5.2.4 Δρομολόγηση και πλοήγηση

- *Πρόβλημα*: Το React και το Ionic έχουν διαφορετικά παραδείγματα δρομολόγησης και πλοήγησης, που απαιτούν προσαρμογές κατά τη διάρκεια της μετάβασης.
- *Λύση*: Επαναρυθμίζεται προσεκτικά τη λογική δρομολόγησης της εφαρμογής ώστε να ταιριάζει με το σύστημα πλοήγησης του Ionic.

5.2.5 Δοκιμές και αποσφαλμάτωση:

- *Πρόβλημα*: Οι αυστηρές δοκιμές και η αποσφαλμάτωση είναι απαραίτητες αλλά μπορεί να είναι πολύπλοκες, ειδικά σε ένα υβριδικό περιβάλλον.
- *Λύση*: Χρησιμοποιούμε ειδικά εργαλεία εντοπισμού σφαλμάτων Ionic, και διεξάγουμε εκτεταμένες δοκιμές σε πραγματικές κινητές συσκευές για τον άμεσο εντοπισμό και την επίλυση προβλημάτων. Εφαρμόζουμε δοκιμές μονάδας και δοκιμές από άκρο σε άκρο, όπως απαιτείται.

5.2.6 Δομή και οργάνωση κώδικα:

- *Πρόβλημα*: Η διατήρηση μιας καθαρής και οργανωμένης δομής κώδικα καθίσταται ζωτικής σημασίας για τη διαχείριση της πολυπλοκότητας της μετάβασης.
- *Λύση*: Εξασφαλίζουμε μια συνεπή και καλά οργανωμένη δομή έργου τόσο στις βάσεις κώδικα του React όσο και του Ionic. Τεκμηριώνουμε διεξοδικά τον κώδικα και τηρούμε τα πρότυπα κωδικοποίησης για να ενισχύσουμε την αναγνωσιμότητα και τη συντηρησιμότητα.

5.2.7 Βελτιστοποίηση επιδόσεων:

- *Πρόβλημα*: Οι εφαρμογές για κινητά απαιτούν ιδιαίτερη προσοχή στις επιδόσεις και η μετάβαση από το React στο Ionic μπορεί να εισάγει προβλήματα στις επιδόσεις.
- *Λύση*: Εφαρμόζονται στρατηγικές βελτιστοποίησης επιδόσεων, όπως lazy loading των ενοτήτων και η μείωση των περιττών επανεκτελέσεων, ώστε να διασφαλιστεί η βέλτιστη απόδοση της εφαρμογής στις κινητές συσκευές.

5.2.8 Χαρακτηριστικά ειδικά για πλατφόρμες:

- *Πρόβλημα*: Η ενσωμάτωση χαρακτηριστικών που αφορούν συγκεκριμένες πλατφόρμες, όπως ειδοποιήσεις push ή αισθητήρες συσκευών, στην εφαρμογή Ionic μπορεί να είναι πολύπλοκη.
- *Λύση*: Αξιοποιούνται τα εγγενή πρόσθετα του Ionic ή προσαρμοσμένα πρόσθετα του Cordova για πρόσβαση σε χαρακτηριστικά που αφορούν συγκεκριμένες πλατφόρμες. Εξασφαλίζονται έτσι τις κατάλληλες δοκιμές σε διαφορετικές πλατφόρμες κινητών τηλεφώνων για να επιβεβαιώσουμε τη λειτουργικότητα.

Η αντιμετώπιση αυτών των κοινών προβλημάτων που σχετίζονται με τη μετάβαση απαιτεί ένα συνδυασμό τεχνικής εμπειρογνωμοσύνης, προσεκτικού σχεδιασμού και ενδελεχούς δοκιμής. Οι προγραμματιστές θα πρέπει επίσης να ενημερώνονται για τις τελευταίες ενημερώσεις και τις βέλτιστες πρακτικές τόσο για το React όσο και για το Ionic, ώστε να διασφαλίζεται μια επιτυχημένη και απρόσκοπτη μετάβαση από τον ιστό στα κινητά. Με τον προληπτικό εντοπισμό και την αντιμετώπιση αυτών των προκλήσεων, οι προγραμματιστές μπορούν να μετριάσουν τα πιθανά εμπόδια και προβλήματα και να διασφαλίσουν μια ομαλότερη διαδικασία μετάβασης.

5.3 Testing και διασφάλιση ποιότητας (Quality Assurance QA)

5.3.1 Η σημασία του Testing

Οι δοκιμές και η διασφάλιση ποιότητας (QA) αποτελούν βασικές πτυχές κάθε διαδικασίας μετάβασης, είτε πρόκειται για τη μετάβαση λογισμικού, δεδομένων ή συστημάτων από ένα περιβάλλον σε άλλο. Οι πρακτικές αυτές είναι απαραίτητες για τη διασφάλιση της ομαλής και επιτυχημένης μετάβασης συστημάτων και εφαρμογών, την ελαχιστοποίηση των κινδύνων και τη διατήρηση προτύπων υψηλής ποιότητας.

Η σημασία των δοκιμών στη διαδικασία μετάβασης δεν μπορεί να υπερεκτιμηθεί. Χρησιμεύει ως μια ισχυρή στρατηγική μετριασμού των κινδύνων, βοηθώντας στον εντοπισμό και τον

μετριασμό πιθανών προβλημάτων σε πρώιμο στάδιο της διαδικασίας μετάβασης. Με τον εντοπισμό και την αντιμετώπιση αυτών των ζητημάτων πριν αυτά κλιμακωθούν, η δοκιμή μειώνει σημαντικά την πιθανότητα δαπανηρών αποτυχιών και διαταραχών. Επιπλέον, διαδραματίζει ζωτικό ρόλο στη διατήρηση της ακεραιότητας των δεδομένων κατά τη διάρκεια της μετάβασης, διασφαλίζοντας ότι τα δεδομένα παραμένουν ακριβή και πλήρη καθ' όλη τη διάρκεια της διαδικασίας. Αυτή η πτυχή είναι ιδιαίτερα κρίσιμη κατά τη μεταφορά ευαίσθητων ή κρίσιμων δεδομένων.

Η ικανοποίηση των χρηστών είναι ένα άλλο κρίσιμο στοιχείο για την επιτυχία της μετάβασης. Οι αυστηρές δοκιμές εγγυώνται ότι το μεταφερόμενο σύστημα ή η μεταφερόμενη εφαρμογή λειτουργεί όπως προβλέπεται, παρέχοντας μια αξιόπιστη και φιλική προς τον χρήστη εμπειρία. Οι προσδοκίες των χρηστών ικανοποιούνται και τα ζητήματα που θα μπορούσαν να οδηγήσουν σε απογοήτευση των χρηστών επιλύονται αμέσως, συμβάλλοντας στη θετική εμπειρία των χρηστών. Επιπλέον, σε κλάδους που υπόκεινται σε ειδικούς κανονισμούς, όπως η υγειονομική περίθαλψη ή τα οικονομικά, η συμμόρφωση με τους κανονισμούς αυτούς είναι υψίστης σημασίας. Οι δοκιμές διασφαλίζουν ότι η διαδικασία μετάβασης συμμορφώνεται με αυτές τις ειδικές για τον κλάδο απαιτήσεις, διασφαλίζοντας τους οργανισμούς από νομικές και οικονομικές συνέπειες.

Διάφοροι τύποι δοκιμών διαδραματίζουν διακριτούς ρόλους στη διαδικασία μετάβασης. Οι λειτουργικές δοκιμές επικεντρώνονται στην επαλήθευση ότι το μεταφερόμενο σύστημα ή η εφαρμογή εκτελεί με ακρίβεια τις προβλεπόμενες λειτουργίες του. Η δοκιμή ευχρηστίας αξιολογεί τη φιλικότητα προς τον χρήστη, διασφαλίζοντας ότι οι χρήστες μπορούν εύκολα να πλοηγηθούν και να αλληλεπιδράσουν με το σύστημα. Η δοκιμή συμβατότητας διασφαλίζει ότι το μεταφερόμενο σύστημα λειτουργεί με συνέπεια σε διάφορες πλατφόρμες, συμπεριλαμβανομένων διαφορετικών προγραμμάτων περιήγησης, συσκευών και λειτουργικών συστημάτων. Η δοκιμή επιδόσεων αξιολογεί την ταχύτητα, την απόκριση και την επεκτασιμότητα του συστήματος, διασφαλίζοντας ότι μπορεί να διαχειριστεί αποτελεσματικά τον αναμενόμενο φόρτο εργασίας. Τέλος, οι δοκιμές ασφαλείας εντοπίζουν τα τρωτά σημεία και τις αδυναμίες του μεταφερόμενου συστήματος, διασφαλίζοντας έτσι τις πιθανές απειλές στον κυβερνοχώρο.

5.3.2 Η σημασία της διασφάλισης ποιότητας

Η δοκιμή αξιοποιεί μια ποικιλία εργαλείων και μεθοδολογιών για την επίτευξη των στόχων της. Εργαλεία αυτοματοποιημένων δοκιμών, όπως το Selenium και το JUnit, αυτοματοποιούν επαναλαμβανόμενες εργασίες δοκιμών, ενισχύοντας την αποτελεσματικότητα και την κάλυψη. Οι χειροκίνητες δοκιμές από εξειδικευμένους επαγγελματίες QA συμπληρώνουν την αυτοματοποίηση, εντοπίζοντας ζητήματα που η αυτοματοποίηση μπορεί να παραβλέψει, ενώ ταυτόχρονα αξιολογεί την εμπειρία και τη χρηστικότητα του χρήστη. Οι δοκιμές παλινδρόμησης (regression tests) διασφαλίζουν ότι οι νέες αλλαγές που εισάγονται κατά τη διάρκεια της μετάβασης δεν επηρεάζουν αρνητικά τις υπάρχουσες λειτουργίες, με εργαλεία και σενάρια δοκιμών παλινδρόμησης που βελτιστοποιούν αυτή τη διαδικασία. Οι πρακτικές Συνεχούς Ολοκλήρωσης και Συνεχούς Δοκιμής CI/CD (Continuous Integration / Continuous Delivery) περιλαμβάνουν την εκτέλεση αυτοματοποιημένων δοκιμών σε αλλαγές κώδικα καθώς αυτές ενσωματώνονται, εντοπίζοντας προβλήματα νωρίς στον κύκλο ανάπτυξης.

Η αντιμετώπιση κοινών ζητημάτων που σχετίζονται με τη μετάβαση είναι ζωτικής σημασίας για μια επιτυχημένη μετάβαση. Η χαρτογράφηση και ο μετασχηματισμός δεδομένων διαδραματίζουν συχνά κεντρικό ρόλο, απαιτώντας σχολαστικές δοκιμές για τη διασφάλιση της

ακρίβειας των δεδομένων κατά τη διάρκεια της μετάβασης. Τα σχέδια διακοπής λειτουργίας και επαναφοράς είναι κρίσιμα ζητήματα για την ελαχιστοποίηση των διαταραχών και την παροχή ενός δικτύου ασφαλείας σε περίπτωση απρόβλεπτων προβλημάτων. Οι δοκιμές επικύρωσης και επαλήθευσης δεδομένων επιβεβαιώνουν την ακεραιότητα και τη συνέπεια των δεδομένων, ενώ η ρύθμιση των επιδόσεων εντοπίζει και βελτιστοποιεί τα σημεία συμφόρησης των επιδόσεων. Αυτά τα μέτρα συμβάλλουν συλλογικά σε μια ομαλότερη διαδικασία μετάβασης.

Η διασφάλιση μιας απρόσκοπτης εμπειρίας χρήστη είναι ο απώτερος στόχος. Οι δοκιμές αποδοχής χρηστών (UAT) επιτρέπουν στους τελικούς χρήστες να επικυρώσουν ότι το μεταφερόμενο σύστημα ανταποκρίνεται στις προσδοκίες και τις ανάγκες τους, παρέχοντας πολύτιμη ανατροφοδότηση. Η ολοκληρωμένη εκπαίδευση και η τεκμηρίωση των χρηστών διευκολύνουν την προσαρμογή των χρηστών στο μεταφερόμενο σύστημα, μειώνοντας την καμπύλη εκμάθησης. Η συνεχής παρακολούθηση μετά τη μετεγκατάσταση βοηθά στον άμεσο εντοπισμό και αντιμετώπιση προβλημάτων, ενώ οι δοκιμές επεκτασιμότητας διασφαλίζουν ότι το σύστημα μπορεί να αναπτυχθεί με τις μελλοντικές απαιτήσεις. Υιοθετώντας μια προσέγγιση με επίκεντρο τον χρήστη και εμπλέκοντας τους χρήστες στη διαδικασία δοκιμών και ανατροφοδότησης, οι οργανισμοί μπορούν να ευθυγραμμίσουν τη μετάβαση με τις προσδοκίες των χρηστών και να ενισχύσουν τη συνολική ικανοποίηση των χρηστών.

Εν κατακλείδι, οι δοκιμές και η διασφάλιση της ποιότητας είναι αναπόσπαστα στοιχεία για την επιτυχία των έργων μετάβασης. Παίζουν πολύπλευρο ρόλο στον μετριασμό των κινδύνων, την ακεραιότητα των δεδομένων, την ικανοποίηση των χρηστών και τη συμμόρφωση. Η χρήση των σωστών μεθοδολογιών, εργαλείων και πρακτικών δοκιμών είναι απαραίτητη για την επίτευξη μιας απρόσκοπτης και ευημερούσας μετάβασης, την προστασία από αποτυχίες και τη διασφάλιση μιας μετάβασης υψηλής ποιότητας.

5.3.2.1 Παραδείγματα testing

Για ακόμα μία φορά χρησιμοποιούνται ως παράδειγμα κώδικας ο οποίος αφορά την κατηγορία της βιβλιοθήκης (library).

Λειτουργική δοκιμή

Ακολουθεί ένα λεπτομερές παράδειγμα δοκιμής μονάδας για τη συνάρτηση handleDelete:

```
// Εισαγωγή των απαραίτητων εξαρτήσεων και της συνάρτησης handleDelete
import { handleDelete } from './yourCodeFile';

// Προσομοίωση των απαιτούμενων εξαρτήσεων όπως navigate και
toastSuccess
const navigate = jest.fn();
const toastSuccess = jest.fn();

// Ορισμός ενός περιγραφικού τέλεσης
describe('handleDelete', () => {
  it('διαγράφει το έγγραφο βιβλιοθήκης και εμφανίζει ένα μήνυμα
επιτυχίας', async () => {
    // Διάταξη: Διάταξη των δεδομένων δοκιμής και των απαιτούμενων
εξαρτήσεων
    const id = 123;
```

```
    // Ενέργεια: Κλήση της συνάρτησης με τα δεδομένα δοκιμής και τις
εξαρτήσεις
    await handleDelete(id, navigate, toastSuccess);

    // Επιβεβαίωση: Επιβεβαίωση της αναμενόμενης συμπεριφοράς
    expect(navigate).toHaveBeenCalledWith('/library'); // Έλεγχος εάν η
navigate κλήθηκε με το αναμενόμενο URL
    expect(toastSuccess).toHaveBeenCalledWith('Η ανάρτηση διαγράφηκε');
// Έλεγχος εάν η toastSuccess κλήθηκε με το μήνυμα επιτυχίας
  });

  it('χειρίζεται την αποτυχία διαγραφής και εμφανίζει ένα μήνυμα
σφάλματος', async () => {
    // Διάταξη: Διάταξη των δεδομένων δοκιμής και των απαιτούμενων
εξαρτήσεων
    const id = 456;

    // Προσομοίωση μιας περιπτώσεως σφάλματος όπου η διαγραφή
αποτυγχάνει
    // Μπορείτε να χρησιμοποιήσετε μια βιβλιοθήκη προσομοίωσης όπως
jest.mock ή sinon για την προσομοίωση της αποτυχίας

    // Ενέργεια: Κλήση της συνάρτησης με τα δεδομένα δοκιμής και τις
εξαρτήσεις
    await handleDelete(id, navigate, toastSuccess);

    // Επιβεβαίωση: Επιβεβαίωση της αναμενόμενης συμπεριφοράς
    expect(navigate).not.toHaveBeenCalled(); // Βεβαιωθείτε ότι η
navigate δεν κλήθηκε σε περίπτωση αποτυχίας
    // Έλεγχος εάν η toastError (που δεν φαίνεται στον αρχικό κώδικα)
κλήθηκε με το μήνυμα σφάλματος
  });
});
```

Δοκιμή ευχρηστίας:

Οι δοκιμές ολοκλήρωσης και ευχρηστίας μπορεί να περιλαμβάνουν τη δοκιμή του τρόπου με τον οποίο αλληλεπιδρούν διαφορετικά στοιχεία ή ενότητες μέσα στην εφαρμογή. Ακολουθεί ένα παράδειγμα:

```
import { render, fireEvent, screen } from '@testing-library/react';
import Create from './Create';

describe('Create Component', () => {
  it('submits a new library document when the form is filled', () => {
    // Εναρξη: Πραγματοποίηση απεικόνισης του στοιχείου Create με τα
απαραίτητα ψεύτικα δεδομένα

    // Δράση: Προσομοίωση αλληλεπίδρασης του χρήστη συμπληρώνοντας τη φόρμα
    fireEvent.change(screen.getByLabelText('Τίτλος'), { target: { value:
'Νέο Έγγραφο' } });
    fireEvent.change(screen.getByLabelText('Φάκελος'), { target: { value:
'Φάκελος Α' } });
    fireEvent.click(screen.getByText('Υποβολή'));

    // Επικύρωση: Έλεγχος αν το στοιχείο συμπεριφέρεται όπως αναμένεται
```

```
    // Για παράδειγμα, μπορείτε να επικυρώσετε αν εμφανίζεται ένα μήνυμα
επιτυχίας ή αν ο χρήστης ανακατευθύνεται σωστά
  });

  it('handles form validation and displays error messages', () => {
    // Έναρξη: Πραγματοποίηση απεικόνισης του στοιχείου Create

    // Δράση: Προσπάθεια υποβολής κενής φόρμας ή με μη έγκυρα δεδομένα

    // Επικύρωση: Έλεγχος αν το στοιχείο εμφανίζει κατάλληλα μηνύματα
σφάλματος και αποτρέπει την υποβολή
  });
});
```

Δοκιμές συμβατότητας (Compatibility Testing):

Για τη δοκιμή συμβατότητας, δοκιμάζεται χειροκίνητα η εφαρμογή σε διάφορα προγράμματα περιήγησης και συσκευές. Αυτό περιλαμβάνει την εκτέλεση της εφαρμογής σε διαφορετικές διαμορφώσεις και τον έλεγχο αυτών για τυχόν προβλήματα διάταξης ή λειτουργικότητας. Δεν εμπλέκεται συγκεκριμένος κώδικας σε αυτή τη διαδικασία, καθώς πρόκειται για μια προσπάθεια χειροκίνητης δοκιμής.

Δοκιμή επιδόσεων:

Η δοκιμή επιδόσεων γίνεται συνήθως με τη χρήση εξειδικευμένων εργαλείων όπως το Apache JMeter. Μέσω τέτοιων εργαλείων μπορεί να γίνει η προσομοίωση ενεργειών του χρήστη (π.χ. πρόσβαση σε σελίδες), να δημιουργούνται ομάδες νημάτων για την προσομοίωση ταυτόχρονων χρηστών αλλά και να συλλέγονται δεδομένα απόδοσης. Αυτό βοηθάει να εντοπίζονται τα σημεία συμφόρησης της απόδοσης της εφαρμογής.

Δοκιμές ασφαλείας:

Τα εργαλεία δοκιμών ασφαλείας όπως το OWASP ZAP μπορούν να βοηθήσουν στον εντοπισμό ευπαθειών στην εφαρμογή. Μέσω σάρωσης της εφαρμογής για τον εντοπισμό ευπαθειών ασφαλείας, μπορούν να παραχθούν αποτελέσματα, τα οποία μπορεί να περιλαμβάνουν ζητήματα όπως η έγχυση SQL ή οι ευπάθειες Cross-Site Scripting (XSS) και να καταδεικνύουν τρωτά σημεία της εφαρμογής .

5.3.3 Η σημασία του συνεπούς στυλ κωδικοποίησης σε εφαρμογές React

Το React έχει γίνει μια δημοφιλής επιλογή για τη δημιουργία δυναμικών εφαρμογών ιστού, προσφέροντας μια αρχιτεκτονική βασισμένη σε components που προωθεί την επαναχρησιμοποίηση και τη συντηρησιμότητα του κώδικα. Ωστόσο, για να διασφαλιστεί η επιτυχία ενός έργου React, η διατήρηση ενός συνεπούς στυλ κωδικοποίησης σε όλη την εφαρμογή είναι ζωτικής σημασίας. Παρακάτω διερευνώνται οι διάφοροι λόγοι για τους οποίους η συνέπεια στο στυλ κωδικοποίησης είναι απαραίτητη στην ανάπτυξη της React, τονίζοντας τον αντίκτυπό της στη συντηρησιμότητα, τη συνεργασία, την ποιότητα του κώδικα και τη συνολική επιτυχία του έργου.

Ένα από τα θεμελιώδη χαρακτηριστικά της React είναι η αρχιτεκτονική του που βασίζεται σε στοιχεία, η οποία επιτρέπει στους προγραμματιστές να δημιουργούν επαναχρησιμοποιήσιμα και αρθρωτά στοιχεία UI. Ενώ το React παρέχει ένα ισχυρό θεμέλιο για την ανάπτυξη εφαρμογών ιστών, η σημασία της τήρησης ενός συνεπούς στυλ κωδικοποίησης μέσα σε μια

εφαρμογή React είναι σημαντική. Οι λόγοι που οδηγούν για τη διατήρηση μιας τέτοιας συνέπειας και στις επιπτώσεις της στην ανάπτυξη έργων React περιγράφονται στις ακόλουθες παραγράφους.

5.3.4 Συντηρησιμότητα

Η συντήρηση μιας εφαρμογής React είναι μια συνεχής προσπάθεια που εκτείνεται σε ολόκληρο τον κύκλο ζωής της. Ένα συνεπές στυλ κωδικοποίησης βοηθά σημαντικά σε αυτή την προσπάθεια.

- *Αναγνωσιμότητα κώδικα*: Η συνέπεια στο στυλ κωδικοποίησης διασφαλίζει ότι όλα τα μέρη της βάσης κώδικα ακολουθούν τις ίδιες συμβάσεις. Αυτή η ομοιομορφία ενισχύει την αναγνωσιμότητα του κώδικα, διευκολύνοντας τους προγραμματιστές να κατανοούν, να περιηγούνται και να τροποποιούν τα διάφορα τμήματα της εφαρμογής. Οι προγραμματιστές ξοδεύουν λιγότερο χρόνο για την αποκρυπτογράφηση άγνωστων στυλ κώδικα, μειώνοντας τις πιθανότητες εισαγωγής σφαλμάτων ή λαθών κατά τη συντήρηση.
- *Παραγωγικότητα των προγραμματιστών*: Μια συνεπής βάση κώδικα προάγει την παραγωγικότητα των προγραμματιστών. Όταν οι προγραμματιστές συναντούν ένα οικείο και συνεπές στυλ κωδικοποίησης, μπορούν να κατανοήσουν γρήγορα τη δομή και τη λογική του κώδικα. Αυτή η αποδοτικότητα εξορθολογίζει τη διαδικασία ανάπτυξης και επιταχύνει τις διορθώσεις σφαλμάτων και τις προσθήκες χαρακτηριστικών, εξοικονομώντας τελικά χρόνο και πόρους.

5.3.5 Συνεργασία

Οι εφαρμογές React αναπτύσσονται συχνά από ομάδες προγραμματιστών που συνεργάζονται. Η συνέπεια στο στυλ κωδικοποίησης παίζει καθοριστικό ρόλο στην προώθηση της αποτελεσματικής συνεργασίας:

- *Επικοινωνία*: Ένα κοινό στυλ κωδικοποίησης χρησιμεύει ως μορφή επικοινωνίας εντός της ομάδας ανάπτυξης. Καθιερώνει μια κοινή κατανόηση του τρόπου με τον οποίο πρέπει να δομείται και να μορφοποιείται ο κώδικας. Οι προγραμματιστές μπορούν εύκολα να συζητούν θέματα που σχετίζονται με τον κώδικα, να ανταλλάσσουν ιδέες και να παρέχουν εποικοδομητική ανατροφοδότηση χωρίς να επιβαρύνονται από διαφορές στο στυλ.
- *Μειωμένες συγκρούσεις*: Όταν τα μέλη της ομάδας ακολουθούν τις ίδιες συμβάσεις κωδικοποίησης, οι συγκρούσεις και οι διαφωνίες σχετικά με το στυλ του κώδικα ελαχιστοποιούνται. Αυτό προάγει ένα αρμονικό περιβάλλον εργασίας, μειώνει την ένταση και ενθαρρύνει τη συνεργατική επίλυση προβλημάτων.

5.3.6 Ποιότητα κώδικα

Η διατήρηση ενός συνεπούς στυλ κωδικοποίησης συμβάλλει άμεσα στη συνολική ποιότητα της εφαρμογής React:

- *Κριτικές κώδικα*:΄Οι ανασκοπήσεις κώδικα αποτελούν αναπόσπαστο μέρος της διαδικασίας ανάπτυξης. Η συνέπεια στο στυλ κωδικοποίησης απλοποιεί τη διαδικασία αναθεώρησης. Οι επιθεωρητές μπορούν να επικεντρωθούν στην αξιολόγηση της λογικής, της λειτουργικότητας και των αρχιτεκτονικών επιλογών, αντί να σπαταλούν

χρόνο σε ασήμαντα ζητήματα που σχετίζονται με το στυλ. Αυτό έχει ως αποτέλεσμα πιο αποτελεσματικές και παραγωγικές αναθεωρήσεις κώδικα.

- *Αποσφαλμάτωση*: Μια συνεπής βάση κώδικα βοηθά στις προσπάθειες αποσφαλμάτωσης. Οι προγραμματιστές μπορούν να εντοπίζουν γρήγορα τις αποκλίσεις από το καθιερωμένο στυλ κατά την αντιμετώπιση προβλημάτων, διευκολύνοντας τον εντοπισμό των προβλημάτων και τη βελτίωση της ποιότητας του κώδικα. Αυτή η εξορθολογισμένη διαδικασία αποσφαλμάτωσης μειώνει τον χρόνο διακοπής λειτουργίας και ενισχύει την αξιοπιστία της εφαρμογής.

5.3.7 Επεκτασιμότητα

Καθώς οι εφαρμογές React αυξάνονται σε μέγεθος και πολυπλοκότητα, η διατήρηση ενός συνεπούς στυλ κωδικοποίησης γίνεται όλο και πιο κρίσιμη:

- *Τεχνικό χρέος*: Τα ασυνεπή στυλ κώδικα μπορεί να οδηγήσουν σε τεχνικό χρέος - μια κατάσταση όπου το κόστος συντήρησης και επέκτασης της βάσης κώδικα γίνεται δυσανάλογα υψηλό. Η διαχείριση και η κλιμάκωση της εφαρμογής καθίσταται δύσκολη, όταν τα στυλ κωδικοποίησης αποκλίνουν, γεγονός που ενδεχομένως εμποδίζει τις μελλοντικές προσπάθειες ανάπτυξης.
- *Ευκολία επέκτασης*: Ένα συνεπές στυλ κωδικοποίησης απλοποιεί τη διαδικασία επέκτασης της εφαρμογής με νέα χαρακτηριστικά ή στοιχεία. Οι προγραμματιστές μπορούν εύκολα να ενσωματώσουν νέο κώδικα στην υπάρχουσα βάση κώδικα χωρίς να ανησυχούν για συγκρούσεις στυλ ή προβλήματα συμβατότητας.

5.3.8 Ενσωμάτωση

Οι νέοι προγραμματιστές που εντάσσονται στην ομάδα επωφελούνται σημαντικά από ένα συνεπές στυλ κωδικοποίησης:

- *Μειωμένη καμπύλη μάθησης*: Όταν τα νέα μέλη της ομάδας συναντούν μια ομοιόμορφη βάση κώδικα, μπορούν να ενταχθούν ταχύτερα. Δεν χρειάζεται να ξοδέψουν υπερβολικό χρόνο για την εκμάθηση διαφορετικών στυλ κωδικοποίησης- αντίθετα, μπορούν να επικεντρωθούν στην κατανόηση της αρχιτεκτονικής και της επιχειρηματικής λογικής της εφαρμογής.

5.3.9 Αισθητική της βάσης κώδικα

Η συνέπεια στο στυλ κωδικοποίησης συμβάλλει στη συνολική αισθητική της βάσης κώδικα:

- *Επαγγελματισμός*: Μια καλά δομημένη και με συνέπεια στο στυλ κωδικοποιημένη βάση κώδικα αντανακλά επαγγελματισμό και δέσμευση στην ποιότητα του κώδικα. Στέλνει ένα θετικό μήνυμα στην ομάδα ανάπτυξης και στους ενδιαφερόμενους, αποδεικνύοντας ότι το έργο διαχειρίζεται με προσοχή και φροντίδα στη λεπτομέρεια.
- *Ηθική της ομάδας*: Μια καθαρή και οργανωμένη βάση κώδικα έχει θετικό αντίκτυπο στο ηθικό της ομάδας. Οι προγραμματιστές θεωρούν πιο ευχάριστο να εργάζονται και να συντηρούν έναν τέτοιο κώδικα, γεγονός που οδηγεί σε αυξημένη ικανοποίηση από την εργασία και κίνητρα.

5.3.10 Επαναχρησιμοποίηση

Το συνεπές στυλ κωδικοποίησης προωθεί τη δημιουργία επαναχρησιμοποιήσιμων στοιχείων και ενοτήτων μέσα σε μια εφαρμογή React:

- Αποδοτικότητα
  Οι προγραμματιστές μπορούν να δημιουργήσουν επαναχρησιμοποιήσιμα components πιο αποτελεσματικά όταν τηρούν ένα συνεπές στυλ κωδικοποίησης. Αυτή η αποδοτικότητα οδηγεί σε πιο οργανωμένες βάσεις κώδικα, μειώνει τον πλεονασμό και απλοποιεί τη συντήρηση των κοινών στοιχείων.

Συμπερασματικά, η διατήρηση ενός συνεπούς στυλ κωδικοποίησης σε μια εφαρμογή React δεν είναι απλώς θέμα αισθητικής, αλλά ένας κρίσιμος παράγοντας που επηρεάζει τη συνολική επιτυχία της. Η συνέπεια ενισχύει τη συντηρησιμότητα, διευκολύνει τη συνεργασία, βελτιώνει την ποιότητα του κώδικα και προετοιμάζει την εφαρμογή για επεκτασιμότητα. Εξορθολογίζει την εισαγωγή νέων προγραμματιστών, βελτιώνει την αισθητική της βάσης κώδικα και προωθεί τη δημιουργία επαναχρησιμοποιήσιμων στοιχείων. Για τη μεγιστοποίηση των πλεονεκτημάτων της ανάπτυξης React, είναι επιτακτική ανάγκη οι ομάδες ανάπτυξης να θέτουν ως προτεραιότητα και να επιβάλλουν τη συνέπεια του στυλ κωδικοποίησης καθ' όλη τη διάρκεια του κύκλου ζωής του έργου.

Αναγνωρίζοντας και υποστηρίζοντας τη σημασία ενός συνεπούς στυλ κωδικοποίησης, μπορεί να διασφαλιστεί η μακροπρόθεσμη επιτυχία και βιωσιμότητα των εφαρμογών σε ένα διαρκώς εξελισσόμενο τοπίο ανάπτυξης λογισμικού.

5.3.11 Η σημασία του συνεπούς στυλ στις εφαρμογές React

Παρακάτω διερευνάται η ύψιστη σημασία της διατήρησης μιας συνεπούς προσέγγισης styling εντός των εφαρμογών React. Διευκρινίζονται οι πολύπλευροι λόγοι για τους οποίους η συνεπής μορφοποίηση είναι απαραίτητη για την προώθηση μιας εξαιρετικής εμπειρίας χρήστη, την ενίσχυση της ταυτότητας της μάρκας, την ενίσχυση του επαγγελματισμού, τη διευκόλυνση της ευκολίας συντήρησης, την προώθηση της συνεργασίας, την προσαρμογή στην επεκτασιμότητα και τη διασφάλιση της προσβασιμότητας. Επιπλέον, εκθέτει τις στρατηγικές για την επίτευξη συνέπειας στη διαμόρφωση και διευκρινίζει τις εκτεταμένες επιπτώσεις της για τη συνολική επιτυχία των εφαρμογών React.

Το React, ως μια βιβλιοθήκη JavaScript, δίνει τη δυνατότητα στους προγραμματιστές να δημιουργούν δυναμικές και συναρπαστικές εφαρμογές ιστού. Ενώ η ευελιξία της React διευκολύνει την καινοτομία, ο οπτικός σχεδιασμός και η εμπειρία χρήστη είναι καθοριστικής σημασίας για την επιτυχία μιας εφαρμογής. Η συνέπεια στο στυλ παίζει καθοριστικό ρόλο στη δημιουργία μιας απρόσκοπτης και φιλικής προς το χρήστη διεπαφής. Οι λόγοι αυτοί είναι:

5.3.11.1 Εμπειρία χρήστη (UX)

Η συνέπεια στο styling παίζει καθοριστικό ρόλο στη διαμόρφωση μιας θετικής εμπειρίας χρήστη. Οι χρήστες αναμένουν ομοιομορφία στην εμφάνιση και τη συμπεριφορά κοινών στοιχείων, όπως κουμπιά, μενού πλοήγησης και είσοδοι φόρμας. Αυτή η προβλεψιμότητα εξορθολογίζει την αλληλεπίδραση των χρηστών, μετριάζει τη σύγχυση και ελαχιστοποιεί την απογοήτευση.

5.3.11.2 Επωνυμία

Η διασφάλιση μιας συνεπούς οπτικής ταυτότητας είναι αναπόσπαστο στοιχείο της αποτελεσματικής επωνυμίας. Μια εφαρμογή React που ακολουθεί έναν συνεκτικό οδηγό στυλ ενισχύει την ταυτότητα της μάρκας, καθιστώντας την πιο αναγνωρίσιμη και αξιομνημόνευτη. Η συνεπής χρήση των χρωμάτων, της τυπογραφίας και των οπτικών στοιχείων έχει απήχηση στο μήνυμα και τις αξίες της μάρκας.

5.3.11.3 Επαγγελματισμός

Μια καλοσχεδιασμένη και με συνέπεια στολισμένη εφαρμογή αποπνέει επαγγελματισμό και σχολαστικότητα. Οι χρήστες τείνουν να εμπιστεύονται περισσότερο και να βασίζονται σε εφαρμογές που φαίνονται αξιόπιστες και καλά συντηρημένες. Αυτό επηρεάζει θετικά τη φήμη και τη συνολική αντίληψη της εφαρμογής.

5.3.11.4 Ευκολία συντήρησης

Το συνεπές στυλ απλοποιεί τη διαδικασία ανάπτυξης και συντήρησης. Χρησιμοποιώντας μια βιβλιοθήκη επαναχρησιμοποιήσιμων στοιχείων και στυλ, οι προγραμματιστές μπορούν να κάνουν αλλαγές ή ενημερώσεις πιο αποτελεσματικά. Αυτό ελαχιστοποιεί τα σφάλματα και τις ασυνέπειες που μπορεί να προκύψουν κατά την τροποποίηση διαφορετικών στοιχείων σε ολόκληρη την εφαρμογή.

5.3.11.5 Συνεργασία

Σε συνεργατικά περιβάλλοντα ανάπτυξης, η τήρηση της συνεπούς μορφοποίησης είναι ζωτικής σημασίας. Όταν πολλοί προγραμματιστές εργάζονται σε διάφορα τμήματα της εφαρμογής, ένα κοινό σύστημα styling εξασφαλίζει ομοιομορφία στην τήρηση των κατευθυντήριων γραμμών. Αυτό προάγει την αποτελεσματικότητα και την αρμονία στη διαδικασία ανάπτυξης.

5.3.11.6 Επεκτασιμότητα

Καθώς οι εφαρμογές React επεκτείνονται σε πολυπλοκότητα και χαρακτηριστικά, η συνεπής μορφοποίηση γίνεται όλο και πιο κρίσιμη. Ένα ομοιόμορφο σύστημα σχεδίασης δίνει τη δυνατότητα στους προγραμματιστές να ενσωματώνουν απρόσκοπτα νέα χαρακτηριστικά ή τμήματα χωρίς να διακυβεύεται η εμπειρία του χρήστη.

5.3.11.7 Προσβασιμότητα

Η συνέπεια στη μορφοποίηση αυξάνει επίσης την προσβασιμότητα. Μια συνεπής δομή και συμπεριφορά των στοιχείων του UI διευκολύνει την πλοήγηση και την αλληλεπίδραση για τους χρήστες με αναπηρίες, ιδίως εκείνους που βασίζονται σε υποστηρικτικές τεχνολογίες. Η συμμετοχικότητα είναι απαραίτητη για την προσέγγιση ενός διαφορετικού κοινού.

5.3.11.8 Επίτευξη συνεπούς μορφοποίησης

Για την υλοποίηση συνεπούς μορφοποίησης σε εφαρμογές React, οι προγραμματιστές συχνά βασίζονται σε καθιερωμένα frameworks και βιβλιοθήκες όπως το Material-UI, το Bootstrap ή σε προσαρμοσμένα συστήματα σχεδιασμού. Αυτοί οι πόροι προσφέρουν προκαθορισμένα στοιχεία και στυλ που μπορούν να εφαρμοστούν συστηματικά, εξασφαλίζοντας μια συνεκτική και οπτικά ελκυστική διεπαφή χρήστη.

Συμπερασματικά, η ύψιστη σημασία της συνεπούς μορφοποίησης στις εφαρμογές React είναι αναμφισβήτητη. Επηρεάζει βαθιά την εμπειρία χρήστη, την ταυτότητα της μάρκας, τον επαγγελματισμό, την αποτελεσματικότητα της συντήρησης, τη συνεργατική ανάπτυξη, την επεκτασιμότητα και την προσβασιμότητα. Οι προγραμματιστές και οι σχεδιαστές πρέπει να δίνουν προτεραιότητα στη συνεπή μορφοποίηση για να δημιουργήσουν εφαρμογές React που θα ευδοκιμήσουν στο έντονα ανταγωνιστικό ψηφιακό τοπίο. Η τήρηση ενός ενιαίου συστήματος styling αποτελεί τον ακρογωνιαίο λίθο για την παροχή μιας συνεκτικής και αξιομνημόνευτης εμπειρίας χρήστη που δημιουργεί εμπιστοσύνη και δέσμευση.

5.4 Δημοσίευση και διανομή της εφαρμογής

Η δημοσίευση μιας εφαρμογής για κινητά είναι μια διαδικασία πολλών βημάτων που περιλαμβάνει τη διάθεση της εφαρμογής σας στους χρήστες σε δημοφιλείς πλατφόρμες διανομής εφαρμογών. Ενώ τα καταστήματα εφαρμογών είναι τα κύρια κανάλια για την προσέγγιση ενός ευρέος κοινού, είναι επίσης επωφελές να προσφέρουμε στους χρήστες τη δυνατότητα λήψης της εφαρμογής απευθείας από τον ιστότοπό μας. Παρακάτω περιγράφονται τα βασικά βήματα, οι βέλτιστες πρακτικές και οι εκτιμήσεις για μια επιτυχημένη κυκλοφορία της εφαρμογής.

5.4.1 Δημοσίευση σε καταστήματα εφαρμογών

1. *Προετοιμασία της εφαρμογής*: Πριν από τη δημοσίευση, βεβαιωνόμαστε ότι η Ionic εφαρμογή έχει ελεγχθεί διεξοδικά, έχει βελτιστοποιηθεί και συμμορφώνεται με τις οδηγίες και τις πολιτικές των αντίστοιχων καταστημάτων εφαρμογών.
2. *Δημιουργία Builds παραγωγής*: Χρησιμοποιούμε το Ionic CLI για να δημιουργούμε builds έτοιμα για παραγωγή τόσο για τις πλατφόρμες iOS όσο και για τις πλατφόρμες Android.

```
ionic build ios
ionic build android
```

3. *Διαμόρφωση για συγκεκριμένες πλατφόρμες:*
    - Για iOS: Διαμορφώνουμε το αρχείο `config.xml`, λαμβάνουμε πιστοποιητικά και προφίλ παροχής και δημιουργούμε έναν λογαριασμό Apple Developer.
    - Για Android: Διαμορφώνουμε το αρχείο `config.xml` και ‘υπογράφουμε’ την εφαρμογή Android με ένα keystore.
4. *Ρύθμιση λογαριασμών App Store*:
    - Δημιουργούμε έναν λογαριασμό Apple Developer και εγγραφόμαστε στο πρόγραμμα Apple Developer Program για iOS.
    - Δημιουργούμε έναν λογαριασμό προγραμματιστή στην κονσόλα Google Play για το Android.
5. *Προετοιμασία περιουσιακών στοιχείων και πληροφοριών της εφαρμογής*:
    - Δημιουργούμε εικονίδια εφαρμογών, στιγμιότυπα οθόνης και διαφημιστικό υλικό σύμφωνα με τις οδηγίες του καταστήματος εφαρμογών.
    - Κατασκευάζουμε μια κατάλληλη περιγραφή της εφαρμογής και παρέχουμε μεταδεδομένα, συμπεριλαμβανομένων λέξεων-κλειδιών και κατηγοριών.

6. *Υποβολή στο κατάστημα εφαρμογών:*
    - *Για το App Store της Apple*: Αρχειοθετούμε την εφαρμογή μας στο Xcode και την υποβάλουμε μέσω του ταμπλό App Store Connect.
    - *Για το Google Play Store*: Δημιουργούμε μια νέα καταχώρηση εφαρμογής στην κονσόλα του Google Play και υποβάλετε το APK.
7. *Επανεξέταση και έγκριση εφαρμογών:* Τόσο η Apple όσο και η Google θα επανεξετάσουν την εφαρμογή μας για να διασφαλίσουν ότι συμμορφώνεται με τις αντίστοιχες οδηγίες και πολιτικές τους.
8. *Αποδέσμευση της εφαρμογής:* Επιλογή πότε θα διαθέσουμε την εφαρμογή μας στο κοινό, είτε άμεσα είτε σε προγραμματισμένη ημερομηνία.
9. *Μάρκετινγκ και προώθηση*: Προωθούμε την εφαρμογή μας μέσω διαφόρων καναλιών μάρκετινγκ, συμπεριλαμβανομένων των μέσων κοινωνικής δικτύωσης, των ενημερωτικών δελτίων ηλεκτρονικού ταχυδρομείου, των ιστότοπων και της πληρωμένης διαφήμισης.
10. *Ενημερώσεις της εφαρμογής:* Εκδίδουμε περιοδικές ενημερώσεις για να βελτιώσουμε την εφαρμογή μας, να διορθώσουμε σφάλματα και να βελτιώσουμε την εμπειρία του χρήστη.
11. *Παρακολούθηση και ανταπόκριση*: Παρακολουθούμε τακτικά τα σχόλια και τις αξιολογήσεις των χρηστών και αντιμετωπίζουμε άμεσα τα ζητήματα και τις προτάσεις.
12. *Τήρηση των πολιτικών του App Store*: Ενημερωνόμαστε για τις αλλαγές πολιτικής και διασφαλίζουμε ότι η εφαρμογή μας παραμένει συμβατή.

5.4.2 Προσφορά άμεσων λήψεων από τον ιστότοπο

Εναλλακτικά προς τη διάθεση της εφαρογής μέσω των επίσημων καταστημάτων, ή συμπληρωματικά με αυτή, είναι δυνατόν η εφαρμογή να διατίθεται μέσω του ιστότοπου. Η διαδικασία της διάθεσης μέσω του ιστότοπου έχει ως ακολούθως:

1. *Δημιουργία σελίδας λήψης στον ιστότοπό μας:* Σχεδιάζουμε μια ειδική ιστοσελίδα στον ιστότοπό μας, όπου οι χρήστες μπορούν εύκολα να κατεβάζουν την εφαρμογή.
2. *Φιλοξενία αρχείων εφαρμογών*: Φιλοξενούμε τα αρχεία της εφαρμογής (APK για Android και IPA για iOS) στον διακομιστή του ιστότοπού μας.
3. *Ενεργοποίηση της εγκατάστασης της εφαρμογής στο Android*: Παρέχουμε οδηγίες στους χρήστες Android να ενεργοποιήσουν τις "Άγνωστες πηγές" στις ρυθμίσεις της συσκευής τους για να εγκαθιστούν εφαρμογές από άλλες πηγές εκτός από το Google Play Store.
4. *Σύνδεσμοι διανομής*: Δημιουργούμε απευθείας συνδέσμους λήψης στην ιστοσελίδα λήψης που παραπέμπουν στα φιλοξενούμενα αρχεία εφαρμογών (APK και IPA).
5. *Κώδικες QR*: Προαιρετικά, δημιουργούμε κωδικούς QR που παραπέμπουν στις διευθύνσεις URL λήψης της εφαρμογής για να διευκολύνουμε την πρόσβαση των χρηστών.
6. *Προώθηση των λήψεων από τον ιστότοπο*: Προωθούμε τη δυνατότητα λήψης της εφαρμογής από τον ιστότοπό μας μέσω υλικού μάρκετινγκ, μηνυμάτων ηλεκτρονικού ταχυδρομείου και μέσων κοινωνικής δικτύωσης.

7. *Ενημέρωση των λήψεων*: Βεβαιωνόμαστε ότι τα αρχεία της εφαρμογής που είναι διαθέσιμα στον ιστότοπό μας είναι πάντα ενημερωμένα με την τελευταία έκδοση της εφαρμογής.
8. *Παροχή υποστήριξης*: Προσφέρουμε υποστήριξη και καθοδήγηση στους χρήστες που επιλέγουν να κατεβάσουν την εφαρμογή από τον ιστότοπό μας, ειδικά αν αντιμετωπίζουν προβλήματα εγκατάστασης.

Η δημοσίευση και η διανομή μιας εφαρμογής Ionic είναι μια πολύπλευρη διαδικασία που περιλαμβάνει τόσο την υποβολή σε καταστήματα εφαρμογών όσο και την προσφορά άμεσων λήψεων από τον ιστότοπό μας. Ακολουθώντας τα βήματα που περιγράφονται παραπάνω, οι προγραμματιστές και οι δημιουργοί εφαρμογών μπορούν να εξασφαλίσουν μια επιτυχημένη κυκλοφορία της εφαρμογής και να προσεγγίσουν ένα ευρύτερο κοινό, λαμβάνοντας υπόψη τις προτιμήσεις και τους περιορισμούς των συσκευών των χρηστών.

6 Σελίδες - Οθόνες

Η συγκεκριμένη ενότητα αναφέρεται στη διαίρεση του Συστήματος Ηλεκτρονικής Διαχείρισης Εγγράφων "Ίριδα" σε διάφορες κατηγορίες, καθένα από τις οποίες αποτελεί μια ουσιώδη συνιστώσα του συστήματος. Αυτές οι κατηγορίες, που επισημαίνουν και καθορίζουν τη λειτουργικότητα του, αναφέρονται στα εξής: έγγραφα, εργασίες, σημαντικά θέματα, στοχοθεσία, βιβλιοθήκη, ημερολόγιο, εργαλεία, επαφές, διεκπεραιώσεις, προσαρμοσμένες λίστες, πρωτόκολλο, άδειες, αντικαταστάσεις και υποστήριξη. Για κάθε μία από αυτές τις κατηγορίες, παρουσιάζονται δύο πίνακες πληροφοριών, ένας που αφορά το frontend και ένας που αφορά το backend.

Στον πίνακα του frontend, καταγράφονται αναλυτικά όλες οι διαθέσιμες σελίδες για κάθε κατηγορία, συμπεριλαμβανομένης της URL διεύθυνσής τους και του κεντρικού component που χρησιμοποιείται για την προβολή και τη λειτουργία τους. Έχει προηγηθεί αναφορά στη διαδρομή των αντίστοιχων component σε προηγούμενη παράγραφο. Επιπλέον, υπάρχει μια στήλη που καθορίζει σε ποιο περιβάλλον (π.χ., Πολεμικής Αεροπορίας, Ελληνικού Δημοσίου ή γενικό) είναι διαθέσιμη η συγκεκριμένη σελίδα και οι λειτουργίες της. Η στήλη αυτή τιτλοφορείται στον πίνακα «ΠΑ / ΕΔ».

Σχετικά με τον δεύτερο πίνακα, που αφορά το backend, παρέχεται μια λεπτομερής λίστα με όλα τα endpoints που χρησιμοποιούνται από κάθε κατηγορία. Αυτά τα endpoints μπορεί να είναι είτε query, χρησιμοποιούμενα για την προβολή κάποιου στοιχείου, είτε command, ώστε να εκτελέσουν κάποια ενέργεια ή λειτουργία στο σύστημα.

Με αυτόν τον τρόπο, αυτή η ενότητα παρέχει έναν αναλυτικό οδηγό για τη διαχείριση των διαφορετικών κατηγοριών και τη χρήση και των δύο πλευρών, του frontend και του backend, του Συστήματος Ηλεκτρονικής Διαχείρισης Εγγράφων "Ίριδα".

6.1 Έγγραφα - Documents

6.1.1 Πίνακας Front-end:

Όνομα	Url	Component	ΠΑ / ΕΔ
Έγγραφα / Για ενέργεια	/documents/action	Action	
Έγγραφα / Εισερχόμενα έγγραφα αρχείου	/documents/archive/inbox	Archived	
Έγγραφα / Ενημερωτικά έγγραφα αρχείου	/documents/master	ArchivedMaster	
Έγγραφα / Κλωνοποίηση εγγράφου	/documents/clone/{docId}	Clone	
Έγγραφα / Κλωνοποίηση ενημερωτικού	/documents/clone/{docId}	CloneMaster	
Έγγραφα / Συνέχεια ενημέρωσης	/documents/continue-master/{docId}	ContinueMaster	
Έγγραφα / Ορθή επανάληψη εγγράφου	/documents/cp/{docId}	CorrectRepetition	
Έγγραφα / Ορθή επανάληψη εισαγωγής εγγράφου	/documents/cpimported/{docId}	CorrectRepetitionImported	
Έγγραφα / Νέο έγγραφο	/documents/create	Create	
Έγγραφα / Προσθήκη εγγράφου προς Υπογραφή	/documents/{masterDocId}/children/{docId}	CreateChild	
Έγγραφα / Νέο έγγραφο	/documents/create	CreateFromEndpoint	
Έγγραφα / Εισαγωγή εγγράφου	/documents/import/create	CreateImport	
Έγγραφα / Νέο ενημερωτικό	/documents/create-master-document	CreateMaster	
Έγγραφα / Υπογεγραμμένα έγγραφα αρχείου	/documents/distributed	Distributed	
Έγγραφα / Για διανομή	/documents/distribution	Distribution	
Έγγραφα / Εργασίες επί του εγγράφου	/documents/tasks/{docId}	DocumentTasks	
Έγγραφα / Νέα έκδοση εγγράφου ή Έγγραφα	/documents/edit/{docId}	Edit	

Όνομα	Url	Component	ΠΑ / ΕΔ
/ Επεξεργασία έκδοσης εγγράφου			
Έγγραφα / Νέα έκδοση εγγράφου ή Έγγραφα / Επεξεργασία έκδοσης εγγράφου	/documents/edit/{masterDocId}/children/{docId}	EditChild	
Έγγραφα / Επεξεργασία εγγράφου	/documents/import/edit/{docId}	EditImport	
Έγγραφα / Νέα έκδοση εγγράφου ή Έγγραφα / Επεξεργασία έκδοσης εγγράφου	/documents/edit/{docId}	EditMaster	
Έγγραφα / Εξωτερική Αλληλογραφία	/documents/external	External	
Έγγραφα / Εισαχθέντα έγγραφα αρχείου	/documents/archive/imported	Imported	
Έγγραφα / Για κοινοποίηση	/documents/info	Info	
Έγγραφα / Σε εξέλιξη	/documents/in-progress	InProgress	
Homepage	/documents	Landing	
Έγγραφα / Για υπογραφή	/documents/pending	Pending	
Έγγραφα / Νέα έκδοση εγγράφου ή Έγγραφα / Επεξεργασία έκδοσης εγγράφου	/documents/edit/registered/{docId}	RegisteredEdit	
Έγγραφα / Απορριφθέντα έγγραφα αρχείου	/documents/archive/rejected	Rejected	
Έγγραφα / Απορριφθέντα	/documents/rejected	Rejection	
Έγγραφα / Απάντηση σε έγγραφο	/documents/reply/{docId}	Reply	
Έγγραφα / Ετικέτα	/documents/tagged	Tagged	
Έγγραφα / Έγγραφα αρχείου από εργασίες	/documents/archive/tasked	Tasked	
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}	View	
Έγγραφα / Προβολή εγγράφου	/documents/view/{masterDocId}/children/{docId}	ViewChild	
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}	ViewMaster	

Όνομα	Url	Component	ΠΑ / ΕΔ
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}/versions/{version}	ViewMasterVersion	
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}/reference/{id}	ViewReference	
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}/reference/{id}	ViewReferenceMaster	
Έγγραφα / Προβολή εγγράφου	/documents/view/{docId}/versions/{version}	ViewVersion	

6.1.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Νέου Εγγράφου	POST	api/v2/documents	InsertDocumentCommand
Δημιουργία Νέας Έκδοσης Βάση Id Εγγράφου	POST	api/v2/documents/versions/{id}	CreateVersionCommand
Ενημέρωση Έκδοσης Βάση Id Εγγράφου και Id Έκδοσης	PUT	api/v2/documents/versions/{id}/{version}	UpdateVersionCommand
Ενημέρωση κοινοποίησης ΑΔΑ/ΦΕΚ	PUT	api/v2/documents/adaFek/{id}	UpdateDocumentAdaFekCommand
Υπογραφή Έκδοσης Βάση Id Εγγράφου και Id Έκδοσης	POST	api/v2/documents/versions/accept/{id}/{version}	AcceptVersionCommand
Απόρριψη Έκδοσης Βάση Id Εγγράφου και Id Έκδοσης	POST	api/v2/documents/versions/reject/{id}/{version}	RejectVersionCommand
Πρωτοκόλληση Εγγράφου βάσει Id	POST	api/v2/documents/register/{id}	RegisterDocumentCommand
Διανομή Εγγράφου βάσει Id	POST	api/v2/documents/distribute/{id}	DistributeDocumentCommand
Αρχειοθέτηση Εγγράφου βάσει Id	POST	api/v2/documents/archive/{id}	ArchiveDocumentCommand
Διαγραφή Εγγράφου	DELETE	api/v2/documents/{id}	DeleteDocumentCommand
Πολλαπλή Αρχειοθέτηση Εγγράφων	POST	api/v2/documents/archive	ArchiveDocumentsCommand

Περιγραφή	HTTPS	endpoint	Command/Query
Προώθηση εγγράφων σε άλλους αποδέκτες	POST	api/v2/documents/forward	ForwardDocumentsCommand
Μετατροπή Εγγράφου σε μη αναγνωσμένο	POST	api/v2/documents/unread/{id}	UnreadDocumentCommand
Επιστροφή εγγράφου στα εισερχόμενα	GET	api/v2/documents/unarchive/{id}	UnarchiveDocumentCommand
Δημιουργία Νέου Εγγράφου	GET	api/v2/documents/new	NewDocumentQuery
Προβολή Λίστας Εκδόσεων Του Εγγράφου	GET	api/v2/documents/{DocumentId}/versions	VersionQuery
Προβολή Έκδοσης Του Εγγράφου	GET	api/v2/documents/{DocumentId}/version/{Version}	ViewVersionQuery
Προβολή Σχετικού Εγγράφου	GET	api/v2/documents/{DocId}/reference/{RefId}	ViewReferenceDocQuery
Προβολή Λίστας Εγγράφων Για Ενέργεια	GET	api/v2/documents/inbox/action	InboxActionQuery
Προβολή Λίστας Εγγράφων Για Ενημέρωση	GET	api/v2/documents/inbox/info	InboxInfoQuery
Προβολή Λίστας Εγγράφων Εξωτερικής Αλληλογραφίας	GET	api/v2/documents/inbox/external	InboxExternalQuery
Προβολή Λίστας Εγγράφων Για Υπογραφή	GET	api/v2/documents/pending	PendingQuery
Προβολή Λίστας Εγγράφων Σε Εξέλιξη	GET	api/v2/documents/outbox/inprogress	InProgressQuery
Προβολή Λίστας Εγγράφων Για Διανομή	GET	api/v2/documents/outbox/distribution	ForDistributionQuery
Προβολή Λίστας Απορριφθέντων Εγγράφων	GET	api/v2/documents/outbox/rejected	RejectedQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Προβολή Λίστας Εισερχόμενων Αρχειοθετημένων Εγγράφων	GET	api/v2/documents/archive/inbox	ArchiveInboxQuery
Προβολή Λίστας Εξερχομένων Αρχειοθετημένων Εγγράφων	GET	api/v2/documents/archive/outbox	ArchiveOutboxQuery
Προβολή Λίστας Απορριφθέντων Αρχειοθετημένων Εγγράφων	GET	api/v2/documents/archive/rejected	ArchiveRejectedQuery
Προβολή Λίστας Αρχειοθετημένων Εγγράφων Προερχόμενα Από Εργασίες	GET	api/v2/documents/archive/tasked	ArchiveTaskedQuery
Προβολή Λίστας Εισαχθέντων Αρχειοθετημένων Εγγράφων	GET	api/v2/documents/archive/imported	ArchiveImportedQuery
Προβολή Λίστας Ενημερωτικών Αρχειοθετημένων Εγγράφων	GET	api/v2/documents/archive/master	ArchiveMasterQuery
Κλωνοποίηση Εγγράφου	GET	api/v2/documents/clone/{Id}	GetCloneDocumentQuery
Λίστα Σχετικών Εγγράφων	GET	api/v2/documents/references	ReferencesQuery
Απάντηση Εγγράφου	GET	api/v2/documents/reply/{Id}	GetReplyDocumentQuery
Ορθή Επανάληψη Εγγράφου	GET	api/v2/documents/correctRepetition/{Id}	GetCorrectRepetitionQuery
Λίστα αποδεκτών Εγγράφου	GET	api/v2/documents/{Id}/recipients/{EntryId}	GetDocumentRecipientsQuery
Εισαγωγή Νέου Εγγράφου	POST	api/v2/importDocuments	InsertImportedDocumentCommand
Επεξεργασία Εισαχθέντος Εγγράφου	PUT	api/v2/importDocuments/{id}	UpdateImportedDocumentCommand
Διανομή Εισαχθέντος Εγγράφου	POST	api/v2/importDocuments/distribute/{id}	DistributeImportedDocumentCommand
Αρχικοποίηση Εισαχθέντος Εγγράφου	GET	api/v2/importDocuments/new	NewImportedDocumentQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Αναζήτηση Εισαχθέντος Εγγράφου	GET	api/v2/importDocuments/{DocumentId}	ImportedVersionQuery
Ορθή Επανάληψη Εισαχθέντος Εγγράφου	GET	api/v2/importDocuments/correctRepetition/{id}	GetCorrectRepetitionImportedDocumentQuery
Αποθήκευση Νέου Ενημερωτικού Εγγράφου	POST	api/v2/master-documents	InsertMasterDocumentCommand
Δημιουργία Νέας Έκδοσης Ενημερωτικού Εγγράφου	POST	api/v2/master-documents/versions/{id}	CreateMasterVersionCommand
Ενημέρωση Ενημερωτικού Εγγράφου	PUT	api/v2/master-documents/versions/{id}/{version}	UpdateMasterVersionCommand
Διαγραφή Ενημ. Εγγράφου	DELETE	api/v2/master-documents/{id}	DeleteMasterDocument
Έγκριση Ενημερωτικού Εγγράφου	POST	api/v2/master-documents/versions/accept/{id}/{version}	AcceptMasterVersionCommand
Απόρριψη Ενημερωτικού Εγγράφου	POST	api/v2/master-documents/versions/reject/{id}/{version}	RejectMasterVersionCommand
Δημιουργία Εγγράφου Παιδιού	POST	api/v2/master-documents/children/{id}	InsertChildDocumentCommand
Ενημέρωση Εγγράφου Παιδιού	PUT	api/v2/master-documents/versions/children/{id}/{version}	UpdateChildVersionCommand
Δημιουργία Νέας Έκδοσης Εγγράφου Παιδιού	POST	api/v2/master-documents/versions/children/{masterId}/{id}	CreateChildVersionCommand
Διαγραφή Εγγράφου Παιδιού	DELETE	api/v2/master-documents/{masterId}/children/{id}	DeleteChildDocumentCommand
Αρχικοποίηση νέου Ενημερωτικού Εγγράφου	GET	api/v2/master-documents/new	NewMasterDocumentQuery
Προβολή Ενημερωτικού Εγγράφου	GET	api/v2/master-documents/{id}	ViewMasterQuery
Προβολή Σχετικού Ενημερωτικού Εγγράφου	GET	api/v2/master-documents/{DocId}/reference/{RefId}	ViewReferenceMasterQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Επεξεργασία Ενημερωτικού Εγγράφου	GET	api/v2/master-documents/edit/{id}	MasterVersionQuery
Προβολή Προηγούμενης Έκδοσης Ενημερωτικού	GET	api/v2/master-documents/{Id}/version/{Version}	ViewMasterVersionQuery
Διασύνδεση Ενημερωτικού με Έγγραφα Παιδιά	GET	api/v2/master-documents/{Id}/children/new	NewChildDocumentQuery
Προβολή Εγγράφου Παιδιού Ενημερωτικού	GET	api/v2/master-documents/{masterId}/children/{Id}	ViewChildQuery
Επεξεργασία Εγγράφου Παιδιού Ενημερωτικού	GET	api/v2/master-documents/{masterId}/children/{Id}/edit	ChildVersionQuery
Συνέχεια Ενημέρωσης	GET	api/v2/master-documents/continue/{Id}	ContinueMasterQuery
Κλωνοποίηση ενημερωτικού εγγράφου	GET	api/v2/master-documents/clone/{Id}	GetCloneMasterQuery

6.2 Εργασίες - Tasks

6.2.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Εργασίες / Δημιουργία εργασίας	/tasks/create	Create	
Εργασίες / Δημιουργία εξαρτώμενης εργασίας	/tasks/create/{taskID}	CreateChild	
Εργασίες / Δημιουργία εργασίας από έγγραφο	/tasks/create/document/{docId}	CreateFromDoc	
Εργασίες / Τίτλος εργασίας	/tasks/edit/{taskId}	Edit	
Εργασίες / Εισερχόμενες	/tasks/inbox	InboxAll	
Εργασίες / Εισερχόμενες εκκρεμότητες	/tasks/inbox-pending/{taskType}	InboxPending	

Όνομα	Url	Component	ΠΑ / ΕΔ
Εργασίες / Εισερχόμενες ολοκληρωμένες	/tasks/inbox-complete{taskType}	InboxComplete	
Homepage	/tasks	Landing	
Εργασίες / Εξερχόμενες	/tasks/outbox	OutboxAll	
Εργασίες / Εξερχόμενες εκκρεμότητες	/tasks/outbox-pending/{taskType}	OutboxPending	
Εργασίες / Εξερχόμενες ολοκληρωμένες	/tasks/outbox-complete{taskType}	OutboxComplete	
Εργασίες / Προβολή εργασίας	/tasks/view/{taskId}	View	

6.2.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Εργασίας	POST	api/v2/tasks	CreateTaskCommand
Δημιουργία Εργασίας Τύπου Ενημέρωσης	POST	api/v2/tasks/info	CreateTasksCommand
Δημιουργία Εργασίας Τύπου Ενέργειας	POST	api/v2/tasks/action	CreateTasksCommand
Ενημέρωση Εργασίας βάσει id	PUT	api/v2/tasks/{id}	UpdateTaskCommand
Αλλαγή Κατάστασης Εργασίας σε Ολοκληρωμένη	PUT	api/v2/tasks/finish/{id}	FinishTaskCommand
Αλλαγή Κατάστασης Λίστας Εργασιών σε Ολοκληρωμένες	PUT	api/v2/tasks/finishGroup	FinishTasksCommand
Διαγραφή Εργασίας	DELETE	api/v2/tasks/{id}	DeleteTaskCommand
Προβολή εργασίας	GET	api/v2/tasks/view/{Id}	GetViewTaskQuery
Αναζήτηση εργασίας βάσει Id	GET	api/v2/tasks/{Id}	GetTaskByIdQuery
Όλες οι εισερχόμενες εργασίες	GET	api/v2/tasks/inbox/all	InboxAllQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Λίστα Εισερχόμενων Εργασιών σε εκκρεμότητα	GET	api/v2/tasks/inbox/pending/{taskType}	InboxPendingQuery
Λίστα Εισερχόμενων Ολοκληρωμένων Εργασιών	GET	api/v2/tasks/inbox/completed/{taskType}	InboxCompletedQuery
Λίστα Εισερχόμενων Εργασιών για Ενημέρωση	GET	api/v2/tasks/inbox/delegates	InboxDelegatesQuery
Όλες οι εξερχόμενες εργασίες	GET	api/v2/tasks/outbox/all	OutboxAllQuery
Λίστα Εξερχόμενων Ολοκληρωμένων Εργασιών	GET	api/v2/tasks/outbox/completed/{taskType}	OutboxCompletedQuery
Λίστα Εξερχόμενων Εργασιών σε εκκρεμότητα	GET	api/v2/tasks/outbox/pending/{taskType}	OutboxPendingQuery
Λίστα Εξερχόμενων Εργασιών για Ενημέρωση	GET	api/v2/tasks/outbox/delegates	OutboxDelegatesQuery
Λίστα εργασιών βάσει Id εγγράφου	GET	api/v2/tasks/document/{docId}	DocumentTasksQuery
Λίστα σχετικών εργασιών	GET	api/v2/tasks/references	ReferenceTasksQuery
Αρχικοποίηση Νέας Εργασίας	GET	api/v2/tasks/new	GetNewTaskQuery
Αρχικοποίηση Νέας Υποεργασίας	GET	api/v2/tasks/new/task/{Id}	GetChildTaskQuery
Διασύνδεση Εργασίας με Έγγραφο	GET	api/v2/tasks/new/document/{DocId}	GetNewTaskFromDocumentQuery
Προβολή Ημερολογίου	GET	api/v2/tasks/calendar/{year}/{month}	CalendarTaskQuery

6.3 Σημαντικά θέματα - Issues

6.3.1 Πίνακας Front-end:

Όνομα	Url	Component	ΠΑ / ΕΔ
Σημαντικά θέματα / Νέο σημαντικό θέμα	/issues/create	Create	
Σημαντικά θέματα / Νέο σημαντικό θέμα από έγγραφο	/issues/create/document/{docId}	CreateFromDoc	
Σημαντικά θέματα / Νέο σημαντικό θέμα από εργασία	/issues/create/task/{taskId}	CreateFromTask	
Σημαντικά θέματα / Τίτλος σημαντικού θέματος	/issues/edit/{issueId}}	Edit	
Εργασίες / Σημαντικά θέματα	/issues/	IssuesAll	
Σημαντικά θέματα / Προβολή	/issues/view/{issueId}	View	

6.3.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Σημαντικού Θέματος	POST	api/v2/significantIssues	CreateSignificantIssueCommand
Ενημέρωση Σημαντικού Θέματος	PUT	api/v2/significantIssues/{id}	UpdateSignificantIssueCommand
Διασύνδεση Εγγράφου και Σημαντικού Θέματος	GET	api/v2/significantIssues/{id}/add/document/{docId}	AddDocumentReferenceToIssueCommand
Διασύνδεση Εργασίας και Σημαντικού Θέματος	GET	api/v2/significantIssues/{id}/add/task/{taskId}	AddTaskReferenceToIssueCommand
Διαγραφή Σημαντικού Θέματος	DELETE	api/v2/significantIssues/{id}	DeleteSignificantIssueCommand
Προβολή Σημαντικών Θεμάτων	GET	api/v2/significantIssues	GetSignificantIssuesQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Αρχικοποίηση Νέου Σημαντικού Θέματος	GET	api/v2/significantIssues/new	GetNewSignificantIssueQuery
Δημιουργία Νέου Σημαντικού Θέματος από Έγγραφο	GET	api/v2/significantIssues/new/document/{DocId}	GetSignificantIssueFromDocumentQuery
Δημιουργία Νέου Σημαντικού Θέματος από Εργασία	GET	api/v2/significantIssues/new/task/{TaskId}	GetSignificantIssueFromTaskQuery
Αναζήτηση Σημαντικού Θέματος βάση id	GET	api/v2/significantIssues/{id}	GetSignificantIssueByIdQuery
Προβολή Σημαντικού Θέματος βάσει id	GET	api/v2/significantIssues/view/{id}	GetViewSignificantIssueByIdQuery
Λίστα Σημαντικών Θεμάτων για επιλογή	GET	api/v2/significantIssues/picker	SignificantIssuesPickerQuery

6.4 Στοχοθεσία / Ωρίων - Goals

6.4.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Στοχοθεσία ή Ωρίων / Νέος στόχος	/goals/create	Create	
Στοχοθεσία ή Ωρίων / Τίτλος στόχου	/goals/edit/{issueId}}	Edit	
Εργασίες / Στοχοθεσία ή Ωρίων	/goals/	IssuesAll	
Στοχοθεσία ή Ωρίων / Προβολή	/goals/view/{issueId}	View	

6.4.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Αποθήκευση Νέου Στόχου	POST	api/v2/goals	CreateGoalCommand
Διασύνδεση Στόχου με Έγγραφο Βάσει των ids	GET	api/v2/goals/{id}/add/document/{docId}	AddDocumentReferenceToGoalCommand
Διασύνδεση Στόχου με Εργασία Βάσει των ids	GET	api/v2/goals/{id}/add/task/{taskId}	AddTaskReferenceToGoalCommand
Ενημέρωση Στόχου βάσει id	PUT	api/v2/goals/{id}	UpdateGoalCommand
Διαγραφή Στόχου Βάσει του id	DELETE	api/v2/goals/{id}	DeleteGoalCommand
Προβολή όλων των στόχων	GET	api/v2/goals	GetGoalsQuery
Δημιουργία Νέου Στόχου	GET	api/v2/goals/new	GetNewGoalQuery
Αναζήτηση Στόχου βάσει του id	GET	api/v2/goals/{id}	GetGoalByIdQuery
Προβολή Στόχου Βάσει Id	GET	api/v2/goals/view/{id}	GetGoalViewQuery
Λίστα Στόχων για επιλογή	GET	api/v2/goals/view/picker	GetGoalsForPickerQuery

6.5 Βιβλιοθήκη - Library

6.5.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Βιβλιοθήκη / Δημιουργία ανάρτησης	/library/create	Create	
Βιβλιοθήκη / Επεξεργασία ανάρτησης	/library/edit/{libId}}	Edit	
Εργαλεία / Βιβλιοθήκη	/library/	Library	
Βιβλιοθήκη / Προβολή εγγράφου βιβλιοθήκης	/library/edit/{libId}}	View	

6.5.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Αποθήκευση Νέου Εγγράφου Βιβλιοθήκης	POST	api/v2/library	CreateLibraryDocumentCommand
Ενημέρωση Εγγράφου Βιβλιοθήκης	PUT	api/v2/library/{id}	UpdateLibraryDocumentCommand
Διαγραφή Εγγράφου Βιβλιοθήκης	DELETE	api/v2/library/{id}	DeleteLibraryDocumentCommand
Προβολή Εγγράφων σε Βιβλιοθήκη	GET	api/v2/library	GetLibraryDocumentsQuery
Προβολή Εγγράφου σε Βιβλιοθήκη βάσει id	GET	api/v2/library/view/{Id}	GetViewLibraryDocumentByIdQuery
Αναζήτηση Εγγράφου σε Βιβλιοθήκη βάσει id	GET	api/v2/library/{Id}	GetLibraryDocumentByIdQuery
Αρχικοποίηση Νέου Εγγράφου σε Βιβλιοθήκη	GET	api/v2/library/new	GetNewLibraryDocumentQuery

6.6 Ημερολόγιο - Calendar

6.6.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Εργαλεία / Ημερολόγιο	/calendar	Calendar	

6.7 Εργαλεία - Tools

6.7.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Homepage	/tools	Landing	

6.8 Επαφές - Contacts

6.8.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Εργαλεία / Επαφές εκτός Ίριδα	/contacts/external	ContactsExternal	
Εργαλεία / Επαφές προσωπικού Φορέα	/contacts/haf	ContactsHaf	
Εργαλεία / Φορείς Ίριδα	/contacts/roots	ContactsRoots	
Εργαλεία / Όνομα Φορέα - Οργανόγραμμα	/contacts/roots/{rootId}	ContactsRootsOrg	
Επαφές / Δημιουργία επαφής εκτός Ίριδα	/contacts/create	Create	
Επαφές / Επεξεργασία επαφής εκτός Ίριδα	/contacts/external/edit/{contId}	EditExternalContact	
Επαφές / Τίτλος επαφής	/contacts/haf/view/{contId}	ViewContactHaf	
Επαφές / Τίτλος επαφής	/contacts/external/view/{contId}	ViewExternalContact	

6.8.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Νέας Επαφής Εκτός Ίριδα	POST	api/v2/contacts/external	CreateAssociateCommand
Ενημέρωση Νέας Επαφής Εκτός Ίριδα	PUT	api/v2/contacts/external/{id}	UpdateAssociateCommand
Διαγραφή Νέας Επαφής Εκτός Ίριδα	DELETE	api/v2/contacts/external/{id}	DeleteAssociateCommand
Λίστα φορέων ίριδα	GET	api/v2/contacts/roots	RootsContactsQuery
Λίστα φορέων ίριδα	GET	api/v2/contacts/roots/{id}	RootsContactsByIdQuery
Λίστα Χρηστών Ίριδας Εντός Φορέα	GET	api/v2/contacts/root	RootContactsQuery
Χρήστης Ίριδας Εντός Φορέα	GET	api/v2/contacts/root/{id}	RootContactByIdQuery
Λίστα Χρηστών Ίριδας Εκτός Ίριδα	GET	api/v2/contacts/external	ExternalContactsQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Χρήστης Ίριδας Εκτός Ίριδα	GET	api/v2/contacts/external/{id}	ExternalContactByIdQuery

6.9 Διεκπεραιώσεις - Coordinations

6.9.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Διεκπεραιώσεις / Νέα διεκπεραίωση	/coordinations/create	Create	
Διεκπεραιώσεις / Επεξεργασία διεκπεραίωσης	/coordinations/edit/{coorId}	Edit	
Εργαλεία / Ανατιθέμενες διεκπεραιώσεις	/coordinations/provided	Provided	

6.9.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Διεκπεραίωσης	POST	api/v2/coordinations	CreateCoordinationCommand
Ενημέρωση Διεκπεραίωσης	PUT	api/v2/coordinations/{id}	UpdateCoordinationCommand
Διαγραφή Διεκπεραίωσης	DELETE	api/v2/coordinations/{id}	DeleteCoordinationCommand
Διεκπεραιώσεις	GET	api/v2/coordinations/provided	GetProvidedCoordinationsQuery
Αναζήτηση διεκπεραίωσης	GET	api/v2/coordinations/{id}	GetCoordinationByIdQuery

6.10 Προσαρμοσμένες λίστες - CustomLists

6.10.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Λίστες υπογραφών / Νέα λίστα	/customlists/approvals/create	CreateCustomApprovalTable	
Λίστες αποδεκτών / Νέα λίστα	/customlists/distribution/create	CreateCustomDistributionTable	
Λίστες εργασιών / Νέα λίστα	/customlists/recipients/create	CreateCustomRecipientTable	
Εργαλεία / Λίστες υπογραφών	/customlists/approvals	CustomApprovalTables	
Εργαλεία / Λίστες αποδεκτών	/customlists/distribution	CustomDistributionTables	
Εργαλεία / Λίστες εργασιών	/customlists/recipients	CustomRecipientTables	
Λίστες υπογραφών / Τίτλος λίστας	/customlists/approvals/edit/{listId}	EditCustomApprovalTable	
Λίστες αποδεκτών / Τίτλος λίστας	/customlists/distribution/edit/{listId}	EditCustomDistributionTable	
Λίστες εργασιών / Τίτλος λίστας	/customlists/recipients/edit/{listId}	EditCustomRecipientTable	
Λίστες υπογραφών / Τίτλος λίστας	/customlists/approvals/view/{listId}	ViewCustomApprovalTable	
Λίστες αποδεκτών / Τίτλος λίστας	/customlists/distribution/view/{listId}	ViewCustomDistributionTable	
Λίστες εργασιών / Τίτλος λίστας	/customlists/recipients/view/{listId}	ViewCustomRecipientTable	

Πίνακας Back-end:

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία λίστας υπογραφών	POST	api/v2/positions/customApproval	CreateCustomApprovalCommand
Ενημέρωση λίστας υπογραφών	PUT	api/v2/positions/customApproval/{id}	UpdateCustomApprovalCommand

Περιγραφή	HTTPS	endpoint	Command/Query
Διαγραφή λίστας υπογραφών	DELETE	api/v2/positions/customApproval/{id}	DeleteCustomApprovalCommand
Δημιουργία λίστας αποδεκτών	POST	api/v2/positions/customDistribution	CreateCustomDistributionCommand
Ενημέρωση λίστας αποδεκτών	PUT	api/v2/positions/customDistribution/{Id}	UpdateCustomDistributionCommand
Διαγραφή λίστας αποδεκτών	DELETE	api/v2/positions/customDistribution/{Id}	DeleteCustomDistributionCommand
Δημιουργία λίστας υπογραφών εργασίας	POST	api/v2/positions/customRecipient	CreateCustomRecipientsCommand
Ενημέρωση λίστας υπογραφών εργασίας	PUT	api/v2/positions/customRecipient/{id}	UpdateCustomRecipientCommand
Διαγραφή λίστας υπογραφών εργασίας	DELETE	api/v2/positions/customRecipient/{id}	DeleteCustomRecipientsCommand
Λίστα Υπογραφόντων	GET	api/v2/positions/approval api/v2/positions/approval/iris	ApprovalQuery
Λίστα Υπογραφόντων Άδειας	GET	api/v2/positions/leaveApproval api/v2/positions/leaveApproval/iris	LeaveApprovalQuery
Λίστα Θέσεων για Διανομή Μέσω Ίριδας	GET	api/v2/positions/distribution/iris/{forRoot?	IrisDistributionQuery
Λίστα Θέσεων για Διανομή Μέσω ΕΣΔΑ	GET	api/v2/positions/distribution/esda	EsdaDistributionQuery
Λίστα Θέσεων για Διανομή Μέσω Επαφών	GET	api/v2/positions/distribution/external	ExternalDistributionQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Λίστα με Πίνακες Αποδεκτών για Διανομή από Πίνακες Διανομής	GET	api/v2/positions/distribution/distributionTables	DistributionTablesQuery
Λίστα με Αποδέκτες ενός Πίνακα Αποδεκτών βάσει id	GET	api/v2/positions/distribution/distributionTables/{id}	DistributionTableQuery
Αποδέκτες με σελιδοποίηση	GET	api/v2/positions/recipients/{isUnit}	RecipientsQuery
Αποδέκτες με αναζήτηση	GET	api/v2/positions/recipients/all/{isUnit}	RecipientsAllQuery
Αποδέκτες πρωτοκόλλου	GET	api/v2/positions/registry/recipients	RegistryRecipientQuery
Λίστες Υπογραφόντων	GET	api/v2/positions/customApproval api/v2/positions/approval/customTables api/v2/positions/leaveApproval/customTables	CustomApprovalQuery
Λίστα Υπογραφόντων βάσει Id	GET	api/v2/positions/customApproval/{id}	CustomApprovalByIdQuery
Λίστες Αποδεκτών	GET	api/v2/positions/customDistribution api/v2/positions/distribution/customTables	CustomDistributionQuery
Λίστα Αποδεκτών	GET	api/v2/positions/customDistribution/{Id}	CustomDistributionByIdQuery
Λίστες Υπογραφόντων εργασίας	GET	api/v2/positions/customRecipient api/v2/positions/recipients/customTables	CustomRecipientQuery
Λίστα Υπογραφόντων εργασίας βάσει Id	GET	api/v2/positions/customRecipient/{Id}	CustomRecipientByIdQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Λίστα θέσεων φορέα	GET	api/v2/positions/managed/roots/{Id}	GetRootPositionsQuery

6.11 Πρωτόκολλο - Registry

6.11.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Πρωτόκολλο / Εισαγωγή εισερχόμενης εγγραφής	/registry/insert/inbound	InsertInbound	
Πρωτόκολλο / Εισαγωγή εξερχόμενης εγγραφής	/registry/insert/outbound	InsertOutbound	
Πρωτόκολλο / Πρωτόκολλο εισερχόμενης αλληλογραφίας	/registry/inbound	RegistryIn	
Πρωτόκολλο / Πρωτόκολλο εξερχόμενης αλληλογραφίας	/registry/outbound	RegistryOut	

6.11.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Πρωτοκόλληση Εγγράφου	POST	api/v2/registry	InsertRegistryEntryCommand
Απόρριψη Πρωτοκόλλησης	PUT	api/v2/registry/{id}	CancelRegistryEntryCommand
Αρχικοποίηση νέου Πρωτοκολλημένου Εγγράφων	GET	api/v2/registry/new	NewRegistryEntryQuery
Προβολή Εισερχόμενων Πρωτοκολλημένων Εγγράφων	GET	api/v2/registry/inbound	GetRegistryInboundEntriesQuery
Προβολή Εξερχομένων Πρωτοκολλημένων Εγγράφων	GET	api/v2/registry/outbound	GetRegistryOutboundEntriesQuery
Λίστα σχετικών πρωτοκόλλου	GET	api/v2/registry/references	RegistryReferencesQuery

6.12 Άδειες - Leaves

6.12.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Εργαλεία / Υπογεγραμμένες άδειες	/leaves/approved	Approved	ΠΑ
Άδειες / Επεξεργασία αδείας	/leaves/clone/{leaveId}	Clone	ΠΑ
Άδειες / Δημιουργία αδείας	/leaves/create	Create	ΠΑ
Άδειες / Επεξεργασία αδείας	/leaves/edit/{leaveId}	Edit	ΠΑ
Εργαλεία / Αιτήσεις αδειών	/leaves	LeavesAll	ΠΑ
Εργαλεία / Άδειες προς υπογραφή	/leaves/pending	Pending	ΠΑ
Εργαλεία / Υπογεγραμμένες άδειες	/leaves/view/{leaveId}	View	ΠΑ

6.12.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Αποθήκευση Νέας Άδειας	POST	api/v2/leaves	CreateLeaveCommand
Υπογραφή/Αποδοχή Άδειας	POST	api/v2/leaves/{id}/accept	AcceptLeaveCommand
Απόρριψη Άδειας	POST	api/v2/leaves/{id}/reject	RejectLeaveCommand
Διαγραφή Άδειας	DELETE	api/v2/leaves/{id}	DeletelLeaveCommand
Ενημέρωση Άδειας	PUT	api/v2/leaves/{id}	UpdateLeaveCommand
Προβολή Λίστας Αδειών	GET	api/v2/leaves	GetLeavesQuery
Προβολή Λίστας μη Εγκεκριμένων Αδειών	GET	api/v2/leaves/approval/pending	GetApprovalPendingLeavesQuery

Περιγραφή	HTTPS	endpoint	Command/Query
Προβολή Λίστας Εγκεκριμένων Αδειών	GET	api/v2/leaves/approval/approved	GetApprovedLeavesQuery
Αρχικοποίηση Νέας Άδειας	GET	api/v2/leaves/new	GetNewLeaveQuery
Προβολή Άδειας Βάσει id	GET	api/v2/leaves/{id}	GetLeaveByIdQuery
Κλωνοποίηση Εγκεκριμένης Άδειας	GET	api/v2/leaves/{id}/clone	GetLeaveCloneQuery
Λίστα Χωρών για Επιλογή Προορισμού	GET	api/v2/leaves/countries	GetCountriesQuery
Λίστα Πόλεων για Επιλογή Προορισμού	GET	api/v2/leaves/cities	GetCitiesQuery
Λίστα Μέσων Μεταφοράς για Επιλογή Μετακίνησης	GET	api/v2/leaves/transports	GetTransportsQuery
Σύνολα Αδειών Υπαλλήλου	GET	api/v2/leaves/totals	GetTotalsQuery
Λίστα Αδειών ως Σχετικά	GET	api/v2/leaves/references	GetLeaveReferencesQuery
Αποθήκευση Απόδειξης Εγγράφου Αδείας	GET	api/v2/leaves/receipt/{Id}	GetLeaveReceiptQuery

6.12.3 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Αντικατάστασης	POST	api/v2/substitutions	CreateSubstitutionCommand
Ενημέρωση Αντικατάστασης	PUT	api/v2/substitutions/{id}	UpdateSubstitutionCommand
Διαγραφή Αντικατάστασης	DELETE	api/v2/substitutions/{id}	DeleteSubstitutionCommand
Προβολή Καθηκόντων από Αντικαταστάσεις	GET	api/v2/substitutions/received	GetReceivedDutiesQuery
Προβολή Ανατιθέμενων Αντικαταστάσεων	GET	api/v2/substitutions/provided	GetProvidedDutiesQuery
Αναζήτηση Αντικατάστασης βάσει του Id	GET	api/v2/substitutions/{id}	GetSubstitutionByIdQuery

6.13 Αντικαταστάσεις - Substitutions

6.13.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Αντικαταστάσεις / Νέα αντικατάσταση	/substitutions/create	Create	
Αντικαταστάσεις / Επεξεργασία αντικατάστασης	/substitutions/edit/{subId}	Edit	
Εργαλεία / Καθήκοντα από αντικαταστάσεις	/substitutions/fromsubstitution	FromSubstitution	
Εργαλεία / Ανατιθέμενα καθήκοντα	/substitutions/substituted	Substituted	

6.13.2 Πίνακας Back-end

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Αντικατάστασης	POST	api/v2/substitutions	CreateSubstitutionCommand
Ενημέρωση Αντικατάστασης	PUT	api/v2/substitutions/{id}	UpdateSubstitutionCommand
Διαγραφή Αντικατάστασης	DELETE	api/v2/substitutions/{id}	DeleteSubstitutionCommand
Προβολή Καθηκόντων από Αντικαταστάσεις	GET	api/v2/substitutions/received	GetReceivedDutiesQuery
Προβολή Ανατιθέμενων Αντικαταστάσεων	GET	api/v2/substitutions/provided	GetProvidedDutiesQuery
Αναζήτηση Αντικατάστασης βάσει του Id	GET	api/v2/substitutions/{id}	GetSubstitutionByIdQuery

6.14 Υποστήριξη - Support

6.14.1 Πίνακας Front-end

Όνομα	Url	Component	ΠΑ / ΕΔ
Υποστήριξη / Ημερολόγιο Τροποποιήσεων	/support/changelog	ChangeLog	

Όνομα	Url	Component	ΠΑ / ΕΔ
Υποστήριξη / Συχνές ερωτήσεις	/support/faqs	Faqs	
Υποστήριξη	/support	Landing	
Υποστήριξη / Ρυθμίσεις	/support/settings	Settings	
Γνωρίστε την ομάδα ανάπτυξης	/support/team	Team	

6.14.2 Σημειώσεις Χρήστη

Περιγραφή	HTTPS	endpoint	Command/Query
Δημιουργία Σημείωσης	POST	api/v2/userNotes	CreateUserNoteCommand
Ενημέρωση Σημείωσης	PUT	api/v2/userNotes/{id}	UpdateUserNoteCommand
Διαγραφή Σημείωσης	DELETE	api/v2/userNotes/{id}	DeleteUserNoteCommand
Λίστα με τις σημειώσεις του Χρήστη	GET	api/v2/userNotes	GetNotesByUserQuery

6.14.3 Στατιστικά

Περιγραφή	HTTPS	endpoint	Command/Query
Στατιστικά Εγγράφων	GET	api/v2/stats/documents	StatsDocumentsQuery
Στατιστικά Εργασιών	GET	api/v2/stats/tasks	StatsTasksQuery
Στατιστικά Εργαλείων	GET	api/v2/stats/tools	StatsToolsQuery

6.14.4 Συχνές ερωτήσεις

Περιγραφή	HTTPS	endpoint	Command/Query
Λίστα Συχνών Ερωτήσεων	GET	api/v2/faqs	GetFAQsQuery
Αναζήτηση Συχνής Ερώτησης βάσει του Id	GET	api/v2/faqs/{id}	GetFaqQuery

6.14.5 Ανακοινώσεις

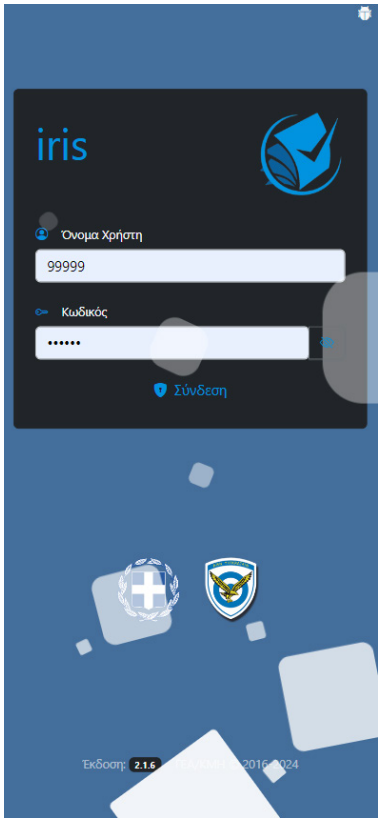
Περιγραφή	HTTPS	endpoint	Command/Query
Ανακοινώσεις Χρήστη	GET	api/v2/support/announcements	GetUserAnnouncementsQuery

7 Επισκόπηση λειτουργιών

Παρακάτω παρουσιάζονται οι οθόνες της εφαρμογής, καθώς και οι λειτουργίες αυτών.

7.1 Οθόνη κλειδώματος

Η οθόνη κλειδώματος (Εικόνα 1) είναι η πρώτη οθόνη που βλέπει ο χρήστης καθώς μπαίνει στην εφαρμογή. Εκεί υπάρχει το πεδίο Όνομα Χρήστη και Κωδικός, όπου μπορεί να εισάγει τα διαπιστευτήριά του για να συνδεθεί στην εφαρμογή.



Εικόνα 1. Οθόνη κλειδώματος

7.2 Υποστήριξη

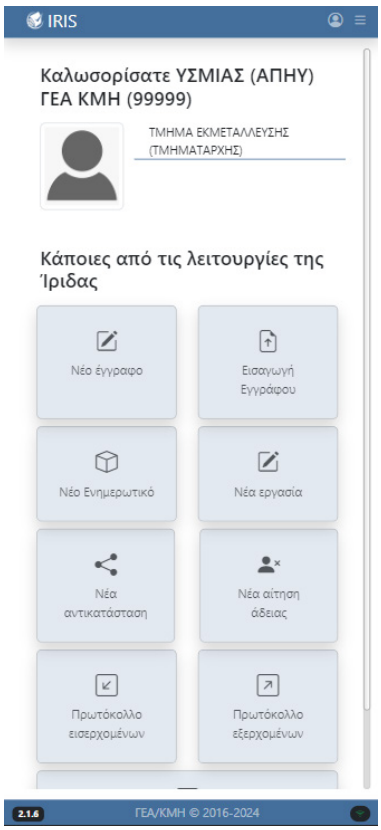
Ο χρήστης μπορεί να πλοηγηθεί στην υποστήριξη πατώντας το εικονίδιο πάνω δεξιά, δίπλα σε αυτό του μενού.

7.3 Αρχική Οθόνη

Στην αρχική οθόνη της εφαρμογής (Εικόνα 2) παρουσιάζονται τα στοιχεία του προφίλ με το οποίο έχει κάνει είσοδο ο χρήστης (π.χ. *ΥΣΜΙΑΣ (ΑΠΗΥ) ΓΕΑ ΚΜΗ (999999) - ΤΜΗΜΑ ΕΚΜΕΤΑΛΛΕΥΣΗΣ (ΤΜΗΜΑΤΑΡΧΗΣ)*). Στη συνέχεια με τη βοήθεια εικονιδίων παρουσιάζονται κάποιες βασικές λειτουργίες της εφαρμογής ως εξής:

- Η δημιουργία ενός εγγράφου με την επιλογή *Νέο Έγγραφο*

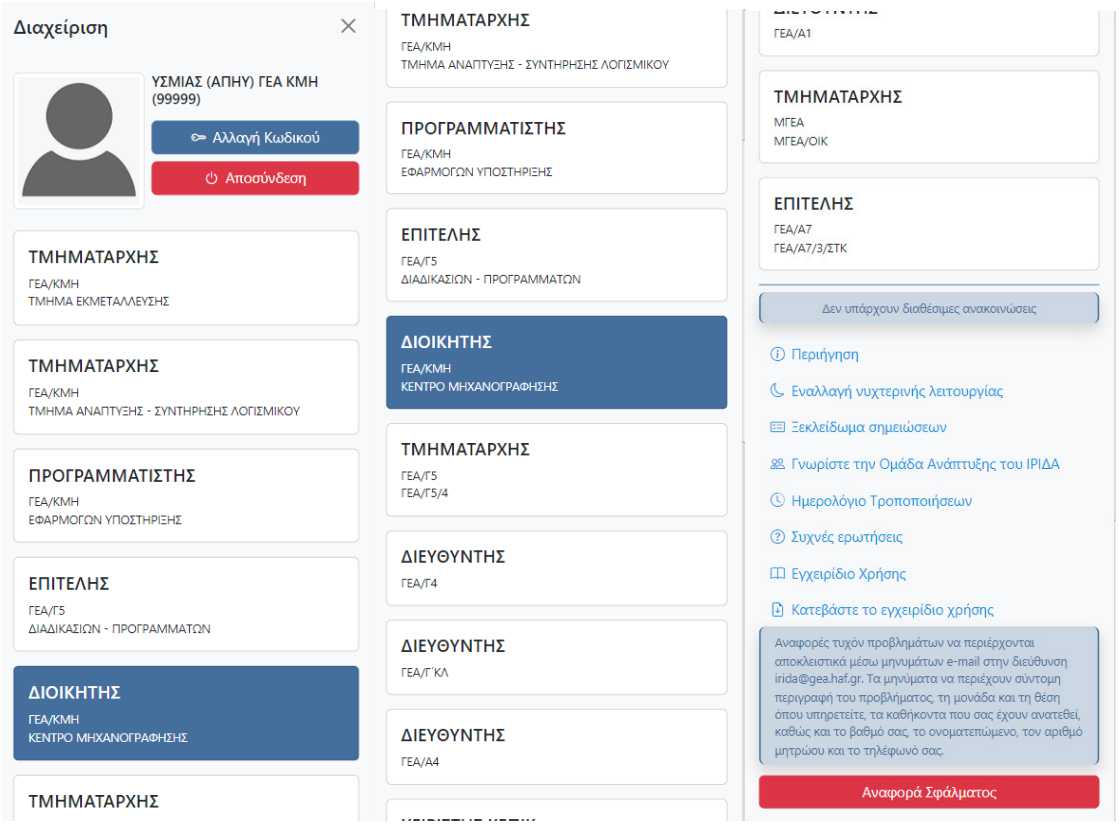
- Η εισαγωγή εγγράφου με την αντίστοιχη επιλογή
- Η δημιουργία ενημερωτικού εγγράφου με την επιλογή *Νέο Ενημερωτικό*
- Η δημιουργία νέας εργασίας με την αντίστοιχη επιλογή
- Η δημιουργία νέας αντικατάστασης με την αντίστοιχη επιλογή
- Η δημιουργία νέας αίτησης άδειας με την αντίστοιχη επιλογή
- Η προβολή του πρωτοκόλλου εισερχομένων με την αντίστοιχη επιλογή
- Η προβολή του πρωτοκόλλου εξερχομένων με την αντίστοιχη επιλογή



Εικόνα 2. Αρχική οθόνη

7.4 Επιλογή ρόλου (προφίλ)

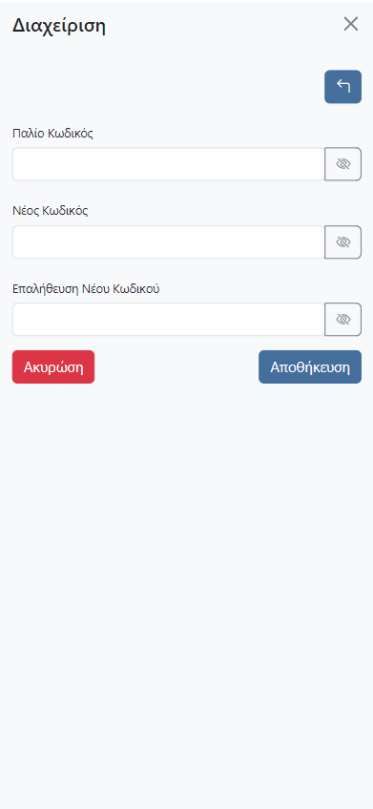
Ο ρόλος ή προφίλ του χρήστη αποτελείται από το συνδυασμό του καθήκοντος και της θέσης στην οποία ασκείται αυτό. Τα διαθέσιμα προφίλ μπορεί να είναι είτε ορισμένα από το διαχειριστή της Μονάδας του χρήστη, είτε να προκύπτουν από Αντικατάσταση ή Συντονισμό. Επισημαίνεται ότι χρήστες με τον ίδιο ρόλο έχουν ίδια πρόσβαση στα ίδια έγγραφα της εφαρμογής.



Εικόνα 3. Επιλογή ρόλου (προφίλ)

7.5 Αλλαγή Κωδικού Πρόσβασης

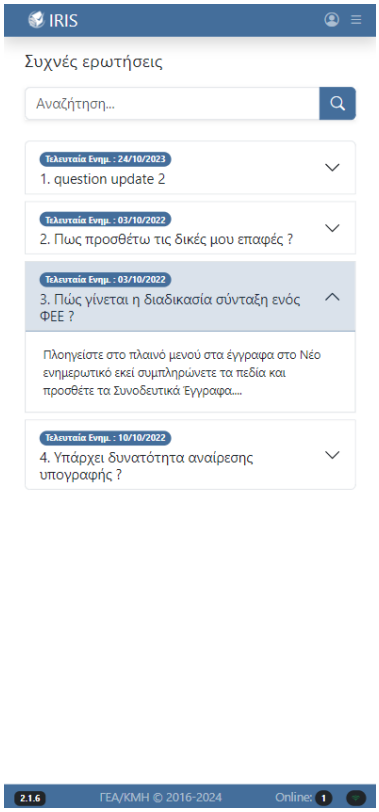
Η αλλαγή του password (Εικόνα 4) μπορεί να γίνει οποιαδήποτε στιγμή το επιθυμεί ο χρήστης και όχι ανά κάποιο ορισμένο χρονικό διάστημα. Θα πρέπει ο χρήστης να εισάγει τον τρέχων κωδικό του και δύο φορές τον νέο, ο οποίος θα πρέπει να συμμορφώνεται με την ακόλουθη πολιτική πολυπλοκότητας: να αποτελείται από τουλάχιστον 8 χαρακτήρες, ένα κεφαλαίο, ένα πεζό γράμμα, έναν αριθμό και έναν ειδικό χαρακτήρα από τους ! @ # $ % ^ & *.



Εικόνα 4. Οθόνη αλλαγής κωδικού

7.6 Συχνές Ερωτήσεις

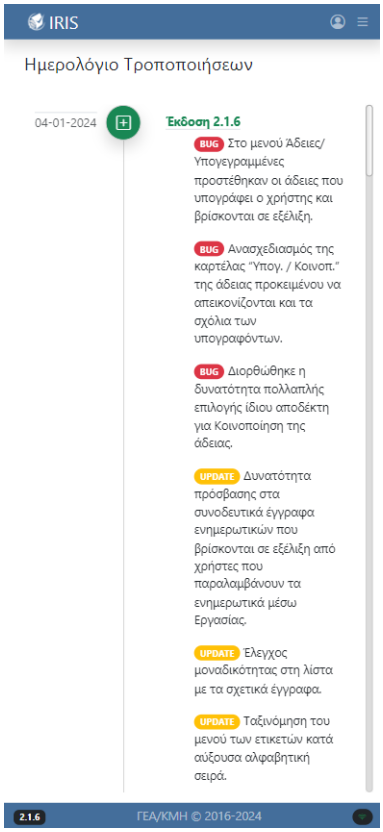
Ο χρήστης μπορεί να ενημερώνεται για τις λειτουργίες της εφαρμογής μέσω αυτής της σελίδας ερωτήσεων και απαντήσεων (Εικόνα 5). Εδώ υπάρχουν συχνές ερωτήσεις χρηστών για την εφαρμογή και τις λειτουργίες της με τις απαντήσεις τους, ώστε να μπορεί εύκολα ο χρήστης να λύνει ότι πρόβλημα του παρουσιάζεται.



Εικόνα 5. Οθόνη συχνών ερωτήσεων και απαντήσεων

7.7 Ημερολόγιο Τροποποιήσεων

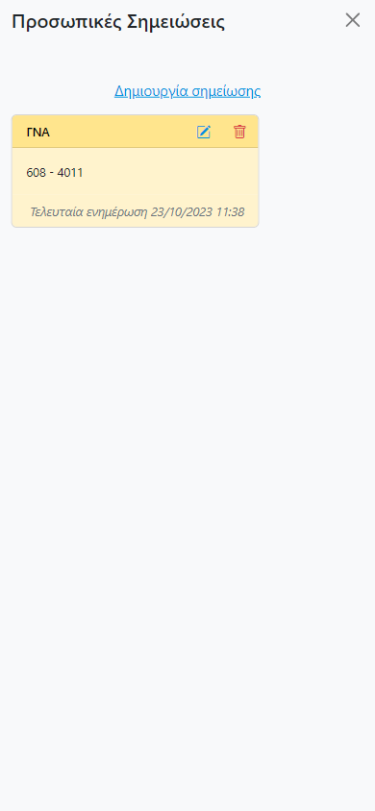
Ο χρήστης στη συγκεκριμένη σελίδα (Εικόνα 6) μπορεί να βλέπει ιστορικό των αλλαγών που έχουν γίνει στην εφαρμογή, και να ενημερώνεται για τυχόν προσθήκες, ενημερώσεις και διορθώσεις λειτουργιών.



Εικόνα 6. Ιστορικό τροποποιήσεων

7.8 Σημειώσεις

Ο χρήστης μπορεί να δημιουργεί προσωπικές σημειώσεις οι οποίες απεικονίζονται στο δεξί μέρος της εφαρμογής για τους χρήστες επιτραπέζιων εφαρμογών, είτε στο αναδυόμενο παράθυρο για τους χρήστες κινητών συσκευών (Εικόνα 7). Παρέχεται δυνατότητα δημιουργίας απεριόριστου πλήθους σημειώσεων, επεξεργασίας και διαγραφής αυτών. Οι σημειώσεις είναι δεν είναι ορατές μεταξύ χρηστών που έχουν το ίδιο προφίλ. Η διαγραφή είναι οριστική και η σημείωση μη ανακτήσιμη.

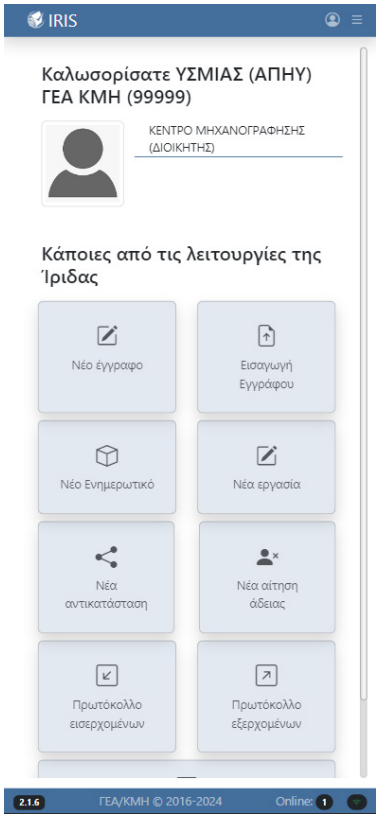


Εικόνα 7. Οθόνη σημειώσεων

7.9 Έγγραφα

7.9.1 Αρχική Σελίδα Ενότητας

Η αρχική σελίδα της ενότητας των εγγράφων (Εικόνα 8) ομοιάζει με εκείνη της αρχικής οθόνης της εφαρμογής με τα εικονίδια να εκτελούν ακριβώς τις ίδιες συντομεύσεις όπως περιγράφηκαν προηγουμένως.



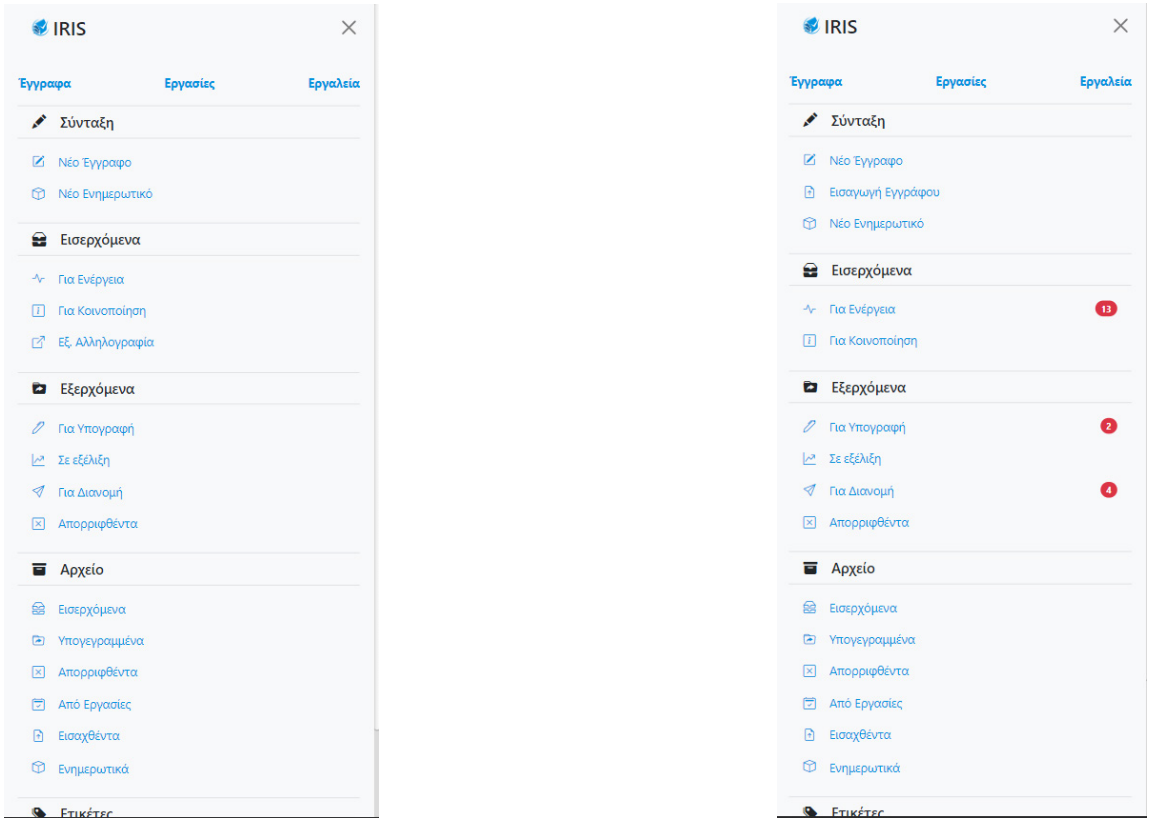
Εικόνα 8. Αρχική σελίδα ενότητας εγγράφων

7.9.2 Μενού Ενότητας Εγγράφων

Στο μενού της ενότητας των εγγράφων (Εικόνα 9) παρέχονται οι εξής επιλογές:

- Σύνταξη
    - Νέο έγγραφο
    - Εισαγωγή Εγγράφου
    - Νέο Ενημερωτικό
- Εισερχόμενα
    - Για Ενέργεια (Εμφανίζεται αριθμητική ένδειξη για μη αναγνωσμένα - νέα έγγραφα)
    - Για Κοινοποίηση
    - Εξ. Αλληλογραφία (Μόνο για τους χρήστες του Δημοσίου)
- Εξερχόμενα
    - Για Υπογραφή (Εμφανίζεται αριθμητική ένδειξη για μη αναγνωσμένα - νέα έγγραφα)
    - Σε εξέλιξη
    - Για Διανομή (Εμφανίζεται αριθμητική ένδειξη για μη αναγνωσμένα - νέα έγγραφα)
    - Απορριφθέντα

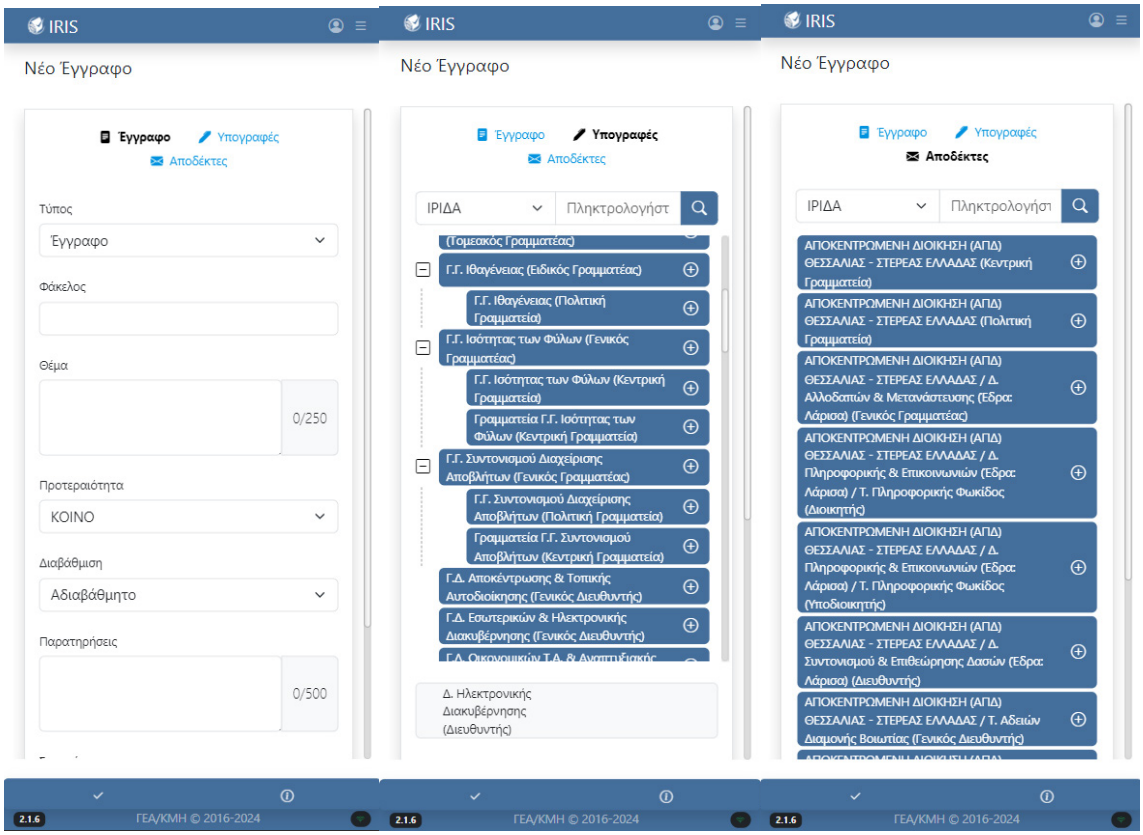
- Αρχείο
    - Εισερχόμενα
    - Υπογεγραμμένα
    - Απορριφθέντα
    - Από Εργασίες
    - Εισαχθέντα
    - Ενημερωτικά
- Ετικέτες
    - Ετικέτες του χρήστη…



Εικόνα 9. Μενού ενότητας εγγράφων

7.9.3 Νέο Έγγραφο

Η δημιουργία νέου εγγράφου χωρίζεται σε 3 καρτέλες (Εικόνα 10) ως ακολούθως. Η αποθήκευση γίνεται και για τις 3 καρτέλες και μεταφέρει το έγγραφο στο μενού “Για Υπογραφή” του συντάκτη.



Εικόνα 10. Δημιουργία νέου εγγράφου

7.9.3.1 Βασικά Στοιχεία του Εγγράφου

Κάνοντας κλικ στο Νέο Έγγραφο μεταφερόμαστε στην παραπάνω σελίδα όπου και συμπληρώνουμε τα στοιχεία του εγγράφου.

- Στον Τύπο εγγράφου (υποχρεωτικό πεδίο) επιλέξτε μια από τις διαθέσιμες επιλογές της λίστας.
- Στον Φάκελο επιλέξτε τον κατάλληλο αναζητώντας (είτε αριθμητικά, είτε λεκτικά) από το ενιαίο θεματολόγιο.
- Το Θέμα (υποχρεωτικό πεδίο) συμπληρώνεται με την επιλογή του φακέλου και μπορεί να τροποποιηθεί.
- Στην Προτεραιότητα (υποχρεωτικό πεδίο) επιλέξτε μια από τις διαθέσιμες επιλογές της λίστας.
- Στην Διαβάθμιση (υποχρεωτικό πεδίο) επιλέξτε μια από τις διαθέσιμες επιλογές της λίστας.
- Στις Παρατηρήσεις προαιρετικά συμπληρώνονται σύντομες σημειώσεις που αφορούν το έγγραφο (πρέπει οι σημειώσεις είναι ορατές από όλους, υπογράφοντες και αποδέκτες).
- Στα Σχετικά, μπορείτε να συνδέσετε το έγγραφο σας με άλλα σχετικά έγγραφα όπως περιγράφεται παρακάτω:
    1. Εισαγωγή σχετικού εγγράφου, στο οποίο ο χρήστης έχει ήδη πρόσβαση στο Ίριδα.

2. Προσθήκη σχετικού εκτός συστήματος. Εγγραφές δηλαδή πρωτοκόλλου για έγγραφα τα οποία δεν έχουν παραχθεί από το σύστημα.

Στις δύο πρώτες περιπτώσεις υπάρχει η επιλογή της συσχέτισης του νέου εγγράφου ως απάντηση στο σχετικό.

3. Εισαγωγή ταυτότητας σχετικού εγγράφου ως απλό κείμενο.
4. Προσθήκη σχετικού από τον υπολογιστή σας.
5. Προσθήκη σχετικού από την βιβλιοθήκη.

- Στα Σχέδια, μπορείτε να προσθέσετε ένα δικό σας αρχείο από τον υπολογιστή σας (όπως σχέδια εγγράφου, συνημμένα και παραρτήματα).

7.9.3.2 Επιλογή Υπογραφόντων

Η συμπλήρωση της λίστας των υπογραφών δεν έχει κάποια αυστηρή ιεραρχική λογική και πρέπει ο χρήστης να την ενημερώσει ως ακολούθως:

1. Τρόπος αναζήτησης υπογράφοντος (μεμονωμένης θέσης ή προσωποποιημένου πίνακα). 2. Αναζήτηση υπογράφοντος ή προσωποποιημένου πίνακα.
2. Τα αποτελέσματα της αναζήτησης εμφανίζονται στην αριστερή λίστα της οθόνης.
3. Με αυτό το κουμπί (εικονίδιο της πρόσθεσης) επιλέγεται ένας υπογράφων (ή και ολόκληρος πίνακας) και προστίθεται στην δεξιά λίστα της οθόνης.
4. Η λίστα πρέπει να έχει τουλάχιστον έναν υπογράφων - τον συντάκτη - ο οποίος δεν μπορεί να αφαιρεθεί.
5. Σε περίπτωση λάθους αφαιρούμε κάποιον από την λίστα.
6. Με αυτό το κουμπί επιλέγεται ο χαρακτηρισμός του υπογράφοντος (Για υπογραφή ή για Συντονισμό). Η επιλογή ενός υπογράφοντα ως Συντονιστή δηλώνει ότι αυτός ο χρήστης έχει μόνο δικαίωμα υπογραφής του εγγράφου και όχι απόρριψης ή επεξεργασίας.
7. Ένας υπογράφων μπορεί να βρίσκεται πάνω από μία φορά στη λίστα. Έτσι απεικονίζεται το πλήθος των συμμετοχών του.
8. Η σειρά των υπογραφόντων αλλάζει μετακινώντας (drag & drop) στην κατάλληλη θέση τον υπογράφοντα.

7.9.3.3 Επιλογή Αποδεκτών

Η συμπλήρωση της λίστας των αποδεκτών γίνεται ως ακολούθως:

1. Παρέχονται 5 τρόποι αναζήτησης Αποδέκτη:
   i. Από Ίριδα. Θέσεις εντός συστήματος που έχουν οριστεί ως αποδέκτες αλληλογραφίας.
   ii. Από ΕΣΔΑ. Θέσεις εντός ΕΣΔΑ (σύστημα διακίνησης εγγράφων που διαθέτει ο στρατός ξηράς) που έχουν οριστεί ως αποδέκτες αλληλογραφίας.
   iii. Από Επαφές. Επαφές του χρήστη. Σε περίπτωση αδυναμίας εύρεσης της επαφής, μπορεί να προστεθεί επί τόπου στο έγγραφο. Σε αυτήν την περίπτωση δεν θα γίνει αποθήκευση στις επαφές.
   iv. Από πίνακα Διανομής. Προκαθορισμένοι πίνακες - διαθέσιμοι για όλους τους χρήστες - που δύναται να περιέχουν συνδυαστικά και τις 3 προαναφερθείσες κατηγορίες επαφών.

v. Από προσωποποιημένο πίνακα διανομής, όπου και αυτοί δύναται να περιέχουν συνδυαστικά και τις 3 κατηγορίες επαφών. (Παραμετροποιείται μέσω της ενότητας "Εργαλεία")

2. Αναζήτηση Αποδέκτη ή προσωποποιημένου πίνακα.
3. Με αυτό το κουμπί επιλέγεται ένας αποδέκτης (ή και ολόκληρος πίνακας) και προστίθεται στην δεξιά λίστα της οθόνης.
4. Επιλογή πλήθους αποδεκτών που εμφανίζονται ανά σελίδα.
5. Αποτελέσματα αναζήτησης.
6. Μετάβαση σε σελίδα.
7. Σε περίπτωση λάθους αφαιρούμε κάποιον από την λίστα.
8. Διαγραφή ολόκληρης της λίστας. Η ύπαρξη τουλάχιστον ενός αποδέκτη είναι υποχρεωτική.
9. Με αυτό το κουμπί επιλέγεται ο χαρακτηρισμός του αποδέκτη. (Για ενέργεια ή για κοινοποίηση).

7.9.4 Ενέργειες επί του Εγγράφου

Γενικά, οι ενέργειες επί του εγγράφου συνοψίζονται στην ακόλουθη λίστα και θα αναλυθούν περαιτέρω στη συνέχεια. Οι ενέργειες αναπαρίστανται με εικονίδια (Εικόνα 11). Ποτέ όλα τα εικονίδια δεν εμφανίζονται ταυτόχρονα αλλά με κριτήριο την κατάσταση του εγγράφου και τα δικαιώματα του χρήστη.

1. Επεξεργασία

2. Υπογραφή

3. Απόρριψη

4. Πρωτοκόλληση

5. Διανομή

6. Εργασίες Επί του Εγγράφου

7. Προσθήκη σε Σημαντικό Θέμα

8 Προσθήκη σε Στόχο

9. Χρέωση Εγγράφου για Ενέργεια

10. Χρέωση Εγγράφου για Ενημέρωση

11. Διαγραφή

12. Αρχειοθέτηση Εγγράφου

13. Απάντηση

14. Κλωνοποίηση Εγγράφου

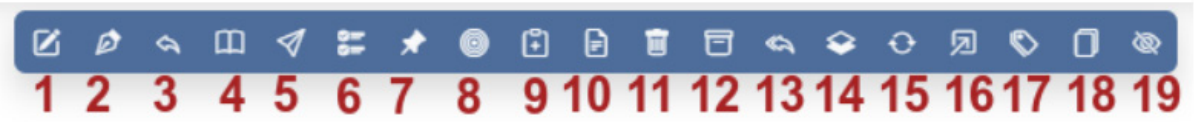
15. Ορθή Επανάληψη

16. Μετάβαση στο Ενημερωτικό

17. Επεξεργασία ετικετών

18. Συνέχεια Ενημέρωσης

19. Επισήμανση ως μη αναγνωσμένο

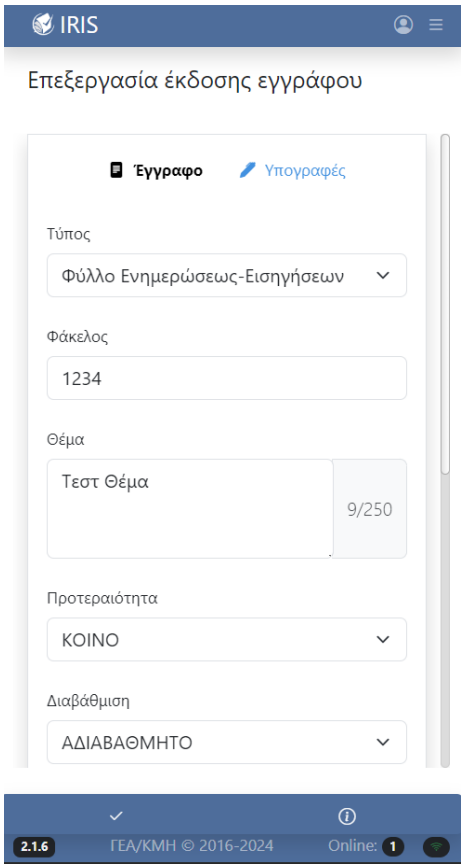


Εικόνα 11. Εικονίδια ενεργειών εγγράφου

7.9.4.1 Επεξεργασία Εγγράφου

Η επεξεργασία του εγγράφου [εικονίδιο 1] (Εικόνα 12) γίνεται από κάποιον Υπογράφοντα εφόσον το έγγραφο βρίσκεται στα έγγραφα “Για Υπογραφή” του και έχει τα απαραίτητα δικαιώματα. Κατά την επεξεργασία ο χρήστης μπορεί να αλλάξει τα πάντα στο έγγραφο, εκτός από τους προηγούμενους αποδέκτες και τον τύπο του. Την πρώτη φορά που ένας χρήστης επεξεργάζεται το έγγραφο δημιουργείται νέα έκδοση σε αυτό. Τέλος, επεξεργασία επιτρέπεται και στον συντάκτη του εγγράφου σε ακόμα δύο περιπτώσεις:

1) Το έγγραφο να έχει απορριφθεί και επιθυμεί να δημιουργήσει νέα έκδοση
2) Το έγγραφο να έχει πρωτοκολληθεί και ο συντάκτης να επεξεργάζεται το Ακριβές Αντίγραφο.

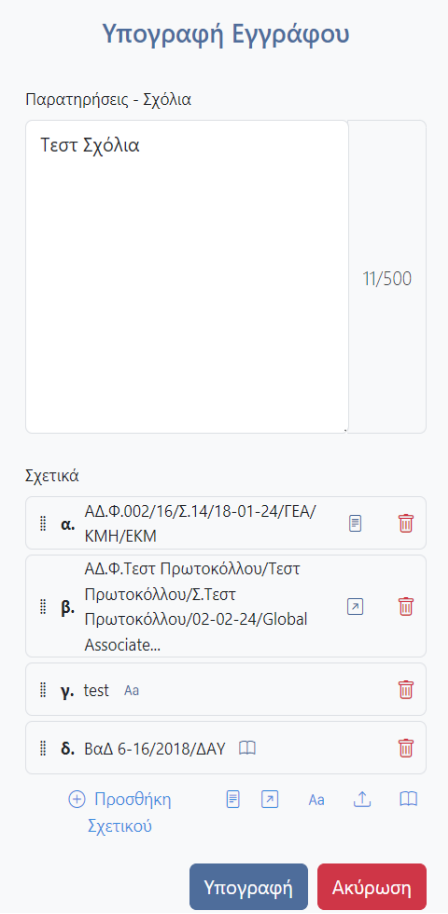


Εικόνα 12. Επεξεργασία εγγράφου

7.9.4.2 Υπογραφή Εγγράφου

Η υπογραφή του εγγράφου [εικονίδιο 2] (Εικόνα 13) γίνεται από κάποιον Υπογράφοντα εφόσον το έγγραφο βρίσκεται στα έγγραφα “Για Υπογραφή”. Εκτελώντας την ενέργεια της

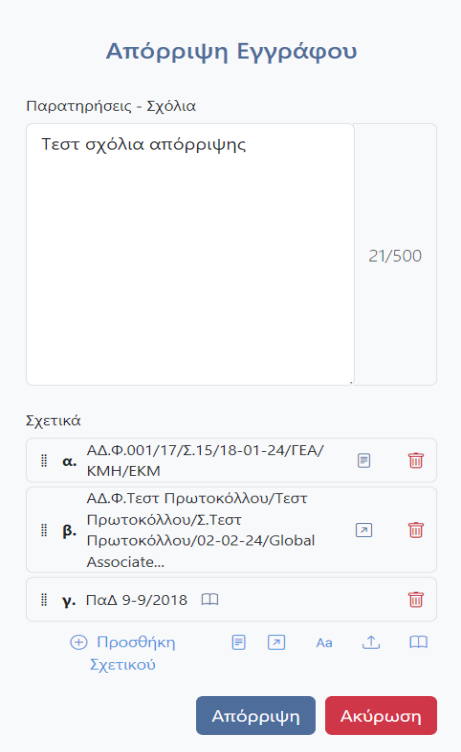
υπογραφής εμφανίζεται το παραπάνω παράθυρο, στο οποίο ο χρήστης μπορεί να συμπληρώσει κάποιες Παρατηρήσεις - Σχόλια ή επιφύλαξη / διαφωνία που έχει για το συγκεκριμένο έγγραφο και να εισάγει κάποια σχετικά (με την λογική των σχετικών που ακολουθείται στην εφαρμογή). Η υπογραφή του εγγράφου ως ενέργεια είναι μη αναιρέσιμη.



Εικόνα 13. Υπογραφή εγγράφου

7.9.4.3 Απόρριψη Εγγράφου

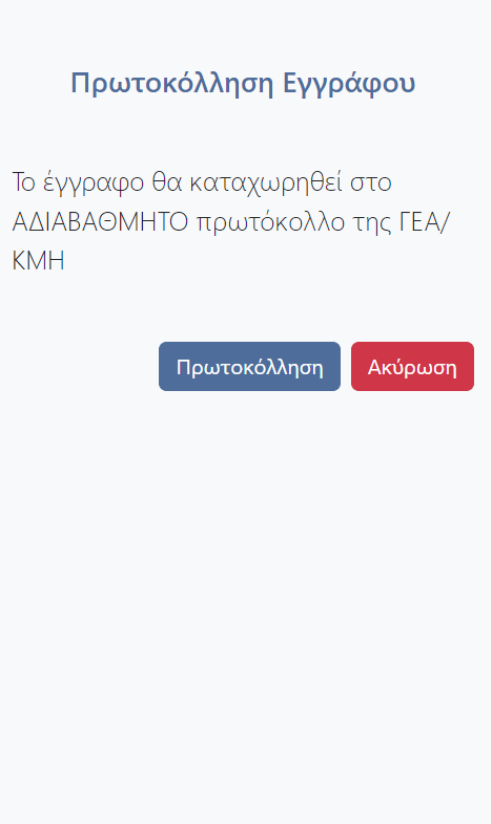
Η απόρριψη του εγγράφου [εικονίδιο 3] (Εικόνα 14) γίνεται από κάποιον Υπογράφων εφόσον το έγγραφο βρίσκεται στα έγγραφα “Για Υπογραφή” του και έχει τα απαραίτητα δικαιώματα (να έχει οριστεί “Υπογράφων” και όχι “Συντονιστής”). Εκτελώντας την ενέργεια της απόρριψης, εμφανίζεται το παρακάτω παράθυρο, αντίστοιχο με της υπογραφής. Η απόρριψη μεταφέρει το έγγραφο στο μενού “Εξερχόμενα -> Απορριφθέντα” (όλων των υπογραφόντων), από όπου ο συντάκτης έχει την επιλογή είτε να το τροποποιήσει δημιουργώντας νέα έκδοση και εκκίνηση της ροής των υπογραφών από την αρχή, είτε να το αρχειοθετήσει οδηγώντας το έγγραφο στο μενού “Αρχείο -> Απορριφθέντα”. Η απόρριψη του εγγράφου είναι μη αναιρέσιμη.



Εικόνα 14. Απόρριψη εγγράφου

7.9.4.4 Πρωτοκόλληση Εγγράφου

Η Πρωτοκόλληση του εγγράφου [εικονίδιο 4] (Εικόνα 15) γίνεται από τον συντάκτη του εγγράφου εφόσον αυτό έχει ολοκληρώσει την ροή των υπογραφών του και βρίσκεται στο μενού “Για Διανομή”. Εκτελώντας την ενέργεια της Πρωτοκόλλησης εμφανίζεται το παρακάτω παράθυρο, το οποίο ενημερώνει τον συντάκτη ότι θα γίνει μια εγγραφή στο βιβλίο Πρωτοκόλλου της Μονάδας του. Τέλος ενημερώνεται ο αριθμός Πρωτοκόλλου και αριθμός Σχεδίου επί του εγγράφου και αυτό επανέρχεται σε επεξεργάσιμη μορφή.



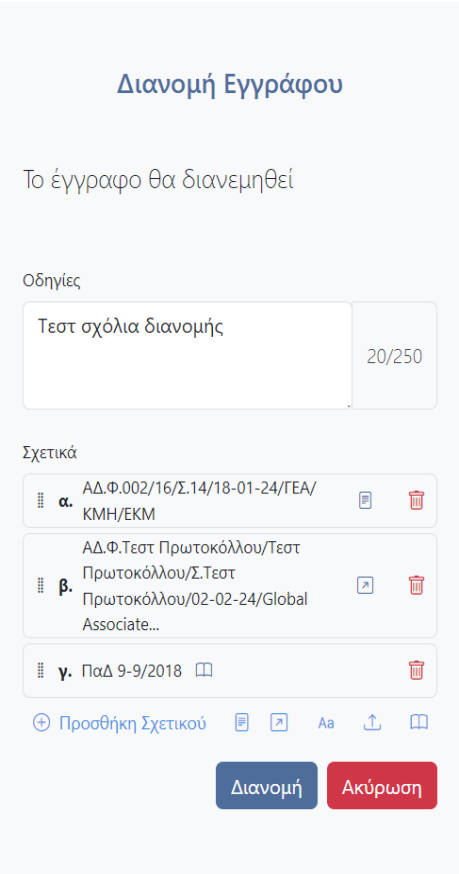
Εικόνα 15. Πρωτοκόλληση εγγράφου

7.9.4.5 Διανομή Εγγράφου

Η Διανομή του εγγράφου [εικονίδιο 5] (Εικόνα 16) γίνεται από τον συντάκτη του εγγράφου εφόσον αυτό έχει Πρωτοκολληθεί και βρίσκεται στο μενού “Για Διανομή”. Πριν την ολοκλήρωση της ενέργειας ο συντάκτης μπορεί- εφόσον αυτό προκύπτει από σχόλια των υπογραφόντων - να τροποποιήσει τον πίνακα Διανομής, το/τα σχέδια εγγράφου, τα σχετικά και την προτεραιότητα. Κατά τη φάση της διανομής το/ τα σχέδια εγγράφου μετατρέπονται σε pdf, στα οποία ενσωματώνεται υποσέλιδο που ακολουθεί την κάτωθι μορφή:

Οι αποδέκτες του συστήματος παραλαμβάνουν άμεσα το έγγραφο και πραγματοποιείται αυτόματα πρωτοκόλληση αυτού στα εισερχόμενα τους. Επίσης το έγγραφο διαβιβάζεται αυτόματα στο ΕΣΔΑ προκειμένου να ενημερώσει τους αποδέκτες του, ενώ για τους λοιπούς αποδέκτες η διανομή γίνεται εκτός Ίριδα με ευθύνη του συντάκτη. Εκτελώντας την ενέργεια της Διανομής εμφανίζεται παρακάτω το παράθυρο, αντίστοιχο με της υπογραφής.

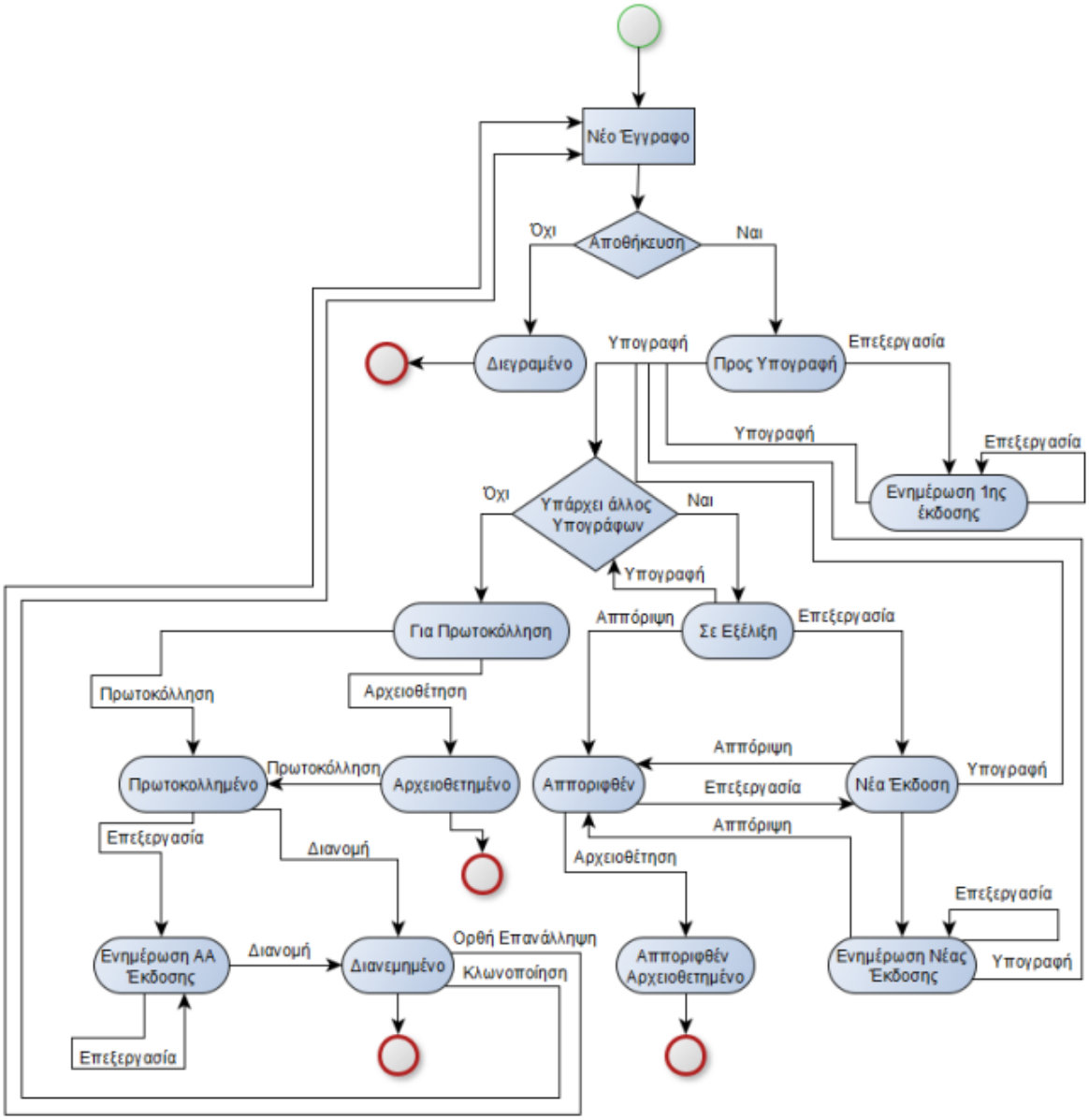
Σε περίπτωση που το έγγραφο είναι τύπου “Σήμα”, η Διανομή του γίνεται από τον χειριστή του γραφείου σημάτων που εξυπηρετεί τη μονάδα του συντάκτη, αφού ο τελευταίος ολοκληρώσει την πρωτοκόλληση. Ο χειριστής αυτός θα πρέπει πρώτα να διαβιβάσει το σήμα χειροκίνητα, στους αποδέκτες που βρίσκονται εκτός συστήματος και στη συνέχεια να πραγματοποιήσει τη διανομή του συμπληρώνοντας την ημερομηνία και ώρα εκπομπής.



Εικόνα 16. Διανομή εγγράφου

7.9.5 Διάγραμμα Ροής Νέου Εγγράφου

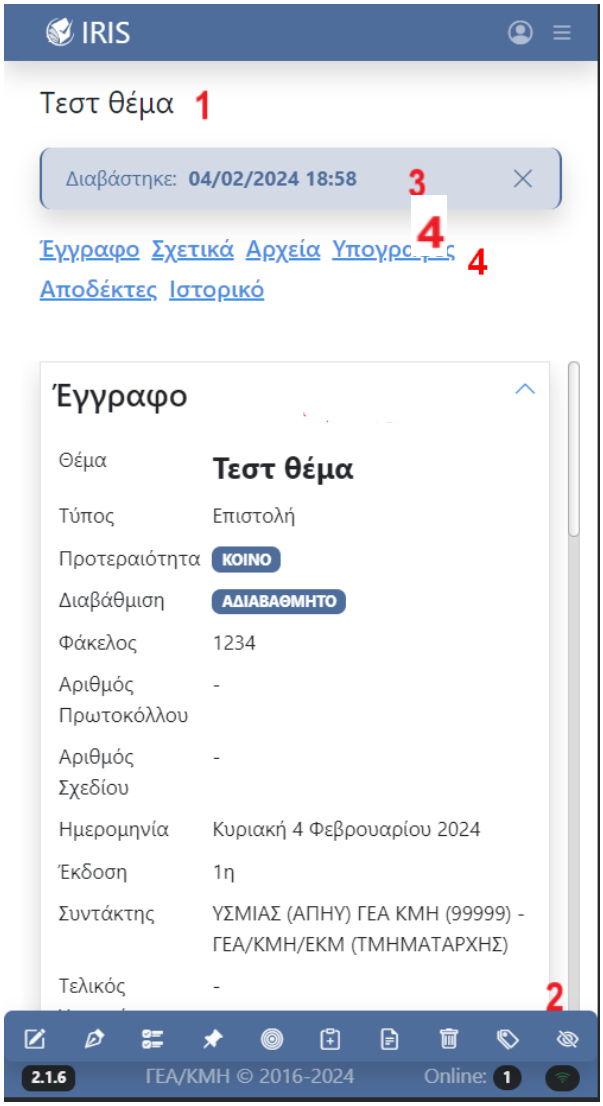
Στη συνέχεια περιγράφεται η ροή δημιουργίας ενός Νέου Εγγράφου (Εικόνα 17).



Εικόνα 17. Διάγραμμα ροής νέου εγγράφου

7.9.5.1 Επισκόπηση Εγγράφου

Όταν ο χρήστης επιλέξει ένα έγγραφο τότε του εμφανίζεται η επισκόπηση του, η οποία αναλύεται ως ακολούθως:

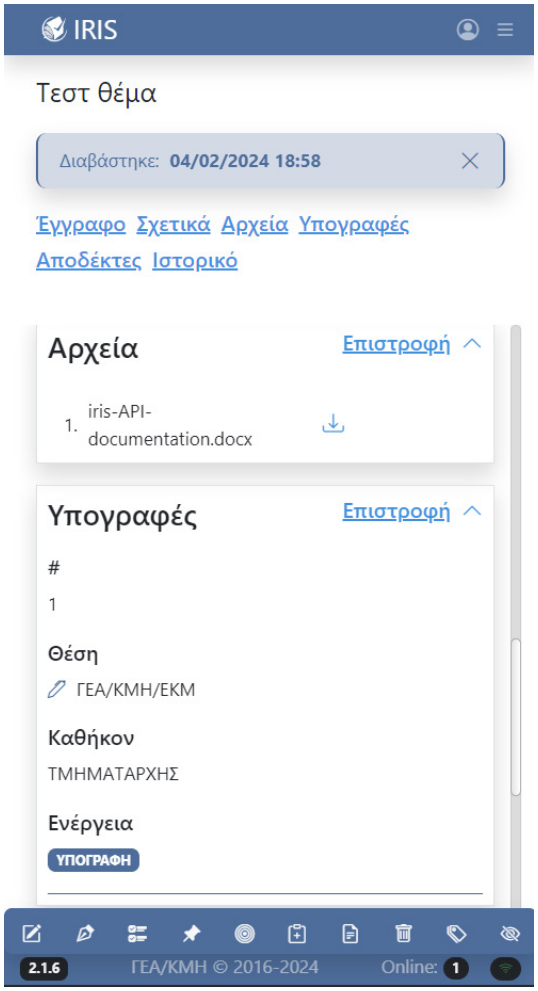


Εικόνα 18. Επισκόπηση νέου εγγράφου

1) Τίτλος Εγγράφου
2) Διαθέσιμες ενέργειες
3) Πληροφορίες ανάγνωσης, διανομής κλπ
4) Σύνδεσμοι για τις ενότητες της επισκόπησης.

7.9.5.2 Έγγραφο

Αποτελεί την 1η καρτέλα της επισκόπησης στην οποία παρουσιάζονται τα βασικά στοιχεία του εγγράφου.



Εικόνα 19. Οθόνη εγγράφου

7.9.5.3 Αρχεία.

Αποτελεί την 2η καρτέλα της επισκόπησης στην οποία εμφανίζονται τα αρχεία του εγγράφου(σχέδια εγγράφου), τα οποία μπορεί ο χρήστης είτε να δει μέσω της εφαρμογής (εφόσον ο τύπος τους είναι συμβατός, δηλαδή να είναι αρχεία κειμένου docx κλπ ή pdf) είτε να τα κατεβάσει τοπικά στον υπολογιστή του.

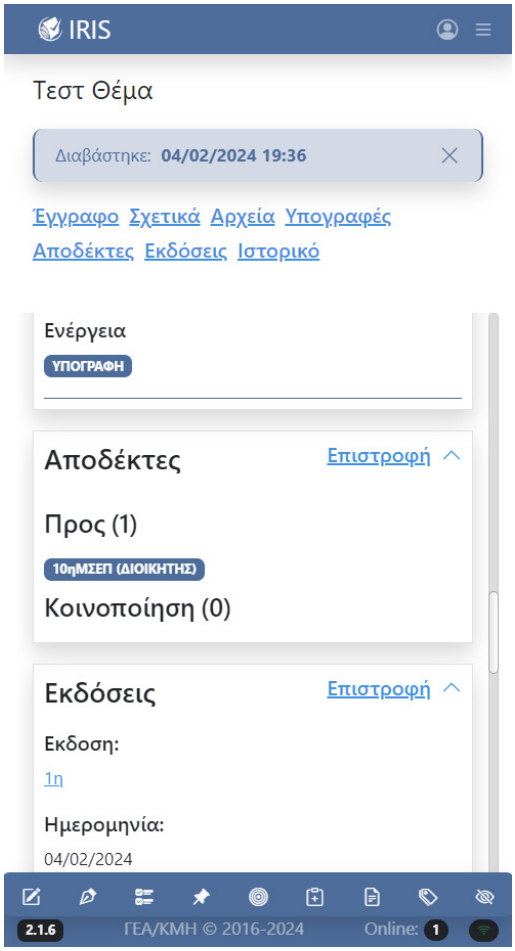
7.9.5.4 Υπογραφές.

Αποτελεί την 3η καρτέλα της επισκόπησης στην οποία εμφανίζεται η λίστα με τους υπογράφοντες τους εγγράφου της τρέχουσας έκδοσης ως εξής:

1. Ένδειξη για τον ρόλο στον οποίο βρίσκεται το έγγραφο για υπογραφή
2. Χαρακτηρισμός του υπογράφοντος (Για υπογραφή ή για Συντονισμό).
3. Κουμπί επιστροφής στην 1η καρτέλα της επισκόπησης.

7.9.5.5 Αποδέκτες.

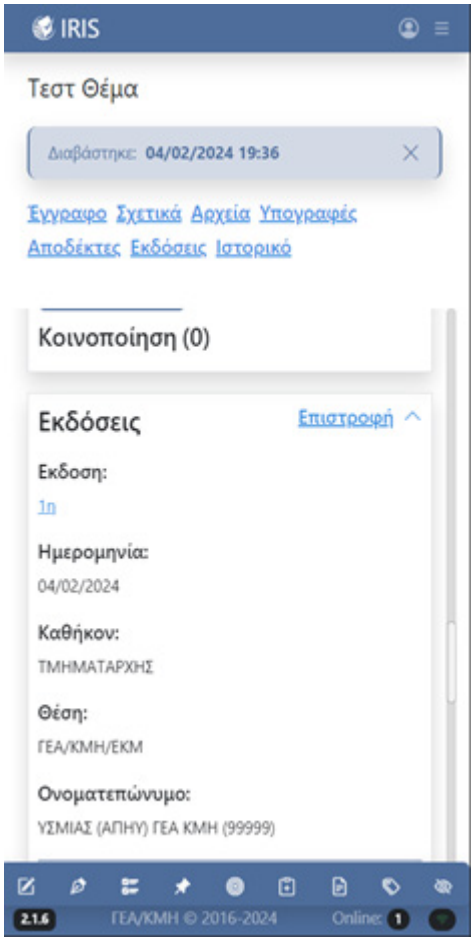
Αποτελεί την 4η καρτέλα της επισκόπησης στην οποία εμφανίζεται η λίστα με τους αποδέκτες του εγγράφου της τρέχουσας έκδοσης, με χρωματική αποτύπωση (εντός / εκτός συστήματος).



Εικόνα 20. Αποδέκτες

7.9.5.6 Εκδόσεις.

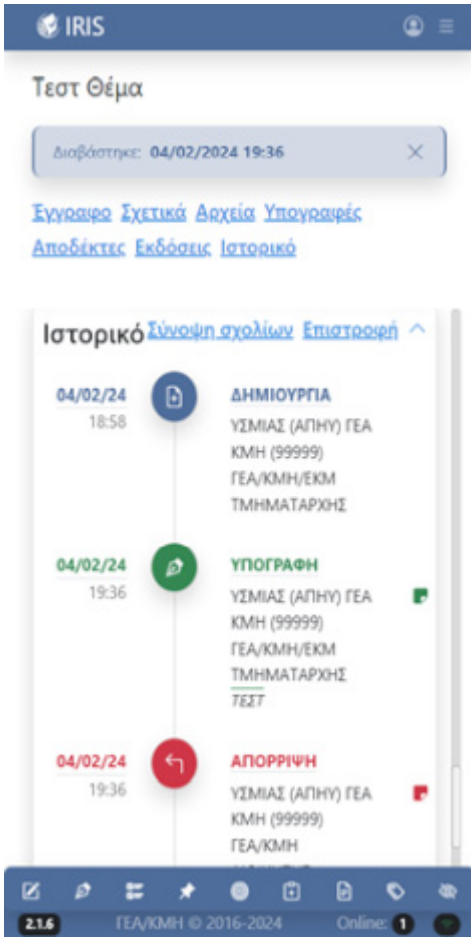
Αποτελεί την 5η καρτέλα της επισκόπησης στην οποία εμφανίζονται οι εκδόσεις του εγγράφου. Οι εκδόσεις δημιουργούνται αυτόματα κάθε φορά που κάποιος στην αλυσίδα των υπογραφών τροποποιεί ή απορρίπτει το έγγραφο. Για να περιηγηθούμε στις προηγούμενες εκδόσεις κάνουμε κλικ πάνω στον Α/Α της έκδοσης. Μετά την συλλογή όλων των υπογραφών το σύστημα δημιουργεί μία έκδοση Ακριβές Αντίγραφο όπου και επιτρέπεται στον συντάκτη να προβεί σε αλλαγές).



Εικόνα 21. Εκδόσεις

Ιστορικό.

Αποτελεί την 6η και τελευταία καρτέλα της επισκόπησης στην οποία αποτυπώνεται σε πραγματικό χρόνο η πορεία του εγγράφου και επιπλέον πληροφορίες όπως πότε έγινε κάποια ενέργεια (Δημιουργία, Υπογραφή, Απόρριψη, Επεξεργασία, Διανομή, Παραλαβή) , διάφορα σχόλια καθώς και ο Αριθμός Πρωτοκόλλου που έχει λάβει . Όλα τα προηγούμενα είναι ορατά μόνο από όσους είναι στην αλυσίδα υπογραφών.



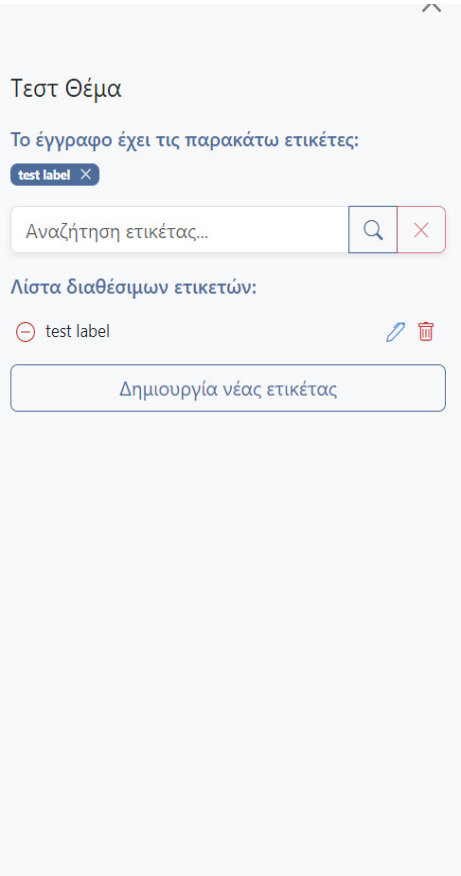
Εικόνα 22. Ιστορικό

7.9.6 Λοιπές Ενέργειες επί του Εγγράφου

Στην παρούσα ενότητα αναλύονται οι υπόλοιπες ενέργειες επί των εγγράφων που αναφέρθηκαν στην παράγραφο 7.9.4 του παρόντος.

7.9.6.1 Επεξεργασία ετικετών

Η παρούσα λειτουργικότητα [εικονίδιο 17] είναι διαθέσιμη σε όλους τους χρήστες της εφαρμογής και αφορά όλα τα έγγραφα στα οποία έχουν πρόσβαση. Σκοπό έχει το χαρακτηρισμό τους για λόγους οργάνωσης και αρχειοθέτησης. Επισημαίνονται τα ακόλουθα:

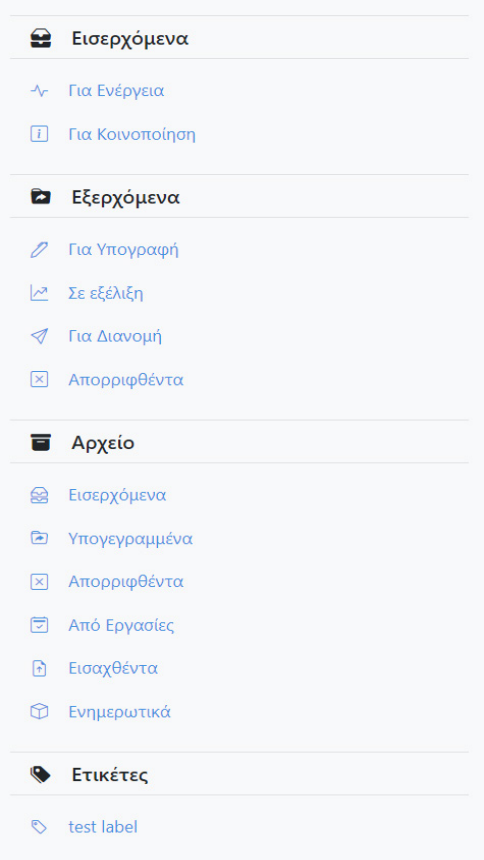


Εικόνα 23. Επεξεργασία ετικετών

✓ Η καταχώρηση γίνεται μέσα από το έγγραφο και είναι ανεξάρτητη της κατάστασης του εγγράφου.

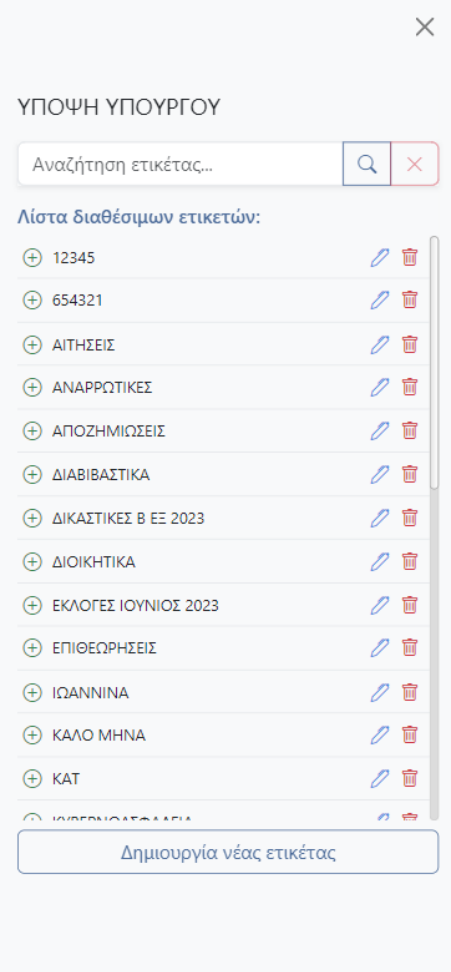
✓ Οι ετικέτες απεικονίζονται στις λίστες των εγγράφων, στην στήλη «Θέμα».

✓ Για κάθε ετικέτα δημιουργείται ξεχωριστό μενού.



Εικόνα 24. Ετικέτες και μενού ετικετών

✓ Χρήστες με τον ίδιο ρόλο (Θέση – Καθήκον) έχουν κοινές ετικέτες.



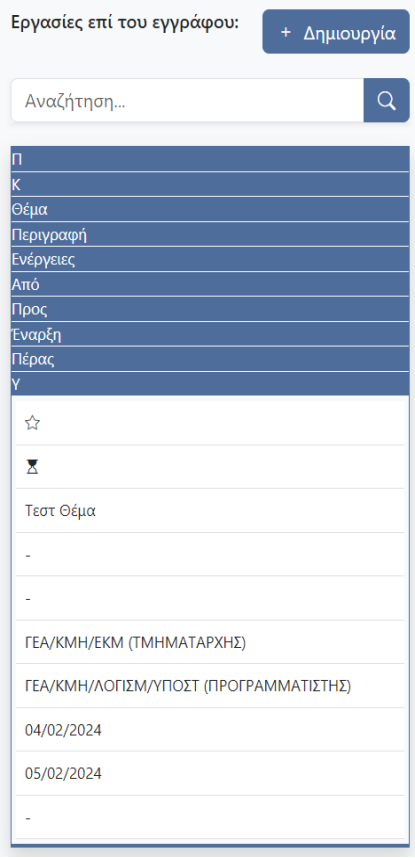
Εικόνα 25. Αναζήτηση ετικετών ή δημιουργία νέας ετικέτας

Η σελίδα των ετικετών περιλαμβάνει:

1. Θέμα Εγγράφου
2. Αναζήτηση ετικέτας από τη λίστα ετικετών.
3. Προσθήκη ετικέτας στο συγκεκριμένο έγγραφο. Εφόσον έχει ήδη προστεθεί αντίστοιχα αφαιρείται μόνο από αυτό το έγγραφο.
4. Επεξεργασία ονομασίας ετικέτας. Η νέα ονομασία τροποποιεί την ετικέτα και σε όλα με αυτή τα συνδεδεμένα έγγραφα.
5. Διαγραφή ετικέτας από όλα με αυτή τα συνδεδεμένα έγγραφα, καθώς και από τη λίστα ετικετών.
6. Δημιουργία νέας ετικέτας. Η νέα ετικέτα θα ενσωματωθεί στο έγγραφο και θα είναι διαθέσιμη στη λίστα των ετικετών.

7.9.6.2 Εργασίες Επί του Εγγράφου

Πατώντας το κουμπί [εικονίδιο 6] των εργασιών , μπορεί ο χρήστης να δει τις συνδεδεμένες με το έγγραφο εργασίες που τον αφορούν. Η ενέργεια μπορεί να γίνει μόνο μέσα από το έγγραφο. Στη συνέχεια μπορεί μέσα από την νέα οθόνη να δημιουργήσει νέα εργασία επί του εγγράφου.



Εικόνα 26. Συνδεδεμένες με το έγγραφο εργασίες που τον αφορούν

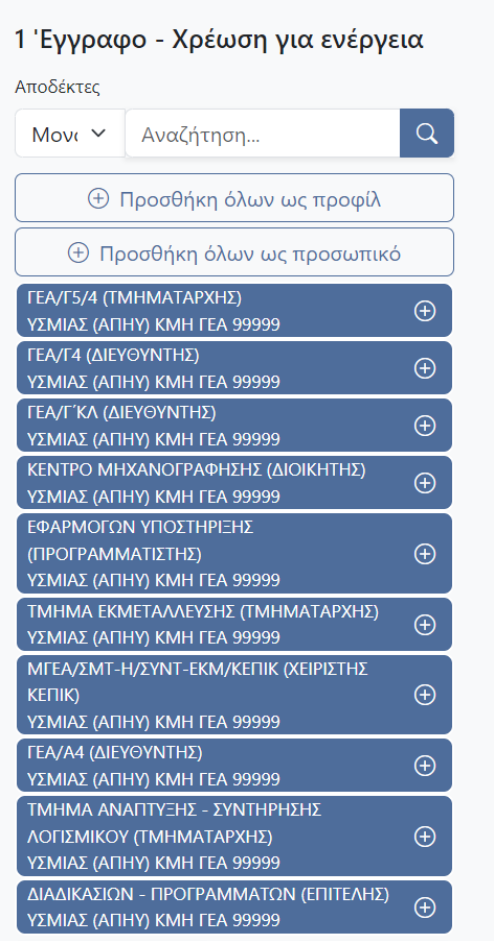
7.9.6.3 Προσθήκη σε Σημαντικό Θέμα

Πατώντας το κουμπί [εικονίδιο 7] της προσθήκης, μπορεί ο χρήστης να συνδέσει το έγγραφο με ένα υπάρχον ή νέο Σημαντικό Θέμα. Η ενέργεια μπορεί να γίνει μόνο μέσα από το έγγραφο. Παρόμοια λογική έχει και το κουμπί **Προσθήκη σε Στόχο** [8], για χρήστες που έχουν πρόσβαση στον Ωρίων.

Εικόνα 27. Προσθήκη σε Σημαντικό Θέμα

7.9.6.4 Χρέωση Εγγράφου για Ενέργεια

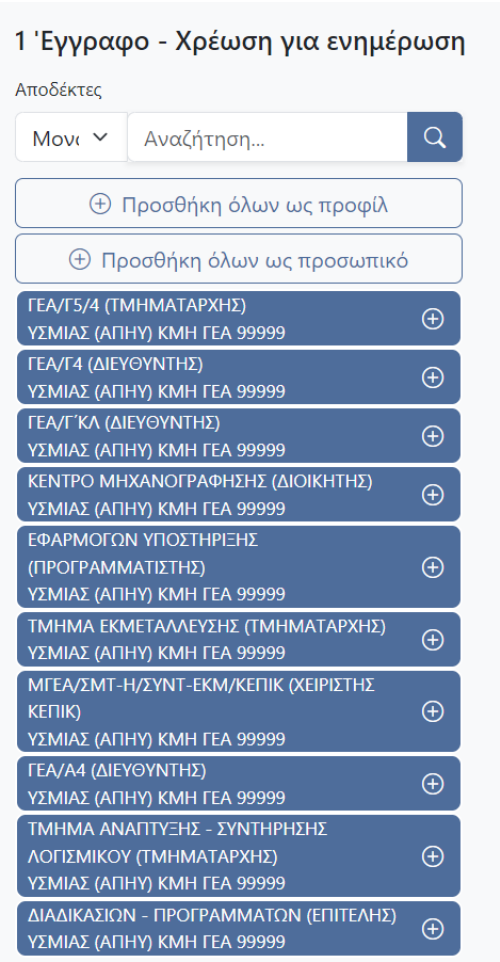
Η λειτουργία της χρέωσης εγγράφου για ενέργεια [εικονίδιο 9] είναι διαθέσιμη μόνο μέσω των μενού των εγγράφων: “Εισερχόμενα -> Για Ενέργεια” και “Εισερχόμενα -> Για Κοινοποίηση”. Ο χρήστης μπορεί να επιλέξει ένα ή περισσότερα έγγραφα από τη λίστα του και να τα χρεώσει σε έναν ή περισσότερους αποδέκτες. Για κάθε ένα έγγραφο και για κάθε έναν αποδέκτη θα δημιουργηθεί μια εργασία για Ενέργεια (από τον χρήστη προς τον αποδέκτη), η οποία θα έχει το έγγραφο ως σχετικό.



Εικόνα 28. Χρέωση για ενέργεια

7.9.6.5 Χρέωση Εγγράφου για Ενημέρωση

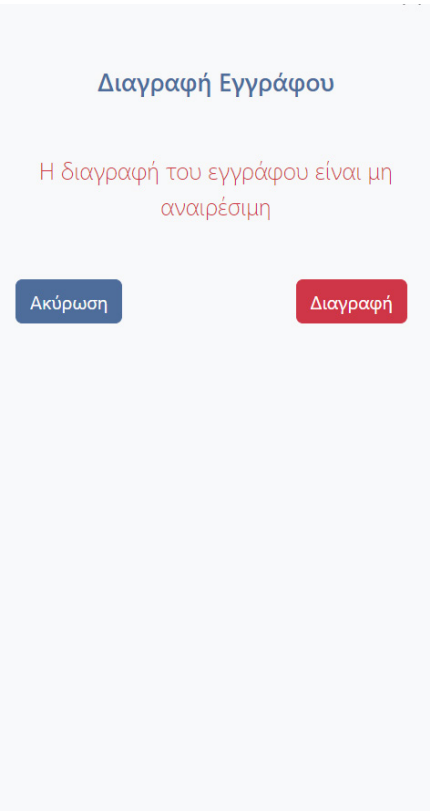
Η λειτουργία της χρέωσης εγγράφου για ενημέρωση[εικονίδιο 10] είναι διαθέσιμη μόνο μέσω των μενού των εγγράφων: “Εισερχόμενα -> Για Ενέργεια” και “Εισερχόμενα -> Για Κοινοποίηση”. Ο χρήστης μπορεί να επιλέξει ένα ή περισσότερα έγγραφα από τη λίστα του και να τα χρεώσει σε έναν ή περισσότερους αποδέκτες. Για κάθε ένα έγγραφο και για κάθε έναν αποδέκτη θα δημιουργηθεί μια εργασία για Ενημέρωση (από τον χρήστη προς τον αποδέκτη), η οποία θα έχει το έγγραφο ως σχετικό.



Εικόνα 29. Χρέωση Εγγράφου για Ενημέρωση

7.9.6.6 Διαγραφή Εγγράφου

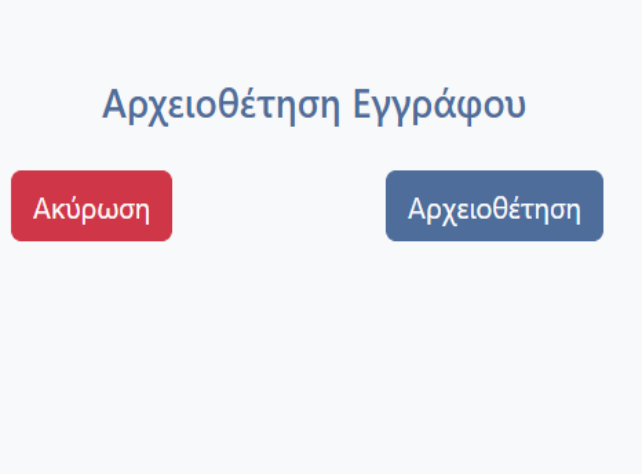
Η διαγραφή του εγγράφου [εικονίδιο 11] παρέχεται μόνο στον συντάκτη του, κατά την πρώτη του έκδοση και εφόσον δεν έχει υπογράψει ούτε αυτός το έγγραφο. *Παρατήρηση:* Τα διεγραμμένα έγγραφα δεν εμπεριέχονται σε κάποιο μενού και δεν δύναται να ανακτηθούν.



Εικόνα 30. Διαγραφή εγγράφου

7.9.6.7 Αρχειοθέτηση Εγγράφου

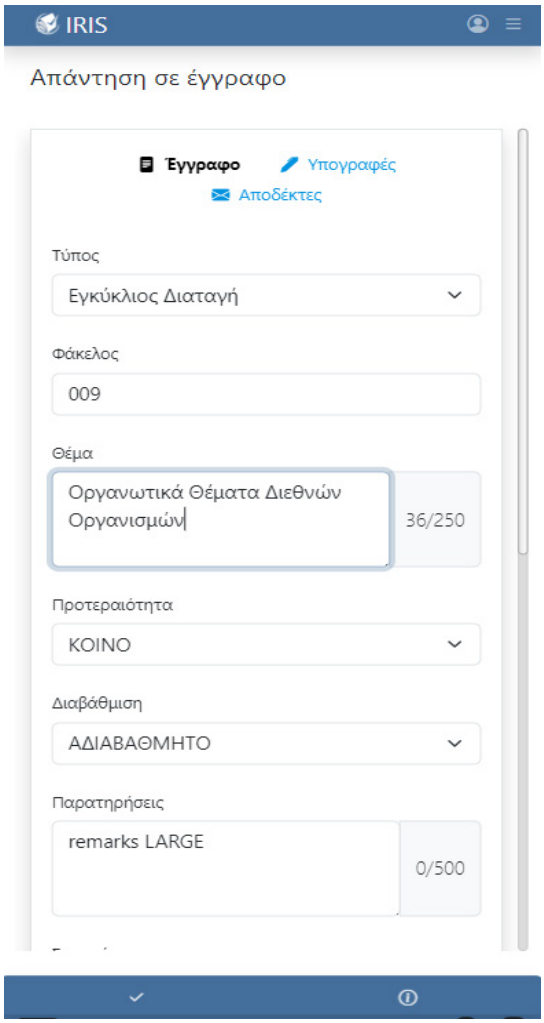
Η λειτουργία της αρχειοθέτησης του εγγράφου [εικονίδιο 12] είναι διαθέσιμη είτε μέσα από το έγγραφο είτε μέσω των μενού των εγγράφων: “Εισερχόμενα -> Για Ενέργεια” και “Εισερχόμενα -> Για Κοινοποίηση”. Ο χρήστης μπορεί να επιλέξει ένα ή περισσότερα έγγραφα από τη λίστα του (ή μεμονωμένα) και να τα αρχειοθετήσει. Η ενέργεια αυτή είναι μη αναιρέσιμη και μεταβαίνει τα έγγραφα στο μενού των εγγράφων “Αρχείο-> Εισερχόμενα”.



Εικόνα 31. Αρχειοθέτηση εγγράφου

7.9.6.8 Απάντηση

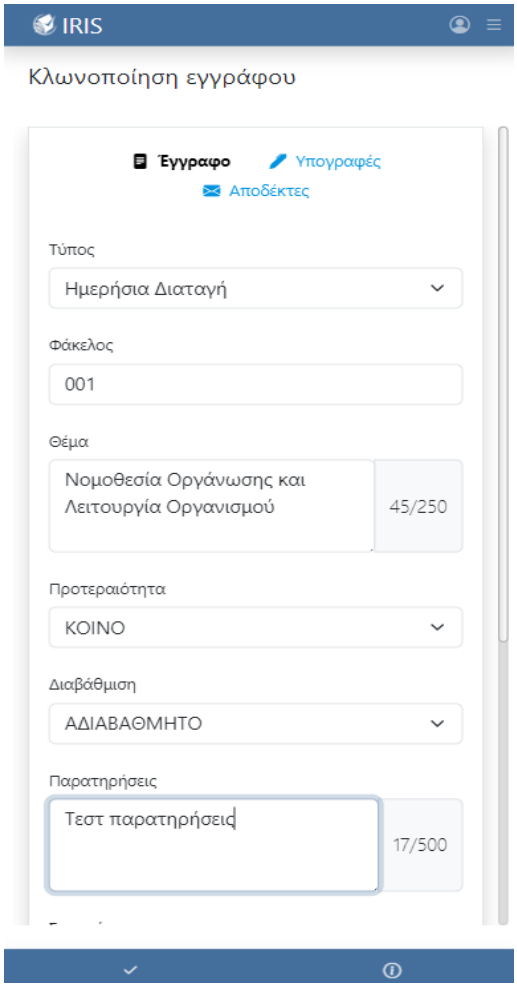
Η λειτουργία της απάντησης επί εγγράφου [εικονίδιο 13] είναι διαθέσιμη μόνο μέσα από ένα εισερχόμενο έγγραφο (αρχειοθετημένο ή μη). Η εκτέλεση της ενέργειας αυτής δημιουργεί ένα νέο έγγραφο που έχει ως αποδέκτη για Ενέργεια τον αποστολέα του αρχικού εγγράφου και ως α. σχετικό το αρχικό έγγραφο(μαρκαρισμένο ως απάντηση). Με την αποθήκευση του απαντητικού εγγράφου εκκινεί η διαδικασία σύνταξης ενός νέου εγγράφου. Με την ολοκλήρωση της ροής του τα δύο έγγραφα συσχετίζονται και μέσα των βιβλίων πρωτοκόλλων τους (βλ. Ενότητα Πρωτόκολλο).



Εικόνα 32. Απάντηση σε έγγραφο

7.9.6.9 Κλωνοποίηση Εγγράφου.

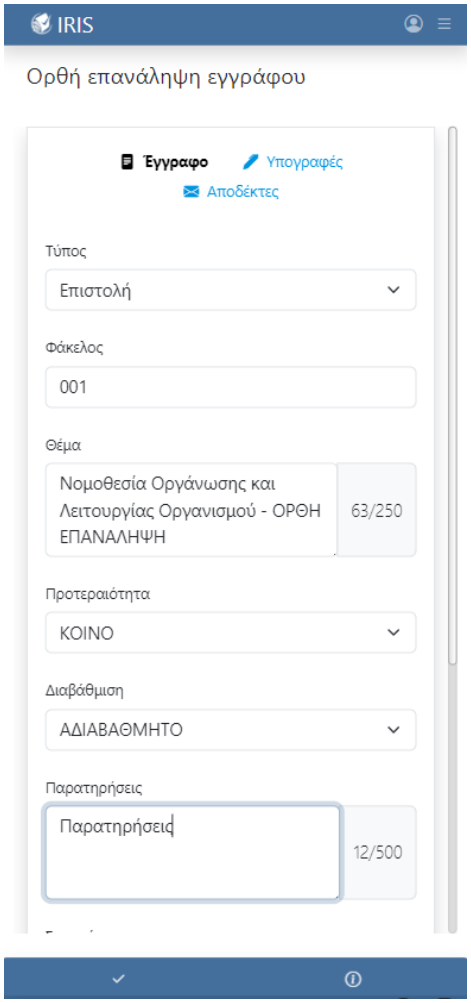
Η λειτουργία της Κλωνοποίησης [εικονίδιο 14] είναι διαθέσιμη μόνο μέσα από ένα εξερχόμενο έγγραφο. Η εκτέλεση της ενέργειας αυτής δημιουργεί ένα νέο έγγραφο έχοντας αντιγράψει όλα τα δεδομένα του αρχικού εγγράφου (εκτός από τα σχέδια Εγγράφου). Με την αποθήκευση του κλωνοποιημένου εγγράφου εκκινεί η διαδικασία σύνταξης ενός νέου εγγράφου. Με την ολοκλήρωση της ροής τα δύο έγγραφα δεν συσχετίζονται.



Εικόνα 33. Κλωνοποίηση εγγράφου

7.9.6.10 Ορθή Επανάληψη

Η λειτουργία της Ορθής Επανάληψης [εικονίδιο 15] είναι διαθέσιμη μόνο μέσα από ένα εξερχόμενο έγγραφο (Διανεμημένο). Η εκτέλεση της ενέργειας αυτής δημιουργεί ένα νέο έγγραφο έχοντας αντιγράψει όλα τα δεδομένα του αρχικού εγγράφου (εκτός από τα σχέδια Εγγράφου) με προσθήκη του λεκτικού “ΟΡΘΗ ΕΠΑΝΑΛΗΨΗ” στο θέμα και έχοντας ως α. σχετικό το αρχικό έγγραφο(μαρκαρισμένο ως ΟΕ). Με την αποθήκευση του απαντητικού εγγράφου εκκινεί η διαδικασία σύνταξης ενός νέου εγγράφου. Με την ολοκλήρωση της ροής, το νέο έγγραφο θα πρωτοκολληθεί εκ νέου και τα δύο έγγραφα θα συσχετίζονται αμφίδρομα μέσω των βιβλίων πρωτοκόλλων τους.



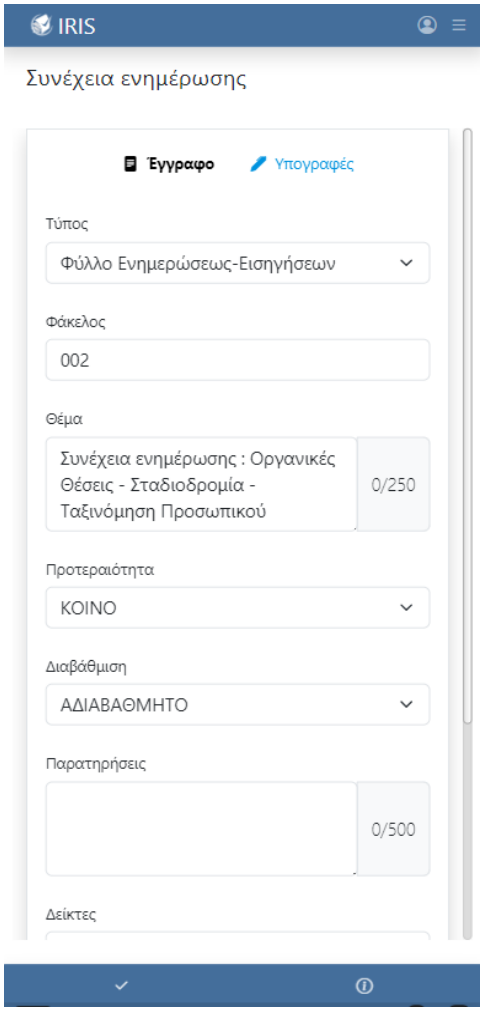
Εικόνα 34. Ορθή επανάληψη

7.9.6.11 Μετάβαση στο Ενημερωτικό

Η Μετάβαση στο Ενημερωτικό [16] είναι διαθέσιμη μόνο μέσα από κάποιο Συνοδευτικό Έγγραφο ενός Ενημερωτικού Εγγράφου. Η εκτέλεση της ενέργειας αυτής μεταφέρει τον χρήστη στην οθόνη του ενημερωτικού.

7.9.6.12 Συνέχεια Ενημέρωσης

Η λειτουργία της Συνέχειας Ενημέρωσης [εικονίδιο 18] είναι διαθέσιμη μόνο μέσα από ένα ολοκληρωμένο Ενημερωτικό έγγραφο (Αρχείο -> Ενημερωτικά). Η εκτέλεση της ενέργειας αυτής δημιουργεί ένα νέο ενημερωτικό έγγραφο έχοντας αντιγράψει όλα τα δεδομένα του αρχικού εγγράφου (εκτός από τα σχέδια Εγγράφου και τους υπογράφοντες) με προσθήκη του λεκτικού “Συνέχεια ενημέρωσης ” στο θέμα και έχοντας ως τελευταίο Δείκτη το αρχικό έγγραφο. Με την αποθήκευση του νέου εγγράφου εκκινεί η διαδικασία σύνταξης ενός νέου ενημερωτικού εγγράφου.



Εικόνα 35. Συνέχεια ενημέρωσης

7.9.6.13 Επισήμανση Εγγράφου ως μη αναγνωσμένο

Η εν λόγω λειτουργικότητα [19] είναι διαθέσιμη μόνο για τα έγγραφα που ανοίγονται μέσω των μενού “Εισερχόμενα” και “Εξερχόμενα-> Για υπογραφή”. Η εκτέλεση της ενέργειας αυτής θέτει το έγγραφο ως μη αναγνωσμένο για τον χρήστη και αυξάνει την αριθμητική πληροφορία των μη αναγνωσμένων εγγράφων του αντίστοιχου μενού κατά ένα.

Εικόνα 36. Επισήμανση Εγγράφου ως μη αναγνωσμένο

7.9.6.14 Εισαγωγή Εγγράφου

Η Εισαγωγή Εγγράφου χρησιμοποιείται για να εισάγουμε ένα έγγραφο- που δημιουργήθηκε εκτός συστήματος - στο Ίριδα. Η διαδικασία προσομοιάζει αυτήν της δημιουργίας νέου εγγράφου, ωστόσο με τις εξής επισημάνσεις:

✓ Δεν υπάρχει καρτέλα Υπογραφών, αφού το έγγραφο έχει υπογραφεί ήδη και αποσταλεί στον χρήστη.

✓ Θα πρέπει στην καρτέλα έγγραφο να εισάγουμε τον Αριθμός Εγγράφου με τον οποίο εκδόθηκε το έγγραφο. (Αρ. Πρωτοκόλλου)

✓ Προαιρετικά (εφόσον υπάρχει) εισάγεται και ο αριθμός Σχεδίου.

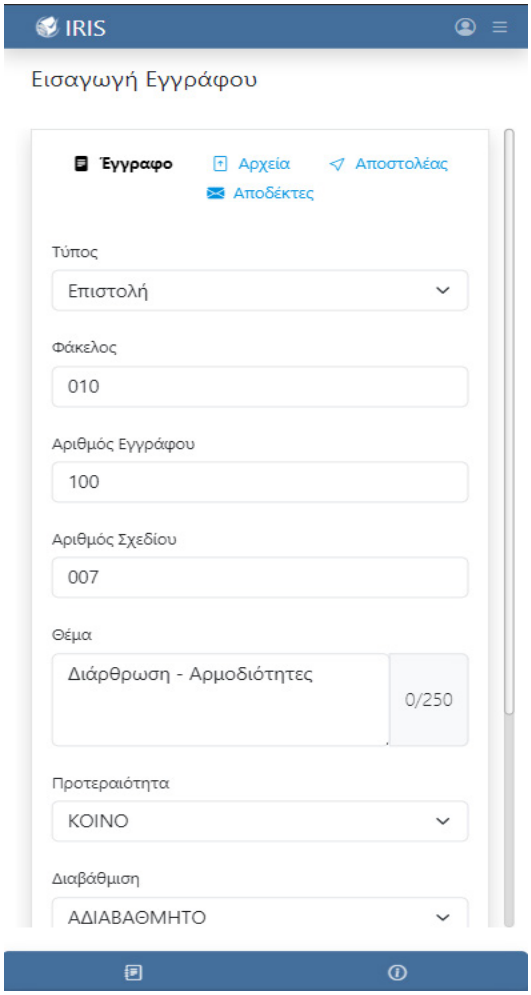
✓ Θα πρέπει να εισαχθεί τουλάχιστον ένα αρχείο στο πεδίο Αρχεία προς Εισαγωγή.

✓ Στην καρτέλα Αποστολέας εισάγεται μία επαφή του χρήστη (ή νέα επαφή αν δεν υπάρχει).

✓ Η Διανομή μπορεί να γίνει μόνο εντός της Μονάδας του χρήστη.

✓ Με την συμπλήρωση όλων των απαραίτητων πεδίων, ο χρήστης πρέπει να πατήσει το κουμπί Πρωτοκόλλησης και το έγγραφο θα μεταφερθεί στα έγγραφα Για Διανομή του.

✓ Τα έγγραφα που εισάγονται στο σύστημα, συμπεριφέρονται όπως και τα συστημικά έγγραφα και διαχωρίζονται.



Εικόνα 37. Εισαγωγή εγγράφου

7.9.6.15 Νέο Ενημερωτικό

Στην παρούσα έκδοση της εφαρμογής υπάρχει μια ξεχωριστή κατηγορία εγγράφων που αναφέρονται ως Ενημερωτικά και χρησιμοποιούνται για την καλύτερη επιτελική οργάνωση ενός τμήματος ή διεύθυνσης. Η διαδικασία διακίνησης Ενημερωτικών Εγγράφων έχει ως εξής:

✓ Το ενημερωτικό διακινείται και μέσω αυτού τα συνοδευτικά έγγραφα, εφόσον υφίστανται.

✓ Τα συνοδευτικά έγγραφα δημιουργούνται μέσω του ενημερωτικού και δεν έχουν ξεχωριστό πίνακα υπογραφών.

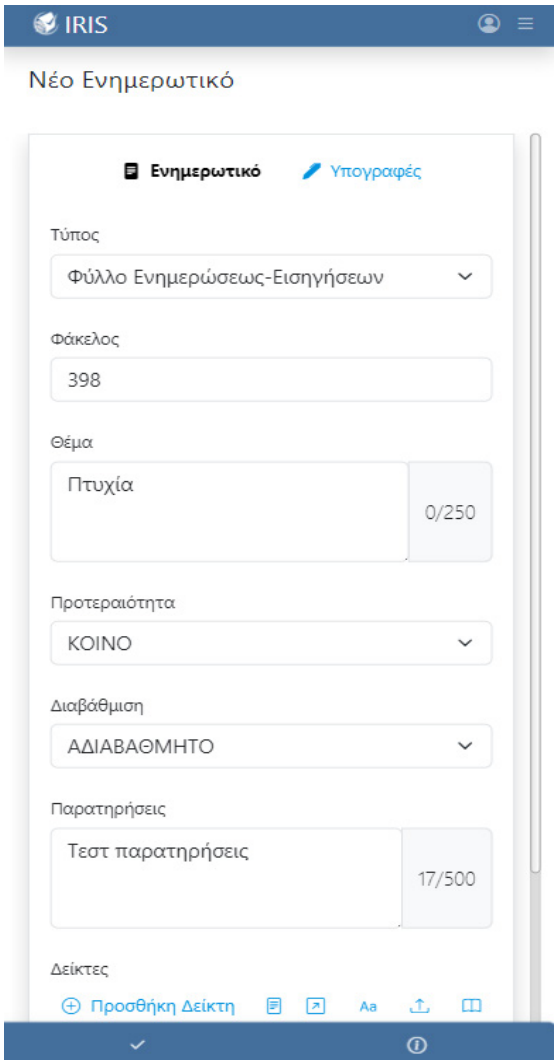
✓ Η υπογραφή / απόρριψη στο ενημερωτικό συνεπάγεται και υπογραφή / απόρριψη του συνόλου των συνοδευτικών.

✓ Ένας υπογράφων του ενημερωτικού δεν είναι απαραίτητο να υπογράφει το σύνολο των συνοδευτικών.

✓ Η υπογραφή του τελικού υπογράφοντος του ενημερωτικού, μεταφέρει τα συνοδευτικά έγγραφα στο μενού «Εξερχόμενα -> Για Διανομή» του συντάκτη και το ίδιο το ενημερωτικό στο «Αρχείο-> Ενημερωτικά» των υπογραφόντων.

✓ Στη συνέχεια, τα συνοδευτικά διανέμονται όπως και τα υπόλοιπα έγγραφα.

✓ Τα ενημερωτικά δεν έχουν πίνακα αποδεκτών, δεν διανέμονται και πρωτοκολλούνται αυτόματα με την υπογραφή του τελευταίου υπογράφοντος.



Εικόνα 38. Νέο ενημερωτικό

7.9.7 Λίστες Εγγράφων

Οι λίστες των εγγράφων στο Ίριδα (ανεξάρτητα το μενού) έχουν πανομοιότυπη μορφή με μικρές διαφορές στις στήλες, ανά περίπτωση. Η λειτουργικότητά τους αναλύεται παρακάτω:

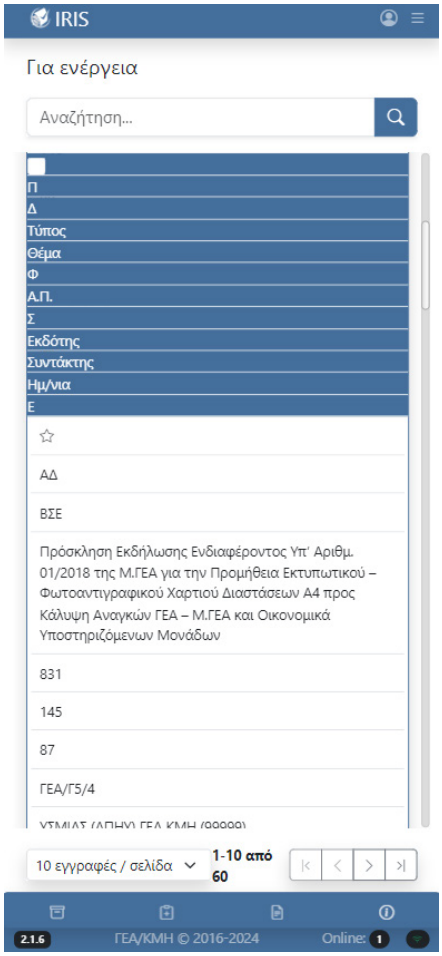
1. Πεδίο Αναζήτησης εγγράφων. Η αναζήτηση επιδρά σε όλες τις στήλες (και στις ετικέτες). Η αναζήτηση ημερομηνίας μπορεί να γραφεί με διάφορους έγκυρους τρόπους. Π.χ για αναζήτηση εγγράφου με ημερομηνία 14/04/2022, μπορούμε να

πληκτρολογήσουμε ένα από τα εξής: 14/04/2022, 14/4/22, 14-04-2022, 14-4-22. Επίσης στην αναζήτηση λεκτικών έχει σημασία ο τονισμός αλλά δεν έχει αν θα γράψουμε τη λέξη με πεζά ή κεφαλαία γράμματα.
2. Κουμπί Αναζήτησης (εναλλακτικά μπορούμε να πατήσουμε το πλήκτρο “Enter”).
3. Οι στήλες της λίστας. Μπορούμε να πατήσουμε πάνω στην κεφαλίδα κάθε στήλης για να ταξινομήσουμε τη λίστα με αύξουσα ή φθίνουσα σειρά.
4. Έγγραφα της λίστας τα οποία δεν έχει αναγνώσει ο χρήστης έχουν ένα γκρίζο background.
5. Στις λίστες των εισερχομένων υπάρχει και η στήλη Ε (εργασίες). Έγγραφα με καμία συνδεδεμένη εργασία δεν έχουν κάποια πληροφορία, έγγραφα με όλες τις συνδεδεμένες εργασίες τους ολοκληρωμένες έχουν ένα πράσινο εικονίδιο, ενώ έγγραφα με έστω και μια συνδεδεμένη εργασία ανολοκλήρωτη έχουν ένα κίτρινο εικονίδιο. Πηγαίνοντας το ποντίκι πάνω στο εικονίδιο εμφανίζεται ο λόγος των ολοκληρωμένων προς των συνολικών εργασιών, καθώς και το ποσοστό ολοκλήρωσης. Η στήλη είναι ταξινομήσιμη ως προς το ποσοστό.

7.9.7.1 Εισερχόμενα / Για Ενέργεια

Στη λίστα εισερχομένων εγγράφων για ***ενέργεια*** υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου
- Εργασίες επί του εγγράφου

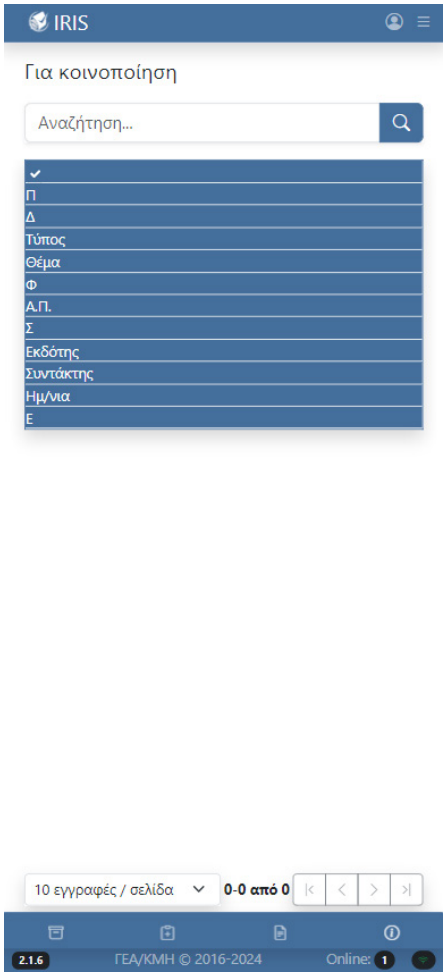


Εικόνα 39. Λίστα εγγράφων για ενέργεια

7.9.7.2 Εισερχόμενα / Για Κοινοποίηση

Στη λίστα εισερχομένων εγγράφων για ***κοινοποίηση*** υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου
- Εργασίες επί του εγγράφου

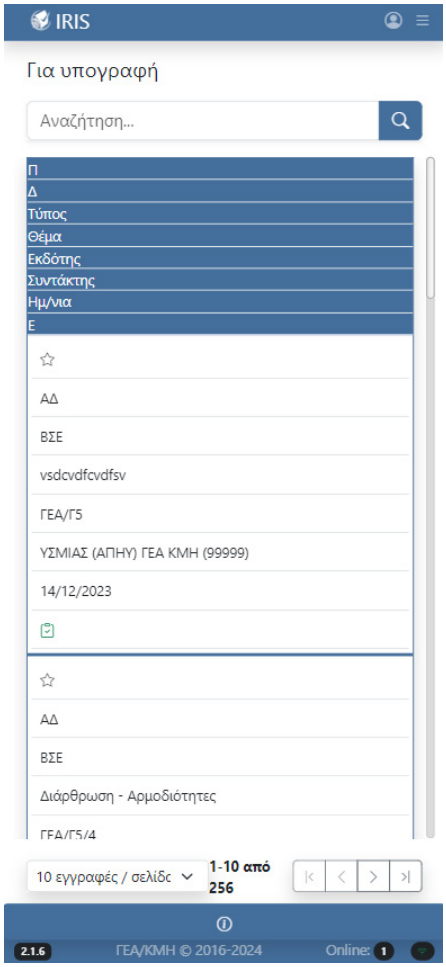


Εικόνα 40. Εισερχόμενα / για κοινοποίηση

7.9.7.3 Εξερχόμενα / Για Υπογραφή

Στη λίστα εξερχομένων εγγράφων για ***υπογραφή*** υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου
- Εργασίες επί του εγγράφου

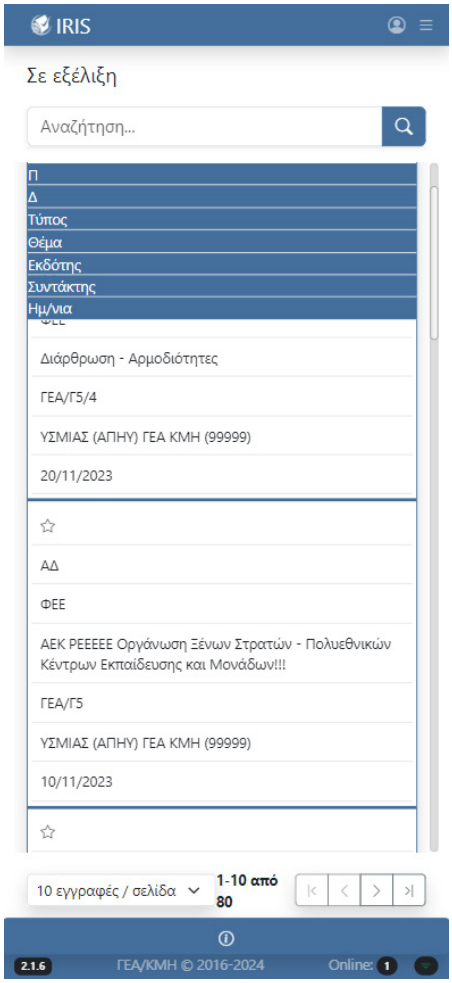


Εικόνα 41. Εξερχόμενα/για υπογραφή

7.9.7.4 Εξερχόμενα / Σε εξέλιξη

Στη λίστα εξερχομένων εγγράφων σε ***εξέλιξη*** υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

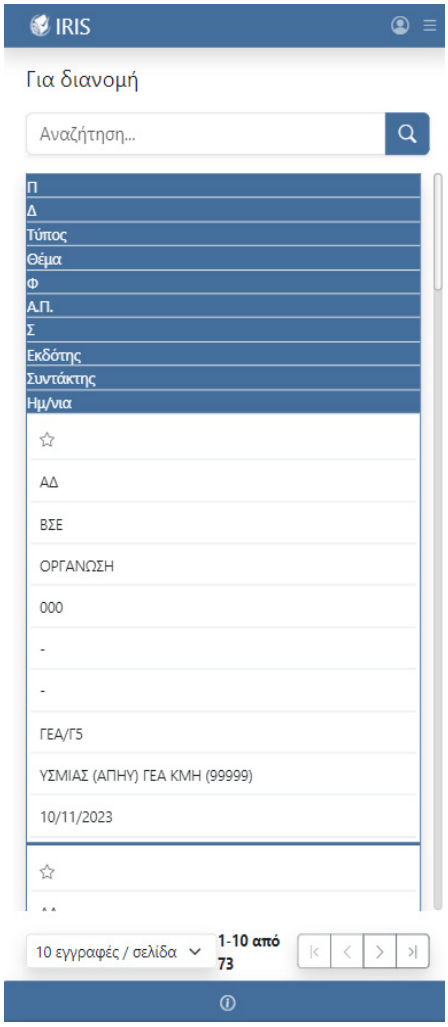


Εικόνα 42. Εξερχόμενα / σε εξέλιξη

7.9.7.5 Εξερχόμενα / Για διανομή

Στη λίστα εξερχομένων εγγράφων για ***διανομή*** υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

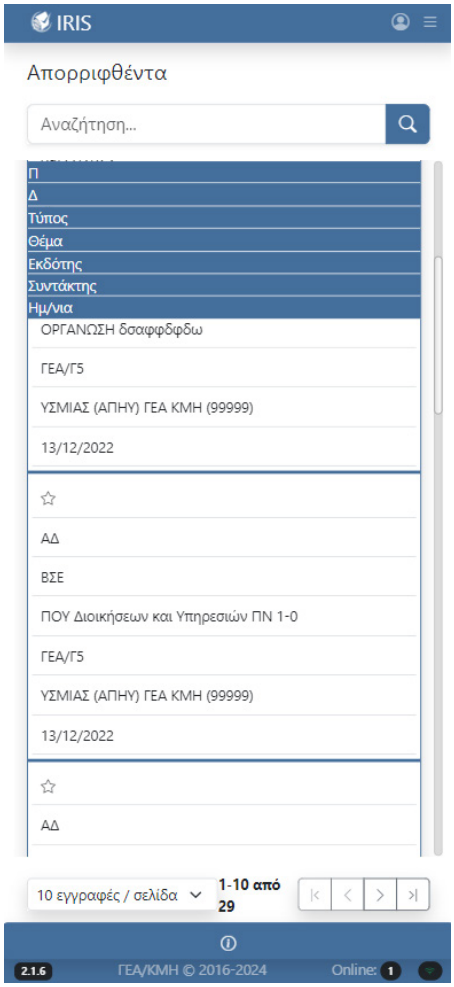


Εικόνα 43. Εξερχόμενα/για διανομή

7.9.7.6 Εξερχόμενα / Απορριφθέντα

Στη λίστα εξερχομένων ***απορριφθέντων*** εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

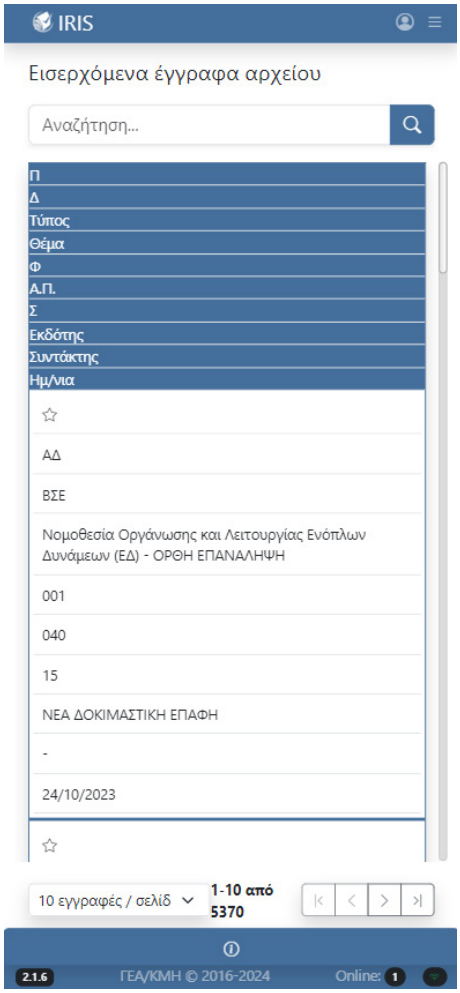


Εικόνα 44. Εξερχόμενα/απορριφθέντα

7.9.7.7 Αρχείο / Εισερχόμενα

Στη λίστα αρχείου ***εισερχομένων*** εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

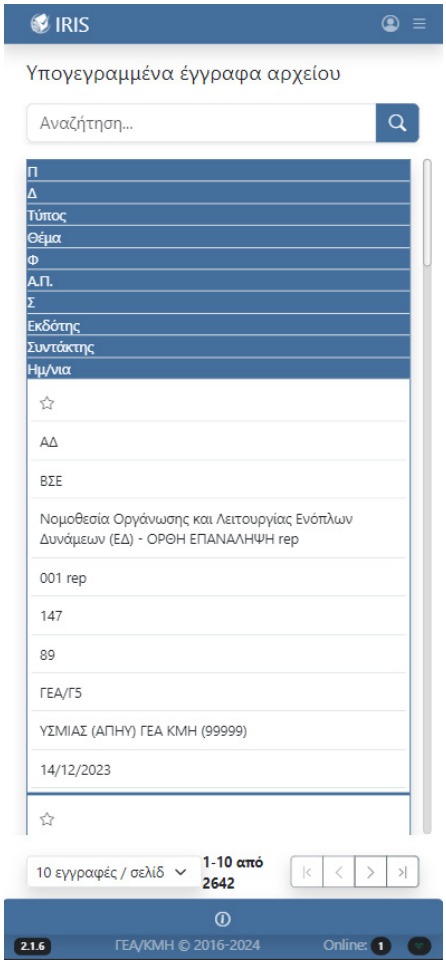


Εικόνα 45. Αρχείο/εισερχόμενα

7.9.7.8 Αρχείο / Υπογεγραμμένα

Στη λίστα αρχείου *υπογεγραμμένων* εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

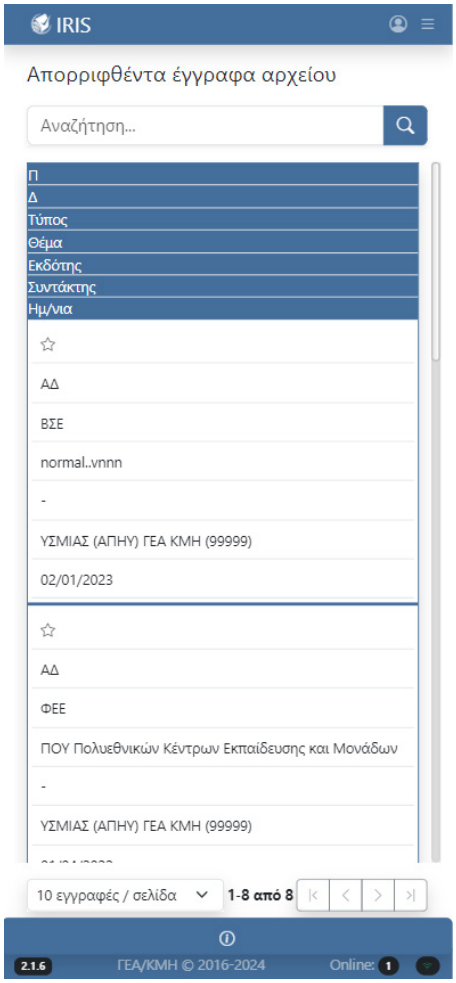


Εικόνα 46. Αρχείο/υπογεγραμμένα

7.9.7.9 Αρχείο / Απορριφθέντα

Στη λίστα αρχείου ***απορριφθέντων*** εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

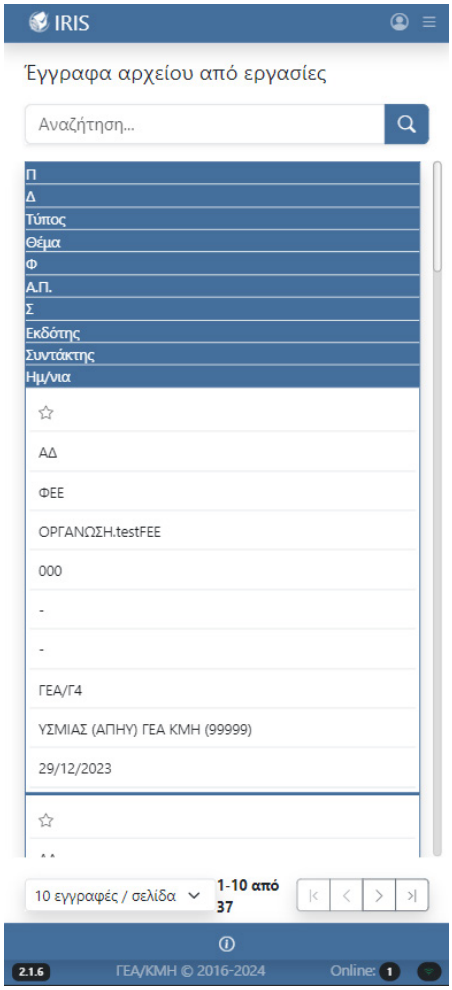


Εικόνα 47. Αρχείο/απορριφθέντα

7.9.7.10 Αρχείο / Από Εργασίες

Στη λίστα αρχείου *από εργασίες* εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Φάκελος εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Αριθμός σχεδίου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

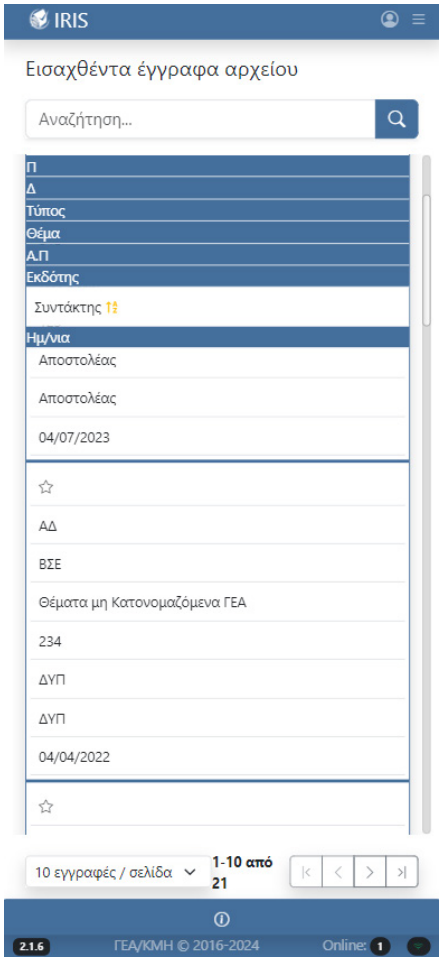


Εικόνα 48. Αρχείο/από εργασίες

7.9.7.11 Αρχείο / Εισαχθέντα

Στη λίστα αρχείου ***εισαχθέντων*** εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Αριθμός Πρωτοκόλλου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου

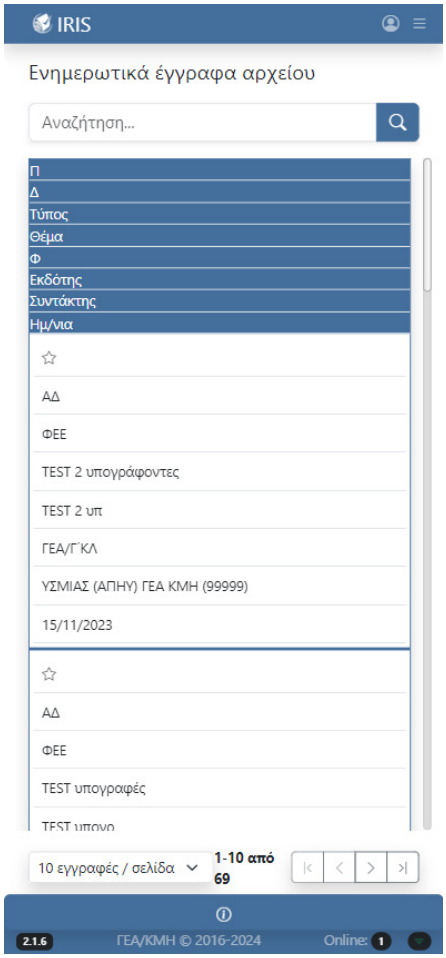


Εικόνα 49. Αρχείο/εισαχθέντα

7.9.7.12 Αρχείο / Ενημερωτικά

Στη λίστα αρχείου ***ενημερωτικών*** εγγράφων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εγγράφου
- Διαβάθμιση εγγράφου
- Τύπος εγγράφου
- Θέμα εγγράφου
- Αριθμός φακέλου εγγράφου
- Εκδότης εγγράφου
- Συντάκτης εγγράφου
- Ημερομηνία έκδοσης εγγράφου



Εικόνα 50. Αρχείο/ενημερωτικά

7.10 Εργασίες

7.10.1 Αρχική Σελίδα Ενότητας

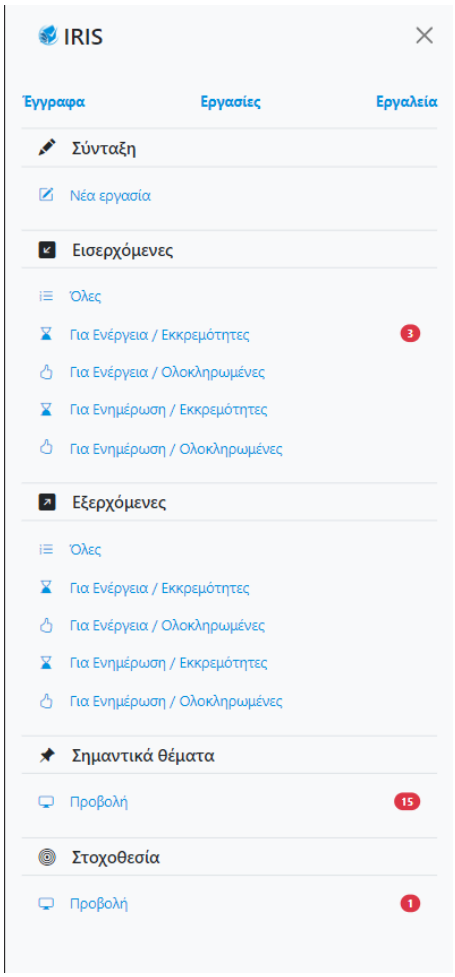
Στην αρχική σελίδα της ενότητας των εργασιών αποτυπώνονται αριθμητικές πληροφορίες για το σύνολο των εργασιών του χρήστη, ομαδοποιημένες ανάλογα την κατάσταση τους και διαχωρισμένες σε Εξερχόμενες και Εισερχόμενες.

7.10.2 Μενού Ενότητας

Στο μενού της ενότητας των εργασιών παρέχονται οι εξής επιλογές:

- Σύνταξη
    - Νέα εργασία
- Εισερχόμενες
    - Όλες
    - Για Ενέργεια / Εκκρεμότητες
    - Για Ενέργεια / Ολοκληρωμένες
    - Για Ενημέρωση / Εκκρεμότητες

    - Για Ενημέρωση / Ολοκληρωμένες
- Εξερχόμενες
    - Όλες
    - Για Ενέργεια / Εκκρεμότητες
    - Για Ενέργεια / Ολοκληρωμένες
    - Για Ενημέρωση / Εκκρεμότητες
    - Για Ενημέρωση / Ολοκληρωμένες
- Σημαντικά Θέματα
    - Προβολή
- Ωρίων / Στοχοθεσία
    - Προβολή

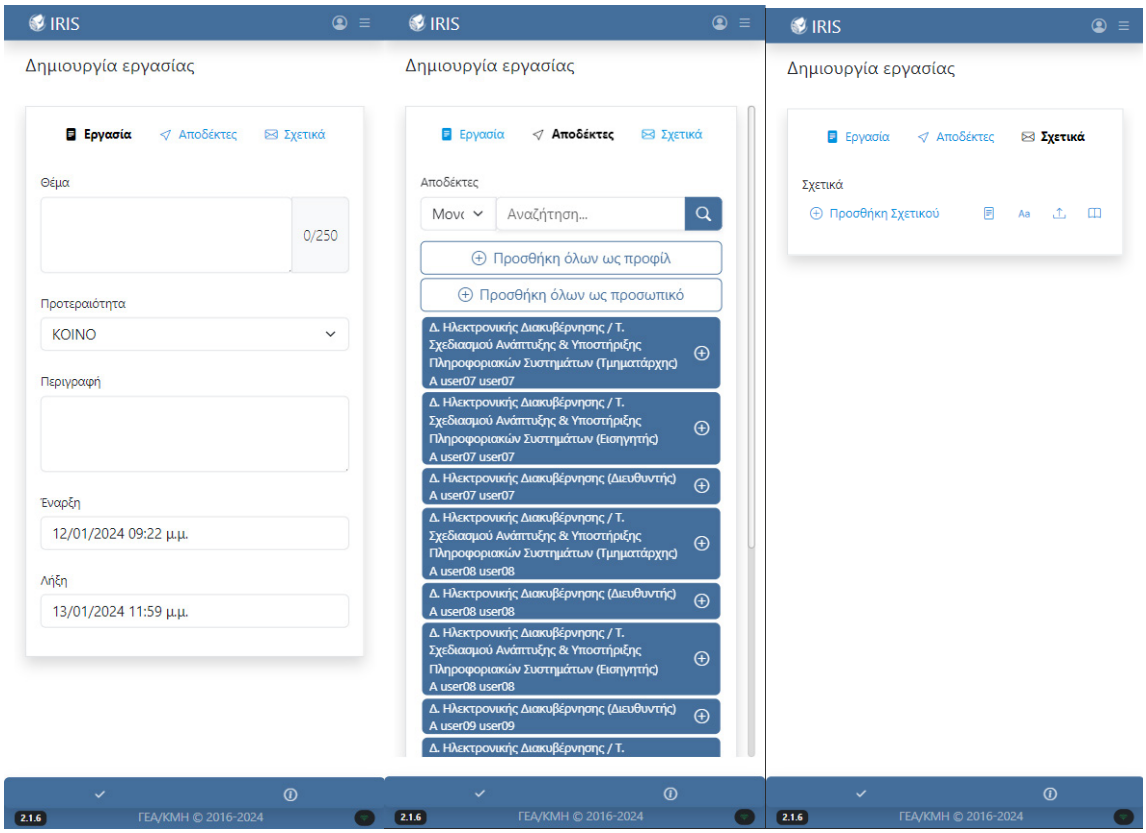


Εικόνα 51. Μενού ενότητας εργασιών

7.10.3 Νέα Εργασία

Εργασία είναι ένας τρόπος χρέωσης (μεταβίβασης δικαιωμάτων) επί ενός ή και περισσοτέρων εγγράφων. Παρόλα αυτά, μια εργασία δεν είναι απαραίτητο να συνδέεται με κάποιο έγγραφο.

Η δημιουργία νέας εργασίας χωρίζεται σε 3 καρτέλες ως ακολούθως: Η αποθήκευση γίνεται και για τις 3 καρτέλες και ολοκληρώνει τη διαδικασία της νέας εργασίας.



Εικόνα 52. Δημιουργία εργασίας

7.10.3.1 Βασικά Στοιχεία της Εργασίας

Κάνοντας κλικ στη Νέα Εργασία, μεταφερόμαστε στην παραπάνω σελίδα όπου και συμπληρώνουμε τα στοιχεία της εργασίας.

7.10.3.2 Αποδέκτες της Εργασίας

Κάθε εργασία πρέπει να έχει τουλάχιστον ένα Αποδέκτη. Για κάθε αποδέκτη δημιουργείται μια εξερχόμενη εργασία από το προφίλ του συντάκτη προς αυτόν (τον αποδέκτη). Οι δυνατότητες επιλογής αποδεκτών αναλύονται ως ακολούθως:

1. Παρέχονται δύο τρόποι αναζήτησης Αποδέκτη:

    i. Όλα. Αποδέκτες που ανήκουν στον φορέα του συντάκτη.

    ii. Μονάδα. Αποδέκτες που ανήκουν στην Μονάδα του συντάκτη. Παρατήρηση: Διαφοροποίηση υφίσταται μόνο για τα επιτελεία όπου φορέας είναι π.χ το ΓΕΑ ενώ μονάδες οι διευθύνσεις (ΓΕΑ/Α1, ΓΕΑ/Γ5 κλπ).

2. Αναζήτηση Αποδέκτη .

3. Επιλογή του αποδέκτη προσωπικά. Αυτό γίνεται όταν η εργασία αφορά προσωπικά τον αποδέκτη και όχι άλλους με τους οποίους μοιράζεται ενδεχομένως το ίδιο προφίλ. Ο αποδέκτης έχει πρόσβαση μόνο για όσο έχει το συγκεκριμένο προφίλ.

4. Επιλογή του αποδέκτη ως προφίλ. Την εργασία την “βλέπουν” όσοι χρήστες έχουν πρόσβαση στο συγκεκριμένο προφίλ.

5. Τα υπόλοιπα στοιχεία της οθόνης (σελιδοποίηση, επιλογή και προσθαφαίρεση αποδεκτών) ακολουθούν τη λογική που έχει περιγραφεί ως τώρα και στην επιλογή αποδέκτη εγγράφου.

7.10.3.3 Σχετικά επί της Εργασίας

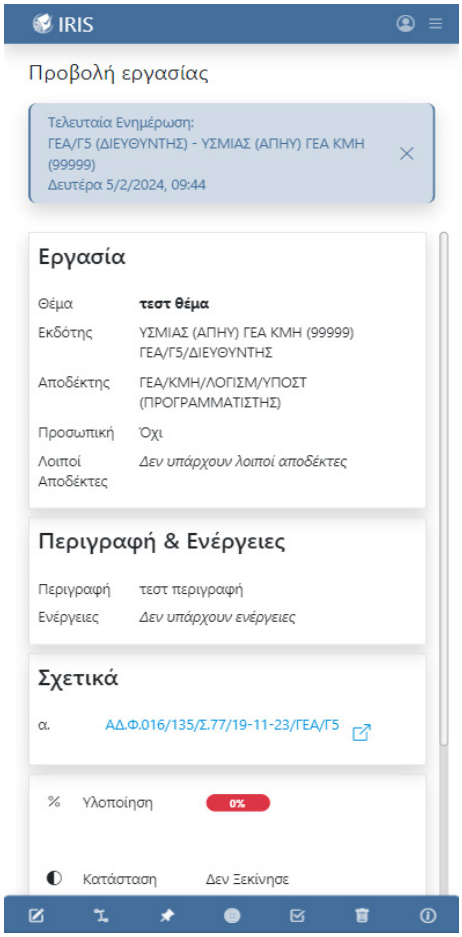
Η επιλογή των σχετικών ακολουθεί τη λογική που έχει περιγραφεί ως τώρα και στην επιλογή σχετικών επί του εγγράφου.

7.10.4 Επισκόπηση Εργασίας

Η προβολή της εργασίας γίνεται σε μια οθόνη όπως αυτή που ακολουθεί και παρέχει την εξής λειτουργικότητα:

1. Αναγράφεται ο χρήστης που τροποποίησε τελευταίος την εργασία.
2. Καρτέλα με τα βασικά στοιχεία της εργασίας. Στους λοιπούς Αποδέκτες εμφανίζεται μια λίστα με όλους όσοι έχουν λάβει την ίδια εργασία από τον εκδότη.
3. Η περιγραφή ενημερώνεται από τον εκδότη της εργασίας ενώ οι ενέργειες από τον αποδέκτη.
4. Λίστα με τα συνδεδεμένα έγγραφα και τα υπόλοιπα σχετικά
5. Πληροφορίες προόδου της εργασίας.
6. Μια εργασία μπορεί να συνδέεται με μια άλλη (υποεργασία) και αντίστοιχα να έχει και μια γονεϊκή εργασία (Εργασία Γονέας).
7. Γραμμή εργαλείων της Εργασίας.

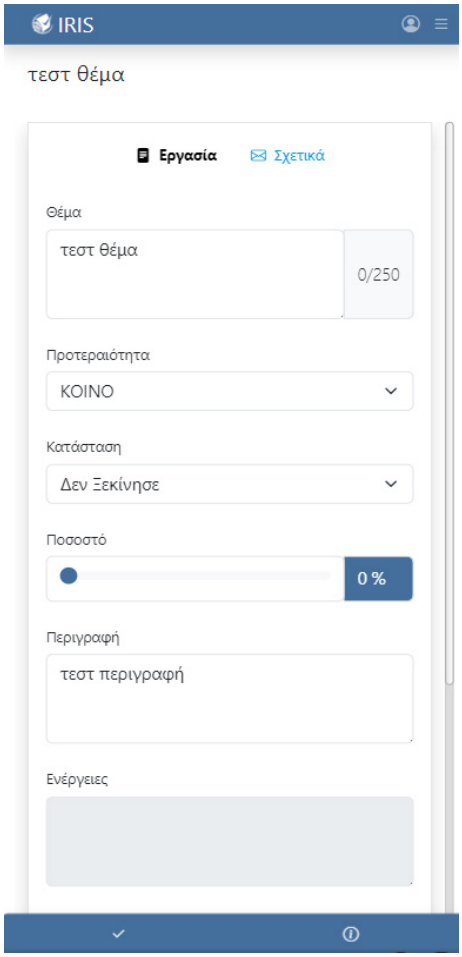
Επεξεργασία μπορεί να κάνει τόσο ο εκδότης όσο και ο αποδέκτης της (διαφορετική σε κάθε περίπτωση). Το κουμπί της εξαρτώμενης εργασίας δημιουργεί νέα υποεργασία. Από εδώ μπορεί ο χρήστης να συνδέσει την εργασία με κάποιο σημαντικό θέμα ή/και στόχο (όπως περιγράφηκε στην ενότητα των εγγράφων). Επίσης παρέχεται δυνατότητα άμεσης ολοκλήρωσης (η οποία θέτει την εργασία σε ολοκληρωμένη και την υλοποίηση στο 100%) και τέλος επιτρέπεται και η διαγραφή της, μόνο από τον εκδότη.



Εικόνα 53. Προβολή εργασίας

7.10.5 Επεξεργασία Εργασίας

Η επεξεργασία της εργασίας είναι δυνατή σε οποιαδήποτε κατάσταση και αν βρίσκεται αυτή.



Εικόνα 54. Επεξεργασία εργασίας

7.10.6 Λίστες Εργασιών

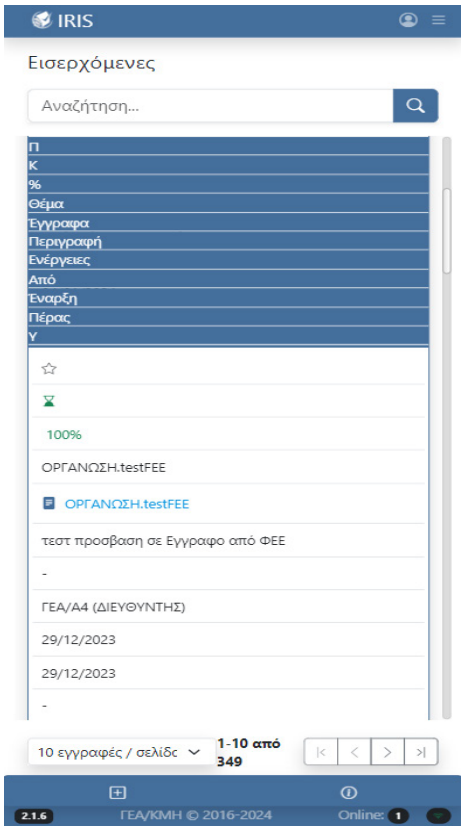
Οι λίστες των εργασιών ακολουθούν τη λογική που περιεγράφηκε και στις λίστες εγγράφων με διαφοροποίηση στις λίστες των εισερχομένων όπου υπάρχει η στήλη Υ (υποεργασίες). Εργασίες με καμία υποεργασία δεν έχουν κάποια πληροφορία, εργασίες με συνδεδεμένες υποεργασίες έχουν ένα πράσινο εικονίδιο . Πηγαίνοντας το ποντίκι πάνω στο εικονίδιο, εμφανίζεται το σύνολο των υποεργασιών. Η στήλη είναι ταξινομήσιμη.

7.10.6.1 Εισερχόμενες / Όλες

Στη λίστα όλων των εισερχομένων εργασιών υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας

- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες



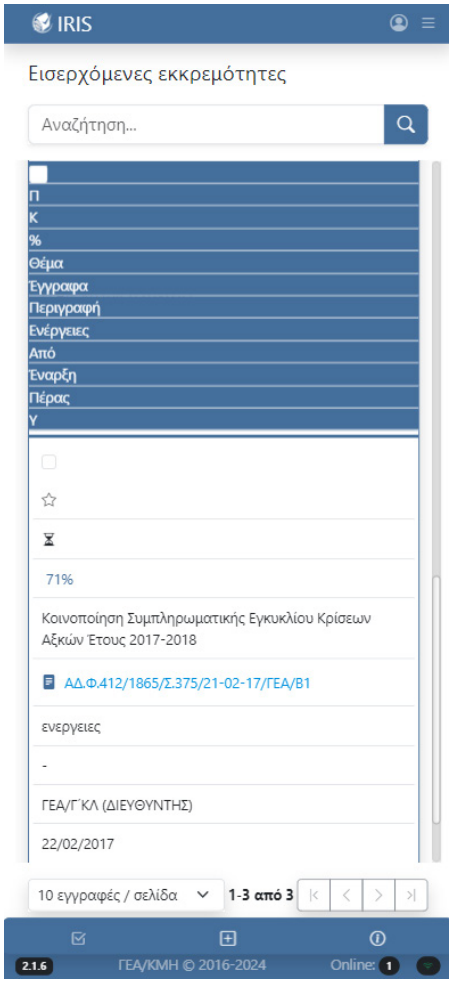
Εικόνα 55. Εισερχόμενες εργασίες/όλες οι εργασίες

7.10.6.2 Εισερχόμενες / Για Ενέργεια / Εκκρεμότητες

Στη λίστα των εισερχομένων εργασιών για *ενέργεια/εκκρεμότητες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας

- Υποεργασίες

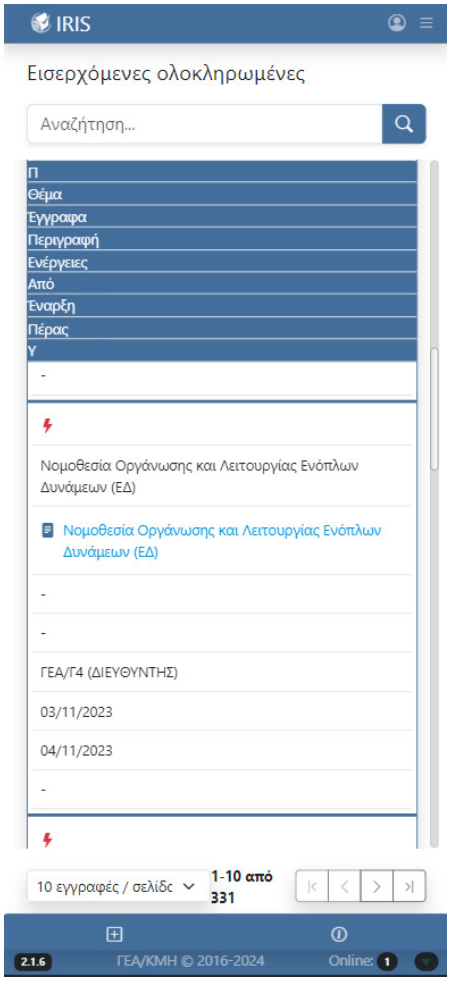


Εικόνα 56. Εισερχόμενες / Για Ενέργεια / Εκκρεμότητες

7.10.6.3 Εισερχόμενες / Για Ενέργεια / Ολοκληρωμένες

Στη λίστα των εισερχομένων εργασιών για *ενέργεια/ολοκληρωμένες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες

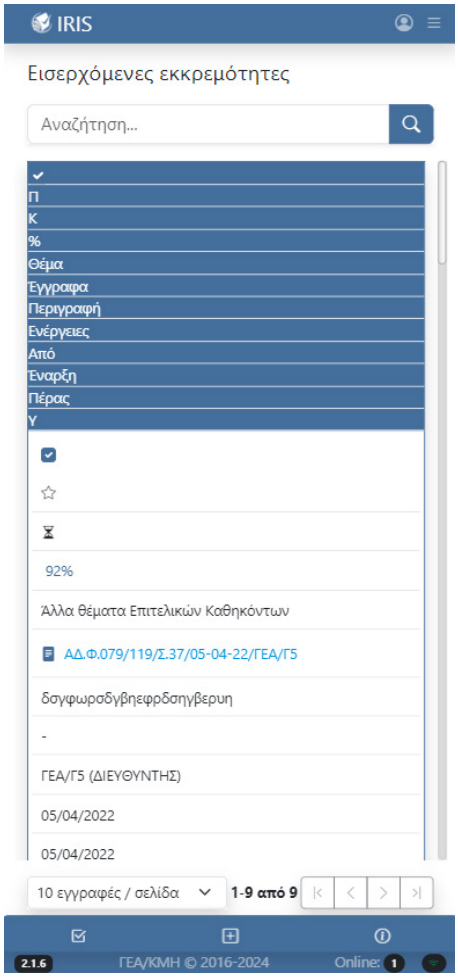


Εικόνα 57. Εισερχόμενες / Για Ενέργεια / Ολοκληρωμένες

7.10.6.4 Εισερχόμενες / Για Ενημέρωση / Εκκρεμότητες

Στη λίστα των εισερχομένων εργασιών για *ενέργεια/εκκρεμότητες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες

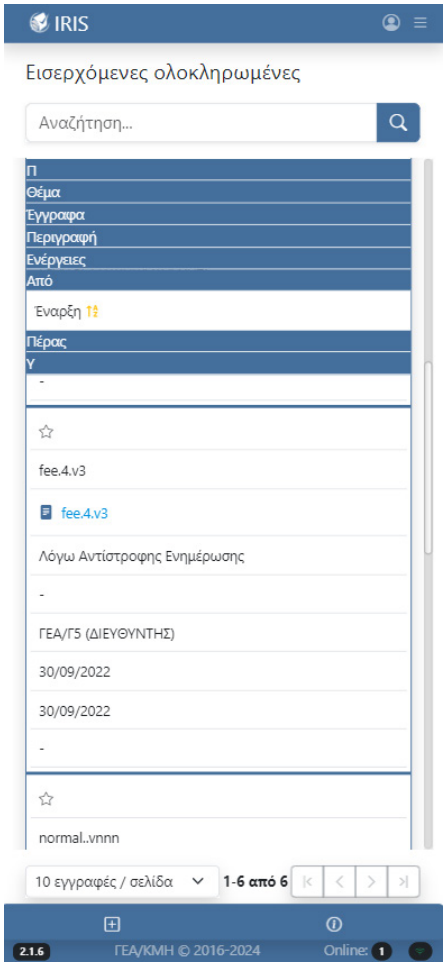


Εικόνα 58. Εισερχόμενες / Για Ενημέρωση / Εκκρεμότητες

7.10.6.5 Εισερχόμενες / Για Ενημέρωση / Ολοκληρωμένες

Στη λίστα των εισερχομένων εργασιών για *ενέργεια/ολοκληρωμένες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες

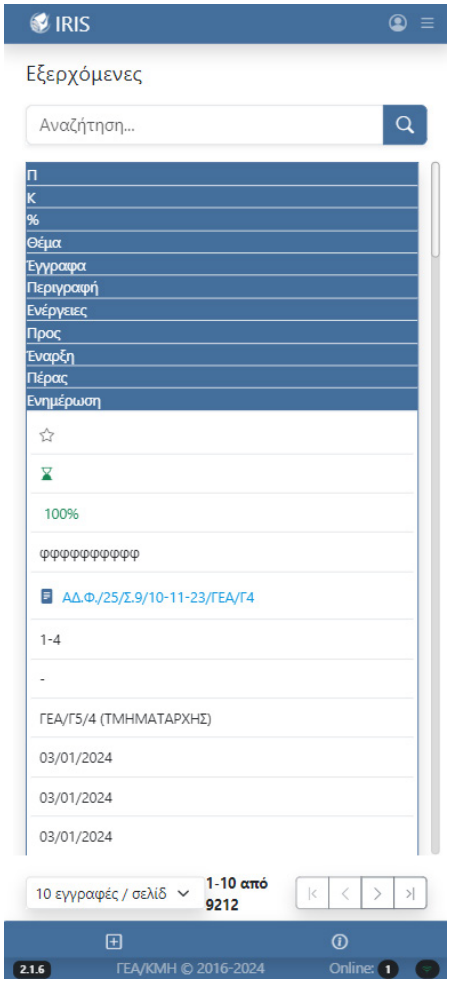


Εικόνα 59. Εισερχόμενες / Για Ενημέρωση / Ολοκληρωμένες

7.10.6.6 Εξερχόμενες / Όλες

Στη λίστα όλων των εξερχομένων εργασιών υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Ημερμομηνία ενημέρωσης της εργασίας

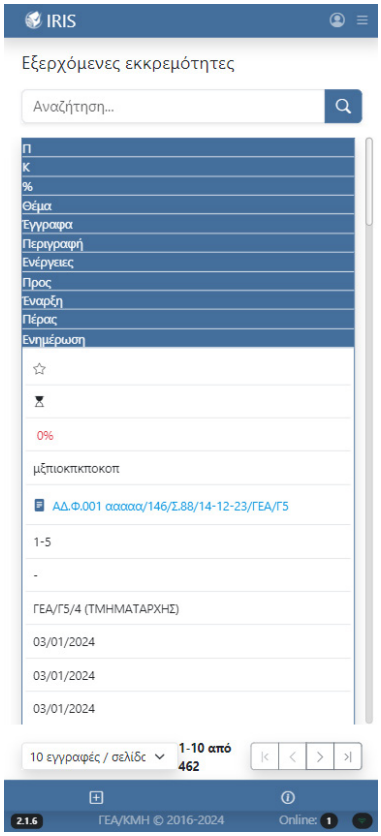


Εικόνα 60. Εξερχόμενες / Όλες

7.10.6.7 Εξερχόμενες / Για Ενέργεια / Εκκρεμότητες

Στη λίστα των εξερχομένων εργασιών για ***ενέργεια/εκκρεμότητες*** εργασιών υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Ημερμομηνία ενημέρωσης της εργασίας

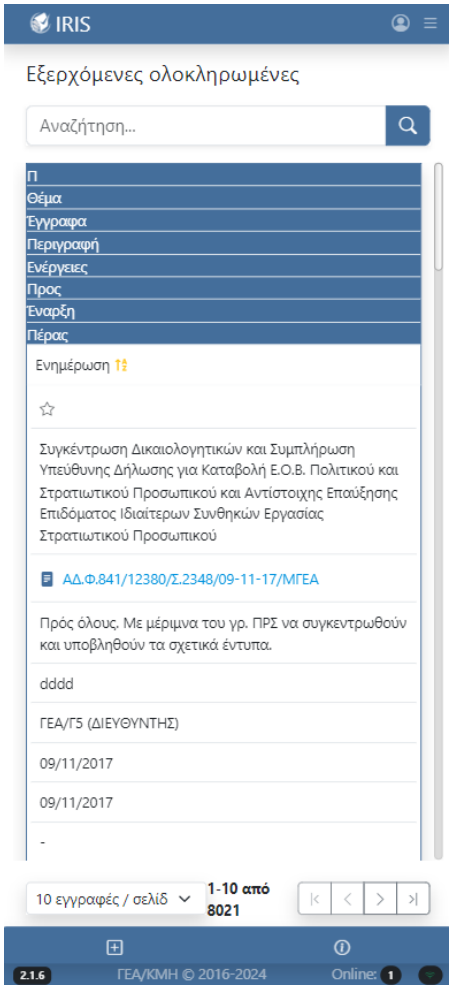


Εικόνα 61. Εξερχόμενες / Για Ενέργεια / Εκκρεμότητες

7.10.6.8 Εξερχόμενες / Για Ενέργεια / Ολοκληρωμένες

Στη λίστα των εξερχομένων εργασιών για *ενέργεια/ολοκληρωμένες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες

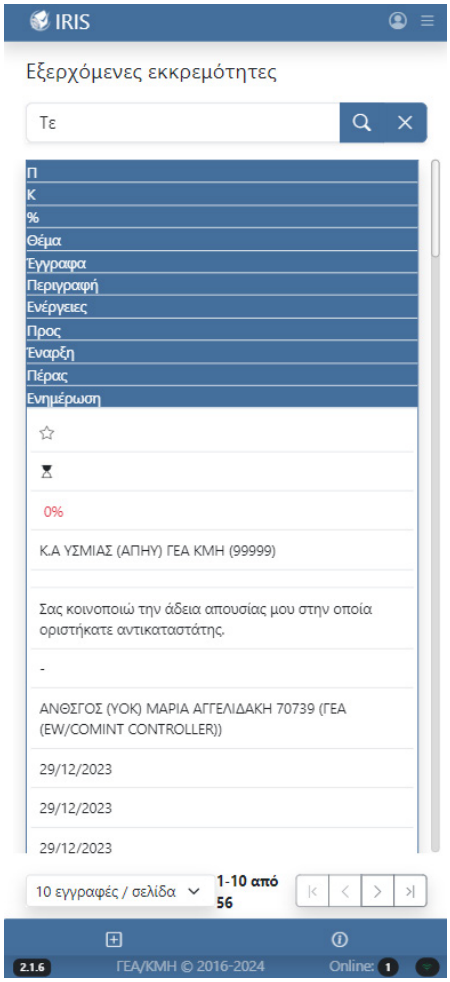


Εικόνα 62. Εξερχόμενες / Για Ενέργεια / Ολοκληρωμένες

7.10.6.9 Εξερχόμενες / Για Ενημέρωση / Εκκρεμότητες

Στη λίστα των εξερχομένων εργασιών για *ενημέρωση/εκκρεμότητες* εργασιών υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Κατάσταση εργασίας
- Ποσοστό υλοποίησης της εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Ημερομηνία ενημέρωσης της εργασίας

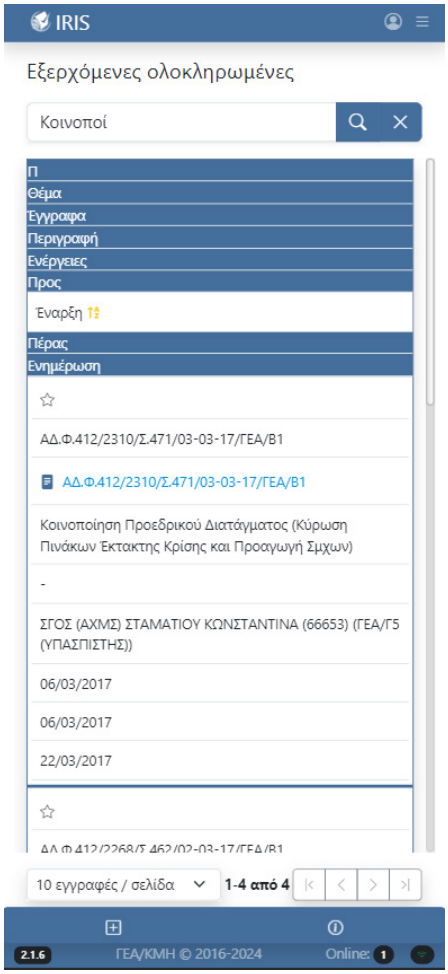


Εικόνα 63. Εξερχόμενες / Για Ενημέρωση / Εκκρεμότητες

7.10.6.10 Εξερχόμενες / Για Ενημέρωση / Ολοκληρωμένες

Στη λίστα των εξερχομένων εργασιών για *ενημέρωση/ολοκληρωμένες* υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα εργασίας
- Θέμα εργασίας
- Λίστα εγγράφων επί της εργασίας
- Περιγραφή εργασίας
- Ενέργειες εργασίας
- Συντάκτης εργασίας (Από)
- Ημερομηνίας έναρξης εργασίας
- Ημερομηνίας πέρατος εργασίας
- Υποεργασίες

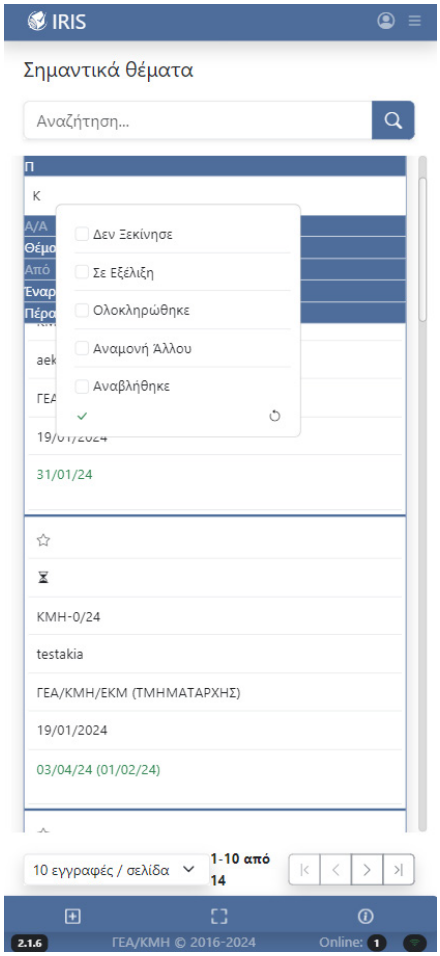


Εικόνα 64. Εξερχόμενες / Για Ενημέρωση / Ολοκληρωμένες

7.11 Σημαντικά θέματα

Στη λίστα των σημαντικών θεμάτων υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Προτεραιότητα σημαντικού θέματος
- Κατάσταση σημαντικού θέματος (με ανάλογο φίλτρο αναζήτησης)
- Αύξων αριθμός σημαντικού θέματος
- Θέμα σημαντικού θέματος
- Συντάκτης σημαντικού θέματος (Από)
- Ημερομηνίας έναρξης σημαντικού θέματος
- Ημερομηνίας πέρατος σημαντικού θέματος



Εικόνα 65. Σημαντικά θέματα

7.11.1 Νέο Σημαντικό Θέμα

Σημαντικά Θέματα είναι ένας διαφορετικός τρόπος οργάνωσης της εργασίας του χρήστη στο Ίριδα, με τον οποίο μπορεί να ομαδοποιούνται Έγγραφα και Εργασίες. Η λειτουργικότητα ομοιάζει με αυτήν της εργασίας με τις εξής διαφοροποιήσεις:

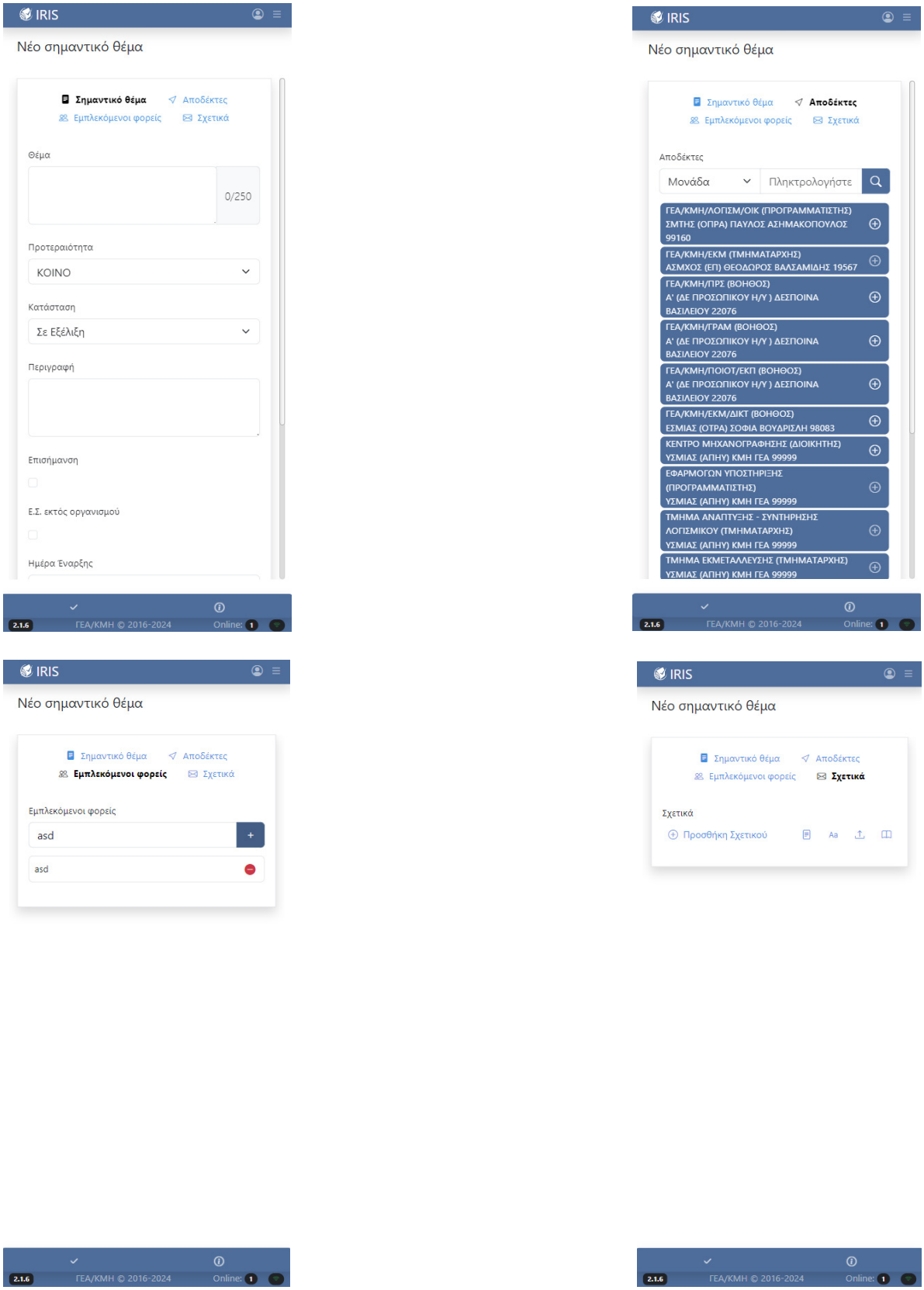
✓ Υπάρχουν τα πεδία “Επισήμανση” και “Ε.Σ. εκτός οργανισμού” τα οποία ο χρήστης μπορεί να τα επιλέξει.

✓ Υπάρχει η καρτέλα “Εμπλεκόμενοι φορείς” μέσω της οποίας εισάγονται (με την μορφή λεκτικού) τυχόν άλλοι φορείς που εμπλέκονται στο θέμα.

✓ Ένα θέμα μπορεί να συνδέεται τόσο με Έγγραφα όσο και με Εργασίες.

✓ Διαθέτουν ημερομηνίας πέρατος, για την οποία γίνεται παρακολούθηση της τροποποίησής της, από το σύστημα.

✓ Ένα θέμα μπορεί να έχει πολλούς αποδέκτες.

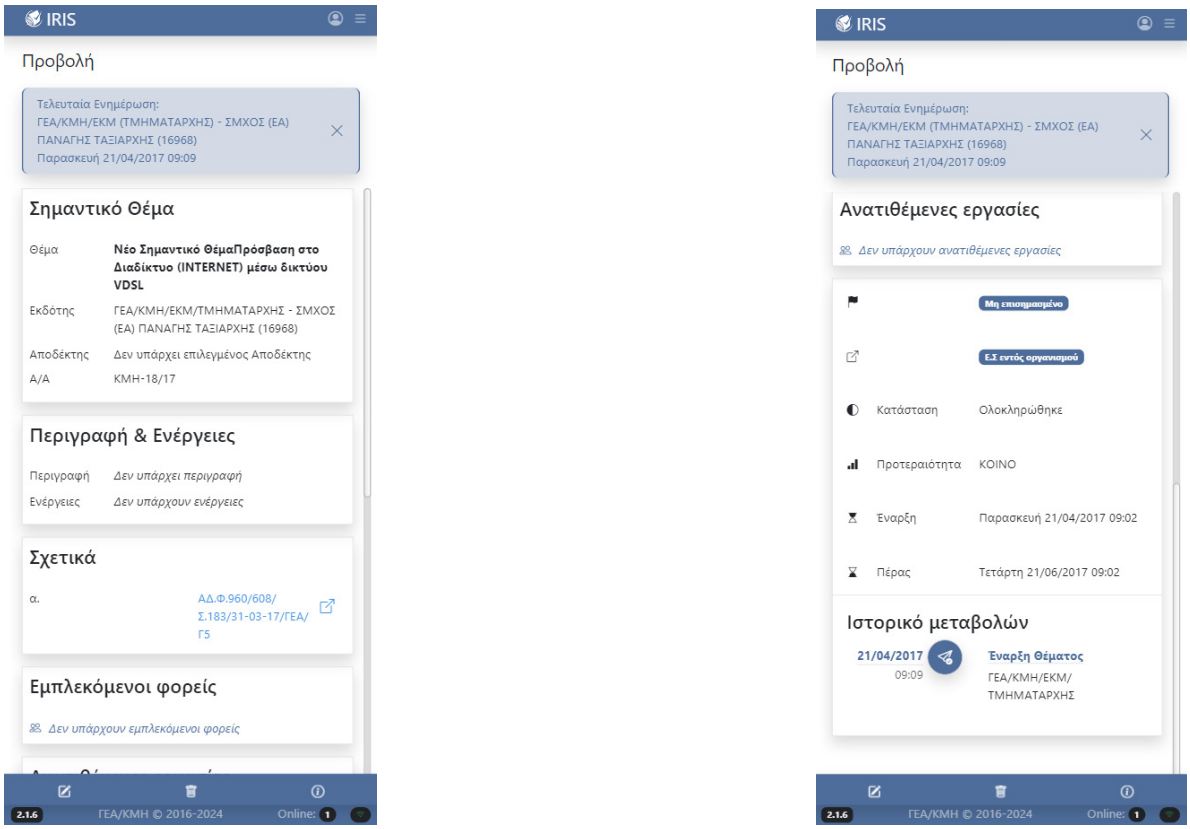


Εικόνα 66. Νέο σημαντικό θέμα

7.11.2 Επισκόπηση Σημαντικού Θέματος

Η προβολή Σημαντικού Θέματος γίνεται σε μια οθόνη όπως αυτή που ακολουθεί και παρέχει την εξής λειτουργικότητα:

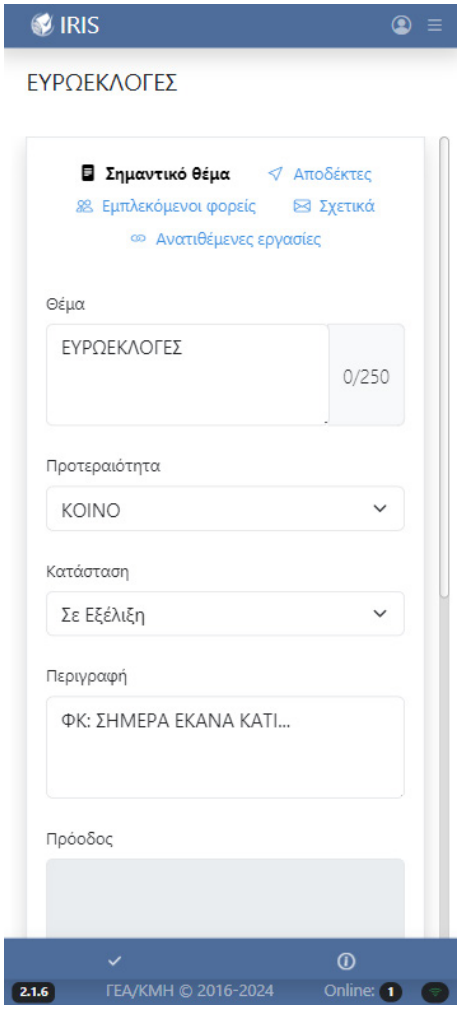
1. Αναγράφεται ο χρήστης που τροποποίησε τελευταίος το θέμα
2. Καρτέλα με τα βασικά στοιχεία του θέματος. Το Α/Α αποδίδεται αυτόματα και συντίθεται από τα εξής: Δνση - αύξουσα αρίθμηση ανά έτος / έτος.
3. Η περιγραφή ενημερώνεται από τον εκδότη του θέματος ενώ οι ενέργειες από τους αποδέκτες.
4. Λίστα με τα συνδεδεμένα έγγραφα και τα υπόλοιπα σχετικά.
5. Λίστα με τους Εμπλεκόμενοι φορείς.
6. Λίστα με τις συνδεδεμένες εργασίες και το ποσοστό υλοποίησής τους.
7. Πληροφορίες προόδου της εργασίας.
8. Λίστα με το ιστορικό μεταβολών η οποία ενημερώνεται οποτεδήποτε αλλάζει η ημερομηνία πέρατος.
9. Γραμμή εργαλείων του θέματος. Επεξεργασία μπορεί να κάνει τόσο ο εκδότης όσο και ο αποδέκτης της (διαφορετική σε κάθε περίπτωση).



Εικόνα 67. Προβολή σημαντικού θέματος

7.11.3 Επεξεργασία Σημαντικού Θέματος

Η επεξεργασία του θέματος είναι δυνατή σε οποιαδήποτε κατάσταση και αν βρίσκεται αυτό. Εφόσον ο χρήστης μεταβάλει την ημερομηνία λήξης θα πρέπει να συμπληρώσει έναν λόγο στον διάλογο που απεικονίζεται παραπάνω.



Εικόνα 68. Επεξεργασία σημαντικού θέματος

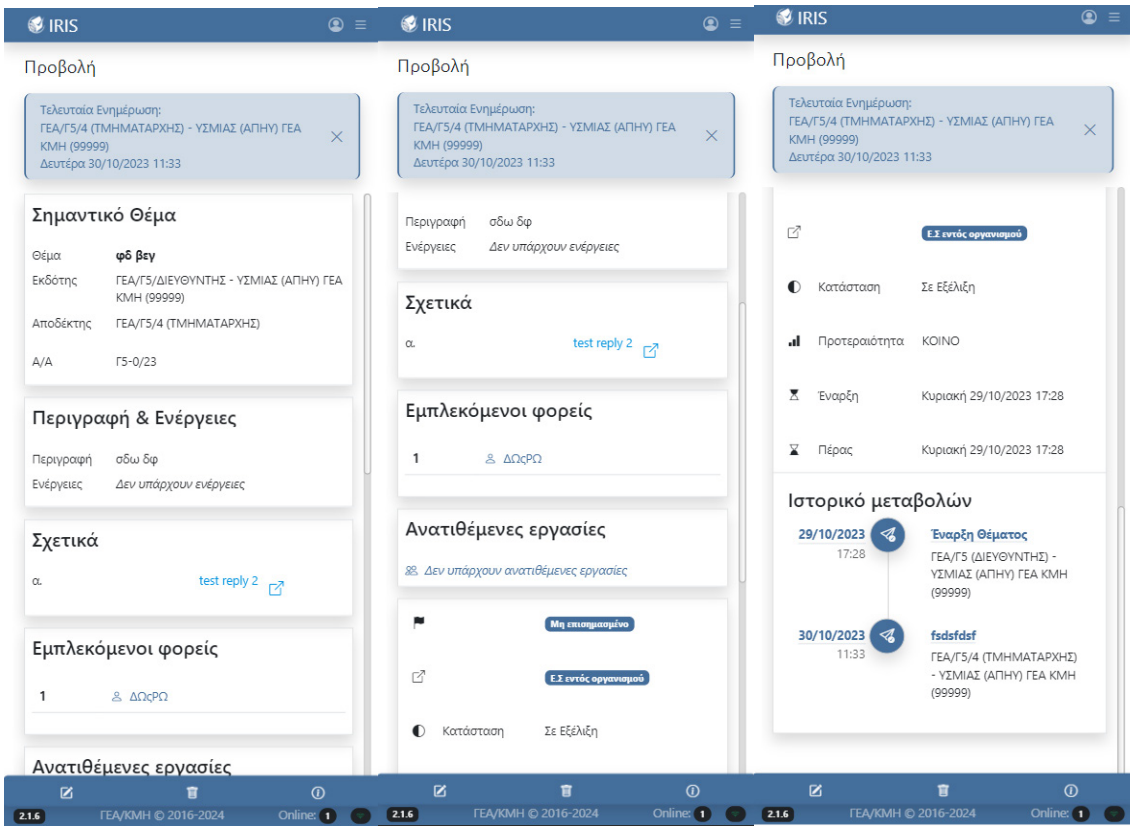
7.11.4 Λίστες Σημαντικών Θεμάτων

Οι λίστες των σημαντικών θεμάτων ακολουθούν τη λογική που περιεγράφηκε και στις λίστες εγγράφων με βασική διαφοροποίηση ως προς τη στήλη “Πέρας” η οποία έχει μία μόνο ημερομηνία εφόσον δεν έχει τροποποιηθεί αυτή πότε και δύο σε αντίθετη περίπτωση (η τρέχουσα και δίπλα σε παρένθεση η πριν την τροποποίηση ημερομηνία).

Ακολουθείται και χρωματική κωδικοποίηση ως εξής:

- Κόκκινη Γραμματοσειρά: Εκπρόθεσμη ημερομηνία.
- Πράσινη Γραμματοσειρά: Εμπρόθεσμη ημερομηνία.
- Κίτρινη Γραμματοσειρά: Επικείμενη λήξη (σε 2 μήνες).

Επίσης παρέχεται η δυνατότητα φιλτραρίσματος ως προς τις καταστάσεις των σημαντικών θεμάτων.

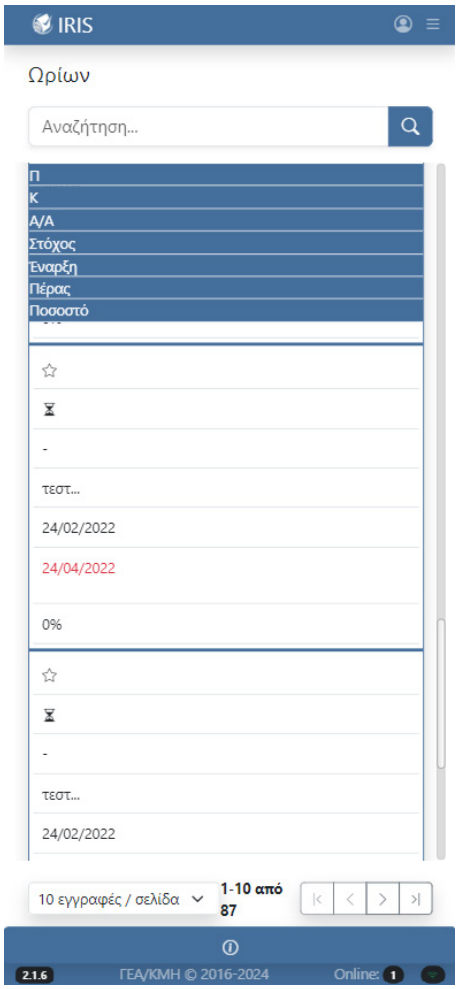


Εικόνα 69. Λίστες σημαντικών θεμάτων

7.12 Ωρίων / Στοχοθεσία

Η λογική και η υλοποίηση των στόχων είναι ίδια με των σημαντικών θεμάτων σε όλα τα επιμέρους στοιχεία με μόνη διαφορά ότι δεν μπορεί οποιοσδήποτε χρήστης να δημιουργήσει

στόχους αλλά και να έχει πρόσβαση. Δημιουργία νέου στόχου κάνει μόνο η ΓΕΑ/Δ1/ 4 και θέαση σε αυτούς έχει εν δυνάμει όλα το προσωπικό του ΓΕΑ και κάποιες εξαιρέσεις.



Εικόνα 70. Ωρίων/Στοχοθεσία

7.14 Εργαλεία

7.14.1 Αρχική Σελίδα Ενότητας

Στην αρχική σελίδα της ενότητας των εργαλείων αποτυπώνονται αριθμητικές πληροφορίες για το σύνολο των επαφών του χρήστη, των εγγράφων της βιβλιοθήκης του συστήματος, των αδειών που έχει λάβει, καθώς και τυχόν αντικαταστάσεις.

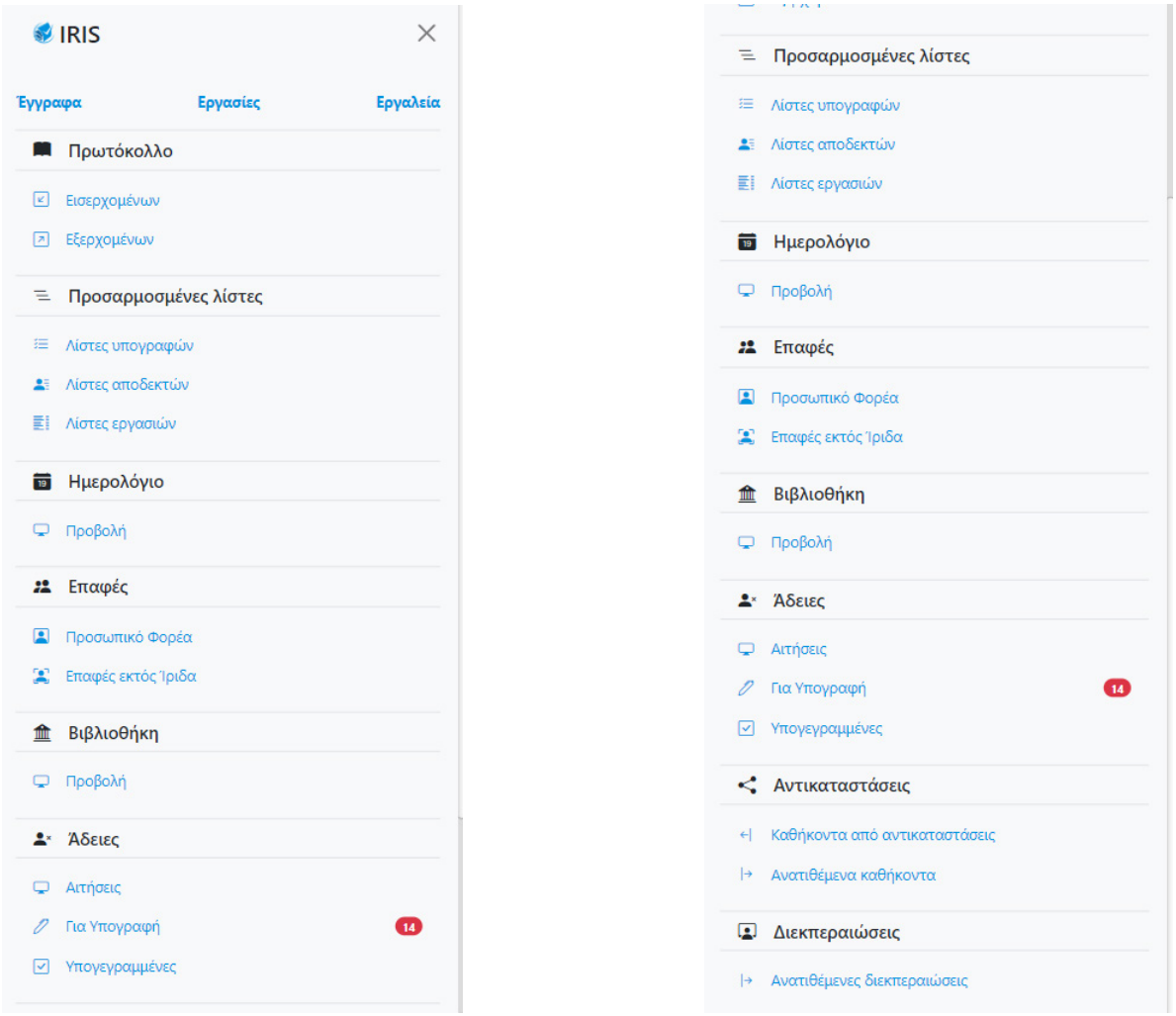
Εικόνα 71. Αρχική σελίδα ενότητας

7.14.2 Μενού Ενότητας

Στο αριστερό μενού της ενότητας των εργαλείων παρέχονται οι εξής επιλογές:

- Πρωτόκολλο
    - Εισερχομένων
    - Εξερχομένων
- Προσαρμοσμένες λίστες
    - Λίστες υπογραφών

    - Λίστες αποδεκτών
    - Λίστες εργασιών
- Ημερολόγιο
    - Προβολή
- Επαφές
    - Προσωπικό Φορέα
    - Επαφές εκτός Ίριδα
    - Φορείς Ίριδα
- Βιβλιοθήκη
    - Προβολή
- Άδειες
    - Αιτήσεις
    - Για Υπογραφή
    - Υπογεγγραμένες
- Αντικαταστάσεις
    - Καθήκοντα από αντικαταστάσεις
    - Ανατιθέμενα καθήκοντα
- Διεκπεραιώσεις
    - Ανατιθέμενες διεκπεραιώσεις



Εικόνα 72. Μενού ενότητας

7.14.3 Πρωτόκολλο

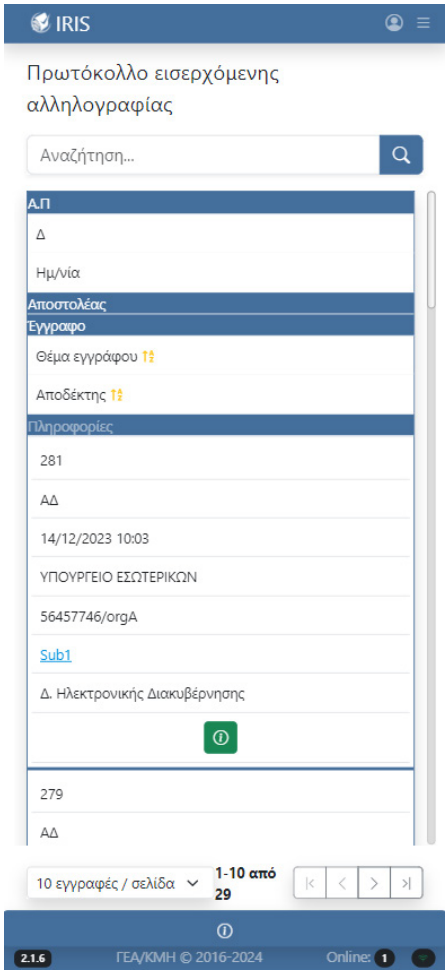
Στο Πρωτόκολλο γίνεται καταγραφή όλων εγγράφων- τόσο αυτών που διακινούνται μέσω Ίριδα όσο και αυτών που εξαιρούνται - είτε είναι εισερχόμενα είτε εξερχόμενα. Το βιβλίο πρωτοκόλλου είναι κοινό για όλους τους χρήστες της μονάδας αλλά η πρόσβαση στα έγγραφα που φιλοξενεί, εξαρτάται από τα δικαιώματα του καθενός σε αυτά(τα έγγραφα) .

7.14.3.1 Πρωτόκολλο Εισερχομένων

Κάνοντας κλικ στο Πρωτόκολλο Εισερχομένων μεταβαίνουμε στην παρακάτω οθόνη με τη λειτουργικότητα που περιγράφεται ως ακολούθως:

1. Η αναζήτηση λειτουργεί όπως και η αναζήτηση στις λίστες των εγγράφων.
2. Πέραν της αναζήτησης παρέχεται και η δυνατότητα φιλτραρίσματος των εγγραφών με βάση τη Διαβάθμισή τους (πολλαπλή επιλογή) και την Ημερομηνία πρωτοκόλλησης τους (εύρος ημερομηνιών).
3. Με αυτό το εικονίδιο επισημαίνονται οι manual πρωτοκολλήσεις εγγράφων εκτός συστήματος. Με τη διαγράμμιση υποτυπώνονται οι ακυρωμένες εγγραφές (είτε από ακύρωση manual πρωτοκόλλησης εγγράφων εκτός συστήματος, είτε από καταχώρηση εγγράφου ως Ορθή επανάληψη).

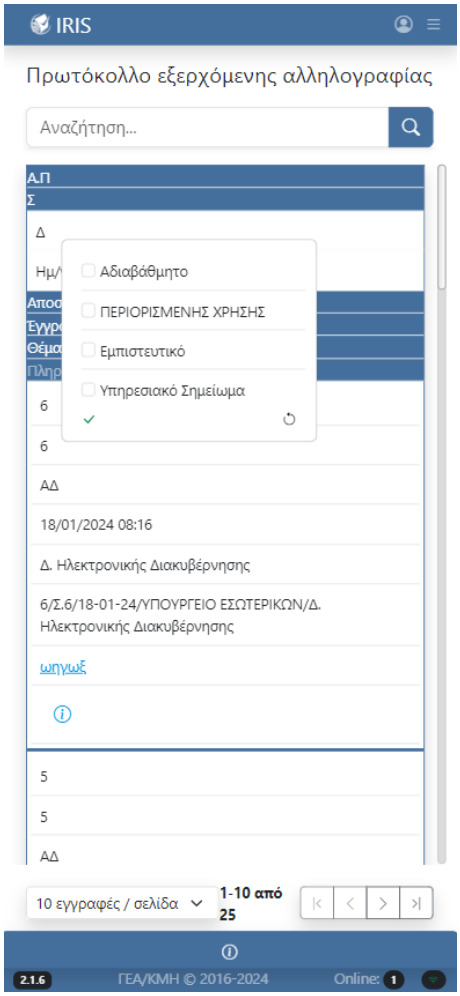
4. Όταν μια εγγραφή πρωτοκόλλου δεν έχει υπερσύνδεσμο στο θέμα του εγγράφου, ο χρήστης δεν έχει πρόσβαση στο έγγραφο .
5. Πατώντας το κουμπί μεταβαίνουμε σε καρτέλα με πληροφορίες της εγγραφή.



Εικόνα 73. Πρωτόκολλο εισερχομένων

7.14.3.2 Πρωτόκολλο Εξερχομένων

Το Πρωτόκολλο Εξερχομένων έχει παρόμοια λειτουργικότητα με αυτό των εισερχομένων, αλλά αφορά τα εξερχόμενα έγγραφα της Μονάδας.



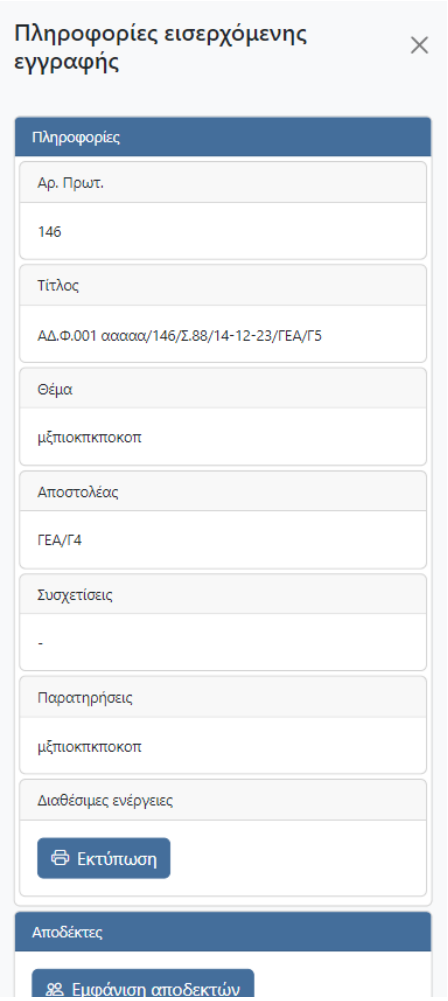
Εικόνα 74. Πρωτόκολλο εξερχομένων

7.14.3.3 Πληροφορίες εισερχόμενης εγγραφής

Κάνοντας κλικ στη στήλη “Πληροφορίες” του Πρωτοκόλλου Εισερχομένων εμφανίζεται παράθυρο με τις εξής πληροφορίες:

1. Βασικές πληροφορίες εγγραφής όπως Αρ. Πρωτοκόλλου, τίτλος εγγράφου, θέμα και αποστολέας.
2. Τυχόν συσχετίσεις που έχει το έγγραφο. Όταν για το έγγραφο που αφορά η εγγραφή του πρωτοκόλλου, έχει εκδοθεί Ορθή Επανάληψη, η αποτύπωση στις πληροφορίες του βιβλίου είναι η ανωτέρω. Αντίστοιχα στο νέο (ορθό) έγγραφο η αποτύπωση στις πληροφορίες του βιβλίου γίνεται ως εξής: Τέλος, αν ένα εισερχόμενο έγγραφο απαντήθηκε με κάποιο εξερχόμενο, η αποτύπωση στις πληροφορίες του βιβλίου γίνεται ως εξής: Το έγγραφο δεν σχετίζεται με κάποιο άλλο έγγραφο και δεν έχει αρχειοθετηθεί Το έγγραφο δεν σχετίζεται με κάποιο άλλο έγγραφο και έχει αρχειοθετηθεί Το έγγραφο σχετίζεται με άλλο έγγραφο και δεν έχει αρχειοθετηθεί Το έγγραφο σχετίζεται με άλλο έγγραφο και έχει αρχειοθετηθεί
3. Εκτυπωτικό κουμπί εγγραφής Πρωτοκόλλου.
4. Κάνοντας κλικ στο κουμπί των αποδεκτών εμφανίζεται λίστα με τους αποδέκτες του εγγράφου εντός της Μονάδας (ή διεύθυνσης για επιτελεία). Σε κάθε αποδέκτη πάνω αριστερά αποτυπώνεται ο τρόπος με τον οποίο έλαβε το έγγραφο (αρχικός αποδέκτης

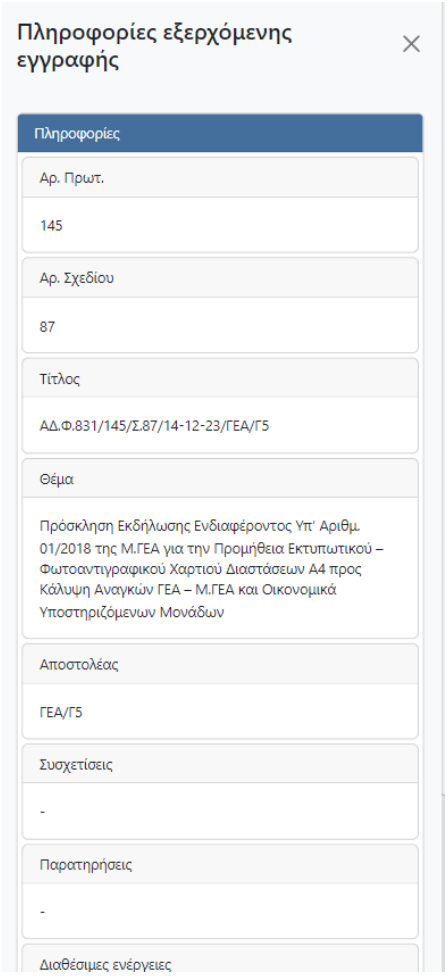
ή μέσω εργασίας) και από κάτω λοιπές πληροφορίες, όπως αν έχει αναγνώσει το έγγραφο ή όχι καθώς και διαθέσιμα στοιχεία επικοινωνίας



Εικόνα 75. Πληροφορίες εισερχόμενης εγγραφής

7.14.3.4 Πληροφορίες εξερχόμενης εγγραφής

Η λογική είναι παρόμοια με τις πληροφορίες της προηγούμενης παραγράφου με διαφορά στους αποδέκτες, όπου είναι μια λίστα με τους αποδέκτες του εγγράφου χωρίς περισσότερα στοιχεία.



Εικόνα 76. Πληροφορίες εξερχόμενης εγγραφής

7.14.3.5 Εισαγωγή εισερχόμενης / εξερχόμενης εγγραφής

Στο Ίριδα μπορούμε να πρωτοκολλούμε έγγραφα τα οποία εξαιρούνται για κάποιο λόγο αλλά θέλουμε να τα παρακολουθούμε μέσω ενός κοινού πρωτοκόλλου.

✓ Στην Διαβάθμιση επιλέξτε μια από τις διαθέσιμες επιλογές της λίστας (υποχρεωτικό πεδίο).

✓ Το Θέμα του εγγράφου (υποχρεωτικό πεδίο).

✓ Τρόπος Παραλαβής/Αποστολής(για εισερχόμενη/εξερχόμενη εγγραφή αντίστοιχα) του εγγράφου.

✓ Ημερομηνία Έκδοσης του εγγράφου.

✓ Ώρα Έκδοσης του εγγράφου.

✓ Στις Παρατηρήσεις προαιρετικά συμπληρώνονται σημειώσεις που αφορούν το έγγραφο.

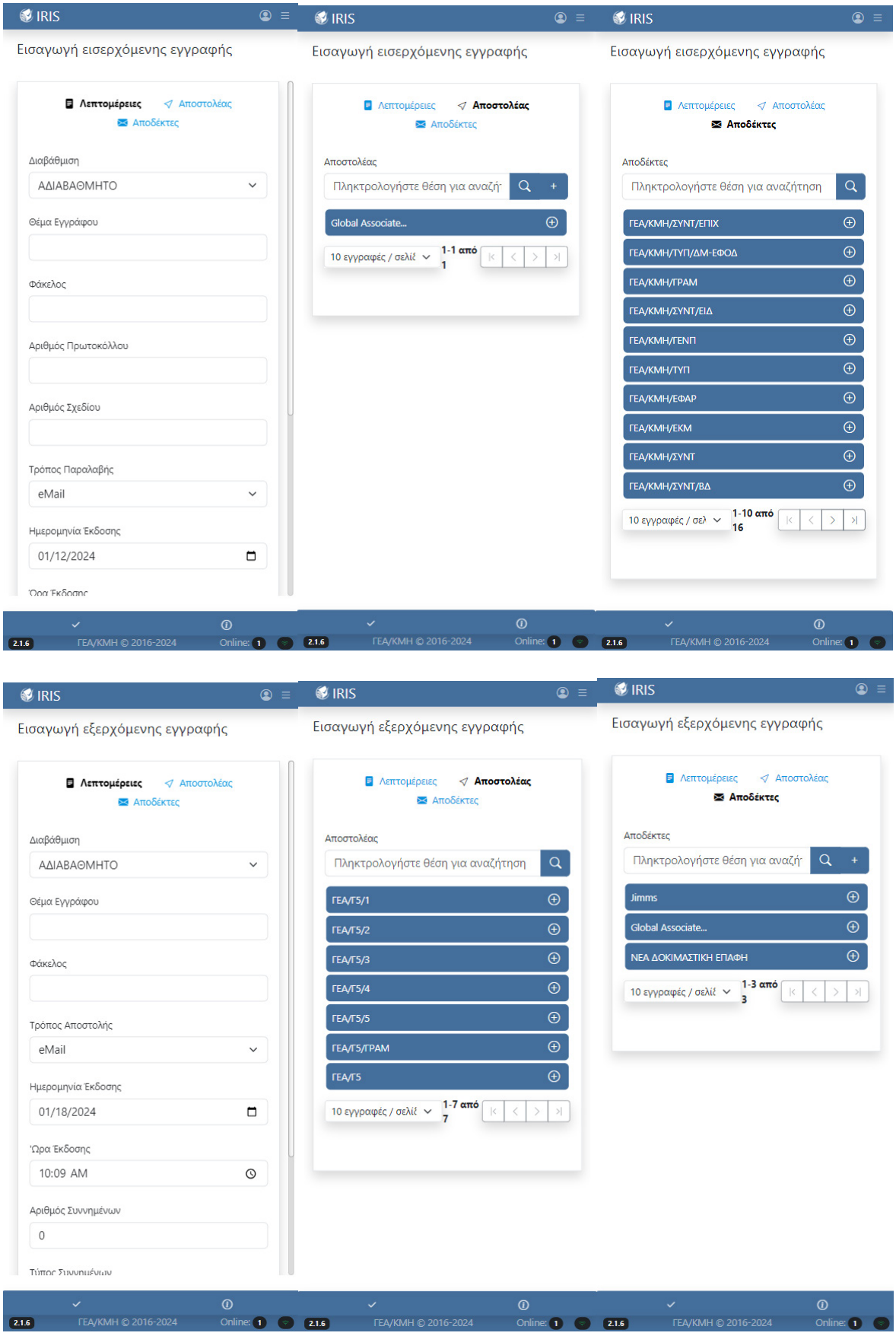
✓ Αριθμός και τύπος Συνημμένων

✓ Αποστολέας (υποχρεωτικό πεδίο)

1. Για εισερχόμενη εγγραφή επιλέγεται από Επαφές του χρήστη. Σε περίπτωση μη εύρεσης της επαφής, μπορεί να προστεθεί επί τόπου χειροκίνητα ένας αποστολέας.
2. Για εξερχόμενη εγγραφή επιλέγεται από τη μονάδα (ή διεύθυνση) του χρήστη.

✓ Αποδέκτες (υποχρεωτικό πεδίο).

1. Για εξερχόμενη εγγραφή επιλέγεται από Επαφές του χρήστη. Σε περίπτωση μη εύρεσης της επαφής, μπορεί να προστεθεί επί τόπου χειροκίνητα ένας αποδέκτης.
2. Για εισερχόμενη εγγραφή επιλέγεται από τη μονάδα (ή διεύθυνση) του χρήστη.



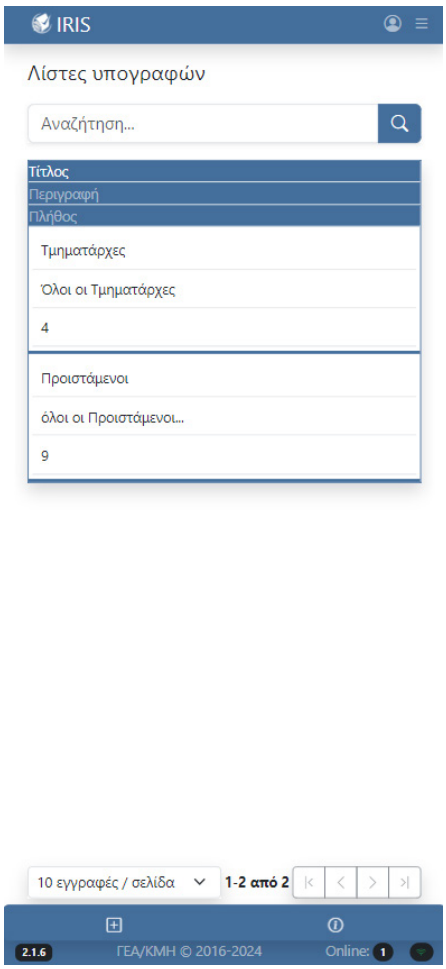
Εικόνα 77. Πληροφορίες εξερχόμενης εγγραφής

7.14.4 Προσαρμοσμένες Λίστες

Παρέχεται η δυνατότητα στον χρήστη να δημιουργήσει τις δικές του λίστες υπογραφών , αποδεκτών και εργασιών, οι οποίες είναι αξιοποιήσιμες στη δημιουργία εγγράφων αλλά και στη διαχείριση των εργασιών. Οι λίστες αυτές είναι κοινές για χρήστες που έχουν τον ίδιο ρόλο (Καθήκον και θέση).

7.14.4.1 Λίστες Υπογραφών

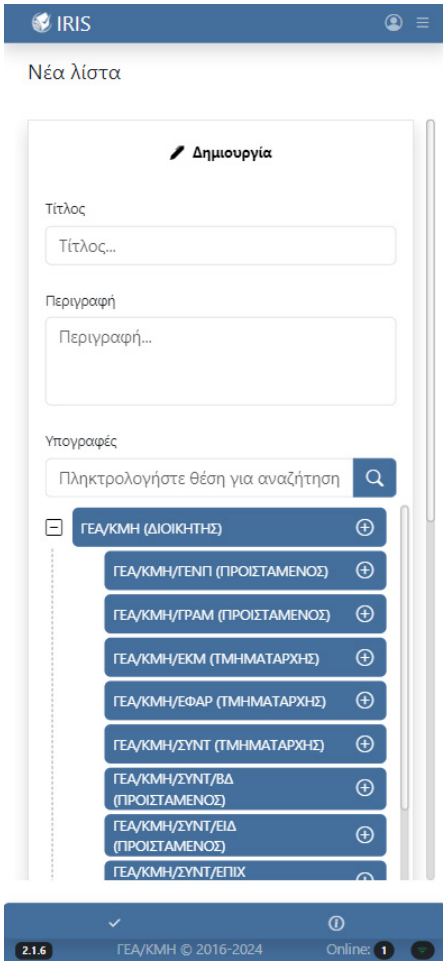
Εμφανίζονται όλες οι λίστες που έχει δημιουργήσει ο χρήστης με το πλήθος των υπογραφόντων και την περιγραφή τους. Η αναζήτηση γίνεται στον τίτλο της λίστας.



Εικόνα 78. Λίστες υπογραφών

7.14.4.2 Νέα Λίστα Υπογραφών

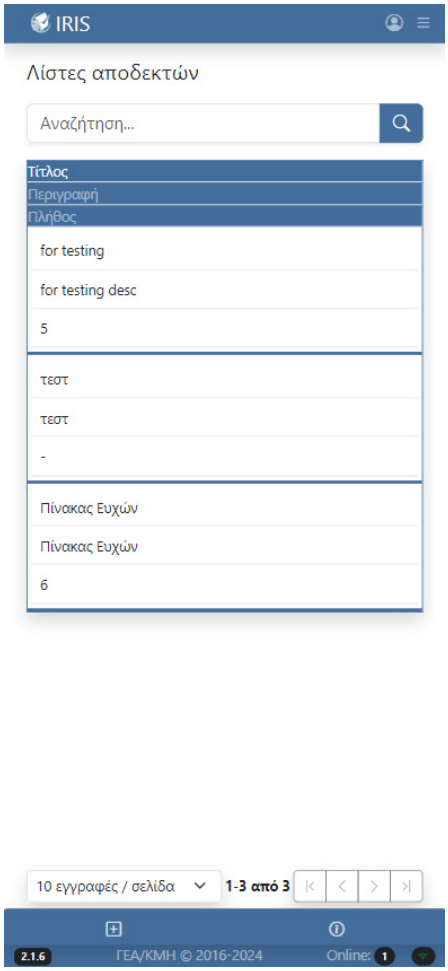
Η δημιουργία της λίστας υπογραφών έχει τη λογική που περιγράφηκε και προηγουμένως και στις γενικές λίστες υπογραφών του νέου εγγράφου, με μοναδική διαφοροποίηση ότι το προφίλ του χρήστη δεν μπορεί να τοποθετηθεί πρώτο στη λίστα.



Εικόνα 79. Νέα λίστα υπογραφών

7.14.5 Λίστες Αποδεκτών

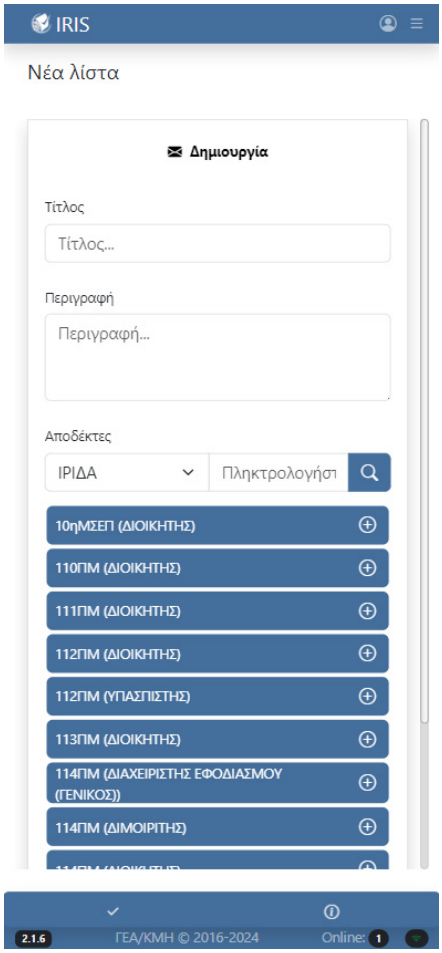
Εμφανίζονται όλες οι λίστες που έχει δημιουργήσει ο χρήστης με το πλήθος των αποδεκτών και την περιγραφή τους. Η αναζήτηση γίνεται στον τίτλο.



Εικόνα 80. Λίστες αποδεκτών

7.14.5.1 Νέα Λίστα Αποδεκτών

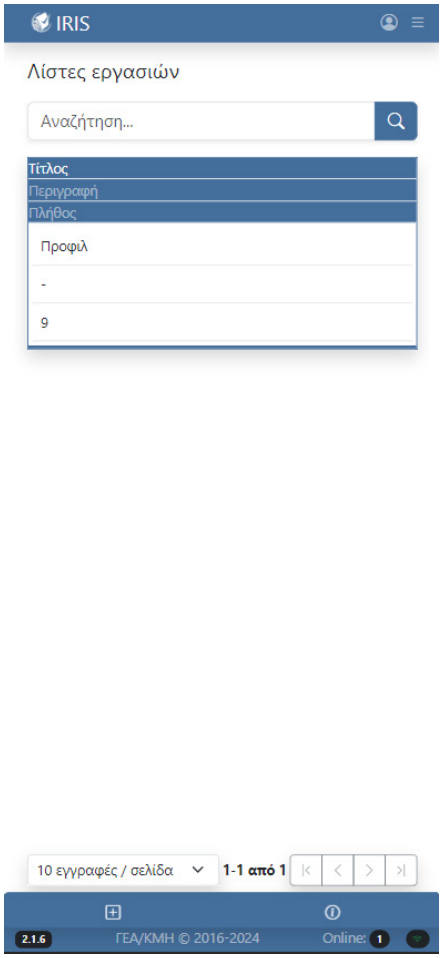
Η δημιουργία της λίστας αποδεκτών έχει τη λογική που περιγράφηκε και προηγουμένως και στις γενικές λίστες υπογραφών του νέου εγγράφου.



Εικόνα 81. Νέα λίστα αποδεκτών

7.14.5.2 Λίστες Εργασιών

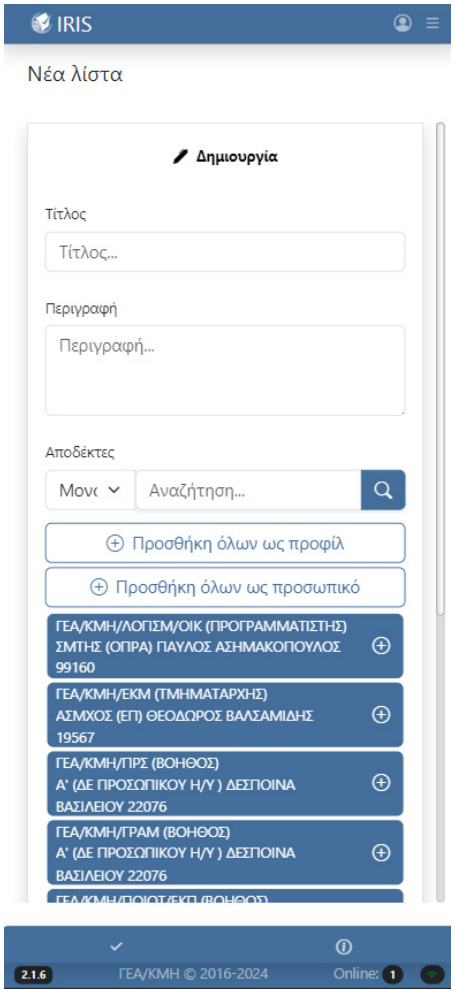
Εμφανίζονται όλες οι λίστες που έχει δημιουργήσει ο χρήστης με το πλήθος των υπογραφόντων και την περιγραφή τους. Η αναζήτηση γίνεται στον τίτλο της λίστας.



Εικόνα 82. Λίστες εργασιών

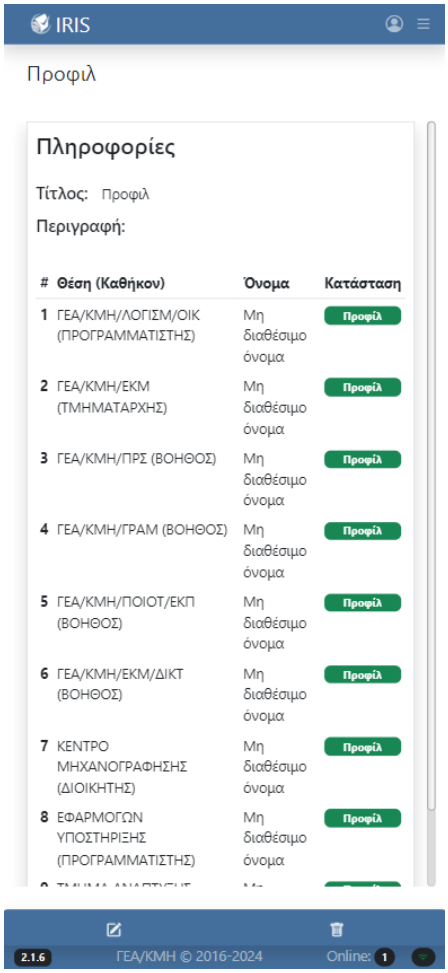
7.14.5.3 Νέα Λίστα Εργασιών

Η δημιουργία της λίστας υπογραφών έχει τη λογική που περιεγράφηκε και προηγουμένως και στις γενικές λίστες υπογραφών του νέου εγγράφου, με μοναδική διαφοροποίηση ότι το προφίλ του χρήστη δεν μπορεί να τοποθετηθεί πρώτο στη λίστα.



Εικόνα 83. Νέα λίστα εργασιών

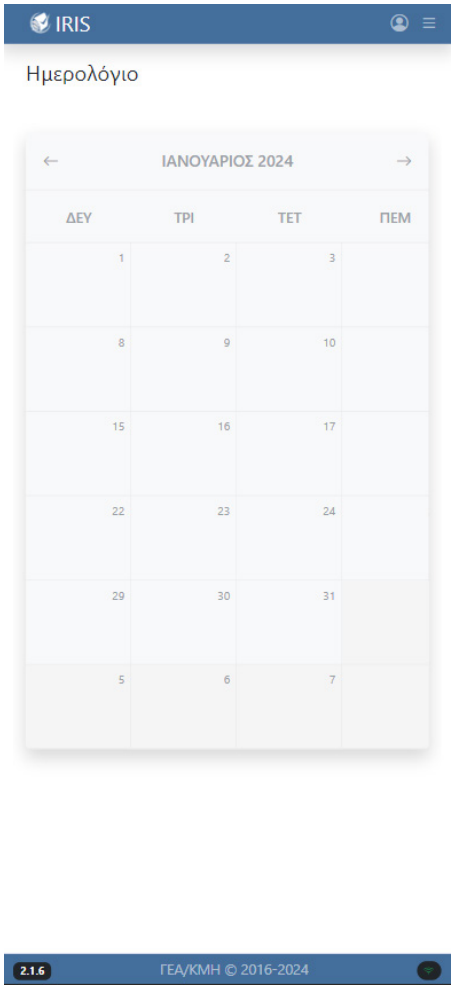
7.14.5.4 Προβολή Λίστας Εργασιών



Εικόνα 84. Προβολή λίστας εργασιών

7.14.6 Ημερολόγιο

Αποτελεί μια ημερολογιακή προβολή των ενεργών εργασιών του χρήστη. Επιλέγοντας μια εργασία από το ημερολόγιο μεταβαίνουμε σε αυτήν.

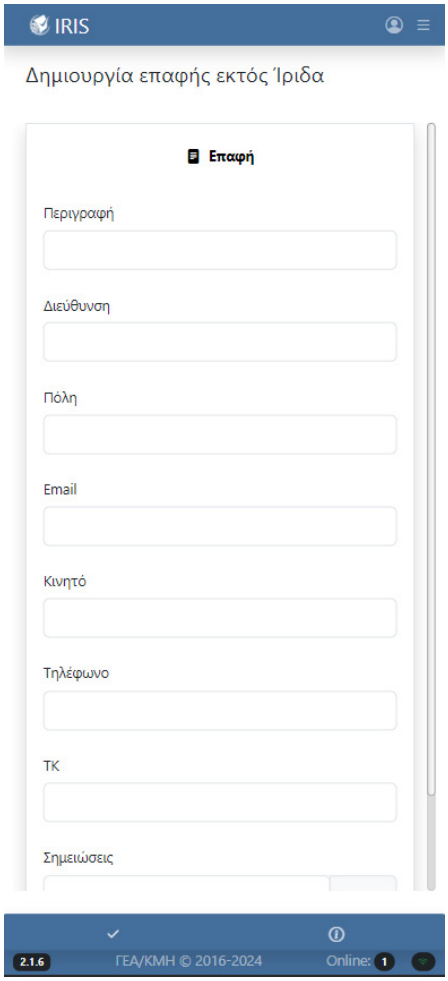


Εικόνα 85. Ημερολόγιο

7.14.7 Επαφές

7.14.7.1 Νέα Επαφή

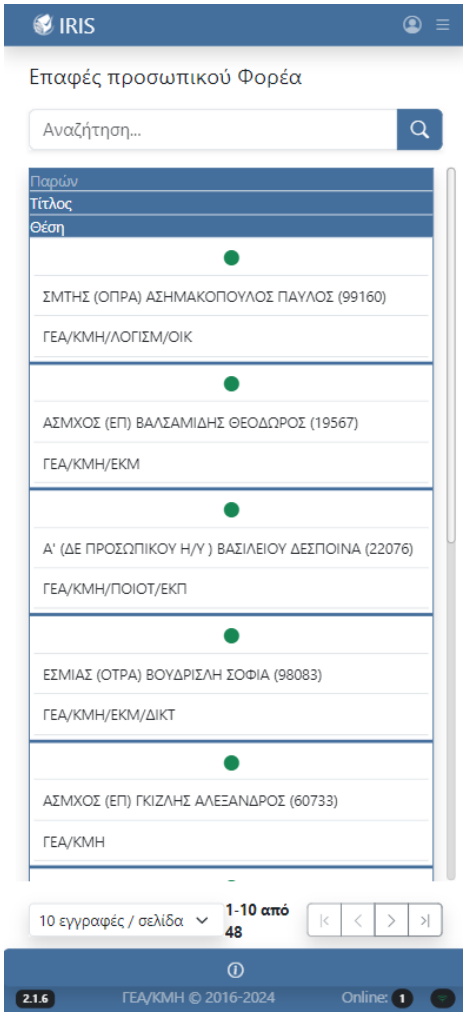
Για τη Δημιουργία επαφής εκτός Ίριδα αρκεί να συμπληρώσουμε την παρακάτω φόρμα: Η περιγραφή είναι το μοναδικό υποχρεωτικό πεδίο και αυτό που φαίνεται όπου γίνεται χρήση της επαφής (έγγραφα κλπ).



Εικόνα 86. Νέα επαφή

7.14.7.2 Προσωπικό Φορέα

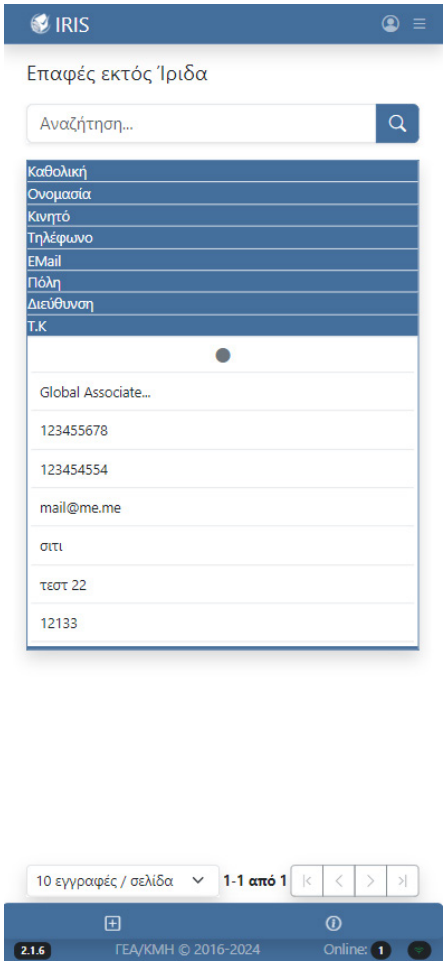
Λίστα με το προσωπικό του φορέα του χρήστη. Επιλέγοντας μια επαφή μπορούμε να δούμε κάποια βασικά στοιχεία και στοιχεία επικοινωνίας.



Εικόνα 87. Προσωπικό φορέα

7.14.7.3 Επαφές εκτός Ίριδα

Λίστα με τις επαφές του χρήστη εκτός Ίριδα. Επιλέγοντας μια επαφή μπορούμε να δούμε κάποια βασικά στοιχεία και στοιχεία επικοινωνίας. Χρήστες που είναι τοποθετημένοι στην ίδια θέση, έχουν κοινές επαφές. Οι επαφές αυτές είναι αξιοποιήσιμες στη επιλογή αποδεκτών ενός εγγράφου.

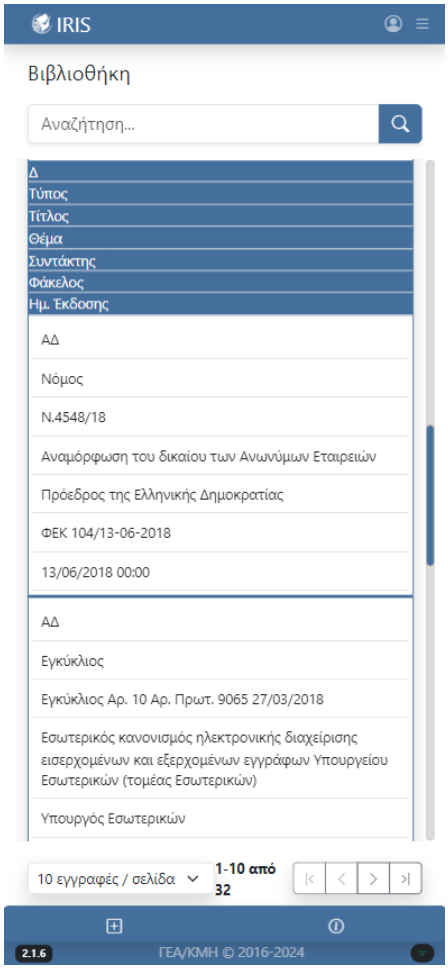


Εικόνα 88. Επαφές εκτός Ίριδα

7.14.8 Βιβλιοθήκη

Στη λίστα της βιβλιοθήκης υπάρχει ένας σελιδοποιημένος πίνακας με δυνατότητα αναζήτησης, με τα εξής πεδία:

- Διαβάθμιση εγγράφου βιβλιοθήκης
- Τύπος εγγράφου βιβλιοθήκης
- Τίτλος εγγράφου βιβλιοθήκης
- Θέμα εγγράφου βιβλιοθήκης
- Συντάκτης εγγράφου βιβλιοθήκης
- Φάκελος εγγράφου βιβλιοθήκης
- Ημερομηνία έκδοσης εγγράφου

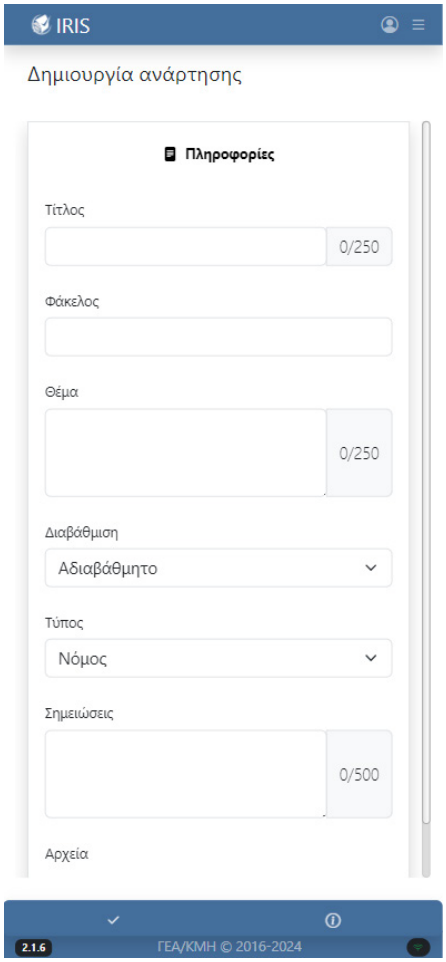


Εικόνα 89. Βιβλιοθήκη

7.14.8.1 Νέο Έγγραφο βιβλιοθήκης

Για τη Δημιουργία νέου Έγγραφου βιβλιοθήκης αρκεί να συμπληρώσουμε την παρακάτω φόρμα (με κόκκινο χρώμα φαίνονται τα υποχρεωτικά πεδία):

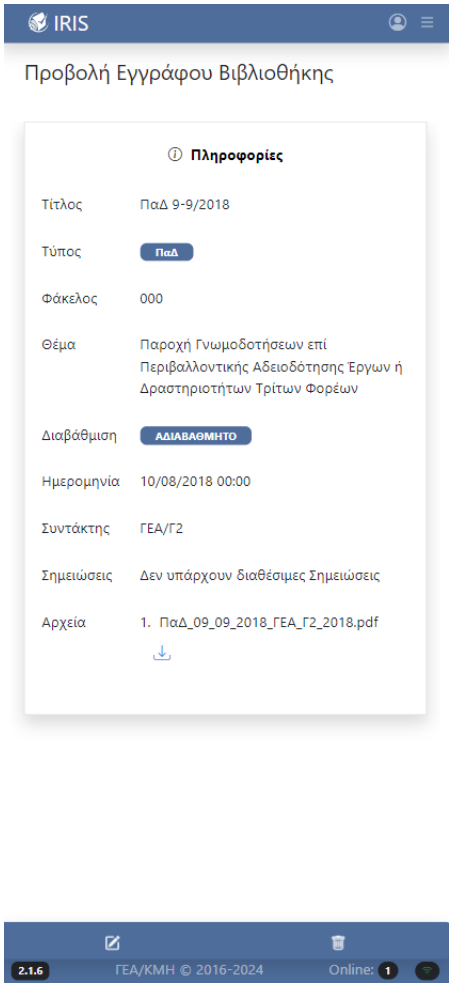
Με την εισαγωγή κάποιου Εγγράφου Βιβλιοθήκης, μπορεί οποιοσδήποτε χρήστης του συστήματος έπειτα να το χρησιμοποιήσει ως σχετικό σε έγγραφα κλπ.



Εικόνα 90. Νέο έγγραφο βιβλιοθήκης

7.14.8.2 Προβολή εγγράφου βιβλιοθήκης

Αποτυπώνονται σε λίστα τα έγγραφα της βιβλιοθήκης. Η λίστα αυτή ακολουθεί τη λογική που περιγράφηκε και στις λίστες εγγράφων.



Εικόνα 91. Προβολή εγγράφου βιβλιοθήκης

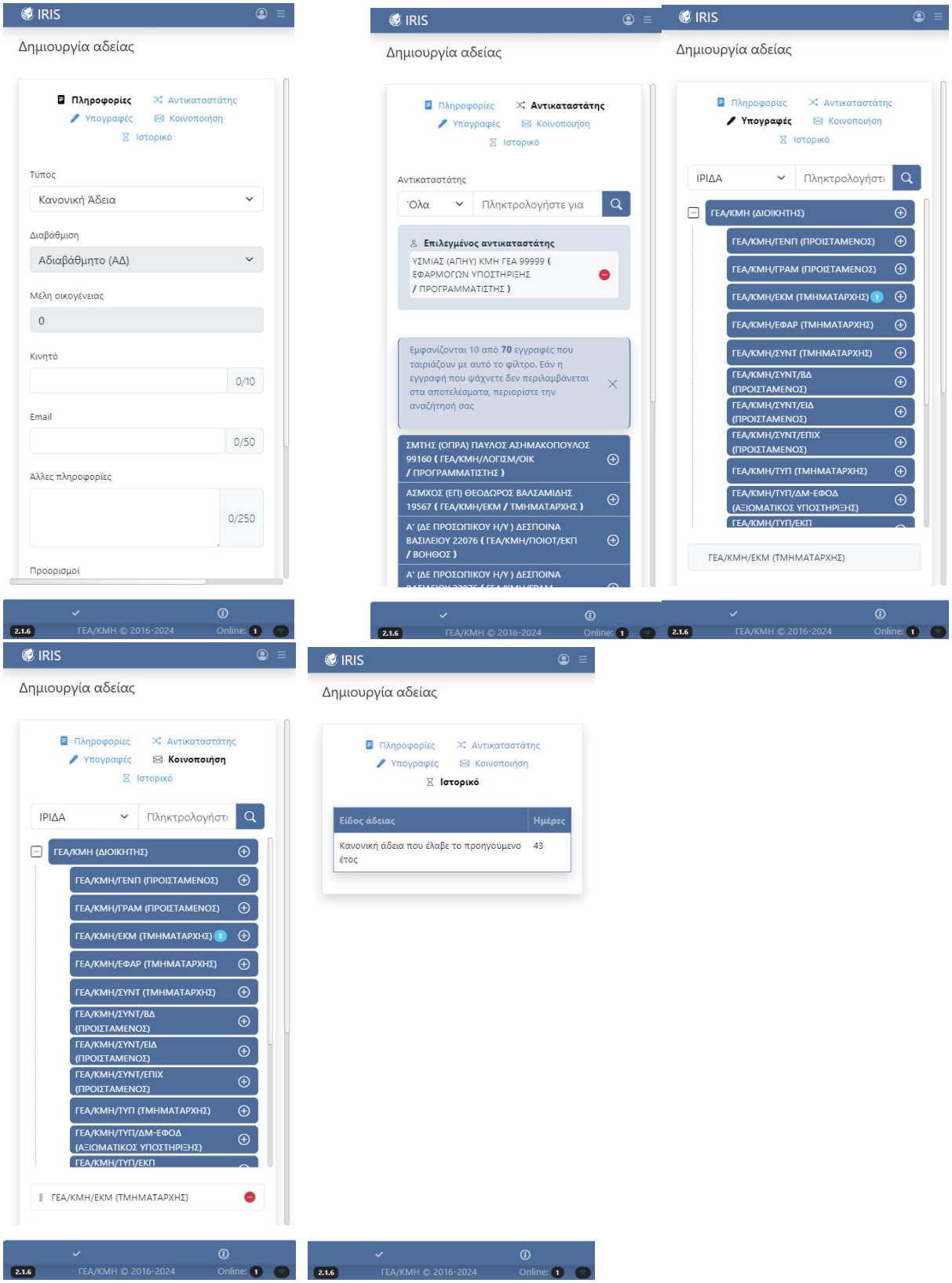
7.14.9 Άδειες

7.14.9.1 Νέα Αίτηση

Για τη Δημιουργία νέας αίτησης άδειας εκτός Ίριδα αρκεί να συμπληρώσουμε την παρακάτω φόρμα:

1. Στον τύπο της άδειας επιλέγουμε έναν από τους διαθέσιμους της λίστας.
2. Τα πεδία Διαβάθμιση και Μέλη οικογένειας ενεργοποιούνται εφόσον έχουμε έστω και έναν προορισμό εξωτερικού στην αίτηση.
3. Τα πεδία κινητό και Email προ συμπληρώνονται από τα στοιχεία που έχουμε δώσει στη Μονάδα μας.
4. Συμπληρώνονται τυχόν άλλες πληροφορίες για την αίτηση.
5. Κάθε άδεια πρέπει να έχει τουλάχιστον ένα προορισμό. Εδώ μπορούμε να επεξεργαστούμε ή να διαγράψουμε κάποιον προορισμό.
6. Μπορούμε να προσθέσουμε όσους προορισμούς επιθυμούμε. Η διεύθυνση είναι υποχρεωτική και η Ημ.Έναρξης δεν μπορεί να είναι μικρότερη από την λήξη του προηγούμενου
7. Εδώ αθροίζεται το σύνολο των ημερών της αίτησης. 8. Η επιλογή αντικαταστάτη είναι υποχρεωτική.
8. Συμπληρώνεται η σειρά των υπογραφόντων

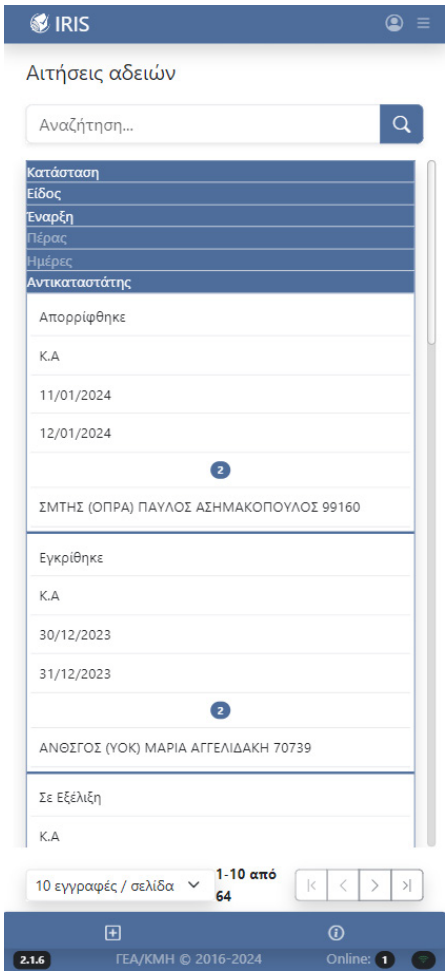
9. Αν το επιθυμούμε μπορούμε να κοινοποιήσουμε την αίτηση μας (εφόσον εγκριθεί) σε άλλο χρήστη(π.χ. Γρ. Προσωπικού Μονάδας)
10. Ιστορικό των αδειών που έχουμε λάβει κατά το τρέχων έτος , καθώς και οι ημέρες Κ.Α του προηγούμενου.



Εικόνα 92. Νέα αίτηση άδειας

7.14.9.2 Αιτήσεις αδειών

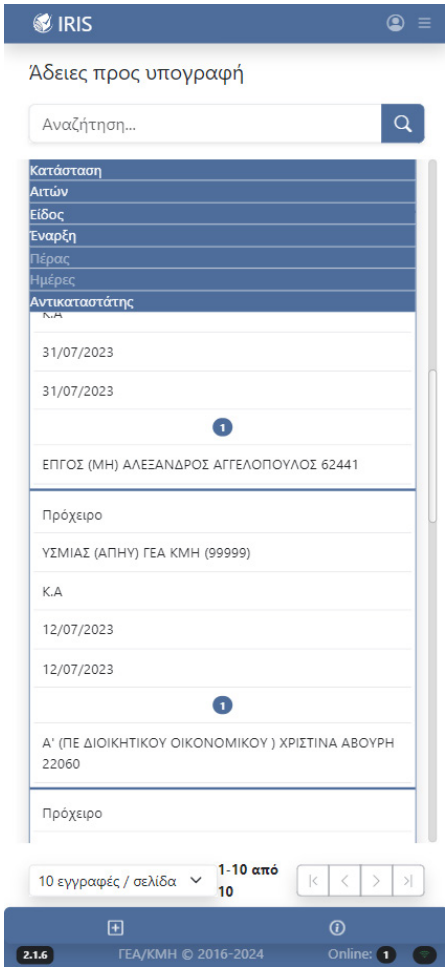
Εμφάνιση λίστας με όλες τις άδειες που έχει δημιουργήσει ο χρήστης ανεξάρτητα από την κατάστασή τους. Επιλέγοντας μια αίτηση από τη λίστα μεταβαίνουμε στην επισκόπηση της άδειας, όπου μπορούμε να ενημερωθούμε για το ποιος έχει υπογράψει κτλ.



Εικόνα 93. Αιτήσεις αδειών

7.14.9.3 Για Υπογραφή

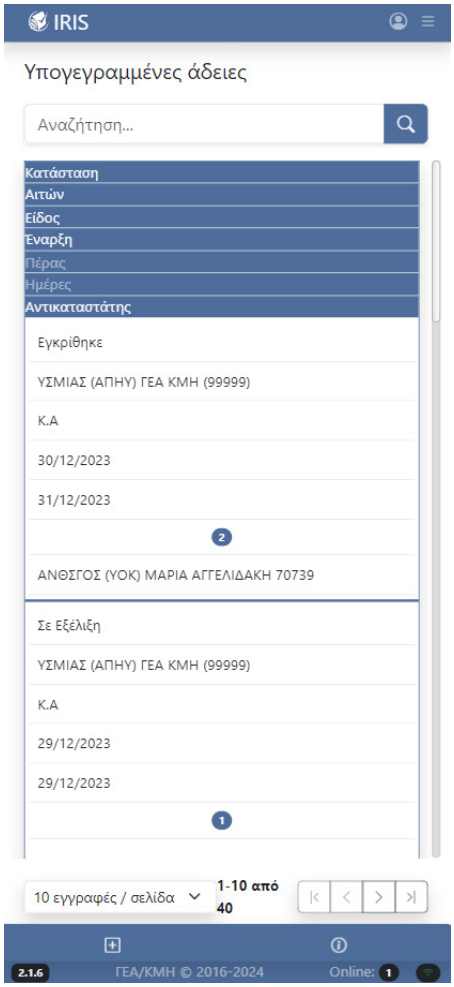
Εμφάνιση λίστας με όλες τις άδειες, στις οποίες ο χρήστης είναι ο επόμενος υπογράφων. Επιλέγοντας μια αίτηση από τη λίστα, μεταβαίνουμε στην επισκόπηση της άδειας, όπως και στην προηγούμενη παράγραφο, με τη διαφορά ότι έχουμε δύο διαθέσιμες ενέργειες: Υπογραφή και Απόρριψη. Με την υπογραφή η αίτηση δρομολογείται στον επόμενο στη λίστα των υπογραφών (εκτός αν είμαστε τελευταίος, όπου η άδεια εγκρίνεται και μεταβαίνει στις υπογεγραμμένες) , ενώ με την απόρριψη σταματάει η ροή της και ο συντάκτης της θα πρέπει να δημιουργήσει νέα αν το επιθυμεί.



Εικόνα 94. Για υπογραφή

7.14.9.4 Υπογεγραμμένες

Εμφάνιση λίστας με όλες τις Υπογεγραμμένες άδειες.

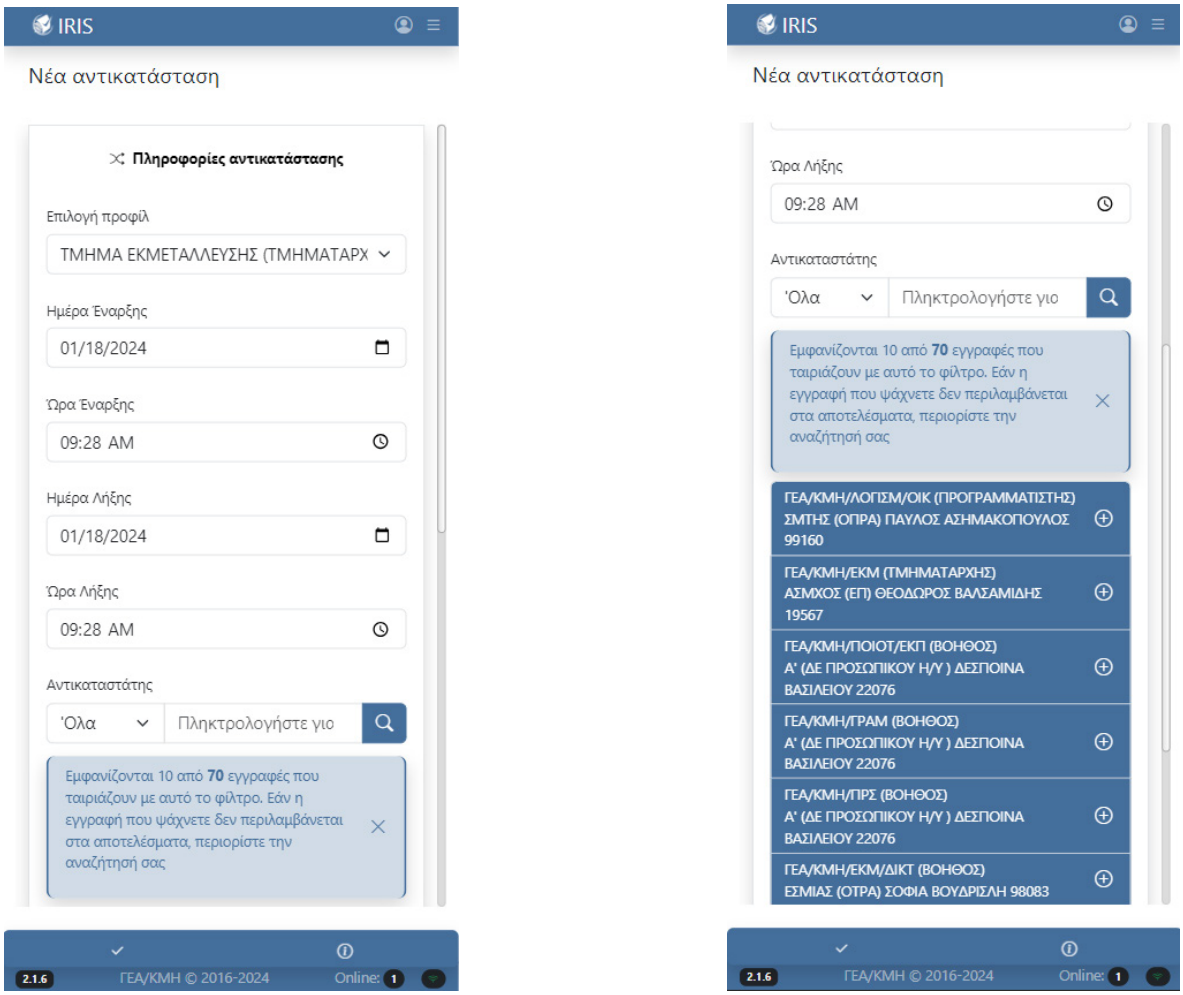


Εικόνα 95. Υπογεγραμμένες άδειες

7.14.10 Αντικαταστάσεις

7.14.10.1 Νέα Αντικατάσταση

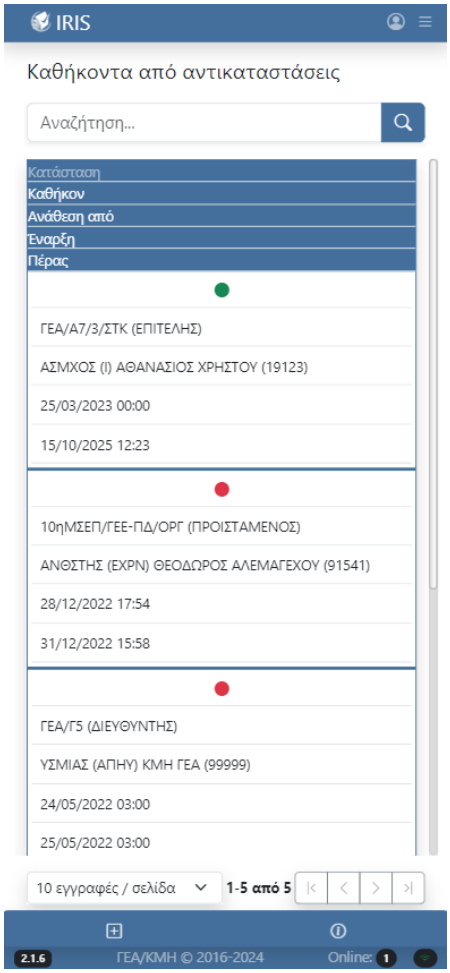
Για τη Δημιουργία Αντικατάστασης αρκεί να συμπληρώσουμε την παρακάτω φόρμα: Θα πρέπει να επιλέξουμε ένα από τα διαθέσιμα προφίλ, χρονικό πλαίσιο αντικατάστασης και υποχρεωτικά έναν αντικαταστάτη, όπως απεικονίζεται παραπάνω. Προσοχή: Με την Αντικατάσταση, ο επιλεγμένος αντικαταστάτης μας έχει πρόσβαση σε όλο το αρχείο μας στο Ίριδα και μπορεί να εκτελεί ενέργειες με τον επιλεγμένο ρόλο



Εικόνα 96. Νέα αντικατάσταση

7.14.10.2 Καθήκοντα από αντικαταστάσεις

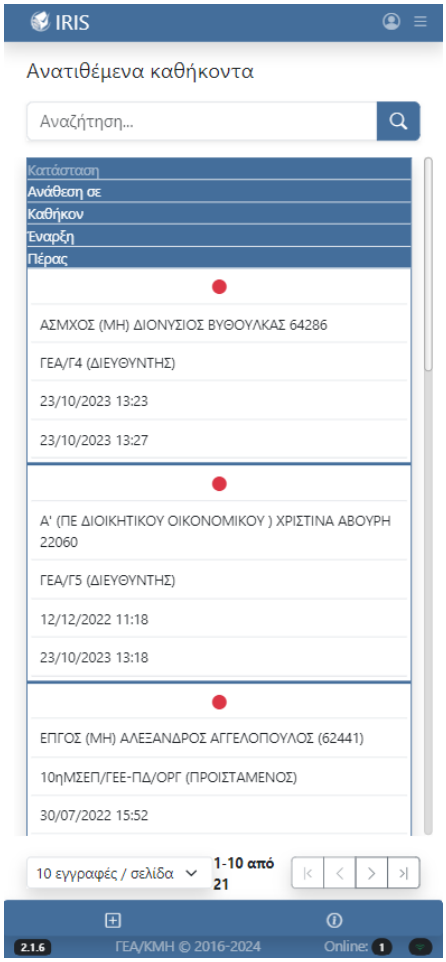
Εμφάνιση λίστας με όλα τα Καθήκοντα που έχουμε λάβει κατά καιρούς στο Ίριδα από αντικατάσταση, στην οποία μπορούμε να δούμε την κατάσταση της, τον χρήστη που μας ανέθεσε το καθήκον και το χρονικό πλαίσιο της. Δεν μπορούμε να επεξεργαστούμε αυτές τις εγγραφές.



Εικόνα 97. Καθήκοντα από αντικαταστάσεις

7.14.10.3 Αντιτιθέμενα καθήκοντα

Εμφάνιση λίστας με όλα τα Καθήκοντα που έχουμε αναθέσει κατά καιρούς στο Ίριδα σε κάποιον άλλο χρήστη. Εκεί μπορούμε να δούμε την κατάσταση της, τον χρήστη που επιλέξατε, το καθήκον και το χρονικό πλαίσιο της. Μπορούμε να επεξεργαστούμε μόνο τις ενεργές αντικαταστάσεις (πράσινη κατάσταση), τροποποιώντας οποιαδήποτε από τα πεδία της φόρμας ή ακόμα και λήγοντας την (από την γραμμή εργαλείων).



Εικόνα 98. Ανατιθέμενα καθήκοντα

7.14.11 Συντονισμοί / Διεκπεραιώσεις

Ο συντονισμός είναι μια λειτουργικότητα η οποία υποβοηθά το έργο των χρηστών που έχουν μεγάλο όγκο εισερχόμενης αλληλογραφίας.

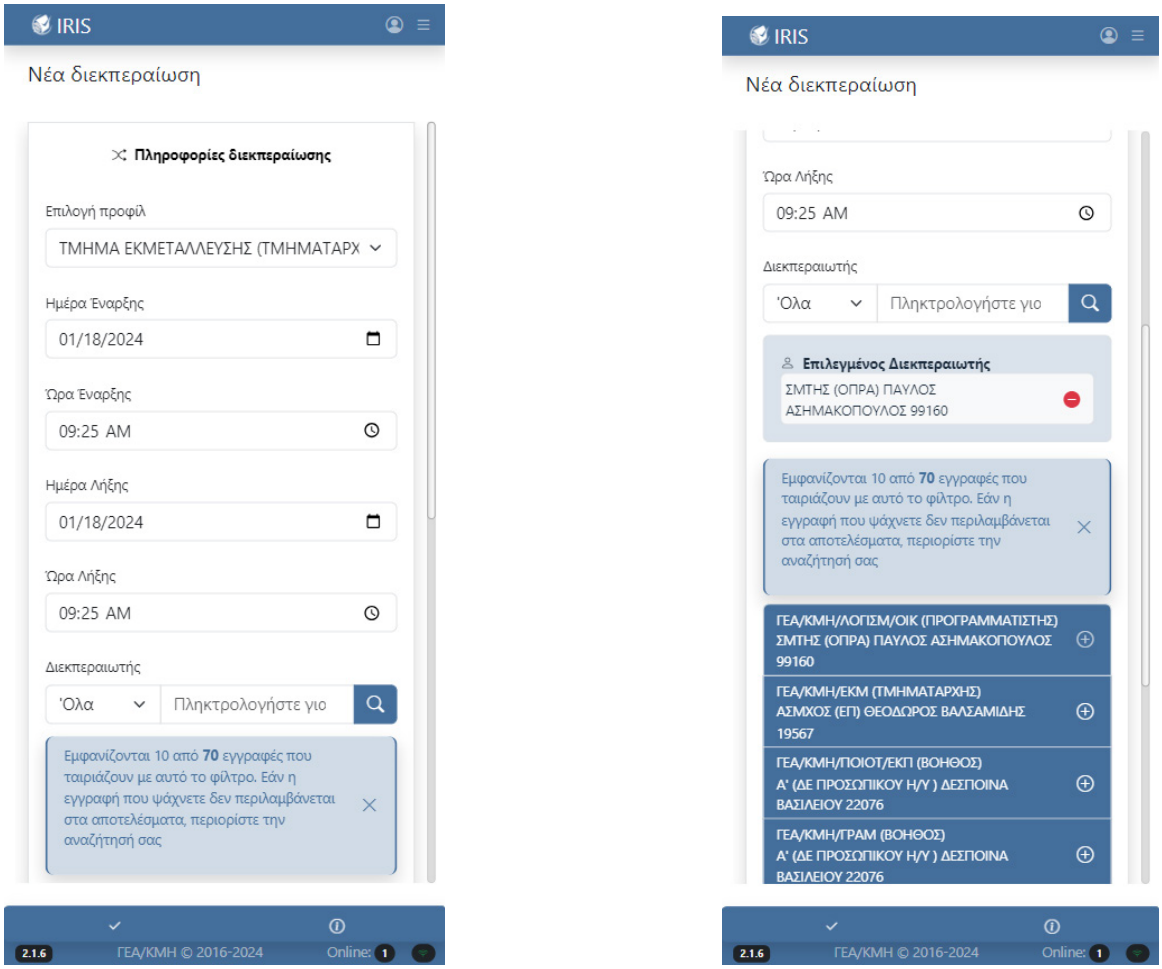
Γενικά, ισχύουν οι εξής παραδοχές:

- Συντονιστή μπορεί να ορίσει μόνο χρήστης με το “αρχαιότερο” καθήκον μιας θέσης. Π.χ στη θέση ΓΕΑ/Γ5 μπορεί μόνο ο Δντης, στη θέση ΓΕΑ/Γ5/1 μόνο ο Τμχης κ.ο.κ.
- Η ανάθεση μπορεί να είναι πολλαπλή, δηλαδή ένας χρήστης μπορεί να έχει πολλούς συντονιστές.
- Ο συντονισμός έχει μια ημερομηνία έναρξης και μία λήξης.
- Ένας συντονιστής μπορεί να οριστεί μόνο από έναν αναθέτων για ένα συγκεκριμένο χρονικό διάστημα.
- Ο συντονιστής, για το διάστημα που είναι ορισμένος, έχει πρόσβαση στο μενού Εισερχομένων εγγράφων (για ενέργεια, για κοινοποίηση και από εργασίες) του αναθέτοντος.
- Η ανάγνωση ενός εγγράφου από τον συντονιστή δεν αλλάζει την κατάσταση σε “αναγνωσμένο” - αυτή αλλάζει μόνο αν το έγγραφο διαβαστεί από τον αναθέτων.

- Ο αναθέτων σε όλα τα μενού των εργασιών του, εκτός από τις εργασίες που τον αφορούν, έχει πρόσβαση και στις εργασίες των συντονιστών του, τις οποίες μπορεί και να επεξεργαστεί.

7.14.11.1 Νέος συντονισμός

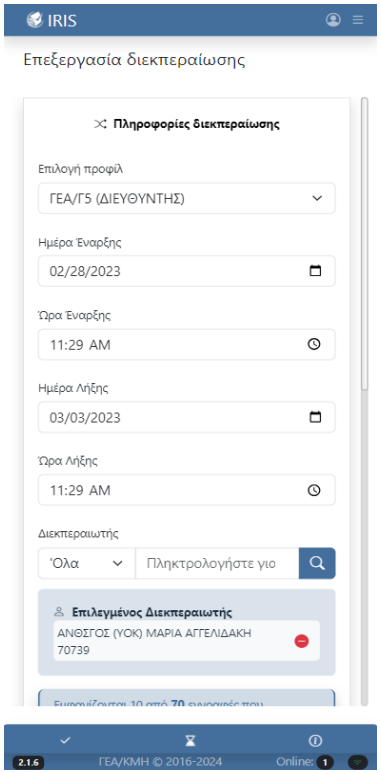
Για τη Δημιουργία Συντονισμού αρκεί να συμπληρώσουμε την παρακάτω φόρμα: Θα πρέπει να επιλέξουμε ένα από τα διαθέσιμα προφίλ, χρονικό πλαίσιο συντονισμού και υποχρεωτικά έναν αντικαταστάτη, όπως απεικονίζεται παραπάνω.



Εικόνα 99. Νέος συντονισμός

7.14.11.2 Επεξεργασία συντονισμού

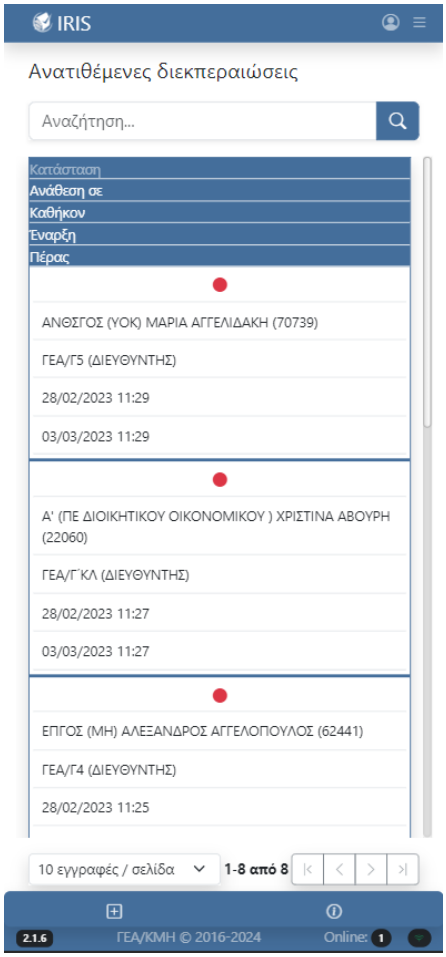
Όμοια με τη δημιουργία συντονισμού λειτουργεί και η επεξεργασία.



Εικόνα 100. Επεξεργασία συντονισμού

7.14.11.3 Ανατιθέμενοι συντονισμοί

Εμφάνιση λίστας με όλα τους συντονισμούς που έχουμε αναθέσει κατά καιρούς στο Ίριδα σε κάποιον άλλο χρήστη, στην οποία μπορούμε να δούμε την κατάσταση της, τον χρήστη τον οποίο επιλέξαμε, το καθήκον και το χρονικό πλαίσιο της. Μπορούμε να επεξεργαστούμε όλους τους συντονισμούς, τροποποιώντας οποιαδήποτε από τα πεδία της φόρμας ή ακόμα και λήγοντας τον (από την γραμμή εργαλείων).



Εικόνα 101. Ανατιθέμενοι συντονισμοί

8 Case studies και Best Practices

8.1 Πραγματικές περιπτώσεις μετατροπών React App σε Ionic

Στο σημερινό ταχέως εξελισσόμενο ψηφιακό τοπίο, η ανάγκη για απρόσκοπτες εφαρμογές πολλαπλών πλατφορμών έχει καταστεί υψίστης σημασίας. Η αυξανόμενη ζήτηση για κινητές εφαρμογές έχει οδηγήσει τους οργανισμούς να διερευνήσουν αποτελεσματικούς τρόπους επέκτασης των διαδικτυακών εφαρμογών τους σε κινητές πλατφόρμες, διατηρώντας παράλληλα μια συνεπή εμπειρία χρήστη. Η μετάβαση από το React στο Ionic έχει αναδειχθεί ως μια συναρπαστική λύση για την αντιμετώπιση αυτής της πρόκλησης.

Παρακάτω παρουσιάζονται πραγματικές μελέτες περιπτώσεων επιτυχημένων μεταβάσεων από εφαρμογές ιστού React σε Ionic, αναδεικνύοντας τα οφέλη από τη χρήση του πλαισίου Ionic για την επίτευξη ισοτιμίας χαρακτηριστικών και βελτιωμένων εμπειριών χρήστη σε όλες τις πλατφόρμες ιστού και κινητών. Οι παρακάτω περιπτώσεις μεταβάσεων μελετήθηκαν με σκοπό την επιτυχέστερη και περισσότερο ανώδυνη μετάβαση της εφαρμογής “Ιριδα”.

8.1.1 Udemy

Η Udemy, μια κορυφαία εκπαιδευτική πλατφόρμα, ξεκίνησε μια μετάβαση από τη διαδικτυακή εφαρμογή της που βασίζεται στο React στο Ionic για να διευρύνει την παρουσία της στα κινητά. Αξιοποιώντας τις δυνατότητες διαπλατφορμών του Ionic και τη συμβατότητα με τεχνολογίες ιστού, η Udemy μετέφερε αποτελεσματικά ένα σημαντικό τμήμα της βάσης κώδικά της. Η μετάβαση αυτή, είχε ως αποτέλεσμα μια ευέλικτη εφαρμογή για κινητά που προσέφερε στους χρήστες μια συνεπή εμπειρία μάθησης σε διάφορες συσκευές.

8.1.2 The Weather Channel

Το The Weather Channel, μια πλατφόρμα πρόγνωσης καιρού, επεδίωξε να προσφέρει μια απρόσκοπτη εμπειρία χρήστη σε όλες τις διαδικτυακές και κινητές συσκευές. Μεταφέροντας την εφαρμογή ιστού React στο Ionic, πέτυχαν με επιτυχία την ισοτιμία των χαρακτηριστικών, ενώ απλοποίησαν τις προσπάθειες ανάπτυξης και συντήρησης. Η μετάβαση επέτρεψε στο Weather Channel να προσφέρει μια ενοποιημένη εμπειρία στους χρήστες του.

8.1.3 MarketWatch

Το MarketWatch, ένας δικτυακός τόπος οικονομικών ειδήσεων και πληροφοριών, αναγνώρισε τη σημασία της εξυπηρέτησης του κινητού κοινού. Επέλεξαν τη μετάβαση από το React στο Ionic, αξιοποιώντας τα προκατασκευασμένα στοιχεία UI του Ionic και τα plugins για οικονομικά δεδομένα. Αυτή η μετάβαση επέτρεψε στο MarketWatch να δημιουργήσει μια responsive εφαρμογή για κινητά με ευκολία.

8.1.4 FitBod:

Η FitBod, μια εφαρμογή γυμναστικής που ειδικεύεται σε εξατομικευμένες ρουτίνες προπόνησης, επεκτάθηκε από μια web εφαρμογή React σε μια εφαρμογή για κινητά με πολλές δυνατότητες χρησιμοποιώντας το Ionic. Η μετάβαση εξασφάλισε μια συνεπή γλώσσα σχεδιασμού και ροή χρήστη, ενισχύοντας τη χρηστικότητα μέσω της υποστήριξης χειρονομιών αφής του Ionic και της ενσωμάτωσης εγγενών χαρακτηριστικών της συσκευής.

8.1.5 Untappd

Το Untappd, ένα κοινωνικό δίκτυο για τους λάτρεις της μπύρας, είχε ως στόχο να αναπαράγει τη λειτουργικότητα της διαδικτυακής πλατφόρμας του σε κινητές συσκευές. Μεταπήδησαν από το React στο Ionic, διευκολύνοντας τη γρήγορη ανάπτυξη και την ενσωμάτωση με τις υπάρχουσες backend υπηρεσίες. Τα εργαλεία του Ionic για την ανάπτυξη UI και τον έλεγχο ταυτότητας χρηστών απλοποίησαν τη διαδικασία μετάβασης, προσεγγίζοντας ένα ευρύτερο κοινό από λάτρεις της μπύρας.

Οι μελέτες περίπτωσης που παρουσιάστηκαν παραπάνω αναδεικνύουν την αποτελεσματικότητα της μετάβασης από το React στο Ionic για την επίτευξη μιας ενοποιημένης λύσης πολλαπλών πλατφορμών με παράλληλη επαναχρησιμοποίηση κώδικα και πόρων. Η συμβατότητα του Ionic με τεχνολογίες ιστού, το εκτεταμένο οικοσύστημα πρόσθετων στοιχείων και η υποστήριξη εγγενών συσκευών το καθιστούν μια εγγυημένη επιλογή ως λύση για οργανισμούς που επιδιώκουν να επεκτείνουν την παρουσία τους σε κινητά. Αυτά τα παραδείγματα από τον πραγματικό κόσμο υπογραμμίζουν την ευελιξία και την αποτελεσματικότητα του Ionic στη διευκόλυνση επιτυχημένων μεταβάσεων εφαρμογών ιστού React σε εφαρμογές για κινητά, βελτιώνοντας τελικά τις εμπειρίες των χρηστών και εξορθολογίζοντας τις προσπάθειες ανάπτυξης σε έναν όλο και περισσότερο κινητοκεντρικό κόσμο.

8.2 Συμπεράσματα

Στο σημερινό δυναμικό ψηφιακό τοπίο, όπου οι χρήστες απαιτούν απρόσκοπτες και ευέλικτες εμπειρίες σε διάφορες πλατφόρμες, η μετάβαση από διαδικτυακές εφαρμογές σε εφαρμογές για κινητά έχει γίνει επιτακτική για τις επιχειρήσεις και τους οργανισμούς. Μια ιδιαίτερα αποτελεσματική προσέγγιση για την επίτευξη αυτής της μετάβασης είναι η μετάβαση από το React, μια δημοφιλής βιβλιοθήκη JavaScript για την κατασκευή διεπαφών ιστού, στο Ionic, ένα ισχυρό framework για την ανάπτυξη εφαρμογών για κινητές συσκευές πολλαπλών πλατφορμών.

Το κίνητρο πίσω από αυτή τη μετάβαση είναι σαφές: δίνει τη δυνατότητα στους οργανισμούς να αξιοποιήσουν την υπάρχουσα βάση κώδικα, να διατηρήσουν μια ενιαία εμπειρία χρήστη και να προσεγγίσουν ένα ευρύτερο κοινό επεκτείνοντας την παρουσία τους τόσο σε πλατφόρμες ιστού όσο και σε κινητές πλατφόρμες. Παρακάτω διερευνώνται οι βέλτιστες πρακτικές και τα διδάγματα που αντλήθηκαν από την επιτυχημένη μεταβάση από το React στο Ionic. Επίσης παρέχεται μια ολοκληρωμένη κατανόηση των πλεονεκτημάτων, των προκλήσεων και των βέλτιστων πρακτικών που σχετίζονται με αυτή τη μετάβαση.

8.2.1 Συνολικός σχεδιασμός

Μια επιτυχημένη μετάβαση ξεκινά με ένα ολοκληρωμένο σχεδιασμό. Αυτό περιλαμβάνει μια βαθιά ανάλυση της υφιστάμενης εφαρμογής ιστού React, κατανοώντας την αρχιτεκτονική, τα στοιχεία και τις εξαρτήσεις της. Είναι ζωτικής σημασίας ο καθορισμός σαφών στόχων και σκοπών για τη διαδικασία μετάβασης, όπως η επίτευξη ισοτιμίας χαρακτηριστικών στις πλατφόρμες κινητών, η βελτίωση των επιδόσεων ή η επέκταση της βάσης χρηστών. Πραγματοποιώντας ενδελεχή σχεδιασμό, μπορείτε να εντοπίσετε πιθανές προκλήσεις, να

κατανείμετε αποτελεσματικά τους πόρους και να καταρτίσετε έναν οδικό χάρτη για το έργο μετάβασης.

8.2.2 Επαναχρησιμοποίηση της βάσης κώδικα

Η επαναχρησιμοποίηση του υπάρχοντος κώδικα της διαδικτυακής εφαρμογής React είναι μια βέλτιστη πρακτική που μπορεί να επιταχύνει σημαντικά τη διαδικασία μετάβασης. Στοιχεία όπως η επιχειρησιακή λογική, η άντληση δεδομένων και η διαχείριση κατάστασης μπορούν συχνά να ενσωματωθούν απρόσκοπτα στην εφαρμογή Ionic για κινητά. Αυτή η προσέγγιση όχι μόνο εξοικονομεί χρόνο ανάπτυξης, αλλά βοηθά επίσης στη διατήρηση της συνοχής και μειώνει τον κίνδυνο εισαγωγής νέων σφαλμάτων. Ωστόσο, είναι σημαντικό να αξιολογείται η καταλληλότητα κάθε στοιχείου για επαναχρησιμοποίηση και να γίνονται οι απαραίτητες προσαρμογές ανάλογα με τις ανάγκες.

8.2.3 Προσέγγιση βασισμένη σε στοιχεία (components)

Η υιοθέτηση μιας αρχιτεκτονικής βασισμένης σε συστατικά είναι ζωτικής σημασίας τόσο στο React όσο και στο Ionic. Το React είναι εγγενώς βασισμένο σε components και το Ionic προωθεί επίσης αυτή την προσέγγιση. Όταν τα components οργανώνονται συστηματικά, η αντιστοίχιση στοιχείων και λειτουργιών UI από το React στο Ionic γίνεται πιο απλή. Η συνέπεια στην ονομασία και τη δομή των συστατικών μεταξύ των δύο πλαισίων διευκολύνει μια ομαλότερη διαδικασία μετάβασης και διασφαλίζει ότι η διεπαφή χρήστη παραμένει συνεκτική.

8.2.4 Συμβατότητα με τεχνολογίες ιστού

Ένα από τα βασικά πλεονεκτήματα του Ionic είναι η συμβατότητά του με τεχνολογίες ιστού, όπως η HTML, η CSS και η JavaScript (ή η TypeScript). Αυτή η συμβατότητα επιτρέπει στους προγραμματιστές να προσαρμόζουν την υπάρχουσα βάση κώδικα ιστού στο Ionic με σχετική ευκολία. Η εξοικείωση των τεχνολογιών ιστού και στα δύο frameworks απλοποιεί τη διαδικασία μετάβασης, διευκολύνοντας τους προγραμματιστές να μεταβούν από το React στο Ionic χωρίς απότομη καμπύλη εκμάθησης.

8.2.5 Αξιοποίηση τα στοιχείων του Ionic UI

Το Ionic προσφέρει μια πλούσια βιβλιοθήκη προ-κατασκευασμένων στοιχείων UI που έχουν σχεδιαστεί ειδικά για την ανάπτυξη εφαρμογών για κινητά. Η αξιοποίηση αυτών των στοιχείων μπορεί να επιταχύνει τη δημιουργία της διεπαφής χρήστη της εφαρμογής για κινητά. Υιοθετώντας τα στοιχεία UI του Ionic, όχι μόνο εξοικονομείται χρόνο ανάπτυξης, αλλά και εξασφαλίζεται μια συνεπή και οπτικά ελκυστική διεπαφή χρήστη σε διαφορετικές συσκευές και πλατφόρμες. Αυτή η συνέπεια συμβάλλει στη θετική εμπειρία του χρήστη.

8.2.6 Χαρακτηριστικά εγγενούς συσκευής

Το Ionic παρέχει ένα ευρύ φάσμα πρόσθετων που παρέχουν πρόσβαση σε εγγενή χαρακτηριστικά της συσκευής, όπως η κάμερα, ο γεωγραφικός εντοπισμός, οι ειδοποιήσεις push και πολλά άλλα. Η αξιοποίηση αυτών των plugins επιτρέπει τη βελτίωση της λειτουργικότητα της εφαρμογής για κινητά, με σκοπό να παρέχεται στους χρήστες μια πιο καθηλωτική εμπειρία. Η ενσωμάτωση των εγγενών λειτουργιών της συσκευής είναι ιδιαίτερα

σημαντική για τις εφαρμογές κινητών τηλεφώνων, καθώς μπορεί να διαφοροποιήσει την εφαρμογή από τις εναλλακτικές λύσεις μόνο για τον ιστό.

8.2.7 Δοκιμές και διασφάλιση ποιότητας

Οι αυστηρές δοκιμές και η διασφάλιση ποιότητας αποτελούν κρίσιμα στοιχεία μιας επιτυχημένης μετάβασης. Μετά την ολοκλήρωση της μετάβασης, θα πρέπει να διεξάγονται ενδελεχείς δοκιμές σε διάφορες συσκευές, λειτουργικά συστήματα και μεγέθη οθόνης για τον εντοπισμό και την αντιμετώπιση προβλημάτων συμβατότητας, προβλημάτων διάταξης και συμφόρησης επιδόσεων. Αυτή η φάση διασφαλίζει ότι η μεταστεγασμένη εφαρμογή λειτουργεί όπως προβλέπεται και παρέχει μια συνεπή εμπειρία χρήστη. Τα αυτοματοποιημένα εργαλεία δοκιμών και οι χειροκίνητες δοκιμές από τους μηχανικούς QA είναι αμφότερα πολύτιμα κατά τη διάρκεια αυτού του σταδίου.

8.2.8 Συνέπεια της εμπειρίας χρήστη

Η διατήρηση μιας συνεπούς εμπειρίας χρήστη μεταξύ της διαδικτυακής και της κινητής έκδοσης της εφαρμογής αποτελεί πρωταρχικό στόχο. Οι χρήστες θα πρέπει να αισθάνονται εξοικειωμένοι με το σχεδιασμό και τη λειτουργικότητα της εφαρμογής, ανεξάρτητα από την πλατφόρμα που χρησιμοποιούν. Η συνέπεια στα στοιχεία σχεδιασμού, τα πρότυπα πλοήγησης και τις ροές χρήστη βοηθά στην προώθηση της δέσμευσης και της ικανοποίησης των χρηστών. Οι αποκλίσεις στην εμπειρία χρήστη μπορεί να οδηγήσουν σε σύγχυση και δυσαρέσκεια των χρηστών.

8.2.9 Βελτιστοποίηση επιδόσεων

Οι χρήστες κινητών τηλεφώνων έχουν διαφορετικές προσδοκίες και περιορισμούς σε σύγκριση με τους χρήστες του διαδικτύου. Ως εκ τούτου, η βελτιστοποίηση των επιδόσεων είναι ζωτικής σημασίας για τη διασφάλιση μιας ομαλής και ευέλικτης εφαρμογής για κινητά. Κατά τη διάρκεια της μετάβασης, οι προγραμματιστές θα πρέπει να εντοπίζουν πιθανά σημεία συμφόρησης επιδόσεων, όπως η αργή φόρτωση στοιχείων ή η αναποτελεσματική άντληση δεδομένων, και να τα αντιμετωπίζουν. Για τη βελτιστοποίηση των επιδόσεων της εφαρμογής για κινητά μπορούν να χρησιμοποιηθούν τεχνικές όπως η τεμπέλικη φόρτωση (lazy loading), η προσωρινή αποθήκευση δεδομένων και η διάσπαση κώδικα.

8.2.10 Τακτικές ενημερώσεις και συντήρηση

Μόλις ολοκληρωθεί η μετάβαση και ξεκινήσει η εφαρμογή για κινητά, είναι σημαντικό να δεσμευτείτε για τακτικές ενημερώσεις και συντήρηση. Οι πλατφόρμες, τα λειτουργικά συστήματα και οι συσκευές κινητής τηλεφωνίας συνεχίζουν να εξελίσσονται και η εφαρμογή πρέπει να προσαρμόζεται σε αυτές τις αλλαγές. Οι τακτικές ενημερώσεις όχι μόνο διατηρούν την εφαρμογή ενημερωμένη με τις τελευταίες τεχνολογίες και διορθώσεις ασφαλείας, αλλά διασφαλίζουν επίσης ότι παραμένει συμβατή με νεότερες συσκευές και εκδόσεις λειτουργικών συστημάτων. Η συνεχής συντήρηση περιλαμβάνει επίσης την αντιμετώπιση των ανατροφοδοτήσεων των χρηστών, τη διόρθωση σφαλμάτων και την προσθήκη νέων χαρακτηριστικών για να κρατάει τους χρήστες απασχολημένους.

8.2.11 Διαλειτουργικές ομάδες

Η αποτελεσματική επικοινωνία και συνεργασία στο πλαίσιο διαλειτουργικών ομάδων είναι απαραίτητες για ένα επιτυχημένο έργο μετάβασης. Οι προγραμματιστές, οι σχεδιαστές, οι μηχανικοί διασφάλισης ποιότητας και οι διαχειριστές έργου θα πρέπει να συνεργάζονται στενά καθ' όλη τη διάρκεια της διαδικασίας μετάβασης. Η σαφής επικοινωνία βοηθά στην άμεση αντιμετώπιση των προκλήσεων, στην ευθυγράμμιση των στοιχείων σχεδιασμού και στη διασφάλιση ότι η εφαρμογή για κινητά ανταποκρίνεται στις προσδοκίες των χρηστών. Μια καλά συντονισμένη ομάδα διασφαλίζει ότι το έργο μετάβασης παραμένει εντός τροχιάς και παραδίδει μια κινητή εφαρμογή υψηλής ποιότητας.

8.2.12 Προοδευτική μετάβαση

Αυτή η προσέγγιση περιλαμβάνει τη σταδιακή μετανάστευση τμημάτων ή ενοτήτων της εφαρμογής σας React στο Ionic. Επιτρέπει να ελαχιστοποιηθεί η διακοπή της τρέχουσας ανάπτυξης και διασφαλίζει ότι η διαδικασία μετάβασης είναι διαχειρίσιμη. Με τη διάσπαση της μετάβασης σε μικρότερες, διαχειρίσιμες εργασίες, μπορείτε να διατηρήσετε καλύτερο έλεγχο του χρονοδιαγράμματος του έργου και να δώσετε προτεραιότητα στα κρίσιμα στοιχεία.

8.2.13 Συγκριτική αξιολόγηση επιδόσεων

Πριν και μετά τη μετάβαση, χρειάζεται η διεξαγωγή μιας ολοκληρωμένης συγκριτικής αξιολόγησης επιδόσεων για τη μέτρηση βασικών δεικτών επιδόσεων, όπως οι χρόνοι φόρτωσης εφαρμογών, οι χρόνοι απόκρισης και η χρήση μνήμης. Τα δεδομένα αυτά βοηθούν στην ποσοτικοποίηση των βελτιώσεων που επιτεύχθηκαν μέσω της μετάβασης και τον εντοπισμό περιοχών που ενδέχεται να χρειάζονται ακόμη βελτιστοποίηση. Η συνεχής παρακολούθηση των επιδόσεων μετά τη μετεγκατάσταση διασφαλίζει ότι η εφαρμογή για κινητά παραμένει ευέλικτη και αποδοτική.

8.2.14 Συμβατότητα εκδόσεων

Θα πρέπει να εξασφαλίζεται ότι οι εκδόσεις της React και του Ionic που χρησιμοποιούνται είναι συμβατές. Και τα δύο πλαίσια λαμβάνουν ενημερώσεις και αλλαγές με την πάροδο του χρόνου, οι οποίες μπορεί να επηρεάσουν τη διαδικασία μετάβασης. Η διατήρηση των εξαρτήσεων ενημερωμένων αλλά και η γνώση για τυχόν διαφορές αλλαγές μεταξύ των εκδόσεων είναι ιδιαίτερα σημαντικό. Η ευθυγράμμιση με τις τελευταίες εκδόσεις τόσο της React όσο και του Ionic μπορεί να βοηθήσει στην αποφυγή προβλημάτων συμβατότητας και ευπαθειών ασφαλείας.

8.2.15 Δοκιμές σε διασταυρούμενα προγράμματα περιήγησης

Εκτός από τις δοκιμές διασταυρούμενων πλατφορμών, εκτελέστε ενδελεχείς δοκιμές διασταυρούμενων προγραμμάτων περιήγησης στην έκδοση web της εφαρμογής. Η μετάβαση στο Ionic μπορεί να έχει ακούσιες συνέπειες στον τρόπο λειτουργίας της εφαρμογής ιστού σε διαφορετικά προγράμματα περιήγησης. Θα πρέπει να εξασφαλίζεται ότι η διαδικτυακή εφαρμογή παραμένει συμβατή με τα σημαντικότερα προγράμματα περιήγησης, όπως το Chrome, το Firefox, το Safari και το Edge, για τη διατήρηση μιας συνεπούς εμπειρίας χρήσης για τους χρήστες του διαδικτύου.

8.2.16 Δεδομένα χρήστη και πιστοποίηση ταυτότητας

Χρειάζεται ιδιαίτερη προσοχή στα δεδομένα χρηστών και στους μηχανισμούς ελέγχου ταυτότητας κατά τη διάρκεια της μετάβασης. Πρέπει να εξασφαλιστεί η απρόσκοπτη μετάβαση για τους χρήστες, μεταφέροντας τους λογαριασμούς χρηστών, τις προτιμήσεις και τα δεδομένα συνόδου. Ακόμα θα πρέπει να διατηρηθούν τα πρότυπα ασφάλειας δεδομένων και προστασίας της ιδιωτικής ζωής κατά τη διάρκεια αυτής της διαδικασίας. Τέλος θα πρέπει να γίνουν με προσοχή τυχόν αλλαγές στις μεθόδους ελέγχου ταυτότητας για την αποφυγή της διακοπής των υφιστάμενων συνόδων χρήστη.

8.2.17 Τεκμηρίωση και μεταφορά γνώσεων

Αναγκαία είναι η τεκμηριώστε της διαδικασίας μετάβασης. Αυτή η τεκμηρίωση θα πρέπει να περιλαμβάνει λεπτομέρειες σχετικά με τις αλλαγές στον κώδικα, τις αντιστοιχίσεις συστατικών και τυχόν προσαρμογές που έγιναν κατά τη διάρκεια της μετάβασης. Επιπλέον, έτσι διασφαλίζεται ότι τα μέλη της ομάδας έχουν πρόσβαση σε αυτή την τεκμηρίωση. Αυτό είναι ζωτικής σημασίας για τη διατήρηση της βιωσιμότητας του έργου και για να μπορέσουν τα νέα μέλη της ομάδας να κατανοήσουν τη μεταφερόμενη βάση κώδικα.

8.2.18 Σχέδιο επαναφοράς

Ενώ ο στόχος είναι μια επιτυχημένη μετάβαση, είναι σημαντικό να υπάρχει ένα σχέδιο επαναφοράς σε περίπτωση που προκύψουν απροσδόκητα προβλήματα. Καθορίζοντας έτσι, σαφή βήματα για την επαναφορά στην προηγούμενη κατάσταση της εφαρμογής, εάν είναι απαραίτητο. Η ύπαρξη ενός καλά προετοιμασμένου σχεδίου επαναφοράς ελαχιστοποιεί τον χρόνο διακοπής λειτουργίας και τις πιθανές διακοπές για τους χρήστες σε περίπτωση που η μετάβαση αντιμετωπίσει ανυπέρβλητες προκλήσεις.

8.2.19 Ανατροφοδότηση χρηστών και επανάληψη

Μετά την κυκλοφορία της μεταφερθείσας εφαρμογής, χρειάζεται η συλλογή των σχόλια των χρηστών. Η ανατροφοδότηση των χρηστών μπορεί να αποκαλύψει ζητήματα ευχρηστίας, σφάλματα ή ανεκπλήρωτες ανάγκες που μπορεί να μην ήταν εμφανείς κατά τη διάρκεια της ανάπτυξης. Η συνεχής βελτίωση με βάση τις πληροφορίες των χρηστών είναι ζωτικής σημασίας για τη διατήρηση της δέσμευσης και της ικανοποίησης των χρηστών από την εφαρμογή για κινητά.

8.2.20 Εκπαίδευση και ανάπτυξη δεξιοτήτων

Χρειάζεται η επένδυση στην κατάρτιση και την ανάπτυξη δεξιοτήτων για την ομάδα ανάπτυξης. Εάν τα μέλη της ομάδας δεν είναι ήδη εξοικειωμένα με το Ionic, θα πρέπει να παρέχονται πόροι και εκπαιδευτικές συνεδρίες για να ενισχύσεται η επάρκειά τους στο Ionic. Η ανάπτυξη ειδικής τεχνογνωσίας για το Ionic εντός της ομάδας εξασφαλίζει την αποτελεσματική ανάπτυξη και συντήρηση της εφαρμογής για κινητά και ενθαρρύνει την ανταλλαγή γνώσεων.

Η ενσωμάτωση αυτών των πρόσθετων βέλτιστων πρακτικών, συμπερασμάτων και διδαγμάτων στο έργο μετάβασης θα συμβάλει σε μια πιο ομαλή και επιτυχημένη μετάβαση από το React στο Ionic. Κάθε μία από αυτές τις πτυχές, από την προοδευτική μετάβαση έως την τεκμηρίωση και τον προγραμματισμό επαναφοράς, διαδραματίζει κρίσιμο ρόλο στη

διασφάλιση μιας επιτυχημένης μετάβασης που ανταποκρίνεται στις προσδοκίες των χρηστών και στους επιχειρηματικούς στόχους.

8.3 Στρατηγικές για τη συντήρηση και την ενημέρωση της εφαρμογής για κινητά τηλέφωνα

Στο διαρκώς εξελισσόμενο τοπίο των εφαρμογών κινητής τηλεφωνίας, η ανάπτυξη μιας επιτυχημένης εφαρμογής κινητής τηλεφωνίας είναι μόνο η αρχή ενός βαθύτερου ταξιδιού. Η διατήρηση και η ενημέρωση μιας εφαρμογής για κινητά με την πάροδο του χρόνου είναι μια περίπλοκη διαδικασία που απαιτεί προσεκτικό σχεδιασμό, προληπτικές στρατηγικές και βαθιά κατανόηση των προσδοκιών των χρηστών. Με εκατομμύρια εφαρμογές να διεκδικούν την προσοχή στα καταστήματα εφαρμογών, η διατήρηση της κινητής εφαρμογής σε συνάφεια και ανταγωνιστικότητα είναι επιτακτική ανάγκη για μακροπρόθεσμη επιτυχία. Παρακάτω αναλύονται κάποιες στρατηγικές και βέλτιστες πρακτικές για τη συντήρηση και την ενημέρωση των εφαρμογών για κινητά, αντλώντας παραδείγματα από τον πραγματικό κόσμο και διδάγματα από κορυφαίους οργανισμούς στο χώρο των εφαρμογών για κινητά. Εξετάζοντας τις εμπειρίες εταιρειών όπως το Facebook, το Instagram, η Airbnb και το Netflix, στοχεύουμε στην παροχή μιας ολοκληρωμένης επισκόπησης των βασικών προβληματισμών και των αποδεδειγμένων στρατηγικών που επιτρέπουν τη βιώσιμη ανάπτυξη και τη συνεχή βελτίωση των εφαρμογών για κινητά τηλέφωνα. Είτε μια νεοσύστατη επιχείρηση που θέλει να εδραιώσει μια ισχυρή παρουσία είτε μια καθιερωμένη εταιρεία που επιδιώκει να διατηρήσει και να δεσμεύσει τη βάση των χρηστών της, οι παρακάτω στρατηγικές μπορούν να εφαρμοστούν και να έχουν άμεσο αντίκτυπο στις πωλήσεις - downloads της εκάστοτε εφαρμογής..

8.3.1 Συνεχής παρακολούθηση και ανατροφοδότηση

Για να διασφαλίσουμε τη συνεχή επιτυχία της εφαρμογής μας για κινητά, είναι ζωτικής σημασίας να δημιουργήσουμε ένα σύστημα συνεχούς παρακολούθησης. Θα πρέπει να παρακολουθούμε τακτικά τις επιδόσεις της εφαρμογής μας μέσω εργαλείων ανάλυσης, αναλύοντας δείκτες όπως η δέσμευση των χρηστών, τα ποσοστά διατήρησης και τα ποσοστά μετατροπής. Παράλληλα, πρέπει να συλλέγουμε ενεργά τα σχόλια των χρηστών μέσω κριτικών της εφαρμογής, έρευνας και κοινωνικών δικτύων, προκειμένου να κατανοήσουμε τις προτιμήσεις και τα προβλήματά τους. Με αυτήν την προσέγγιση, βασιζόμαστε σε δεδομένα για να ανιχνεύσουμε άμεσα πιθανά προβλήματα ή περιοχές που χρειάζονται βελτίωση. Η ανταπόκριση στα σχόλια των χρηστών και τα δεδομένα επίδοσης μας επιτρέπει να λαμβάνουμε ενημερώσεις και βελτιώσεις με βάση την κατάλληλη και τεκμηριωμένη πληροφορία, προκειμένου να διατηρήσουμε την εφαρμογή μας ανταγωνιστική και επικεντρωμένη στις ανάγκες των χρηστών.

8.3.2 Ευέλικτη ανάπτυξη και επανάληψη

Για να διασφαλίσουμε την αποτελεσματική συντήρηση και ενημέρωση των εφαρμογών, είναι απαραίτητο να υιοθετήσουμε μια ευέλικτη προσέγγιση στην ανάπτυξη. Οι ευέλικτες μεθοδολογίες, όπως η Scrum ή η Kanban, παρέχουν τη δυνατότητα να διαχωρίσουμε τη διαδικασία ανάπτυξης σε μικρότερες επαναλήψεις ή sprints. Αυτή η προσέγγιση επιτρέπει στην ομάδα ανάπτυξης να προχωρά σε συχνές κυκλοφορίες χαρακτηριστικών και διορθώσεις σφαλμάτων, μειώνοντας τον χρόνο μέχρι την ενσωμάτωση των βελτιώσεων στην αγορά. Η ευέλικτη ανάπτυξη προωθεί την προσαρμοστικότητα, επιτρέποντάς μας να ανταποκρινόμαστε

άμεσα στις μεταβαλλόμενες ανάγκες των χρηστών και στις τάσεις της αγοράς. Αυξάνει τη συνεργασία εντός της ομάδας, προάγει τη διαρκή βελτίωση και διασφαλίζει ότι η εφαρμογή παραμένει ευέλικτη και προσαρμόσιμη στις αναπτυσσόμενες απαιτήσεις.

8.3.3 Συμβατότητα πλατφόρμας

Για να διασφαλίσουμε τη μακροζωία της εφαρμογής μας για κινητά, είναι ζωτικής σημασίας να διατηρούμε τη συμβατότητά της με τις τελευταίες εκδόσεις των λειτουργικών συστημάτων iOS και Android. Πρέπει να παρακολουθούμε τις επερχόμενες ενημερώσεις των λειτουργικών συστημάτων και να διασφαλίζουμε ότι η εφαρμογή μας υποβάλλεται σε τακτικές δοκιμές για τον εντοπισμό και την αντιμετώπιση προβλημάτων συμβατότητας. Η αποτυχία σε αυτόν τον τομέα μπορεί να οδηγήσει σε απογοήτευση των χρηστών και αρνητικές κριτικές για την εφαρμογή μας. Επιπλέον, πρέπει να εκμεταλλευτούμε τα τελευταία χαρακτηριστικά και δυνατότητες που προσφέρουν οι νέες εκδόσεις του λειτουργικού συστήματός μας. Αυτό μας δίνει τη δυνατότητα να βελτιώσουμε τη λειτουργικότητα και την εμπειρία των χρηστών μας, προσφέροντάς τους έναν κίνητρο για να συνεχίσουν να χρησιμοποιούν την εφαρμογή. Με αυτόν τον τρόπο, διατηρούμε την εφαρμογή ανταγωνιστική και εξασφαλίζουμε ότι παραμένει ελκυστική για τους χρήστες, συνεχίζοντας να παρέχει υψηλής ποιότητας υπηρεσίες.

8.3.4 Ασφάλεια και απόρρητο δεδομένων

Καθώς οι απειλές ασφάλειας και οι ανησυχίες σχετικά με την προστασία του απορρήτου των δεδομένων συνεχίζουν να αυξάνονται, αναγκάζεται να δοθεί ύψιστη προτεραιότητα στην ασφάλεια της εφαρμογής μας για κινητά. Είναι απαραίτητο να αναθεωρούμε και να ενημερώνουμε τακτικά τα μέτρα ασφαλείας μας προκειμένου να προστατεύουμε τα δεδομένα των χρηστών και να προλαμβάνουμε τρύπες ασφαλείας. Πρέπει να είμαστε ενήμεροι για τις πιο πρόσφατες απειλές ασφάλειας και να ακολουθούμε τις βέλτιστες πρακτικές στους τομείς της κρυπτογράφησης, του ελέγχου ταυτότητας και της ασφαλούς αποθήκευσης δεδομένων. Επιπλέον, πρέπει να διασφαλίζουμε τη συμμόρφωσή μας με τους κανονισμούς προστασίας δεδομένων, όπως ο GDPR ή ο CCPA, προκειμένου να διατηρήσουμε την εμπιστοσύνη των χρηστών και να παραμείνουμε νομικά συμμορφοι. Οι τακτικοί έλεγχοι ασφαλείας και οι δοκιμές διείσδυσης μπορούν να βοηθήσουν στον προληπτικό εντοπισμό και την αντιμετώπιση τυχόν τρυπών ασφαλείας.

8.3.5 Σχεδιασμός με επίκεντρο τον χρήστη

Οι αρχές του σχεδιασμού με επίκεντρο τον χρήστη θα πρέπει να καθοδηγούν τις ενημερώσεις της εφαρμογής μας. Πρέπει να παραμένουμε διαρκώς ανοικτοί σε ευκαιρίες βελτίωσης της διεπαφής χρήστη (UI) και της εμπειρίας χρήστη (UX) βασιζόμενοι στα σχόλια των χρηστών και τις τάσεις του κλάδου. Είναι σημαντικό να πραγματοποιούμε τακτικές δοκιμές ευχρηστίας προκειμένου να ανιχνεύουμε περιοχές που χρειάζονται βελτίωση και να διασφαλίζουμε ότι οι αλλαγές στον σχεδιασμό συμβαδίζουν με τις προτιμήσεις των χρηστών. Μια καλά σχεδιασμένη εφαρμογή που προσφέρει μια φιλική προς τον χρήστη εμπειρία είναι πιο πιθανό να διατηρήσει τους υπάρχοντες χρήστες και να προσελκύσει νέους. Η θετική εμπειρία χρήστη αποτελεί ουσιώδη παράγοντα ικανοποίησης των χρηστών και επιτυχίας της εφαρμογής

8.3.6 Δοκιμές A/B

Για να αξιολογήσουμε τον αντίκτυπο των αλλαγών που εφαρμόζουμε στις ενημερώσεις της εφαρμογής, υιοθετούμε τις δοκιμές A/B ως μια αποτελεσματική στρατηγική. Οι δοκιμές A/B περιλαμβάνουν τη σύγκριση διαφορετικών εκδόσεων της εφαρμογής προκειμένου να μετρήσουμε πώς οι αλλαγές στα χαρακτηριστικά, το σχεδιασμό ή το περιεχόμενο επηρεάζουν τη συμπεριφορά και τη δέσμευση των χρηστών. Μέσω αυτής της διαδικασίας, μπορούμε να αναλύσουμε ποιες αλλαγές επιφέρουν θετικά αποτελέσματα και ποιες όχι, με βάση τα δεδομένα που συλλέγουμε. Αυτή η προσέγγιση βασισμένη στα δεδομένα μας βοηθά να λαμβάνουμε τεκμηριωμένες αποφάσεις σχετικά με τις αλλαγές που πρέπει να εφαρμόσουμε, λαμβάνοντας υπόψη τον αντίκτυπό τους στις αλληλεπιδράσεις των χρηστών και τις μετατροπές. Οι τακτικές δοκιμές A/B μας επιτρέπουν να συνεχώς βελτιώνουμε τις επιδόσεις της εφαρμογής μας και να αυξάνουμε την ικανοποίηση των χρηστών μας, προσφέροντας ένα πιο αποτελεσματικό και προσαρμοσμένο περιβάλλον χρήστη.

8.3.7 Τακτικές ενημερώσεις περιεχομένου

Για εφαρμογές που περιλαμβάνουν δυναμικό περιεχόμενο ή τακτικά ενημερωμένες πληροφορίες, η διατήρηση ενημερωμένου περιεχομένου αποτελεί απαραίτητη πρακτική. Η χρήση παλαιού ή ξεπερασμένου περιεχομένου μπορεί να οδηγήσει σε απώλεια χρηστών και αναγκάζει τους χρήστες να απομακρυνθούν από την εφαρμογή. Για να αντιμετωπίσουμε αυτό το πρόβλημα, μπορούμε να εφαρμόσουμε ένα σύστημα διαχείρισης περιεχομένου (CMS) ή να χρησιμοποιήσουμε αυτοματοποιημένους μηχανισμούς ενημέρωσης περιεχομένου. Το CMS είναι ένα εργαλείο που διευκολύνει την προσθήκη, την επεξεργασία και τη διαχείριση του περιεχομένου στην εφαρμογή σας, χωρίς την ανάγκη για προηγούμενες τεχνικές γνώσεις. Αυτό επιτρέπει στους διαχειριστές να ενημερώνουν το περιεχόμενο της εφαρμογής με ευκολία και ταχύτητα, προσφέροντας πάντα φρέσκες και σχετικές πληροφορίες στους χρήστες. Επίσης, είναι σημαντικό να διασφαλίσουμε ότι το περιεχόμενο ευθυγραμμίζεται με τις τρέχουσες τάσεις, γεγονότα ή ανάγκες των χρηστών. Αυτό διασφαλίζει τη διατήρηση του ενδιαφέροντος και της συμμετοχής των χρηστών, παρέχοντας τους συνεχώς πολύτιμες και ενημερωμένες πληροφορίες. Συνολικά, η διατήρηση του περιεχομένου ενημερωμένο και σχετικό είναι καθοριστική για την επιτυχία και την προσέλκυση των χρηστών στην εφαρμογή.

8.3.8 Βελτιστοποίηση επιδόσεων

Για τη διατήρηση μιας ευέλικτης και αξιόπιστης εφαρμογής, η συνεχής βελτιστοποίηση των επιδόσεων είναι ζωτική. Πρέπει να διεξάγονται τακτικές δοκιμές επιδόσεων για τον εντοπισμό και την αντιμετώπιση διεργασιών που επιβαρύνουν τους πόρους, διαρροές μνήμης ή άλλα σημεία που μπορεί να προκαλέσουν προβλήματα επιδόσεων. Επιπλέον, είναι σημαντικό να βελτιστοποιηθούν οι εικόνες, να ελαχιστοποιηθούν οι αιτήσεις προς το δίκτυο και να εφαρμοστούν μηχανισμοί προσωρινής αποθήκευσης προκειμένου να μειωθούν οι χρόνοι φόρτωσης και η κατανάλωση δεδομένων. Η διατήρηση των επιδόσεων της εφαρμογής υπό έλεγχο είναι απαραίτητη για την αποφυγή προβλημάτων και αναδεικνύεται σε κρίσιμο παράγοντα για τη βελτίωση της ικανοποίησης των χρηστών. Επιπλέον, αυτή η διαδικασία συμβάλλει στη διατήρηση του ανταγωνιστικού πλεονεκτήματος στην αγορά εφαρμογών, καθώς η απρόσκοπτη και αποτελεσματική λειτουργία της εφαρμογής εξασφαλίζει την προτίμηση των χρηστών και την διατήρηση της εφαρμογής ως μία αξιόπιστη επιλογή για τις ανάγκες τους.

8.3.9 Παρακολούθηση και επίλυση σφαλμάτων

Για να διασφαλιστεί η σταθερότητα και η αξιοπιστία της εφαρμογής, είναι σημαντικό να θεσπιστεί μια ισχυρή διαδικασία παρακολούθησης και επίλυσης σφαλμάτων. Καλό είναι να ενθαρρύνουμε τους χρήστες να αναφέρουν σφάλματα και προβλήματα μέσω μηχανισμών αναφοράς εντός της εφαρμογής ή μέσω καναλιών υποστήριξης. Επίσης, είναι σημαντικό να χρησιμοποιηθεί λογισμικό παρακολούθησης σφαλμάτων για την κατηγοριοποίηση, ιεράρχηση και ανάθεση των αναφερόμενων ζητημάτων στις ομάδες ανάπτυξης. Διατηρείται διαφάνεια με τους χρήστες αναγνωρίζοντας τα αναφερόμενα σφάλματα και παρέχοντας πληροφορίες σχετικά με το πότε μπορούν να περιμένουν επίλυση. Είναι σημαντικό να αντιμετωπίζονται και να επιλύονται τα σφάλματα άμεσα, προκειμένου να αποφεύγεται την απογοήτευση των χρηστών και τις αρνητικές κριτικές. Μια εφαρμογή χωρίς σφάλματα αποτελεί απαραίτητη προϋπόθεση για τη διατήρηση και την ικανοποίηση των χρηστών, και είναι κρίσιμη για τη διατήρηση ανταγωνιστικού πλεονεκτήματος στην αγορά εφαρμογών.

8.3.10 Μάρκετινγκ και δέσμευση χρηστών

Για την προώθηση των ενημερώσεων της εφαρμογής και τη διατήρηση της ενεργής συμμετοχής των χρηστών, απαιτούνται αποτελεσματικές στρατηγικές μάρκετινγκ και εμπλοκής. Μια προτεινόμενη προσέγγιση είναι η χρήση ειδοποιήσεων push, ηλεκτρονικού ταχυδρομείου και κοινωνικών μέσων για να ενημερώνουμε τους χρήστες σχετικά με νέα χαρακτηριστικά, βελτιώσεις και διορθώσεις σφαλμάτων που περιέχονται στις ενημερώσεις. Επίσης, είναι σημαντικό να αναδείξουμε την αξία αυτών των ενημερώσεων για να ενθαρρύνουμε τους χρήστες να τις εγκαταστήσουν. Επιπλέον, η εφαρμογή τακτικών δέσμευσης των χρηστών εντός της εφαρμογής είναι ουσιώδης. Αυτό μπορεί να περιλαμβάνει την παροχή μηνυμάτων εντός της εφαρμογής, εξατομικευμένων συστάσεων περιεχομένου και προγραμμάτων πιστότητας. Αυτές οι τακτικές συμβάλλουν στη διατήρηση της ενεργής χρήσης της εφαρμογής από τους χρήστες και στην προώθηση της μακροπρόθεσμης συμμετοχής τους. Συνοψίζοντας, η συνδυασμένη χρήση ειδοποιήσεων και τακτικών δέσμευσης των χρηστών είναι κρίσιμη για την επιτυχή προώθηση και διατήρηση της ενεργής συμμετοχής σε μια εφαρμογή, και πρέπει να σχεδιαστούν κατάλληλα για να προσελκύσουν και να διατηρήσουν το ενδιαφέρον των χρηστών.

8.3.11 Εκπαίδευση και υποστήριξη χρηστών

Για να βοηθηθούν οι χρήστες να προσαρμοστούν στα νέα χαρακτηριστικά ή στις αλλαγές που εισάγονται στις ενημερώσεις, είναι σημαντικό να παρέχουμε πόρους που είναι σαφείς και προσβάσιμοι εντός της εφαρμογής. Μπορούμε να προσφέρουμε μαθήματα εντός της εφαρμογής, συμβουλές εργαλείων ή οδηγούς που καθοδηγούν τους χρήστες στη χρήση της ενημερωμένης διεπαφής ή την εξοικείωσή τους με τις νέες λειτουργίες. Επιπλέον, είναι σημαντικό να διατηρηθούν ευέλικτα κανάλια υποστήριξης πελατών, όπως το ηλεκτρονικό ταχυδρομείο, η συνομιλία ή μια βάση γνώσεων, ώστε να ανταποκρινόμαστε άμεσα στα ερωτήματα και τα προβλήματα των χρηστών. Η αποτελεσματική εκπαίδευση και υποστήριξη των χρηστών είναι ουσιώδης για την ελαχιστοποίηση της απογοήτευσής τους και τη βελτίωση της συνολικής εμπειρίας τους με την εφαρμογή. Με αυτόν τον τρόπο, παρέχουμε στους χρήστες τα εργαλεία και την υποστήριξη που χρειάζονται για να αντιμετωπίσουν τις αλλαγές με αυτοπεποίθηση και να εκμεταλλευτούν πλήρως τις νέες δυνατότητες της εφαρμογής. Αυτό ενισχύει την ευχάριστη εμπειρία τους και συμβάλλει στην αποτελεσματική χρήση της εφαρμογής.

8.3.12 Σχέδιο ελέγχου εκδόσεων και επαναφοράς

Η διατήρηση αυστηρού ελέγχου των εκδόσεων της βάσης κώδικα της εφαρμογής είναι κρίσιμη για την εξασφάλιση της απρόσκοπτης μετάβασης μεταξύ των ενημερώσεων. Μέσω του ελέγχου εκδόσεων, έχουμε τη δυνατότητα να παρακολουθούμε τις αλλαγές που γίνονται στον κώδικα, να εντοπίζουμε προβλήματα και, εάν απαιτείται, να επιστρέφουμε σε μια προηγούμενη σταθερή έκδοση. Αυτό είναι κρίσιμο για τη διατήρηση της σταθερότητας και της αξιοπιστίας της εφαρμογής. Επίσης, η ύπαρξη ενός καλά καθορισμένου σχεδίου επαναφοράς είναι απαραίτητη σε περίπτωση που προκύψει ένα κρίσιμο πρόβλημα μετά από μια ενημέρωση. Αυτό το σχέδιο μάς επιτρέπει να επαναφέρουμε γρήγορα την εφαρμογή σε μια προηγούμενη, γνωστή και σταθερή κατάσταση, ελαχιστοποιώντας την αναστάτωση των χρηστών και διατηρώντας την αξιοπιστία της εφαρμογής. Αυτές οι πρακτικές συμβάλλουν σημαντικά στη διατήρηση της απρόσκοπτης λειτουργίας της εφαρμογής και στη διαχείριση απρόοπτων προβλημάτων με αποτελεσματικό τρόπο.

Η ενσωμάτωση αυτών των λεπτομερών στρατηγικών στο σχέδιο συντήρησης και ενημέρωσης της εφαρμογής μας για κινητά τηλέφωνα θα μας επιτρέψει να περιηγηθούμε αποτελεσματικά στις πολυπλοκότητες της διαχείρισης εφαρμογών. Κάθε στρατηγική αντιμετωπίζει κρίσιμες πτυχές της συντήρησης και βελτίωσης της εφαρμογής για κινητά, ώστε να διασφαλιστεί ότι η εφαρμογή παραμένει ανταγωνιστική, φιλική προς το χρήστη, ασφαλής και αξιόπιστη στο συνεχώς εξελισσόμενο τοπίο των εφαρμογών για κινητά.

9 Συμπεράσματα

Η ανάπτυξη και η μετάβαση της "Ίριδας" σε κινητές συσκευές αποτελούν σημαντική πρόοδο στον τομέα της ανταλλαγής και διαχείρισης εγγράφων. Αυτή η εφαρμογή για κινητά σχεδιάστηκε με το όραμα να παρέχει στους χρήστες μια ισχυρή, αισθητική και διαπλατφορμική λύση για τις ανάγκες τους που σχετίζονται με έγγραφα. Κατά τη διάρκεια του παρόντος εγγράφου, διερευνήσαμε τις διάφορες πτυχές και φάσεις του έργου, ρίχνοντας φως στις βασικές αποφάσεις, προκλήσεις και στρατηγικές που διαμόρφωσαν την ανάπτυξή του.

Το ταξίδι της δημιουργίας της "Ίριδας" ξεκίνησε με τη σαφή κατανόηση της αυξανόμενης ζήτησης για αποτελεσματικές λύσεις διαχείρισης και διακίνησης εγγράφων. Με τον πολλαπλασιασμό των κινητών συσκευών, η ανάγκη για μια ευέλικτη και φιλική προς τον χρήστη εφαρμογή έγινε εμφανής. Η "Ίριδα" σχεδιάστηκε για να ικανοποιήσει αυτή τη ζήτηση, προσφέροντας χαρακτηριστικά που απλοποιούν τις ροές εργασίας εγγράφων, ενισχύουν την ασφάλεια και διασφαλίζουν την προσβασιμότητα στις πλατφόρμες iOS και Android.

Η επιλογή του Ionic Framework ως πλατφόρμα ανάπτυξης αποδείχθηκε καθοριστική για την επίτευξη των στόχων μας όσον αφορά τη συμβατότητα μεταξύ των πλατφορμών. Το εκτεταμένο σύνολο εργαλείων και στοιχείων του Ionic έδωσε τη δυνατότητα στην ομάδα ανάπτυξής μας να δημιουργήσει μια απρόσκοπτη εμπειρία χρήστη σε μια ποικιλία συσκευών, χωρίς να χρειάζεται να δημιουργούμε ξεχωριστές εφαρμογές για διαφορετικά λειτουργικά συστήματα.

Τα πλεονεκτήματα από τη χρήση της "Ίριδας" είναι πολλά. Εξορθολογίζει τις διαδικασίες ανταλλαγής εγγράφων, μειώνοντας την πιθανότητα λαθών και ελαχιστοποιώντας τον χρόνο και την προσπάθεια που απαιτούνται για τη διαχείριση εγγράφων. Αυτή η εφαρμογή απευθύνεται σε ένα ευρύ φάσμα χρηστών, από ιδιώτες που αναζητούν έναν εύκολο τρόπο αποστολής προσωπικών εγγράφων έως οργανισμούς που επιθυμούν να βελτιστοποιήσουν τις ροές εργασίας εγγράφων τους. Η ευελιξία και η ευκολία χρήσης της την καθιστούν πολύτιμη προσθήκη στην κινητή εργαλειοθήκη οποιουδήποτε.

Έπειτα, συγκρίναμε το Ionic Framework με το React Native, τονίζοντας γιατί τελικά επιλέξαμε το Ionic για το "Ίριδα". Οι διαφορές στην προσέγγιση ανάπτυξης, την υποστήριξη του οικοσυστήματος και τη συμβατότητα με τις πλατφόρμες επιβεβαίωσαν την απόφασή μας και υπογράμμισαν τα πλεονεκτήματα του Ionic στο δικό μας πλαίσιο.

Καθώς το Σύστημα Ηλεκτρονικής Διαχείρισης Εγγράφων "Ίριδα" προχωράει, είναι σημαντικό να αναγνωρίζουμε τη σημασία κάθε φάσης ανάπτυξης. Η φάση Pre-Migration δίνει έμφαση στον σχολαστικό σχεδιασμό, τον σχεδιασμό της εμπειρίας χρήστη και την κατανόηση των προσδοκιών των χρηστών. Η λεπτομερής ανάλυση των δομών των φακέλων και των αρχείων μας αποτελεί παράδειγμα του τρόπου με τον οποίο η αρχιτεκτονική της εφαρμογής υποστηρίζει την αποτελεσματική ανάπτυξη και συντήρηση.

Η φάση Migration αναδεικνύει τις στρατηγικές και τις εκτιμήσεις που αφορούν τη μετάβαση από μια εφαρμογή React σε μια Ionic. Αναφέραμε τόσο την πλήρη μετάβαση όσο και τις προσεγγίσεις σταδιακής μετάβασης, επιτρέποντας την ευελιξία στην προσαρμογή των υφιστάμενων βάσεων κώδικα. Τα πρακτικά βήματα και τα παραδείγματα μετατροπής κώδικα χρησιμεύουν ως πολύτιμοι πόροι για τους προγραμματιστές που ξεκινούν παρόμοια ταξίδια.

Στη φάση Post - Migration, δώσαμε έμφαση στον κρίσιμο ρόλο των δοκιμών και της διασφάλισης ποιότητας για την παροχή αξιόπιστης εμπειρίας χρήστη. Τονίσαμε επίσης τη

σημασία της διατήρησης συνεπών στυλ κωδικοποίησης και μοιραστήκαμε πρακτικές διαδικασίες δοκιμών. Αυτές οι πτυχές διασφαλίζουν τη μακροπρόθεσμη βιωσιμότητα και σταθερότητα του "Ίριδα".

Κοιτάζοντας μπροστά, το παρόν έγγραφο εμβάθυνε επίσης σε πραγματικές μελέτες περιπτώσεων και βέλτιστες πρακτικές για τη μετατροπή εφαρμογών React σε Ionic. Αυτές οι γνώσεις προσφέρουν καθοδήγηση για προγραμματιστές και οργανισμούς που επιδιώκουν να επαναλάβουν το έργο μας. Επιπλέον, διερευνήθηκαν στρατηγικές για τη διατήρηση και την ενημέρωση της "Ίριδας" ώστε να παραμείνει ευθυγραμμισμένη με τις εξελισσόμενες ανάγκες των χρηστών και τις τεχνολογικές εξελίξεις.

Έτσι, το "Ίριδα" αποτελεί ένα μεγάλο βήμα προς τα εμπρός στην κοινή χρήση και διαχείριση εγγράφων. Το παρόν έγγραφο παρείχε μια ολοκληρωμένη επισκόπηση της πορείας ανάπτυξής του, από τη σύλληψη έως την υλοποίηση, τονίζοντας τη σημασία κάθε φάσης και απόφασης. Προβλέπουμε ότι το "Ίριδα" θα συνεχίσει να εξελίσσεται και να χρησιμεύει ως πολύτιμο εργαλείο για άτομα και οργανισμούς που αναζητούν αποτελεσματικές λύσεις εγγράφων στο συνεχώς μεταβαλλόμενο τοπίο των κινητών συσκευών.

10 Λεξιλόγιο όρων

10.1 React

Το React (συχνά αναφέρεται ως React.js ή ReactJS) είναι μια ανοικτού κώδικα βιβλιοθήκη JavaScript που χρησιμοποιείται για τη δημιουργία διεπαφών χρήστη (UI) για web εφαρμογές. Δημιουργήθηκε αρχικά από το Facebook και πλέον διαχειρίζεται από το Facebook και μια κοινότητα ατόμων και εταιρειών. Το React κυκλοφόρησε για πρώτη φορά το 2013 και έκτοτε έχει αποκτήσει ευρεία δημοτικότητα στην κοινότητα ανάπτυξης ιστού.

Το React σχεδιάστηκε ειδικά για να βοηθά τους προγραμματιστές στη δημιουργία διαδραστικών και δυναμικών διεπαφών χρήστη με ευκολία. Ακολουθεί μια αρχιτεκτονική βασισμένη σε components, όπου η UI διασπάται σε επαναχρησιμοποιήσιμα στοιχεία. Κάθε στοιχείο μπορεί να έχει τη δική του λογική και κατάσταση, καθιστώντας ευκολότερη τη διαχείριση και τη συντήρηση πολύπλοκων διεπαφών χρήστη.

10.1.1 Βασικά χαρακτηριστικά και έννοιες του React περιλαμβάνουν:

- *Virtual DOM*: Το React χρησιμοποιεί μια εικονική αναπαράσταση του πραγματικού DOM (Document Object Model) για τη βελτίωση της απόδοσης. Αντί να αλληλεπιδρά με το DOM απευθείας, το React ενημερώνει πρώτα το εικονικό DOM και στη συνέχεια υπολογίζει τον πιο αποτελεσματικό τρόπο ενημέρωσης του πραγματικού DOM. Αυτό ελαχιστοποιεί τον αριθμό των πραγματικών αλλαγών στο DOM, βελτιώνοντας την απόδοση και την εμπειρία του χρήστη.
- *Βασισμένο σε Components*: Το React ενθαρρύνει τους προγραμματιστές να διασπάσουν τη διεπαφή σε μικρά, επαναχρησιμοποιήσιμα στοιχεία. Τα στοιχεία μπορούν να συνδυαστούν για να δημιουργήσουν πολύπλοκες διεπαφές χρήστη, και οι αλλαγές που γίνονται σε ένα στοιχείο δεν επηρεάζουν απαραίτητα τα υπόλοιπα, βελτιώνοντας τη δυνατότητα διαχείρισης και συντήρησης του κώδικα.
- *JSX (JavaScript XML)*: Το React επιτρέπει στους προγραμματιστές να γράφουν στοιχεία διεπαφής χρήστη χρησιμοποιώντας το JSX, που είναι μια συντακτική επέκταση για τη γλώσσα JavaScript. Το JSX καθιστά ευκολότερο τον καθορισμό της δομής και της εμφάνισης των στοιχείων με declarative και παρόμοιο με HTML τρόπο.
- *Μονόδρομη Ροή Δεδομένων*: Το React ακολουθεί μια μονόδρομη ροή δεδομένων, πράγμα που σημαίνει ότι τα δεδομένα ρέουν μονόδρομα από τα γονικά προς τα παιδιά στοιχεία. Αυτό βοηθά στη διατήρηση προβλέψιμων και διαχειρίσιμων ενημερώσεων δεδομένων στην εφαρμογή.
- *React Router*: Το React Router είναι μια δημοφιλής βιβλιοθήκη δρομολόγησης που ενσωματώνεται ομαλά στο React για τη δυνατότητα δρομολόγησης και πλοήγησης στον πελάτη σε εφαρμογές μονής σελίδας (SPAs).
- *Διαχείριση Κατάστασης*: Το React παρέχει μηχανισμό ενσωματωμένης διαχείρισης της κατάστασης του στοιχείου χρησιμοποιώντας το hook useState (σε λειτουργικά στοιχεία) ή τη μέθοδο setState (σε στοιχεία κλάσης). Για πιο πολύπλοκες ανάγκες διαχείρισης κατάστασης, οι προγραμματιστές συνήθως ενσωματώνουν βιβλιοθήκες τρίτων, όπως το Redux, το Recoil ή το Mobx.

Η δημοφιλία του React οφείλεται στην απλότητά του, στις βελτιστοποιήσεις της απόδοσης και στον ζωντανό οικοσύστημα βιβλιοθηκών και εργαλείων που έχουν αναπτυχθεί γύρω του. Χρησιμοποιείται ευρέως στην ανάπτυξη ιστού και αποτελεί μια πολύτιμη δεξιότητα για τους προγραμματιστές προσβάσιμης προς τον χρήστη πλευράς (front-end developers).

10.2 Component

Στο React, ένα component είναι ένα επαναχρησιμοποιήσιμο και αυτοτελές δομικό στοιχείο μιας διεπαφής χρήστη (UI). Τα components είναι τα θεμελιώδη δομικά στοιχεία των εφαρμογών React και ενθυλακώνουν τη λογική, τη δομή και την παρουσίαση ενός τμήματος της διεπαφής χρήστη.

Τα components της React μπορούν να κατηγοριοποιηθούν σε δύο κύριους τύπους:

- **Function Components:** Αυτά είναι επίσης γνωστά ως stateless ή functional components. Ορίζονται ως συναρτήσεις JavaScript και δέχονται props (συντομογραφία για τις ιδιότητες) ως είσοδο και επιστρέφουν JSX (JavaScript XML) για να περιγράψουν τι πρέπει να αποδοθεί στην οθόνη. Τα Function Components χρησιμοποιούνται κυρίως για την απόδοση στοιχείων UI με βάση τα δεδομένα εισόδου.

  Παράδειγμα:

```
import React from 'react';

function MyComponent(props) {
  return (
    <div>
      <h1>Hello, {props.name}!</h1>
    </div>
  );
}

export default MyComponent;
```

- **Class Components:** Αυτά είναι επίσης γνωστά ως components με state (stateful components). Τα Class Components ορίζονται ως κλάσεις JavaScript και μπορούν να έχουν τη δική τους εσωτερική κατάσταση (state) εκτός από το να λαμβάνουν και να χρησιμοποιούν props. Χρησιμοποιούνται όταν ένα component πρέπει να διαχειρίζεται την κατάστασή του, να χειρίζεται συμβάντα κύκλου ζωής και να εκτελεί πιο σύνθετη λογική.

  Παράδειγμα

```
import React, { Component } from 'react';

class MyComponent extends Component {
  constructor(props) {
    super(props);
    this.state = {
      count: 0,
    };
  }

  handleClick = () => {
```

```
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <h1>Count: {this.state.count}</h1>
        <button onClick={this.handleClick}>Increment</button>
      </div>
    );
  }
}

export default MyComponent;
```

10.3 Hook

Στο React, ένα "hook" είναι μια ειδική λειτουργία που σας επιτρέπει να "γαντζώνεστε" ή να έχετε πρόσβαση σε ορισμένα χαρακτηριστικά της React σε functional components. Τα hooks εισήχθησαν στο React 16.8 ως ένας τρόπος να προστεθούν χαρακτηριστικά διαχείρισης κατάστασης και κύκλου ζωής σε functional components, τα οποία προηγουμένως περιορίζονταν σε στοιχεία κλάσεων.

Τα hooks παρέχουν έναν πιο ευέλικτο και συνθέσιμο τρόπο διαχείρισης της κατάστασης, των παρενεργειών και του κύκλου ζωής των συστατικών σε λειτουργικά συστατικά, καθιστώντας ευκολότερη την επαναχρησιμοποίηση της λογικής και τη διαχείριση της σύνθετης συμπεριφοράς των συστατικών. Επιτρέπουν τη χρησιμοποίηση της κατάστασης και άλλα χαρακτηριστικά της React χωρίς να χρειάζεται να γραφτεί ένα class component.

Μερικά από τα συχνά χρησιμοποιούμενα hook της React περιλαμβάνουν:

- **useState:** Αυτό το hook επιτρέπει να προσθέσουμε κατάσταση σε ένα functional components. Επιστρέφει έναν πίνακα με την τρέχουσα τιμή της κατάστασης και μια συνάρτηση για την ενημέρωση αυτής της κατάστασης.

  Παράδειγμα:

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={() => setCount(count + 1)}>Increment</button>
    </div>
  );
}
```

- **useEffect:** Αυτό το hook επιτρέπει την εκτέλεση δευτερευόντων εφέ στο components μας, όπως η άντληση δεδομένων, ή η χειροκίνητη αλληλεπίδραση με το DOM. Αντικαθιστά τις μεθόδους κύκλου ζωής όπως οι componentDidMount, componentDidUpdate και componentWillUnmount σε class components.

  Παράδειγμα.

```
import React, { useEffect } from 'react';
```

```
function Example() {
  useEffect(() => {
    // This code runs after the component renders
    document.title = 'Updated Title';

    // Cleanup function (equivalent to componentWillUnmount)
    return () => {
      document.title = 'Original Title';
    };
  }, []);

  return <div>Example Component</div>;
}
```

- **Custom hooks:** Μπορούμε επίσης να δημιουργήσουμε τα δικά μας προσαρμοσμένα hooks για να ενθυλακώσουμε και να επαναχρησιμοποιήσουμε τη λογική σε διαφορετικά στοιχεία. Τα προσαρμοσμένα hooks είναι απλώς συναρτήσεις JavaScript που χρησιμοποιούν άλλα hook.

  Παράδειγμα

```
import { useState } from 'react';

function useCounter(initialValue) {
  const [count, setCount] = useState(initialValue);

  const increment = () => {
    setCount(count + 1);
  };

  return [count, increment];
}
```

Τα hooks έχουν γίνει αναπόσπαστο μέρος της σύγχρονης ανάπτυξης της React και χρησιμοποιούνται ευρέως για την απλοποίηση της λογικής των συστατικών και τη βελτίωση της αναγνωσιμότητας του κώδικα σε λειτουργικά συστατικά. Παρέχουν έναν πιο λειτουργικό και προβλέψιμο τρόπο εργασίας με την κατάσταση και τα πλευρικά αποτελέσματα, καθιστώντας ευκολότερη τη δημιουργία και τη συντήρηση σύνθετων εφαρμογών.

10.4 State

Στο React, η "κατάσταση" αναφέρεται σε ένα αντικείμενο που περιέχει δεδομένα και αντιπροσωπεύει την εσωτερική κατάσταση ενός στοιχείου. Είναι μία από τις βασικές έννοιες της React και χρησιμοποιείται για την αποθήκευση και διαχείριση δυναμικών δεδομένων που μπορούν να αλλάξουν με την πάροδο του χρόνου ως αποτέλεσμα των αλληλεπιδράσεων του χρήστη, των αποκρίσεων του διακομιστή ή άλλων παραγόντων. Η κατάσταση επιτρέπει στα components της React να επανεκδίδουν και να ενημερώνουν το UI όταν αλλάζουν τα δεδομένα που κατέχουν, διασφαλίζοντας ότι η διεπαφή χρήστη αντικατοπτρίζει την τρέχουσα κατάσταση της εφαρμογής.

Ακολουθούν ορισμένα βασικά σημεία σχετικά με την κατάσταση της React.

- **Η κατάσταση είναι μεταβλητή:** Η κατάσταση είναι μεταβλητή, πράγμα που σημαίνει ότι μπορούμε να την τροποποιήσουμε χρησιμοποιώντας τη μέθοδο setState που παρέχει το React. Ωστόσο, δεν θα πρέπει ποτέ να τροποποιούμε απευθείας το αντικείμενο κατάστασης χρησιμοποιώντας ανάθεση (this.state = ...). Αντ' αυτού,

χρησιμοποιούμε πάντα τη setState για να ενημερώσουμε την κατάστασή μας, καθώς διασφαλίζει ότι το React επανεκτυπώνει σωστά το στοιχείο και ενεργοποιεί τις απαραίτητες ενημερώσεις.

- **Αρχική κατάσταση:** Συνήθως ορίζουμε την αρχική κατάσταση ενός στοιχείου στον κατασκευαστή του στοιχείου ή χρησιμοποιούμε το άγκιστρο useState (για functional components). Για class components, η αρχική κατάσταση ορίζεται στον κατασκευαστή - constructor. Για functional components, μπορούμε να χρησιμοποιήσουμε το hook useState για τη διαχείριση της κατάστασης.
- **Ενημέρωση της κατάστασης:** Για να ενημερώσουμε την κατάσταση, καλούμε τη μέθοδο setState με ένα νέο αντικείμενο κατάστασης ή μια συνάρτηση που παράγει μια νέα κατάσταση με βάση την προηγούμενη κατάσταση. Στη συνέχεια, το React συγχωνεύει τη νέα κατάσταση με την υπάρχουσα κατάσταση και προγραμματίζει την εκ νέου εμφάνιση του στοιχείου.
- **Διάδοση:** Αλλαγές στην κατάσταση προκαλούν επαναπροβολή του στοιχείου και των θυγατρικών του στοιχείων, διασφαλίζοντας ότι το περιβάλλον εργασίας αντικατοπτρίζει την ενημερωμένη κατάσταση. Το React ενημερώνει αποτελεσματικά μόνο τα τμήματα του DOM που έχουν αλλάξει, βελτιστοποιώντας την απόδοση.
- **Τοπική κατάσταση component:** Η κατάστασή μας συνήθως περιορίζεται σε ένα μόνο στοιχείο και δεν διαμοιράζεται άμεσα μεταξύ διαφορετικών. Για να μοιραστούμε την κατάστασή μας μεταξύ συστατικών, μπορούμε να χρησιμοποιήσουμε props και να ανυψώσουμε την κατάστασή μας σε ένα κοινό προγονικό συστατικό ή να χρησιμοποιήσουμε βιβλιοθήκες διαχείρισης κατάστασης όπως το Redux ή το React Context.

Παράδειγμα 1

```
class Counter extends React.Component {
  constructor(props) {
    super(props);
    this.state = { count: 0 };
  }

  increment = () => {
    this.setState({ count: this.state.count + 1 });
  }

  render() {
    return (
      <div>
        <p>Count: {this.state.count}</p>
        <button onClick={this.increment}>Increment</button>
      </div>
    );
  }
}
```

Παράδειγμα 2

```
import React, { useState } from 'react';

function Counter() {
  const [count, setCount] = useState(0);

  const increment = () => {
```

```
    setCount(count + 1);
  };

  return (
    <div>
      <p>Count: {count}</p>
      <button onClick={increment}>Increment</button>
    </div>
  );
}
```

10.5 props

Στο React, το "props" είναι η συντομογραφία του "properties" και αναφέρεται σε έναν μηχανισμό για τη μετάδοση δεδομένων από ένα στοιχείο σε ένα άλλο. Τα props είναι μια θεμελιώδης έννοια στο React και μας επιτρέπουν να περνάμε δεδομένα από ένα γονικό συστατικό (αυτό που απεικονίζει το Child Component) σε ένα Child Component (αυτό που λαμβάνει και χρησιμοποιεί τα δεδομένα). Τα props είναι μόνο για ανάγνωση και δεν μπορούν να τροποποιηθούν από το στοιχείο-παιδί που τα λαμβάνει.

Ακολουθεί ο τρόπος με τον οποίο λειτουργούν τα props στο React.

- **Parent Component:** Σε μια εφαρμογή React, υπάρχει συνήθως μια ιεραρχία components. Το γονικό στοιχείο είναι υπεύθυνο για την απόδοση του Child Component και τη διαβίβαση δεδομένων σε αυτό. Τα δεδομένα που πρέπει να διαβιβαστούν καθορίζονται ως χαρακτηριστικά στο στοιχείο JSX του στοιχείου παιδί.

  Παράδειγμα

```
import React from 'react';

function ParentComponent() {
  const data = "Hello from Parent!";

  return (
    <ChildComponent message={data} />
  );
}
```

- **Child Component:** Το Child Component λαμβάνει τα δεδομένα που μεταβιβάζονται ως props και μπορεί να έχει πρόσβαση σε αυτά ως παράμετρο στη συνάρτησή του ή ως ιδιότητα στην περίπτωση της κλάσης του. Τα δεδομένα αυτά χρησιμοποιούνται στη συνέχεια για την απόδοση περιεχομένου ή την επιρροή της συμπεριφοράς του Child Component.

  Παράδειγμα function component

```
import React from 'react';

function ChildComponent(props) {
  return (
    <div>
      <p>{props.message}</p>
    </div>
  );
```

```
}
```

Παράδειγμα class component

```
import React, { Component } from 'react';

class ChildComponent extends Component {
  render() {
    return (
      <div>
        <p>{this.props.message}</p>
      </div>
    );
  }
}
```

- **Rendering:** Το React ενημερώνει αυτόματα το στοιχείο-παιδί κάθε φορά που αλλάζουν τα props. Αυτό επιτρέπει τη δυναμική απόδοση με βάση τις αλλαγές στα δεδομένα που μεταβιβάζονται από το γονέα.

Τα props αποτελούν βασικό μέρος της αρχιτεκτονικής της React που βασίζεται σε components, επιτρέποντας τη σύνθεση σύνθετων διεπαφών χρήστη από μικρότερα, επαναχρησιμοποιήσιμα δομικά στοιχεία. Διευκολύνουν την κοινή χρήση και τη διαχείριση δεδομένων μεταξύ συστατικών, επιτρέποντάς μας να δημιουργήσουμε ευέλικτο και συντηρήσιμο κώδικα. Επιπλέον, τα στηρίγματα βοηθούν στη δημιουργία μιας σαφούς και προβλέψιμης ροής δεδομένων εντός της εφαρμογής.

10.6 Codebase - Βάση κώδικα

Η βάση κώδικα ενός έργου React αναφέρεται σε όλο τον πηγαίο κώδικα και τα σχετικά αρχεία που συνθέτουν το έργο. Ενα έργο React αποτελείται συνήθως από διάφορα αρχεία και φακέλους που περιέχουν κώδικα, στοιχεία, αρχεία ρυθμίσεων και άλλα.

Ακολουθούν ορισμένα κοινά στοιχεία της βάσης κώδικα ενός έργου React.

- **Source code:**Περιλαμβάνει αρχεία JavaScript (ή TypeScript) που καθορίζουν τα στοιχεία της React, τη λογική τους και τον τρόπο με τον οποίο αλληλεπιδρούν μεταξύ τους. Αυτά τα αρχεία συχνά έχουν επέκταση `.js` ή `.jsx` (ή `.ts` ή `.tsx` για TypeScript).
- **Components:** Τα έργα React οργανώνονται σε επαναχρησιμοποιήσιμα στοιχεία. Κάθε στοιχείο μπορεί να έχει τον δικό του φάκελο που περιέχει τον κώδικα του, τα στυλ του και ενδεχομένως τις δοκιμές του.
- **Stylesheets:** Φύλλα στυλ CSS ή προεπεξεργασμένα φύλλα στυλ (π.χ. SASS, LESS) που χρησιμοποιούνται για το στυλ των στοιχείων React. Αυτά συχνά αποθηκεύονται σε ξεχωριστά αρχεία `.css` ή `.scss`.
- **Πρότυπα HTML:** Τα συστατικά της React συνήθως παράγουν HTML, αλλά μερικές φορές μπορεί να υπάρχουν στατικά πρότυπα HTML που χρησιμοποιούνται για συγκεκριμένους σκοπούς.
- **Στοιχεία:** Εικόνες, γραμματοσειρές, εικονίδια και άλλα στατικά στοιχεία που χρησιμοποιούνται στο έργο αποθηκεύονται συχνά σε έναν κατάλογο "assets".
- **Αρχεία ρυθμίσεων:** Διάφορα αρχεία ρυθμίσεων όπως `package.json` (για εξαρτήσεις και σενάρια npm), `.babelrc` (για ρυθμίσεις της Babel), `.eslintrc` (για ρυθμίσεις του ESLint) και άλλα ανάλογα με τη ρύθμιση του έργου.

- **Δοκιμές:** Οι δοκιμές μονάδας και οι δοκιμές ολοκλήρωσης είναι απαραίτητες για τη διασφάλιση της ορθότητας της εφαρμογής React. Τα αρχεία δοκιμών συχνά βρίσκονται σε ξεχωριστό κατάλογο, όπως `__tests__` ή `tests`.
- **Αρχεία build:** Τα αρχεία που παράγονται από εργαλεία κατασκευής, όπως το Webpack, το Babel ή το Create React App, συχνά βρίσκονται σε έναν κατάλογο "build" ή "dist". Αυτά τα αρχεία βελτιστοποιούνται και ομαδοποιούνται για την ανάπτυξη της παραγωγής.
- **Διαμόρφωση δρομολόγησης:** Εάν το έργο χρησιμοποιεί δρομολόγηση από την πλευρά του πελάτη, ενδέχεται να υπάρχουν αρχεία ρυθμίσεων που σχετίζονται με τη δρομολόγηση (π.χ. ρύθμιση `react-router`).
- **Διαχείριση κατάστασης:** Εάν το έργο χρησιμοποιεί μια βιβλιοθήκη διαχείρισης καταστάσεων όπως η Redux ή η Mobx, υπάρχεις κώδικας και ρυθμίσεις που σχετίζονται με τη διαχείριση καταστάσεων στην κωδικοβάση.
- **Κώδικας από την πλευρά του διακομιστή / server:** Σε ορισμένες περιπτώσεις, ένα έργο React μπορεί να περιλαμβάνει κώδικα από την πλευρά του διακομιστή, όπως σημεία API.

Η συγκεκριμένη δομή και οργάνωση της βάσης κώδικα ενός έργου React μπορεί να ποικίλλει ανάλογα με την πολυπλοκότητα του έργου και τις προτιμήσεις της ομάδας ανάπτυξης. Πολλά έργα React ξεκινούν με τη χρήση εργαλείων όπως το Create React App, το οποίο παρέχει μια τυπική δομή έργου για να ξεκινήσετε γρήγορα. Ωστόσο, καθώς το έργο εξελίσσεται, οι προγραμματιστές συχνά προσαρμόζουν τη δομή ώστε να ανταποκρίνεται στις συγκεκριμένες ανάγκες τους.

10.7 Dependency - Εξάρτηση

Σε ένα έργο React, μια εξάρτηση αναφέρεται σε ένα εξωτερικό πακέτο ή μια βιβλιοθήκη στην οποία βασίζεται το έργο μας για να παρέχει συγκεκριμένη λειτουργικότητα. Οι εξαρτήσεις διαχειρίζονται και εγκαθίστανται χρησιμοποιώντας διαχειριστές πακέτων όπως το npm (Node Package Manager) ή το Yarn. Αυτά τα εξωτερικά πακέτα ή βιβλιοθήκες μπορεί να περιλαμβάνουν ένα ευρύ φάσμα λειτουργιών, όπως βοηθητικές λειτουργίες, στοιχεία UI, εργαλεία διαχείρισης καταστάσεων, συστήματα δρομολόγησης και άλλα.

Οι εξαρτήσεις είναι απαραίτητες επειδή επιτρέπουν να αξιοποιούμε τον υπάρχοντα κώδικα και τις λύσεις που έχουν δημιουργηθεί από την κοινότητα ανοικτού κώδικα ή από τρίτους προγραμματιστές αντί να ανακαλύπτουμε εκ νέου τον τροχό για κάθε πτυχή του έργου. Αυτό μπορεί να επιταχύνει σημαντικά την ανάπτυξη και να βελτιώσει τη συντηρησιμότητα της εφαρμογής.

Ακολουθεί ο τρόπος με τον οποίο λειτουργούν οι εξαρτήσεις σε ένα έργο React.

10.7.1 Διαχείριση πακέτων

Συνήθως ορίζονται οι εξαρτήσεις του έργου σας σε ένα αρχείο package.json, το οποίο βρίσκεται στη ρίζα του έργου σας. Αυτό το αρχείο περιέχει μια λίστα με όλα τα πακέτα από τα οποία εξαρτάται το έργο, μαζί με τις εκδόσεις τους.

10.7.2 Εγκατάσταση

Για να εγκαταστήσουμε τις καθορισμένες εξαρτήσεις, εκτελούμε μια εντολή όπως η npm install ή η yarn install στον κατάλογο του έργου. Αυτή η εντολή διαβάζει το αρχείο package.json και κατεβάζει τα απαιτούμενα πακέτα από το μητρώο npm ή ένα άλλο μητρώο πακέτων όπως το yarn.

10.7.3 Ενότητες κόμβου

Αφού εγκατασταθούν, τα πακέτα αποθηκεύονται σε έναν κατάλογο που ονομάζεται node_modules εντός του ριζικού καταλόγου του έργου σας. Αυτός ο κατάλογος περιέχει τον κώδικα για όλες τις εξαρτήσεις του έργου μας.

10.7.4 Εισαγωγή εξαρτήσεων

Στον κώδικα της React μας, μπορούμε να εισάγουμε και να χρησιμοποιήσουμε συναρτήσεις, κλάσεις, στοιχεία ή άλλους πόρους από αυτά τα εγκατεστημένα πακέτα. Για παράδειγμα, μπορεί να εισάγουμε ένα στοιχείο της React από μια βιβλιοθήκη UI όπως η Material-UI ή μια βιβλιοθήκη διαχείρισης καταστάσεων όπως η Redux.

Ακολουθεί ένα παράδειγμα για το πώς μπορείτε να εισάγετε και να χρησιμοποιήσετε μια εξάρτηση σε ένα στοιχείο React:

Παράδειγμα:

```
import React from 'react';
import axios from 'axios'; // Axios is a popular HTTP client library

class MyComponent extends React.Component {
  componentDidMount() {
    // Use Axios to make an HTTP request
    axios.get('https://api.example.com/data')
      .then(response => {
        // Handle the response data
        console.log(response.data);
      })
      .catch(error => {
        // Handle any errors
        console.error(error);
      });
  }

  render() {
    return (
      // Your component's rendering logic here
    );
  }
}

export default MyComponent;
```

Σε αυτό το παράδειγμα, το axios είναι μια εξάρτηση που εισάγεται και χρησιμοποιείται για την πραγματοποίηση αιτημάτων HTTP εντός του στοιχείου React.

Οι εξαρτήσεις μπορεί να διαφέρουν σε μεγάλο βαθμό ως προς τον σκοπό και την πολυπλοκότητά τους, αλλά παίζουν καθοριστικό ρόλο στην κατασκευή εφαρμογών React,

παρέχοντας επαναχρησιμοποιήσιμο κώδικα και λειτουργικότητα για την ενίσχυση του έργου σας.