Πανεπιστήμιο Πελοποννήσου
Τμήμα Πληροφορικής και Τηλεπικοινωνιών

# Ανάπτυξη εφαρμογής διαχείρισης εξόδων για κινητά τερματικά


**Στεφανία Τζανερά**
2022202202017

*Επιβλέπων:* Νικόλαος Τσελίκας - Καθηγητής

Διπλωματική Εργασία

Απρίλιος 2024

University of Peloponnese
Department of Informatics and Telecommunications

# Expense Manager Mobile Application


**Stefania Tzanera**
2022202202017

*Supervisor :* Nikolaos Tselikas - Professor

Diploma thesis for the Master's program "Computer Science"

April 2024

# Abstract

The development of applications on Android mobile terminals has been underway since 2008 with one of the first applications being "Barcode Scanner", which allowed users to scan Barcodes and QR codes using their smartphone's camera. Although by today's standards it is considered a simple and taken-for-granted application, back then it was something pioneering and showed other possibilities of mobile devices beyond traditional communication. Now with the passage of time, we see that this field is constantly growing and applications are being released that aim to facilitate the user in their daily life. In this master thesis, we will present the implementation of a new Android application, Expense Manager. It is an idea that aims to help the user in better managing his/her finances.In recent years with the increased use of electronic payment methods, the average person often cannot manage his expenses and as a result he loses control. Through it, the user can store his financial data in order to better manage them. The innovative thing about this app is the use of QR scanner through which he can scan the receipts and add/remove/transfer monetary items. Further, we will talk about the general features of Kotlin that were used for the functionality of the app. Finally, we will comment on difficulties encountered during the implementation of the application and future plans for the development of the application.

**Keywords**: Kotlin, QR, Barcode, Android, applications, Expense Manager, Asynchronous Web Crawling, URL, smartphones, Asynchronous Programming, Coroutines

# Περίληψη

Η ανάπτυξη των εφαρμογών σε κινητά τερματικά με Android λειτουργικό έχει ξεκινήσει από το 2008 με μία από τις πρώτες εφαρμογές να είναι η "Barcode Scanner", η οποία επέτρεπε στους χρήστες να σαρώνουν Barcodes και κωδικούς QR χρησιμοποιώντας την κάμερα του smartphone τους. Αν και για τα σημερινά δεδομένα θεωρείται μία απλή και δεδομένη εφαρμογή, τότε ήταν κάτι πρωτοπόρο και έδειχνε και άλλες δυνατότητες των κινητών συσκευών πέρα από την παραδοσιακή επικοινωνία. Πλέον με το πέρασμα του χρόνου, βλέπουμε ότι ο τομέας αυτός διαρκώς αναπτύσσεται και κυκλοφορούν εφαρμογές που στόχο έχουν τη διευκόλυνση του χρήστη στην καθημερινότητά του. Σε αυτήν τη μεταπτυχιακή διπλωματική εργασία, θα παρουσιάσουμε την υλοποίηση μιας νέας εφαρμογής Android, την Expense Manager. Αποτελεί μία ιδέα που σκοπό έχει να βοηθήσει τον χρήστη στην καλύτερη διαχείριση των οικονομικών του. Μέσω αυτής, ο user μπορεί να αποθηκεύσει τα οικονομικά του στοιχεία με στόχο την καλύτερη διαχείριση τους. Το καινοτόμο σε αυτήν την εφαρμογή είναι η χρήση του QR scanner μέσω του οποίου μπορεί να σκανάρει τις αποδείξεις και να προσθέτει/αφαιρεί/μεταφέρει χρηματικά στοιχεία. Ακόμη, θα μιλήσουμε για τα γενικά χαρακτηριστικά της Kotlin που χρησιμοποιήθηκαν για τη λειτουργικότητα της εφαρμογής. Τέλος, θα σχολιαστούν δυσκολίες που υπήρξαν κατά την υλοποίηση της εφαρμογής καθώς και μελλοντικά πλάνα για την ανάπτυξη της εφαρμογής.

**Λέξεις-κλειδιά**: κινητά τερματικά, εφαρμογή, κωδικοί γρήγορης απόκρισης , υπορουτίνες, διαχείριση εξόδων, προσθήκη χρημάτων, αφαίρεση χρημάτων, μεταφορά χρημάτων, Ασύγχρονος Προγραμματισμός

# Περιεχόμενα

# Κατάλογος Σχημάτων

# Κατάλογος αποσπασμάτων κώδικα

# 1 Εισαγωγή

Η παγκόσμια οικονομία συνεχίζει να εξελίσσεται και να ψηφιοποιείται, ενώ η αποτελεσματική διαχείριση των χρηματοοικονομικών πόρων και η προσεκτική εποπτεία των συναλλαγών θεωρούνται υψίστης σημασίας τόσο για τα άτομα μεμονωμένα όσο και για τους οργανισμούς.

## 1.1 Τρόποι Πληρωμής

Στον σημερινό κόσμο, παρατηρείται πληθώρα μέσων διευκόλυνσης των συναλλαγών, τα οποία ανταποκρίνονται στις διαφορετικές προτιμήσεις και τις ανάγκες των ατόμων, καθώς κινούνται παράλληλα με τις τεχνολογικές εξελίξεις. Οι μέθοδοι πληρωμής έχουν εξελιχθεί σημαντικά από τις παραδοσιακές συναλλαγές με μετρητά και περιλαμβάνουν ψηφιακές εναλλακτικές λύσεις που προσφέρουν ευκολία, ασφάλεια και αποτελεσματικότητα.

Τα μετρητά, η παλαιότερη μορφή πληρωμής, παραμένει ένα ευρέως αποδεκτό μέσο για συναλλαγές. Ωστόσο, η άνοδος των ηλεκτρονικών μεθόδων πληρωμής έχει φέρει επανάσταση. Οι πιστωτικές, οι χρεωστικές και οι προπληρωμένες κάρτες παρέχουν έναν βολικό και ασφαλή τρόπο για την πραγματοποίηση αγορών τόσο αυτοπροσώπως όσο και διαδικτυακά. Τα τραπεζικά εμβάσματα επιτρέπουν την άμεση μετακίνηση κεφαλαίων μεταξύ λογαριασμών, προσφέροντας έναν εύκολο τρόπο πραγματοποίησης πληρωμών και διακανονισμού οφειλών. Οι λύσεις που έχουν δοθεί για πληρωμές μέσω κινητών τηλεφώνων έχουν κερδίσει δημοτικότητα τα τελευταία χρόνια, επιτρέποντας στους χρήστες να πραγματοποιούν συναλλαγές χρησιμοποιώντας απλώς τις κινητές τους συσκευές. Τα ηλεκτρονικά πορτοφόλια και οι εφαρμογές πληρωμών peer-to-peer έχουν γίνει κοινός τόπος, προσφέροντας ανέπαφους και αποτελεσματικούς τρόπους μεταφοράς χρημάτων και πραγματοποίησης αγορών. Επιπλέον, η εμφάνιση των διαδικτυακών πλατφόρμων πληρωμών έχει διευκολύνει τις συναλλαγές ηλεκτρονικού εμπορίου, παρέχοντας ασφαλή κανάλια για τις επιχειρήσεις και τους καταναλωτές ώστε να συμμετέχουν στο διαδικτυακό εμπόριο.

Σε αυτό το ποικιλόμορφο τοπίο μεθόδων πληρωμής, οι ιδιώτες και οι επιχειρήσεις έχουν να διαλέξουν από μια πληθώρα επιλογών, καθεμία από τις οποίες έχει τα δικά της πλεονεκτήματα και εκτιμήσεις. Είτε πρόκειται για την ευκολία που προσφέρει η χρήση μιας κάρτας, είτε για την ασφάλεια των ηλεκτρονικών συναλλαγών, το σύγχρονο οικοσύστημα πληρωμών προσφέρει λύσεις που ανταποκρίνονται στις εξελισσόμενες ανάγκες τόσο των καταναλωτών όσο και των επιχειρήσεων.

Ορισμένες συνήθεις κατηγορίες και μέθοδοι πληρωμής είναι:

1. **Μετρητά:** ο πιο παραδοσιακός τρόπος πληρωμής που αποτελείται από φυσικά κέρματα και χαρτονομίσματα.

2. **Κάρτα:**

   Πιστωτική: Επιτρέπει στους χρήστες να δανείζονται χρήματα μέχρι ένα όριο.

   Χρεωστική: Συνδέεται απευθείας με τραπεζικό λογαριασμό και αφαιρεί χρήματα απευθείας από αυτόν.

Προπληρωμένη: Φορτίζεται με ένα συγκεκριμένο χρηματικό ποσό μέχρι να εξαντληθεί το υπόλοιπο.

3. **Τραπεζικές μεταφορές:** Άμεση μεταφορά χρημάτων από έναν τραπεζικό λογαριασμό σε έναν άλλο.

4. **Mobile πληρωμές:**

   Mobile πορτοφόλια: Εφαρμογές όπως το Apple Pay, το Google Pay ή το Samsung Pay που αποθηκεύουν με ασφάλεια πληροφορίες πιστωτικών/χρεωστικών καρτών για ανέπαφες πληρωμές μέσω smartphones ή smartwatches.

   Εφαρμογές πληρωμών peer-to-peer (P2P): Υπηρεσίες όπως το Venmo, το Cash App ή το PayPal που επιτρέπουν σε άτομα να στέλνουν χρήματα μεταξύ τους ηλεκτρονικά.

5. **Διαδικτυακές πλατφόρμες πληρωμών:**

   PayPal: Επιτρέπει στους χρήστες να πραγματοποιούν πληρωμές στο διαδίκτυο με ασφάλεια.

   Stripe: Πλατφόρμα επεξεργασίας πληρωμών που επιτρέπει στις επιχειρήσεις να δέχονται πληρωμές μέσω του διαδικτύου.

   Square: Προσφέρει λύσεις πληρωμών για επιχειρήσεις, συμπεριλαμβανομένων αναγνωστών καρτών και ηλεκτρονικής τιμολόγησης.

6. **Πληρωμές χωρίς επαφή:**

   NFC (Near Field Communication): Επιτρέπει στις συσκευές να επικοινωνούν όταν βρίσκονται κοντά μεταξύ τους, χρησιμοποιείται συχνά για ανέπαφες πληρωμές.

   Κωδικοί QR: Μπορούν να σαρωθούν για να ξεκινήσει μια συναλλαγή πληρωμής.

7. **Επιταγές:** Γραπτές εντολές για την πληρωμή συγκεκριμένου χρηματικού ποσού από ένα πρόσωπο ή μια επιχείρηση σε μια άλλη.

Αυτές είναι μερικές από τις κύριες μεθόδους, αλλά η διαθεσιμότητα μπορεί να διαφέρει ανάλογα με τη χώρα, την επιχείρηση και τις τεχνολογικές εξελίξεις. Η διαχείριση των οικονομικών δεδομένων στον σημερινό κόσμο μπορεί πράγματι να αποτελέσει πρόκληση λόγω του πλήθους των διαθέσιμων επιλογών πληρωμής. Με τις πιστωτικές κάρτες, τις χρεωστικές κάρτες, τις εφαρμογές πληρωμών μέσω κινητού, τις ηλεκτρονικές τραπεζικές συναλλαγές και πολλά άλλα, η παρακολούθηση των δαπανών και η τήρηση του προϋπολογισμού μπορεί να γίνει αρκετά περίπλοκη.

## 1.2 Διαχείριση Εξόδων

Μία από τις πρωταρχικές δυσκολίες ενός ατόμου είναι ο πειρασμός της υπερκατανάλωσης, καθώς η ευκολία και η άνεση των ψηφιακών πληρωμών μπορεί μερικές φορές να θολώσει τα όρια μεταξύ επιθυμιών και αναγκών. Επιπλέον, η διαχείριση πολλαπλών λογαριασμών σε διάφορες πλατφόρμες μπορεί να οδηγήσει σε σύγχυση και πιθανή έλλειψη ελέγχου των συναλλαγών. Επίσης, κάθε μέθοδος πληρωμής συνοδεύεται από τα δικά της ζητήματα ασφάλειας, όπως η προστασία από απάτες και μη εξουσιοδοτημένη πρόσβαση. Αυτό απαιτεί επαγρύπνηση στην παρακολούθηση των λογαριασμών και την εφαρμογή ισχυρών μέτρων ασφαλείας. Ο προϋπολογισμός και ο οικονομικός προγραμματισμός καθίστανται υψίστης σημασίας για την αποτελεσματική πλοήγηση σε αυτό το τοπίο. Η υιοθέτηση εργαλείων, όπως εφαρμογές προϋπολογισμού ή λογιστικά φύλλα, μπορεί να βοηθήσει τα άτομα να παρακολουθούν τις δαπάνες τους και να διατηρούν οικονομική πειθαρχία. Ακόμη, η ενημέρωση σχετικά με τις βέλτιστες πρακτικές προσωπικής χρηματοδότησης και η τακτική επανεξέταση της οικονομικής κατάστασης του ατόμου μπορεί να βοηθήσει στη λήψη τεκμηριωμένων αποφάσεων εν μέσω της πληθώρας των επιλογών πληρωμής.

Η καλή διαχείριση των δαπανών απαιτεί πειθαρχία, οργάνωση και βαθιά κατανόηση των οικονομικών συνηθειών και στόχων που έχει θέσει το ίδιο το άτομο. Ο πολλαπλασιασμός των επιλογών πληρωμής προσθέτει επίπεδα πολυπλοκότητας σε αυτό το έργο. Ακολουθούν ορισμένες συγκεκριμένες προκλήσεις που μπορεί να αντιμετωπίσουν τα άτομα:

1. **Παρακολούθηση των δαπανών σε πολλαπλές πλατφόρμες:** Με διάφορους τρόπους πληρωμής, όπως πιστωτικές κάρτες, χρεωστικές κάρτες και εφαρμογές πληρωμών μέσω κινητού, μπορεί να είναι δύσκολο να συγκεντρώσετε όλα τα έξοδα σε ένα μέρος για ανάλυση. Κάθε πλατφόρμα μπορεί να έχει τη δική της διεπαφή και τα δικά της εργαλεία αναφοράς, γεγονός που καθιστά δύσκολη την ολιστική εικόνα των συνηθειών δαπανών του ατόμου.

2. **Καθυστερημένη αναγνώριση των δαπανών:** Ορισμένες μέθοδοι πληρωμής, όπως οι πιστωτικές κάρτες, μπορεί να επιτρέπουν την αναβολή πληρωμής, το οποίο σημαίνει ότι τα έξοδα πραγματοποιούνται κατά τη στιγμή της αγοράς αλλά εξοφλούνται σε μεταγενέστερη ημερομηνία. Αυτή η καθυστέρηση μπορεί να καταστήσει εύκολη την απώλεια της παρακολούθησης των δαπανών και την υπερκατανάλωση χωρίς άμεσες συνέπειες.

3. **Αυθόρμητες αγορές και εύκολη πρόσβαση σε πίστωση:** Η ευκολία των ψηφιακών πληρωμών, σε συνδυασμό με τη διαθεσιμότητα πίστωσης, μπορεί να οδηγήσει σε παρορμητικές δαπάνες. Με λίγα μόνο "κλικ" ή πατήματα, τα άτομα μπορούν να κάνουν αγορές χωρίς να εξετάσουν πλήρως τις οικονομικές επιπτώσεις, θέτοντας ενδεχομένως σε κίνδυνο τον προϋπολογισμό τους.

4. **Υπερφόρτωση συνδρομών:** Οι υπηρεσίες που βασίζονται σε συνδρομές είναι όλο και πιο διαδεδομένες, καλύπτοντας τα πάντα, από τη ροή ψυχαγωγίας μέχρι την παράδοση γευμάτων. Ενώ μεμονωμένα αυτές οι συνδρομές μπορεί να φαίνονται

προσιτές, το σωρευτικό τους κόστος μπορεί να αθροιστεί σημαντικά με την πάροδο του χρόνου, ειδικά αν δεν γίνεται ενεργή διαχείριση.

5. **Ανακριβής προϋπολογισμός:** Χωρίς σαφή κατανόηση των εσόδων και των εξόδων κάποιου, ο προϋπολογισμός καθίσταται αναποτελεσματικός. Η μη ακριβής παρακολούθηση των δαπανών μπορεί να οδηγήσει σε υπερκατανάλωση σε ορισμένες κατηγορίες και παραμέληση άλλων, οδηγώντας σε οικονομική ανισορροπία.

6. **Ανησυχίες για την ασφάλεια:** Ενώ οι ψηφιακές πληρωμές προσφέρουν ευκολία, παρουσιάζουν επίσης κινδύνους ασφαλείας, όπως κλοπή ταυτότητας, απάτη και παραβίαση δεδομένων. Η διασφάλιση των προσωπικών και οικονομικών πληροφοριών είναι απαραίτητη για την αποτροπή μη εξουσιοδοτημένων συναλλαγών και την προστασία από οικονομικές απώλειες.

Για να ξεπεράσουν αυτές τις προκλήσεις, τα άτομα πρέπει να υιοθετήσουν στρατηγικές όπως η δημιουργία ενός λεπτομερούς προϋπολογισμού, η τακτική επανεξέταση των δαπανών, ο καθορισμός ορίων δαπανών, η ιεράρχηση των αναγκών έναντι των επιθυμιών και η αξιοποίηση της τεχνολογίας για τον εξορθολογισμό της οικονομικής διαχείρισης. Επιπλέον, η καλλιέργεια προσεκτικών συνηθειών δαπανών και η αναζήτηση υποστήριξης από επαγγελματίες μπορούν να ωθήσουν τα άτομα να αναλάβουν αποτελεσματικά τον έλεγχο των οικονομικών τους.

## 1.3 Αποδείξεις και ηλεκτρονικά παραστατικά

Οι αποδείξεις και τα ηλεκτρονικά έγγραφα διαδραματίζουν κρίσιμο ρόλο στο σύγχρονο εμπόριο και τις οικονομικές συναλλαγές. Παραδοσιακά, οι αποδείξεις ήταν έγγραφα σε χαρτί που παρέχονταν στους πελάτες ως απόδειξη αγοράς, στα οποία αναφέρονταν λεπτομερώς τα είδη που αγοράστηκαν, οι τιμές τους και το συνολικό ποσό που καταβλήθηκε. Ωστόσο, με την ψηφιοποίηση των επιχειρηματικών δραστηριοτήτων, οι ηλεκτρονικές αποδείξεις έχουν επικρατήσει όλο και περισσότερο.

Οι ηλεκτρονικές αποδείξεις, οι οποίες συχνά αποστέλλονται με ηλεκτρονικό ταχυδρομείο ή μέσω εφαρμογών για κινητά τηλέφωνα, προσφέρουν αρκετά πλεονεκτήματα σε σχέση με τις παραδοσιακές χάρτινες αποδείξεις. Είναι φιλικές προς το περιβάλλον, μειώνοντας τα απορρίμματα χαρτιού και την ανάγκη για φυσική αποθήκευση. Είναι επίσης πιο βολικές τόσο για τις επιχειρήσεις όσο και για τους πελάτες, καθώς μπορούν εύκολα να αποθηκευτούν, να προσπελαστούν και να αναζητηθούν ψηφιακά. Επιπλέον, τα ηλεκτρονικά έγγραφα, συμπεριλαμβανομένων των τιμολογίων, των συμβάσεων και των δηλώσεων, έχουν καταστεί βασικά στοιχεία των επιχειρηματικών δραστηριοτήτων. Τα έγγραφα αυτά διευκολύνουν τις συναλλαγές, την τήρηση αρχείων και τη συμμόρφωση με τις νομικές και κανονιστικές απαιτήσεις. Με την έλευση των συστημάτων ηλεκτρονικής τιμολόγησης (e-invoicing), οι επιχειρήσεις μπορούν να δημιουργούν, να αποστέλλουν και να επεξεργάζονται τιμολόγια ηλεκτρονικά, βελτιώνοντας τη διαδικασία τιμολόγησης και μειώνοντας τα λάθη και τις καθυστερήσεις που συνδέονται με τη χειροκίνητη τιμολόγηση. Τα ηλεκτρονικά έγγραφα προσφέρουν πλεονεκτήματα όπως ταχύτερη επεξεργασία, βελτιωμένη

ακρίβεια, μειωμένο διοικητικό κόστος και αυξημένη ασφάλεια. Μπορούν να κρυπτογραφηθούν, να υπογραφούν ψηφιακά και να αποθηκευτούν σε ασφαλείς βάσεις δεδομένων ή συστήματα αποθήκευσης στο νέφος, προστατεύοντας τις ευαίσθητες πληροφορίες από μη εξουσιοδοτημένη πρόσβαση και παραποίηση. Ωστόσο, παρά τα πλεονεκτήματα των ηλεκτρονικών εγγράφων, υπάρχουν προκλήσεις για την ευρεία υιοθέτησή τους. Οι προκλήσεις αυτές περιλαμβάνουν ανησυχίες σχετικά με το απόρρητο και την ασφάλεια των δεδομένων, τη διαλειτουργικότητα μεταξύ διαφορετικών συστημάτων και μορφών, τη συμμόρφωση με τις κανονιστικές διατάξεις και το ψηφιακό χάσμα, το οποίο μπορεί να περιορίσει την πρόσβαση σε συστήματα ηλεκτρονικών εγγράφων για ορισμένα άτομα ή επιχειρήσεις.

Συνολικά, οι αποδείξεις και τα ηλεκτρονικά έγγραφα αποτελούν βασικά στοιχεία των σύγχρονων επιχειρήσεων και του εμπορίου, διευκολύνοντας τις αποτελεσματικές και ασφαλείς συναλλαγές, την τήρηση αρχείων και τη συμμόρφωση με τις νομικές και κανονιστικές απαιτήσεις. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, τα συστήματα ηλεκτρονικής διαχείρισης εγγράφων θα διαδραματίζουν ολοένα και σημαντικότερο ρόλο στη βελτιστοποίηση των επιχειρηματικών διαδικασιών και στην ενίσχυση της παραγωγικότητας και της διαφάνειας στις οικονομικές συναλλαγές.

Στην Ελλάδα, οι αποδείξεις και τα ηλεκτρονικά έγγραφα έχουν υποστεί σημαντικές αλλαγές τα τελευταία χρόνια, λόγω κανονιστικών μεταρρυθμίσεων και τεχνολογικών εξελίξεων. Παραδοσιακά, οι χάρτινες αποδείξεις ήταν ο κανόνας, οι οποίες εκδίδονταν από τις επιχειρήσεις στους πελάτες ως απόδειξη αγοράς. Ωστόσο, με την εισαγωγή συστημάτων ηλεκτρονικής τιμολόγησης (e-invoicing) και τις ψηφιακές φορολογικές μεταρρυθμίσεις, τα ηλεκτρονικά έγγραφα έχουν αποκτήσει εξέχουσα θέση στο ελληνικό επιχειρηματικό τοπίο. Μια αξιοσημείωτη εξέλιξη είναι η εφαρμογή ηλεκτρονικών φορολογικών συστημάτων (EFS) για την έκδοση αποδείξεων και τιμολογίων. Σύμφωνα με το πλαίσιο ΕΦΣ, οι επιχειρήσεις υποχρεούνται να χρησιμοποιούν ηλεκτρονικές ταμειακές μηχανές ή συστήματα σημείων πώλησης (POS) που παράγουν ηλεκτρονικές αποδείξεις με μοναδικά αναγνωριστικά και ψηφιακές υπογραφές. Αυτές οι ηλεκτρονικές αποδείξεις αποθηκεύονται κεντρικά και μπορούν να έχουν πρόσβαση σε αυτές και να τις επαληθεύουν με ευκολία οι φορολογικές αρχές. Επιπλέον, η Ελλάδα έχει υιοθετήσει την ηλεκτρονική τιμολόγηση (e-invoicing) στο πλαίσιο των προσπαθειών της για την καταπολέμηση της φοροδιαφυγής και τη βελτίωση της φορολογικής συμμόρφωσης. Οι επιχειρήσεις μεταβαίνουν ολοένα και περισσότερο από την τιμολόγηση σε χαρτί σε συστήματα ηλεκτρονικής τιμολόγησης, τα οποία απλοποιούν τη διαδικασία αυτή, μειώνουν το διοικητικό φόρτο και ενισχύουν την ακρίβεια και τη διαφάνεια στις οικονομικές συναλλαγές. Επιπλέον, η ελληνική κυβέρνηση έχει δημιουργήσει πλατφόρμες όπως το myDATA (ΜονοΔιάστατα Αρχεία Τηλεφορολογικών Αρχείων) για τη διευκόλυνση της ηλεκτρονικής ανταλλαγής δεδομένων που σχετίζονται με τη φορολογία μεταξύ επιχειρήσεων και φορολογικών αρχών. Μέσω του myDATA, οι επιχειρήσεις μπορούν να υποβάλλουν ηλεκτρονικά έγγραφα, συμπεριλαμβανομένων αποδείξεων, τιμολογίων και άλλων οικονομικών αρχείων, στις φορολογικές αρχές σε πραγματικό χρόνο ή σε περιοδική βάση. Η εισαγωγή των ηλεκτρονικών εγγράφων στην Ελλάδα χαρακτηρίζεται από πολλά οφέλη, όπως η βελτίωση της αποτελεσματικότητας, η μείωση της γραφειοκρατίας, η ενίσχυση της διαφάνειας και η καλύτερη συμμόρφωση με

τους φορολογικούς κανονισμούς. Ωστόσο, η μετάβαση σε ηλεκτρονικά συστήματα έχει επίσης δημιουργήσει προκλήσεις για τις επιχειρήσεις, συμπεριλαμβανομένης της ανάγκης να επενδύσουν σε νέες τεχνολογίες, να διασφαλίσουν την ασφάλεια και την προστασία της ιδιωτικής ζωής των δεδομένων και να συμμορφωθούν με τις κανονιστικές απαιτήσεις. Συνολικά, οι αποδείξεις και τα ηλεκτρονικά έγγραφα έχουν υποστεί μετασχηματισμό στην Ελλάδα, μεταβαίνοντας από τα παραδοσιακά χάρτινα συστήματα σε ηλεκτρονικά συστήματα που προσφέρουν μεγαλύτερη αποτελεσματικότητα, διαφάνεια και συμμόρφωση με τους φορολογικούς κανονισμούς. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, τα συστήματα ηλεκτρονικής διαχείρισης εγγράφων θα διαδραματίζουν όλο και πιο σημαντικό ρόλο στον εκσυγχρονισμό των επιχειρηματικών πρακτικών και στη βελτίωση της οικονομικής διαφάνειας και της λογοδοσίας στην Ελλάδα.

Στην Ελλάδα, οι αποδείξεις συχνά περιλαμβάνουν κωδικούς QR ως μέρος των προσπαθειών της χώρας να εκσυγχρονίσει τη φορολογική συμμόρφωση και να ενισχύσει τη διαφάνεια στις χρηματοοικονομικές συναλλαγές.

Οι κωδικοί QR στις ελληνικές αποδείξεις χρησιμεύουν ως μέσο παροχής πρόσθετων πληροφοριών και επαλήθευσης για τη συναλλαγή. Αποτελούν μέρος του Ηλεκτρονικού Δημοσιονομικού Συστήματος (EFS) που εφαρμόζει η ελληνική κυβέρνηση για την καταπολέμηση της φοροδιαφυγής και τη βελτίωση της φορολογικής συμμόρφωσης. Οι κωδικοί QR περιέχουν συνήθως βασικές λεπτομέρειες σχετικά με τη συναλλαγή, όπως την ημερομηνία, την ώρα, τα στοιχεία του εμπόρου και το ποσό που καταβλήθηκε. Αυτές οι πληροφορίες επιτρέπουν στα άτομα να επαληθεύσουν τη γνησιότητα της απόδειξης και να διασφαλίσουν ότι αντιστοιχεί με ακρίβεια στη συναλλαγή. Σαρώνοντας τον κωδικό QR χρησιμοποιώντας ένα smartphone ή έναν σαρωτή κώδικα QR, τα άτομα μπορούν να έχουν πρόσβαση στα υποκείμενα δεδομένα που είναι ενσωματωμένα στον κώδικα. Αυτό τους δίνει τη δυνατότητα να επαληθεύουν τις λεπτομέρειες της συναλλαγής και να επιβεβαιώνουν ότι έχει καταγραφεί με ακρίβεια από τον έμπορο και ότι έχει διαβιβαστεί στις φορολογικές αρχές. Ο κωδικός QR μπορεί να ανακατευθύνει τα άτομα σε ιστότοπο ή πλατφόρμα που διαχειρίζεται η ελληνική φορολογική αρχή (Ανεξάρτητη Αρχή Δημοσίων Εσόδων - ΑΑΔΕ), όπου μπορούν να έχουν πρόσβαση σε λεπτομερείς πληροφορίες σχετικά με τη συναλλαγή. Αυτές οι πληροφορίες παρέχουν διαφάνεια και υπευθυνότητα στις οικονομικές συναλλαγές, δίνοντας τη δυνατότητα στα άτομα να παρακολουθούν τις δαπάνες τους και να εντοπίζουν τυχόν αποκλίσεις. Οι κωδικοί QR στις αποδείξεις αποτελούν ρυθμιστική απαίτηση για τις επιχειρήσεις στην Ελλάδα, η οποία επιβάλλεται από την κυβέρνηση ως μέρος του EFS. Οι επιχειρήσεις υποχρεούνται να εκδίδουν αποδείξεις με κωδικούς QR για τη διασφάλιση της συμμόρφωσης με τους φορολογικούς κανονισμούς και τη διευκόλυνση της ηλεκτρονικής ανταλλαγής δεδομένων συναλλαγών με τις φορολογικές αρχές.

Συνολικά, οι κωδικοί QR στις ελληνικές αποδείξεις διαδραματίζουν ζωτικό ρόλο στον εκσυγχρονισμό της φορολογικής συμμόρφωσης, στην ενίσχυση της διαφάνειας και στην ενδυνάμωση των ατόμων να επαληθεύουν την ακρίβεια των οικονομικών τους συναλλαγών. Αποτελούν ένα απτό παράδειγμα του τρόπου με τον οποίο η τεχνολογία αξιοποιείται για τη βελτίωση της λογοδοσίας και την καταπολέμηση της φοροδιαφυγής στο χρηματοπιστωτικό οικοσύστημα της Ελλάδας.

## 1.4 Εφαρμογές για παρακολούθηση οικονομικών συναλλαγών

Η έλλειψη εφαρμογών για τη διαχείριση των οικονομικών ενός χρήστη είναι ένα πολύπλευρο ζήτημα που προκύπτει από διάφορες προκλήσεις στο τοπίο των προσωπικών οικονομικών στοιχείων.

Μία από τις πρωταρχικές προκλήσεις είναι ο κατακερματισμός των οικονομικών εργασιών. Τα προσωπικά οικονομικά περιλαμβάνουν ένα ευρύ φάσμα δραστηριοτήτων, όπως ο προϋπολογισμός, η παρακολούθηση δαπανών, η διαχείριση επενδύσεων, ο προγραμματισμός συνταξιοδότησης και πολλά άλλα. Ενώ υπάρχουν διαθέσιμες εφαρμογές για την αντιμετώπιση καθεμιάς από αυτές τις εργασίες ξεχωριστά, η εύρεση μιας ενιαίας εφαρμογής που να ενσωματώνει απρόσκοπτα όλες τις πτυχές της οικονομικής διαχείρισης μπορεί να είναι δύσκολη. Ως αποτέλεσμα, οι χρήστες συχνά καταφεύγουν στη χρήση πολλαπλών εφαρμογών, με αποτέλεσμα τον κατακερματισμό των οικονομικών δεδομένων και των ροών εργασίας. Μια άλλη σημαντική πρόκληση είναι η πολυπλοκότητα των οικονομικών δεδομένων. Η αποτελεσματική διαχείριση αυτών στο πλαίσιο μιας ενιαίας εφαρμογής απαιτεί ισχυρή υποδομή και εξελιγμένους αλγόριθμους, ιδίως λαμβάνοντας υπόψη τις ποικίλες πηγές οικονομικών πληροφοριών, όπως είναι οι τραπεζικοί λογαριασμοί, πιστωτικές κάρτες, επενδυτικά χαρτοφυλάκια και άλλα. Η διασφάλιση της ακρίβειας και της ασφάλειας αυτών των δεδομένων είναι απαραίτητη για την επιτυχία των εφαρμογών οικονομικής διαχείρισης. Οι ανησυχίες για το απόρρητο και την ασφάλεια παίζουν επίσης καθοριστικό ρόλο στον περιορισμό της διαθεσιμότητας και της υιοθέτησης των εφαρμογών διαχείρισης οικονομικών. Οι χρήστες είναι λογικό να είναι επιφυλακτικοί όσον αφορά την ανταλλαγή ευαίσθητων προσωπικών και οικονομικών πληροφοριών με εφαρμογές τρίτων, ειδικά αν δεν είναι σίγουροι για τα μέτρα ασφαλείας και τις πρακτικές χειρισμού δεδομένων της εφαρμογής. Η οικοδόμηση εμπιστοσύνης με τους χρήστες και η αντιμετώπιση των ανησυχιών τους για την προστασία της ιδιωτικής τους ζωής είναι ζωτικής σημασίας για τους προγραμματιστές εφαρμογών σε αυτόν τον χώρο. Επιπλέον, η εμπειρία του χρήστη και ο σχεδιασμός της διεπαφής είναι σημαντικοί παράγοντες που επηρεάζουν την αποτελεσματικότητα και την υιοθέτηση των εφαρμογών διαχείρισης οικονομικών. Οι χρήστες αναμένουν διαισθητικές διεπαφές, εύκολη πλοήγηση και προσαρμόσιμες λειτουργίες που να ανταποκρίνονται στις ατομικές τους προτιμήσεις και τους οικονομικούς τους στόχους. Οι εφαρμογές που δεν ανταποκρίνονται σε αυτές τις προσδοκίες ενδέχεται να δυσκολευτούν να προσελκύσουν και να διατηρήσουν χρήστες σε μια ανταγωνιστική αγορά.Επιπλέον, η κανονιστική συμμόρφωση προσθέτει άλλο ένα επίπεδο πολυπλοκότητας στην ανάπτυξη και λειτουργία των εφαρμογών διαχείρισης οικονομικών. Οι προγραμματιστές πρέπει να διασφαλίζουν τη συμμόρφωση με τους σχετικούς κανονισμούς και πρότυπα, ιδίως όσον αφορά την προστασία των δεδομένων, τις οικονομικές συναλλαγές και τους νόμους περί προστασίας της ιδιωτικής ζωής.Η επίτευξη και η διατήρηση της συμμόρφωσης μπορεί να αποτελέσει πρόκληση, ιδίως για τους μικρότερους προγραμματιστές εφαρμογών με περιορισμένους πόρους.

Συμπερασματικά, η αντιμετώπιση της έλλειψης εφαρμογών για τη διαχείριση των οικονομικών ενός χρήστη απαιτεί συνδυασμό τεχνολογικής καινοτομίας, σχεδιασμού με

επίκεντρο τον χρήστη, ισχυρών μέτρων ασφαλείας και συμμόρφωσης με τις κανονιστικές απαιτήσεις. Οι προγραμματιστές πρέπει να δώσουν προτεραιότητα στη διαφάνεια, στις προσπάθειες οικοδόμησης εμπιστοσύνης και στις ολοκληρωμένες λύσεις που ενσωματώνουν όλες τις πτυχές της οικονομικής διαχείρισης για να ανταποκριθούν στις εξελισσόμενες ανάγκες και προσδοκίες των χρηστών στον χώρο των προσωπικών οικονομικών.

# 2 App Development

Οι εφαρμογές έχουν διάφορες μορφές προσαρμοσμένες σε διαφορετικές ανάγκες και πλατφόρμες. Η κατανόηση αυτών των μορφών - native, web και hybrid- παρέχει τα θεμέλια για την πλοήγηση στο διαρκώς διευρυνόμενο πεδίο της ανάπτυξης εφαρμογών. Επιπλέον, μπορούμε να τις κατηγοριοποιήσουμε σύμφωνα με το περιεχόμενο και τους σκοπούς που εξυπηρετούν.

## 2.1 Τύποι εφαρμογών βάσει τεχνολογιών ανάπτυξης λογισμικού

- **Native εφαρμογές**
  Μία κατηγορία εφαρμογών είναι οι εγγενείς εφαρμογές (native apps). Αυτό το είδος των εφαρμογών εξυπηρετούν συγκεκριμένα λειτουργικά συστήματα ή πλατφόρμες (Android, iOS, Windows Phone) και χρησιμοποιούν συγκεκριμένη γλώσσα προγραμματισμού σύμφωνα με το είδος τους. Οι εφαρμογές Android κατασκευάζονται χρησιμοποιώντας Java, Kotlin και Flutter, ενώ για το frontend χρησιμοποιείται η γλώσσα δέσμης ενεργειών XML. Αντίστοιχα, για τις εφαρμογές IOS οι προγραμματιστές γράφουν σε Swift, Flutter/Dart και C#. Τέλος, για εφαρμογές Windows χρησιμοποιείται κυρίως C#, Visual Basic .NET, C++ και πιο σπάνια HTML, CSS και JavaScript που είναι γλώσσες προγραμματισμού ιστού(Web) μαζί με XAML για τη δημιουργία διεπαφών(UI). Στη συγκεκριμένη εργασία θα ασχοληθούμε με το Android λειτουργικό και την Kotlin ως γλώσσα προγραμματισμού. Οι εγγενείς εφαρμογές κατασκευάζονται χρησιμοποιώντας μια ποικιλία αρχιτεκτονικών μοτίβων και πλαισίων, καθένα από τα οποία έχει σχεδιαστεί για τη βελτιστοποίηση της απόδοσης, της δυνατότητας συντήρησης και της επεκτασιμότητας. Ενώ υπάρχουν αρκετές αρχιτεκτονικές προσεγγίσεις για την ανάπτυξη εγγενών εφαρμογών, ένα από τα πιο συχνά χρησιμοποιούμενα μοτίβα είναι η αρχιτεκτονική Model-View-Controller (MVC). Η αρχιτεκτονική Model-View-Controller (MVC) είναι ένα μοτίβο σχεδιασμού λογισμικού που διαχωρίζει μια εφαρμογή σε τρία κύρια στοιχεία: Model, View και Controller. Αυτός ο διαχωρισμός των ανησυχιών επιτρέπει την καλύτερη οργάνωση του κώδικα και προωθεί τη συναρμολόγηση και την επαναχρησιμοποίηση του. Το Model στοιχείο αντιπροσωπεύει τα δεδομένα και τη λογική της εφαρμογής. Ενσωματώνει δομές δεδομένων, λειτουργίες βάσης δεδομένων και κανόνες, παρέχοντας έναν τρόπο αλληλεπίδρασης και χειρισμού δεδομένων χωρίς να ανησυχεί ο χρήστης για το πώς παρουσιάζονται αυτά. Το View στοιχείο αντιπροσωπεύει τη διεπαφή χρήστη (UI) της εφαρμογής. Είναι υπεύθυνο για την παρουσίαση δεδομένων στον χρήστη με οπτικά ελκυστικό και διαδραστικό τρόπο, χρησιμοποιώντας στοιχεία διεπαφής χρήστη όπως είναι τα κουμπιά, οι ετικέτες και τα πεδία κειμένου για την απόδοση του περιεχομένου της εφαρμογής. Το Controller στοιχείο λειτουργεί ως ενδιάμεσος κρίκος μεταξύ των στοιχείων Model και View. Λαμβάνει πληροφορίες από το χρήστη από την προβολή, αλληλεπιδρά με το Model για την εκτέλεση λογικής και λειτουργιών δεδομένων και ενημερώνει το View, ώστε να α-

ντικατοπτρίζει τυχόν αλλαγές που έγιναν στα υποκείμενα δεδομένα. Ο Controller ενορχηστρώνει τη ροή δεδομένων και συμβάντων εντός της εφαρμογής, διασφαλίζοντας απρόσκοπτη επικοινωνία μεταξύ του Model και του View. Υπάρχουν κάποια πλεονεκτήματα και μειονεκτήματα.

**Πλεονεκτήματα**

- Απόδοση:
  Οι εγγενείς εφαρμογές αναπτύσσονται ειδικά για μια συγκεκριμένη πλατφόρμα χρησιμοποιώντας γλώσσες προγραμματισμού και API για συγκεκριμένες πλατφόρμες. Αυτό τους επιτρέπει να αξιοποιούν τις πλήρεις δυνατότητες του υλικού της συσκευής, με αποτέλεσμα ανώτερη απόδοση και ανταπόκριση σε σύγκριση με άλλους τύπους εφαρμογών.

- Εμπειρία χρήστη:
  Οι εγγενείς εφαρμογές προσφέρουν μια απρόσκοπτη και διαισθητική εμπειρία χρήστη, καθώς συμμορφώνονται με τις οδηγίες σχεδίασης και τις συμβάσεις διεπαφής χρήστη της πλατφόρμας για την οποία έχουν κατασκευαστεί. Αυτή η εξοικείωση ενισχύει την ικανοποίηση και την αφοσίωση των χρηστών, οδηγώντας σε υψηλότερα ποσοστά διατήρησης και αυξημένης χρήσης.

- Πρόσβαση στις λειτουργίες της συσκευής:
  Οι εγγενείς εφαρμογές έχουν πρόσβαση σε ένα ευρύ φάσμα λειτουργιών και λειτουργιών της συσκευής, όπως η κάμερα, το GPS, οι ειδοποιήσεις push κ.ά. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν πλούσιες και διαδραστικές εμπειρίες, ώστε να αξιοποιούν πλήρως τις δυνατότητες της συσκευής.

- Λειτουργικότητα εκτός σύνδεσης:
  Οι εγγενείς εφαρμογές μπορούν συχνά να παρέχουν λειτουργίες εκτός σύνδεσης, επιτρέποντας στους χρήστες να έχουν πρόσβαση σε ορισμένες λειτουργίες ή περιεχόμενο ακόμη και όταν δεν είναι συνδεδεμένοι στο διαδίκτυο. Αυτό είναι ιδιαίτερα χρήσιμο για εφαρμογές που απαιτούν συνεχή πρόσβαση σε δεδομένα ή που πρέπει να λειτουργούν σε περιοχές με κακή συνδεσιμότητα δικτύου.

- Βελτιστοποίηση για τα App Store(αντίστοιχα σε κάθε είδος λειτουργικού):
  Οι εγγενείς εφαρμογές μπορούν να βελτιστοποιηθούν για καταστήματα εφαρμογών, καθιστώντας τις πιο ανιχνεύσιμες στους χρήστες. Οι προγραμματιστές μπορούν να αξιοποιήσουν τεχνικές βελτιστοποίησης στα καταστήματα εφαρμογών, όπως βελτιστοποίηση μέσω χρήσεις λέξεων-κλειδιών, περιγραφές εφαρμογών και αξιολογήσεις, για να βελτιώσουν την προβολή της εφαρμογής τους και να προσελκύσουν περισσότερες λήψεις.

**Μειονεκτήματα**

- Χρόνος και κόστος ανάπτυξης:
  Η δημιουργία εγγενών εφαρμογών για πολλές πλατφόρμες μπορεί να είναι χρονοβόρα

και δαπανηρή, καθώς οι προγραμματιστές πρέπει να γράψουν ξεχωριστές βάσεις κώδικα για κάθε πλατφόρμα. Αυτό αυξάνει τον χρόνο και το κόστος ανάπτυξης, ειδικά για μικρές επιχειρήσεις ή ανεξάρτητους προγραμματιστές με περιορισμένους πόρους.

- Κατακερματισμός πλατφόρμας:
  Το οικοσύστημα των κινητών είναι κατακερματισμένο, με πολλές πλατφόρμες (όπως το iOS και το Android) και πολλά μοντέλα συσκευών και μεγέθη οθονών. Η ανάπτυξη και η διατήρηση εγγενών εφαρμογών για όλες αυτές τις πλατφόρμες και συσκευές μπορεί να είναι προκλητική και απαιτούν πόρους.

- Ενημερώσεις και συντήρηση:
  Οι εγγενείς εφαρμογές απαιτούν τακτικές ενημερώσεις και συντήρηση για να διασφαλιστεί η συμβατότητα με τις πιο πρόσφατες εκδόσεις λειτουργικού συστήματος και μοντέλων συσκευών. Αυτό μπορεί να συνεπάγεται με τεράστια προσπάθεια από την πλευρά των προγραμματιστών, ειδικά για εφαρμογές με μεγάλες βάσεις χρηστών ή πολύπλοκη λειτουργικότητα.

- Διαδικασία έγκρισης:
  Οι εγγενείς εφαρμογές πρέπει να περάσουν από μια διαδικασία έγκρισης για να μπορέσουν να δημοσιευτούν στα καταστήματα εφαρμογών. Αυτή η διαδικασία μπορεί να είναι χρονοβόρα και απρόβλεπτη, καθώς τα καταστήματα εφαρμογών έχουν αυστηρές οδηγίες και κριτήρια που πρέπει να πληρούν οι εφαρμογές για να εγκριθούν.

- Περιορισμένη προσέγγιση χρηστών:
  Οι εγγενείς εφαρμογές συνδέονται με συγκεκριμένες πλατφόρμες, που σημαίνει ότι μπορούν να έχουν πρόσβαση και να εγκατασταθούν μόνο από χρήστες σε αυτές τις πλατφόρμες. Αυτό περιορίζει την εμβέλεια της εφαρμογής σε σύγκριση με εφαρμογές ιστού ή υβριδικές εφαρμογές, στις οποίες είναι δυνατή η πρόσβαση από οποιαδήποτε συσκευή με συμβατό πρόγραμμα περιήγησης.

Συνολικά, ενώ οι εγγενείς εφαρμογές προσφέρουν απαράμιλλη απόδοση και εμπειρία χρήστη, έχουν επίσης ορισμένα μειονεκτήματα, ιδιαίτερα όσον αφορά την πολυπλοκότητα και το κόστος ανάπτυξης. Οι προγραμματιστές πρέπει να σταθμίσουν προσεκτικά αυτούς τους παράγοντες όταν αποφασίζουν για την καλύτερη προσέγγιση για την εφαρμογή τους.

- **Web εφαρμογές**
  Εφαρμογές βάσει τεχνολογιών ανάπτυξης λογισμικού είναι οι εφαρμογές Ιστού(Web). Οι εφαρμογές Ιστού είναι εφαρμογές λογισμικού που έχουν πρόσβαση μέσω προγραμμάτων περιήγησης Ιστού μέσω του Διαδικτύου. Σε αντίθεση με τις παραδοσιακές εφαρμογές για επιτραπέζιους υπολογιστές ή για φορητές συσκευές, οι εφαρμογές ιστού δεν χρειάζεται να ληφθούν ή να εγκατασταθούν σε μια συσκευή. Αντίθετα, οι χρήστες μπορούν απλώς να έχουν πρόσβαση σε αυτά εισάγοντας μια διεύθυνση URL στο πρόγραμμα περιήγησής τους στον ιστό, καθιστώντας τα εξαιρετικά προσιτά και ευέλικτα.

Οι εφαρμογές Ιστού κατασκευάζονται χρησιμοποιώντας τυπικές τεχνολογίες Ιστού όπως HTML (Γλώσσα σήμανσης υπερκειμένου), CSS (Cascading Style Sheets) και JavaScript. Το HTML χρησιμοποιείται για τη δομή του περιεχομένου της εφαρμογής, το CSS για το στυλ της εφαρμογής και τον καθορισμό της διάταξης της και η JavaScript για την προσθήκη διαδραστικότητας και λειτουργικότητας στην εφαρμογή. Αυτές οι τεχνολογίες υποστηρίζονται από όλα τα σύγχρονα προγράμματα περιήγησης ιστού, διασφαλίζοντας τη συμβατότητα σε ένα ευρύ φάσμα συσκευών και πλατφορμών.

Οι εφαρμογές Ιστού συνήθως ακολουθούν μια αρχιτεκτονική πελάτη-διακομιστή, όπου ο πελάτης (δηλαδή το πρόγραμμα περιήγησης Ιστού) επικοινωνεί με έναν απομακρυσμένο διακομιστή για την ανάκτηση και εμφάνιση δεδομένων. Αυτό επιτρέπει στις εφαρμογές Ιστού να εκφορτώνουν μεγάλο μέρος των απαιτήσεων επεξεργασίας και αποθήκευσης στον διακομιστή, καθιστώντας τις ελαφριές και επεκτάσιμες. Οι κοινές τεχνολογίες διακομιστή που χρησιμοποιούνται στην ανάπτυξη εφαρμογών ιστού περιλαμβάνουν τα PHP, Node.js, Ruby on Rails και Django. Υπάρχουν κάποια πλεονεκτήματα και μειονεκτήματα.

**Πλεονεκτήματα**

- Συμβατότητα πολλαπλών πλατφορμών:
  Ένα από τα βασικά πλεονεκτήματα των διαδικτυακών εφαρμογών είναι η ικανότητά τους να εκτελούνται σε οποιαδήποτε συσκευή με συμβατό πρόγραμμα περιήγησης στο διαδίκτυο, ανεξάρτητα από το λειτουργικό σύστημα ή το hardware. Αυτό τις καθιστά ιδιαίτερα προσβάσιμες και ευέλικτες, καθώς οι χρήστες μπορούν να έχουν πρόσβαση στην εφαρμογή από επιτραπέζιους υπολογιστές, φορητούς υπολογιστές, tablet και smartphones.

- Ευκολία ανάπτυξης:
  Οι διαδικτυακές εφαρμογές δεν χρειάζεται να διανέμονται ή να εγκαθίστανται σε μεμονωμένες συσκευές, καθώς η πρόσβαση σε αυτές γίνεται μέσω του διαδικτύου. Αυτό απλοποιεί τη διαδικασία ανάπτυξης και μειώνει την ανάγκη για χειροκίνητες ενημερώσεις ή εγκαταστάσεις, καθώς οι ενημερώσεις μπορούν να διανεμηθούν κεντρικά στο διακομιστή και να είναι άμεσα διαθέσιμες σε όλους τους χρήστες.

- Αποδοτικότητα κόστους:
  Η ανάπτυξη εφαρμογών ιστού είναι γενικά πιο αποδοτική από την ανάπτυξη εγγενών εφαρμογών, καθώς οι προγραμματιστές χρειάζεται να γράψουν κώδικα μόνο μία φορά χρησιμοποιώντας τυποποιημένες τεχνολογίες ιστού. Αυτό μειώνει το χρόνο και τους πόρους ανάπτυξης, καθιστώντας τις εφαρμογές ιστού μια ελκυστική επιλογή για επιχειρήσεις με περιορισμένο προϋπολογισμό ή περιορισμένο χρόνο.

- Άμεσες ενημερώσεις:
  Δεδομένου ότι οι διαδικτυακές εφαρμογές φιλοξενούνται σε διακομιστές ιστού, οι ενημερώσεις και οι αλλαγές μπορούν να εφαρμοστούν άμεσα και απρόσκοπτα, χωρίς

να απαιτείται από τους χρήστες να κατεβάζουν ή να εγκαθιστούν οτιδήποτε. Αυτό επιτρέπει στους προγραμματιστές να κυκλοφορούν γρήγορα διορθώσεις σφαλμάτων, νέες λειτουργίες ή ενημερώσεις περιεχομένου, διασφαλίζοντας ότι οι χρήστες έχουν πάντα πρόσβαση στην τελευταία έκδοση της εφαρμογής.

- Προσβασιμότητα:
  Οι διαδικτυακές εφαρμογές είναι κατά κύριο λόγο προσβάσιμες για ένα ευρύ φάσμα χρηστών, συμπεριλαμβανομένων των ατόμων με αναπηρίες, καθώς μπορούν να προσεγγίσουν αυτήν την κατηγορία με τη χρήση προγραμμάτων ανάγνωσης οθόνης και άλλων υποστηρικτικών τεχνολογιών. Αυτό συμβάλλει στη διασφάλιση ότι η εφαρμογή δεν αποκλείει κανέναν και συμμορφώνεται με τα πρότυπα προσβασιμότητας, κάτι που είναι ολοένα και πιο σημαντικό για τις επιχειρήσεις και τους οργανισμούς.

**Μειονεκτήματα**

- Περιορισμένη λειτουργικότητα εκτός σύνδεσης:
  Οι εφαρμογές αυτές βασίζονται σε διακομιστές ιστού για την παροχή περιεχομένου και λειτουργιών. Αν και, ορισμένες εφαρμογές ιστού μπορεί να προσφέρουν περιορισμένη λειτουργικότητα εκτός σύνδεσης μέσω προσωρινής αποθήκευσης ή τοπικής αποθήκευσης, γενικά δεν μπορούν να φτάσουν τις δυνατότητες εκτός σύνδεσης των εγγενών εφαρμογών.

- Απόδοση:
  Οι διαδικτυακές εφαρμογές ενδέχεται να μην έχουν τόσο καλές επιδόσεις όσο οι εγγενείς εφαρμογές, ιδίως σε κινητές συσκευές με περιορισμένη επεξεργαστική ισχύ και μνήμη. Αυτό οφείλεται στο γεγονός ότι οι εφαρμογές ιστού πρέπει να φορτώνονται και να εκτελούνται μέσα σε ένα πρόγραμμα περιήγησης ιστού, το οποίο μπορεί να προκαλέσει επιβάρυνση και καθυστέρηση σε σύγκριση με τις εγγενείς εφαρμογές που εκτελούνται απευθείας στη συσκευή.

- Λιγότερη ενσωμάτωση με τα χαρακτηριστικά της συσκευής:
  Οι εφαρμογές ιστού έχουν περιορισμένη πρόσβαση στα χαρακτηριστικά και τις λειτουργίες της συσκευής σε σύγκριση με τις εγγενείς εφαρμογές, καθώς εκτελούνται εντός των ορίων του sandbox του προγράμματος περιήγησης ιστού. Αυτό σημαίνει ότι οι εφαρμογές ιστού ενδέχεται να μην είναι σε θέση να αξιοποιήσουν πλήρως χαρακτηριστικά της συσκευής, όπως η κάμερα, το GPS ή οι ειδοποιήσεις push, περιορίζοντας τις δυνατότητές τους σε σύγκριση με τις εγγενείς εφαρμογές.

- Ανησυχίες σχετικά με την ασφάλεια:
  Οι εφαρμογές ιστού είναι εγγενώς πιο ευάλωτες σε απειλές ασφαλείας, όπως cross-site scripting (XSS) και cross-site request forgery (CSRF), καθώς βασίζονται στην επικοινωνία πελάτη-εξυπηρετητή μέσω του διαδικτύου. Οι προγραμματιστές πρέπει να εφαρμόζουν ισχυρά μέτρα ασφαλείας, όπως κρυπτογράφηση και έλεγχο ταυτότητας, για την προστασία ευαίσθητων δεδομένων και την αποτροπή μη εξουσιοδοτημένης πρόσβασης.

- Δέσμευση χρηστών:
  Οι διαδικτυακές εφαρμογές ενδέχεται να αντιμετωπίσουν προκλήσεις όσον αφορά τη δέσμευση των χρηστών και τη διατήρηση του ενδιαφέροντός τους, καθώς δεν επωφελούνται από την προβολή και την αναγνωρισιμότητα που παρέχουν τα καταστήματα εφαρμογών. Οι χρήστες ενδέχεται επίσης να είναι λιγότερο πρόθυμοι να χρησιμοποιούν εφαρμογές ιστού λόγω ανησυχιών σχετικά με την απόδοση, την αξιοπιστία και την ασφάλεια σε σύγκριση με τις εγγενείς εφαρμογές.

Συμπερασματικά, οι διαδικτυακές εφαρμογές προσφέρουν μια ευέλικτη και προσιτή προσέγγιση στην ανάπτυξη λογισμικού, επιτρέποντας στους προγραμματιστές να δημιουργούν πλούσιες και διαδραστικές εφαρμογές στις οποίες μπορεί να έχει κανείς πρόσβαση από οποιαδήποτε συσκευή με συμβατό πρόγραμμα περιήγησης στο διαδίκτυο. Αν και, προσφέρουν αρκετά πλεονεκτήματα, όπως η συμβατότητα πολλαπλών πλατφορμών και η ευκολία ανάπτυξης, οι διαδικτυακές εφαρμογές συνοδεύονται επίσης από ορισμένους περιορισμούς και προκλήσεις που οι προγραμματιστές πρέπει να λάβουν υπόψη τους όταν επιλέγουν την καλύτερη προσέγγιση για την εφαρμογή τους.

- **Hybrid εφαρμογές**
  Οι υβριδικές εφαρμογές ονομάζονται επίσης εφαρμογές Cross Platform Applications. Οι υβριδικές εφαρμογές εκτελούνται σε πολλές πλατφόρμες όπως το Android και το IOS. Επίσης, αυτά γίνονται από την ενοποίηση web και εγγενών εφαρμογών. Επειδή οι υβριδικές εφαρμογές χρησιμοποιούν μια ενιαία βάση κώδικα, μπορούν να αναπτυχθούν σε όλες τις συσκευές. Ως επιλογή ανάπτυξης πολλαπλών πλατφορμών, οι προγραμματιστές έχουν μεγαλύτερη ελευθερία κατά το σχεδιασμό των εφαρμογών τους, καθώς δεν χρειάζεται να τηρούν συγκεκριμένες οδηγίες σχεδίασης από την Apple ή την Google. Οι υβριδικές εφαρμογές κατασκευάζονται χρησιμοποιώντας τεχνολογίες ιστού, όπως HTML, CSS και JavaScript, όπως ακριβώς και οι εφαρμογές ιστού. Αυτό επιτρέπει στους προγραμματιστές να χρησιμοποιούν τις υπάρχουσες δεξιότητες και τα εργαλεία ανάπτυξης ιστού για τη δημιουργία εφαρμογών για κινητά. Επιπλέον, τα πλαίσια υβριδικών εφαρμογών, όπως το Apache Cordova, το Ionic και το React Native, παρέχουν βιβλιοθήκες και εργαλεία που επιτρέπουν στους προγραμματιστές να έχουν πρόσβαση σε χαρακτηριστικά και API της συσκευής, όπως η κάμερα, το GPS και το επιταχυνσιόμετρο, χρησιμοποιώντας JavaScript. Οι υβριδικές εφαρμογές ακολουθούν συνήθως μια υβριδική αρχιτεκτονική, συνδυάζοντας τεχνολογίες ιστού για τη διεπαφή χρήστη και εγγενή στοιχεία για την πρόσβαση σε χαρακτηριστικά της συσκευής. Ο πυρήνας της εφαρμογής, συμπεριλαμβανομένου του UI και της επιχειρησιακής λογικής, υλοποιείται χρησιμοποιώντας HTML, CSS και JavaScript, ενώ τα εγγενή συστατικά, όπως τα plugins ή τα modules, χρησιμοποιούνται για την αλληλεπίδραση με λειτουργίες που αφορούν συγκεκριμένες συσκευές. Αυτά τα εγγενή συστατικά παρέχονται συνήθως από πλαίσια υβριδικών εφαρμογών και συσκευάζονται με την εφαρμογή κατά τη διαδικασία κατασκευής. Υπάρχουν κάποια πλεονεκτήματα και μειονεκτήματα.

  **Πλεονεκτήματα**

- Συμβατότητα πολλαπλών πλατφορμών:
  Ένα από τα βασικά πλεονεκτήματα των υβριδικών εφαρμογών είναι η ικανότητά τους να τρέχουν σε πολλαπλές πλατφόρμες με ελάχιστες αλλαγές στον κώδικα. Δεδομένου ότι κατασκευάζονται με τη χρήση τεχνολογιών ιστού, οι υβριδικές εφαρμογές μπορούν να αναπτυχθούν σε iOS, Android και άλλες πλατφόρμες με μια ενιαία βάση κώδικα, μειώνοντας κατ' αυτόν τον τρόπο τον χρόνο και την προσπάθεια ανάπτυξης.

- Ταχεία ανάπτυξη:
  Οι υβριδικές εφαρμογές επιτρέπουν την ταχεία ανάπτυξη και επανάληψη, καθώς οι προγραμματιστές μπορούν να αξιοποιήσουν τις υπάρχουσες δεξιότητες και εργαλεία ανάπτυξης ιστού. Αυτό επιταχύνει τη διαδικασία ανάπτυξης και επιτρέπει την ταχύτερη διάθεση της εφαρμογής στην αγορά.

- Πρόσβαση στα χαρακτηριστικά της συσκευής:
  Οι υβριδικές εφαρμογές έχουν πρόσβαση σε ένα ευρύ φάσμα χαρακτηριστικών συσκευής και API, χάρη στα εγγενή στοιχεία που παρέχονται από τα πλαίσια υβριδικών εφαρμογών. Αυτό επιτρέπει στους προγραμματιστές να δημιουργούν πλούσιες, σε χαρακτηριστικά, εφαρμογές που μπορούν να εκμεταλλευτούν τις λειτουργίες που αφορούν συγκεκριμένες συσκευές, όπως ο γεωγραφικός εντοπισμός(GPS), την κάμερα και τις ειδοποιήσεις push.

- Αποδοτικότητα κόστους:
  Η δημιουργία μιας υβριδικής εφαρμογής μπορεί να είναι πιο αποδοτική από την ανάπτυξη ξεχωριστών εγγενών εφαρμογών για κάθε πλατφόρμα, καθώς απαιτεί λιγότερο χρόνο και πόρους ανάπτυξης. Αυτό καθιστά τις υβριδικές εφαρμογές μια ελκυστική επιλογή για επιχειρήσεις με περιορισμένο προϋπολογισμό ή πόρους.

**Μειονεκτήματα**

- Απόδοση:
  Ειδικά για εργασίες που εκτελούνται στην κάρτα γραφικών ή στην CPU. Δεδομένου ότι βασίζονται σε τεχνολογίες ιστού και εκτελούνται εντός ενός εγγενούς περιέκτη, οι υβριδικές εφαρμογές ενδέχεται να παρουσιάζουν συμφόρηση επιδόσεων και βραδύτερους χρόνους απόκρισης σε σύγκριση με τις πλήρως εγγενείς εφαρμογές.

- Λειτουργικότητα εκτός σύνδεσης:
  Οι υβριδικές εφαρμογές ενδέχεται να έχουν περιορισμένη λειτουργικότητα εκτός σύνδεσης, καθώς βασίζονται σε τεχνολογίες ιστού και συνήθως απαιτούν σύνδεση στο διαδίκτυο για να λειτουργήσουν σωστά. Ενώ ορισμένα πλαίσια υβριδικών εφαρμογών παρέχουν υποστήριξη για προσωρινή αποθήκευση και τοπική αποθήκευση, οι δυνατότητες εκτός σύνδεσης ενδέχεται να μην είναι τόσο ισχυρές όσο εκείνες των native εφαρμογών.

- Περιορισμένη πρόσβαση στις λειτουργίες της συσκευής:
  Παρόλο που οι υβριδικές εφαρμογές έχουν πρόσβαση σε λειτουργίες και API της

συσκευής μέσω εγγενών στοιχείων, ενδέχεται να μην έχουν πρόσβαση σε όλες τις λειτουργίες που είναι διαθέσιμες στις native εφαρμογές. Ορισμένες λειτουργίες που αφορούν συγκεκριμένες συσκευές ενδέχεται να μην υποστηρίζονται πλήρως ή να απαιτούν προσαρμοσμένη ανάπτυξη με χρήση εγγενούς κώδικα.

- Εξάρτηση από πλαίσια τρίτων κατασκευαστών:
  Οι υβριδικές εφαρμογές βασίζονται σε πλαίσια και βιβλιοθήκες τρίτων, όπως το Apache Cordova ή το Ionic, για να παρέχουν πρόσβαση σε λειτουργίες και API των συσκευών. Αυτή η εξάρτηση μπορεί να δημιουργήσει πρόσθετη πολυπλοκότητα και πιθανούς κινδύνους, όπως προβλήματα συμβατότητας ή έλλειψη υποστήριξης για νέα χαρακτηριστικά της πλατφόρμας.

Εν κατακλείδι, οι υβριδικές εφαρμογές προσφέρουν έναν συναρπαστικό συμβιβασμό μεταξύ της ανάπτυξης εφαρμογών στο διαδίκτυο και της ανάπτυξης native εφαρμογών, παρέχοντας συμβατότητα σε πολλαπλές πλατφόρμες, ταχεία ανάπτυξη και πρόσβαση σε χαρακτηριστικά της συσκευής. Παρόλο που μπορεί να μην προσφέρουν το ίδιο επίπεδο απόδοσης ή εμπειρία χρήστη με τις πλήρως εγγενείς εφαρμογές, οι υβριδικές εφαρμογές αποτελούν μια οικονομικότερη και αποδοτικότερη λύση για τις επιχειρήσεις που επιθυμούν να προσεγγίσουν ένα ευρύ κοινό με μια ενιαία βάση κώδικα.

## 2.2 Τύποι εφαρμογών βάσει περιεχομένου

Στη σημερινή ψηφιακή εποχή, οι εφαρμογές για κινητά έχουν γίνει αναπόσπαστο μέρος της καθημερινότητάς μας, προσφέροντας πληθώρα λειτουργιών και υπηρεσιών προσαρμοσμένων στις ποικίλες ανάγκες και προτιμήσεις των χρηστών. Από την ενίσχυση της παραγωγικότητας έως την παροχή ψυχαγωγίας, από τη διευκόλυνση της επικοινωνίας έως την προαγωγή της υγείας και της ευεξίας, οι εφαρμογές για κινητά έχουν φέρει επανάσταση στον τρόπο με τον οποίο εργαζόμαστε, συνδεόμαστε και εμπλεκόμαστε με τον κόσμο γύρω μας. Υπάρχουν διάφορες κατηγορίες εφαρμογών με βάση το περιεχόμενο και τον σκοπό τους. Κάθε κατηγορία αντιπροσωπεύει μια ξεχωριστή θέση μέσα στο τεράστιο οικοσύστημα των εφαρμογών για κινητά τηλέφωνα, καλύπτοντας συγκεκριμένα ενδιαφέροντα, στόχους και δημογραφικά χαρακτηριστικά χρηστών. Εξερευνώντας αυτές τις κατηγορίες, αποκτούμε εικόνα της πολύπλευρης φύσης της ανάπτυξης εφαρμογών για κινητά και των αμέτρητων τρόπων με τους οποίους οι εφαρμογές εμπλουτίζουν τις ψηφιακές μας εμπειρίες. Οι κατηγορίες είναι οι εξής:

- **Εφαρμογές παραγωγικότητας:**
  Οι εφαρμογές παραγωγικότητας έχουν σχεδιαστεί για να βελτιώνουν την αποτελεσματικότητα και την οργάνωση σε διάφορες πτυχές της ζωής, συμπεριλαμβανομένης της εργασίας, της μελέτης και των προσωπικών καθηκόντων. Συχνά περιλαμβάνουν λειτουργίες όπως διαχείριση εργασιών, καταγραφή σημειώσεων, ενσωμάτωση ημερολογίου και επεξεργασία εγγράφων.(πχ. Ωορδ)

- **Εφαρμογές ψυχαγωγίας:**
  Οι εφαρμογές ψυχαγωγίας παρέχουν στους χρήστες περιεχόμενο για αναψυχή και

απόλαυση, όπως βίντεο ροής, μουσική, παιχνίδια και πλατφόρμες κοινωνικής δικτύωσης. Αυτές οι εφαρμογές στοχεύουν στην ψυχαγωγία και τη συμμετοχή των χρηστών, ενσωματώνοντας συχνά διαδραστικά χαρακτηριστικά και εξατομικευμένες συστάσεις(πχ. Instagram).

- **Εφαρμογές επικοινωνίας:**
  Οι εφαρμογές επικοινωνίας διευκολύνουν την αλληλεπίδραση και την ανταλλαγή πληροφοριών μεταξύ ατόμων ή ομάδων, επιτρέποντας την αποστολή μηνυμάτων σε πραγματικό χρόνο, τις φωνητικές κλήσεις, τις βιντεοκλήσεις και την κοινή χρήση μέσων κοινωνικής δικτύωσης. Αυτές οι εφαρμογές διαδραματίζουν κεντρικό ρόλο στη σύγχρονη επικοινωνία(πχ. WhatsApp).

- **Εφαρμογές κοινής ωφέλειας:**
  Οι βοηθητικές εφαρμογές προσφέρουν πρακτικές λειτουργίες που βοηθούν τους χρήστες σε συγκεκριμένες εργασίες ή δραστηριότητες, όπως πλοήγηση, πρόγνωση καιρού, μετάφραση και διαχείριση αρχείων. Παρέχουν πολύτιμα εργαλεία και πόρους για την απλοποίηση της καθημερινής ζωής (πχ.Google Maps).

- **Εφαρμογές υγείας και γυμναστικής:**
  Οι εφαρμογές υγείας και γυμναστικής επικεντρώνονται στην προώθηση της σωματικής και ψυχικής ευεξίας μέσω λειτουργιών όπως η παρακολούθηση της άσκησης, ο σχεδιασμός διατροφής, η καθοδήγηση διαλογισμού και η παρακολούθηση του ύπνου. Στόχος τους είναι να ενθαρρύνουν τους χρήστες να υιοθετήσουν υγιεινές συνήθειες και επιλογές τρόπου ζωής(πχ. MyFitnessPal).

- **Εκπαιδευτικές εφαρμογές:**
  Οι εκπαιδευτικές εφαρμογές υποστηρίζουν τη μάθηση και την ανάπτυξη δεξιοτήτων σε διάφορα μαθήματα και κλάδους, απευθυνόμενες σε μαθητές όλων των ηλικιών και επιπέδων. Προσφέρουν διαδραστικά μαθήματα, κουίζ, φροντιστήρια και εκπαιδευτικούς πόρους για να διευκολύνουν την αυτόνομη μάθηση(πχ. Duolingo).

- **Εφαρμογές ταξιδιού και πλοήγησης:**
  Οι εφαρμογές ταξιδιού και πλοήγησης βοηθούν τους χρήστες στον προγραμματισμό και την πλοήγηση των ταξιδιών, παρέχοντας πληροφορίες για πτήσεις, καταλύματα, αξιοθέατα και επιλογές μεταφοράς. Προσφέρουν λειτουργίες όπως χάρτες, σχεδιασμό δρομολογίων και τοπικές συστάσεις (πχ. Airbnb).

- **Εφαρμογές αγορών και ηλεκτρονικού εμπορίου:**
  Οι εφαρμογές αγορών και ηλεκτρονικού εμπορίου επιτρέπουν στους χρήστες να περιηγούνται, να αγοράζουν και να διαχειρίζονται προϊόντα και υπηρεσίες στο διαδίκτυο. Προσφέρουν μια βολική και εξατομικευμένη εμπειρία αγορών, με χαρακτηριστικά όπως αναζήτηση προϊόντων, κριτικές και ασφαλείς πληρωμές (πχ. Amazon).

- **Εφαρμογές ειδήσεων και πληροφοριών:**
  Οι εφαρμογές ειδήσεων και πληροφοριών παρέχουν έγκαιρες ενημερώσεις και περιεχόμενο σχετικά με τρέχοντα γεγονότα, θέματα ενδιαφέροντος και τάσεις από όλο

τον κόσμο. Παρέχουν πρόσβαση σε άρθρα, βίντεο, podcasts και επιμελημένες ροές ειδήσεων (πχ.The New York Times).

- **Οικονομικές εφαρμογές:**
  Οι οικονομικές εφαρμογές βοηθούν τους χρήστες να διαχειρίζονται τα οικονομικά τους, συμπεριλαμβανομένου του προϋπολογισμού, της παρακολούθησης δαπανών, της παρακολούθησης επενδύσεων και των τραπεζικών συναλλαγών. Παρέχουν γνώσεις και εργαλεία που ενδυναμώνουν τους χρήστες να λαμβάνουν τεκμηριωμένες οικονομικές αποφάσεις: παραδείγματα(πχ. Mint). Σε αυτήν την κατηγορία θα κατατάξουμε τη δική μας εφαρμογή.

  Αυτές οι κατηγορίες περιλαμβάνουν ένα ευρύ φάσμα εφαρμογών, καθεμία από τις οποίες ανταποκρίνεται σε συγκεκριμένες ανάγκες και προτιμήσεις των χρηστών. Οι προγραμματιστές μπορούν να αξιοποιήσουν αυτές τις κατηγορίες για να δημιουργήσουν ποικίλες και εντυπωσιακές εφαρμογές που αφορούν διάφορες πτυχές της ζωής των χρηστών.

# 3 Quick Response (QR) Κωδικοί

Οι κωδικοί QR, συντομογραφία του Quick Response, είναι δισδιάστατοι γραμμωτοί κώδικες που περιέχουν κωδικοποιημένα δεδομένα. Οι κωδικοί QR έχουν σχεδιαστεί για να σαρώνονται γρήγορα και εύκολα από έναν αναγνώστη γραμμωτού κώδικα, την κάμερα smartphone ή άλλες συσκευές απεικόνισης.

Οι κώδικες QR αποτελούνται από μαύρα τετράγωνα τοποθετημένα σε λευκό φόντο, συνήθως σε μοτίβο τετραγωνικού πλέγματος. Αυτά τα τετράγωνα κωδικοποιούν πληροφορίες με τη μορφή δυαδικών δεδομένων, τα οποία μπορούν να αντιπροσωπεύουν διάφορους τύπους περιεχομένου, όπως κείμενο, διευθύνσεις URL, πληροφορίες επικοινωνίας και άλλα.

Οι κωδικοί QR έχουν γίνει αναπόσπαστο μέρος της σύγχρονης ζωής, παρέχοντας έναν βολικό τρόπο πρόσβασης σε ψηφιακό περιεχόμενο, πραγματοποίησης συναλλαγών και ανταλλαγής πληροφοριών σε διάφορα πλαίσια. Η απλότητα, η ευελιξία και η ευκολία χρήσης τους τους καθιστούν ένα πολύτιμο εργαλείο για επιχειρήσεις, οργανισμούς και ιδιώτες.

## 3.1 Ιστορική αναδρομή

Η Denso Wave, θυγατρική εταιρεία της Denso Corporation, διαδραμάτισε καθοριστικό ρόλο στη δημιουργία και ανάπτυξη των κωδικών QR. Η Denso Corporation ιδρύθηκε το 1949 και είναι ένας παγκόσμιος κατασκευαστής εξαρτημάτων αυτοκινήτων με έδρα την Ιαπωνία. Η Denso Wave, που ιδρύθηκε το 1984, ειδικεύεται στα συστήματα αυτόματης καταγραφής δεδομένων, συμπεριλαμβανομένων των σαρωτών γραμμωτού κώδικα, των συστημάτων RFID και, κυρίως, των κωδικών Quick Response.

Ο Masahiro Hara, ένας καταξιωμένος μηχανικός και καθηγητής, συνέβαλε καθοριστικά στην ανάπτυξη των κωδικών QR. Ο Hara εντάχθηκε στη Denso Wave στις αρχές της δεκαετίας του 1990, όπου ηγήθηκε μιας ομάδας που είχε ως αποστολή τη βελτίωση της αποτελεσματικότητας ως προς τη διαχείριση αποθεμάτων και των διαδικασιών logistics στην αυτοκινητοβιομηχανία. Η τεχνογνωσία και το όραμα του Hara ήταν καθοριστικής σημασίας για τη σύλληψη της έννοιας των κωδικών γρήγορης απόκρισης. Αξιοποιώντας το υπόβαθρό του στην ηλεκτρολογία και την επιστήμη των υπολογιστών, καθώς και την ακαδημαϊκή του έρευνα στη θεωρία της πληροφορίας και τη θεωρία της κωδικοποίησης, ο Hara ήταν πρωτοπόρος στο σχεδιασμό και την υλοποίηση των κωδικών QR.

Υπό την ηγεσία του Hara, η ομάδα μηχανικών της Denso Wave ξεκίνησε ένα ταξίδι για τη δημιουργία ενός επαναστατικού συστήματος κωδικοποίησης δεδομένων, το οποίο είναι ικανό να αποθηκεύει εκτεταμένο όγκο πληροφοριών σε μια συμπαγή, εύκολα ανιχνεύσιμη μορφή. Ο πρωταρχικός σκοπός των κωδικών Quick Response ήταν να αντιμετωπιστούν οι περιορισμοί των παραδοσιακών γραμμωτών κωδικών, οι οποίοι μπορούσαν να αποθηκεύσουν μόνο περιορισμένο αριθμό δεδομένων. Η Denso Wave χρειαζόταν ένα πιο προηγμένο σύστημα που θα μπορούσε να κωδικοποιήσει μεγαλύτερο όγκο πληροφοριών, όπως λεπτομέρειες προϊόντων, δεδομένα κατασκευής και πληροφορίες παρακολούθησης, σε μια συμπαγή και εύκολα ανιχνεύσιμη μορφή. Το αποτέλεσμα των προσπαθειών τους

ήταν η εφεύρεση των κωδικών QR - ένας δισδιάστατος πίνακας από μαύρα τετράγωνα τοποθετημένα σε λευκό φόντο. Η καινοτόμος προσέγγιση της Hara για την κωδικοποίηση δεδομένων, σε συνδυασμό με τη μηχανική επάρκεια της Denso Wave, επέτρεψε την ανάπτυξη των κωδικών γρήγορης απόκρισης ως ευέλικτη λύση για ένα ευρύ φάσμα εφαρμογών.

Ο Masahiro Hara αντιμετώπισε αρκετές προκλήσεις κατά τη διάδοση των κωδικών QR στην αγορά, ιδίως κατά τα πρώτα στάδια της υιοθέτησής τους. Ένα σημαντικό εμπόδιο ήταν η έλλειψη ευαισθητοποίησης και κατανόησης των κωδικών Quick Response μεταξύ των δυνητικών χρηστών, γεγονός που απαιτούσε εκτεταμένες εκπαιδευτικές προσπάθειες για την παρουσίαση των πλεονεκτημάτων και των εφαρμογών τους. Επιπλέον, η περιορισμένη τεχνολογική υιοθέτηση εκείνη την εποχή, με σχετικά λίγες συσκευές εξοπλισμένες για τη σάρωση κωδικών QR, εμπόδιζε την ευρεία χρήση τους. Η υποδομή για τη σάρωση των κωδικών δεν ήταν επίσης τόσο ανεπτυγμένη όσο σήμερα, δημιουργώντας προκλήσεις ενσωμάτωσης. Επιπλέον, οι κώδικες QR θεωρούνταν πολύπλοκοι από ορισμένους χρήστες, γεγονός που ενίσχυσε την αντίσταση στην υιοθέτηση. Τα ζητήματα τυποποίησης και η αντίσταση στην αλλαγή από τα παραδοσιακά συστήματα γραμμωτού κώδικα περιέπλεξαν περαιτέρω τη διαδικασία. Παρά τις προκλήσεις αυτές, ο Hara και η ομάδα του επέμειναν στην προώθηση των κωδικών QR μέσω στοχευμένου μάρκετινγκ, συνεργασιών και συνεχούς καινοτομίας, ξεπερνώντας τελικά τα εμπόδια και ανοίγοντας το δρόμο για την ευρεία υιοθέτησή τους.

Ιστορικά, χάρη στη συμβολή του Masahiro Hara, οι κωδικοί QR εξελίχθηκαν από μια λύση για τη διαχείριση αποθεμάτων στην αυτοκινητοβιομηχανία σε ένα πανταχού παρόν χαρακτηριστικό της σύγχρονης ζωής. Έχουν γίνει απαραίτητα εργαλεία τόσο για τις επιχειρήσεις όσο και για τους καταναλωτές, διευκολύνοντας τα πάντα, από τις εκστρατείες μάρκετινγκ και τις πληρωμές μέσω κινητού τηλεφώνου έως την ανέπαφη ανταλλαγή πληροφοριών και τα συστήματα έκδοσης εισιτηρίων. Η επιμονή και η καινοτομία της Hara έπαιξαν καθοριστικό ρόλο στην αντιμετώπιση των αρχικών προκλήσεων και στην προώθηση της ευρείας υιοθέτησης των κωδικών QR, φέρνοντας τελικά επανάσταση στην κωδικοποίηση δεδομένων και ανοίγοντας το δρόμο για την ενσωμάτωσή τους σε διάφορες βιομηχανίες και εφαρμογές σε παγκόσμιο επίπεδο.

## 3.2 Είδη QR κωδικών

Οι κωδικοί QR υπάρχουν σε διάφορες παραλλαγές, η καθεμία με τα δικά της μοναδικά χαρακτηριστικά και δυνατότητες. Ορισμένες από τις πιο κοινές παραλλαγές είναι οι παρακάτω:

- **Standard QR κωδικοί**
  Οι Standard QR είναι οι πιο ευρέως αναγνωρισμένοι και συχνά χρησιμοποιούμενοι τύποι κωδικού XP. Αποτελούνται από ένα τετράγωνο πλέγμα μαύρων τετραγώνων τοποθετημένων σε λευκό φόντο. Οι τυποποιημένοι κωδικοί QR μπορούν να κωδικοποιήσουν διάφορους τύπους δεδομένων, όπως κείμενο, διευθύνσεις URL, πληροφορίες επικοινωνίας και άλλα.

- **Dynamic QR κωδικοί**
  Οι δυναμικοί κώδικες QR επιτρέπουν την επεξεργασία και την ενημέρωση των κωδικοποιημένων δεδομένων ακόμη και μετά τη δημιουργία και διανομή του κώδικα QR. Αυτή η ευελιξία τους καθιστά ιδανικούς για καταστάσεις όπου οι κωδικοποιημένες πληροφορίες μπορεί να αλλάζουν συχνά, όπως εκστρατείες μάρκετινγκ ή προωθητικές ενέργειες εκδηλώσεων.

- **Static QR κωδικοί**
  Οι στατικοί κωδικοί QR περιέχουν σταθερές πληροφορίες που δεν μπορούν να αλλάξουν μετά τη δημιουργία του κωδικού. Είναι κατάλληλοι για καταστάσεις όπου τα κωδικοποιημένα δεδομένα θα παραμείνουν σταθερά με την πάροδο του χρόνου, όπως η σύνδεση με τον ιστότοπο μιας εταιρείας ή η εμφάνιση πληροφοριών επικοινωνίας.

- **Customized QR κωδικοί**
  Οι προσαρμοσμένοι κωδικοί QR επιτρέπουν την προσαρμογή της εμφάνισής τους, συμπεριλαμβανομένης της προσθήκης λογότυπων, χρωμάτων και εικόνων φόντου. Αυτή η προσαρμογή μπορεί να συμβάλει στην ενίσχυση της ταυτότητας της μάρκας και να κάνει τον κωδικό QR πιο ελκυστικό οπτικά στους χρήστες.

- **High-Capacity QR κωδικοί**
  Οι κώδικες QR υψηλής χωρητικότητας έχουν σχεδιαστεί για να αποθηκεύουν μεγαλύτερες ποσότητες δεδομένων σε σύγκριση με τους τυπικούς κώδικες QR. Το επιτυγχάνουν αυτό με τη χρήση πιο σύνθετων αλγορίθμων κωδικοποίησης και μπορεί να είναι χρήσιμοι για εφαρμογές που απαιτούν την κωδικοποίηση εκτεταμένων πληροφοριών, όπως δεδομένα vCard ή μεγάλες διευθύνσεις URL.

- **Secure QR κωδικοί**
  Οι ασφαλείς κώδικες QR ενσωματώνουν κρυπτογράφηση και άλλα χαρακτηριστικά ασφαλείας για την προστασία των κωδικοποιημένων δεδομένων από μη εξουσιοδοτημένη πρόσβαση ή παραποίηση. Χρησιμοποιούνται συχνά για εφαρμογές όπου η ασφάλεια των δεδομένων αποτελεί πρόβλημα, όπως οι πληρωμές μέσω κινητών τηλεφώνων ή τα συστήματα έκδοσης εισιτηρίων.

- **Multi-QR κωδικοί**
  Οι κώδικες Multi-QR είναι ένας τύπος κώδικα QR που αποτελείται από πολλούς μικρότερους κώδικες QR τοποθετημένους σε μοτίβο πλέγματος. Κάθε μικρότερος κωδικός QR περιέχει ένα τμήμα των συνολικών κωδικοποιημένων δεδομένων, επιτρέποντας την αποτελεσματική αποθήκευση και σάρωση μεγαλύτερων ποσοτήτων πληροφοριών.

## 3.3 Οφέλη

Οι κωδικοί γρήγορης απόκρισης προσφέρουν πληθώρα πλεονεκτημάτων που τους καθιστούν ανεκτίμητα εργαλεία σε διάφορους κλάδους και εφαρμογές.

Ένα από τα κύρια πλεονεκτήματά τους είναι η απαράμιλλη ευκολία τους. Με μια απλή σάρωση χρησιμοποιώντας ένα smartphone ή μια συμβατή συσκευή, οι χρήστες μπορούν άκοπα να έχουν πρόσβαση σε πληροφορίες ή να εκτελούν ενέργειες χωρίς την ανάγκη χειροκίνητης εισαγωγής δεδομένων ή πολύπλοκης πλοήγησης. Αυτή η απρόσκοπτη διαδικασία όχι μόνο βελτιώνει την εμπειρία του χρήστη αλλά και εξοικονομεί ωφέλιμο χρόνο. Επιπλέον, οι κωδικοί QR είναι απίστευτα ευέλικτοι, ικανοί να κωδικοποιήσουν διάφορους τύπους δεδομένων, όπως διευθύνσεις URL, κείμενο, πληροφορίες επικοινωνίας, διαπιστευτήρια Wi-Fi και πολλά άλλα. Αυτή η ευελιξία τους επιτρέπει να εξυπηρετούν ένα ευρύ φάσμα σκοπών, από τη διευκόλυνση εκστρατειών μάρκετινγκ και ψηφιακών πληρωμών έως τη δυνατότητα διαχείρισης αποθεμάτων και την ανέπαφη ανταλλαγή πληροφοριών. Είτε χρησιμοποιούνται για διαφημιστικούς σκοπούς, είτε για υλικοτεχνικές λειτουργίες, είτε για πρωτοβουλίες δέσμευσης πελατών, οι κωδικοί QR προσφέρουν απαράμιλλη ευελιξία και προσαρμοστικότητα. Ένα άλλο σημαντικό πλεονέκτημα τους είναι η ικανότητά τους να διευκολύνουν την παρακολούθηση και την ανάλυση σε πραγματικό χρόνο. Με την ενσωμάτωση κωδικών QR σε υλικά μάρκετινγκ, προϊόντα ή ψηφιακές πλατφόρμες, οι επιχειρήσεις μπορούν να συλλέγουν πολύτιμες πληροφορίες σχετικά με τη συμπεριφορά των καταναλωτών, τα επίπεδα δέσμευσης και την απόδοση της εκστρατείας. Αυτή η προσέγγιση με βάση τα δεδομένα επιτρέπει στους οργανισμούς να λαμβάνουν τεκμηριωμένες αποφάσεις, να βελτιώνουν τις στρατηγικές και να βελτιστοποιούν τις προσπάθειες μάρκετινγκ για μέγιστο αντίκτυπο. Επιπλέον, οι κωδικοί QR είναι οικονομικά αποδοτικοί και εύκολοι στην εφαρμογή, απαιτώντας ελάχιστους πόρους και υποδομές. Χωρίς να χρειάζονται εξειδικευμένο εξοπλισμό ή λογισμικό, οι επιχειρήσεις μπορούν να δημιουργήσουν και να αναπτύξουν γρήγορα κωδικούς QR για να καλύψουν τις συγκεκριμένες ανάγκες και τους στόχους τους. Αυτή η προσβασιμότητα καθιστά τους κωδικούς QR προσιτούς σε επιχειρήσεις όλων των μεγεθών, από μικρές νεοσύστατες επιχειρήσεις έως μεγάλες εταιρείες, εξισώνοντας τους όρους ανταγωνισμού και εκδημοκρατίζοντας την πρόσβαση σε καινοτόμες τεχνολογίες.

Συνολικά, τα οφέλη από τη χρήση των κωδικών γρήγορης απόκρισης εκτείνονται πολύ πέρα από την απλή ευκολία. Δίνουν τη δυνατότητα στις επιχειρήσεις να ενισχύσουν τη δέσμευση των πελατών, να εξορθολογήσουν τις λειτουργίες και να προωθήσουν την ανάπτυξη σε έναν ολοένα και πιο ψηφιακό κόσμο. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, οι κωδικοί QR παραμένουν ένα ευέλικτο και απαραίτητο εργαλείο για τις επιχειρήσεις που επιδιώκουν να παραμείνουν μπροστά από τις εξελίξεις και να προσφέρουν εξαιρετικές εμπειρίες στους πελάτες τους.

## 3.4 QR codes - Barcodes

Οι γραμμωτοί κώδικες (Barcodes) και οι κώδικες QR είναι και οι δύο τύποι μηχανικώς αναγνώσιμων συμβόλων που χρησιμοποιούνται για την κωδικοποίηση δεδομένων, αλλά διαφέρουν σε αρκετές βασικές πτυχές.

Οι γραμμωτοί κώδικες, η παλαιότερη από τις δύο τεχνολογίες, αποτελούνται από μια σειρά παράλληλων γραμμών διαφορετικού πάχους και διαστήματος. Οι γραμμές αυτές αντιπροσωπεύουν αριθμητικά ή αλφαριθμητικά δεδομένα και συνήθως σαρώνονται με σαρω-

τή λέιζερ. Οι γραμμωτοί κώδικες χρησιμοποιούνται ευρέως για την ταυτοποίηση προϊόντων και τη διαχείριση αποθεμάτων στο λιανικό εμπόριο, την εφοδιαστική και τη μεταποιητική βιομηχανία. Ωστόσο, οι παραδοσιακοί γραμμωτοί κώδικες έχουν περιορισμούς όσον αφορά τη χωρητικότητα και την ευελιξία των δεδομένων, καθώς μπορούν να αποθηκεύσουν μόνο περιορισμένο αριθμό πληροφοριών, συνήθως έως 20-25 χαρακτήρες. Αντίθετα, οι κωδικοί QR είναι δισδιάστατα σύμβολα που μπορούν να αποθηκεύσουν σημαντικά περισσότερα δεδομένα σε σύγκριση με τους γραμμωτούς κώδικες. Οι κωδικοί γρήγορης απόκρισης αποτελούνται από μαύρα τετράγωνα τοποθετημένα σε λευκό φόντο σε ένα τετράγωνο μοτίβο πλέγματος. Μπορούν να κωδικοποιήσουν διάφορους τύπους δεδομένων, όπως κείμενο, διευθύνσεις URL, πληροφορίες επικοινωνίας και άλλα. Οι κωδικοί αυτοί σαρώνονται με τη χρήση μιας συσκευής που διαθέτει κάμερα, όπως ένα smartphone ή μια ειδική εφαρμογή σάρωσης κωδικών QR. Ένα από τα σημαντικά πλεονεκτήματα τους είναι η ικανότητά τους να κωδικοποιούν μεγάλες ποσότητες δεδομένων σε συμπαγή μορφή, καθιστώντας τους ευέλικτους και κατάλληλους για ένα ευρύ φάσμα εφαρμογών πέραν των παραδοσιακών περιπτώσεων χρήσης γραμμωτού κώδικα.

Μια άλλη βασική διαφορά μεταξύ των γραμμωτών κωδικών και των κωδικών QR έγκειται στους μηχανισμούς κωδικοποίησης και αποκωδικοποίησης. Οι γραμμωτοί κώδικες κωδικοποιούν δεδομένα χρησιμοποιώντας μια μονοδιάστατη αναπαράσταση γραμμών και διαστημάτων, ενώ οι κώδικες QR χρησιμοποιούν μια δισδιάστατη μορφή μαύρων τετραγώνων και λευκών διαστημάτων. Αυτή η διαφορά στην κωδικοποίηση επιτρέπει στους κώδικες γρήγορης απόκρισης να αποθηκεύουν περισσότερες πληροφορίες και επιτρέπει να σαρώνονται από οποιονδήποτε προσανατολισμό, σε αντίθεση με τους παραδοσιακούς γραμμωτούς κώδικες, οι οποίοι απαιτούν συγκεκριμένο προσανατολισμό για ακριβή σάρωση. Επιπλέον, οι κωδικοί γρήγορης απόκρισης προσφέρουν πρόσθετα χαρακτηριστικά, όπως διόρθωση σφαλμάτων και πλεονασμό δεδομένων, τα οποία ενισχύουν την αξιοπιστία και την ανθεκτικότητά τους σε δύσκολα περιβάλλοντα σάρωσης. Μπορούν επίσης να ενσωματώσουν επιλογές προσαρμογής, επιτρέποντας την προσθήκη στοιχείων επωνυμίας, λογότυπων και χρωμάτων στον κωδικό, καθιστώντας τους οπτικά ελκυστικούς και κατάλληλους για σκοπούς μάρκετινγκ και διαφήμισης.

Συνοπτικά, ενώ τόσο οι γραμμωτοί κώδικες όσο και οι κώδικες QR εξυπηρετούν τον σκοπό της κωδικοποίησης δεδομένων για μηχανικά αναγνώσιμη σάρωση, οι κώδικες γρήγορης απόκρισης προσφέρουν αρκετά πλεονεκτήματα σε σχέση με τους παραδοσιακούς γραμμωτούς κώδικες, συμπεριλαμβανομένης της υψηλότερης χωρητικότητας δεδομένων, της ευελιξίας και των επιλογών προσαρμογής. Οι κωδικοί QR έχουν γίνει πανταχού παρόντες σε διάφορους κλάδους και εφαρμογές, από τη συσκευασία προϊόντων και τις εκστρατείες μάρκετινγκ έως τις πληρωμές μέσω κινητού τηλεφώνου και την ανέπαφη ανταλλαγή πληροφοριών, αντανακλώντας την ευρεία υιοθέτηση και χρησιμότητά τους στον σύγχρονο ψηφιακό κόσμο.

**Εικόνα 1:** *Παράδειγμα QR κωδικού*


**Εικόνα 2:** *Παράδειγμα Barcode κωδικού*

## 3.5 Χρήσεις

Οι κωδικοί QR έχουν γίνει πανταχού παρόντες στο σημερινό ψηφιακό τοπίο, προσφέροντας μια ευέλικτη και βολική λύση για ένα ευρύ φάσμα εφαρμογών. Από τις εκστρατείες μάρκετινγκ και τις πληρωμές μέσω κινητού τηλεφώνου έως τη διαχείριση αποθεμάτων και την ανέπαφη ανταλλαγή πληροφοριών, οι κωδικοί QR χρησιμεύουν ως ισχυρά εργαλεία που βελτιώνουν την εμπειρία των χρηστών, εκλογικεύουν τις λειτουργίες και προωθούν τη δέσμευση. Ορισμένες από τις πιο συνηθισμένες χρήσεις των κωδικών γρήγορης απόκρισης περιλαμβάνουν:

1. **Μάρκετινγκ και διαφήμιση:** Οι κωδικοί QR χρησιμοποιούνται ευρέως σε εκστρατείες μάρκετινγκ για την παροχή γρήγορης πρόσβασης σε διαφημιστικό περιεχόμενο, όπως ιστότοποι, πληροφορίες για προϊόντα, εκπτώσεις και ειδικές προσφορές. Μπορούν να τυπωθούν σε διαφημίσεις, φυλλάδια, αφίσες και συσκευασίες προϊόντων για να προσελκύσουν πελάτες και να προωθήσουν μετατροπές.

2. **Πληρωμές μέσω κινητών τηλεφώνων:** Σε αυτήν την κατηγορία χρησιμεύουν ως ασφαλής και αποτελεσματική μέθοδος για τη διενέργεια πληρωμών μέσω κινητού τηλεφώνου. Χρησιμοποιούνται σε διάφορα συστήματα πληρωμών και εφαρμογές κινητών πορτοφολιών, επιτρέποντας στους χρήστες να πραγματοποιούν αγορές, να μεταφέρουν χρήματα και να ολοκληρώνουν συναλλαγές σαρώνοντας κωδικούς QR που εμφανίζονται στα ταμεία ή σε ψηφιακές οθόνες.

3. **Ανταλλαγή πληροφοριών χωρίς επαφή:** Οι κωδικοί QR επιτρέπουν την ανέπαφη ανταλλαγή πληροφοριών, όπως επαγγελματικές κάρτες, στοιχεία επικοινωνίας, προφίλ στα μέσα κοινωνικής δικτύωσης και προσκλήσεις για εκδηλώσεις. Με τη σάρωση ενός κωδικού γρήγορης απόκρισης, οι χρήστες μπορούν να ανακτήσουν και να αποθηκεύσουν άμεσα πληροφορίες επαφής στα smartphones τους, εξαλείφοντας την ανάγκη για χειροκίνητη εισαγωγή δεδομένων.

4. **Διαχείριση αποθεμάτων:** Τα QR codes χρησιμοποιούνται σε συστήματα διαχείρισης αποθεμάτων για την παρακολούθηση και διαχείριση προϊόντων, περιουσιακών στοιχείων και εξοπλισμού. Σε κάθε στοιχείο ανατίθεται ένας μοναδικός

κωδικός QR που περιέχει σχετικές πληροφορίες, όπως σειριακούς αριθμούς, προδιαγραφές προϊόντων και δεδομένα θέσης, επιτρέποντας την εύκολη αναγνώριση και παρακολούθηση σε όλη την αλυσίδα εφοδιασμού.

5. **Πιστοποίηση ταυτότητας και ασφάλεια:** Οι κωδικοί QR χρησιμοποιούνται για σκοπούς ελέγχου ταυτότητας και ασφάλειας σε διάφορες εφαρμογές, όπως ο έλεγχος ταυτότητας δύο παραγόντων (2FA), τα συστήματα ασφαλούς σύνδεσης και οι μηχανισμοί ελέγχου πρόσβασης. Με τη σάρωση ενός κωδικού QR, οι χρήστες μπορούν να επαληθεύσουν την ταυτότητά τους ή να αποκτήσουν πρόσβαση σε απαγορευμένες περιοχές με ασφάλεια.

6. **Διαχείριση εισιτηρίων και εκδηλώσεων:** Οι κωδικοί QR χρησιμοποιούνται συνήθως για συστήματα ηλεκτρονικής έκδοσης εισιτηρίων και διαχείρισης εκδηλώσεων. Οι συμμετέχοντες μπορούν να λαμβάνουν ψηφιακά εισιτήρια ή πάσο εκδήλωσης με ενσωματωμένους κωδικούς QR, οι οποίοι σαρώνονται για την είσοδο σε χώρους ή σημεία ελέγχου, απλοποιώντας τη διαδικασία έκδοσης εισιτηρίων και ενισχύοντας την ασφάλεια.

7. **Εκπαίδευση και μάθηση:** Οι κωδικοί γρήγορης απόκρισης χρησιμοποιούνται σε εκπαιδευτικά περιβάλλοντα για την παροχή συμπληρωματικού μαθησιακού υλικού, διαδραστικών κουίζ και περιεχομένου πολυμέσων. Οι δάσκαλοι και οι εκπαιδευτικοί μπορούν να δημιουργήσουν κωδικούς QR που συνδέονται με διαδικτυακούς πόρους, βίντεο ή οδηγούς μελέτης, επιτρέποντας στους μαθητές να έχουν πρόσβαση σε πρόσθετες πληροφορίες από τα smartphones ή τα tablet τους.

8. **Τουρισμός και ταξίδια:** Οι κώδικες QR χρησιμοποιούνται στον τουρισμό και τα ταξίδια για να παρέχουν στους τουρίστες πρόσβαση σε πληροφορίες βάσει τοποθεσίας, ηχητικούς οδηγούς, χάρτες και εικονικές περιηγήσεις. Τα τουριστικά αξιοθέατα, τα μουσεία και οι ιστορικοί χώροι διαθέτουν συχνά κωδικούς QR που οι επισκέπτες μπορούν να σαρώσουν για να βελτιώσουν την εμπειρία τους και να μάθουν περισσότερα για το περιβάλλον τους.

Αυτά είναι μερικά μόνο παραδείγματα από τις πολλές χρήσεις των κωδικών QR σε διάφορους κλάδους και εφαρμογές. Καθώς η τεχνολογία συνεχίζει να εξελίσσεται, οι κωδικοί γρήγορης απόκρισης αναμένεται να διαδραματίζουν ολοένα και πιο αναπόσπαστο ρόλο στη διευκόλυνση της απρόσκοπτης και αποτελεσματικής αλληλεπίδρασης μεταξύ επιχειρήσεων και καταναλωτών στην ψηφιακή εποχή.

# 4 Γλώσσα ανάπτυξης λογισμικού - Kotlin

## 4.1 Γενικά στοιχεία του Android λογισμικού

Ο κύκλος ζωής του Android αναφέρεται στις διάφορες καταστάσεις από τις οποίες περνάει μια εφαρμογή Android και τα συστατικά της, όπως τα Activities και τα Fragments, κατά τη διάρκεια του κύκλου ζωής τους. Η κατανόηση και η διαχείριση του είναι ζωτικής σημασίας για τη διασφάλιση της σωστής συμπεριφοράς και χρήσης των πόρων στις εφαρμογές Android. Ακολουθεί μια επισκόπηση του κύκλου ζωής του Android και των βασικών καταστάσεών του:

1. **Δημιουργήθηκε(Created):** Αυτή είναι η αρχική κατάσταση κατά την οποία το στοιχείο δημιουργείται για πρώτη φορά, αλλά δεν είναι ακόμα ορατό στο χρήστη. Σε αυτή την κατάσταση, καλείται η μέθοδος **onCreate()** του συστατικού, επιτρέποντας την εκτέλεση εργασιών αρχικοποίησης, όπως η ρύθμιση στοιχείων UI και δομών δεδομένων.

2. **Ξεκίνησε(Started):** Αφού ολοκληρωθεί η εκτέλεση της μεθόδου onCreate(), το στοιχείο εισέρχεται στην κατάσταση started. Σε αυτό το σημείο, το στοιχείο είναι ορατό στο χρήστη, αλλά μπορεί να μην να μην έχει γίνει ακόμη αντιληπτό. Η μέθοδος **onStart()** καλείται κατά τη διάρκεια αυτής της κατάστασης, όπου μπορούν να εκτελεστούν πρόσθετες εργασίες ρύθμισης, όπως η εγγραφή δεκτών εκπομπής ή η έναρξη κινούμενων σχεδίων.

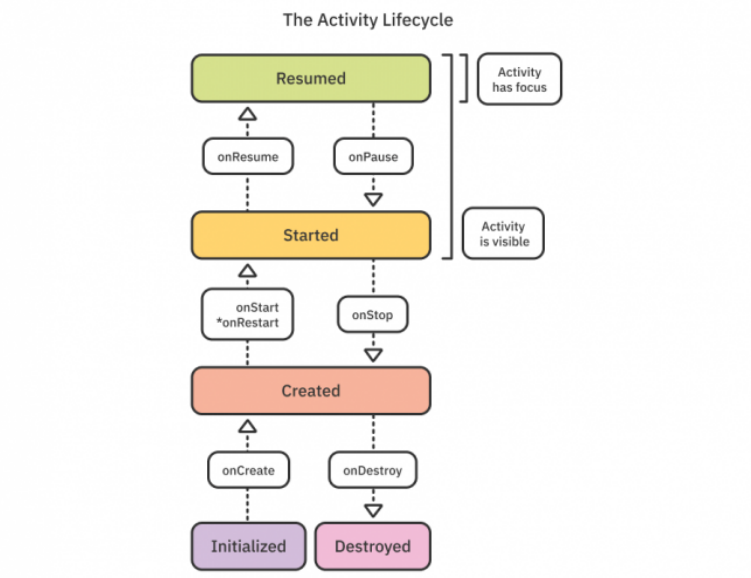
3. **Συνεχίζεται(Resumed)**:Όταν το στοιχείο εμφανίζεται στο προσκήνιο και αποκτά το ενδιαφέρον του χρήστη, εισέρχεται στην κατάσταση resumed. Αυτή είναι η ενεργή κατάσταση του συστατικού, όπου αλληλεπιδρά με τον χρήστη και χειρίζεται την είσοδο του χρήστη. Η μέθοδος **onResume()** καλείται κατά τη διάρκεια αυτής της κατάστασης, όπου μπορούν να εκτελεστούν εργασίες όπως η απόκτηση πόρων συστήματος ή η εκκίνηση νημάτων παρασκηνίου.

4. **Παύση(Paused)**: Εάν μια άλλη δραστηριότητα έρθει στο προσκήνιο ή η τρέχουσα δραστηριότητα χάσει εν μέρει την ορατότητά της, το στοιχείο εισέρχεται στην κατάσταση paused. Η μέθοδος **onPause()** καλείται κατά τη διάρκεια αυτής της κατάστασης, επιτρέποντας στο συστατικό να αποθηκεύσει πληροφορίες κατάστασης ή να αποδεσμεύσει πόρους που δεν χρειάζονται πλέον.

5. **Σταμάτησε(Stopped)**: Όταν το στοιχείο δεν είναι πλέον ορατό στο χρήστη, μεταβαίνει στην κατάσταση Stop. Αυτό μπορεί να συμβεί όταν ο χρήστης πλοηγείται σε άλλη δραστηριότητα ή όταν η δραστηριότητα καταστρέφεται. Η μέθοδος **onStop()** καλείται κατά τη διάρκεια αυτής της κατάστασης, όπου μπορούν να εκτελεστούν εργασίες καθαρισμού, όπως η αποδέσμευση πόρων συστήματος ή η διαγραφή ακροατών.

6. **Καταστράφηκε(Destroyed)**: Η τελική κατάσταση του κύκλου ζωής του Android είναι η κατάσταση καταστροφής, όπου το στοιχείο αφαιρείται από τη μνήμη

και απελευθερώνονται οι πόροι του. Η μέθοδος **onDestroy()** καλείται κατά τη διάρκεια αυτής της κατάστασης, επιτρέποντας στο συστατικό να εκτελέσει τις τελικές εργασίες καθαρισμού πριν από τη συλλογή σκουπιδιών.




**Εικόνα 3:** *Ο κύκλος ζωής των Fragments*

Η διαχείριση του κύκλου ζωής του Android περιλαμβάνει την υπέρβαση αυτών των μεθόδων του κύκλου ζωής στις δραστηριότητες και τα Fragments σας για την εκτέλεση εργασιών αρχικοποίησης, καθαρισμού και αποκατάστασης της κατάστασης στις κατάλληλες χρονικές στιγμές. Επιπλέον, οι προγραμματιστές μπορούν να χρησιμοποιούν στοιχεία και πρότυπα αρχιτεκτονικής με επίγνωση του κύκλου ζωής, όπως το ViewModel και το LiveData, για να χειρίζονται πιο αποτελεσματικά τα συμβάντα του κύκλου ζωής και τη διατήρηση δεδομένων.

Η αποθήκευση δεδομένων UI με ViewModels στο Android είναι μια κοινή πρακτική που χρησιμοποιείται για τη διατήρηση των δεδομένων σε όλες τις αλλαγές διαμόρφωσης και τη διασφάλιση μιας ομαλής εμπειρίας χρήστη. Τα ViewModels αποτελούν μέρος των Android Architecture Components και είναι ειδικά σχεδιασμένα για να κρατούν και να διαχειρίζονται δεδομένα που σχετίζονται με το UI με τρόπο που να γνωρίζει τον κύκλο ζωής. Ακολουθεί ο τρόπος λειτουργίας τους:

- **Επίγνωση του κύκλου ζωής** Τα ViewModels είναι στοιχεία με επίγνωση του κύκλου ζωής, δηλαδή συνδέονται με τον κύκλο ζωής ενός στοιχείου UI, όπως μια δραστηριότητα ή ένα Fragment. Διατηρούνται αυτόματα κατά τη διάρκεια αλλαγών διαμόρφωσης, όπως είναι οι περιστροφές οθόνης, και καταστρέφονται μόνο όταν το σχετικό στοιχείο UI καταστρέφεται πλήρως, όπως είναι για παράδειγμα όταν τελειώνει η δραστηριότητα.

- **Διαχωρισμός των προβλημάτων** Τα ViewModels βοηθούν στο διαχωρισμό της λογικής του UI από τα δεδομένα και την επιχειρησιακή λογική, προωθώντας μια πιο αρθρωτή και συντηρήσιμη αρχιτεκτονική. Εκφορτώνοντας τη διαχείριση των δεδομένων UI στα ViewModels, οι προγραμματιστές μπορούν να διατηρούν τα στοιχεία UI ελαφριά και να επικεντρώνονται στην απόδοση της διεπαφής χρήστη.

- **Διατήρηση δεδομένων** Τα ViewModels αποθηκεύουν τα δεδομένα UI στη μνήμη, γεγονός που τους επιτρέπει να διατηρούν τα δεδομένα σε όλες τις αλλαγές διαμόρφωσης χωρίς την ανάγκη πολύπλοκων μηχανισμών αποθήκευσης δεδομένων. Αυτό διασφαλίζει ότι η κατάσταση του UI παραμένει συνεπής και ανταποκρίνεται, βελτιώνοντας την εμπειρία του χρήστη.

- **Επικοινωνία με στοιχεία UI** Τα ViewModels εκθέτουν LiveData ή άλλα παρατηρήσιμα αντικείμενα δεδομένων που μπορούν να παρατηρήσουν τα στοιχεία του UI, όπως οι δραστηριότητες ή τα fragments. Αυτό επιτρέπει στα στοιχεία UI να αντιδρούν σε αλλαγές στα υποκείμενα δεδομένα και να ενημερώνουν αναλόγως το UI, διευκολύνοντας τον απρόσκοπτο συγχρονισμό και την παρουσίαση δεδομένων.

- **Επεκτείνεται στο στοιχείο UI** Κάθε συστατικό UI (π.χ. δραστηριότητα ή Fragment) έχει τη σχετική του περίπτωση ViewModel, με εμβέλεια στον κύκλο ζωής του. Αυτό διασφαλίζει ότι τα δεδομένα UI είναι scoped στο αντίστοιχο στοιχείο UI, αποτρέποντας τη διαρροή δεδομένων και ελαχιστοποιώντας τη χρήση μνήμης.

Για να χρησιμοποιήσουν ViewModels για την αποθήκευση δεδομένων UI στο Android, οι προγραμματιστές συνήθως δημιουργούν κλάσεις ViewModel που επεκτείνουν την κλάση βάσης ViewModel που παρέχεται από τα συστατικά της αρχιτεκτονικής Android. Στη συνέχεια, εκθέτουν LiveData ή άλλα παρατηρήσιμα αντικείμενα δεδομένων από το ViewModel, τα οποία τα στοιχεία UI μπορούν να παρατηρούν και να αντιδρούν σε αυτά. Αξιοποιώντας τα ViewModels με αυτόν τον τρόπο, οι προγραμματιστές μπορούν να δημιουργήσουν ισχυρές και ευέλικτες εφαρμογές Android που παρέχουν συνεπή εμπειρία χρήστη σε διαφορετικές διαμορφώσεις συσκευών.

Στην ανάπτυξη Android, υπάρχουν διάφορες επιλογές για την αποθήκευση δεδομένων, η καθεμία από τις οποίες είναι κατάλληλη για διαφορετικές περιπτώσεις χρήσης, απαιτήσεις και τύπους δεδομένων. Ακολουθεί μια επισκόπηση των πρωταρχικών επιλογών αποθήκευσης δεδομένων στο Android:

1. **Shared Preferences:** Τα Shared Preferences είναι ένας απλός μηχανισμός αποθήκευσης ζεύγους κλειδιών-τιμών που παρέχεται από το πλαίσιο Android. Χρησιμοποιείται συνήθως για την αποθήκευση μικρών ποσοτήτων δεδομένα βασικών τύπων (primitives), όπως οι προτιμήσεις του χρήστη, οι ρυθμίσεις ή η κατάσταση της εφαρμογής. Οι κοινόχρηστες προτιμήσεις είναι ιδιωτικές για την εφαρμογή και διατηρούνται σε όλες τις εκκινήσεις της εφαρμογής.

2. **Internal Storage:** Το Internal Storage αναφέρεται στον ιδιωτικό αποθηκευτικό χώρο που είναι διαθέσιμος σε κάθε εφαρμογή Android. Χρησιμοποιείται κυρίως για

την αποθήκευση ιδιωτικών αρχείων δεδομένων που δεν πρέπει να είναι προσβάσιμα σε άλλες εφαρμογές ή χρήστες. Είναι ιδανικό για την αποθήκευση ευαίσθητων πληροφοριών ή αρχείων που αφορούν ειδικά την εφαρμογή, όπως βάσεις δεδομένων, αρχεία προσωρινής αποθήκευσης ή αρχεία ρυθμίσεων.

3. **External Storage:** Το External Storage αναφέρεται στον κοινόχρηστο αποθηκευτικό χώρο που είναι προσβάσιμος σε όλες τις εφαρμογές και τους χρήστες της συσκευής. Αυτό περιλαμβάνει τον εξωτερικό αποθηκευτικό χώρο της συσκευής (π.χ. κάρτα SD) και οποιεσδήποτε άλλες κοινές θέσεις αποθήκευσης. Είναι κατάλληλος για την αποθήκευση μεγάλων αρχείων, όπως αρχεία πολυμέσων, λήψεις ή περιεχόμενο που δημιουργείται από χρήστες. Ωστόσο, η πρόσβαση στον εξωτερικό αποθηκευτικό χώρο απαιτεί άδεια από τον χρήστη και ενδέχεται να περιορίζεται από τον διαθέσιμο χώρο στο δίσκο.

4. **Βάσεις Δεδομένων SQLite:** SQLite είναι ένα ελαφρύ σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που είναι ενσωματωμένο στην πλατφόρμα Android. Χρησιμοποιείται συνήθως για την αποθήκευση δομημένων δεδομένων με τη μορφή πινάκων με γραμμές και στήλες. Οι βάσεις δεδομένων SQLite είναι ιδανικές για τη διαχείριση δομημένων δεδομένων, όπως προφίλ χρηστών, κατάλογοι προϊόντων ή αρχεία καταγραφής εφαρμογών. Το Android παρέχει εγγενή υποστήριξη για τις βάσεις δεδομένων SQLite μέσω των κλάσεων SQLiteDatabase και SQLiteOpenHelper.

5. **Room Persistence Library:** Η Room είναι μια ια βιβλιοθήκη διαχείρισης και διατήρησης δεδομένων που παρέχεται από τα στοιχεία αρχιτεκτονικής Android Architecture Components και απλοποιεί την πρόσβαση και τη διαχείριση βάσεων δεδομένων σε εφαρμογές Android. Η Room βασίζεται πάνω στην SQLite και παρέχει αφαιρέσεις υψηλότερου επιπέδου για την εργασία με βάσεις δεδομένων, όπως οντότητες, DAOs (Data Access Objects) και μετακινήσεις δεδομένων μεταξύ βάσεων δεδομένων (data migration σε data bases. Το Room συμβάλλει στον εξορθολογισμό των λειτουργιών της βάσης δεδομένων και προωθεί μια πιο δομημένη και συντηρίσιμη προσέγγιση στην αποθήκευση δεδομένων.

6. **Network Storage:** Οι εφαρμογές Android μπορούν επίσης να αποθηκεύουν δεδομένα σε απομακρυσμένους διακομιστές ή σε υπηρεσίες αποθήκευσης cloud χρησιμοποιώντας πρωτόκολλα επικοινωνίας δικτύου, όπως HTTP ή HTTPS. Αυτό επιτρέπει στις εφαρμογές να εκφορτώνουν την αποθήκευση και την επεξεργασία δεδομένων σε απομακρυσμένους διακομιστές, επιτρέποντας λειτουργίες όπως αντίγραφα ασφαλείας στο σύννεφο, συγχρονισμό δεδομένων και απομακρυσμένη πρόσβαση σε δεδομένα.

Συνολικά, η επιλογή του μηχανισμού αποθήκευσης δεδομένων στο Android εξαρτάται από διάφορους παράγοντες, συμπεριλαμβανομένου του τύπου των δεδομένων που αποθηκεύονται, των απαιτήσεων απορρήτου και ασφάλειας, των επιδόσεων και των στόχων της

εμπειρίας του χρήστη. Αξιοποιώντας τις κατάλληλες επιλογές αποθήκευσης δεδομένων, οι προγραμματιστές μπορούν να δημιουργήσουν ισχυρές και επεκτάσιμες εφαρμογές Android που ανταποκρίνονται αποτελεσματικά στις ανάγκες των χρηστών τους.

## 4.2 Υπορουτίνες και ασύγχρονος προγραμματισμός

Στην ανάπτυξη του Android, οι υπορουτίνες, που συνήθως αναφέρονται επίσης ως μέθοδοι ή συναρτήσεις, παίζουν καθοριστικό ρόλο στην οργάνωση και τη δόμηση του κώδικα για την επίτευξη αρθρωτότητας, επαναχρησιμοποίησης και συντηρησιμότητας. Οι υπορουτίνες είναι ενθυλακωμένα μπλοκ κώδικα που εκτελούν συγκεκριμένες εργασίες ή λειτουργίες μέσα σε μια εφαρμογή Android. Επιτρέπουν στους προγραμματιστές να αναλύουν την πολύπλοκη λογική σε μικρότερες, πιο διαχειρίσιμες μονάδες, οι οποίες μπορούν στη συνέχεια να κληθούν όπως απαιτείται σε όλη την κωδικοποιημένη βάση.

Οι υπορουτίνες στο Android ορίζονται μέσα σε κλάσεις και μπορούν να είναι είτε μέθοδοι παραδείγματος είτε στατικές μέθοδοι. Οι μέθοδοι περίπτωσης συνδέονται με συγκεκριμένα αντικείμενα ή περιπτώσεις μιας κλάσης και μπορούν να έχουν πρόσβαση σε μεταβλητές περίπτωσης και άλλα μη στατικά μέλη της κλάσης. Οι στατικές μέθοδοι, από την άλλη πλευρά, δεν συνδέονται με κάποια συγκεκριμένη περίπτωση και μπορούν να κληθούν απευθείας στην ίδια την κλάση.

Ο ασύγχρονος προγραμματισμός αποτελεί ακρογωνιαίο λίθο της ανάπτυξης Android και θεωρείται ζωτικής σημασίας για την εξασφάλιση εφαρμογών που ανταποκρίνονται και έχουν υψηλές επιδόσεις. Στο οικοσύστημα του Android, όπου η απόκριση είναι υψίστης σημασίας, χρησιμοποιούνται τεχνικές ασύγχρονου προγραμματισμού για την εκτέλεση εργασιών μεγάλης διάρκειας χωρίς να μπλοκάρεται το κύριο νήμα του UI. Αυτό διασφαλίζει ότι η διεπαφή χρήστη παραμένει ομαλή και ανταποκρίνεται, ακόμη και όταν εκτελούνται στο παρασκήνιο εργασίες όπως αιτήματα δικτύου, ερωτήματα βάσης δεδομένων ή λειτουργίες εισόδου/εξόδου αρχείων.

Παραδοσιακά, ο ασύγχρονος προγραμματισμός στο Android επιτυγχάνεται μέσω μηχανισμών όπως οι ανακλήσεις και τα νήματα. Τα callbacks επιτρέπουν στους προγραμματιστές να ορίζουν κώδικα που εκτελείται μόλις ολοκληρωθεί μια συγκεκριμένη εργασία ή συμβεί ένα συμβάν. Αν και αποτελεσματικός, ο ασύγχρονος προγραμματισμός που βασίζεται σε callback μπορεί να οδηγήσει στην κόλαση των callback, όπου η διαχείριση πολλαπλών ασύγχρονων λειτουργιών και των σχετικών callbacks γίνεται περίπλοκη και δύσκολα συντηρήσιμη.

Τα νήματα παρέχουν μια προσέγγιση χαμηλότερου επιπέδου στον ασύγχρονο προγραμματισμό, επιτρέποντας στους προγραμματιστές να δημιουργούν ξεχωριστά νήματα εκτέλεσης για εργασίες μεγάλης διάρκειας. Εκφορτώνοντας αυτές τις εργασίες σε νήματα παρασκηνίου, οι προγραμματιστές μπορούν να αποτρέψουν το μπλοκάρισμα του νήματος UI και να εξασφαλίσουν μια ευέλικτη εμπειρία χρήστη. Ωστόσο, η χειροκίνητη διαχείριση νημάτων εισάγει πολυπλοκότητες όπως ο συγχρονισμός νημάτων, οι συνθήκες ανταγωνισμού και οι πιθανές διαρροές μνήμης, οι οποίες μπορεί να είναι δύσκολο να αντιμετωπιστούν.

Τα τελευταία χρόνια, η εισαγωγή των coroutines στην Kotlin έφερε επανάσταση στον

ασύγχρονο προγραμματισμό στο Android. Οι coroutines παρέχουν έναν ελαφρύ και αποδοτικό τρόπο για τη συγγραφή ασύγχρονου κώδικα με διαδοχικό και σύγχρονο τρόπο, παρόμοιο με τη συγγραφή κώδικα μπλοκαρίσματος. Με τις coroutines, οι προγραμματιστές μπορούν να χρησιμοποιήσουν οικείες γλωσσικές κατασκευές, όπως συναρτήσεις αναστολής και σύνταξη async/await, για να εκτελούν ασύγχρονες εργασίες συνοπτικά και εκφραστικά. Οι coroutines απομακρύνουν τις πολυπλοκότητες της διαχείρισης νημάτων, καθιστώντας τον ασύγχρονο προγραμματισμό πιο προσιτό και λιγότερο επιρρεπή σε σφάλματα για τους προγραμματιστές.

Ο ασύγχρονος προγραμματισμός στο Android δεν αφορά μόνο την απόκριση, αλλά και τη βελτιστοποίηση της χρήσης των πόρων και τη βελτίωση της απόδοσης της εφαρμογής. Αξιοποιώντας αποτελεσματικά τις ασύγχρονες τεχνικές, οι προγραμματιστές του Android μπορούν να δημιουργήσουν εφαρμογές που δεν ανταποκρίνονται μόνο αλλά και είναι αποδοτικές, εξασφαλίζοντας μια ομαλή και απρόσκοπτη εμπειρία χρήστη σε μια ποικιλία συσκευών και συνθηκών δικτύου. Καθώς το οικοσύστημα Android συνεχίζει να εξελίσσεται, η γνώση του ασύγχρονου προγραμματισμού παραμένει απαραίτητη για τη δημιουργία σύγχρονων και φιλικών προς το χρήστη εφαρμογών Android.

## 4.3 Σύγκριση με Java

Η Java, ένα μεγάλο όνομα στον κόσμο του προγραμματισμού, μπορεί να υπερηφανεύεται για δεκαετίες βιομηχανικής κυριαρχίας και ευρείας υιοθέτησης. Η ωριμότητά της αποδεικνύεται από το εύρωστο οικοσύστημά της, το οποίο περιλαμβάνει ένα τεράστιο φάσμα βιβλιοθηκών, πλαισίων και εργαλείων που καλύπτουν ένα ευρύ φάσμα αναπτυξιακών αναγκών. Ωστόσο, η σύνταξη της Java, επηρεασμένη από τους προκατόχους της C και C++, τείνει να είναι πιο φλύαρη. Αυτή η πολυλογία μπορεί να οδηγήσει σε μεγαλύτερες γραμμές κώδικα, ειδικά όταν πρόκειται για πολύπλοκες δομές δεδομένων. Επιπλέον, η Java στερείται ενσωματωμένων χαρακτηριστικών ασφάλειας null, καθιστώντας την ευάλωτη σε NullPointerExceptions, μια κοινή πηγή σφαλμάτων εκτέλεσης. Ενώ η Java παρέχει υποστήριξη για πολυνηματικότητα και ταυτόχρονη εκτέλεση μέσω μηχανισμών όπως τα νήματα και ο συγχρονισμός, η διαχείριση ταυτόχρονων εργασιών μπορεί να είναι πολύπλοκη και επιρρεπής σε σφάλματα. Αν και η Java υποστηρίζει σε κάποιο βαθμό τον λειτουργικό προγραμματισμό, οι δυνατότητές της σε αυτόν τον τομέα είναι περιορισμένες σε σύγκριση με πιο σύγχρονες γλώσσες όπως η Kotlin.

Αντίθετα, η Kotlin, αν και σχετικά νεότερη, συγκέντρωσε γρήγορα την προσοχή για τη συνοπτικότητα και την εκφραστικότητά της. Η σύνταξη της Kotlin είναι αξιοσημείωτα πιο βελτιωμένη από εκείνη της Java, μειώνοντας τον κώδικα boilerplate και βελτιώνοντας την αναγνωσιμότητα του κώδικα. Αυτή η βελτίωση της αναγνωσιμότητας ενισχύεται περαιτέρω από τα ενσωματωμένα χαρακτηριστικά ασφάλειας null της Kotlin, όπως οι nullable και non-nullable τύποι. Με την ενσωμάτωση της ασφάλειας null στο σχεδιασμό της, η Kotlin μετριάζει τον κίνδυνο NullPointerExceptions, ενισχύοντας έτσι την αξιοπιστία και την ευρωστία του κώδικα. Επιπλέον, η υποστήριξη της Kotlin για λειτουργικό προγραμματισμό είναι ισχυρή, διαθέτοντας χαρακτηριστικά όπως οι εκφράσεις λάμδα, οι συναρτήσεις ανώτερης τάξης και οι αμετάβλητες δομές δεδομένων. Αυτά τα

χαρακτηριστικά δίνουν τη δυνατότητα στους προγραμματιστές να υιοθετήσουν με ευκολία σύγχρονες πρακτικές ανάπτυξης, όπως ο αντιδραστικός προγραμματισμός(reactive programming) και ο functional composition. Η διαλειτουργικότητα της Kotlin με τη Java αυξάνει περαιτέρω την ελκυστικότητά της, επιτρέποντας την απρόσκοπτη ενσωμάτωση με τις υπάρχουσες βάσεις κώδικα της Java, προσφέροντας παράλληλα σύγχρονα χαρακτηριστικά και βελτιώσεις της γλώσσας.

Η σύγκριση της Java και της Kotlin αποκαλύπτει ξεχωριστές πτυχές μεταξύ των δύο γλωσσών. Ενώ η Java επωφελείται από το εκτεταμένο οικοσύστημα και την ευρεία υιοθέτησή της, η Kotlin υπερέχει όσον αφορά τη συνοπτικότητα του συντακτικού, την ασφάλεια null και την υποστήριξη σύγχρονων παραδειγμάτων ανάπτυξης, όπως ο λειτουργικός προγραμματισμός. Και οι δύο γλώσσες είναι πλήρως διαλειτουργικές, επιτρέποντας στους προγραμματιστές να αξιοποιούν τις υπάρχουσες βιβλιοθήκες και τα πλαίσια της Java σε έργα Kotlin. Ωστόσο, τα σύγχρονα χαρακτηριστικά της Kotlin και το εξορθολογισμένο συντακτικό της την καθιστούν ελκυστική επιλογή για πολλούς προγραμματιστές, ιδίως για εκείνους που επιδιώκουν να βελτιώσουν την αναγνωσιμότητα, την αξιοπιστία και την παραγωγικότητα του κώδικα. Καθώς η υιοθέτηση της Kotlin συνεχίζει να αυξάνεται, ειδικά στην κοινότητα ανάπτυξης Android, παρουσιάζει μια συναρπαστική εναλλακτική λύση στη Java για τη δημιουργία αξιόπιστων και συντηρήσιμων λύσεων λογισμικού.

# 5 Υλοποίηση της εφαρμογής Expense Manager

## 5.1 Logo Εφαρμογής και Design/Layouts

Το λογότυπο μιας εφαρμογής θα πρέπει να είναι αντιπροσωπευτικό του περιεχομένου της, ώστε να παρέχει στους χρήστες μια οπτική ένδειξη που μεταφέρει με μια ματιά τον σκοπό, τη λειτουργικότητα και την ταυτότητα της εφαρμογής, συμβάλλοντας στην εδραίωση μιας ισχυρής παρουσίας του εμπορικού σήματος και στην αποτελεσματική προσέλκυση του κοινού.

Το Logo της εφαρμογής είναι εμπνευσμένο από τον σκοπό της, ο οποίος είναι η διαχείριση των οικονομικών στοιχείων του χρήστη από τον ίδιο, είτε είναι αποθηκευμένα μέσα σε μία κάρτα είτε υπάρχουν στο πορτοφόλι του. Τα εικονίδια στο Logo όπως και τα υπόλοιπα που έχουν χρησιμοποιηθεί από το https://fonts.google.com/icons και από το https://icons8.com/icons, καθώς παρέχονται δωρεάν προς χρήση.

**Εικόνα 4:** *Logo του Expense Manager*

Τα Layouts παίζουν καθοριστικό ρόλο στον καθορισμό της οπτικής δομής των διεπαφών χρήστη μέσα στις εφαρμογές. Οι διατάξεις είναι αρχεία XML που καθορίζουν τη διάταξη και την εμφάνιση των στοιχείων του UI, όπως κουμπιά, προβολές κειμένου και εικόνες, στην οθόνη. Η Kotlin παρέχει μια ποικιλία τύπων διάταξης, συμπεριλαμβανομένων των ConstraintLayout, LinearLayout και RelativeLayout, που ο καθένας προσφέρει διαφορετικές προσεγγίσεις για την οργάνωση των στοιχείων UI. Αυτές οι διατάξεις διάταξης επιτρέπουν στους προγραμματιστές να δημιουργούν ευέλικτες διεπαφές χρήστη που προσαρμόζονται σε διάφορα μεγέθη και προσανατολισμούς οθόνης, βελτιώνοντας τη συνολική εμπειρία του χρήστη. Επιπλέον, η εκφραστική σύνταξη της Kotlin και η υποστήριξη για view binding καθιστούν εύκολη την προγραμματιστική αλληλεπίδραση με τα layouts, διευκολύνοντας τις δυναμικές ενημερώσεις του UI και την αλληλεπίδραση του χρήστη.

Στην ανάπτυξη του Kotlin Android, ο κατάλογος "res", η συντομογραφία για τα resources, παίζει ζωτικό ρόλο στην υποστήριξη των διατάξεων. Μέσα σε αυτόν, υπάρχουν υποκατάλογοι αφιερωμένοι σε διαφορετικούς τύπους πόρων, συμπεριλαμβανομένων των layout resources, τα οποία είναι αρχεία XML που βρίσκονται στον κατάλογο "res/layout" και καθορίζουν τη δομή και την εμφάνιση των διεπαφών χρήστη στις εφαρμογές Android. Αυτά τα αρχεία διάταξης περιέχουν ιεραρχικές διατάξεις στοιχείων UI, όπως TextViews, ImageViews, Buttons και άλλα, χρησιμοποιώντας ετικέτες και χαρακτηριστικά XML. Οι προγραμματιστές μπορούν να δημιουργήσουν πολλαπλά αρχεία διάταξης για

να υποστηρίξουν διάφορα μεγέθη οθόνης, προσανατολισμούς και διαμορφώσεις συσκευών, εξασφαλίζοντας μια συνεπή εμπειρία χρήστη σε διαφορετικές συσκευές. Επιπλέον, ο κατάλογος "res" περιέχει και άλλους πόρους, όπως drawables, strings, colours και διαστάσεις, στους οποίους μπορούν να γίνει αναφορά μέσα στα αρχεία διάταξης για να βελτιωθεί η οπτική εμφάνιση και η λειτουργικότητα της διεπαφής χρήστη. Στο Expense Manager έχουν χρησιμοποιηθεί κάποια drawables που αφορούν την εμφάνιση κάποιον χαρακτηριστικών για τα layouts.

Απόσπασμα κώδικα 1: Παράδειγμα Drawable: circle_btn.xml

```
<?xml version="1.0" encoding="utf-8"?>
                        <selector xmlns:android="http://
                            ↪ schemas.android.com/apk/res/
                            ↪ android">
                        <item android:state_pressed="false">
                                <shape android:shape="oval">
                                        <solid android:color="
                                            ↪ @color/create\
                                            ↪ _btn"/>
                                </shape>
                        </item>
                        <item android:state_pressed="true">
                                <shape android:shape="oval">
                                        <solid android:color="
                                            ↪ #c20586"/>
                                </shape>
                        </item>
                </selector>
```

Παρακάτω εμφανίζεται ένα παράδειγμα Layout που αφορα την εμφάνιση του πώς θα εμφανίζεται το κάθε currency για το spinner. Σε αυτό χρησιμοποιείται ένα LinearLayout που έχει τα χαρακτηριστικά του πού βρίσκεται στον χώρο και ένα TextView που είναι το ίδιο το κείμενο του νομίσματος.

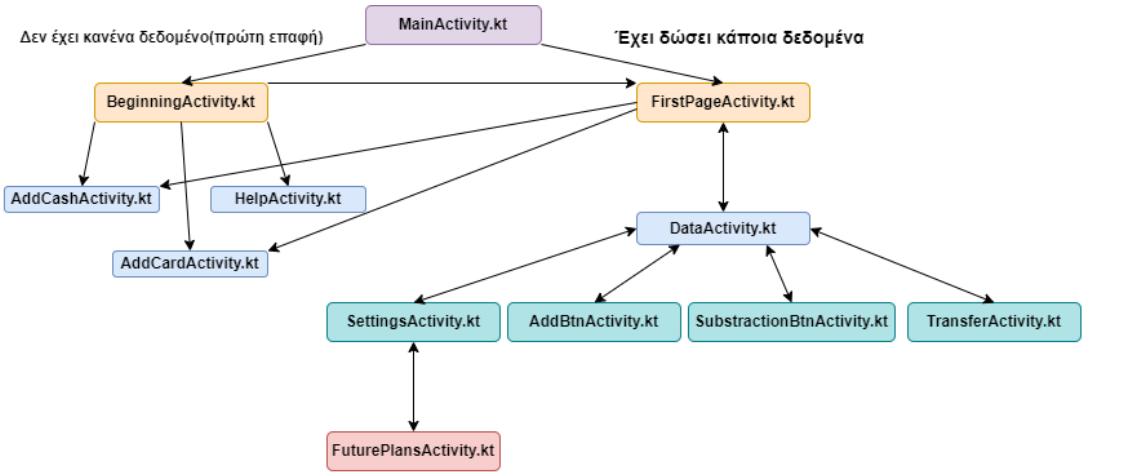
Απόσπασμα κώδικα 2: Παράδειγμα Layout: layout_for_currencies.xml

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/
    ↪ res/android"
     xmlns:tools="http://schemas.android.com/tools"
     android:layout_width="match_parent"
     android:layout_height="wrap_content"
     android:orientation="vertical"
     android:layout_marginLeft="100dp"
     android:padding="10dp"
     android:gravity="center"
     android:layout_marginRight="100dp">
     <TextView
          android:id="@+id/currencies"
          android:layout_width="match_parent"
          android:gravity="center"
          android:layout_height="wrap_content"
          android:textColor="@color/white"
          android:textSize="20sp"/>
</LinearLayout>
```

## 5.2 Activities - Adapters - DataClasses

Στην Kotlin, τα Activities χρησιμεύουν ως θεμελιώδη δομικά στοιχεία των εφαρμογών Android, αντιπροσωπεύοντας μεμονωμένες οθόνες ή παράθυρα μέσω των οποίων οι χρήστες αλληλεπιδρούν με το περιεχόμενο της εφαρμογής. Ορίζονται συνήθως ως κλάσεις που επεκτείνουν την κλάση AppCompatActivity που παρέχεται από την Kotlin. Ενθυλακώνουν τα στοιχεία UI, όπως Layouts και στοιχεία διεπαφής χρήστη, καθώς και τη λογική για το χειρισμό των αλληλεπιδράσεων των χρηστών και τη διαχείριση του κύκλου ζωής της εφαρμογής. Οι δραστηριότητες παίζουν καθοριστικό ρόλο στην πλοήγηση μεταξύ διαφορετικών οθονών εντός της εφαρμογής, διευκολύνοντας τις μεταβάσεις και διατηρώντας την κατάσταση της διεπαφής χρήστη σε διάφορα συμβάντα του κύκλου ζωής, όπως onCreate(), onStart() κτλ. . Στην παρακάτω φωτογραφία φαίνεται η ροή των Activities της εφαρμογής.

Τα Activities αφορούν τόσο τη μεταφορά σε άλλο Activity όσο και την λειτουργικότητα των διαφόρων στοιχείων. Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η λειτουργικότητα του AddBtnActivity.kt στο οποίο ο χρήστης έχει να επιλέξει ανάμεσα σε δύο κουμπιά τον τρόπο που επιθυμεί να προσθέσει ένα χρηματικό ποσό. Αρχικά, με την onCreate διασφαλίζεται ότι κάθε απαραίτητη αρχικοποίηση που ορίζεται από την υπερκλάση εκτελείται πριν από οτιδήποτε άλλο. Η setContentView ορίζει τη διάταξη για τη δραστηριότητα χρησιμοποιώντας τη μέθοδο setContentView(). Φορτώνει το αρχείο πόρων διάταξης με όνομα activity_add_btn.xml και το ορίζει ως προβολή περιεχομένου για το UI της δραστηριότητας.




- **MainActivity.kt:** Είναι το πρώτο Activity όταν ανοίγει την εφαρμογή ο χρήστης
- **BegginngActivity.kt**: Δίνει τις πρώτες δυνατότητες στον χρήστη για την εισαγωγή των πρώτων δεδομένων.
- **AddCashActivity.kt:** Προσθήκη "πορτοφολιού".
- **AddCardActivity.kt:** Προσθήκη κάρτας/τράπεζας.
- **HelpActivity.kt:** Δίνει πληροφορίες στον χρήστη για την πρώτη επαφή με την εφαρμογή.
- **FirstPageActivity.kt:** Οι πληροφορίες που έχει δημιουργήσει ο χρήστης.
- **DataActivity.kt:** Συγκεκριμένο δεδομένο που έχει επιλέξει ο χρήστης να δει/επεξεργαστεί.
- **SettingsActivity.kt:** Ρυθμίσεις.
- **FuturePlansActivtiy.kt:** Πληροφορίες για μελλοντικές υλοποιήσεις.
- **AddBtnActivity.kt:** Προσθήκη χρηματικού ποσού.
- **SubstractionActivity.kt:** Αφαίρεση χρηματικού ποσού.
- **TransferActivity.kt:** Μεταφορά χρηματικού ποσού.

**: Μπορείς να επιστρέψεις στο προηγούμενο Activity**

**: Δεν μπορείς να επιστρέψεις στο προηγούμενο Activity**

**Εικόνα 5:** *Ροή των Activities*

Με άλλα λόγια, καθορίζει τη διάταξη που θα εμφανίζεται στην οθόνη όταν δημιουργείται η δραστηριότητα. Στη συνέχεια, αφού γίνουν οι απαραίτητες αρχικοποιήσεις μεταβλητών με τα αντίστοιχα αντικείμενα των Layouts, έχουμε τη λειτουργικότητα των δύο δυνατοτήτων που έχει ο χρήστης σε αυτό το Activity. Η ανάλυση της υλοποίησης του QR θα γίνει σε παρακάτω υποκεφάλαιο. Στο τέλος, αναπτύσσεται ο κώδικας που αφορά την προσθήκη του χρηματικού ποσού στο εκάστοτε δεδομένο που έχει επιλέξει ο χρήστης. Ομοίως δουλεύει και το SubstractionBtnActivity που αφορά την αφαίρεση του χρηματικού ποσού από ένα συγκεκριμένο αντικείμενο.

Απόσπασμα κώδικα 3: AddBtnActivity.kt

```
override fun onCreate(savedInstanceState: Bundle?) {
    super.onCreate(savedInstanceState)
    setContentView(R.layout.activity_add_btn)
    qrbtn.setOnClickListener {...}
    val dialog = Dialog(this)
    addbtn.setOnClickListener {
        dialog.setContentView(R.layout.
            ↪ layout_for_alert_dialog_add_manual_amount)
        dialog.window?.setBackgroundDrawable(ColorDrawable(
            ↪ Color.TRANSPARENT))
        val amountofdata = dialog.findViewById<TextView>(R.id.
            ↪ availableamount)
        val currencydata = dialog.findViewById<TextView>(R.id.
            ↪ cur)
        amountofdata.text = getString(R.string.availableamount
            ↪ , amount.toString())
        currencydata.text = getString(R.string.currency,
            ↪ currency)
        val okaybtn = dialog.findViewById<Button>(R.id.add)
        okaybtn.setOnClickListener {
            val enteranamount = dialog.findViewById<EditText>(
                ↪ R.id.enteranamount)
            val enteredamount = enteranamount.text.toString()
                ↪ // the amount which user entered
            amountofdata.text = getString(R.string.
                ↪ availableamount, amount.toString())
            currencydata.text = getString(R.string.currency,
                ↪ currencydata)
            val res = amount + enteredamount.toFloat()
            val editor = sp.edit()
            if (feature == "w")
                editor.putFloat("amount_in_wallet_$keyString",
                    ↪  res)
```

```
                else
                    editor.putFloat("amount_in_bank_$keyString",
                        ↪ res)
                editor.apply()
                Toast.makeText(this@AddBtnActivity, "result␣=␣$res
                    ↪ ", Toast.LENGTH_LONG).show()
                dialog.dismiss()
                val intent = Intent(this@AddBtnActivity,
                    ↪ FirstPageActivity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK or
                    ↪ Intent.FLAG_ACTIVITY_CLEAR_TASK
                startActivity(intent)
            }
                ...
    }
}
```

Οι Adapters χρησιμεύουν ως στοιχεία για τη σύνδεση πηγών δεδομένων με τα στοιχεία διεπαφής χρήστη, όπως τα RecyclerViews, ListViews και Spinners. Ενεργούν ως ενδιάμεσοι και απλοποιούν τη διαδικασία συμπλήρωσης στοιχείων UI με δεδομένα από συλλογές, όπως λίστες ή πίνακες. Παρακάτω αυτός ο Adapter απλοποιεί τη διαδικασία συμπλήρωσης του Spinner με αντικείμενα DestinationItem, διευκολύνοντας έτσι την εμφάνιση προσαρμοσμένων δεδομένων στο αναπτυσσόμενο μενού του Spinner. Περιέχει μεθόδους που αφορούν την προβολή και τη δέσμευση δεδομένων για προσαρμοσμένη διάταξη

Απόσπασμα κώδικα 4: DestinationAdapter.kt

```
class DestinationAdapter(context: Context, private val items:
    ↪ List<DestinationItem>) : ArrayAdapter<DestinationItem>(
    ↪ context, 0, items) {
    override fun getView(position: Int, convertView: View?,
        ↪ parent: ViewGroup): View {
        return getCustomView(position, convertView, parent)
    }
    override fun getDropDownView(position: Int, convertView:
        ↪ View?, parent: ViewGroup): View {
        return getCustomView(position, convertView, parent)
    }
    private fun getCustomView(position: Int, convertView: View
        ↪ ?, parent: ViewGroup): View {
        var itemView = convertView
        if (itemView == null) {
            itemView = LayoutInflater.from(context).inflate(R.
```

```
                ↪ layout.spinner_bankwallet_data, parent,
                ↪ false)
        }
        val currentItem = items[position]
        val nameTextView = itemView!!.findViewById<TextView>(R
            ↪ .id.name)
        val amountTextView = itemView.findViewById<TextView>(R
            ↪ .id.amount)
        val currencyTextView = itemView.findViewById<TextView
            ↪ >(R.id.currency)
        nameTextView.text = currentItem.name
        amountTextView.text = currentItem.amount
        currencyTextView.text = currentItem.currency
        return itemView
    }
}
```

Ένα DataClass στην Kotlin είναι ένας ειδικός τύπος κλάσης που έχει σχεδιαστεί για την αποθήκευση δεδομένων. Δημιουργεί αυτόματα χρήσιμες μεθόδους όπως οι toString(), equals() και hashCode() με βάση τις ιδιότητες που ορίζονται στον κύριο κατασκευαστή. Χρησιμοποιούνται συνήθως για τη μοντελοποίηση immutable data structures και διευκολύνουν την ύπαρξη ενός καθαρού και συνοπτικού κώδικα για την αναπαράσταση αντικειμένων. Στην συγκεκριμένη περίπτωση έχουμε ένα DataClass που αφορά την αποθήκευση των χαρακτηριστών μίας κάρτας/τράπεζας όπως το όνομα, το χρηματικό ποσό και άλλα.

Απόσπασμα κώδικα 5: Dataclass.kt

```
data class BankInfos(val bank_name: String, val b_amount:
    ↪ Float, val b_currency: String, val key: String, val
    ↪ feature: String)
```

## 5.3 Αποδοτική αποθήκευση δεδομένων - SharedPreferences

Τα SharedPreferences στην Kotlin είναι ένας βολικός και ελαφρύς μηχανισμός για την αποθήκευση και ανάκτηση ζευγών κλειδιών-τιμών. Οι λόγοι που προτιμήθηκαν στην συγκεκριμένη εργασία είναι οι παρακάτω:

1. Είναι κατάλληλα για την αποθήκευση μικρών ποσοτήτων πρωτόγονων τύπων δεδομένων, όπως ακέραιοι, booleans, floats και συμβολοσειρές, καθιστώντας τα ιδανικά για την αποθήκευση προτιμήσεων, ρυθμίσεων και απλής κατάστασης της εφαρμογής.

2. Τα SharedPreferences προσφέρουν ένα απλό API για την ανάγνωση και εγγραφή δεδομένων, απαιτώντας ελάχιστο κώδικα και παρέχοντας έναν απλό τρόπο διαχείρισης των προτιμήσεων της εφαρμογής χωρίς την ανάγκη πολύπλοκων ρυθμίσεων βάσεων δεδομένων.

3. Τα SharedPreferences είναι συγκεκριμένες για την εφαρμογή στην οποία δημιουργούνται. Αυτό σημαίνει ότι τα δεδομένα που αποθηκεύονται στα SharedPreferences ανήκουν αποκλειστικά και μόνο στη συγκεκριμένη εφαρμογή και δεν είναι προσβάσιμα από άλλες εφαρμογές στη συσκευή.

4. Τα SharedPreferences παρέχουν έναν συγκεντρωτικό και απλό τρόπο για μόνιμη αποθήκευση και ανάκτηση ζευγών κλειδιών-τιμών. Αυτή η κεντρική αποθήκευση εξασφαλίζει ότι όλα τα στοιχεία της εφαρμογής μπορούν να έχουν πρόσβαση και να χειρίζονται το ίδιο σύνολο προτιμήσεων, καθιστώντας εύκολη τη διατήρηση της συνέπειας σε ολόκληρη την εφαρμογή.

Συνολικά, τα SharedPreferences προσφέρουν έναν ευέλικτο και αποτελεσματικό μηχανισμό για τη διαχείριση της κατάστασης της εφαρμογής και των προτιμήσεων των χρηστών σε εφαρμογές Android, χάρη στην οριοθετημένη, προσβάσιμη και συγκεντρωτική φύση τους.

Παρακάτω παραθέτονται κώδικες για τον τρόπο χρήσης του συγκεκριμένου μηχανισμού αποθήκευσης.

Απόσπασμα κώδικα 6: Χρήση του SharedPreferences στο AddCardActivity.kt

```
...
val sharedPref = getSharedPreferences("userOptions",
    ↪ MODE_PRIVATE)
val editor = sharedPref.edit()
...
val id = sharedPref.getInt("bankID", 0) + 1
editor.apply {
    putInt("bankID", id)
    putString("name_of_bank_$id", nameOfBank)
    putFloat("amount_in_bank_$id", amountInBank.toFloat())
    putString("currency_bank_$id", selectedCurrency)
}
editor.apply()
...
```

Ξεκινάμε με την αρχικοποίηση μέσω της μεθόδου getSharedPreferences(), η οποία χρησιμοποιείται για να ληφθεί μια περίπτωση SharedPreferences με όνομα "userOptions". Η δεύτερη παράμετρος MODE_PRIVATE καθορίζει ότι το αρχείο SharedPreferences θα πρέπει να είναι προσβάσιμο μόνο από την καλούσα εφαρμογή. Στην συνέχεια, δημιουργούμε μία μεταβλητή editor η οποία λαμβάνεται από το αντικείμενο SharedPreferences χρησιμοποιώντας τη μέθοδο edit(). Αυτή χρησιμοποιείται για την τροποποίηση των δεδομένων του SharedPreferences. Μέσω αυτής ανακτάται το τρέχον αναγνωριστικό τράπεζας από το αντικείμενο SharedPreferences χρησιμοποιώντας την getInt(). Εάν δεν βρεθεί τιμή για το "bankID", η τιμή είναι προεπιλεγμένη σε 0. Το ανακτηθέν ID αυξάνεται στη συνέχεια κατά 1 για να δημιουργηθεί ένα νέο ID για την επόμενη καταχώρηση τράπεζας. Η

μέθοδος apply() καλείται για να ξεκινήσει η επεξεργασία των SharedPreferences. Μέσα στις αγκύλες τα δεδομένα προστίθενται ή ενημερώνονται χρησιμοποιώντας τις μεθόδους putInt(), putString() και putFloat(). Κάθε μέθοδος καθορίζει ένα ζεύγος κλειδιού-τιμής που θα αποθηκευτεί στο SharedPreferences. Τέλος, αφού προστεθούν ή ενημερωθούν όλα τα επιθυμητά δεδομένα, οι αλλαγές δεσμεύονται στο SharedPreferences καλώντας ξανά την apply() στον editor. Με αυτόν τον τρόπο διασφαλίζεται ότι οι αλλαγές αποθηκεύονται μόνιμα. Ομοίως λειτουργεί και για την καταχώρηση πορτοφολιού/μετρητών.

Απόσπασμα κώδικα 7: Λειτουργία διαγραφής αντικειμένου wallet

```
...
val editor = sp.edit()
if (feature == "w") {
    editor.remove("name_of_wallet_$keyString")
    editor.remove("amount_in_wallet_$keyString")
    editor.remove("currency_wallet_$keyString")
}
...
editor.apply()
...
```

Στο παραπάνω απόσπασμα παρουσιάζεται η διαγραφή ενός αντικειμένου wallet μέσω της μεθόδου remove() σε περίπτωση που η μεταβλητή feature ισούται με το αλφαριθμητικό "w". Με την μέθοδο apply() όπως προαναφέραμε εφαρμόζεται οι αλλαγές στο SharedPreferences.

Απόσπασμα κώδικα 8: Λειτουργία διαγραφής αντικειμένου wallet

```
sp = getSharedPreferences("userOptions", MODE_PRIVATE)
...
val b_id = sp.getInt("bankID", 0)
for (b in 1..b_id) {
    val bankName = sp.getString("name_of_bank_$b", "null")
    if (bankName == "null")
        continue
    val bankAmount = sp.getFloat("amount_in_bank_$b", 0F)
    val bankCur = sp.getString("currency_bank_$b", "null")!!
    val bankdata = BankInfos(bankName!!, bankAmount, bankCur,
      ↪ b.toString(), "feature")
    adapter.addInfo(bankdata)
}
...
```

Παραπάνω μπορούμε να δούμε πώς γίνεται η ανάκτηση δεδομένων μέσω SharedPreferences ξεκινώντας με την αρχικοποίηση του τρέχον αναγνωριστικού τράπεζας (b_id) από το

SharedPreferences χρησιμοποιώντας την getInt(). Εάν δεν βρεθεί τιμή για το "bankID", η τιμή είναι προεπιλεγμένη σε 0. Αυτό το ID χρησιμοποιείται για την επανάληψη όλων των αποθηκευμένων τραπεζικών δεδομένων. Στη συνέχεια, επαναλαμβάνει κάθε καταχώρηση τράπεζας που είναι αποθηκευμένη στο SharedPreferences χρησιμοποιώντας έναν βρόχο for. Ο βρόχος επαναλαμβάνει από το 1 έως το ανακτημένο b_id, που υποδεικνύει τον αριθμό των τραπεζικών καταχωρήσεων. Εντός του βρόχου, τα δεδομένα για κάθε τραπεζική καταχώρηση ανακτώνται από το SharedPreferences χρησιμοποιώντας μεθόδους όπως η getString() και η getFloat(). Τα δεδομένα που ανακτώνται περιλαμβάνουν το όνομα της τράπεζας, το ποσό και το νόμισμα. Εάν το όνομα της τράπεζας είναι "null", ο βρόχος μεταβαίνει στην επόμενη επανάληψη χρησιμοποιώντας continue. Για κάθε έγκυρη καταχώρηση τράπεζας, δημιουργείται ένα αντικείμενο BankInfos χρησιμοποιώντας τα ανακτηθέντα δεδομένα. Αυτό το αντικείμενο αντιπροσωπεύει μια καταχώρηση τραπεζικών πληροφοριών και κατασκευάζεται χρησιμοποιώντας το όνομα της τράπεζας, το ποσό, το νόμισμα και ένα μοναδικό κλειδί (b.toString()). Το "feature" είναι μία μεταβλητή που δηλώνει το είδος του αντικειμένου(στην συγκεκριμένη περίπτωση ισούται με "b", δηλαδή bank). Κάθε αντικείμενο BankInfos προστίθεται σε έναν Adapter χρησιμοποιώντας τη μέθοδο addInfo(). Αυτό τον ενημερώνει με τις πληροφορίες της τράπεζας για εμφάνιση σε ένα στοιχείο UI, RecyclerView.

## 5.4 Υποσύστημα QR

Στόχος της διπλωματικής εργασίας είναι να αναδείξει την χρήση του QR στις ελληνικές αποδείξεις. Γι΄ αυτόν τον λόγο χρησιμοποιήθηκε μία έτοιμη βιβλιοθήκη που αφορά την λειτουργία του και μπορείτε να την βρείτε εδώ: https://github.com/G00fY2/quickie. Να σημειωθεί ότι για την χρήση της κάμερας ερωτάται ο χρήστης, καθώς η εφαρμογή σέβεται τα προσωπικά δικαιώματα και επιλογές του.

Πριν ξεκινήσουμε να γράφουμε κώδικα πρέπει να κάνουμε δύο βασικά βήματα για να προστεθεί η βιβλιοθήκη στο πρότζεκτ μας.

1. Εισαγωγή στα dependencies δύο implementantions στο build.gradle.
2. Να κάνουμε import στο κατάλληλο Activity.

Απόσπασμα κώδικα 9: Dependencies

```
implementation("io.github.g00fy2.quickie:quickie-bundled:1.9.0
    ↪ ")
implementation("org.jsoup:jsoup:1.17.2")
```

Απόσπασμα κώδικα 10: Imports

```
implementation("io.github.g00fy2.quickie:quickie-bundled:1.9.0
    ↪ ")
implementation("org.jsoup:jsoup:1.17.2")
```

Παρακάτω παρουσιάζεται ο κώδικας για την μεταφορά χρηματικού ποσού από ένα στοιχείο σε ένα άλλο. Ομοίως το QR λειτουργεί και στα υπόλοιπα Activities.

Απόσπασμα κώδικα 11: Κώδικας υλοποίησης του QR

```
qr.setOnClickListener{
    scanQrCodeLauncher.launch(null)
}
...
val scanQrCodeLauncher = registerForActivityResult(ScanQRCode
    ↪ ()) { result ->
    // handle QRResult
    if (result is QRResult.QRSuccess) {
        val content = result.content
        if (content is QRContent.Url) {
            val url = content.url
            lifecycleScope.launch( context = Dispatchers.Main)
                ↪ {
                task.execute(url)
            }
            lifecycleScope.launch( context = Dispatchers.Main)
                ↪ {
                val resp = task.get()
                val enteredamount = resp.toFloatOrNull()
                if (enteredamount == null) {
                    Toast.makeText(this@TransferActivity, "
                        ↪ Something␣went␣wrong.␣Try␣manually!"
                        ↪ , Toast.LENGTH_SHORT).show()
                    finish()
                    return@launch
                }
                val res = amount - enteredamount.toFloat()
                val editor = sp.edit()
                val toKey = (destination.selectedItem as
                    ↪ DestinationItem).key
                val toKeyAmount = (destination.selectedItem as
                    ↪  DestinationItem).amount.toFloat() +
                    ↪ enteredamount.toFloat()
                val toKeyFeature = (destination.selectedItem
                    ↪ as DestinationItem).feature
                if (feature == "w")
                    editor.putFloat("
                        ↪ amount_in_wallet_$keyString", res)
                else
```

```
                    editor.putFloat("amount_in_bank_$keyString
                        ↪ ", res)

                if (toKeyFeature == "w")
                    editor.putFloat("amount_in_wallet_$toKey",
                        ↪  toKeyAmount)
                else
                    editor.putFloat("amount_in_bank_$toKey",
                        ↪ toKeyAmount)
                editor.apply()
                Toast.makeText(this@TransferActivity, "
                    ↪ Transfer Successful", Toast.LENGTH_LONG)
                    ↪ .show()
                dialog.dismiss()
                val intent = Intent(this@TransferActivity,
                    ↪ FirstPageActivity::class.java)
                intent.flags = Intent.FLAG_ACTIVITY_NEW_TASK
                    ↪ or Intent.FLAG_ACTIVITY_CLEAR_TASK
                startActivity(intent)
            }
        }
    }
}
```

Στην αρχή έχουμε το κουμπί του QR το οποίο όταν πατηθεί εκτελείται ο κώδικας scanQrCodeLauncher.launch(null). Αυτό εκκινεί μια δραστηριότητα σάρωσης κώδικα QR χρησιμοποιώντας έναν προκαθορισμένο launcher (scanQrCodeLauncher). Ο scanQrCodeLauncher δημιουργείται με τη μέθοδο registerForActivityResult(). Καταχωρεί έναν launcher αποτελεσμάτων δραστηριότητας για τη δραστηριότητα ScanQRCode(), η οποία είναι υπεύθυνη για τη σάρωση κωδικών QR. Όταν λαμβάνεται το αποτέλεσμα της δραστηριότητας σάρωσης, καλείται η παρεχόμενη συνάρτηση για τον χειρισμό του αποτελέσματος. Μέσα στη συνάρτηση, ελέγχεται το αποτέλεσμα της δραστηριότητας σάρωσης κωδικών QR. Εάν το αποτέλεσμα υποδεικνύει επιτυχή σάρωση κώδικα QR (QRResult.QRSuccess), εξάγεται το περιεχόμενο του κώδικα QR. Εάν το περιεχόμενο αντιπροσωπεύει μια διεύθυνση URL (QRContent.Url), πραγματοποιείται περαιτέρω επεξεργασία. Με τον εντοπισμό μιας διεύθυνσης URL στο περιεχόμενο QR, η εφαρμογή ξεκινά ένα αίτημα δικτύου χρησιμοποιώντας το task.execute(url) για να αντλήσει πρόσθετες πληροφορίες που σχετίζονται με τη διεύθυνση URL. Αυτό το αίτημα εκτελείται ασύγχρονα μέσα σε ένα coroutine που εκκινείται με τη χρήση του lifecycleScope.launch. Μετά τη λήψη της απάντησης από το αίτημα δικτύου, η εφαρμογή επεξεργάζεται τα δεδομένα και ενημερώνει ανάλογα τα SharedPreferences (sp). Αυτό περιλαμβάνει την ενημέρωση του ποσού που σχετίζεται με τον τρέχοντα χρήστη (keyString) και του ποσού που σχετίζεται με τον προορισμό που καθορίζεται στον κωδικό QR (toKey). Το χαρακτηριστικό ("w" για πορτοφόλι, "b"

για τράπεζα) καθορίζει πού ενημερώνονται τα ποσά. Τέλος, ο χρήστης λαμβάνει ανατροφοδότηση σχετικά με την επιτυχία της λειτουργίας μεταφοράς μέσω ενός μηνύματος Toast. Επιπλέον, απορρίπτονται τυχόν ανοικτά παράθυρα διαλόγου (dialog.dismiss()) και ο χρήστης πλοηγείται πίσω στην πρώτη σελίδα της εφαρμογής (FirstPageActivity) για να αντικατοπτρίζει τις αλλαγές στο περιβάλλον εργασίας.

Συνοπτικά, αυτό το απόσπασμα κώδικα υλοποιεί μια λειτουργία σάρωσης κώδικα QR που επιτρέπει στους χρήστες να ξεκινούν μεταφορές μεταξύ διαφορετικών λογαριασμών σαρώνοντας κωδικούς QR που περιέχουν πληροφορίες προορισμού. Το περιεχόμενο του σαρωμένου κώδικα QR επεξεργάζεται και τα αντίστοιχα δεδομένα χρησιμοποιούνται για την ενημέρωση της κατάστασης της εφαρμογής που είναι αποθηκευμένη στο SharedPreferences.

## 5.5 Ασύγχρονο web crawling από URL

Η ασύγχρονη ανάκτηση δεδομένων στην Kotlin αναφέρεται στη διαδικασία ανάκτησης δεδομένων από εξωτερικές πηγές, όπως βάσεις δεδομένων, υπηρεσίες ιστού ή αρχεία, χωρίς να μπλοκάρει το κύριο νήμα εκτέλεσης. Αυτό είναι σημαντικό για τη διατήρηση μιας ευέλικτης διεπαφής χρήστη σε εφαρμογές Android, καθώς το μπλοκάρισμα του κύριου νήματος μπορεί να οδηγήσει σε μη ανταποκρινόμενα UI και κακή εμπειρία χρήστη.

Στην Kotlin, η ασύγχρονη ανάκτηση δεδομένων επιτυγχάνεται συνήθως με τη χρήση τεχνικών όπως οι coroutines ή κατασκευές ασύγχρονου προγραμματισμού όπως η AsyncTask (στο Android). Εμείς επιλέξαμε να το κάνουμε με τον δεύτερο τρόπο.

Η σημασία της ασύγχρονης ανάκτησης δεδομένων στην Kotlin έγκειται στην ικανότητά της να βελτιώνει την απόδοση της εφαρμογής, την απόκριση και την εμπειρία του χρήστη. Εκφορτώνοντας χρονοβόρες εργασίες, οι εφαρμογές μπορούν να παραμείνουν ευέλικτες στις αλληλεπιδράσεις των χρηστών, ενώ εκτελούν εργασίες όπως αιτήσεις δικτύου, ερωτήματα βάσης δεδομένων ή λειτουργίες αρχείων. Αυτό διασφαλίζει ότι το UI παραμένει ομαλό και διαδραστικό, βελτιώνοντας τη συνολική ευχρηστία και την ικανοποίηση των χρηστών. Επιπλέον, η ασύγχρονη ανάκτηση δεδομένων επιτρέπει στους προγραμματιστές να σχεδιάζουν πιο κλιμακούμενες και αποδοτικές εφαρμογές αξιοποιώντας την ταυτόχρονη χρήση για τη μεγιστοποίηση της χρήσης των πόρων και την ελαχιστοποίηση της καθυστέρησης. Επιτρέπει στις εφαρμογές να χειρίζονται αποτελεσματικά πολλαπλές ταυτόχρονες λειτουργίες, οδηγώντας σε ταχύτερη επεξεργασία δεδομένων και βελτιωμένη συνολική απόδοση του συστήματος.

Στο παρακάτω απόσπασμα κώδικα παρουσιάζεται η AsyncTask που χρησιμοποιήθηκε για την υλοποίηση όπως και η επεξήγηση της.

Απόσπασμα κώδικα 12: Κώδικας υλοποίησης του QR

```
val task = TransferActivity.Companion.MyAsyncTask(this)
...
companion object {
```

```
    class MyAsyncTask internal constructor(context:
        ↪ TransferActivity) : AsyncTask<String, String, String
        ↪ >() {
        private var resp: String? = null
        private val activityReference: WeakReference<
            ↪ TransferActivity> = WeakReference(context)
        override fun onPreExecute() {
            val activity = activityReference.get()
            Toast.makeText(activity, "Fetching␣data...", Toast
                ↪ .LENGTH_SHORT).show()
        }
        override fun doInBackground(vararg params: String?):
            ↪ String? {
            try {
                val url = params[0]!!
                val doc = Jsoup.connect(url).get()
                val classes: Elements = doc.select(".receipt")
                val text = classes.text()
                resp = text.substringAfter("Synolikoy␣posoy␣")
                    ↪ .substringBefore("␣euro")
            } catch (e: InterruptedException) {
                e.printStackTrace()
                resp = e.message
            } catch (e: Exception) {
                e.printStackTrace()
                resp = e.message
            }
            return resp
        }
        override fun onPostExecute(result: String?) {
            val activity = activityReference.get()
            if (result == null)
                return
            if (activity == null || activity.isFinishing)
                return
        }
    }
}
```

Ένα companion object είναι ένα αντικείμενο singleton που σχετίζεται με μια κλάση. Μπορεί να περιέχει μεθόδους και ιδιότητες που μοιράζονται σε όλες τις περιπτώσεις της κλάσης. Εδώ, το companion object ενθυλακώνει μια ασύγχρονη εργασία για την άντληση δεδομένων. Μέσα σε αυτό, υπάρχει μια ένθετη κλάση με το όνομα MyAsyncTask.

Αυτή η κλάση επεκτείνει την AsyncTask<String, String, String>(), υποδεικνύοντας ότι εκτελεί ασύγχρονα εργασίες παρασκηνίου, λαμβάνοντας μια παράμετρο συμβολοσειράς, δημοσιεύοντας την πρόοδο ως συμβολοσειρά και επιστρέφοντας ένα αποτέλεσμα συμβολοσειράς. Ο κατασκευαστής της MyAsyncTask λαμβάνει ως παράμετρο μια περίπτωση TransferActivity και δημιουργεί μια WeakReference σε αυτήν. Αυτό βοηθά στην αποφυγή διαρροών μνήμης κρατώντας μια ασθενή αναφορά στη δραστηριότητα. Η μέθοδος onPreExecute() καλείται πριν από την έναρξη της εργασίας παρασκηνίου. Εμφανίζει ένα μήνυμα που υποδεικνύει ότι η ανάκτηση δεδομένων βρίσκεται σε εξέλιξη. Η μέθοδος doInBackground() εκτελεί την πραγματική εργασία στο παρασκήνιο, η οποία περιλαμβάνει την άντληση δεδομένων από μια διεύθυνση URL χρησιμοποιώντας την Jsoup, μια βιβλιοθήκη της Java για την εργασία με HTML. Η μέθοδος Jsoup.connect(url) δημιουργεί μια σύνδεση με την καθορισμένη διεύθυνση URL και η μέθοδος .get() αντλεί το περιεχόμενο HTML από αυτή τη διεύθυνση URL. Αυτό το περιεχόμενο HTML αναπαρίσταται ως αντικείμενο εγγράφου (doc). Η μέθοδος doc.select(".receipt") επιλέγει στοιχεία από το έγγραφο HTML με το όνομα κλάσης "receipt" και τα επιστρέφει ως μια συλλογή στοιχείων. Αυτή η συλλογή ανατίθεται στη μεταβλητή classes. Η μέθοδος .text() καλείται στη συλλογή κλάσεων για την εξαγωγή του περιεχομένου κειμένου των επιλεγμένων στοιχείων HTML. Αυτό το περιεχόμενο κειμένου, που περιέχει πληροφορίες για μια απόδειξη, εκχωρείται στη μεταβλητή text. Η μέθοδος text.substringAfter("Συνολικού ποσού ") εξάγει την υποσειρά του κειμένου που εμφανίζεται μετά την ελληνική φράση "Συνολικού ποσού ". Στη συνέχεια, η μέθοδος .substringBefore("ευρώ") βελτιώνει περαιτέρω την εξαγόμενη υποσειρά εξάγοντας το τμήμα του κειμένου πριν από τη φράση "ευρώ". Η προκύπτουσα υποσειρά αντιπροσωπεύει το συνολικό ποσό που αναφέρεται στην απόδειξη. Το εξαγόμενο συνολικό ποσό εκχωρείται στη μεταβλητή resp, η οποία θα επιστραφεί αργότερα ως αποτέλεσμα. Τέλος, onPostExecute() καλείται μετά την ολοκλήρωση της εργασίας παρασκηνίου. Ανακτά την περίπτωση TransferActivity από την αδύναμη αναφορά και ελέγχει αν είναι ακόμα έγκυρη. Εάν η δραστηριότητα είναι ακόμα ενεργή, μπορεί να χειριστεί τα δεδομένα που έχουν ληφθεί ή να εκτελέσει τυχόν απαραίτητες ενημερώσεις του περιβάλλοντος εργασίας.

## 5.6 Χρήση σύγχρονων μεθοδολογιών και τεχνικών ανάπτυξης λογισμικού

Οι σύγχρονες μεθοδολογίες και τεχνικές ανάπτυξης λογισμικού, όπως η χρήση βιβλιοθηκών όπως η ZXing για την επεξεργασία κωδικών QR και οι coroutines για τον ασύγχρονο προγραμματισμό στην Kotlin, είναι απαραίτητες για τη δημιουργία αξιόπιστων, κλιμακούμενων και συντηρήσιμων λύσεων λογισμικού.

Κατά την ανάπτυξη της εφαρμογής μας, δώσαμε προτεραιότητα στην υιοθέτηση σύγχρονων μεθοδολογιών και τεχνικών ανάπτυξης λογισμικού για να διασφαλίσουμε την ανταγωνιστικότητά της στο σημερινό δυναμικό ψηφιακό τοπίο. Αγκαλιάζοντας τις σύγχρονες τεχνολογίες, υλοποιήσαμε στρατηγικά χαρακτηριστικά όπως ο σαρωτής κωδικών QR χρησιμοποιώντας την πρωτοποριακή βιβλιοθήκη της Google, Quickie. Αυτή η βιβλιο-

θήκη, η οποία χτίστηκε πάνω στο ZXing, μας παρείχε ένα ολοκληρωμένο και φιλικό προς τον χρήστη API, απλοποιώντας την ενσωμάτωση της λειτουργίας σάρωσης κωδικών QR στην εφαρμογή μας. Αξιοποιώντας τις προηγμένες δυνατότητες της Quickie, συμπεριλαμβανομένων του ισχυρού χειρισμού σφαλμάτων και των επιλογών προσαρμογής, μπορέσαμε να προσφέρουμε μια απρόσκοπτη και διαισθητική εμπειρία σάρωσης κωδικών QR στους χρήστες μας. Επιπλέον, στην προσπάθειά μας για βέλτιστη απόδοση και απόκριση, χρησιμοποιήσαμε τεχνικές ασύγχρονης ανάπτυξης λογισμικού σε όλη την αρχιτεκτονική της εφαρμογής. Συγκεκριμένα, χρησιμοποιήσαμε το AsyncTask, ένα ισχυρό εργαλείο για την ασύγχρονη εκτέλεση εργασιών στο παρασκήνιο. Το AsyncTask μας επέτρεψε να αποφορτίσουμε τις λειτουργίες έντασης πόρων, όπως τα αιτήματα δικτύου και η επεξεργασία δεδομένων, από το κύριο νήμα του UI, αποτρέποντας έτσι το πάγωμα του UI και εξασφαλίζοντας μια ομαλή εμπειρία χρήσης. Αξιοποιώντας την απλότητα και την ευελιξία του AsyncTask, πετύχαμε αποτελεσματική διαχείριση της ταυτόχρονης εκτέλεσης και απρόσκοπτη ενσωμάτωση των ασύγχρονων εργασιών στη ροή εργασιών της εφαρμογής μας.

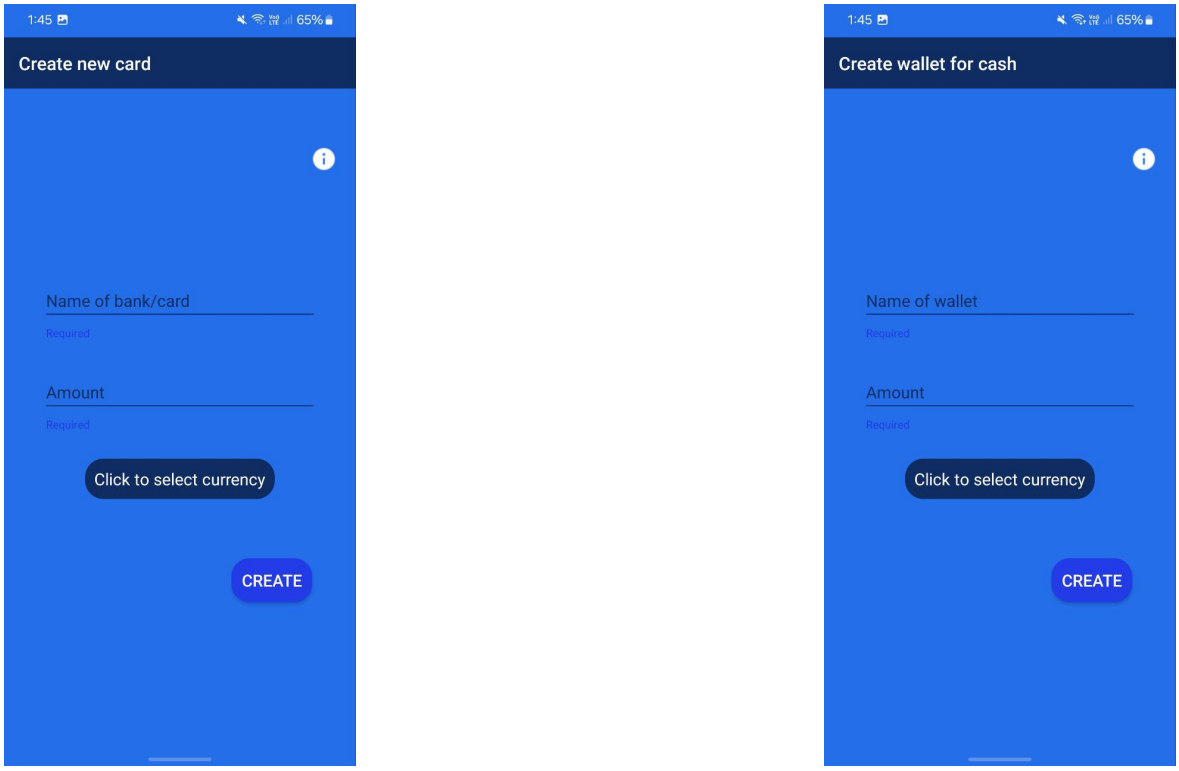
Συνοψίζοντας, αξιοποιώντας τη δύναμη των σύγχρονων μεθοδολογιών και τεχνικών ανάπτυξης λογισμικού, συμπεριλαμβανομένης της ενσωμάτωσης βιβλιοθηκών αιχμής όπως η Quickie και της υιοθέτησης παραδειγμάτων ασύγχρονου προγραμματισμού με την AsyncTask, παραδώσαμε μια υψηλής ποιότητας, ευέλικτη και επικεντρωμένη στον χρήστη εφαρμογή που ανταποκρίνεται στις απαιτήσεις των σημερινών απαιτητικών χρηστών.

# 6 Παραδείγματα χρήσης της εφαρμογής

Το Expense Manager στόχο έχει να βοηθήσει τον χρήστη να χειρίζεται τα οικονομικά στοιχεία που του ανήκουν και να τα βλέπει όλα μαζεμένα μέσα σε μία εφαρμογή.

## 6.1 Προσθήκη κάρτας/μετρητών

Στην εκκίνηση της εφαρμογής εμφανίζονται 3 κουμπιά:Προσθήκη Κάρτας, Προσθήκη Πορτοφολιού, Βοήθεια. Επιλέγοντας ένα από τα πρώτα δύο κουμπιά μπορεί να προσθέσει τα πρώτα δεδομένα, όπως είναι μία κάρτα/τράπεζα ή ένα εικονικό πορτοφόλι. Θα πρέπει να δώσει ένα όνομα, το χρηματικό ποσό του και να διαλέξει το νόμισμα που επιθυμεί. Στο όνομα μπορεί να προσθέσει μέχρι δέκα χαρακτήρες.

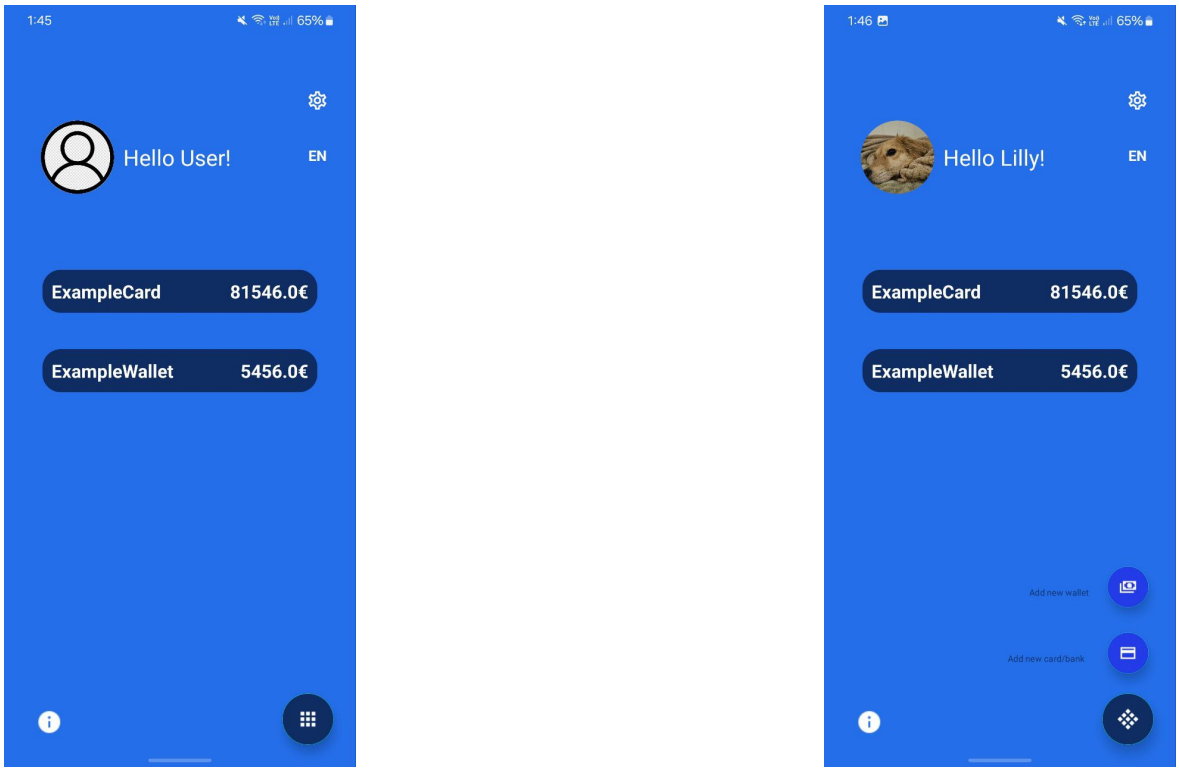



**Εικόνα 6:** *Προσθήκη Δεδομένου*

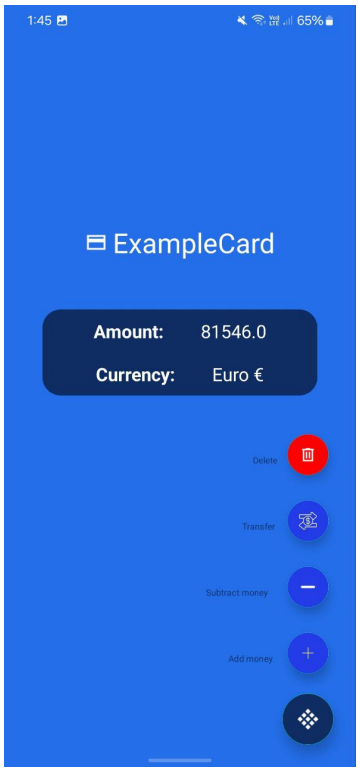
## 6.2 Αρχική οθόνη μετά την δημιουργία δεδομένων

Σε αυτήν την δραστηριότητα ο χρήστης έρχεται σε επαφή με τα δεδομένα που δημιούργησε κατά την έναρξη της εφαρμογής. Εδώ βλέπει όλες τις κάρτες/τράπεζες και τα μετρητά/πορτοφόλια που έχει δημιουργήσει και πατώντας πάνω σε αυτό μπορεί να δει περισσότερες λεπτομέρειες σχετικά με το συγκεκριμένο αντικείμενο. Ακόμη, μπορεί να πατήσει το κουμπί κάτω δεξιά και να δει τις δύο λειτουργίες (Προσθήκη κάρτας, Προσθήκη πορτοφολιού) που του δίνονται. Επιπρόσθετα, κάτω αριστερά υπάρχει κουμπί επεξήγησης της σελίδας όπως και πάνω δεξιά υπάρχουν οι ρυθμίσεις που μπορεί να δει ο χρήστης. Κάτω από το κουμπί των ρυθμίσεων υπάρχει ένα κουμπί που αν πατηθεί μπορεί να αλλάξει την γλώσσα από αγγλικά σε ελληνικά.

Τέλος, δίνεται στον χρήστη η δυνατότητα να αλλάξει το όνομα 'User' όπως και φωτογραφία. Εδώ βλέπει όλες τις κάρτες/τράπεζες και τα μετρητά/πορτοφόλια που έχει δημιουργήσει και πατώντας πάνω σε αυτό μπορεί να δει περισσότερες λεπτομέρειες σχετικά με το συγκεκριμένο αντικείμενο. Ταυτόχρονα μπορεί να διαλέξει μία από τις 4 ενέργειες(Προσθήκη χρημάτων, Αφαίρεση χρημάτων , Μεταφορά χρημάτων, Διαγραφή δεδομένου) που του προσφέρει η εφαρμογή πατώντας το κουμπί κάτω δεξιά.




**Εικόνα 7:** *Αρχική οθόνη*



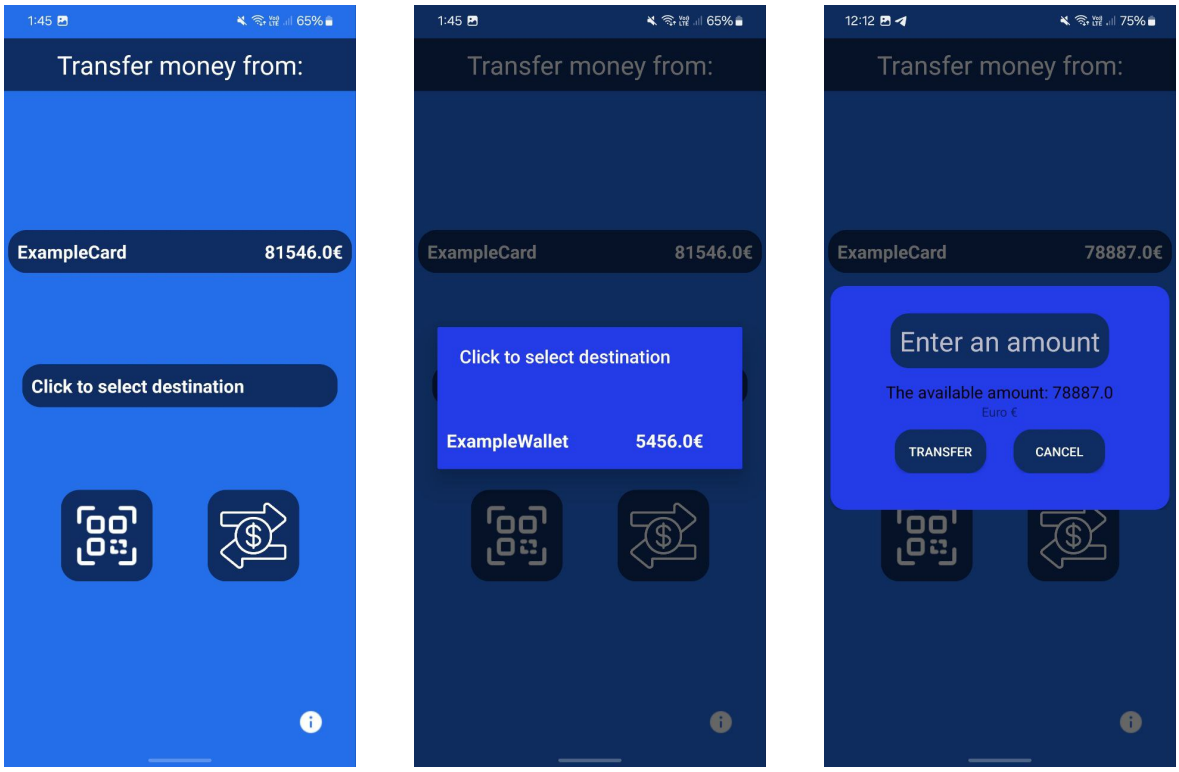

**Εικόνα 8:** *Μορφή ενός δεδομένου*

## 6.3 Προσθήκη μιας συναλλαγής χειροκίνητα

Αρχικά, έχουμε την ¨Προσθήκη Χρημάτων' κατά την οποία πρέπει ο χρήστης να προσθέσει ένα ποσό, ώστε αυτό να προστεθεί το ποσό στο ήδη υπάρχων. Αυτό γίνεται πατώντας το δεύτερο κουμπί με το σύμβολο "+". Τέλος, στη σελίδα αυτή υπάρχει και ένα κουμπί κάτω δεξιά το οποίο εξηγεί τι μπορεί να κάνει ο χρήστης.

Η επόμενη ενέργεια που μπορεί να εκτελέσει ο χρήστης είναι η ¨Αφαίρεση Χρημάτων' από ένα δεδομένο πατώντας το δεύτερο κουμπί που εμφανίζεται με το σύμβολο "-". Εδώ πρέπει να τονιστεί ότι το ποσό μπορεί να είναι και αρνητικό με τη σκέψη ότι ο χρήστης μπορεί να δημιουργήσει μία πιστωτική κάρτα που το ποσό το ξεχρεώνει μετά από κάποιο διάστημα.

Η ¨Μεταφορά Χρημάτων' είναι η τρίτη ενέργεια που συναντάει κάποιος κοιτώντας από κάτω προς τα πάνω. Εδώ, ο χρήστης μπορεί να επιλέξει τον προορισμό στον οποίο θέλει να στείλει το χρηματικό ποσό που έχει εισαγάγει πατώντας το δεύτερο κουμπί που για σύμβολο έχει τα δύο βέλη και το δολάριο στη μέση. Εδώ πρέπει να τονιστεί ότι οι επιλογές που εμφανίζονται στον χρήστη για τον προορισμό πρέπει να έχουν ίδιο νόμισμα ώστε να γίνεται η μεταφορά.




**Εικόνα 9:** *Δραστηριότητα 'Μεταφορά Χρημάτων'*

## 6.4 Προσθήκη συναλλαγή κάνοντας χρήση του QR ελληνικής απόδειξης

Στην υποενότητα αυτή θα παρουσιαστεί η χρήση του QR scanner για τις ενέργειες που πρωτοαναφέρθηκαν. Έστω ότι επιλέγουμε την 'Προσθήκη χρημάτων' κατά την οποία ο χρήστης πρέπει να επιτρέψει στη εφαρμογή να χρησιμοποιήσει την κάμερα είτε μόνο κατά την χρήση της εφαρμογής ή ποτέ ή για πάντα πατώντας το κουμπί του 'QR'. Στην συνέχεια πρέπει να γείρει την κάμερα προς το QR της απόδειξης. Αν υπάρχει κάποιο θέμα με τον κωδικό γρήγορης απόκρισης η εφαρμογή προτρέπει τον χρήστη να εκτελέσει την ενέργεια που επιθυμεί χειροκίνητα. Τέλος, πρέπει να τονιστεί ότι για τη χρήση του QR scanner πρέπει να υπάρχει πρόσβαση στο Διαδίκτυο. Όμοια λειτουργεί και οι άλλες δύο λειτουργίες. Αν η διαδικασία σκαναρίσματος είναι επιτυχής θα εμφανιστούν δύο μηνύματα στο κάτω μέρος της οθόνης με το πρώτο να λέει ότι γίνεται ανάκτηση δεδομένων και το δεύτερο ότι προστέθηκε το ποσό που του έδωσε ο χρήστης. (Ομοίως και στις υπόλοιπες ενέργειες)


**Εικόνα 10:** *Χρήση του QR scanner*

# 7 Δυσκολίες που προέκυψαν κατά την υλοποίηση

Η υλοποίηση μιας εφαρμογής σε οποιαδήποτε γλώσσα προγραμματισμού παρουσιάζει μυριάδες προκλήσεις για τους προγραμματιστές.

Στην συγκεκριμένη εφαρμογή υπήρξαν κάποιες δυσκολίες κατά την υλοποίηση. Ένα βασικό πρόβλημα είναι ότι η σάρωση του κωδικού QR λειτουργεί μέχρι στιγμής σε ελληνικές αποδείξεις με τις οποίες υπήρχε δυσκολία και στο HTML αρχείο που μειώνει την ώρα εκτέλεσης της εφαρμογής καθώς το πεδίο που μας ενδιαφέρει να "τραβήξουμε" δεν έχει κάποιο όνομα στο tag. Άλλο ένα ζήτημα που υπήρχε που υπάρχει αφορά πάλι το QR καθώς δεν μπορεί να λειτουργήσει χωρίς τη σύνδεση σε διαδίκτυο, γιατί για να μπορέσει η εφαρμογή να ανακτήση την τιμή της απόδειξης πρέπει να μεταβιβαστεί στην σελίδα της ΑΑΔΕ. Πιο απλά προβλήματα αφορούσαν τη σχεδίαση των Layouts που για την καλύτερη διαχείριση τους δεν πρέπει να δίνονται συγκεκριμένες διαστάσεις.

Συνολικά, η διαδικασία υλοποίησης μιας εφαρμογής σε οποιαδήποτε γλώσσα προγραμματισμού είναι ένα πολύπλευρο εγχείρημα που απαιτεί τεχνογνωσία, υπομονή και επιμονή από τους προγραμματιστές πόσο μάλλον αν πρόκειται για την πρώτη επαφή με την εκάστοτε γλώσσα.

# 8 Μελλοντικές Υλοποιήσεις

Μία εφαρμογή πρέπει διαρκώς να αναβαθμίζεται και να εξελίσσεται, τόσο για την καλύτερη εξυπηρέτηση των αναγκών των χρηστών όσο και τον συγχρονισμό της με τις νέες τεχνολογίες της εκάστοτε προγραμματιστικής γλώσσας.

Μία από τις μελλοντικές ιδέες για την αναβάθμιση της εφαρμογής είναι η εισαγωγή ενός API που αφορά την αλλαγή νομίσματος σύμφωνα με τις τρέχουσες τιμές στην αγορά. Αυτό θα διευκολύνει τους χρήστες να αλλάζουν τα χρηματικά ποσά που διαθέτουν έχοντας τις τελευταίες ενημερώσεις των ισοτιμιών. Άλλη μία λειτουργία που θα ενταχθεί είναι η παρουσίαση του ιστορικού συναλλαγών του χρήστη. Αυτό θα τον διευκολύνει να διαχειριστεί με τον βέλτιστο δυνατό τρόπο τα οικονομικά του στοιχεία. Συνδυαστικά θα δημιουργηθούν και στατιστικά δεδομένα που ο χρήστης θα έχει τη δυνατότητα να παρακολουθεί παράλληλα τα έσοδα και τα έξοδα του. Επιπλέον, θα φανεί χρήσιμο μέσα στην εφαρμογή ο χρήστης να μπορεί μία ενσωματωμένη αριθμομηχανή και να κάνει αριθμητικές πράξεις, ώστε να μην χρειάζεταινα εισέρχεται και να εξέρχεται της εφαρμογής για να χρησιμοποιήσει την ήδη υπάρχουσα αριθμομηχανή που διαθέτει η συσκευή του. Αυτή η λειτουργία θα είναι χρήσιμη μαζί με ένα σημειωματάριο που θα μπορεί να κρατάει σημειώσεις. Τέλος, μία τελευταία ιδέα που υπάρχει είναι η δημιουργία της εφαρμογής και για iOS περιβάλλον, ώστε να εξυπηρετείται μεγαλύτερο εύρος χρηστών.

Στόχος μας είναι ο χρήστης να μπορεί να χρησιμοποιεί την εφαρμογή, ώστε να προχωρήσει από το παραδοσιακό τετράδιο, το οποίο χρησιμοποιούσαν απο τα παλαιότερα χρόνια οι άνθρωποι για να σημειώνουν τα έσοδα και τα έξοδα τους καθημερινά και μηνιαία, στον εκσυγχρονισμένο τρόπο που παρέχει η εφαρμογή αυτή.

# 9 Online παρουσία

## 9.1 Github

Επιπλέον, ο κώδικας της εφαρμογής είναι ανεβασμένος και στο Github τον οποίο μπορεί να κατεβάσει όποιος θέλει και να εξερευνήσει τον κώδικα ή και να τον επεξεργαστεί. Να τονιστεί ότι δεν υπάρχει το εκτελέσιμο αρχείο.


(https://github.com/stefaniatzanera/Expense-Manager-Mobile-Application )

# 10 Συμπεράσματα

Σε αυτήν την διπλωματική εργασία συζητήθηκαν σημαντικά αντικείμενα ξεκινώντας με τους διάφορους τρόπους πληρωμής και εξηγώντας πόσο σημαντική είναι η διαχείριση εξόδων τονίζοντας τυχόν προκλήσεις που μπορεί να αντιμετωπίσει ο άνθρωπος. Κλείνοντας το πρώτο κεφάλαιο κάποιος μπορεί να διαβάσει πληροφορίες για τις αποδείξεις και τα ηλεκτρονικά παραστατικά επικεντρώνοντας την προσοχή μας κυρίως στο Ελληνικό σύστημα στην έλλειψη εφαρμογών για την οικονομική διαχείριση καθώς και στους λόγους για τους οποίους απουσιάζουν από την κοινωνία. Στην συνέχεια, παρουσιάζονται τα είδη των εφαρμογών βάσει των τεχνολογιών ανάπτυξης λογισμικού και αντίστοιχα βάσει του περιεχομένου τους. Συνεχίζοντας με το τρίτο κεφάλαιο, συζητάμε για θέματα που αφορούν το QR όπως είναι μία ιστορική αναδρομή, από που προήλθε, τα διάφορα είδη που υπάρχουν, τα οφέλη της χρήσης του, τις διαφορές που έχει με το παραδοσιακό Barcode και τις χρήσεις του σε διάφορους τομείς της καθημερινότητας. Επιπρόσθετα αναλύουμε το προγραμματιστικό μέρος της εφαρμογής που αφορά τον ασύγχρονο προγραμματισμό με τον οποίο τραβάμε την τιμή που μας ενδιαφέρει από τις αποδείξεις και για την υλοποίηση του QR. Στο έκτο κεφάλαιο παρουσιάζονται μέσω εικόνων παραδείγματα χρήσης της εφαρμογής για την καλύτερη κατανόηση της εφαρμογής. Τέλος, κλείνουμε με τις δυσκολίες που προέκυψαν κατά την υλοποίηση της εφαρμογής, τους μελλοντικούς στόχους για την εξέλιξη της εφαρμογής και την online παρουσία της εφαρμογής. Συμπερασματικά, η Expense Manager εφαρμογή στοχεύει στην διευκόλυνση του χρήστη στην διαχείριση των οικονομικών δεδομένων με την χρήση μίας μόνο εφαρμογής και την ανάδειξη της χρήσης του QR μέσω των αποδείξεων που λαμβάνει από τις επιχειρήσεις, για την προσθήκη συναλλαγών στην εφαρμογή.

# Βιβλιογραφία


[1] `https://github.com/stefaniatzanera/Expense-Manager-Mobile-Application`

[2] https://developer.android.com/codelabs/build-your-first-android-app-kotlin#2

[3] https://developer.android.com/training/data-storage/shared-preferences

[4] https://www.youtube.com/watch?v=p0nhM5irW7Y

[5] https://kotlinlang.org/

[6] https://blog.logrocket.com/android-data-storage-guide-kotlin-sharedpreferences/

[7] https://github.com/hdodenhof/CircleImageView/blob/master/sample/src/main/res/layout/activit

[8] https://www.geeksforgeeks.org/how-to-use-putextra-and-getextra-for-string-data-in-android/

[9] https://developer.android.com/develop/ui/views/layout/recyclerview

[10] https://stackoverflow.com/questions/7075349/android-clear-activity-stack

[11] https://stackoverflow.com/questions/3669325/notifydatasetchanged-example

[12] https://stackoverflow.com/questions/11591729/how-to-call-listview-adapter-from-another-activity-for-refresh-notifydatasetch

[13] https://developer.android.com/develop/ui/views/components/floating-action-button

[14] https://riptutorial.com/android/example/10205/store–retrieve–remove-and-clear-data-from-sharedpreferences

[15] https://manserpatrice.medium.com/parse-html-with-jsoup-in-kotlin-69ab7fe4cb28

[16] https://www.youtube.com/watch?v=6OFniVVzmgQ

[17] https://www.youtube.com/watch?v=K4CGYiQu52s

[18] https://www.youtube.com/watch?v=g_ZxWOUCbHg

[19] https://github.com/G00fY2/quickie

[20] https://mvnrepository.com/artifact/org.jsoup/jsoup

[21] https://www.geeksforgeeks.org/android-applications-and-their-categories/

[22] https://www.businessofapps.com/app-developers/research/types-of-mobile-apps/

[23] https://www.qrcode.com/en/history/

[24] https://www.qr-code-generator.com/blog/how-qr-codes-work-and-their-history/

[25] https://developer.android.com/guide/components/activities/activity-lifecycle

[26] https://www.kodeco.com/21382977-android-lifecycle

[27] https://stuff.mit.edu/afs/sipb/project/android/docs/guide/topics/data/data-storage.html

[28] https://www.geeksforgeeks.org/storage-system-to-store-data-in-android/

[29] https://www.scaler.com/topics/android/storage-in-android/

[30] https://kotlinlang.org/docs/comparison-to-java.html

[31] https://www.baeldung.com/kotlin/java-vs-kotlin