ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ
ΣΧΟΛΗ ΟΙΚΟΝΟΜΙΑΣ, ΔΙΟΙΚΗΣΗΣ ΚΑΙ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

Εργαστήριο Γνώσης και Αβεβαιότητας

Μεταπτυχιακή Εργασία

Ανάπτυξη Android εφαρμογής καταγραφής της εξειδικευμένης υποστήριξης της ζωής στα παιδιά ”m-PALS”

Γιαννόπουλος Κωνσταντίνος
2022202102004

Επιβλέποντες:

Μανόλης Γουάλλες
Αναπληρωτής Καθηγητής

Βασίλης Πουλόπουλος
Επίκουρος Καθηγητής

Τρίπολη, Ιούλιος 2023

Συμβουλευτική Επιτροπή:

1. Μανόλης Γουάλλες, Αναπληρωτής Καθηγητής
2. Βασίλης Πουλόπουλος, Επίκουρος Καθηγητής

.

Εγκρίθηκε από την εξεταστική επιτροπή την 31/07/2023.

1.Μανόλης Γουάλλες, Αναπληρωτής Καθηγητής

2.Λέπουρας Γεώργιος, Καθηγητής

3.Βασιλάκης Κωνσταντίνος, Καθηγητής

.....

Γιαννόπουλος Κωνσταντίνος
Τμήμα Πληροφορικής και Τηλεπικοινωνιών
Πανεπιστήμιο Πελοποννήσου

Η παρούσα εργασία εκπονήθηκε σε συνεργασία με το Εργαστήριο Γνώσης και Αβεβαιότητας (ΓΑΒ LAB).

Περίληψη

Η παρούσα διπλωματική εργασία έχει ως βασικό στόχο την ανάπτυξη της mobile εφαρμογής m-Pals η χρήση της οποίας πιστεύουμε ότι θα συμβάλει καθοριστικά στην καταγραφή και αντιμετώπιση επειγόντων παιδιατρικών περιστατικών.

Η εφαρμογή, σχεδιάστηκε αξιοποιώντας τεχνολογίες αιχμής ενώ δόθηκε ιδιαίτερη έμφαση στην φιλοκότητα προς τον χρήστη, στην λειτουργικότητα και την ευχρηστία μιας και προορίζεται για χρήση κάτω από δύσκολες και επείγουσες συνθήκες

Η βάση για την έναρξη της εργασίας ήταν η δημοσιεύση «Καταγραφή της εξειδικευμένης υποστήριξης της ζωής στα παιδιά - m-Pals» των Αντωνόπουλος Σταύρος, Γρίβας Γρηγόριος, Καλαράς Γεώργιος, Τσακίρη Δήμητρα, Παπα- δήμα Χριστίνα [5] στην οποία και αναλύεται ο πίνακας/πρωτόκολλο m-PALS. Η mobile εφαρμογή υλοποιήθηκε για συσκευές Android (Tablet και κινητά) και έχουν δημιουργηθεί ξεχωριστές διεπαφές για κάθετο και οριζόντιο προσανατολισμό.

Λέξεις Κλειδιά: Android εφαρμογή, Java, Νοσοκομεία, Παιδιατρική

Abstract

The present thesis aims to develop the mobile application m-Pals, whose use we believe will significantly contribute to the recording and management of urgent pediatric incidents.

The application was designed using cutting-edge technologies, with special emphasis on user-friendliness, functionality, and usability, as it is intended for use under difficult and urgent conditions.

The basis for commencing this work was the publication ”MONITORING PEDIATRIC ADVANCED LIFE SUPPORT ’m-PALS’” by Antonopoulos Stavros, Grivas Grigorios, Kalaras Georgios, Tsakiri Dimitra, Papadima Christina, [5] which analyzes the m-PALS table/protocol. The mobile application was implemented for Android devices (tablets and smartphones), with separate interfaces created for vertical and horizontal orientations.

Keywords: Android Application, Java, Hospitals, Pediatric

Πίνακας περιεχομένων

Πίνακας σχημάτων

Εισαγωγή

Το αντικείμενο της εργασίας είναι η σχεδίαση και ανάπτυξη μιας mobile ιατρικής εφαρμογής της οποίας η κύρια λειτουργία είναι η καταγραφή και επισκόπηση δεδομένων εξειδικευμένης υποστήριξης της ζωής παιδιατρικών ασθενών. Η εφαρμογή αναπτύχθηκε για mobile συσκευές με λειτουργικό σύστημα android και μπορεί να λειτουργήσει σε κάθετο και οριζόντιο προσανατολισμό σε συσκευές κινητής τηλεφωνίας και ταμπλέτες.

Κίνητρο για την ανάπτυξη της εφαρμογής ήταν επί της ουσίας το συμπέρασμα των Αντωνόπουλος Σταύρος, Γρίβας Γρηγόριος, Καλαράς Γεώργιος, Τσακίρη Δήμητρα, Παπαδήμα Χριστίνα [5] από την δημοσίευση με θέμα «Καταγραφή της εξειδικευμένης υποστήριξης της ζωής στα παιδιά - m-Pals». Ποιο συγκεκριμένα από την ανωτέρω δημοσίευση είδαμε ότι υπήρχε έλλειψη από ένα πίνακα-πρωτόκολλο καταγραφής της εξειδικευμένης υποστήριξης σε παιδιά. Οι ανωτέρω πρότειναν τον πίνακα-πρωτόκολλο mPals. Βέβαια ενώ ο πίνακας-πρωτόκολλο έδωσε λύση στο πρόβλημα της καταγραφής εμφανίζει μειονεκτήματα ως προς την χρήση του,όταν αυτή γίνεται σε έντυπα (εκτυπωμένοι πίνακες). Κάποια από τα μειονεκτήματα του έντυπου πίνακα είναι: α) η εύκολη αναζήτηση, β) η γρήγορη επισκόπηση, γ) οι πολλαπλές καταγραφές εξετάσεων που απαιτούν πολλαπλά φύλλα, δ) η διευκόλυνση της συνεργασίας.

Στόχος της εργασίας, όπως έχει αναφερθεί και παραπάνω, είναι η ανάπτυξη μιας mobile εφαρμογής η οποία επί της ουσίας θα κάνει πράξη τον πίνακα-πρωτόκολλο mPals. Πιο συγκεκριμένα η εφαρμογή θα λύσει μεταξύ άλλων τα εξής προβλήματα που παρουσιάζονται στην έντυπη καταγραφή: **εύκολη αναζήτηση:** η αναζήτηση ασθενών μέσω της mobile εφαρμογής γίνεται εξαιρετικά εύκολη μιας και το μόνο που πρέπει να γνωρίζει ο κλινικός ιατρός, είναι το όνομα / επώνυμο του ασθενούς, **γρήγορη επισκόπηση:** η επισκόπηση των καταγραφών γίνεται επίσης πολύ εύκολη μιας και με 1-2 κλικ έχουμε την πλήρη εικόνα του ασθενούς, **πολλαπλές καταγραφές εξετάσεων:** ενώ στην έντυπη καταγραφή θα χρειάζονται πολλαπλά φύλα καταγραφής, στην mobile εφαρμογή καταχωρούνται οι εξετάσεις και παρουσιάζονται σε μορφή λίστας σε μία μόνο οθόνη, **διευκόλυνση συνεργασίας:** τέλος με την χρήση της εφαρμογής, σε συνδυασμό με την τυποποίηση που παρέχει το πρωτόκολλο, εξαλείφει σε μεγάλο ποσοστό τα λάθη καταγραφής, τους δυσανάγνωστους γραφικούς χαρακτήρες κ.λπ.

Τέλος, η δομή της παρούσας εργασίας έχει ώς εξής:
Στο κεφάλαιο 1 γίνεται η ανάλυση του πρωτοκόλλου m-Pals.
Στο κεφάλαιο 2 παρουσιάζονται οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής.
Στο κεφάλαιο 3 γίνεται η παρουσίαση του κώδικα της εφαρμογής με στιγμιότυπα οθόνης και τις αντίστοιχες περιγραφές.
Στο κεφάλαιο 4 γίνεται παρουσίαση (σε μορφή manual) της εφαρμογής m-Pals με σχετικά στιγμιότυπα οθόνης και οδηγίες/σχόλια επί των στιγμιοτύπων.

Κεφάλαιο 1

Ανάλυση πίνακα-πρωτοκόλλου m-PALS

m - PALS

1	***NAME***		***DATE*** ../../....		***TIME*** .. : ..		***BODY WEIGHT*** *kg/lbs*		***BODY HEIGHT*** *cm/inch*	
2	**B**ehaviour		**B**ody color				**B**reathing			
	Safety		**S**hout				**S**timulate			
3	**TIME**									
4	**(AB) AIRWAY/BREATHING**									
	Airway management/ manoeuvre									
	Oropharyngeal/Nasopharyngeal airway									
	Intubation / laryngeal mask airway-*LMA* / AMBU									
	Respiratory Rate									
	Auscultation/Work									
	SpO2									
	Routes of O2 administration									
	Medications									
	(C) CIRCULATION									
	Beats/Min									
	Blood Pressure									
	CRT									
	Preload/Pulse Volume									
	Dxt									
	IV/IO access									
	Fluids									
	Medications (AD,ADN,AMD,LID,ATR,SBC,DEX,NLX)									
	ECG / SHOCK									
	(D) DISABILITY / NEUROLOGICAL STATUS									
	GSC/AVPU									
	Pupillary reactions									
	Increased intracranial pressure/management									
	(E) EXPOSURE									
	Temperature									
	Allergy									
	Medication									
	Past History									
	Last Meal									
	Enviroment/exposure									
5	***Think 4H,4T*** (hypoxia, hypokalaemia/hyperkalaemia, hypothermia/hyperthermia, hypovolaemia, tension pneumothorax, tamponade, thrombosis, toxins)									

AD : Adrenaline, ADN : Adenosine, AMD : Amiodarone, LID : Lidocaine, ATR : Atropine, SBC : Sodium Bicarbonate, DEX : Dextrose, NLX : Naloxone

Στο παρόν κεφάλαιο θα αναλυθούν βάση του πίνακα-πρωτοκόλλου m-PALS, οι ενότητες καθώς και οι προδιαγραφές της εφαρμογής που αναπτύχθηκε ως διπλωματική. Σύμφωνα με την δημοσίευση η προσπάθεια monitoring και η συστηματική και τυποποιημένη καταγραφή αυτού, κρίνεται αναγκαία για τη σωστή διαχείριση του επείγοντος παιδιατρικού περιστατικού με στόχο την επιτυχή έκβασή του. Έτσι, έχοντας ως δεδομένο τον πίνακα-πρωτόκολλο καταγραφής της εξειδικευμένης υποστήριξης της ζωής στα παιδιά, «m-PALS», όπως τον προτείνουν οι Αντωνόπουλος Σταύρος, Γρίβας Γρηγόριος, Καλαράς Γεώργιος, Τσακίρη Δήμητρα, Παπαδήμα Χριστίνα [5] και έχοντας σαν στόχο να αποτελέσει καταγραφικό εργαλείο για κάθε υγειονομικό που θα κληθεί να αντιμετωπίσει το παιδιατρικό επείγον αναπτύχθηκε μιας εύχρηστη mobile Android εφαρμογή η οποία λόγω της τυποποιημένης του δομής (πρωτόκολλο) και της εύκολης καταγραφής των πληροφοριών θα μπορεί να χρησιμοποιηθεί από όλες τις υγειονομικές βαθμίδες και δομές: ΕΚΑΒ, ΤΕΠ, ΜΕΘ Παίδων - προνοσοκομειακή και ενδονοσοκομειακή φροντίδα - πρωτοβάθμια, δευτεροβάθμια, τριτοβάθμια περίθαλψη υγείας, μιλώντας με αυτό τον τρόπο όλοι την ίδια γλώσσα, «τη

γλώσσα της ανάνηψης».

Για τη βέλτιστη μελέτη του πίνακα οι ερευνητές κατηγοριοποίησαν με αριθμούς σε πέντε τμήματα, αν και όπως σημειώνεται στην πράξη η αντιμετώπιση του επείγοντος είναι ενιαία. Στην Εικόνα 1 βλέπουμε τον πίνακα του πρωτοκόλλου που θα αναλύσουμε.

1.1 Τμήμα 1

Στο τμήμα 1 περιλαμβάνονται τα ατομικά στοιχεία του ασθενούς και συγκεκριμένα:

- το ονοματεπώνυμο,
- το βάρος σε kg/lbs,
- το ύψος σώματος σε cm/inches,
- η ώρα και η ημερομηνία αρχικής προσέγγισης του περιστατικού.

1.2 Τμήμα 2

Στο τμήμα 2, λαμβάνοντας υπόψη τον αλγόριθμο του EPALS, περιλαμβάνονται μνημονικά τα 3B «BBB» και 3S «SSS», δίνοντας τη δυνατότητα να καταγραφεί η πρώτη – άμεση και αδρή αξιολόγηση του περιστατικού που προκύπτει μόνο από την επισκόπηση του ασθενούς. Σε αυτό το τμήμα αν βάσει της αξιολόγησης δεν παρατηρηθεί παθολογικό εύρημα καταγράφουμε OK στο αντίστοιχο πεδίο, ενώ σε αντίθετη περίπτωση καταγράφουμε το παθολογικό εύρημα που βρέθηκε ανά περίπτωση.

Behavior (Συμπεριφορά): Σε αυτό το πεδίο σημειώνουμε τα παθολογικά ευρήματα από την εκτίμηση του μυϊκού τόνου και του επιπέδου συνείδησης ώστε να αξιολογηθεί η αναπνευστική, η κυκλοφορική και η εγκεφαλική λειτουργία.

Body Color (Χρώμα δέρματος): Σε αυτό το πεδίο σημειώνουμε παθολογικά ευρήματα όπως κυάνωση, ωχρότητα, στίγματα δέρματος, εξανθήματα) κατά την αξιολόγηση κυρίως της αιμάτωσης του δέρματος, κάτι που αντικατοπτρίζει την κυκλοφορική λειτουργία του ασθενούς.

Breathing (Αναπνοή): Σε αυτό το πεδίο σημειώνουμε παθολογικά ευρήματα κατά την αξιολόγηση της αναπνευστικής λειτουργίας κάτι που αντικατοπτρίζει τη δυσκολία οξυγόνωσης του ασθενούς.

Safety (Ασφάλεια): Σε αυτό το πεδίο σημειώνουμε οποιονδήποτε παράγοντα μας επηρεάζει στο να προσεγγίσουμε άμεσα τον ασθενή π.χ. ηλεκτρικό ρεύμα. Χρησιμοποιούμε μέτρα ατομικής προστασίας για την προσέγγιση του ασθενούς. Προέχει η ασφάλεια πρώτα του ανανήπτη και έπειτα του ασθενή.

Shout (Κλήση για Βοήθεια): Σε αυτό το πεδίο καταγράφουμε την κλήση προς τις υηρεσίες Επείγουσας Προνοσοκομειακής Φροντίδας ή της ενδονοσοκομειακής ομάδας αναζωογόνησης και την ώρα που αυτή πραγματοποιήθηκε.

Stimulate (Παροχή ερεθίσματος): Σε αυτό το πεδίο γίνενται καταγραφή της ανταπόκρισης ή όχι του ασθενούς σε λεκτικά και/ή απτικά ερεθίσματα.

1.3 Τμήμα 3

Στο τμήμα 3 καταγράφεται η ώρα κατά την οποία πραγματοποιείται η εξέταση των παρακάτω παραμέτρων του πίνακα και της κάθε παρέμβασης με οποιοδήποτε τρόπο. Στην αντίστοιχη στήλη κάτω από την ώρα σημειώνεται το είδος της παρέμβασης πχ. λαρυγγική μάσκα, αν χορηγήθηκε κάποιο φάρμακο και ποιο είναι αυτό, την οδό της προσπέλασης κλπ.

1.4 Τμήμα 4

Το τμήμα 4 απαρτίζεται από 4 επιμέρους τμήματα, λαμβάνοντας και πάλι υπόψη τις βασικές αρχές του EPALS και το μνημονικό κανόνα-αλγόριθμο ABCDE. Τα πεδία που είναι υπογραμμισμένα με κίτρινο χρώμα αναφέρονται στα ζωτικά σημεία αλλά και κάποιες μετρήσιμες τιμές άκρως απαραίτητες για την καλύτερη αντιμετώπιση του ασθενούς: αναπνευστική συχνότητα, κορεσμός οξυγόνου, σφύξεις/λεπτό, αρτηριακή πίεση, glu αίματος και θερμοκρασία, κατανεμημένα στις αντίστοιχες υποκατηγορίες του πίνακα.

1.4.1 ΑΒ. ΑΕΡΑΓΩΓΟΣ/ΑΝΑΠΝΟΗ (ΑΒ)

Χειρισμοί διάνοιξης: αν πραγματοποιήθηκαν βάσει της κατάστασης του αεραγωγού (βατός, επαπειλούμενος, μερικώς ή πλήρως αποφραγμένος) και με ποια διαδικασία έγινε η προσπάθεια διάνοιξης, (στοματοφαρυγγικός / ρινοφαρυγγικός αεραγωγός, διασωλήνωση, λαρυγγική μάσκα ή AMBU) κυκλώνοντας κάθε φορά την επιλογή μας και σημειώνοντας δίπλα την ώρα που πραγματοποιήθηκε.

Αναπνευστική συχνότητα: αριθμός αναπνοών του παιδιού ανά λεπτό. Πολύ σημαντική είναι η δυνατότητα να πραγματοποιούνται πολλές καταγραφές την ώρα για να αξιολογείται η εξέλιξη της αναπνοής στην πορεία του χρόνου (ταχύπνοια – βραδύπνοια - άπνοια).

Ακρόαση/Εργο: καταγραφή των ακροαστικών ευρημάτων κάθε δεδομένη χρονική στιγμή και σύγχρονη εκτίμηση του έργου αναπνοής, το οποίο αν είναι αυξημένο μπορούμε να καταγράψουμε με ποιο τρόπο εκδηλώνεται (εισολκές μεσοπλευρίων διαστημάτων, στέρνου και υποχονδρίων, αναπέταση ρινικών πτερυγίων, κίνηση κεφαλής πάνω-κάτω και σύσπαση των μυών του πρόσθιου θωρακικού τοιχώματος) στο πεδίο της ώρας.

Κορεσμός οξυγόνου: Ώρα καταγραφής και πολλαπλές καταγραφές

Τρόπος χορήγησης οξυγόνου: ώρα έναρξης και τρόπος πχ. ρινική κάνουλα, μάσκα Venturi.

Άλλα φάρμακα: που χρειάστηκαν για τη βελτίωση της αναπνευστικής λειτουργίας του παιδιού. Καταγραφή της ώρας χορήγησης και με συντομογραφία διεθνή μπορούμε να καταγράψουμε στο πεδίο της ώρας ποιο φάρμακο χορηγήθηκε.
Ο εξαιρετικά χρήσιμος μνημοτεχνικός κανόνας για την αξιολόγηση της αναπνοής – το ακρωνύμιο RWTO:

- Αναπνευστική συχνότητα (Respiratory rate),
- Έργο της αναπνοής (Work of breathing),
- Αναπνεόμενος όγκος (Tidal volume),
- Οξυγόνωση (Oxygenation)

επισημαίνεται στο διάγραμμα.

1.4.2 C. ΚΥΚΛΟΦΟΡΙΚΟ (C)

Σφύξεις ανά λεπτό: Ώρα καταγραφής και πολλαπλές καταγραφές

Αρτηριακή πίεση: Ώρα καταγραφής και πολλαπλές καταγραφές

Χρόνος τριχοειδικής επαναπλήρωσης (ΧΤΕ): Ώρα καταγραφής και πολλαπλές καταγραφές για την αξιολόγηση των συστηματικών αγγειακών αντιστάσεων (μαζί με θερμοκρασία και ΔΑΠ).Χρήσιμος για την εκτίμηση της αιμάτωσης και βαθμού καταπληξίας. Άμεση εικόνα της βελτίωσης / επιβάρυνσης του ασθενούς

Προφόρτιο/Εύρος σφυγμού: Εκτίμηση το προφόρτιου και του εύρους σφυγμού του ασθενούς βάσει της κλινικής εξέτασης και των ευρημάτων αυτής (π.χ. σφαγίτιδες, ψηλάφηση ήπατος, κεντρικές και περιφερικές σφίξεις). Χρήσιμη η καταγραφή αδρά των μεταβολών.

Dxt (dextrose): Στενή παρακολούθηση επιπέδων γλυκόζης αίματος (mg/dl), ώρα καταγραφής και δυνατότητα πολλαπλών καταγραφών

IV/IO προσπέλαση: ενδοφλέβια ή ενδοοστική προσπέλαση για τη χορήγηση υγρών ή φαρμάκων. Ώρα καταγραφής και σημείωση οδού προσπέλασης

Υγρά: Σε κάθε καταγεγραμμένη χρονική στιγμή σημείωση αν χρειάστηκε η χορήγηση και το είδος των υγρών

Ινότροπα, άλλα φάρμακα - Αδρεναλίνη (επινεφρίνη), Αδενοσίνη, Αμιοδαρόνη, Λιδοκαΐνη, Ατροπίνη, Ναλοξόνη, Γλυκόζη κλπ.: Σε κάθε καταγεγραμμένη χρονική στιγμή σημείωση αν χρειάστηκε η χορήγηση και το είδος φαρμάκου για την αποκατάσταση της αιμοδυναμικής αστάθειας του ασθενούς. Κυριότερες συντομογραφίες στο κάτω μέρος του πίνακα.

ΗΚΓ/SHOCK: Ώρα καταγραφής του ηλεκτροκαρδιογραφήματος και τύπος ρυθμού (απινιδώσιμος και μη απινιδώσιμος ρυθμός). Καταγραφή ώρας και ενέργειας απινίδωσης (απινίδωση με 4J/kg βάρους σώματος με βάση τις υπάρχουσες ενδείξεις). Ο εξαιρετικά χρήσιμος μνημοτεχνικός κανόνας με το ακρωνύμιο 5P για την αξιολόγηση της κυκλοφορίας:

- Σφυγμό (Pulse) – Καρδιακή Συχνότητα,
- Περιφερική Αιμάτωση (Peripheral Perfusion),
- Εύρος σφυγμού (Pulses Volume),
- Αρτηριακή Πίεση (Blood Pressure)
- Προφόρτιο (Preload)

1.4.3 D. ΝΕΥΡΟΛΟΓΙΚΟ (D)

GCS/ΞυΛΕΔ: Σκορ κλίμακας Γλασκώβης - ώρα καταγραφής και πολλαπλές καταγραφές. Αντίστοιχα και στη χρήση της κλίμακας ΞυΛΕΔ καταγραφή κάθε συγκεκριμένη χρονική στιγμή στο σωστό πεδίο του πίνακα αδρά το επίπεδο συνείδησης του παιδιού βάσει του ακρωνυμίου

- ***Ξύπνιο:*** Φυσιολογική αντίδραση,
- ***Λεκτικά ερεθίσματα:*** Όταν απαντά σε λεκτικά ερεθίσματα,
- ***Επώδυνα ερεθίσματα:*** Όταν αντιδρά μόνο σε Επώδυνα ερεθίσματα,
- ***Δεν Απαντά:*** Όταν Δεν Απαντά σε κανένα ερέθισμα,

είτε **Σκορ της Κλίμακας Γλασκώβης.**

- ***Κόρες:*** σχολιασμός αυτών βάσει κλινικής εξέτασης στο αντίστοιχο πεδίο καταγραφής του πίνακα (μέγεθος, αντίδραση στο φως). Χρήσιμη η καταγραφή αδρά των μεταβολών,

- ***Αυξημένη ενδοκράνια πίεση/χειρισμοί:*** Επί κλινικής υποψίας αυξημένης ενδοκράνιας πίεσης καταγραφή των σημείων αυξημένης ενδοκράνιας πίεσης, της ώρας εντοπισμού τους και των χειρισμών για τη διόρθωση στο αντίστοιχο πεδίο της ώρας,

1.4.4 Ε. ΕΚΘΕΣΗ (Ε)

Θερμοκρασία: Ώρα καταγραφής και πολλαπλές καταγραφές **Αλλεργίες:** Καταγραφή με παύλα (-) αν δεν αναφέρονται οποιεσδήποτε αλλεργίες (π.χ. φάρμακα, τροφές) ενώ αν υπάρχουν σημειώνονται ποιες είναι αυτές **Φάρμακα:** Καταγραφή με παύλα (-) αν δε λαμβάνει συστηματικά φαρμακευτική αγωγή ενώ αν λαμβάνει σημειώνεται ποια είναι αυτή **Τελευταίο Γεύμα:** καταγραφή ακριβούς ώρας τελευταίου γεύματος **Ατομικό αναμνηστικό:** Καταγραφή με παύλα (-) αν δεν αναφέρεται προηγούμενο ατομικό αναμνηστικό ενώ αν αναφέρεται σημειώνεται ποιο είναι αυτό **Περιβάλλον:** καταγραφή της πλήρους έκθεσης του παιδιού λαμβάνοντας υπόψη την αξιοπρέπειά του Για την αξιολόγηση της έκθεσης (διερεύνηση των συνθηκών της νόσου ή του τραύματος) υπάρχει ο μνημοτεχνικός κανόνας με το ακρωνύμιο AMPLE που επισημαίνεται στο διάγραμμα:

- Allergy (Αλλεργίες)
- Medication (Φαρμακευτική Αγωγή)
- Past History (Ατομικό Ιστορικό)
- Last Meal (Τελευταίο Γεύμα)
- Environment/exposure (Περιβάλλον/έκθεση)

1.5 Τμήμα 5

Κάθε επαγγελματίας υγείας όταν καλείται να αντιμετωπίσει ένα επείγον παιδιατρικό περιστατικό θα πρέπει, έχοντας πάρει ένα καλό ιστορικό, να αναζητήσει αν υπάρχουν αναστρέψιμα αίτια βάσει του μνημονικού κανόνα 4H, 4T Πίνακας 1 - Μνημονικός κανόνας 4H,4T. Και αυτό διότι, όπως υποδηλώνει ο όρος, είναι αναστρέψιμα και θα πρέπει να αντιμετωπίζονται άμεσα. Έτσι επισημαίνονται στο διάγραμμα καταγραφής για συνεχή επανεκτίμηση.

4H	4T
Hypoxia (Υποξία)	Tension Pneumothorax (υπό τάση πνευμοθώρκας)
Hypokaleaemia/Hyperkalaemia (Υπο-Υπερκαλιαιμία, γενικότερα μεταβολικές διαταραχές)	Tamponade (Καρδιακός επιπωματισμός)
Hypothermia/Hyperthrmia (Υπο-Υπερθερμία)	Thrombosis (Θρόμβωση)
Hypovolaemia (Υπογκαιμία)	Toxins (Τοξίνες - Διαταραχές από θεραπευτικές παρεμβάσεις)

Συμπερασματικά οι ερευνητές θεωρούν ότι το m-PALS μπορεί να χρησιμεύσει ως ένα απαραίτητο εργαλείο καταγραφής κατά τη διάρκεια αντιμετώπισης του παιδιατρικού επείγοντος. Χρήσιμο και απαραίτητο τόσο στην προνοσοκομειακή όσο και στην ενδοσοκομειακή φροντίδα του μικρού ασθενή. Χρήσιμο τόσο για το Τμήμα Επειγόντων Περιστατικών μιας Πρωτοβάθμιας δομής υγείας ή ενός νοσοκομείου όσο και για τις Μονάδες Εντατικής Θεραπείας που αντιμετωπίζουν καθημερινά περιστατικά που απειλείται η ζωή. Ακόμα θεωρούν ότι μπορεί να παίξει σημαντικό ρόλο στη διαδικασία μεταφοράς και παράδοσης του ασθενούς μετά την αναζωογόνηση, συνοδεύοντας τον

στη μονάδα που θα νοσηλευθεί μετά τη σταθεροποίηση του. Έτσι θα βοηθήσει όλους όσους ασχολούνται με το επείγον παιδιατρικό περιστατικό να «μιλούν την ίδια γλώσσα» η οποία να εκπορεύεται από την κλινική εικόνα του ασθενούς και τη συνεχή επανεκτίμηση του. Η αντιμετώπιση ενός βαρέως πάσχοντος παιδιού είναι μία απαιτητική και στρεσογόνος διαδικασία ακόμα και για τον πιο έμπειρο κλινικό παιδίατρο. Η βαρύτητα της κατάστασης, η πιθανή εμπλοκή αρκετών ανθρώπων στην αντιμετώπιση του περιστατικού και η ταχύτητα που απαιτείται στη λήψη αποφάσεων μπορεί να οδηγήσουν σε παραλείψεις ή λάθη. Η ύπαρξη ενός φύλλου καταγραφής βοηθά στη συστηματική αναζήτηση και καταγραφή των απαραίτητων κλινικών πληροφοριών. Ακόμα υπενθυμίζει την ανάγκη της χρονικής οριοθέτησης των ευρημάτων της κλινικής εξέτασης και των θεραπευτικών παρεμβάσεων. Τέλος κωδικοποιεί τις βασικές αναστρέψιμες αιτίες της σοβαρής διαταραχής της ομοιόστασης ενός πάσχοντος παιδιού.

Κεφάλαιο 2

Χρήση Τεχνολογιών & Τεχνολογικών Εργαλείων

2.1 Εισαγωγή

Για την καλύτερη κατανόηση του κώδικα που θα παρουσιαστεί στο επόμενο κεφάλαιο κρίνεται απαραίτητο να παρουσιαστούν σε αυτό το κεφάλαιο οι εφαρμογές/τεχνολογίες/εργαλεία με την βοήθεια των οποίων αναπτύχθηκε η εφαρμογή mPals.

1. Android Studio
2. Java
3. XML
4. SQL - Αποθήκευση δεδομένων τοπικά. (SQLite Βάση δεδομένων)
5. Βιβλιοθήκη Room Database

2.2 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) που χρησιμοποιείται για τη δημιουργία εφαρμογών για το λειτουργικό σύστημα Android. Αναπτύχθηκε από την Google και επί της ουσίας παρέχει ένα εύχρηστο περιβάλλον για την ανάπτυξη εφαρμογών Android.

Το Android Studio συνδυάζει διάφορα εργαλεία και λειτουργίες που καθιστούν τη διαδικασία ανάπτυξης Android εφαρμογών πιο αποδοτική. Περιλαμβάνει έναν γραφικό σχεδιαστή διεπαφής χρήστη (GUI - Graphical User Interface) που επιτρέπει την οπτική σχεδίαση της διεπαφής των εφαρμογών, ενσωματωμένα εργαλεία για τη διαχείριση των αρχείων και τον έλεγχο των εκδόσεων, καθώς και προηγμένους επεξεργαστές κώδικα που παρέχουν αυτόματη συμπλήρωση κώδικα (auto fill), ανίχνευση σφαλμάτων και παρακολούθηση της απόδοσης της εφαρμογής.

Ένα από τα σημαντικότερα χαρακτηριστικά του Android Studio είναι ο ενσωματωμένος εξομοιωτής Android, ο οποίος επιτρέπει στους προγραμματιστές να δοκιμάζουν τις εφαρμογές τους σε διάφορες συσκευές Android χωρίς την ανάγκη να διαθέτουν τις αντίστοιχες φυσικές συσκευές, εξοικονομώντας έτσι χρόνο αλλά και τα αντίστοιχα κόστη προμήθειας συσκευών.

Επιπλέον, το Android Studio υποστηρίζει την διασύνδεση με άλλες πλατφόρμες και εργαλεία ανάπτυξης, όπως το Firebase για την προσθήκη λειτουργιών όπως αυθεντικοποίηση, αποθήκευση δεδομένων και ειδοποιήσεις στις εφαρμογές, καθώς και το Google Cloud Platform για την ενσωμάτωση προηγμένων υπηρεσιών όπως οι τεχνητή νοημοσύνη και η μηχανική μάθηση.

Εν κατακλείδι θα μπορούσαμε αν πούμε ότι το Android Studio είναι το προτιμώμενο εργαλείο ανάπτυξης για προγραμματιστές που επιθυμούν να δημιουργήσουν υψηλής ποιότητας native εφαρμογές για το λειτουργικό σύστημα Android. Με τη βοήθεια του Android Studio, οι προγραμματιστές μπορούν να αναπτύξουν και να διαχειριστούν εφαρμογές με άνεση και να τις προσαρμόζουν σε διάφορες συσκευές Android, προσφέροντας μια ολοκληρωμένη εμπειρία για τους χρήστες. [3]

2.3 Java

Η Java είναι μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού στον κόσμο, που χρησιμοποιείται ευρέως για την ανάπτυξη λογισμικού. Επίσημα παρουσιάστηκε από την εταιρεία Sun Microsystems (πλέον ενσωματώθηκε στην Oracle Corporation) το 1995, και από τότε έχει αποκτήσει μια ισχυρή κοινότητα προγραμματιστών και υποστηρικτών παγκοσμίως. Η Java προσφέρει πολλά πλεονεκτήματα και είναι σχεδιασμένη να είναι ανεξάρτητη από την πλατφόρμα, ασφαλής και ευέλικτη.

Ένα από τα κύρια χαρακτηριστικά της Java είναι η δυνατότητα της να τρέχει σε διαφορετικές πλατφόρμες χωρίς την ανάγκη αναγραφής ειδικού κώδικα για κάθε μία από αυτές. Αυτό επιτυγχάνεται μέσω της χρήσης της εικονικής μηχανής Java (JVM - Java Virtual Machine), η οποία μεταφράζει τον κώδικα Java σε ένα ενδιάμεσο μορφής bytecode, ο οποίος μπορεί να εκτελεστεί σε οποιαδήποτε πλατφόρμα υποστηρίζει την JVM. Αυτό επιτρέπει στους προγραμματιστές να αναπτύσσουν την εφαρμογή μία φορά και να την εκτελούν σε διάφορες συσκευές και λειτουργικά συστήματα.

Έχει ενσωματωμένους μηχανισμούς ασφάλειας που περιορίζουν την πρόσβαση σε ευαίσθητους πόρους και περιορίζουν τη δυνατότητα εκτέλεσης επικίνδυνων λειτουργιών. Αυτό καθιστά την Java μία επιλογή προγραμματισμού ασφαλούς λογισμικού, ιδιαίτερα όταν κάποιος ασχολείται με διαδικτυακές εφαρμογές ή εφαρμογές που αλληλεπιδρούν με ευαίσθητα δεδομένα.

Η γλώσσα υποστηρίζει αντικειμενοστραφή προγραμματισμό, ο οποίος επιτρέπει την οργάνωση του κώδικα σε αντικείμενα που συνδυάζουν δεδομένα και συναρτήσεις. Αυτό καθιστά την ανάπτυξη και τη συντήρηση του λογισμικού πιο ευαίσθητη και ευέλικτη. Επιπλέον, υπάρχει μια ευρεία γκάμα βιβλιοθηκών και frameworks που είναι διαθέσιμα για την Java, τα οποία διευκολύνουν την ανάπτυξη και παρέχουν πρόσθετες λειτουργίες.

Εν κατακλείδι μπορούμε να πούμε ότι η Java αποτελεί ένα ισχυρό εργαλείο για την ανάπτυξη λογισμικού, με δυνατότητες που την καθιστούν ευέλικτη, ασφαλή και ανεξάρτητη από την πλατφόρμα. Η μεγάλη κοινότητα των προγραμματιστών και η διαθεσιμότητα εκτεταμένων βιβλιοθηκών και frameworks κάνουν την Java μία αξιόπιστη επιλογή για την ανάπτυξη εφαρμογών. Είναι μια γλώσσα που συνεχίζει να εξελίσσεται και να προσαρμόζεται στις ανάγκες του σύγχρονου λογισμικού, και αναμένεται να διατηρήσει τη θέση της ως μία από τις προτιμώμενες γλώσσες προγραμματισμού στο μέλλον. [1]

2.4 SQLite Βάση δεδομένων

Η SQL (Structured Query Language) είναι μία γλώσσα προγραμματισμού που χρησιμοποιείται για τη διαχείριση και αλληλεπίδραση με σχεσιακές βάσεις δεδομένων. Η SQL παρέχει ένα σύνολο εντολών για τη δημιουργία, ανάκτηση, ενημέρωση και διαγραφή δεδομένων από μία βάση δεδομένων. Μία από τις πιο δημοφιλείς υλοποιήσεις της SQL είναι η SQLite, μια ανοικτού κώδικα βάση δεδομένων που προσφέρει αξιόπιστη τοπική αποθήκευση δεδομένων.

2.4.1 Κύρια χαρακτηριστικά της SQL και της SQLite

Η SQL παρέχει ένα ευέλικτο μοντέλο για την οργάνωση και αποθήκευση δεδομένων. Με τη βοήθεια των δομών πίνακα, μπορούμε να δημιουργήσουμε πίνακες για την αποθήκευση δεδομένων με συγκεκριμένη δομή και τύπους δεδομένων. Η SQL προσφέρει επίσης δυνατότητες για τη δημιουργία περίπλοκων ερωτημάτων, τη σύνδεση πινάκων, την ταξινόμηση και την ενημέρωση δεδομένων με βάση συγκεκριμένα κριτήρια.

Η SQLite είναι μία ανοικτού κώδικα βάση δεδομένων που χρησιμοποιείται ευρέως για την τοπική αποθήκευση δεδομένων. Μπορεί να ενσωματωθεί εύκολα σε εφαρμογές και δεν απαιτεί μεγάλες απαιτήσεις σε πόρους. Η SQLite υποστηρίζει πλήρως την SQL και παρέχει ένα πλήρες σύνολο λειτουργιών για τη δημιουργία, ενημέρωση, ανάκτηση και διαγραφή δεδομένων.

2.4.2 Οφέλη της SQL και της SQLite

Η χρήση της SQL και της SQLite για την τοπική αποθήκευση δεδομένων προσφέρει πολλά οφέλη. Καταρχάς, επιτρέπει την αποτελεσματική διαχείριση και οργάνωση δεδομένων, επιτρέποντας γρήγορη πρόσβαση και ανάκτηση πληροφοριών. Επίσης, παρέχει εύκολη αναγνωσιμότητα και κατανόηση των δεδομένων μέσω της χρήσης της SQL.

Η SQLite, ειδικότερα, προσφέρει αξιόπιστη και ασφαλή αποθήκευση δεδομένων σε τοπικό επίπεδο. Οι βάσεις δεδομένων SQLite αποθηκεύονται σε ένα μόνο αρχείο κάτι που μεταξύ άλλων καθιστά εύκολη και την δημιουργία / ανάκτηση αντιγράφων ασφαλείας. Η αποθήκευση δεδομένων σε τοπικό επίπεδο με τη χρήση της SQLite επίσης επιτρέπει την ανεξαρτησία από τη σύνδεση σε απομακρυσμένους διακομιστές, επιτρέποντας την εκτέλεση εφαρμογών ακόμη και χωρίς σύνδεση στο διαδίκτυο.

Η SQL και η SQLite αποτελούν ισχυρά εργαλεία για την αποθήκευση και διαχείριση δεδομένων σε τοπικό επίπεδο. Με τη χρήση της SQL, μπορούμε να οργανώσουμε και να ανακτήσουμε δεδομένα με αποδοτικό τρόπο, ενώ η SQLite προσφέρει μία αξιόπιστη και ελαφριά βάση δεδομένων για την αποθήκευση δεδομένων σε τοπικό επίπεδο. Με τις δυνατότητές τους, η SQL και η

SQLite συμβάλλουν στην ανάπτυξη αξιόπιστων και αποδοτικών εφαρμογών που απαιτούν τοπική αποθήκευση δεδομένων. [2]

2.5 Βιβλιοθήκη Room Database

Η βιβλιοθήκη Room Database της Android, είναι μια προηγμένη τεχνολογία που προσφέρει ισχυρές δυνατότητες αποθήκευσης και διαχείρισης δεδομένων για τις εφαρμογές Android που αναπτύσονται για το οικοσύστημα Android..

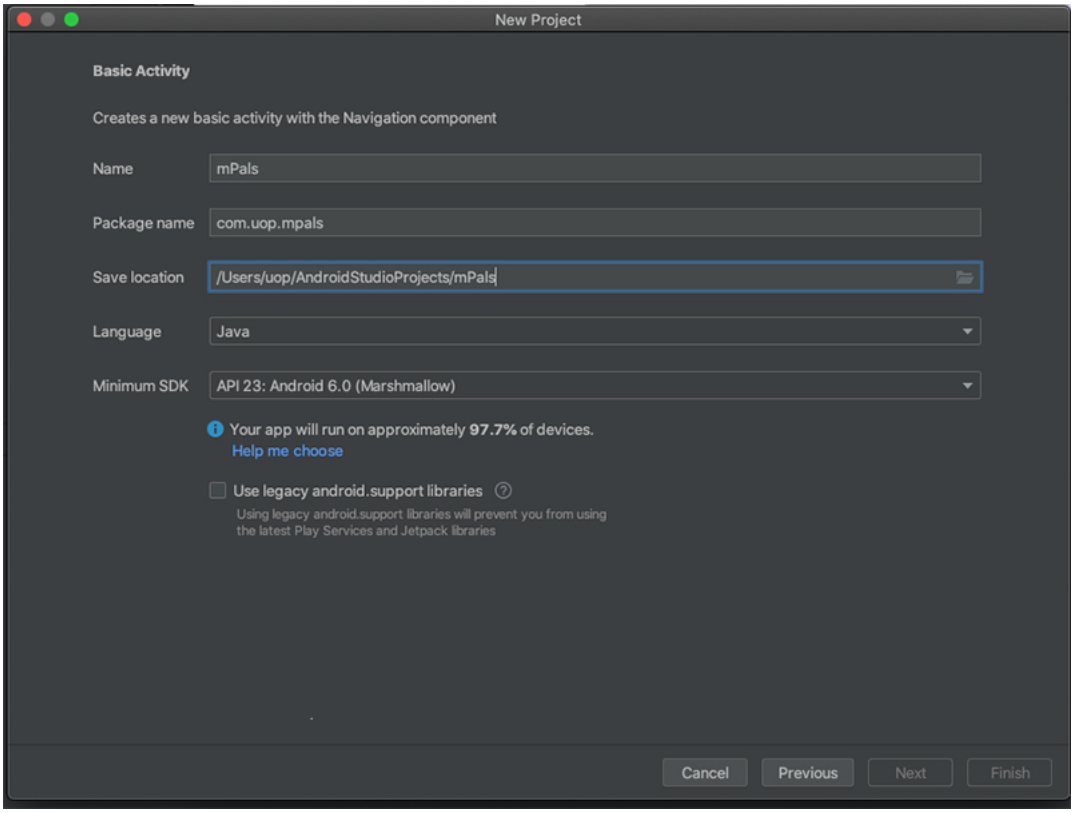
Επί της ουσίας η Room παρέχει μεδόθους που μας επιτρέπουν να αλληλεπιδρούμε με τη βάση δεδομένων μας χρησιμοποιώντας αντικείμενα Java ή Kotlin, αντί για SQL ερωτήματα. Σε αυτή την αλληλεπίδραση συμπεριλαμβάνονται η διαχείριση των σχέσεων μεταξύ των πινάκων, τα αντίγραφα ασφαλείας καθώς και η επαναφορά τους κ.λπ.

Κεφάλαιο 3

Παρουσίαση κώδικα εφαρμογής

3.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει παρουσίαση και ανάλυση του κώδικα της εφαρμογής m-Pals με σχετικά στιγμιότυπα οθόνης από τον κώδικα καθώς και σχόλια επί των στιγμιοτύπων.



Εικόνα 3.1. Δημιουργία εφαρμογής

3.2 Βάση δεδομένων

Η δημιουργία και η διασύνδεση με την βάση δεδομένων πραγματοποιήθηκε με την χρήση της βιβλιοθήκης Room Database της Google. Οι κλάσεις που απαιτήθηκαν είναι οι ακόλουθες:

3.2.1 AppDatabase

Σε αυτήν την κλάση διατηρούμε την βάση δεδομένων. Η κλάση AppDatabase ορίζει τη διαμόρφωση της βάσης δεδομένων και χρησιμεύει ως το κύριο σημείο πρόσβασης της εφαρμογής στα αποθηκευμένα δεδομένα.

```
@Database(entities = {Patient.class, Report.class, BBB.class, SSS.class, Examinations.class, }, version =11 )
public abstract class AppDatabase extends RoomDatabase {
   public abstract PatientsDao taskPatienDao();
   public abstract ReportDao taskReportDao();
   public abstract BBBDao taskBBBDao();
   public abstract SSSDao taskSSSDao();
   public abstract ExaminationDao taskExaminationDao();
}
```

Εικόνα 3.2. Κλάση AppDatabase

3.2.2 DatabaseClient

Η κλάση DatabaseClient είναι το σημείο πρόσβασης για την εκτέλεση όλων των λειτουργιών όπως εισαγωγή, διαγραφή, επιλογή, δημιουργία, ενημέρωση κ.λπ. Εδώ αρχικοποιούμε την βάση δεδομένων και ορίζουμε και το το όνομα της βάσης δεδομένων.

```
public class DatabaseClient {

   private Context mCtx;
   private static DatabaseClient mInstance;

   //Database object
   private AppDatabase appDatabase;

   private DatabaseClient(Context mCtx) {
      this.mCtx = mCtx;

      //creating the app database with Room database builder
      appDatabase = Room.databaseBuilder(mCtx, AppDatabase.class,  name: "MPals").fallbackToDestructiveMigration().build();
   }

   public static synchronized DatabaseClient getInstance(Context mCtx) {
      if (mInstance == null) {
         mInstance = new DatabaseClient(mCtx);
      }
      return mInstance;
   }

   public AppDatabase getAppDatabase() { return appDatabase; }
}
```

Εικόνα 3.3. Κλάση DatabaseClient

3.2.3 Data Access Objects

Στις κλάσεις Data Access Objects ορίζουμε όλες τις λειτουργίες που απαιτούνται για την αλληλεπίδραση με τον πίνακα στην βάση δεδομένων. Μερικές από τις βασικές λειτουργίες είναι SELECT, INSERT, DELETE, UPDATE κ.λπ. Για την εφαρμογή mPals δημιουργήθηκαν τα παρακάτω Data Access Objects:

- PatientDAO
- BBBDAO
- SSSDAO

- ExaminationDAO
- ReportDAO

3.2.4 Entity

Στις κλάσεις Entity ορίζουμε ουσιαστικά τους πίνακες και όλα τα απαραίτητα στοιχεία που θα περιέχει η βάση δεδομένων. Για την εφαρμογή mpals δημιουργήσαμε τις παρακάτω κλάσεις:

Patient

Η κλάση για την διαχείριση των ασθενών

```
@Entity
public class Patient {
    @PrimaryKey(autoGenerate = true)
    public int pid;

    @ColumnInfo(name = "name")
    public String Name;

    @ColumnInfo(name = "dateregistration")
    public String dateregistration;

    @ColumnInfo(name = "weight")
    public int weight;

    @ColumnInfo(name = "height")
    public int height;

    @ColumnInfo(name = "datebirth")
    public String datebirth;

    public int getId() { return pid; }

    public void setId(int pid) { this.pid = pid; }


    public String getName() { return Name; }

    public void setName(String Name) { this.Name = Name; }
    public String getDateregistration() { return dateregistration; }

    public void setDateregistration(String dateregistration) {
        this.dateregistration = dateregistration;
    }
    public int getWeight() { return weight; }

    public void setWeight(int weight) { this.weight = weight; }
    public int getHeight() { return height; }

    public void setHeight(int height) { this.height = height; }
    public String getDateBirth() { return datebirth; }

    public void setDatebirth(String datebirth) { this.datebirth = datebirth; }
```

Εικόνα 3.4. Κλάση Patient Entity

BBB

Η κλάση για την διαχείρηση των δεδομένων της ενότητας BBB

```
@Entity
public class BBB {
 @PrimaryKey(autoGenerate = true)
   public int bid;

   @ColumnInfo(name = "pid")
   public int pid;

   @ColumnInfo(name = "behavior")
   public String behavior;

   @ColumnInfo(name = "bodycolor")
   public String bodycolor;

   @ColumnInfo(name = "breathing")
   public String breathing;


   public int getBid() { return bid; }

   public void setBid(int bid) { this.bid = bid; }


   public int getPid() { return pid; }

   public void setPid(int pid) { this.pid = pid; }


   public String getBehavior() { return behavior; }

   public void setBehavior(String behavior) { this.behavior = behavior; }

   public String getBodycolor() { return bodycolor; }

   public void setBodycolor(String bodycolor) { this.bodycolor = bodycolor; }

   public String getBreathing() { return breathing; }

   public void setBreathing(String breathing) { this.breathing = breathing; }
}
```

Εικόνα 3.5. Κλάση BBB Entity

SSS

Η κλάση για την διαχείρηση των δεδομένων της ενότητας SSS

```java
@Entity
public class SSS {
    @PrimaryKey(autoGenerate = true)
    public int sid;

    @ColumnInfo(name = "pid")
    public int pid;

    @ColumnInfo(name = "safety")
    public String safety;

    @ColumnInfo(name = "shout")
    public String shout;

    @ColumnInfo(name = "stimulate")
    public String stimulate;

    public int getSid() { return sid; }

    public void setSid(int sid) {
        this.sid = sid;
    }


    public int getPid() { return pid; }

    public void setPid(int pid) { this.pid = pid; }


    public String getSafety() { return safety; }

    public void setSafety(String safety) { this.safety = safety; }

    public String getShout() { return shout; }

    public void setShout(String shout) { this.shout = shout; }

    public String getStimulate() { return stimulate; }

    public void setStimulate(String stimulate) { this.stimulate = stimulate; }

}
```

Εικόνα 3.6. Κλάση SSS Entity

Report

Η κλάση για την διαχείρηση των δεδομένων της ενότητας ”ΕΚΘΕΣΗ”

```java
@Entity
public class Report {
    @PrimaryKey(autoGenerate = true)
    public int rpid;

    @ColumnInfo(name = "pid")
    public int pid;

    @ColumnInfo(name = "allergies")
    public String allergies;

    @ColumnInfo(name = "pharmacytreetment")
    public String pharmacytreetment;

    @ColumnInfo(name = "lastmeal")
    public String lastmeal;

    @ColumnInfo(name = "personalhistory")
    public String personalhistory;

    @ColumnInfo(name = "enviroment")
    public String enviroment;

    public int getRpid() { return rpid; }

    public void setRpid(int rpid) { this.rpid = rpid; }

    public int getPid() { return pid; }

    public void setPid(int pid) { this.pid = pid; }

    public String getAllergies() { return allergies; }

    public void setAllergies(String allergies) { this.allergies = allergies; }
    public String getPharmacytreetment() { return pharmacytreetment; }

    public void setPharmacytreetment(String pharmacytreetment) {
        this.pharmacytreetment = pharmacytreetment;
    }

    public String getLastMeal() { return lastmeal; }

    public void setLastmeal(String lastmeal) { this.lastmeal = lastmeal; }

    public String getPersonalhistory() { return personalhistory; }

    public void setPersonalhistory(String personalhistory) {
        this.personalhistory = personalhistory;
    }

    public String getEnviroment() { return enviroment; }

    public void setEnviroment(String enviroment) { this.enviroment = enviroment; }
}
```

Εικόνα 3.7. Κλάση Report Entity

Examinations

Η κλάση για την διαχείρηση των δεδομένων των εξετάσεων

```java
@Entity
public class Examinations {
    @PrimaryKey(autoGenerate = true)
    public int eid;

    @ColumnInfo(name = "pid")
    public int pid;


    @ColumnInfo(name = "examdate")
    public String examdate;

    @ColumnInfo(name = "examtime")
    public String examtime;

    @ColumnInfo(name = "bodycolor")
    public String bodycolor;

    @ColumnInfo(name = "examaer")
    public String examaer;

    @ColumnInfo(name = "examkykl")
    public String examkykl;

    @ColumnInfo(name = "examneyr")
    public String examneyr;

    //AB
    @ColumnInfo(name = "AB")
    public boolean AB;

    // Χειρισμοί διάνοιξης
    @ColumnInfo(name = "examABxeirismoidianoixis")
    public String examABxeirismoidianoixis;
    // Αναπνευστική συχνότητα
    @ColumnInfo(name = "examABanapneystikisixnotita")
    public int examABanapneystikisixnotita;
    // Ακρόαση / Έργο
    @ColumnInfo(name = "examABakroasi")
    public String examABakroasi;
    // Κορεσμός Οξυγόνου
    @ColumnInfo(name = "examABkoresmsooxigonou")
    public int examABkoresmsooxigonou;
    //  Τρόπος χορήγησης Οξυγόνου
    @ColumnInfo(name = "examABtroposxorigisioxigonou")
    public String examABtroposxorigisioxigonou;
    // Άλλα φάρμακα
    @ColumnInfo(name = "examABothermedicine")
    public String examABothermedicine;
```

Εικόνα 3.8. Κλάση Examination Entity [1]

```
//C

@ColumnInfo(name = "C")
public boolean C;

//Σφύξεις ανα λεπτό
@ColumnInfo(name = "examCsfixisanalepto")
public int examCsfixisanalepto;
//Αρτηριακή πίεση
@ColumnInfo(name = "examCartiriakipiesi")
public int examCartiriakipiesi;
//Χρόνος τριχοειδικής επαναπλήρωσης (ΧΤΕ)
@ColumnInfo(name = "examCxronostrixoeidis")
public int examCxronostrixoeidis;
// Προφόρτιο/Εύρος σφυγμού
@ColumnInfo(name = "examCprofortio")
public String examCprofortio;
// Dxt (dextrose)
@ColumnInfo(name = "examCdxt")
public String examCdxt;
// IV/IO προσπέλαση
@ColumnInfo(name = "examCiv")
public String examCiv;
//Υγρά
@ColumnInfo(name = "examCygra")
public String examCygra;
// Άλλα φάρμακα
@ColumnInfo(name = "examCothermedicine")
public String examCothermedicine;
// ΗΚΓ/SHOCK
@ColumnInfo(name = "examChkg")
public String examChkg;
```

Εικόνα 3.9. Κλάση Examination Entity [2]

```
//D.ΝΕΥΡΟΛΟΓΙΚΟ
@ColumnInfo(name = "D")
public boolean D;

//GCS
@ColumnInfo(name = "examDgcs")
public String examDgcs;
//αυρυ
@ColumnInfo(name = "examDanpu")
public String examDanpu;
//Κόρες
@ColumnInfo(name = "examDkores")
public String examDkores;
//Αυξημένη ενδοκράνια πίεση/χειρισμοί
@ColumnInfo(name = "examDendokraniaki")
public String examDendokraniaki;

// Ε.ΕΚΘΕΣΗ
@ColumnInfo(name = "E")
public boolean E;
//θερμοκρασία
@ColumnInfo(name = "examEtemperature")
public String examEtemperature;
```

Εικόνα 3.10. Κλάση Examination Entity [3]

```
///*****************************
 // SET -GET
 //*****************************

  public boolean getAB() { return AB; }

  public void setAB(boolean AB) { this.AB = AB; }

  public boolean getC() { return C; }

  public void setC(boolean C) { this.C = C; }

  public boolean getD() { return D; }

  public void setD(boolean D) { this.D = D; }


  public boolean getE() { return E; }

  public void setE(boolean E) { this.E = E; }

 public int getEid() { return eid; }

 public void setEid(int eid) { this.eid = eid; }


 public int getPid() { return pid; }

 public void setPid(int pid) { this.pid = pid; }

 public String getExamdate() { return examdate; }

 public void setExamdate(String examdate) { this.examdate = examdate; }

 public String getExamtime() { return examtime; }
 }
 public void setExamtime(String examtime) { this.examtime = examtime; }

 public String getBodycolor() { return bodycolor; }

 public void setBodycolor(String bodycolor) { this.bodycolor = bodycolor; }

 public String getExamaer() { return examaer; }

 public void setExamaer(String examaer) { this.examaer = examaer; }

 public String getExamkykl() { return examkykl; }

 public void setExamkykl(String examkykl) { this.examkykl = examkykl; }

 public String getExamneyr() { return examneyr; }

 public void setExamneyr(String examneyr) { this.examneyr = examneyr; }
```

Εικόνα 3.11. Κλάση Examination Entity [4]

```
//AB
// Χειρισμοί διάνοιξης
public String getExamABxeirismoidianoixis() { return examABxeirismoidianoixis; }

public void setExamABxeirismoidianoixis(String examABxeirismoidianoixis) {
    this.examABxeirismoidianoixis = examABxeirismoidianoixis;
}
// Αναπνευστική συχνότητα
public int getExamABanapneystikisixnotita() { return examABanapneystikisixnotita; }

public void setExamABanapneystikisixnotita(int examABanapneystikisixnotita) {
    this.examABanapneystikisixnotita = examABanapneystikisixnotita;
}
// Ακρόαση / Έργο
public String getExamABakroasi() { return examABakroasi; }

public void setExamABakroasi(String examABakroasi) { this.examABakroasi = examABakroasi; }
// Κορεσμός Οξυγόνου
public int getExamABkoresmsooxigonou() { return examABkoresmsooxigonou; }

public void setExamABkoresmsooxigonou(int examABkoresmsooxigonou) {
    this.examABkoresmsooxigonou = examABkoresmsooxigonou;
}

//  Τρόπος χορήγησης Οξυγόνου
public String getExamABtroposxorigisioxigonou() { return examABtroposxorigisioxigonou; }

public void setExamABtroposxorigisioxigonou(String examABtroposxorigisioxigonou) {
    this.examABtroposxorigisioxigonou = examABtroposxorigisioxigonou;
}
// Άλλα φάρμακα
public String getExamABothermedicine() { return examABothermedicine; }

public void setExamABothermedicine(String examABothermedicine) {
    this.examABothermedicine = examABothermedicine;
}

//C
//Σφύξεις ανα λεπτό

public int getExamCsfixisanalepto() { return examCsfixisanalepto; }

public void setExamCsfixisanalepto(int examCsfixisanalepto) {
    this.examCsfixisanalepto = examCsfixisanalepto;
}
//Αρτηριακή πίεση
public int getExamCartiriakipiesi() { return examCartiriakipiesi; }

public void setExamCartiriakipiesi(int examCartiriakipiesi) {
    this.examCartiriakipiesi = examCartiriakipiesi;
}
```

Εικόνα 3.12. Κλάση Examination Entity [5]

```
//D.ΝΕΥΡΟΛΟΓΙΚΟ
//GCS
public String getExamDgcs() { return examDgcs; }

public void setExamDgcs(String examDgcs) { this.examDgcs = examDgcs; }

//αυρυ
public String getExamDanpu() { return examDanpu; }

public void seteExamDanpu(String examDanpu) { this.examDanpu = examDanpu; }

//Κόρες
public String getExamDkores() { return examDkores; }

public void setExamDkores(String examDkores) { this.examDkores = examDkores; }

//Αυξημένη ενδοκράνια πίεση/χειρισμοί
public String getExamDendokraniaki() { return examDendokraniaki; }

public void setExamDendokraniaki(String examDendokraniaki) {
    this.examDendokraniaki = examDendokraniaki;
}

// Ε.ΕΚΘΕΣΗ
//θερμοκρασία
public String getExamEtemperature() { return examEtemperature; }

public void setExamEtemperature(String examEtemperature) {
    this.examEtemperature = examEtemperature;
}
```

Εικόνα 3.13. Κλάση Examination Entity [6]

3.3 ViewModel και MutableLiveData

Για την προσωρινή αποθήκευση των δεδομένων δημιουργήσαμε viewmodel με τα πεδία να είναι MutableLiveData.

```java
public class PatientViewModelnew extends ViewModel {

    /**
     * Live Data Instance
     */
    private MutableLiveData<String> mID = new MutableLiveData<>();
    private MutableLiveData<String> mName = new MutableLiveData<>();
    private MutableLiveData<String> mbirthDate= new MutableLiveData<>();
    private MutableLiveData<String> mregistrationDate = new MutableLiveData<>();
    private MutableLiveData<String> mWeight = new MutableLiveData<>();
    private MutableLiveData<String> mHeight = new MutableLiveData<>();



    public void setID(String id) { mID.setValue(id); }
    public LiveData<String> getID() { return mID; }

    public void setName(String name) { mName.setValue(name); }
    public LiveData<String> getName() { return mName; }

    public void setbirthDate(String birthdate) { mbirthDate.setValue(birthdate); }
    public LiveData<String> getbirthDate() { return mbirthDate; }

    public void setregistrationDate(String registrationDate) { mregistrationDate.setValue(registrationDate);
    }
    public LiveData<String> getregistrationDate() { return mregistrationDate; }

    public void setWeight(String weight) { mWeight.setValue(weight); }
    public LiveData<String> getWeight() { return mWeight; }

    public void setHeight(String height) { mHeight.setValue(height); }
    public LiveData<String> getHeight() { return mHeight; }



    public void initialize(){

        mID = new MutableLiveData<>();
        mName = new MutableLiveData<>();
        mbirthDate= new MutableLiveData<>();
        mregistrationDate = new MutableLiveData<>();
        mWeight = new MutableLiveData<>();
        mHeight = new MutableLiveData<>();

    }
}
```

Εικόνα 3.14. Κλάση Patient ViewModel

```
public class BBBViewModel extends ViewModel {
    private MutableLiveData<String> bID= new MutableLiveData<>();

    private MutableLiveData<String> pID= new MutableLiveData<>();

    private MutableLiveData<String> behaviour = new MutableLiveData<>();

    private MutableLiveData<String> bodycolor= new MutableLiveData<>();

    private MutableLiveData<String> breathing= new MutableLiveData<>();

    public void setID(String id) { bID.setValue(id); }
    public LiveData<String> getID() { return bID; }
    public void setpID(String id) { pID.setValue(id); }
    public LiveData<String> getpID() { return pID; }
    public void setbehaviour(String str) { behaviour.setValue(str); }
    public LiveData<String> getbehaviour() { return behaviour; }
    public void setbodycolor(String str) { bodycolor.setValue(str); }
    public LiveData<String> getbodycolor() { return bodycolor; }
    public void setbreathing(String str) { breathing.setValue(str); }
    public LiveData<String> getbreathing() { return breathing; }

    public void initialize(){

       bID= new MutableLiveData<>();

       pID= new MutableLiveData<>();

       behaviour = new MutableLiveData<>();
       bodycolor= new MutableLiveData<>();
       breathing= new MutableLiveData<>();
    }
}
```

Εικόνα 3.15. Κλάση BBB ViewModel

```
public class SSSViewModel extends ViewModel {

    /**
     * Live Data Instance
     */
    private MutableLiveData<String> sID= new MutableLiveData<>();

    private MutableLiveData<String> pID= new MutableLiveData<>();

    private MutableLiveData<String> safety = new MutableLiveData<>();

    private MutableLiveData<String> shout= new MutableLiveData<>();

    private MutableLiveData<String> stimulate= new MutableLiveData<>();


    public void setID(String id) { sID.setValue(id); }
    public LiveData<String> getID() { return sID; }
    public void setpID(String id) { pID.setValue(id); }
    public LiveData<String> getpID() { return pID; }
    public void setsafety(String str) { safety.setValue(str); }
    public LiveData<String> getsafety() { return safety; }
    public void setshout(String str) { shout.setValue(str); }
    public LiveData<String> getshout() { return shout; }
    public void setstimulate(String str) { stimulate.setValue(str); }
    public LiveData<String> getstimulate() { return stimulate; }

    public void initialize(){

        sID= new MutableLiveData<>();
        pID= new MutableLiveData<>();
        safety = new MutableLiveData<>();
        shout= new MutableLiveData<>();
        stimulate= new MutableLiveData<>();

    }
}
```

Εικόνα 3.16. Κλάση SSS ViewModel

```
public class ReporViewModel extends ViewModel {

    private MutableLiveData<String> rpid= new MutableLiveData<>();

    private MutableLiveData<String> pID= new MutableLiveData<>();

    private MutableLiveData<String> allergies = new MutableLiveData<>();

    private MutableLiveData<String> pharmacytreetment= new MutableLiveData<>();

    private MutableLiveData<String> lastmeal= new MutableLiveData<>();

    private MutableLiveData<String> personalhistory= new MutableLiveData<>();

    private MutableLiveData<String> enviroment= new MutableLiveData<>();

    public void setID(String id) { rpid.setValue(id); }
    public LiveData<String> getID() { return rpid; }

    public void setpID(String id) { pID.setValue(id); }
    public LiveData<String> getpID() { return pID; }

    public void setAllergies(String str) { allergies.setValue(str); }
    public LiveData<String> getAllergies() { return allergies; }

    public void setPharmacytreetment(String str) { pharmacytreetment.setValue(str); }
    public LiveData<String> getPharmacytreetment() { return pharmacytreetment; }

    public void setLastmeal(String str) { lastmeal.setValue(str); }
    public LiveData<String> getLastmeal() { return lastmeal; }

    public void setpersonalhistory(String str) { personalhistory.setValue(str); }
    public LiveData<String> getpersonalhistory() { return personalhistory; }

    public void setenviroment(String str) { enviroment.setValue(str); }
    public LiveData<String> getenviroment() { return enviroment; }

    public void initialize(){

        rpid= new MutableLiveData<>();
        pID= new MutableLiveData<>();
        allergies = new MutableLiveData<>();
        pharmacytreetment= new MutableLiveData<>();
        lastmeal= new MutableLiveData<>();
        personalhistory= new MutableLiveData<>();
        enviroment= new MutableLiveData<>();

    }
}
```

Εικόνα 3.17. Κλάση Report ViewModel

```
public class ExamListViewModel  extends ViewModel {


    private MutableLiveData<List<Examinations>> mExamsList=new MutableLiveData<>();
    private MutableLiveData<Boolean> stateExamFragment=new MutableLiveData<>();


    public MutableLiveData<List<Examinations>> getExamsList() {
        if (mExamsList == null) {
            mExamsList = new MutableLiveData<>();

        }

        return mExamsList;
    }
    public void setExamsList(List<Examinations> exam) {
        if (mExamsList == null) {
            mExamsList = new MutableLiveData<>();

        }
        mExamsList.setValue (exam);
    }

    public MutableLiveData<Boolean> getStateFr() { return stateExamFragment; }
    public void setStateFr(Boolean stateFr) { stateExamFragment.setValue (stateFr); }

    public void initialize(){
        mExamsList=new MutableLiveData<>();
        stateExamFragment=new MutableLiveData<>();
        stateExamFragment.setValue (false);
    }
```

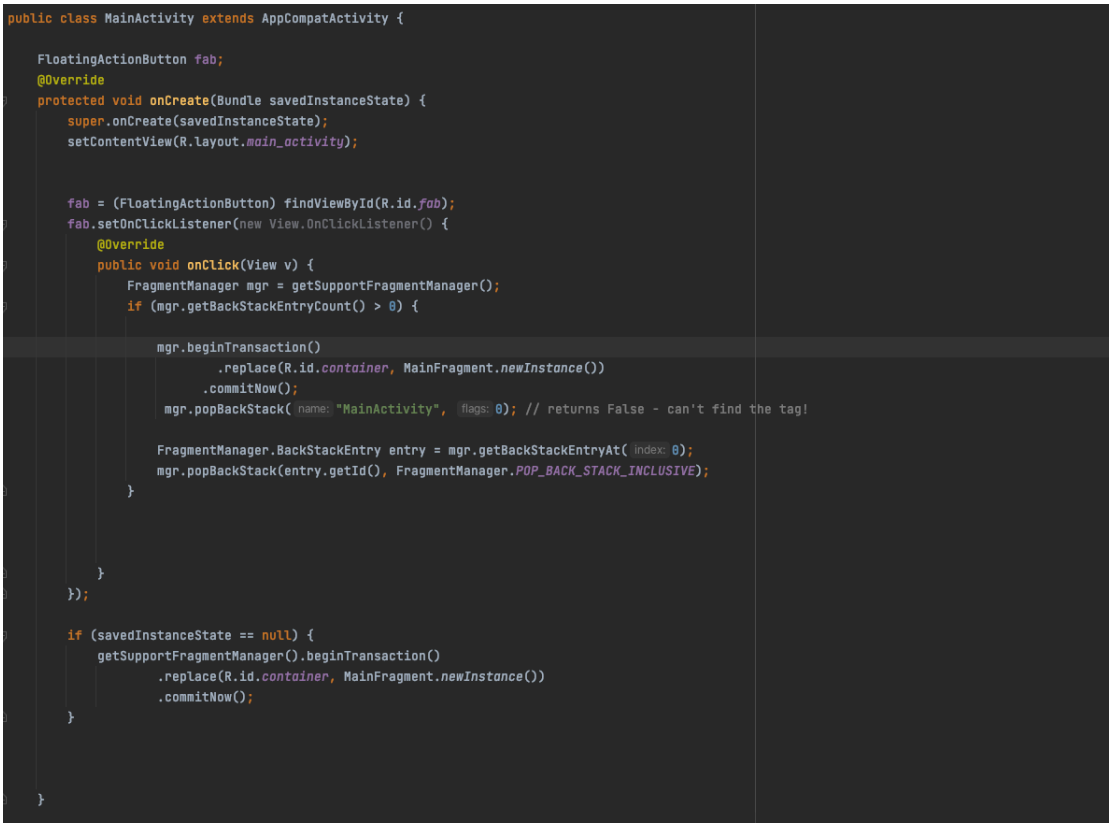
Εικόνα 3.18. Κλάση Examination List ViewModel

```
public class ExamViewModel extends ViewModel {

    private MutableLiveData<String> eID= new MutableLiveData<>();
    private MutableLiveData<String> pID= new MutableLiveData<>();

    private MutableLiveData<Examinations> oneExams= new MutableLiveData<>();

    public void setEID(String id) { eID.setValue(id); }
    public LiveData<String> getEID() { return eID; }
    public void setPID(String id) { pID.setValue(id); }
    public LiveData<String> getPID() { return pID; }

    public void setOneExams(Examinations exam) {

        oneExams.setValue(exam);
    }
    public LiveData<Examinations> getOneExams() { return oneExams; }

    public void initialize(){
      eID= new MutableLiveData<>();
      pID= new MutableLiveData<>();
      oneExams= new MutableLiveData<>();

    }
}
```

Εικόνα 3.19. Κλάση Examination ViewModel

3.4 MainActivity και MainFragment

Η κλάση MainActivity είναι η κεντρική κλάση της εφαρμογής με την οποία ξεκινάει η εφαρμογή όταν κάποιος πατάει το εικονίδιο της εφαρμογής στο tablet.



```java
public class MainActivity extends AppCompatActivity {

    FloatingActionButton fab;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main_activity);


        fab = (FloatingActionButton) findViewById(R.id.fab);
        fab.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                FragmentManager mgr = getSupportFragmentManager();
                if (mgr.getBackStackEntryCount() > 0) {

                    mgr.beginTransaction()
                            .replace(R.id.container, MainFragment.newInstance())
                            .commitNow();
                     mgr.popBackStack( name: "MainActivity",  flags: 0); // returns False - can't find the tag!

                    FragmentManager.BackStackEntry entry = mgr.getBackStackEntryAt( index: 0);
                    mgr.popBackStack(entry.getId(), FragmentManager.POP_BACK_STACK_INCLUSIVE);
                }



            }
        });

        if (savedInstanceState == null) {
            getSupportFragmentManager().beginTransaction()
                    .replace(R.id.container, MainFragment.newInstance())
                    .commitNow();
        }



    }
```

Εικόνα 3.20. Κλάση MainActivity

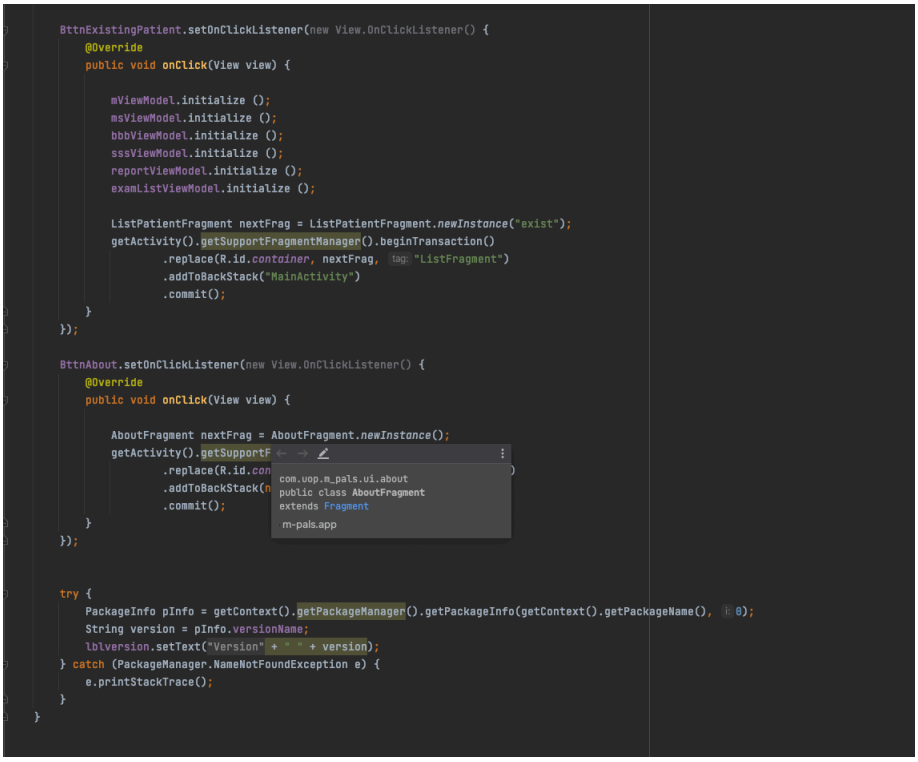
Η κλάση MainFragment είναι η κλάση η οποία φορτώνεται πρώτη και ως μέρος της MainActivity και είναι αυτή που περιέχει τον κώδικα ώστε να λειτουργεί η αρχική οθόνη της εφαρμογής. Με λίγα λόγια συνδέει και διαχειρίζεται το layout της αρχικής οθόνης (main_fragment.xml).

```java
public class MainFragment extends Fragment {

    private Button BttnAddPatient;
    private Button BttnExistingPatient;
    private Button BttnAbout;
    private TextView lblversion;
    private FloatingActionButton fab;
    private com.uop.m_pals.ui.card.patient.PatientViewModel mViewModel; //PATIENT VIEW MODEL
    private PatientViewModelnew msViewModel;
    private BBBViewModel bbbViewModel; //BBB VIEW MODEL
    private SSSViewModel sssViewModel;//SSS VIEW MODEL
    private ReporViewModel reportViewModel;//REPORT VIEW MODEL
    private ExamListViewModel examListViewModel; //LIST OF EXAM


    public static MainFragment newInstance() { return new MainFragment(); }

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /**************************************************
         * CREATE VIEW MODEL
         **************************************************/
        mViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(com.uop.m_pals.ui.card.patient.PatientViewModel.class);
        msViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(PatientViewModelnew.class);
        bbbViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(BBBViewModel.class);
        sssViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(SSSViewModel.class);
        reportViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(ReporViewModel.class);
        examListViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(ExamListViewModel.class);

    }
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container,
                             @Nullable Bundle savedInstanceState) {
        return inflater.inflate(R.layout.main_fragment, container,  attachToRoot: false);
    }
    @Override
    public void onActivityCreated(@Nullable Bundle savedInstanceState) {
        super.onActivityCreated(savedInstanceState);
    }
```

Εικόνα 3.21. Κλάση MainFragment Class [1]

```java
    @Override
    public void onViewCreated(@NonNull View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);
        BttnAddPatient = view.findViewById(R.id.btnAdd);
        BttnExistingPatient = view.findViewById(R.id.btnExist);
        BttnAbout = view.findViewById(R.id.btn_about);
        lblversion= view.findViewById(R.id.lbl_version);
        fab= ((MainActivity)getActivity()).getFloatingActionButton();
        fab.hide();
        BttnAddPatient.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                mViewModel.initialize ();
                msViewModel.initialize ();
                bbbViewModel.initialize ();
                sssViewModel.initialize ();
                reportViewModel.initialize ();
                examListViewModel.initialize ();
                MainTabFragment nextFrag = MainTabFragment.newInstance( pID: -1,  pname: "");
                getActivity().getSupportFragmentManager().beginTransaction()
                        .replace(R.id.container, nextFrag,  tag: "MainCardTabFragment")
                        .addToBackStack("MainActivity")
                        .commit();
            }
        });
```

Εικόνα 3.22. Κλάση MainFragment Class [2]



```
    BttnExistingPatient.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            mViewModel.initialize ();
            msViewModel.initialize ();
            bbbViewModel.initialize ();
            sssViewModel.initialize ();
            reportViewModel.initialize ();
            examListViewModel.initialize ();

            ListPatientFragment nextFrag = ListPatientFragment.newInstance("exist");
            getActivity().getSupportFragmentManager().beginTransaction()
                    .replace(R.id.container, nextFrag, tag: "ListFragment")
                    .addToBackStack("MainActivity")
                    .commit();
        }
    });

    BttnAbout.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {

            AboutFragment nextFrag = AboutFragment.newInstance();
            getActivity().getSupportF
                    .replace(R.id.con
                    .addToBackStack(n
                    .commit();
        }
    });


    try {
        PackageInfo pInfo = getContext().getPackageManager().getPackageInfo(getContext().getPackageName(), 0);
        String version = pInfo.versionName;
        lblversion.setText("Version" + " " + version);
    } catch (PackageManager.NameNotFoundException e) {
        e.printStackTrace();
    }
}
```

Εικόνα 3.23. Κλάση MainFragment Class [3]

3.5 Λίστα ασθενών

Η κλάση ListPatientFragment είναι η κλάση η οποία φορτώνεται όταν ο χρήστης πατήσει στην αρχική οθόνη το κουμπί «EXISTING PATIENT». Εδώ μπορεί να δει ο χρήστης όλους τους ασθενείς που έχουν καταχωρηθεί στην εφαρμογή. Επίσης μπορεί να πραγματοποιήσει αναζήτηση με βάση το ονοματεπώνυμο.

Η κλάση ListPatientAdapter ουσιαστικά βοηθάει στην προβολή των αποτελεσμάτων σε λίστα.

Όταν ο χρήστης πατήσει πάνω σε μία γραμμή της λίστας οδηγείται στην καρτέλα ασθενή.

Τα layout που δημιουργήθηκαν είναι listpatient.xml, listpatientheader.xml, listpatientitem.xml.

```
private void getPatientsAll() {
    class GetTasks extends AsyncTask<Void, Void, List<Patient>> {

        @Override
        protected List<Patient> doInBackground(Void... voids) {
            List<Patient> patientallList = DatabaseClient
                    .getInstance(thiscontext) DatabaseClient
                    .getAppDatabase() AppDatabase
                    .taskPatienDao() PatientsDao
                    .getAll();
            return patientallList;
        }

        @Override
        protected void onPostExecute(List<Patient> tasks) {
            super.onPostExecute(tasks);
            adapter = new ListPatientAdapter (thiscontext, tasks);
            recyclerView.setAdapter(adapter);
            Log.d( tag: "LIST COUNT", String.valueOf(tasks.size()));

        }
    }

    GetTasks gt = new GetTasks();
    gt.execute();
}

@Override
public void onDestroyView() {
    super.onDestroyView ( );

}
```

Εικόνα 3.24. Φόρτωση όλων των ασθενών

```
// Filter Class
public void filterQ(String charText) {
    charText = charText.toLowerCase(Locale.getDefault());
    patientsList.clear();
    if (charText.length() == 0) {
        patientsList.addAll(arraylist);
    } else {
        for (Patient wp : arraylist) {
            if (wp.getName ().toLowerCase(Locale.getDefault()).contains(charText)) {
                patientsList.add(wp);
            }
        }
    }
    notifyDataSetChanged();
}
```

Εικόνα 3.25. Φίλτράρισμα ασθενών βάση ονόματος

3.6 Καρτέλα ασθενούς

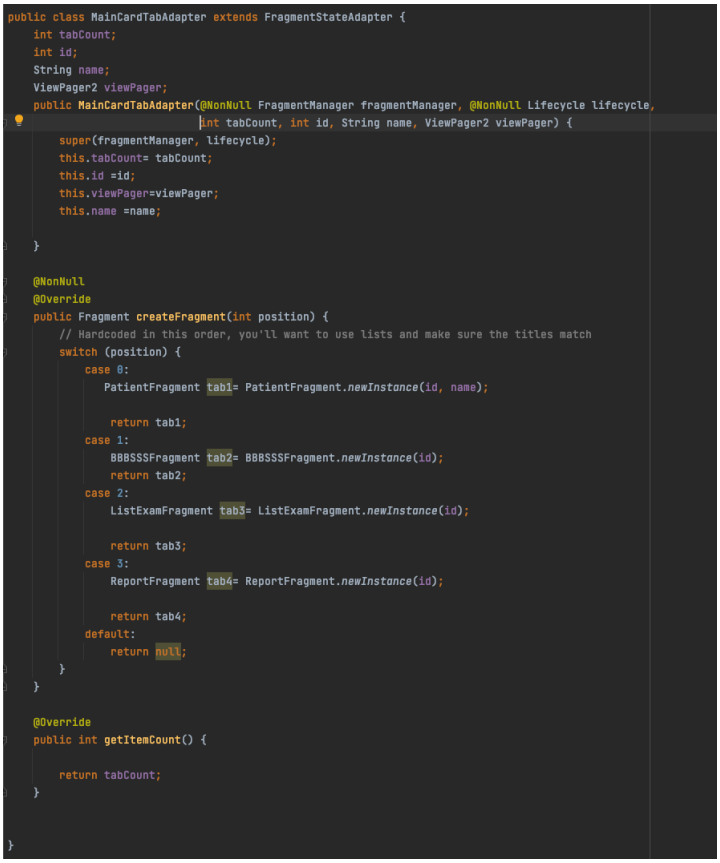
Η καρτέλα του ασθενούς φορτώνεται είτε ο χρήστης πατήσει στην αρχική οθόνη το κουμπί «NEW PATIENT» (ώστε να καταχωρήσει κάποιον νέο) είτε πατήσει σε κάποιο όνομα στην λίστα ασθενών (όπου και φορτώνεται η καρτέλα υφιστάμενου ασθενούς).

Αποτελείται από 4 υποενότητες. Αυτές είναι: η ενότητα PATIENT, η ενότητα BBB & SSS, η ενλότητα EXAMINATION και η ενότητα REPORT. Για τις ανάγκες της καρτέλας αυτής, δημιουργήθηκαν οι ακόλουθες κλάσεις τις οποίες και θα αναλύσουμε αμέσως παρακάτω.

1. MainTabFragment & MainCardTabAdapter 2. PatientFragment 3. BBBSSSFragment 4. ListExamFragment & ExaminationAdapterFragmnent 5. ExamFragment

3.6.1 MainTabFragment & MainCardTabAdapter

Η διαχείριση των 4 TAB γίνεται από τις κλάσεις MainTabFragment και MainCardTabAdapter οι οποίες διαχειρίζονται το layout tab_fragment.xml.



```
public class MainCardTabAdapter extends FragmentStateAdapter {
    int tabCount;
    int id;
    String name;
    ViewPager2 viewPager;
    public MainCardTabAdapter(@NonNull FragmentManager fragmentManager, @NonNull Lifecycle lifecycle,
                              int tabCount, int id, String name, ViewPager2 viewPager) {
        super(fragmentManager, lifecycle);
        this.tabCount= tabCount;
        this.id =id;
        this.viewPager=viewPager;
        this.name =name;

    }

    @NonNull
    @Override
    public Fragment createFragment(int position) {
        // Hardcoded in this order, you'll want to use lists and make sure the titles match
        switch (position) {
            case 0:
               PatientFragment tab1= PatientFragment.newInstance(id, name);

                return tab1;
            case 1:
                BBBSSSFragment tab2= BBBSSSFragment.newInstance(id);
                return tab2;
            case 2:
                ListExamFragment tab3= ListExamFragment.newInstance(id);

                return tab3;
            case 3:
                ReportFragment tab4= ReportFragment.newInstance(id);

                return tab4;
            default:
                return null;
        }
    }

    @Override
    public int getItemCount() {

        return tabCount;
    }


}
```

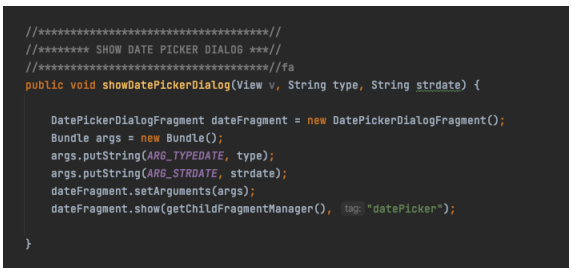
Εικόνα 3.26. Κλάση MainCardTabAdapter - Εναλλαγή καρτελών

3.6.2 PatientFragment

Η κλάση PatientFragment εμφανίζει το layout fragment_patient.xml. Είναι η καρτέλα στην οποία αποθηκεύονται όλα τα στοιχεία του ασθενούς. 1. Registration Date 2. Name 3. Height 4. Weight 5. Birthdate

```
//**************************************
// LOAD PATIENT
//**************************************
private void loadPatient(int id) {

    class loadPatient extends AsyncTask<Void, Void, Void> {

        @Override
        protected Void doInBackground(Void... voids) {


            patientList = DatabaseClient.getInstance (getContext ( )).getAppDatabase ( )
                    .taskPatienDao ( )
                    .loadOneById (id);
            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute (aVoid);

            editTextID.setText (String.valueOf (patientList.get (0).getId ()));
            editTextNAME.setText (patientList.get (0).getName ( ));
            editTextHeight.setText (String.valueOf (patientList.get (0).getHeight ()));
            editTextWeight.setText (String.valueOf (patientList.get (0).getWeight ()));
            datebirthView.setText (patientList.get (0).getDateBirth ());
            dateregistrationView.setText (patientList.get (0).getDateregistration ());
            mViewModel.setmID(String.valueOf (patientList.get (0).getId ()));
            msViewModel.setID(String.valueOf (patientList.get (0).getId ()));

        }

    }

    loadPatient st = new loadPatient ( );
    st.execute();
}
```

Εικόνα 3.27. Κλάση PatientFragment - Αναζήτηση και προβολή δεδομένων βάση του ID του Ασθενούς



```
//************************************//
//******** SHOW DATE PICKER DIALOG ***//
//************************************//fa
public void showDatePickerDialog(View v, String type, String strdate) {

    DatePickerDialogFragment dateFragment = new DatePickerDialogFragment();
    Bundle args = new Bundle();
    args.putString(ARG_TYPEDATE, type);
    args.putString(ARG_STRDATE, strdate);
    dateFragment.setArguments(args);
    dateFragment.show(getChildFragmentManager(), tag: "datePicker");

}
```

Εικόνα 3.28. Προβολή Διαλόγου Επιλογή Ημερομηνίας

3.6.3 BBBSSSFragment

Η κλάση BBBSSSFragment διαχειρίζεται το layout fragment_bbbsss.xml η οποία είναι η καρτέλα στην οποία εισάγονται τα στοιχεία της εξέτασης BBB & SSS.

BBB

1. Behavior 2. BodyColor 3. Breathing

SSS

1. Safety 2. Shout 3. Stimulate

```
//************************************//
//******** LOAD BBB BY ID PATIENT ***//
//************************************//
private void loadBBB(int id) {


    class loadBBB extends AsyncTask<Void, Void, Void> {

        @Override
        protected Void doInBackground(Void... voids) {


            BBBList = DatabaseClient.getInstance(getContext()).getAppDatabase()
                    .taskBBBDao()
                    .loadBBBById(id);
            return null;
        }
        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);
            if (BBBList != null) {
                if (BBBList.size() != 0) {
                    editstoixeiaBehaviorpatient.setText(BBBList.get(0).getBehavior());
                    editstoixeiaBodyColorpatient.setText(BBBList.get(0).getBodycolor());
                    editstoixeiaBreathingpatient.setText(BBBList.get(0).getBreathing());
                    Bid = BBBList.get(0).getBid();
                    bbbViewModel.setID(String.valueOf(BBBList.get(0).getBid()));
                    bbbViewModel.setpID(String.valueOf(id));

                }
            }
        }

    }


    loadBBB st = new loadBBB();

    st.execute();
}
```

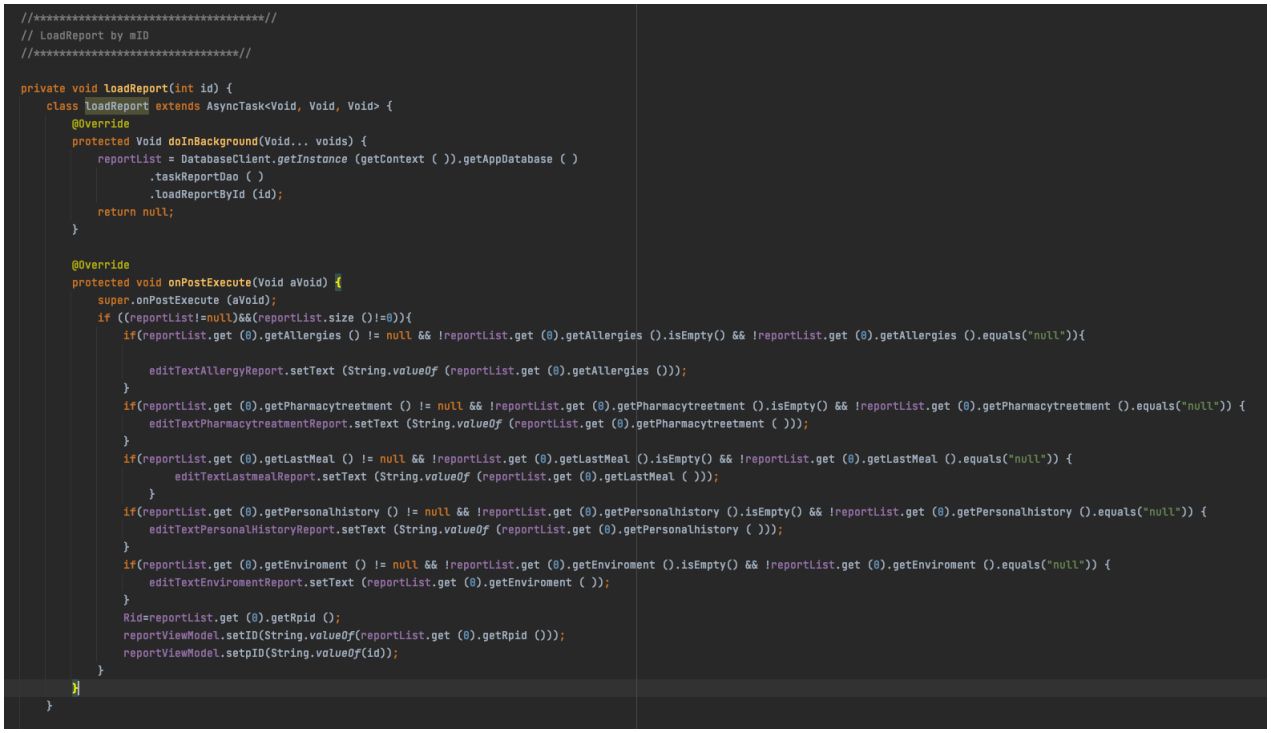
Εικόνα 3.29. Αναζήτηση και προβολή δεδομένων βάση του ID του Ασθενούς (BBB)

```
//************************************//
//******** LOAD SSS BY ID PATIENT ***//
//************************************//
private void loadSSS(int id) {


    class loadSSS extends AsyncTask<Void, Void, Void> {

        @Override
        protected Void doInBackground(Void... voids) {


            SSSList = DatabaseClient.getInstance(getContext()).getAppDatabase()
                    .taskSSSDao()
                    .loadSSSById(id);

            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);
            if (SSSList != null) {
                if (SSSList.size() != 0) {
                    editstoixeiaSafetypatient.setText(SSSList.get(0).getSafety());
                    editstoixeiaShoutpatient.setText(SSSList.get(0).getShout());
                    editstoixeiaStimulatepatient.setText(SSSList.get(0).getStimulate());
                    Sid = SSSList.get(0).getSid();
                    sssViewModel.setID(String.valueOf(SSSList.get(0).getSid()));
                    sssViewModel.setpID(String.valueOf(id));

                }
            }

        }

    }


    loadSSS st = new loadSSS();
    st.execute();
}
```

Εικόνα 3.30. Αναζήτηση και προβολή δεδομένων βάση του ID του Ασθενούς (SSS)

3.6.4 ReportFragment

Η κλάση ReportFragment διαχειρίζεται το layout fragment_report.xml. Είναι η καρτέλα στην οποία εισάγονται τα στοιχεία της εξέτασης Report. 1. Allergy 2. Medication 3. Last meal 4. Personal History 5. Enviroment/exposure



```
//**********************************//
// LoadReport by mID
//**********************************//

private void loadReport(int id) {
    class loadReport extends AsyncTask<Void, Void, Void> {
        @Override
        protected Void doInBackground(Void... voids) {
            reportList = DatabaseClient.getInstance (getContext ( )).getAppDatabase ( )
                    .taskReportDao ( )
                    .loadReportById (id);
            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute (aVoid);
            if ((reportList!=null)&&(reportList.size ()!=0)){
                if(reportList.get (0).getAllergies () != null && !reportList.get (0).getAllergies ().isEmpty() && !reportList.get (0).getAllergies ().equals("null")){

                    editTextAllergyReport.setText (String.valueOf (reportList.get (0).getAllergies ()));
                }
                if(reportList.get (0).getPharmacytreetment () != null && !reportList.get (0).getPharmacytreetment ().isEmpty() && !reportList.get (0).getPharmacytreetment ().equals("null")) {
                    editTextPharmacytreatmentReport.setText (String.valueOf (reportList.get (0).getPharmacytreetment ( )));
                }
                if(reportList.get (0).getLastMeal () != null && !reportList.get (0).getLastMeal ().isEmpty() && !reportList.get (0).getLastMeal ().equals("null")) {
                        editTextLastmealReport.setText (String.valueOf (reportList.get (0).getLastMeal ( )));
                    }
                if(reportList.get (0).getPersonalhistory () != null && !reportList.get (0).getPersonalhistory ().isEmpty() && !reportList.get (0).getPersonalhistory ().equals("null")) {
                    editTextPersonalHistoryReport.setText (String.valueOf (reportList.get (0).getPersonalhistory ( )));
                }
                if(reportList.get (0).getEnviroment () != null && !reportList.get (0).getEnviroment ().isEmpty() && !reportList.get (0).getEnviroment ().equals("null")) {
                    editTextEnvironmentReport.setText (reportList.get (0).getEnviroment ( ));
                }
                Rid=reportList.get (0).getRpid ();
                reportViewModel.setID(String.valueOf(reportList.get (0).getRpid ()));
                reportViewModel.setpID(String.valueOf(id));
            }
        }
    }
```

Εικόνα 3.31. Αναζήτηση και προβολή δεδομένων βάση του ID του Ασθενούς (Report)

3.6.5 ListExamFragment

Η κλάση ListExamFragment διαχειρίζεται το layout fragment_listexam.xml. Είναι η καρτέλα στην οποία εμφανίζονται σε μία λίστα οι καταχωρημένες εξετάσεις. Επίσης σε αυτήν την καρτέλα υπάρχει και το αντίστοιχο κουμπί για την λειτουργία προσθήκη νέας εξέτασης.

```
//************BUTTON NEW EXAMINATION**************************/

addButton.setOnClickListener (new View.OnClickListener ( ) {
    @Override
    public void onClick(View view) {
        ExamFragment AddNewExam = ExamFragment.newInstance(mID,  eid: -1 ,  nString: "new" );
        MainActivity mainActivity = (MainActivity)view.getContext();
        mainActivity.getSupportFragmentManager().beginTransaction()
                .replace(R.id.container, AddNewExam,  tag: "AddNewExam")
                .addToBackStack(null)
                .commit();

    }
});
```

Εικόνα 3.32. Προσθήκη νέας εξέτασης

```
//*********************************************//
//**** LOAD EXAMINATION BY PATIENT ID  *********//
//*********************************************//

private void loadExaminationPatient(int id) {


    class loadExamPatient extends AsyncTask<Void, Void, List<Examinations>> {

        @Override
        protected List<Examinations> doInBackground(Void... voids) {


            examinationList = DatabaseClient.getInstance (getContext ( )).getAppDatabase ( )
                    .taskExaminationDao ( )
                    .loadExaminationsById (id);

            return examinationList;
        }

        @Override
        protected void onPostExecute(List<Examinations> tasks) {
            super.onPostExecute(tasks);

            if (tasks.size ()>0){
                updateData(tasks);

            }else {

            }
        }
    }

    loadExamPatient st = new loadExamPatient ( );
    st.execute();
}
```

Εικόνα 3.33. Αναζήτηση και προβολή εξετάσεων του συγκεκριμένου ασθενούς

```
//*************************************//
//**** SORT BY SECTION LIST  *********//
//*************************************//

private ArrayList sortAndAddSections(ArrayList<ExaminationItemModel> itemList)
{
    ArrayList<ExaminationItemModel> tempList = new ArrayList<>();
    ///SORT ΦΘΙΝΟΥΣΑ ΗΜΕΡΟΜΗΝΙΑ
    Collections.sort(itemList, new Comparator<ExaminationItemModel>() {
        public int compare(ExaminationItemModel o1, ExaminationItemModel o2) {
            if (o1.getDate() == null || o2.getDate() == null)
                return 0;
            SimpleDateFormat sdf = new SimpleDateFormat( pattern: "dd-MM-yyyy");
            Date d1 = null;
            Date d2 = null;
            try {
                d1 = sdf.parse(o1.getDate());
                d2 = sdf.parse(o2.getDate());
            } catch (ParseException e) {
                e.printStackTrace();
            }
            return d2.compareTo(d1);
        }
    });
    //Loops through the list and add a section before each sectioncell start
    String header = "";
    for(int i = 0; i < itemList.size(); i++)
    {
        //If it is the start of a new section we create a new listcell and add it to our array
        if(!(header.equals(itemList.get(i).getDate()))) {
            ExaminationItemModel sectionCell = new ExaminationItemModel( id: 0,  eid: 0,itemList.get(i).getItemTime (), itemAB: false, itemC: false, itemD: false, itemE: false,
                    itemList.get(i).getDate());
            sectionCell.setToSectionHeader();
            tempList.add(sectionCell);
            header = itemList.get(i).getDate();
        }
        tempList.add(itemList.get(i));
    }
    return tempList;
}

private ArrayList<ExaminationItemModel>  getItems( List<Examinations> examinationList){
    ArrayList<ExaminationItemModel> items = new ArrayList<>();
    for (int i=0; i<=examinationList.size()-1;i++){
      if  (examinationList.get(i) !=null){
        items.add(new ExaminationItemModel(examinationList.get(i).getPid (), examinationList.get(i).getEid (),
                examinationList.get(i).getExamtime (), examinationList.get(i).getAB (),examinationList.get(i).getC (),examinationList.get(i).getD (),
                examinationList.get(i).getE (),examinationList.get(i).getExamdate ()));

      }
    }
    ///SORT FROM HOUR
    Collections.sort(items,
            (lhs, rhs) -> lhs.getItemTime ().compareTo(rhs.getItemTime ()));
    return  items;
}
```

Εικόνα 3.34. Ταξινόμηση και ομαδοποίηση των εξετάσεων του ασθενούς βάση ημερομηνίας.

```
//*************************************************//
//********** UPDATE DATA ADAPTER  *************//
//*************************************************//
public void updateData(List<Examinations> tasks) {

    itemsList = sortAndAddSections(getItems (tasks));


    adapter = new ExaminationAdapterFragment(getContext ( ), itemsList, name);
    list.setAdapter(adapter);


}
```

Εικόνα 3.35. Ενημέρωση των δεδομένων της λίστας

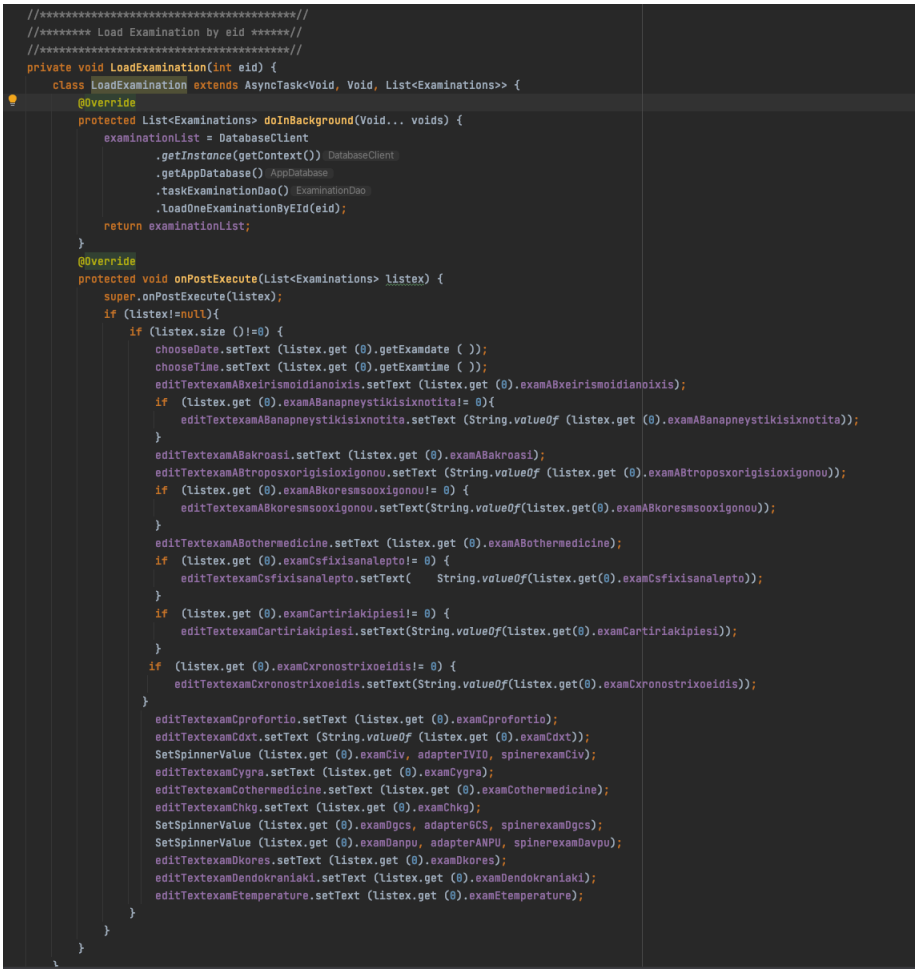
3.6.6 ExaminationAdapterFragment

Η κλάση ExaminationAdapterFragment χρησιμοποιείται για την προβολή της λίστας των εξετάσεων μέσα στην κλάση ListExamFragment. Για την προβολή της λίστας χρησιμοποιούνται τα layout fragment_listexamheader.xml και fragment_listexamitem.xml.

3.6.7 ExamFragment

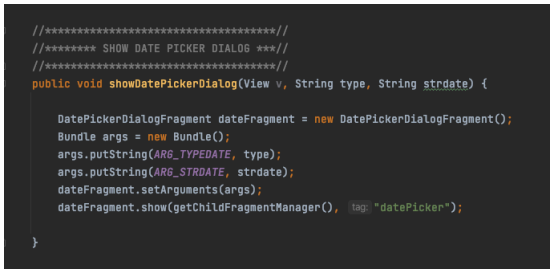
Η κλάση ExamFragment διαχειρίζεται το layout fragment_exam.xml. Είναι η καρτέλα στην οποία εισάγονται τα στοιχεία των εξετάσεων.

- Date
- Time
- AB.AIRWAY/BREATHING
  - Airway management
  - Respiratory rate
  - Auscultation/work
  - Saturation
  - O2 administration
  - Medications
- CIRCULATION
  - BPM/HR
  - Blood pressure
  - CRT
  - Preload/pulse volume
  - Dxt (dextrose)
  - IV/IO access
  - Fluids
  - Medications
  - Ecg/shock
- DISABILITY
  - GCS
  - AVPU
  - Pupillary reaction
  - Intracranial Pressure
- Temperature
  - Temperature



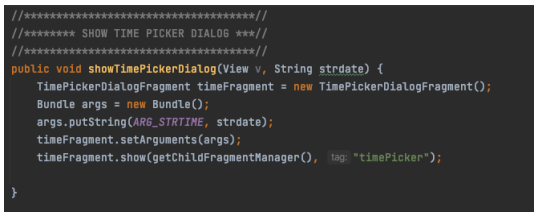
```
//*************************************//
//******** Load Examination by eid ******//
//*************************************//
private void LoadExamination(int eid) {
    class LoadExamination extends AsyncTask<Void, Void, List<Examinations>> {
        @Override
        protected List<Examinations> doInBackground(Void... voids) {
            examinationList = DatabaseClient
                    .getInstance(getContext()) DatabaseClient
                    .getAppDatabase() AppDatabase
                    .taskExaminationDao() ExaminationDao
                    .loadOneExaminationByEId(eid);
            return examinationList;
        }
        @Override
        protected void onPostExecute(List<Examinations> listex) {
            super.onPostExecute(listex);
            if (listex!=null){
                if (listex.size ()!=0) {
                    chooseDate.setText (listex.get (0).getExamdate ( ));
                    chooseTime.setText (listex.get (0).getExamtime ( ));
                    editTextexamABxeirismoidianoixis.setText (listex.get (0).examABxeirismoidianoixis);
                    if  (listex.get (0).examABanapneystikisixnotita!= 0){
                        editTextexamABanapneystikisixnotita.setText (String.valueOf (listex.get (0).examABanapneystikisixnotita));
                    }
                    editTextexamABakroasi.setText (listex.get (0).examABakroasi);
                    editTextexamABtroposxorigisioxigonou.setText (String.valueOf (listex.get (0).examABtroposxorigisioxigonou));
                    if  (listex.get (0).examABkoresmsooxigonou!= 0) {
                        editTextexamABkoresmsooxigonou.setText(String.valueOf(listex.get(0).examABkoresmsooxigonou));
                    }
                    editTextexamABothermedicine.setText (listex.get (0).examABothermedicine);
                    if  (listex.get (0).examCsfixisanalepto!= 0) {
                        editTextexamCsfixisanalepto.setText(    String.valueOf(listex.get(0).examCsfixisanalepto));
                    }
                    if  (listex.get (0).examCartiriakipiesi!= 0) {
                        editTextexamCartiriakipiesi.setText(String.valueOf(listex.get(0).examCartiriakipiesi));
                    }
                   if  (listex.get (0).examCxronostrixoeidis!= 0) {
                       editTextexamCxronostrixoeidis.setText(String.valueOf(listex.get(0).examCxronostrixoeidis));
                   }
                    editTextexamCprofortio.setText (listex.get (0).examCprofortio);
                    editTextexamCdxt.setText (String.valueOf (listex.get (0).examCdxt));
                    SetSpinnerValue (listex.get (0).examCiv, adapterIVIO, spinerexamCiv);
                    editTextexamCygra.setText (listex.get (0).examCygra);
                    editTextexamCothermedicine.setText (listex.get (0).examCothermedicine);
                    editTextexamChkg.setText (listex.get (0).examChkg);
                    SetSpinnerValue (listex.get (0).examDgcs, adapterGCS, spinerexamDgcs);
                    SetSpinnerValue (listex.get (0).examDanpu, adapterANPU, spinerexamDavpu);
                    editTextexamDkores.setText (listex.get (0).examDkores);
                    editTextexamDendokraniaki.setText (listex.get (0).examDendokraniaki);
                    editTextexamEtemperature.setText (listex.get (0).examEtemperature);
                }
            }
        }
    }
```

Εικόνα 3.36. Αναζήτηση και προβολή Εξέτασης βάση του ID του Ασθενή



```
//*************************************//
//******** SHOW DATE PICKER DIALOG ***//
//*************************************//
public void showDatePickerDialog(View v, String type, String strdate) {

    DatePickerDialogFragment dateFragment = new DatePickerDialogFragment();
    Bundle args = new Bundle();
    args.putString(ARG_TYPEDATE, type);
    args.putString(ARG_STRDATE, strdate);
    dateFragment.setArguments(args);
    dateFragment.show(getChildFragmentManager(), tag: "datePicker");

}
```

Εικόνα 3.37. Προβολή Διαλόγου Επιλογή Ημερομηνία

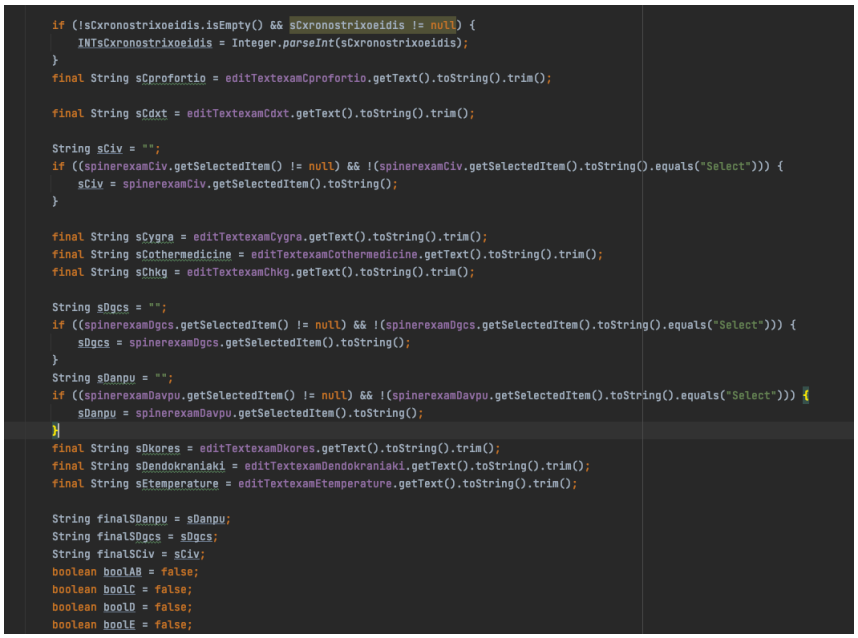


```
//*************************************//
//******** SHOW TIME PICKER DIALOG ***//
//*************************************//
public void showTimePickerDialog(View v, String strdate) {
    TimePickerDialogFragment timeFragment = new TimePickerDialogFragment();
    Bundle args = new Bundle();
    args.putString(ARG_STRTIME, strdate);
    timeFragment.setArguments(args);
    timeFragment.show(getChildFragmentManager(), tag: "timePicker");

}
```

Εικόνα 3.38. Προβολή Διαλόγου Ώρας

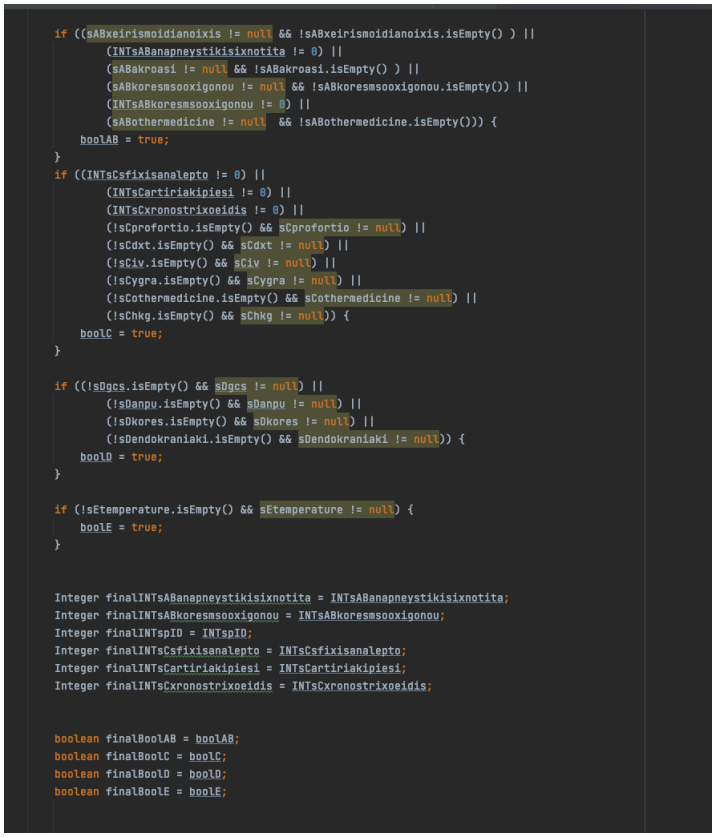
```
//****************************************//
// UPDATE New/Edit Examination by ID/eid    //
//****************************************//

private void saveExamination() {

    final String spID = String.valueOf(mID);
    Integer INTspID = 0;

    if (!spID.isEmpty() && spID != null) {
        INTspID = Integer.parseInt(spID);
    }
    final String sABxeirismoidianoixis = editTextexamABxeirismoidianoixis.getText().toString().trim();
    final String sABanapneystikisixnotita = editTextexamABanapneystikisixnotita.getText().toString().trim();
    Integer INTsABanapneystikisixnotita = 0;
    if (!sABanapneystikisixnotita.isEmpty() && sABanapneystikisixnotita != null) {
        INTsABanapneystikisixnotita = Integer.parseInt(sABanapneystikisixnotita);
    }
    final String sABakroasi = editTextexamABakroasi.getText().toString().trim();
    final String sABtroposxorigisioxigonou = editTextexamABtroposxorigisioxigonou.getText().toString().trim();
    final String sABkoresmsooxigonou = editTextexamABkoresmsooxigonou.getText().toString().trim();

    Integer INTsABkoresmsooxigonou = 0;

    if (!sABkoresmsooxigonou.isEmpty() && sABkoresmsooxigonou != null) {
        INTsABkoresmsooxigonou = Integer.parseInt(sABkoresmsooxigonou);
    }

    final String sABothermedicine = editTextexamABothermedicine.getText().toString().trim();


    final String sCsfixisanalepto = editTextexamCsfixisanalepto.getText().toString().trim();

    Integer INTsCsfixisanalepto = 0;

    if (!sCsfixisanalepto.isEmpty() && sCsfixisanalepto != null) {
        INTsCsfixisanalepto = Integer.parseInt(sCsfixisanalepto);
    }


    final String sCartiriakipiesi = editTextexamCartiriakipiesi.getText().toString().trim();

    Integer INTsCartiriakipiesi = 0;

    if (!sCartiriakipiesi.isEmpty() && sCartiriakipiesi != null) {
        INTsCartiriakipiesi = Integer.parseInt(sCartiriakipiesi);
    }

    final String sCxronostrixoeidis = editTextexamCxronostrixoeidis.getText().toString().trim();

    Integer INTsCxronostrixoeidis = 0;
```

Εικόνα 3.39. Αποθήκευση δεδομένων νέας ή υπάρχουσας εξέτασης [1]



```
    if (!sCxronostrixoeidis.isEmpty() && sCxronostrixoeidis != null) {
        INTsCxronostrixoeidis = Integer.parseInt(sCxronostrixoeidis);
    }
    final String sCprofortio = editTextexamCprofortio.getText().toString().trim();

    final String sCdxt = editTextexamCdxt.getText().toString().trim();

    String sCiv = "";
    if ((spinerexamCiv.getSelectedItem() != null) && !(spinerexamCiv.getSelectedItem().toString().equals("Select"))) {
        sCiv = spinerexamCiv.getSelectedItem().toString();
    }

    final String sCygra = editTextexamCygra.getText().toString().trim();
    final String sCothermedicine = editTextexamCothermedicine.getText().toString().trim();
    final String sChkg = editTextexamChkg.getText().toString().trim();

    String sDgcs = "";
    if ((spinerexamDgcs.getSelectedItem() != null) && !(spinerexamDgcs.getSelectedItem().toString().equals("Select"))) {
        sDgcs = spinerexamDgcs.getSelectedItem().toString();
    }
    String sDanpu = "";
    if ((spinerexamDavpu.getSelectedItem() != null) && !(spinerexamDavpu.getSelectedItem().toString().equals("Select"))) {
        sDanpu = spinerexamDavpu.getSelectedItem().toString();
    }
    final String sDkores = editTextexamDkores.getText().toString().trim();
    final String sDendokraniaki = editTextexamDendokraniaki.getText().toString().trim();
    final String sEtemperature = editTextexamEtemperature.getText().toString().trim();

    String finalSDanpu = sDanpu;
    String finalSDgcs = sDgcs;
    String finalSCiv = sCiv;
    boolean boolAB = false;
    boolean boolC = false;
    boolean boolD = false;
    boolean boolE = false;
```

Εικόνα 3.40. Αποθήκευση δεδομένων νέας ή υπάρχουσας εξέτασης [2]



```
if ((sABxeirismoidianoixis != null && !sABxeirismoidianoixis.isEmpty() ) ||
        (INTsABanapneystikisixnotita != 0) ||
        (sABakroasi != null && !sABakroasi.isEmpty() ) ||
        (sABkoresmsooxigonou != null && !sABkoresmsooxigonou.isEmpty()) ||
        (INTsABkoresmsooxigonou != 0) ||
        (sABothermedicine != null  && !sABothermedicine.isEmpty())) {
    boolAB = true;
}
if ((INTsCsfixisanalepto != 0) ||
        (INTsCartiriakipiesi != 0) ||
        (INTsCxronostrixoeidis != 0) ||
        (!sCprofortio.isEmpty() && sCprofortio != null) ||
        (!sCdxt.isEmpty() && sCdxt != null) ||
        (!sCiv.isEmpty() && sCiv != null) ||
        (!sCygra.isEmpty() && sCygra != null) ||
        (!sCothermedicine.isEmpty() && sCothermedicine != null) ||
        (!sChkg.isEmpty() && sChkg != null)) {
    boolC = true;
}

if ((!sDgcs.isEmpty() && sDgcs != null) ||
        (!sDanpu.isEmpty() && sDanpu != null) ||
        (!sDkores.isEmpty() && sDkores != null) ||
        (!sDendokraniaki.isEmpty() && sDendokraniaki != null)) {
    boolD = true;
}

if (!sEtemperature.isEmpty() && sEtemperature != null) {
    boolE = true;
}


Integer finalINTsABanapneystikisixnotita = INTsABanapneystikisixnotita;
Integer finalINTsABkoresmsooxigonou = INTsABkoresmsooxigonou;
Integer finalINTspID = INTspID;
Integer finalINTsCsfixisanalepto = INTsCsfixisanalepto;
Integer finalINTsCartiriakipiesi = INTsCartiriakipiesi;
Integer finalINTsCxronostrixoeidis = INTsCxronostrixoeidis;


boolean finalBoolAB = boolAB;
boolean finalBoolC = boolC;
boolean finalBoolD = boolD;
boolean finalBoolE = boolE;
```

Εικόνα 3.41. Αποθήκευση δεδομένων νέας ή υπάρχουσας εξέτασης [3]

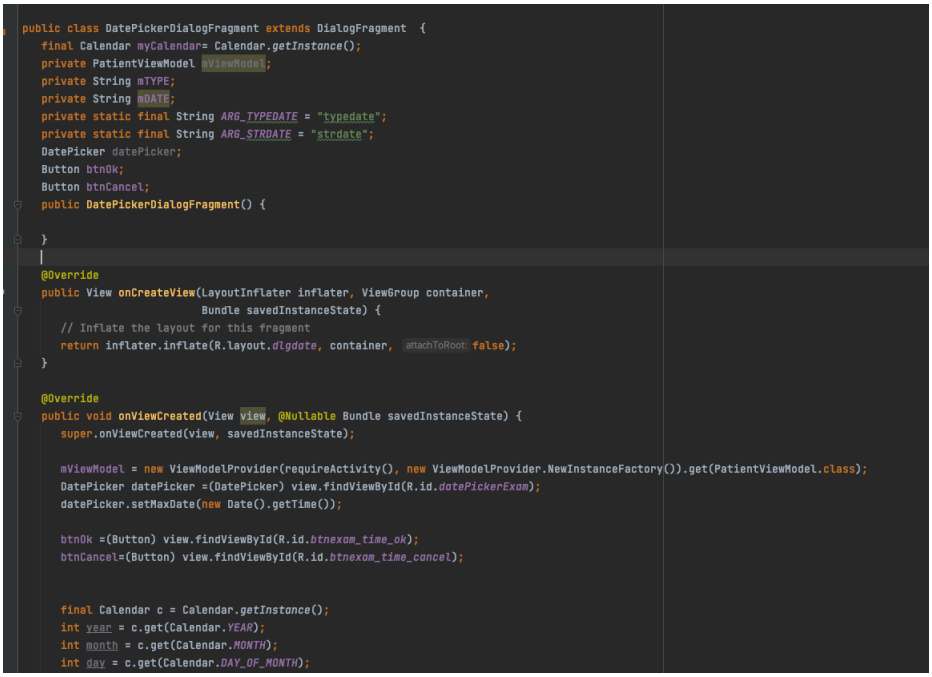
```
class saveExamination extends AsyncTask<Void, Void, Void> {

    @Override
    protected Void doInBackground(Void... voids) {

        //creating a task
        Examinations examinations = new Examinations ( );
        examinations.setPid (finalINTspID);
        examinations.setExamdate (chooseDate.getText ( ).toString ( ));
        examinations.setExamtime (chooseTime.getText ( ).toString ( ));
        //******** AB
        examinations.setAB (finalBoolAB);
        examinations.setExamABxeirismoidianoixis (sABxeirismoidianoixis);
        examinations.setExamABanapneystikisixnotita (finalINTsABanapneystikisixnotita);
        examinations.setExamABakroasi (sABakroasi);
        examinations.setExamABtroposxorigisioxigonou (sABtroposxorigisioxigonou);
        examinations.setExamABkoresmsooxigonou (finalINTsABkoresmsooxigonou);
        examinations.setExamABothermedicine (sABothermedicine);

        //******** C
        examinations.setC (finalBoolC);
        examinations.setExamCsfixisanalepto (finalINTsCsfixisanalepto);
        examinations.setExamCartiriakipiesi (finalINTsCartiriakipiesi);
        examinations.setExamCxronostrixoeidis (finalINTsCxronostrixoeidis);
        examinations.setExamCprofortio (sCprofortio);
        examinations.setExamCdxt (sCdxt);
        examinations.setExamCiv (finalSCiv);
        examinations.setExamCygra (sCygra);
        examinations.setExamCothermedicine (sCothermedicine);
        examinations.setExamChkg (sChkg);
        //******** D
        examinations.setD (finalBoolD);
        examinations.setExamDgcs (finalSDgcs);
        examinations.seteExamDanpu (finalSDanpu);
        examinations.setExamDkores (sDkores);
        examinations.setExamDendokraniaki (sDendokraniaki);
        //******** E
        examinations.setE (finalBoolE);
        examinations.setExamEtemperature (sEtemperature);

        if (eID != -1 || insertId != -1) {
            examinations.setEid (eID);
            DatabaseClient.getInstance (getContext ( )).getAppDatabase ( )
                    .taskExaminationDao ( )
                    .updateExaminations (examinations);

        } else {
            insertId = DatabaseClient.getInstance(getContext()).getAppDatabase()
                 .taskExaminationDao()
                .insertExaminations(examinations);
        }
        eID = Integer.parseInt (String.valueOf (insertId));
        return null;
    }
```

Εικόνα 3.42. Αποθήκευση δεδομένων νέας ή υπάρχουσας εξέτασης [4]

3.7 Γενικές κλάσεις

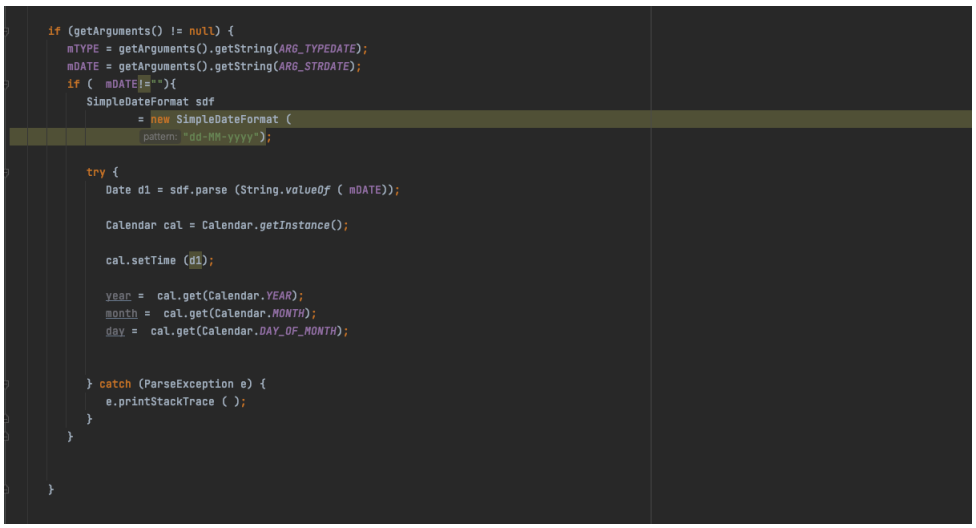
3.7.1 DatePickerDialogFragment

Η κλάση DatePickerDialogFragment εμφανίζει το παράθυρο επιλογής ημερομηνίας στα πεδία που χρειάζεται ο χρήστης να ορίσει ή να αλλάξει ημερομηνία και χρησιμοποιεί το layout dlgdate.xml.



```
public class DatePickerDialogFragment extends DialogFragment  {
    final Calendar myCalendar= Calendar.getInstance();
    private PatientViewModel mViewModel;
    private String mTYPE;
    private String mDATE;
    private static final String ARG_TYPEDATE = "typedate";
    private static final String ARG_STRDATE = "strdate";
    DatePicker datePicker;
    Button btnOk;
    Button btnCancel;
    public DatePickerDialogFragment() {

    }

    @Override
    public View onCreateView(LayoutInflater inflater, ViewGroup container,
                             Bundle savedInstanceState) {
        // Inflate the layout for this fragment
        return inflater.inflate(R.layout.dlgdate, container,  attachToRoot: false);
    }

    @Override
    public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
        super.onViewCreated(view, savedInstanceState);

        mViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(PatientViewModel.class);
        DatePicker datePicker =(DatePicker) view.findViewById(R.id.datePickerExam);
        datePicker.setMaxDate(new Date().getTime());

        btnOk =(Button) view.findViewById(R.id.btnexam_time_ok);
        btnCancel=(Button) view.findViewById(R.id.btnexam_time_cancel);


        final Calendar c = Calendar.getInstance();
        int year = c.get(Calendar.YEAR);
        int month = c.get(Calendar.MONTH);
        int day = c.get(Calendar.DAY_OF_MONTH);
```

Εικόνα 3.43. Κλάση DatePickerDialogFragment - Επιλογέας ημερομηνίας [1]



```
    if (getArguments() != null) {
        mTYPE = getArguments().getString(ARG_TYPEDATE);
        mDATE = getArguments().getString(ARG_STRDATE);
        if (  mDATE!=""){
            SimpleDateFormat sdf
                    = new SimpleDateFormat (
                      pattern: "dd-MM-yyyy");

            try {
                Date d1 = sdf.parse (String.valueOf ( mDATE));

                Calendar cal = Calendar.getInstance();

                cal.setTime (d1);

                year =  cal.get(Calendar.YEAR);
                month =  cal.get(Calendar.MONTH);
                day =  cal.get(Calendar.DAY_OF_MONTH);


            } catch (ParseException e) {
                e.printStackTrace ( );
            }
        }


    }
```

Εικόνα 3.44. Κλάση DatePickerDialogFragment - Επιλογέας ημερομηνίας [2]

```
    btnOk.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) {
            int year=0;
            int month=0;
            int day=0;

            year =  datePicker.getYear();
            month = datePicker.getMonth();
            day =   datePicker.getDayOfMonth();

            myCalendar.set(Calendar.YEAR, year);
            myCalendar.set(Calendar.MONTH,month);
            myCalendar.set(Calendar.DAY_OF_MONTH,day);


            if (mTYPE.equals("datereg")){
                PatientViewModel.txtupdateRegistrationDate(myCalendar);
            }
            if (mTYPE.equals("datebirth")){
                PatientViewModel.txtupdateBirthDate(myCalendar);
            }
            if (mTYPE.equals("")) {
                PatientViewModel.txtupdateExamDate (myCalendar);
            }

            dismiss();

        }
    });

    btnCancel.setOnClickListener(new View.OnClickListener(){
        @Override
        public void onClick(View view) { dismiss(); }
    });

}
}
```

Εικόνα 3.45. Κλάση DatePickerDialogFragment - Επιλογέας ημερομηνίας [3]

3.7.2 TimePickerDialogFragment

Η κλάση TimePickerDialogFragment εμφανίζει το παράθυρο επιλογής ώρας στο πεδίο που χρειάζεται ο χρήστης να ορίσει ή να αλλάξει την ώρα εξέτασης και χρησιμοποιεί το layout dlgtime.xml.

```
public class TimePickerDialogFragment extends DialogFragment  {
   final Calendar myCalendar= Calendar.getInstance();
   private PatientViewModel mViewModel;
   private String mTIME;
   private static final String ARG_STRTIME = "strtime";
   TimePicker timePicker;
   Button btnOk;
   Button btnCancel;
   public TimePickerDialogFragment() {
   }
   @Override
   public View onCreateView(LayoutInflater inflater, ViewGroup container,
                            Bundle savedInstanceState) {
      // Inflate the layout for this fragment
      return inflater.inflate(R.layout.dlgtime, container,  attachToRoot: false);
   }
   @Override
   public void onViewCreated(View view, @Nullable Bundle savedInstanceState) {
      super.onViewCreated(view, savedInstanceState);
      mViewModel = new ViewModelProvider(requireActivity(), new ViewModelProvider.NewInstanceFactory()).get(PatientViewModel.class);
      TimePicker timePicker =(TimePicker) view.findViewById(R.id.timePickerExam);
      btnOk =(Button) view.findViewById(R.id.btnexam_time_ok);
      btnCancel=(Button) view.findViewById(R.id.btnexam_time_cancel);
      ////***** TIMER *****
      int currentHour=0;
      int  currentMinute=0;

      if (savedInstanceState != null) {
         mTIME = getArguments().getString(ARG_STRTIME);
      }
      mTIME = getArguments().getString(ARG_STRTIME);

      if ((mTIME != null) && (mTIME.length() != 0)) {

         SimpleDateFormat sdf = new SimpleDateFormat ( pattern: "HH:mm");
         try {
            Date t1 = sdf.parse (String.valueOf ( mTIME));
            myCalendar.setTime(t1);
            currentHour = myCalendar.get(Calendar.HOUR_OF_DAY);
            currentMinute = myCalendar.get(Calendar.MINUTE);

         } catch (ParseException e) {
            e.printStackTrace();
         }
      } else {
         currentHour = myCalendar.get(Calendar.HOUR_OF_DAY);
         currentMinute = myCalendar.get(Calendar.MINUTE);
      }
      if (Build.VERSION.SDK_INT >= 23) {
         timePicker.setHour(currentHour);
         timePicker.setMinute(currentMinute);
      }
      else {
         timePicker.setCurrentHour(currentHour);
         timePicker.setCurrentMinute(currentMinute);
      }
```

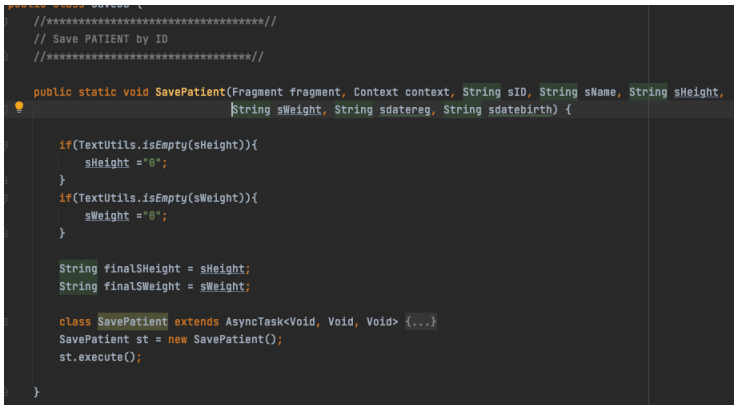
Εικόνα 3.46. Κλάση TimePickerDialogFragment - Επιλογέας ώρας [1]

```
      btnOk.setOnClickListener(new View.OnClickListener(){
         @Override
         public void onClick(View view) {
            int hourOfDay=0;
            int minute=0; ;

            if (Build.VERSION.SDK_INT >= 23) {
               hourOfDay= timePicker.getHour();
               minute=timePicker.getMinute();
            }
            else {
               hourOfDay= timePicker.getCurrentHour();
               minute=timePicker.getCurrentMinute();
            }

            myCalendar.set(Calendar.HOUR_OF_DAY, hourOfDay);
            myCalendar.set(Calendar.MINUTE, minute);

            PatientViewModel.txtupdateExamTime (myCalendar);

            dismiss();

         }
      });
      btnCancel.setOnClickListener(new View.OnClickListener(){
         @Override
         public void onClick(View view) { dismiss(); }
      });

   }
}
```

Εικόνα 3.47. Κλάση TimePickerDialogFragment - Επιλογέας ώρας [2]

3.7.3 SaveDB

Σε αυτή την κλάση υπάρχουν οι συναρτήσεις για την αποθήκευση των δεδομένων στην βάση δεδομένων.



```
//**********************************//
// Save PATIENT by ID
//**********************************//

public static void SavePatient(Fragment fragment, Context context, String sID, String sName, String sHeight,
                               String sWeight, String sdatereg, String sdatebirth) {

    if(TextUtils.isEmpty(sHeight)){
        sHeight ="0";
    }
    if(TextUtils.isEmpty(sWeight)){
        sWeight ="0";
    }

    String finalSHeight = sHeight;
    String finalSWeight = sWeight;

    class SavePatient extends AsyncTask<Void, Void, Void> {...}
    SavePatient st = new SavePatient();
    st.execute();

}
```

Εικόνα 3.48. Κλάση SaveDB - Αποθήκευση Ασθενούς

```
public static void SaveBBBSSS(Fragment fragment, Context context, String sBID, String sSID, String spID,
                              String sBehavior, String sBodyColor, String sBreathing,
                              String sSafety, String sShout, String sStimulate) {

    class SaveBBBSSS extends AsyncTask<Void, Void, Void> {
        Boolean  updated =false;
        long insertIdbbb;
        long insertIdccc;
        @Override
        protected Void doInBackground(Void... voids) {

            //creating a task
            BBB bbb = new BBB ( );
            bbb.setPid (Integer.parseInt (spID));
            bbb.setBehavior (sBehavior);
            bbb.setBodycolor (sBodyColor);
            bbb.setBreathing (sBreathing);

            SSS sss = new SSS ( );
            sss.setPid (Integer.parseInt (spID));
            sss.setSafety (sSafety);
            sss.setShout (sShout);
            sss.setStimulate (sStimulate);

            if (sBID== null|| sBID.isEmpty ( ) ) {
            insertIdbbb=   DatabaseClient.getInstance (context).getAppDatabase ( )
                        .taskBBBDao ( )
                        .insertBBB (bbb);
            } else {
                bbb.setBid (Integer.parseInt(sBID));
                DatabaseClient.getInstance (context).getAppDatabase ( )
                        .taskBBBDao ( )
                        .updateBBB (bbb);
                insertIdbbb=Integer.parseInt(sBID);
            }
            if (sSID== null || sSID.isEmpty ( )  ) {
                insertIdccc=  DatabaseClient.getInstance (context).getAppDatabase ( )
                        .taskSSSDao ( )
                        .insertSSS (sss);
            } else {
                sss.setSid (Integer.parseInt(sSID));
                DatabaseClient.getInstance (context).getAppDatabase ( )
                        .taskSSSDao ( )
                        .updateSSS (sss);
                insertIdccc=Integer.parseInt(sSID);

            }
```

Εικόνα 3.49. Κλάση SaveDB - Αποθήκευση BBB-SSS [1]

```
            updated = true;

            return null;
        }

        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute (aVoid);

            if(fragment instanceof MainTabFragment){
                ((MainTabFragment) fragment).onPlaceInsertedBBB(insertIdbbb);
                ((MainTabFragment) fragment).onPlaceInsertedSSS(insertIdccc);
            }
            if (updated == true) {
                updated = false;
            }
        }
    }

    SaveBBBSSS st = new SaveBBBSSS ( );
    st.execute ( );
}
```

Εικόνα 3.50. Κλάση SaveDB - Αποθήκευση BBB-SSS [2]

```
//***********************************//
// Save REPORT by ID
//***********************************//

public static void SaveReport(Fragment fragment, Context context, String sRID, String spID,
                              String sAllergyReport, String sPharmacytreatmentReport, String sLastmealReport,
                              String sAllergyPersonalHistoryReport, String sEnviromentReport) {

    class SaveReport extends AsyncTask<Void, Void, Void> {
        long insertIdreport;
        @Override
        protected Void doInBackground(Void... voids) {

            Report report = new Report();

            report.setPid (Integer.parseInt (spID));
            report.setAllergies (sAllergyReport);
            report.setPharmacytreetment (sPharmacytreatmentReport);
            report.setLastmeal (sLastmealReport);
            report.setPersonalhistory (sAllergyPersonalHistoryReport);
            report.setEnviroment (sEnviromentReport);

            if ((sRID==null)||sRID.isEmpty ())
            {
                insertIdreport = DatabaseClient.getInstance(context).getAppDatabase()
                        .taskReportDao()
                        .insertReport(report);
            }
            else{
                insertIdreport =Integer.parseInt(sRID);
                report.setRpid (Integer.parseInt(sRID));
                DatabaseClient.getInstance(context).getAppDatabase()
                        .taskReportDao()
                        .updateReport(report);
            }
            return null;
        }
        @Override
        protected void onPostExecute(Void aVoid) {
            super.onPostExecute(aVoid);
            if(fragment instanceof MainTabFragment){
                  ((MainTabFragment) fragment).onPlaceInsertedReport (insertIdreport);

            }

        }
    }

    SaveReport st = new SaveReport();
    st.execute();
}
```

Εικόνα 3.51. Κλάση SaveDB - Αποθήκευση Report

Κεφάλαιο 4

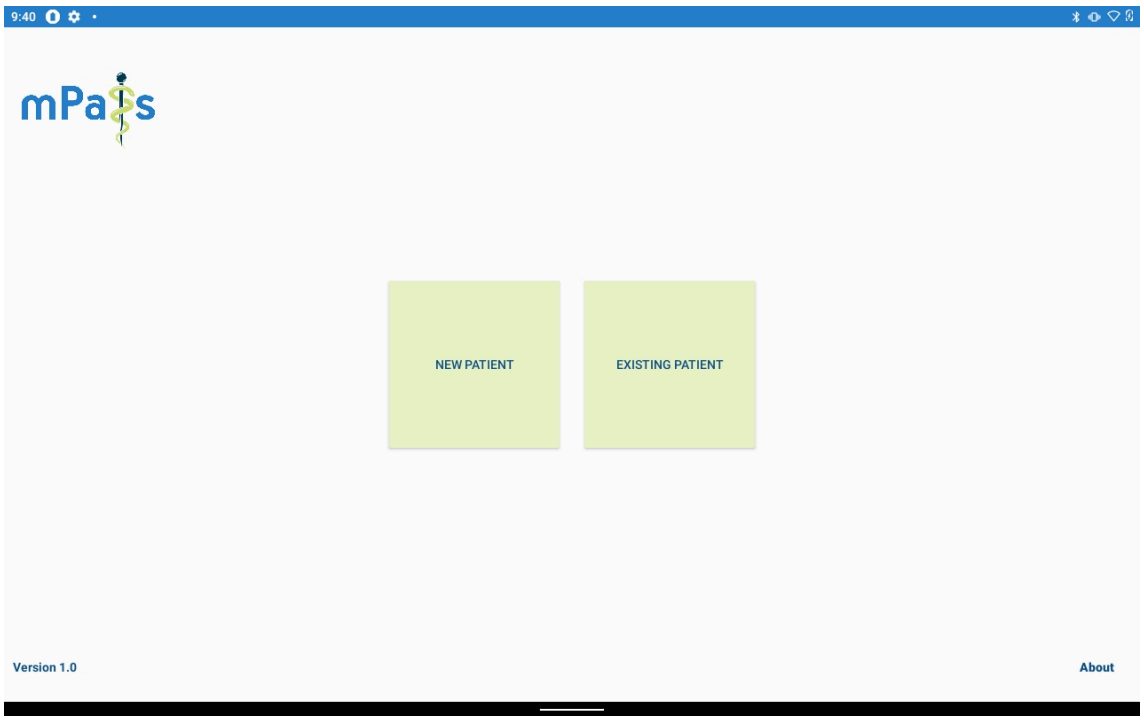
Παρουσίαση εφαρμογής

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει παρουσίαση (σε μορφή manual) της εφαρμογής m-Pals με σχετικά στιγμιότυπα και σχόλια επί των στιγμιοτύπων.

4.2 Αρχική οθόνη

Εφόσον ανοίξουμε την εφαρμογή, βλέπουμε την κεντρική οθόνη η οποία περιλαμβάνει 2 κουμπιά. Το κουμπί ”NEW PATIENT” και το κουμπί ”EXISTING PATIENT”. Σε αυτό το σημείο πατώντας το κουμπί ”NEW PATIENT” μπορούμε να καταχωρήσουμε νέο ασθενή και πατώντας το κουμπί ”EXISTING PATIENT” εμφανίζεται η λίστα των καταχωρημένων ασθενών.



Εικόνα 4.1. Αρχική οθόνη εφαρμογής

4.3 Εισαγωγή νέου ασθενούς

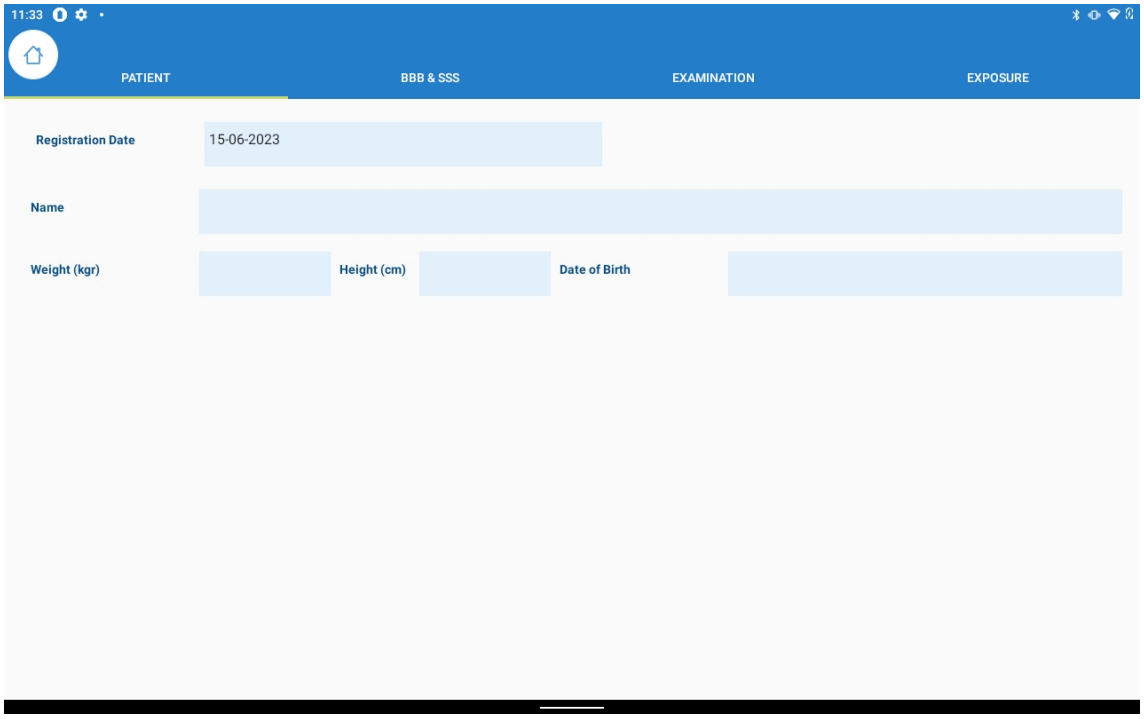
Πατώντας το κουμπί ”NEW PATIENT” (*εικόνα 4.1*) στην πρώτη οθόνη θα οδηγηθούμε στην οθόνη καταχώρησης νέουν ασθενούς.

Στο άνω μέρος της οθόνης, βλέπουμε τα στοιχεία ελέγχου. Το στοιχείου ελέγχου ”PATIENT”, το στοιχείο ελέγχου ”BBB & SSS”, το στοιχείο ελέγχου ”EXAMINATION” και τέλος, το στοιχείο ελέγχου EXPOSURE.

Ξεκινώντας από το στοιχείο ελέγχου ”PATIENT” βλέπουμε ότι το πεδίο της ημερομηνίας έρχεται προσυμπληρωμένο με την τρέχουσα ημερομηνία, ενώ μας δίνεται και η δυνατότητα τροποποίησης της.

Στην συνέχεια συμπληρώνουμε τα βασικά στοιχεία του ασθενούς ως εξής:

- Ονοματεπώνυμο,
- Βάρος (σε κιλά),
- Ύψος (σε εκατοστά),
- Ημερομηνία γέννησης (μέσω του επιλογέα ημερομηνίας)

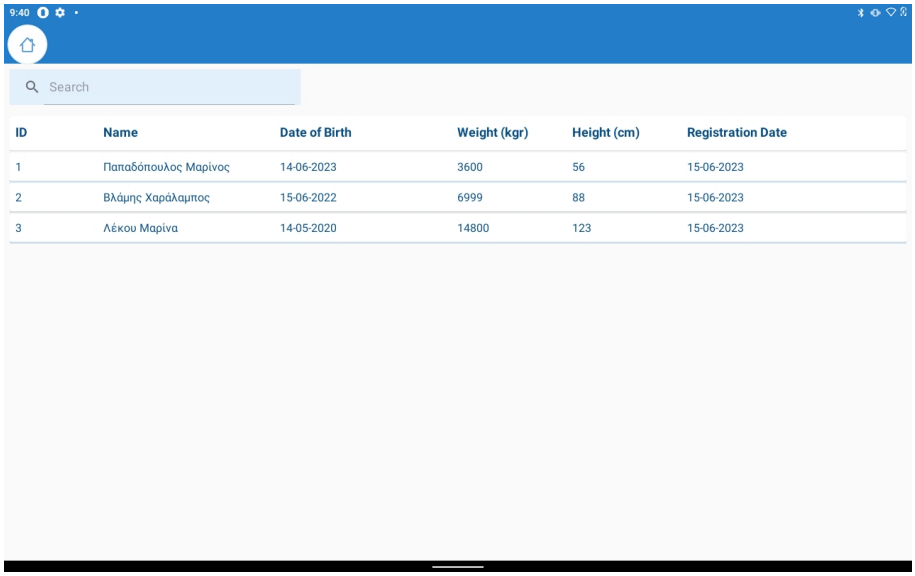


Εικόνα 4.2. Οθόνη καταχώρησης νέου ασθενούς

Σε αυτό το σημείο θα πρέπει να αναφέρουμε ότι για λόγους ταχύτητας καταχώρησης και ευχρηστίας, σε όλες τις οθόνες πραγματοποιείται αυτόματη αποθήκευση των καταχωρούμενων δεδομένων χωρίς να απαιτείται το πάτημα κάποιου κουμπιού.

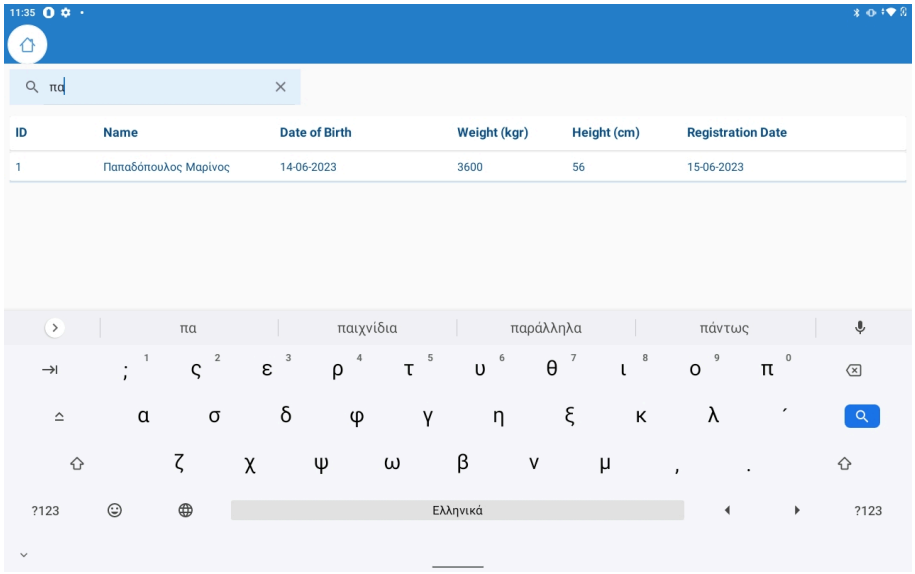
4.4 Λίστα ασθενών

Πατώντας το κουμπί ”EXISTING PATIENT” στην αρχική οθόνη μεταφερόμαστε στην οθόνη με την λίστα των ασθενών.



Εικόνα 4.3. Οθόνη λίστας ασθενών

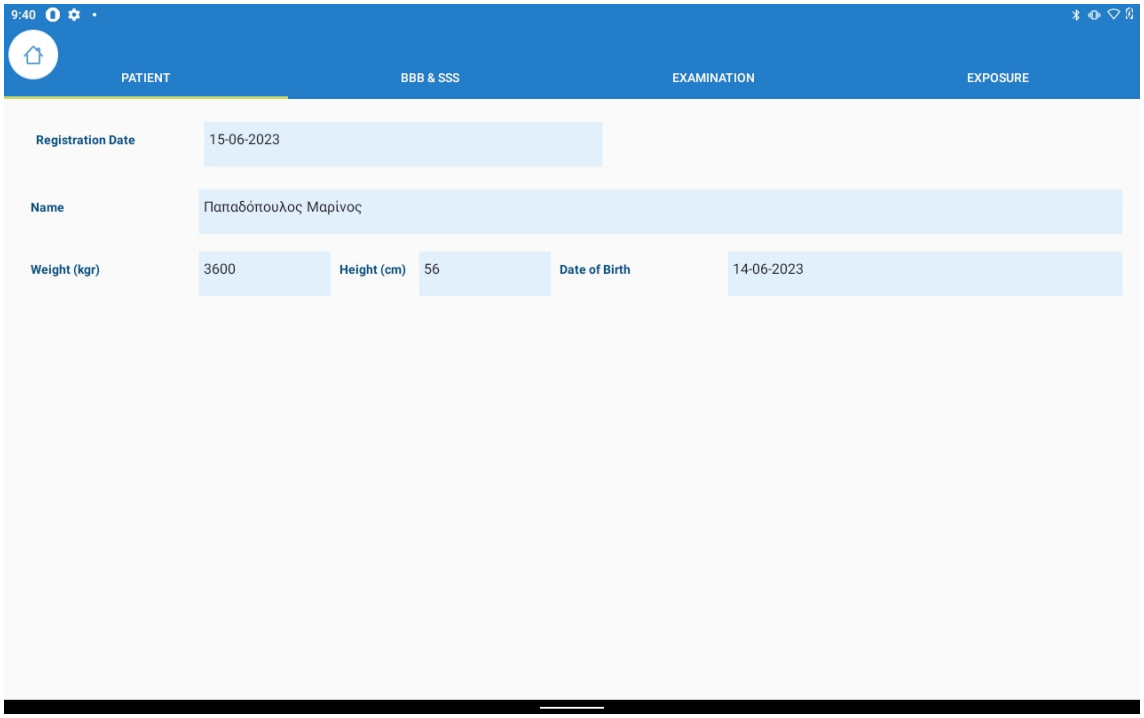
Σε αυτή την οθόνη μπορούμε να πραγματοποιήσουμε και αναζήτηση στην λίστα ασθενών με το ονοματεπώνυμο.



Εικόνα 4.4. Οθόνη αναζήτησης ασθενώνς

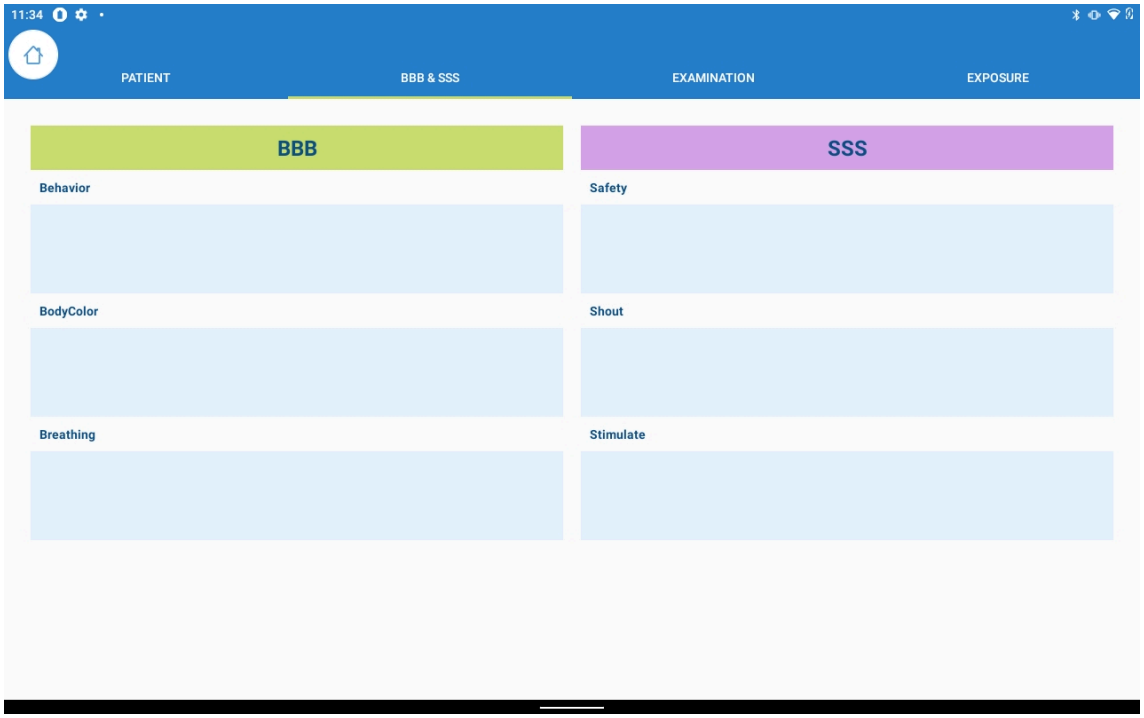
4.5 Επεξεργασία ασθενούς

Πατώντας πάνω στην εγγραφή κάποιου ασθενούς *(εικόνα 4.3)* μεταφερόμαστε στην οθόνη προβολής και επεξεργασίας ασθενούς. Και εδώ ισχύει ότι και κατά την καταχώρηση νέου ασθενούς. Τα πεδία επεξεργάζονται και αποθηκεύονται live και δεν χρειάζεται κάποια ενέργεια από την πλευρά του χρήστη σχετικά με την αποθήκευση.



Εικόνα 4.5. Οθόνη προβολής και επεξεργασίας ασθενούς

4.6 Προβολή - Καταχώριση και επεξεργασία BBB & SSS



Εικόνα 4.6. Οθόνη προβολής και επεξεργασίας των δεδομένων BBB & SSS

Σε αυτή την οθόνη αν βάσει της αξιολόγησης δεν παρατηρηθεί παθολογικό εύρημα καταγράφουμε OK στο αντίστοιχο πεδίο, ενώ σε αντίθετη περίπτωση καταγράφουμε το παθολογικό εύρημα που βρέθηκε.

Πιο συγκεκριμένα πραγματοποιούμε τις παρακάτω καταγραφές:

Behavior (Συμπεριφορά): Καταγράφουμε τα παθολογικά ευρήματα από την εκτίμηση του μυϊκού τόνου και του επιπέδου συνείδησης ώστε να αξιολογηθεί η αναπνευστική, η κυκλοφορική και η εγκεφαλική λειτουργία.

Body Color (Χρώμα δέρματος): Καταγράφουμε παθολογικά ευρήματα όπως κυάνωση, ωχρότητα, στίγματα δέρματος, εξανθήματα) κατά την αξιολόγηση κυρίως της αιμάτωσης του δέρματος, κάτι που αντικατοπτρίζει την κυκλοφορική λειτουργία του ασθενούς.

Breathing (Αναπνοή): Καταγράφουμε παθολογικά ευρήματα κατά την αξιολόγηση της αναπνευστικής λειτουργίας κάτι που αντικατοπτρίζει τη δυσκολία οξυγόνωσης του ασθενούς.

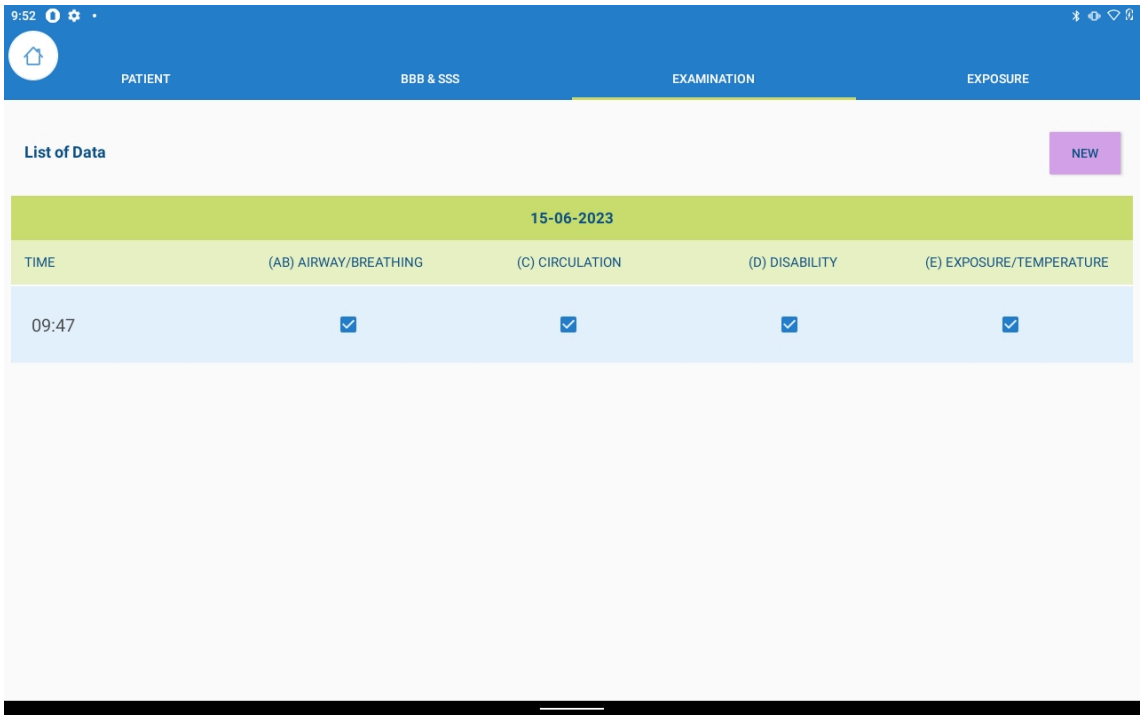
Safety (Ασφάλεια): Καταγράφουμε οποιονδήποτε παράγοντα μας επηρεάζει στο να προσεγγίσουμε άμεσα τον ασθενή π.χ. ηλεκτρικό ρεύμα. Χρησιμοποιούμε μέτρα ατομικής προστασίας για την προσέγγιση του ασθενούς. Προέχει η ασφάλεια πρώτα του ανανήπτη και έπειτα του ασθενή.

Shout (Κλήση για Βοήθεια): Σε περίπτωση που έχει γίνει κλήση προς τις υηρεσίες Επείγουσας Προνοσοκομειακής Φροντίδας ή της ενδονοσοκομειακής ομάδας αναζωογόνησης, καταγράφουμε την κλήση και την ώρα που αυτή πραγματοποιήθηκε.

Stimulate (Παροχή ερεθίσματος): Καταγράφουμε την ανταπόκριση ή όχι του ασθενούς σε λεκτικά και/ή απτικά ερεθίσματα.

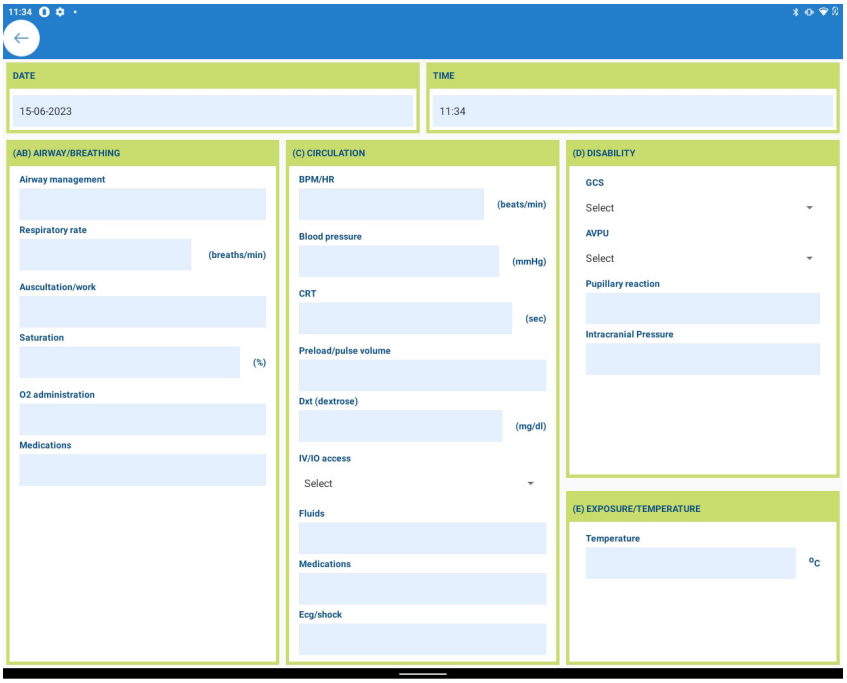
4.7 Προβολή - καταχώρηση και επεξεργασία εξετάσεων

Πατώντας το στοιχείο ελέγχου ”EXAMINATION” εμφανίζεται η οθόνη των εξετάσεων.



Εικόνα 4.7. Οθόνη εξετάσεων

Ενώ μέχρι στιγμής όλα τα στοιχεία που καταχωρούσαμε σε κάποιον ασθενή ήταν μοναδικά, στην καρτέλα ”EXAMINATIONS” έχουμε την δυνατότητα καταχώρησης πολλαπλών εγγραφών - εξετάσεων ώστε να έχουμε έτσι εικόνα της κατάστασης και της πορείας του ασθενούς. Σε αυτό το σημείο μπορούμε είτε να πατήσουμε πάνω σε μία εξέταση από την σχετική λίστα και είτε να δούμε τα δεδομένα και να τα επεξεργαστούμε (δείτε εικόνα 4.9), είτε πατώντας το κουμπί ”NEW” (δείτε εικόνα 4.8) να προσθέσουμε μια νέα. Επίσης, για κάθε εξέταση υπάρχουν και κάποιοι δείκτες κατάστασης οι οποίοι αν είναι τσεκαρισμένοι μας δείχνουν ότι υπάρχουν δεδομένα στο αντίστοιχο τμήμα ενώ αν δεν είναι τσεκαρισμένοι το αντίθετο.



Εικόνα 4.8. Οθόνη καταχώρησης νέας εξέτάσης

Σε αυτό το σημείο θα πρέπει να σημειώσουμε ότι τόσο η ημερομηνία όσο και η ώρα έρχονται προσυμπληρωμένες για λόγους ταχύτητας αλλά ανα πάσα στιγμή μπορούν να τροποποιηθούν από τον χρήστη.

Εικόνα 4.9. Οθόνη επεξεργασίας εξέτάσης

4.7.1 Οδηγίες καταγραφής εξετάσεων

Για κάθε εξέταση καταγράφουμε την ώρα και την ημερομηνία κατά την οποία πραγματοποιείται. (πραγματοποιείται αυτόματα από το σύστημα).

Στην συνέχεια καταγράφουμε το είδος της εξέτασης ή των εξετάσεων ώς εξής:

ΑΒ. ΑΕΡΑΓΩΓΟΣ/ΑΝΑΠΝΟΗ (ΑΒ)

Χειρισμοί διάνοιξης: αν πραγματοποιήθηκαν βάσει της κατάστασης του αεραγωγού (βατός, επαπειλούμενος, μερικώς ή πλήρως αποφραγμένος) και με ποια διαδικασία έγινε η προσπάθεια διάνοιξης, (στοματοφαρυγγικός / ρινοφαρυγγικός αεραγωγός, διασωλήνωση, λαρυγγική μάσκα ή AMBU) κυκλώνοντας κάθε φορά την επιλογή μας και σημειώνοντας δίπλα την ώρα που πραγματοποιήθηκε.
Αναπνευστική συχνότητα: αριθμός αναπνοών του παιδιού ανά λεπτό. Πολύ σημαντική είναι η δυνατότητα να πραγματοποιούνται πολλές καταγραφές την ώρα για να αξιολογείται η εξέλιξη της αναπνοής στην πορεία του χρόνου (ταχύπνοια – βραδύπνοια - άπνοια).
Ακρόαση/Εργο: καταγραφή των ακροαστικών ευρημάτων κάθε δεδομένη χρονική στιγμή και σύγχρονη εκτίμηση του έργου αναπνοής, το οποίο αν είναι αυξημένο μπορούμε να καταγράψουμε με ποιο τρόπο εκδηλώνεται (εισολκές μεσοπλευρίων διαστημάτων, στέρνου και υποχονδρίων, αναπέταση ρινικών πτερυγίων, κίνηση κεφαλής πάνω-κάτω και σύσπαση των μυών του πρόσθιου θωρακικού τοιχώματος) στο πεδίο της ώρας.
Κορεσμός οξυγόνου: Ώρα καταγραφής και πολλαπλές καταγραφές
Τρόπος χορήγησης οξυγόνου: ώρα έναρξης και τρόπος πχ. ρινική κάνουλα, μάσκα Venturi.
Άλλα φάρμακα: που χρειάστηκαν για τη βελτίωση της αναπνευστικής λειτουργίας του παιδιού. Καταγραφή της ώρας χορήγησης και με συντομογραφία διεθνή μπορούμε να καταγράψουμε στο πεδίο της ώρας ποιο φάρμακο χορηγήθηκε.

C. ΚΥΚΛΟΦΟΡΙΚΟ (C)

Σφύξεις ανά λεπτό: Ώρα καταγραφής και πολλαπλές καταγραφές **Αρτηριακή πίεση:** Ώρα καταγραφής και πολλαπλές καταγραφές **Χρόνος τριχοειδικής επαναπλήρωσης (ΧΤΕ):** Ώρα καταγραφής και πολλαπλές καταγραφές για την αξιολόγηση των συστηματικών αγγειακών αντιστάσεων (μαζί με θερμοκρασία και ΔΑΠ).Χρήσιμος για την εκτίμηση της αιμάτωσης και βαθμού καταπληξίας. Άμεση εικόνα της βελτίωσης / επιβάρυνσης του ασθενούς **Προφόρτιο/Εύρος σφυγμού:** Εκτίμηση το προφόρτιου και του εύρους σφυγμού του ασθενούς βάσει της κλινικής εξέτασης και των ευρημάτων αυτής (π.χ. σφαγίτιδες, ψηλάφηση ήπατος, κεντρικές και περιφερικές σφίξεις). Χρήσιμη η καταγραφή αδρά των μεταβολών. **Dxt (dextrose):** Στενή παρακολούθηση επιπέδων γλυκόζης αίματος (mg/dl), ώρα καταγραφής και δυνατότητα πολλαπλών καταγραφών **IV/IO προσπέλαση:** ενδοφλέβια ή ενδοοστική προσπέλαση για τη χορήγηση υγρών ή φαρμάκων. Ώρα καταγραφής και σημείωση οδού προσπέλασης **Υγρά:** Σε κάθε καταγεγραμμένη χρονική στιγμή σημείωση αν χρειάστηκε η χορήγηση και το είδος των υγρών **Ινότροπα, άλλα φάρμακα - Αδρεναλίνη (επινεφρίνη), Αδενοσίνη, Αμιοδαρόνη, Λιδοκαΐνη, Ατροπίνη, Ναλοξόνη, Γλυκόζη κλπ.:** Σε κάθε καταγεγραμμένη χρονική στιγμή σημείωση αν χρειάστηκε η χορήγηση και το είδος φαρμάκου για την αποκατάσταση της αιμοδυναμικής αστάθειας του ασθενούς. Κυριότερες συντομογραφίες στο κάτω μέρος του πίνακα. **ΗΚΓ/SHOCK:** Ώρα καταγραφής του ηλεκτροκαρδιογραφήματος και τύπος ρυθμού (απινιδώσιμος και μη απινιδώσιμος ρυθμός). Καταγραφή ώρας και ενέργειας απινίδωσης (απινίδωση με 4J/kg βάρους σώματος με βάση τις υπάρχουσες ενδείξεις).

D. ΝΕΥΡΟΛΟΓΙΚΟ (D)

GCS/ΞυΛΕΔ: Σκορ κλίμακας Γλασκώβης - ώρα καταγραφής και πολλαπλές καταγραφές. Αντίστοιχα και στη χρήση της κλίμακας ΞυΛΕΔ καταγραφή κάθε συγκεκριμένη χρονική στιγμή στο σωστό πεδίο του πίνακα αδρά το επίπεδο συνείδησης του παιδιού βάσει του ακρωνυμίου

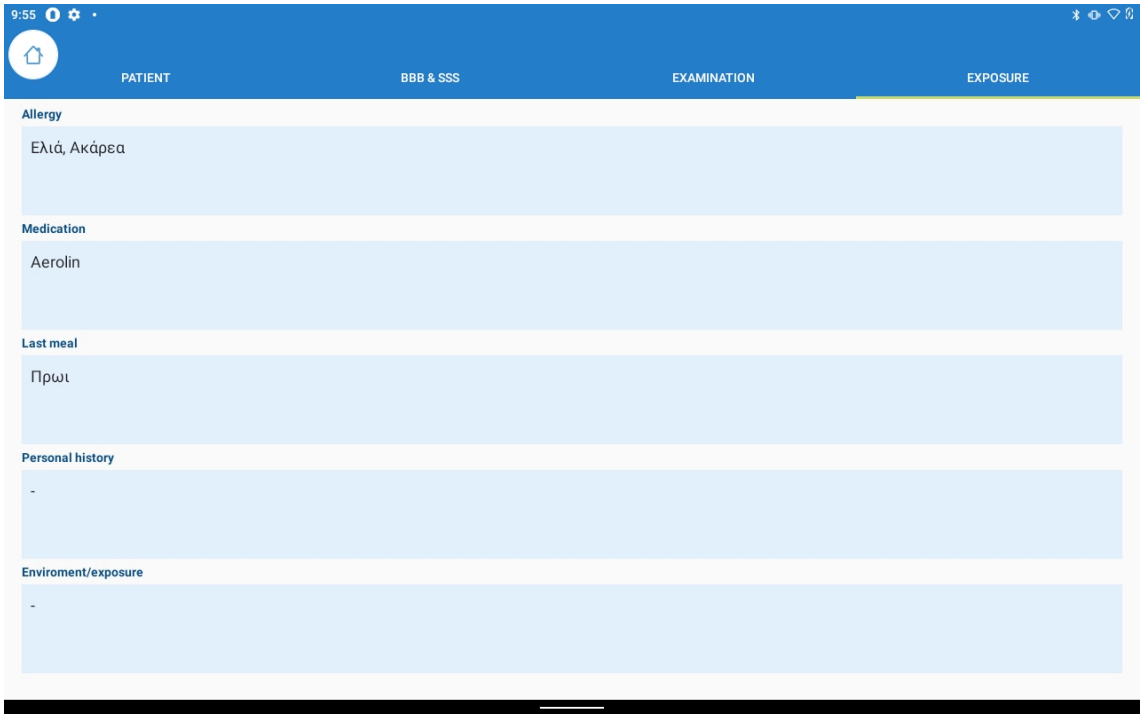
- ***Ξύπνιο:*** Φυσιολογική αντίδραση,
- ***Λεκτικά ερεθίσματα:*** Όταν απαντά σε λεκτικά ερεθίσματα,
- ***Επώδυνα ερεθίσματα:*** Όταν αντιδρά μόνο σε Επώδυνα ερεθίσματα,
- ***Δεν Απαντά:*** Όταν Δεν Απαντά σε κανένα ερέθισμα,

είτε **Σκορ της Κλίμακας Γλασκώβης.**

- ***Κόρες:*** σχολιασμός αυτών βάσει κλινικής εξέτασης στο αντίστοιχο πεδίο καταγραφής του πίνακα (μέγεθος, αντίδραση στο φως). Χρήσιμη η καταγραφή αδρά των μεταβολών,
- ***Αυξημένη ενδοκράνια πίεση/χειρισμοί:*** Επί κλινικής υποψίας αυξημένης ενδοκράνιας πίεσης καταγραφή των σημείων αυξημένης ενδοκράνιας πίεσης, της ώρας εντοπισμού τους και των χειρισμών για τη διόρθωση στο αντίστοιχο πεδίο της ώρας,

Θερμοκρασία - Ώρα καταγραφής και πολλαπλές καταγραφές

4.8 Προβολή επεξεργασία & καταχώρηση δεδομένων έκθεσης



Εικόνα 4.10. Οθόνη στοιχείων έκθεσης

Σε αυτήν την οθόνη καταγράφονται τα δεδομένα ”έκθεσης” τους ασθενούς ώς εξής:

Αλλεργίες: Καταγραφή με παύλα (-) αν δεν αναφέρονται οποιεσδήποτε αλλεργίες (π.χ. φάρμακα, τροφές) ενώ αν υπάρχουν σημειώνονται ποιες είναι αυτές

Φάρμακα: Καταγραφή με παύλα (-) αν δε λαμβάνει συστηματικά φαρμακευτική αγωγή ενώ αν λαμβάνει σημειώνεται ποια είναι αυτή

Τελευταίο Γεύμα: καταγραφή ακριβούς ώρας τελευταίου γεύματος

Ατομικό αναμνηστικό: Καταγραφή με παύλα (-) αν δεν αναφέρεται προηγούμενο ατομικό αναμνηστικό ενώ αν αναφέρεται σημειώνεται ποιο είναι αυτό

Περιβάλλον: καταγραφή της πλήρους έκθεσης του παιδιού λαμβάνοντας υπόψη την αξιοπρέπειά του

Κεφάλαιο 5

Εξέλιξη & Συμπεράσματα

5.1 Προοπτικές Εξέλιξης

Έχοντας φτάσει στη τελική έκδοση της εφαρμογής και εφόσον ήδη έχουν γίνει αρκετές δοκιμές είμαστε βέβαιοι ότι η εφαρμογή m-Pals θα καταλάβει μια θέση στην φαρέτρα των υγειονομικών. Ωστόσο πιστεύουμε ότι τα πραγματικά αποτελέσματα καθώς και οι οποιεσδήποτε προοπτικές εξέλιξης θα προκύψουν έπειτα από δοκιμές από υγειονομικούς χρήστες σε όσο γίνεται πιο πραγματικές συνθήκες.

Επίσης, μια μελλοντική προσέγγιση της εφαρμογής θα μπορούσε να είναι ο συνδυασμός της με εργαλεία τεχνητής νοημοσύνης με την βοήθεια της οποίας θα πραγματοποιείται live ανάλυση των δεδομένων ώστε να προτείνει ενέργειες αντιμετώπισης κ.λπ.

5.2 Συμπεράσματα

Οι τεχνολογικές εξελίξεις της τελευταίας δεκαετίας έχουν επηρεάσει πολλούς τομείς της καθημερινής μας ζωής, συμπεριλαμβανομένης της υγείας. Οι mobile εφαρμογές έχουν αποκτήσει ολοένα και μεγαλύτερη σημασία στην υγειονομική περίθαλψη, παρέχοντας ευκολία, πρόσβαση σε πληροφορίες και υποστήριξη σε επείγουσες καταστάσεις. Η εφαρμοφή mPals έρχεται να δώσει λύσεις ως σύστημα καταγραφής της εξειδικευμένης υποστήριξης της ζωής στα παιδιά και πιστεύουμε ότι το τελικό αποτέλεσμα θα αποτελέσει ένα κρίσιμο όπλο στην φαρέτρα των υγειονομικών.

Βιβλιογραφία

[1] JAVA. *JAVA*. URL: https://www.java.com/en/.

[2] SQLite. *SQLite*. URL: https://www.sqlite.org/index.html.

[3] Android Studio. *Android Studio*. URL: https://developer.android.com/studio.

[4] XML. *XML*. URL: https://www.w3.org/standards/xml/core.

[5] Αντωνόπουλος Σταύρος; Γρίβας Γρηγόριος; Καλαράς Γεώργιος; Τσακίρη Δήμητρα; Παπαδήμα Χριστίνα. «Καταγραφή της εξειδικευμένης υποστήριξης της ζωής στα παιδιά - m-Pals». Στο: *ΠΑΙΔΙΑΤΡΙΚΗ* 84.2 (2021), σσ. 144–151. URL: https://issuu.com/liveloula/docs/paidiatriki_84_2.