ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ
ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ασφαλές Σύστημα «Τεχνητού Παγκρέατος» με Ενοποίηση και Συγχρονισμένη Χρήση Επιμέρους Διαδικασιών/Εφαρμογών (με Χρήση Οντολογιών) για Κινητές Συσκευές.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χατζηδοπαυλάκης Χρήστος

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π

Αθήνα, Φεβρουάριος 2025

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ
ΠΛΗΡΟΦΟΡΙΑΣ
ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Ασφαλές Σύστημα «Τεχνητού Παγκρέατος» με Ενοποίηση και Συγχρονισμένη Χρήση Επιμέρους Διαδικασιών/Εφαρμογών (με Χρήση Οντολογιών) για Κινητές Συσκευές.

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Χατζηδοπαυλάκης Χρήστος

Επιβλέπων: Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή την 26η Φεβρουαρίου 2025.

.....	.....	.....
Βενιέρης Ιάκωβος Καθηγητής Ε.Μ.Π	Κακλαμάνη Ι. Δήμητρα-Θεοδώρα Καθηγήτρια Ε.Μ.Π.	Ματσόπουλος Γεώργιος Καθηγητής Ε.Μ.Π

Αθήνα, Φεβρουάριος 2025

.....
Χρήστος Ι. Χατζηδοπαυλάκης

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Περίληψη

Στη σύγχρονη εποχή, οι ασθενείς με διαβήτη και άλλες συναφείς μεταβολικές διαταραχές αξιοποιούν ολοένα και περισσότερο τη συνδυαστική χρήση έξυπνων κινητών τηλεφώνων (smartphones) με ασύρματα φορέσιμα συστήματα συνεχούς παρακολούθησης γλυκόζης (CGMS). Τα συστήματα αυτά μετρούν σε πραγματικό χρόνο τα επίπεδα γλυκόζης στο αίμα του ασθενούς με χρήση υποδόριου αισθητήρα, και αποστέλλουν την τιμή με ασύρματο τρόπο σε κάποιο smartphone ή κάποιο άλλο ασύρματο δέκτη.

Στην παρούσα διπλωματική εργασία αναπτύχθηκε μια εφαρμογή λήψης και διαχείρισης τιμών γλυκόζης του αίματος για smartphones με λειτουργικό σύστημα Android. Η εφαρμογή μπορεί να λάβει πραγματικές τιμές μέσω της εφαρμογής Xdrip^{+}, ή να παράξει εικονικές τιμές γλυκόζης και να τις αποθηκεύει σε τοπική βάση δεδομένων. Στη συνέχεια, παρέχει στο χρήστη διάφορες δυνατότητες και πληροφορίες σχετικά με την συγκέντρωση γλυκόζης στο αίμα του.

Πιο αναλυτικά, ο χρήστης μπορεί να δει σε πραγματικό χρόνο ένα διάγραμμα με τις μετρήσεις γλυκόζης των τελευταίων ωρών και να κρατάει ψηφιακές σημειώσεις. Μπορεί επίσης να ανατρέχει σε παλαιότερες ημερομηνίες και να βλέπει τις τιμές γλυκόζης αίματος, το αντίστοιχο διάγραμμα και τις σημειώσεις που είχε κάνει τη συγκεκριμένη ημερομηνία. Μέσω της εφαρμογής δίνεται η δυνατότητα στο χρήστη να θέτει συναγερμούς (alerts) αν η τιμή γλυκόζης που έλαβε το σύστημα είναι μικρότερη ή μεγαλύτερη από οριακές τιμές που ορίζει ο χρήστης. Επιπλέον, ο χρήστης έχει τη δυνατότητα, μέσω της εφαρμογής, να υπολογίζει την ποσότητα ινσουλίνης που θα χρειαστεί να εισάγει στον οργανισμό του προκειμένου να μειώσει τη συγκέντρωση γλυκόζης στο αίμα του στο επίπεδο που έχει ο ίδιος ορίσει. Τέλος, η εφαρμογή παρέχει τη δυνατότητα παραγωγής αρχείων αναφορών (reports) που περιέχουν όλες τις ληφθείσες τιμές γλυκόζης, τους υπολογισμούς εφάπαξ δόσεων ινσουλίνης και τους συναγερμούς που αφορούν κάποιο συγκεκριμένο χρονικό διάστημα που επιλέγει ο χρήστης.

Λέξεις Κλειδιά

Έξυπνα κινητά τηλέφωνα, Android , Συστήματα Συνεχούς Παρακολούθησης Γλυκόζης, Firebase, Xdrip^{+}

Abstract

Patients with diabetes and other related metabolic disorders increasingly utilize the combination of smartphones with wireless wearable Continuous Glucose Monitoring Systems (CGMS). CGMS measure glucose levels in real-time using a subcutaneous sensor and can wirelessly transmit the values to a smartphone or a dedicated wireless receiver.

In this thesis, an Android-based smartphone application was developed for receiving and managing blood glucose values. The application is able to receive real readings via Xdrip$^{+.}$ application or create virtual glucose readings, store them in a local database, and then offer the user various functionalities and information regarding their blood glucose concentration levels.

More specifically, the user can view a real-time graph of their glucose readings over the past few hours and make digital notes. Additionally, the user can access past data by checking previous dates to view glucose values, the corresponding graph, and any notes made on that date. The application also allows the user to set alerts if the glucose readings received are below or above threshold values set by the user. Additionally, the user can calculate the amount of insulin they need to administer to reduce their blood glucose to the desired level. Finally, the application offers the ability to generate report files containing all received glucose values, single-dose insulin calculations, and alerts triggered for a specific time period selected by the user.

Keywords

Smartphones, Android, Continuous Glucose Monitoring Systems, Firebase, Xdrip$^{+.}$

Ευχαριστίες

Με την εκπόνηση της διπλωματικής μου εργασίας, θα ήθελα να ευχαριστήσω τον καθηγητή της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών κ. Ιάκωβο Βενιέρη, για την επίβλεψη αυτής της διπλωματικής εργασίας και την ευκαιρία που μου έδωσε να ασχοληθώ με ένα πολλά υποσχόμενο αντικείμενο που αναπτύσσεται ραγδαία. Επιπλέον, θα ήθελα να ευχαριστήσω τους καθηγητές κα. Δήμητρα-Θεοδώρα Κακλαμάνη και κ. Γεώργιο Ματσόπουλο, για τη συμμετοχή τους στην τριμελή εξεταστική επιτροπή.

Θα ήθελα επίσης να ευχαριστήσω την Υποψήφια Διδάκτορα Χριστίνα Δαμιανού για την καθοδήγηση που μου παρείχε σε όλα τα στάδια ανάπτυξης της εφαρμογής και συγγραφής της παρούσας διπλωματικής εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω την οικογένειά μου και τους φίλους μου για όλη την στήριξη και βοήθεια που μου παρείχαν όλα αυτά τα χρόνια.

Πίνακας Περιεχομένων

Κατάλογος Εικόνων

1. Εισαγωγή

Στο πρώτο κεφάλαιο θα γίνει μια παρουσίαση της νόσου του σακχαρώδους διαβήτη, θα γίνει αναφορά στα συστήματα “τεχνητού παγκρέατος” καθώς επίσης και παρουσίαση της εφαρμογής η οποία σχεδιάστηκε και υλοποιήθηκε στα πλαίσια αυτής της εργασίας. Τέλος, θα παρουσιαστεί αναλυτικά η διάρθρωση της παρούσας εργασίας.

1.1 Σακχαρώδης Διαβήτης

Ο σακχαρώδης διαβήτης είναι μια ασθένεια με ραγδαία αύξηση στη συχνότητα εμφάνισης τα τελευταία χρόνια. Χαρακτηρίζεται από αύξηση ή μείωση της συγκέντρωσης σακχάρου στο αίμα (υπεργλυκαιμία ή υπογλυκαιμία) και διαταραχή του μεταβολισμού της γλυκόζης. Προς το παρόν δεν υπάρχει κάποια αποτελεσματική μέθοδος πρόληψης ή μόνιμη θεραπεία για όλους τους τύπους διαβήτη, με αποτέλεσμα η αντιμετώπιση της νόσου να αποτελεί μια εξατομικευμένη διαδικασία η οποία βασίζεται στη σωστή και τακτική μέτρηση τιμών γλυκόζης. Σε αυτή την ενότητα παρουσιάζεται ο σακχαρώδης διαβήτης καθώς επίσης και ένας τρόπος αντιμετώπισης της νόσου σήμερα.

1.1.1 Περιγραφή της νόσου

Ο σακχαρώδης διαβήτης είναι μια χρόνια μεταβολική ασθένεια η οποία προκύπτει όταν το πάγκρεας δεν παράγει αρκετή ινσουλίνη είτε όταν ο οργανισμός δεν μπορεί να αξιοποιήσει αποτελεσματικά την παραγόμενη ινσουλίνη. Η ινσουλίνη είναι η ορμόνη με την οποία ρυθμίζεται από τον οργανισμό το επίπεδο της γλυκόζης στο αίμα. Το πιο σύνηθες παράγωγο της νόσου στον οργανισμό είναι η αύξηση / μείωση της συγκέντρωσης γλυκόζης στο αίμα (υπεργλυκαιμία / υπογλυκαιμία). Η υπεργλυκαιμία μακροχρόνια μπορεί να προκαλέσει σοβαρές βλάβες σε διάφορα συστήματα του οργανισμού και ιδιαίτερα στα νεύρα και τα αγγεία του αίματος [1]. Η υπογλυκαιμία μπορεί να προκαλέσει ακόμα και θάνατο σε μικρό χρόνο.

Οι κύριοι τύποι σακχαρώδους διαβήτη είναι ο διαβήτης τύπου 1, ο διαβήτης τύπου 2 και ο διαβήτης της κύησης. Ο διαβήτης τύπου 1 γνωστός και ως ινσουλινοεξαρτώμενος ή νεανικός διαβήτης χαρακτηρίζεται από ανεπαρκή παραγωγή ινσουλίνης και απαιτεί καθημερινή έγχυση ινσουλίνης. Η αιτία του διαβήτη τύπου 1 δεν είναι πλήρως κατανοητή καθώς παρατηρείται ότι δεν εξαρτάται από την κληρονομικότητα και πιθανόν να προκύπτει με αυτοάνοσο τρόπο. Δεν υπάρχει γνωστός τρόπος πρόληψης του διαβήτη τύπου 1 [2]. Ο διαβήτης τύπου 2, γνωστός και ως διαβήτης των ενηλίκων, είναι το αποτέλεσμα δύο παράλληλων προβλημάτων. Αρχικά το πάγκρεας αδυνατεί να δημιουργήσει αρκετή ινσουλίνη, ώστε να διατηρήσει σε φυσιολογικά επίπεδα τη συγκέντρωση γλυκόζης στο αίμα. Παράλληλα, τα κύτταρα στους μύες, το λίπος και το συκώτι αποκτούν αντίσταση στην ινσουλίνη με αποτέλεσμα να μην απορροφούν αρκετή γλυκόζη. Αντίθετα με τον διαβήτη τύπου 1, ο διαβήτης τύπου 2 (αν και κληρονομικός) είναι προλήψιμος. Αν και επίσης επηρεάζεται από το οικογενειακό ιστορικό του ατόμου, τα άτομα που ακολουθούν υγιεινή διατροφή, ασκούνται τακτικά και διατηρούν το σωματικό τους βάρος κοντά στο φυσιολογικό έχουν λιγότερες πιθανότητες να νοσήσουν, είτε νοσούν σε μεγαλύτερη ηλικία [3]. Ο διαβήτης κύησης εμφανίζεται κατά τη διάρκεια της εγκυμοσύνης σε γυναίκες που

πριν από αυτή δεν ήταν διαβητικές. Παρόμοια με άλλους τύπους διαβήτη διαταράσσει την πρόσληψη γλυκόζης από τα κύτταρα και προκαλεί υπεργλυκαιμία. Ο διαβήτης κύησης αυξάνει τον κίνδυνο επιπλοκών με την εγκυμοσύνη καθώς επίσης και τον κίνδυνο να παρουσιάσουν στο μέλλον διαβήτη τύπου 2 τόσο η μητέρα όσο και το παιδί. Παρότι δεν υπάρχει γνωστός και αποτελεσματικός τρόπος πρόληψης του διαβήτη κύησης, πιθανόν να επηρεάζεται από το σωματικό βάρους και τα επίπεδα άσκησης της γυναίκας πριν την εγκυμοσύνη. Τα επίπεδα σακχάρου των γυναικών που παρουσιάζουν διαβήτη κύησης επιστρέφουν στο φυσιολογικό μετά την γέννα [4].

Σύμφωνα με τον Παγκόσμιο Οργανισμό Υγείας, ο αριθμός των ανθρώπων που πάσχουν από διαβήτη έχει αυξηθεί από 200 εκατομμύρια το 1990 σε 830 εκατομμύρια το 2022. Το 2021 ο διαβήτης και η νεφρική ανεπάρκεια λόγω διαβήτη προκάλεσαν περισσότερους από 2 εκατομμύρια θανάτους και επιπρόσθετα περίπου το 11% των θανάτων από καρδιαγγειακές παθήσεις προήλθε λόγω υπεργλυκαιμίας [5].

1.1.2 Θεραπευτικοί Παράγοντες & Αντιμετώπιση

Λόγω της χρόνιας φύσης της νόσου του σακχαρώδους διαβήτη και δεδομένου ότι δεν υπάρχει μέχρι στιγμής κάποια μόνιμη θεραπεία, η αντιμετώπιση της νόσου στοχεύει στη ρύθμιση και διαχείριση των επιπέδων γλυκόζης στο αίμα, ώστε να πλησιάζουν όσο δυνατό περισσότερο τα φυσιολογικά επίπεδα. Με αυτό τον τρόπο επιτυγχάνεται πρόληψη ή και καθυστέρηση της εμφάνισης των επιπλοκών που σχετίζονται με την νόσο. Οι επιπλοκές της χρόνιας μη φυσιολογικής συγκέντρωσης σακχάρου στο αίμα αφορούν ανάμεσα σε άλλες, καρδιαγγειακές, νεφρικές, νευρικές, ηπατικές και οφθαλμολογικές παθήσεις.

Έχοντας ως στόχο τη συνεχή και μακροχρόνια ρύθμιση του σακχάρου στο αίμα η φαρμακευτική αντιμετώπιση της νόσου διαφέρει ανά τύπο διαβήτη. Οι ασθενείς με διαβήτη τύπου 1, εκτός από αλλαγές στον τρόπο ζωής είναι απαραίτητο να λαμβάνουν ινσουλίνη καθώς το πάγκρεας αδυνατεί να καλύψει τις ανάγκες του οργανισμού. Στους ασθενείς με διαβήτη τύπου 2 η αντιμετώπιση ποικίλει και εξαρτάται από το στάδιο διάγνωσης. Σε προ διαβητικά στάδια τα μέτρα πρόληψης περιλαμβάνουν αλλαγές στον τρόπο ζωής όπως υγιεινή διατροφή, σωματική άσκηση, διατήρηση φυσιολογικού βάρους, διακοπή του καπνίσματος, ρύθμιση της αρτηριακής πίεσης και τακτικός αιματολογικός έλεγχος [6]. Σε επόμενα στάδια η αντιμετώπιση της νόσου περιλαμβάνει φαρμακευτική αγωγή για τη ρύθμιση του σακχάρου αρχικά με από του στόματος φάρμακα όπως μετφορμίνη, SGLT2 αναστολείς και άλλα. Σε περιπτώσεις που τα από του στόματος φάρμακα δεν επαρκούν οι ασθενείς λαμβάνουν ινσουλίνη. Ο συνηθέστερος τρόπος λήψης της ινσουλίνης είναι η υποδόρια έγχυση. Επιπροσθέτως, υπάρχουν παράγοντες στην καθημερινότητα του ασθενούς οι οποίοι επηρεάζουν το σακχαρώδη διαβήτη όπως άγχος, θέματα ψυχικής υγείας, ύπνος, σωστή ενυδάτωση, καφεΐνη, άλλα νοσήματα, φαρμακευτική αγωγή και εν γένει παράγοντες που συντελούν στην απορρύθμιση των ορμονών του ασθενούς.

Ο γλυκαιμικός έλεγχος σε ασθενείς με διαβήτη είναι μια δύσκολη ισορροπία και προϋποθέτει τη διατήρηση των επιπέδων γλυκόζης σε όσο το δυνατόν πιο φυσιολογικά επίπεδα, αποφεύγοντας παράλληλα τις υπογλυκαιμίες και τις υπεργλυκαιμίες. Αυτό απαιτεί από τους ασθενείς την τακτική και σωστή μέτρηση του σακχάρου πολλές φορές την ημέρα. Ο πιο συνήθης τρόπος μέτρησης της γλυκόζης από τον ίδιο τον ασθενή είναι με συσκευή στην

οποία εισάγεται μια σταγόνα αίμα, συνήθως τρυπώντας το δάχτυλο. Οι μετρήσεις αυτές χρειάζεται να γίνονται πολλαπλές φορές καθημερινά ειδικότερα σε περιπτώσεις κατά τις οποίες ο ασθενής χρειάζεται να λάβει ινσουλίνη και θα πρέπει να γνωρίζει ακριβώς την ποσότητα και το είδος της ινσουλίνης που χρειάζεται να εγχυθεί. Ο αριθμός των παραγόντων που επηρεάζουν την αντιμετώπιση της νόσου είναι πολλοί σε αριθμό και ποικίλοι σε είδος κάτι που καθιστά τις διάφορες θεραπευτικές μεθόδους και ιδιαίτερα την ινσουλινοθεραπεία εξαιρετικά εξατομικευμένες διαδικασίες και την εκπαίδευση του διαβητικού ασθενούς τόσο στη νόσο όσο και στη θεραπεία αυτής απαραίτητη.

1.2 Συστήματα “Τεχνητού Παγκρέατος”

Όπως αναφέρθηκε και στην προηγούμενη ενότητα ο γλυκαιμικός έλεγχος απαιτεί από τον ασθενή τακτική μέτρηση του σακχάρου και έγχυση ινσουλίνης σε περιπτώσεις που χρειάζεται. Ακολουθώντας τους συνήθεις τρόπους μέτρησης γλυκόζης και έγχυσης ινσουλίνης ο ασθενής καλείται να τρυπάει το δάχτυλό του και να εγχύει ινσουλίνη με υποδόριο τρόπο πολλές φορές την ημέρα. Ένας σύγχρονος τρόπος ρύθμισης του διαβήτη σε ασθενείς με διαβήτη τύπου 1 και εν γένει σε ασθενείς που ακολουθούν ινσουλινοθεραπεία, είναι η χρήση Συστημάτων “Τεχνητού Παγκρέατος”. Τα συστήματα τεχνητού παγκρέατος προσομοιώνουν τη διαχείριση του σακχάρου με τον τρόπο που θα πραγματοποιούνταν από ένα υγιές πάγκρεας και αποτελούνται από τρία δομικά στοιχεία [7] :

- Τη Συσκευή – Σύστημα Συνεχούς Καταγραφής Γλυκόζης (Continuous Glucose Monitoring – CGM)
- Την Αντλία Συνεχούς Έγχυσης Ινσουλίνης (Insulin Pump)
- Το λογισμικό ελέγχου και διαχείρισης των ανωτέρω συσκευών

Η Συσκευή CGM τοποθετείται υποδόρια στο σώμα του ασθενούς και μετρά τις τιμές γλυκόζης στο μεσοκυττάριο υγρό κάθε 1 έως 5 λεπτά. Στη συνέχεια η μέτρηση μεταδίδεται μέσω πομπού, στο κινητό τηλέφωνο του χρήστη ή σε εξωτερική συσκευή-δέκτη. Η συσκευή CGM, τοποθετείται με γρήγορο, εύκολο και πρακτικό τρόπο και χρειάζεται να αντικαθίσταται μετά το πέρας κάποιων ημερών.

Η Αντλία Συνεχούς Έγχυσης Ινσουλίνης είναι εφοδιασμένη με ταχείας δράσης ινσουλίνη και μπορεί να πραγματοποιεί μικρές εγχύσεις ινσουλίνης κατά τη διάρκεια της ημέρας. Υπάρχουν δύο τύποι εγχύσεων, οι εγχύσεις Βασικού Ρυθμού (Basal Rate) και οι εγχύσεις εφάπαξ δόσεων ινσουλίνης (Bolus). Οι εγχύσεις Βασικού Ρυθμού αφορούν προκαθορισμένες ποσότητες ινσουλίνης. Προγραμματίζονται από τον ασθενή έπειτα από υπόδειξη του ιατρού του και πραγματοποιούνται αυτόματα ανά τακτά χρονικά διαστήματα. Στην αντλία, είναι δυνατόν να προγραμματιστούν περισσότεροι από ένας βασικοί ρυθμοί και μπορούν να εναλλάσσονται από το χρήστη κατά βούληση ανάλογα με τη φυσική δραστηριότητα, την ψυχολογική κατάσταση, πιθανή ασθένεια και άλλους παράγοντες. Οι αποφασισμένες από το χρήστη εφάπαξ δόσεις ινσουλίνης (Bolus) αφορούν εγχύσεις οι οποίες στοχεύουν στη διόρθωση κάποιας υψηλής τιμής γλυκόζης “Διορθωτική Ινσουλίνη” είτε παρέχονται για το μεταβολισμό κάποιου γεύματος “Γευματική Ινσουλίνη”. Η παραπάνω κατηγοριοποίηση εξυπηρετεί διαχωρισμό των εγχύσεων για τη διευκόλυνση διαχείρισης της θεραπείας από τον

ασθενή και της επικοινωνίας με τον ιατρό του. Ο τύπος ινσουλίνης ο οποίος εγχύεται είναι ο ίδιος σε όλες τις περιπτώσεις.

Το λογισμικό ελέγχου πραγματοποιεί τη διασύνδεση της συσκευής CGM και της αντλίας Ινσουλίνης. Επιπλέον, το λογισμικό δύναται να παρέχει στον ασθενή περαιτέρω δυνατότητες που διευκολύνουν τη διαχείριση της νόσου όπως διαγράμματα τιμών, αναφορές, συναγερμούς και άλλα.

Τα συστήματα “Τεχνητού Παγκρέατος” προσφέρουν σημαντικά πλεονεκτήματα στους ασθενείς με διαβήτη. Παρέχουν την δυνατότητα συνεχούς παρακολούθησης του σακχάρου αποφεύγοντας τα δυσάρεστα συνεχή τρυπήματα του δακτύλου, ευκολότερο και ακριβέστερο υπολογισμό της ινσουλίνης που εγχύεται καθώς και πληρέστερη εικόνα των τιμών γλυκόζης κατά τη διάρκεια της ημέρας και της νύχτας .Στην διεθνή βιβλιογραφία υπάρχουν ισχυρές ενδείξεις ότι η χρήση συστημάτων “Τεχνητού Παγκρέατος” είναι ένας αποτελεσματικός και ασφαλής τρόπος αντιμετώπισης του διαβήτη (ιδιαίτερα του διαβήτη τύπου 1). Καταγράφεται αύξηση του χρονικού διαστήματος κατά το οποίο οι ασθενείς βρίσκονται σε φυσιολογικό εύρος γλυκόζης και συνεπώς μείωση των χρονικών διαστημάτων που οι ασθενείς παρουσιάζουν υπεργλυκαιμίες ή υπογλυκαιμίες [8].

1.3 Αντικείμενο Διπλωματικής Εργασίας

Αντικείμενο της διπλωματικής εργασίας αποτελεί ο σχεδιασμός και η υλοποίηση εφαρμογής για κινητές συσκευές, η οποία θα αποτελεί μέρος συστήματος του “Τεχνητού Παγκρέατος”. Θα αφορά χρήστες που νοσούν από διαβήτη, ακολουθούν θεραπεία με έγχυση ινσουλίνης και χρησιμοποιούν σύστημα “Τεχνητού Παγκρέατος”. Η εφαρμογή θα δέχεται πληροφορίες τιμών γλυκόζης του χρήστη σε πραγματικό χρόνο, θα τις αποθηκεύει και στη συνέχεια θα παρέχει στο χρήστη διάφορες λειτουργικότητες με στόχο την ευκολότερη και αποτελεσματικότερη διαχείριση της ινσουλινοθεραπείας.

Πιο συγκεκριμένα, η εφαρμογή θα δύναται να λειτουργεί απερίσπαστα στη συσκευή του χρήστη, θα δέχεται δεδομένα γλυκόζης αίματος μέσω της εφαρμογής xDrip^{+} είτε θα παράγει τοπικά εικονικά δεδομένα. Τα δεδομένα αυτά θα αποθηκεύονται μόνιμα σε τοπική (στη συσκευή του χρήστη), ασφαλή βάση δεδομένων. Στη συνέχεια θα παρέχεται στο χρήστη συνεχής και πλήρης εποπτεία παροντικών και παρελθοντικών τιμών γλυκόζης που έχουν καταγραφεί από την εφαρμογή. Αυτό θα επιτυγχάνεται μέσω: μόνιμης ειδοποίησης (Persistent notification), γραφήματος πραγματικού χρόνου, εμφάνιση καταγραφών σε μορφή ημερολογίου και μέσω αναλυτικών αρχείων αναφορών που ο χρήστης θα μπορεί να παράξει μέσω της εφαρμογής.

Παράλληλα, θα δίνεται στο χρήστη η δυνατότητα να θέτει επιθυμητές οριακές τιμές γλυκόζης και συναγερμούς οι οποίοι θα ενεργοποιούνται και θα τον ειδοποιούν σε πραγματικό χρόνο κάθε φορά που οι λαμβανόμενες τιμές γλυκόζης είναι εκτός των επιθυμητών οριακών τιμών. Επιπλέον, θα δίνεται η δυνατότητα στο χρήστη να ορίζει εντός της εφαρμογής διάφορες βιομετρικές παραμέτρους όπως βασικοί ρυθμοί, παράγοντες ευαισθησίας στην ινσουλίνη και τους υδατάνθρακες και διάρκεια δράσης αυτών στον οργανισμό του. Με βάση αυτές τις τιμές θα υπάρχει η δυνατότητα υπολογισμού δόσεων Γευματικής ή Διορθωτικής ινσουλίνης (Bolus).

Επιπροσθέτως, ο χρήστης θα δύναται να παράξει μέσω της εφαρμογής αρχεία αναφορών τύπου PDF τα οποία θα περιέχουν λεπτομερώς τιμές γλυκόζης, συναγερμούς που ενεργοποιήθηκαν και δόσεις ινσουλίνης που έχει εγχυθεί σε χρονικό διάστημα που ο ίδιος θα επιλέγει. Τέλος, μέσω της εφαρμογής ο χρήστης θα μπορεί να δημιουργεί, αποθηκεύει και προβάλλει βοηθητικές σημειώσεις που αφορούν εξωγενείς παράγοντες που επηρεάζουν τη θεραπεία όπως γεύματα, σωματική καταπόνηση, ψυχολογικές μεταπτώσεις και άλλα. Η εφαρμογή θα ονομάζεται Glucose Monitor App (GMA).

1.4 Διάρθρωση Διπλωματικής Εργασίας

Στο δεύτερο κεφάλαιο της παρούσας διπλωματικής εργασίας παρουσιάζονται οι βασικές τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής GMA. Αρχικά δίνεται μια σύντομη περιγραφή της εκάστοτε τεχνολογίας και στη συνέχεια αναφέρεται ο τρόπος με τον οποίο χρησιμοποιήθηκε η κάθε τεχνολογία κατά την ανάπτυξη της GMA. Επιγραμματικά οι εν λόγω τεχνολογίες είναι το λειτουργικό σύστημα για κινητές συσκευές Android, το ενσωματωμένο περιβάλλον ανάπτυξης Android Studio, η γλώσσα προγραμματισμού Java, το Firebase και η εφαρμογή xDrip^{+}.

Στο τρίτο κεφάλαιο αναλύεται το σύνολο των λειτουργιών της εφαρμογής και όλοι οι τρόποι με τους οποίους κάποιος χρήστης μπορεί να αλληλεπιδράσει με αυτή. Θα παρουσιαστούν αναλυτικά όλα τα δεδομένα που συλλέγει η εφαρμογή, όλες οι περιπτώσεις χρήσης και οι αντίστοιχες οθόνες διεπαφής χρήστη, το σύνολο των λειτουργιών παρασκηνίου που εκτελεί η εφαρμογή, οι άδειες και τα δικαιώματα που χρειάζεται ο χρήστης να παραχωρήσει στην εφαρμογή και τέλος θα αναλυθεί η λειτουργικότητα της εφαρμογής σε διαφορετικές εκδόσεις του λειτουργικού συστήματος Android.

Στο τέταρτο κεφάλαιο, περιγράφεται αναλυτικά η υλοποίηση της εφαρμογής σε επίπεδο πηγαίου κώδικα. Αρχικά παρουσιάζεται το θεωρητικό πλαίσιο της αρχιτεκτονικής που έχει επιλεχθεί. Στη συνέχεια αναλύεται η δομή και οι κλάσεις του πηγαίου κώδικα Android και ο τρόπος με τον οποίο υλοποιούν τις λειτουργικότητες της εφαρμογής. Επιπλέον παρουσιάζονται οι βιβλιοθήκες λογισμικού οι οποίες χρησιμοποιήθηκαν και αναλύεται ο πηγαίος κώδικας που έχει συγγραφεί ώστε να τις υποστηρίξει.

Τέλος, στο πέμπτο κεφάλαιο ακολουθεί ο επίλογος, που περιλαμβάνει συμπεράσματα που προέκυψαν από την ανάπτυξη και δοκιμή της εφαρμογής, καθώς επίσης και προτάσεις μελλοντικών επεκτάσεων αυτής.

2. Τεχνολογίες

2.1 Android

Το Android είναι ένα λειτουργικό σύστημα για κινητές συσκευές και υπολογιστές τύπου ταμπλέτας [9] . Η Αμερικανική εταιρεία Android Inc. ξεκίνησε την ανάπτυξή του ως λειτουργικό σύστημα για ψηφιακές κάμερες το 2003. Το 2004 η εταιρεία αποφάσισε να αλλάξει τον τύπο συσκευών στις οποίες στοχεύει το Android από κάμερες σε κινητά τηλέφωνα. Το 2005 η

Android Inc. εξαγοράστηκε από την επίσης Αμερικανική εταιρεία Google Inc. Η Google αποφάσισε να βασίσει το Android στο λειτουργικό σύστημα ανοικτού κώδικα Linux. Στις 05 Νοεμβρίου 2005 η Google ανακοίνωσε την κοινοπραξία Open Handset Alliance στην οποία εντάχθηκαν δεκάδες εταιρείες τεχνολογίας και κινητών τηλεφώνων οι πιο γνωστές από τις οποίες είναι οι εταιρείες Intel Corporation, Motorola Inc, NVIDIA Corporation, LG Electronics Inc, Samsung Electronics, T-Mobile και Texas Instruments Incorporated. Η κοινοπραξία δημιουργήθηκε με στόχο να αναπτύξει και να προωθήσει το Android ως ένα δωρεάν λειτουργικό σύστημα ανοικτού κώδικα που θα μπορεί να υποστηρίζει εφαρμογές τρίτων (third-party applications). Το πρώτο κινητό τηλέφωνο με λειτουργικό σύστημα Android ήταν το T-Mobile G1, το οποίο κυκλοφόρησε πρώτη φορά στην αγορά στις 22 Οκτωβρίου 2008. Το 2012 το Android έγινε το δημοφιλέστερο λειτουργικό σύστημα για κινητές συσκευές ξεπερνώντας το λειτουργικό σύστημα iOS της εταιρείας Apple, και το 2020 υπολογίστηκε ότι το 75% όλων των κινητών συσκευών έχουν ως λειτουργικό σύστημα το Android.

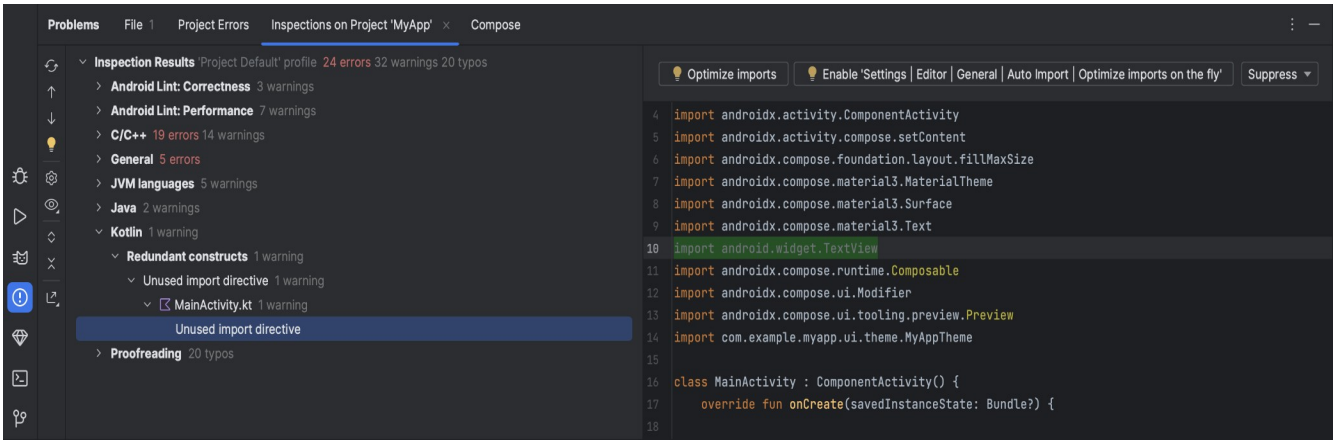
Το Android επιτρέπει στους χρήστες να εγκαθιστούν εφαρμογές είτε με απευθείας εγκατάσταση του εκτελέσιμου αρχείου της εφαρμογής είτε μέσω του Google Play Store. Το Google Play Store είναι ένα διαδικτυακό κατάστημα που παρέχει εφαρμογές, βιβλία, παιχνίδια και άλλα [10]. Είναι το μεγαλύτερο διαδικτυακό κατάστημα εφαρμογών όσον αφορά τον αριθμό μεταφορτώσεων εφαρμογών, και είναι υπεύθυνο για περίπου τις μισές μεταφορτώσεις εφαρμογών παγκοσμίως [11]. Το Google Play Store προσφέρει στους χρήστες υψηλό επίπεδο ασφάλειας καθώς απαιτεί από τους προγραμματιστές να ακολουθούν υψηλά στάνταρ ασφάλειας, ενώ παράλληλα τεχνολογίες όπως το Google Play Protect ελέγχουν 125 δισεκατομμύρια εφαρμογές την ημέρα.

Η GMA σχεδιάστηκε και αναπτύχθηκε ώστε να εγκαθίσταται και να εκτελείται αποκλειστικά σε συσκευές με λειτουργικό σύστημα Android.

2.2 Android Studio

Το Android Studio είναι ένα δωρεάν Ενσωματωμένο Περιβάλλον Ανάπτυξης (IDE) εφαρμογών Android της Google [12]. Είναι το επίσημο IDE για εφαρμογές Android, βασισμένο στο πρόγραμμα επεξεργασίας κώδικα IntelliJ IDEA της εταιρείας Jetbrains. Το Android Studio είναι ένα ενοποιημένο περιβάλλον ανάπτυξης εφαρμογών για όλες τις συσκευές Android. Προσφέρει έναν αριθμό από χαρακτηριστικά και δυνατότητες στους προγραμματιστές. Αρχικά, λειτουργεί ως πρόγραμμα επεξεργασίας κώδικα (code editor) και παρέχει δυνατότητες αποσφαλμάτωσης (debugging), δημιουργίας εικονικών εξομοιωτών (emulators), διασύνδεσης με φυσικές συσκευές και emulators καθώς επίσης και παραγωγή, επεξεργασία και ενημέρωση συνθέσιμων στοιχείων όπως εκτελέσιμα αρχεία τύπου APK (Android Package Kit) τόσο σε φυσικές συσκευές όσο και σε emulators. Διαθέτει επίσης ενσωματωμένη υποστήριξη για την πλατφόρμα Google Cloud διευκολύνοντας την ενσωμάτωση επιπλέον υπηρεσιών.

Το Android Studio είναι το IDE που χρησιμοποιήθηκε για την ανάπτυξη και αποσφαλμάτωση του κώδικα Android της εφαρμογής, καθώς επίσης και στην παραγωγή εκτελέσιμων αρχείων της GMA, όσον αφορά το κομμάτι της εφαρμογής που εγκαθίσταται και εκτελείται σε συσκευές με λειτουργικό σύστημα Android.



Εικόνα 2.1 Τα αποτελέσματα μιας επιθεώρησης ανάλυσης κώδικα στο Android Studio.

2.3 Java

Η Java είναι μια αντικειμενοστρεφής γλώσσα προγραμματισμού και υπολογιστική πλατφόρμα [13]. Κυκλοφόρησε πρώτη φορά το 1995 από την εταιρεία Sun Microsystems. Είναι μια γρήγορη, ασφαλής και αξιόπιστη γλώσσα προγραμματισμού που χρησιμοποιείται σε μια πληθώρα τύπου εφαρμογών όπως εφαρμογές κινητών τηλεφώνων, εφαρμογές για υπολογιστή, διαδικτυακές εφαρμογές κ.α. Αποτελεί μια από τις πλέον δημοφιλείς επιλογές για τους προγραμματιστές με εκατομμύρια εφαρμογές Java να χρησιμοποιούνται σήμερα.

Για την ανάπτυξη και εκτέλεση μιας εφαρμογής σε Java αξιοποιούνται τα τρία στοιχειώδη πακέτα που παρέχει η πλατφόρμα. Αρχικά γράφεται ο πηγαίος κώδικας (source code) της εφαρμογής σε γλώσσα Java με τη χρήση του Java Development Kit (JDK). Το JDK είναι ένα πακέτο εργαλείων λογισμικού το οποίο περιλαμβάνει μεταξύ άλλων συμβολομεταφραστή (interpreter), μεταγλωττιστή (compiler), debugger, απομεταγλωττιστή (disassembler), γεννήτρια αρχείων τύπου stub κ.α. [14].

Το JDK, λαμβάνει ως είσοδο το source code της εφαρμογής και μεταγλωτίζοντας τον παράγει έναν κώδικα μηχανής ακολουθώντας ένα συγκεκριμένο πρότυπο. Ο κώδικας αυτός είναι ένας κώδικας μηχανής Java που ονομάζεται bytecode και είναι ανεξάρτητος της συσκευής στην οποία εκτελείται η μεταγλώττιση. Ο bytecode, δύναται να εκτελεσθεί απευθείας σε οποιοδήποτε μηχανή μπορεί να εκτελέσει κώδικα μηχανής Java (τσιπ Java). Οι υπολογιστές που δεν διαθέτουν ένα τέτοιο κύκλωμα δύναται να εκτελέσουν τον bytecode με διερμηνεία (interpretation). Η εφαρμογή που εκτελεί αυτή τη διερμηνεία ονομάζεται Java Virtual Machine (JVM) [15]. Η JVM φορτώνει το bytecode, εκτελεί κάποιες διεργασίες επαλήθευσης του κώδικα και στη συνέχεια εκτελεί τον κώδικα εικονικά με τον ίδιο τρόπο που θα τον εκτελούσε ένα τσίπ Java. Η υλοποιήσή του JVM για τα διάφορα λειτουργικά συστήματα (Android, Windows, Linux κ.ο.κ) ονομάζεται Java Runtime Environment (JRE) [16]. Με τον τρόπο αυτό εξασφαλίζεται η απόζευξη του τύπου της μηχανής που συγγράφεται ένας Java source code από τον τύπο της μηχανής που εκτελείται η παραγόμενη εφαρμογή Java, αυτό έχει ως αποτέλεσμα να μπορεί να εκτελεστεί ο ίδιος source code σε πολλές διαφορετικού τύπου μηχανές αρκεί να διαθέτουν JRE.

Η γλώσσα προγραμματισμού Java είναι η γλώσσα με την οποία έχει εγγραφεί ο πηγαίος κώδικας της GMA όσον αφορά το μέρος αυτού το οποίο εγκαθίσταται και εκτελείται σε συσκευές με λειτουργικό σύστημα Android.

2.4 Firebase

Το Firebase είναι ένα σύνολο εργαλείων ανάπτυξης λογισμικού που βασίζονται στο cloud (υπολογιστικό νέφος) και βοηθά τους προγραμματιστές εφαρμογών για κινητές συσκευές να δημιουργούν, να αναπτύσσουν και να κλιμακώνουν τις εφαρμογές τους [17]. Αναπτύχθηκε από την εταιρεία Firebase Inc. η οποία ιδρύθηκε το 2011. Το 2014 η Firebase Inc εξαγοράστηκε από την εταιρεία Google η οποία επέκτεινε σημαντικά τις διαθέσιμες λειτουργικότητες του Firebase και το ενσωμάτωσε στα προσφερόμενα από την ίδια εργαλεία ανάπτυξης λογισμικού.

Το Firebase παρέχει μια πληθώρα εργαλείων και λειτουργιών οι σημαντικότερες από τις οποίες αφορούν: Πιστοποίηση χρηστών (Authentication), Βάση Δεδομένων πραγματικού χρόνου (Realtime Database), Μηνύματα στο Cloud, Παρακολούθηση Απόδοσης (Performance Monitoring), Ανάλυση σφαλμάτων καθώς και Εργαστήριο Δοκιμών (Test Lab).

Τα κέντρα δεδομένων (data centers) του Firebase διαθέτουν πιστοποιήσεις SOC 2 Type 2 και ISO 27001. Τα μέτρα ασφαλείας περιλαμβάνουν μεταξύ άλλων κρυπτογράφηση δεδομένων, Πρόσβαση βάσει ρόλων και καταγραφή προσβάσεων στα δεδομένα [18].

Η GMA διαθέτει τη δυνατότητα εγγραφής και πιστοποίησης χρηστών με χρήση e-mail και password. Για την επίτευξη αυτών των λειτουργιών χρησιμοποιήθηκαν ορισμένα από τα εργαλεία Authentication και Realtime Database του Firebase.

2.5 Η εφαρμογή Xdrip^{+}.

Η xDrip^{+} είναι μια ανεπίσημη και ανεξάρτητη εφαρμογή Android ανοιχτού κώδικα, η οποία λειτουργεί ως κέντρο συλλογής, αποθήκευσης, επεξεργασίας, προβολής και διανομής δεδομένων γλυκόζης αίματος και δύναται να είναι μέρος συστημάτων “Τεχνητού Παγκρέατος”. Υποστηρίζει ασύρματη σύνδεση με διάφορα συστήματα συνεχούς παρακολούθησης γλυκόζης (CGMS) και δύναται να λάβει δεδομένα απευθείας από αυτά είτε μέσω διαδικτύου [19]. Η εφαρμογή xDrip^{+}, έχει αναπτυχθεί με την πρωτοβουλία της Αμερικανικής, ανεξάρτητης, μη κερδοσκοπικής οργάνωσης The Nightscout Foundation η οποία ιδρύθηκε το 2014 [20]. Η xDrip^{+}, παρέχει μια πληθώρα λειτουργιών που αφορούν την προβολή και διανομή δεδομένων γλυκόζης, μια από αυτές είναι η δυνατότητα διαμοιρασμού δεδομένων γλυκόζης τοπικά σε άλλες εφαρμογές οι οποίες εκτελούνται στην ίδια συσκευή παράλληλα με την xDrip^{+}. Η εφαρμογή GMA δύναται να λάβει δεδομένα γλυκόζης από το xDrip^{+}.

3. Ανάλυση Εφαρμογής

3.1 Δεδομένα εφαρμογής

Σε αυτή την ενότητα αναλύονται τα δεδομένα που παράγει και αποθηκεύει μόνιμα η εφαρμογή. Παρουσιάζονται οι οντότητες που αναπαριστούν, ο τύπος των δεδομένων, οι οριακές τιμές των δεδομένων (εάν υπάρχουν), η χρήση τους από τις διάφορες λειτουργικότητες και ο τρόπος με τον οποίο αποθηκεύονται από την εφαρμογή. Η εφαρμογή διαθέτει δύο διαφορετικούς τρόπους αποθήκευσης δεδομένων. Ο πρώτος, είναι μια τοπική σχεσιακή βάση δεδομένων, η οποία δημιουργείται σε έναν διακριτό χώρο εντός του εσωτερικού χώρου αποθήκευσης της συσκευής του χρήστη. Ο χώρος αυτός δημιουργείται από το λειτουργικό σύστημα κατά την εγκατάσταση της εφαρμογής, προορίζεται αποκλειστικά για την εφαρμογή και είναι αποκλειστικά προσβάσιμος από την ίδια και καμία άλλη εφαρμογή. Ο συγκεκριμένος τρόπος αποθήκευσης αφορά δεδομένα τα οποία δεν αλλάζουν τιμή και είναι απαραίτητο να διατηρείται η ιστορικότητά τους ώστε να μπορούμε να ανακτήσουμε όλα τα παρελθοντικά δεδομένα του συγκεκριμένου τύπου. Ο δεύτερος τρόπος αποθήκευσης δεδομένων, είναι η αποθήκευση δεδομένων της μορφής κλειδιού-τιμής σε αρχείο τύπου XML. Είναι ένας ελαφρύς μηχανισμός αποθήκευσης δεδομένων που προσφέρει το Android και ονομάζεται Shared Preferences. Το XML αρχείο που περιέχει τα Shared Preferences αποθηκεύεται στον ίδιο χώρο που υπάρχει και η τοπική βάση και είναι επίσης προσβάσιμο αποκλειστικά και μόνο από την εφαρμογή. Ο τρόπος αυτός, αφορά δεδομένα για τα οποία μας ενδιαφέρει μόνο η παροντική τους τιμή και δεν υπάρχει ανάγκη ανάκτησης παρελθοντικών τιμών. Οι κλάσεις Java που αναπαριστούν τα δεδομένα αναλύονται στο επόμενο κεφάλαιο.

3.1.1 Δεδομένα Γλυκόζης Αίματος BG

Η βασικότερη κατηγορία δεδομένων που συλλέγει η εφαρμογή αφορά τα δεδομένα γλυκόζης αίματος του χρήστη. Τα δεδομένα γλυκόζης αίματος του χρήστη που από εδώ και πέρα θα αναφέρονται ως BG (Blood Glucose), αναπαριστούν την τιμή της συγκέντρωσης γλυκόζης στο αίμα για μια συγκεκριμένη χρονική στιγμή. Τα δεδομένα BG, αποθηκεύονται εντός της εφαρμογής ως ζεύγος τιμών. Οι τιμές αυτές είναι η τιμή συγκέντρωσης γλυκόζης στο αίμα εκφρασμένη σε mg/dL δηλαδή χιλιοστόγραμμα γλυκόζης ανά ένα δεκατόλιτρο αίματος και η χρονική στιγμή λήψης της μέτρησης εκφρασμένη σε UNIX Time (Java long), δηλαδή του αριθμού των δευτερολέπτων που έχουν παρέλθει από αρχή της πρώτης Ιανουαρίου 1970 έως την χρονική στιγμή λήψης της BG, χωρίς να συμπεριλαμβάνονται τα δευτερόλεπτα αναπήδησης [21]. Οι BG αποθηκεύονται στην τοπική βάση της εφαρμογής και αναπαρίστανται από την κλάση BGModel που αναλύεται στο επόμενο κεφάλαιο. Είναι σημαντικό να αναφέρουμε ότι για σαφέστερη και αποδοτικότερη διατήρηση των δεδομένων BG στην τοπική βάση, η χρονική στιγμή της BG αποθηκεύεται πάντα σε ζώνη ώρας UTC (Coordinated Universal Time) ανεξάρτητα από την τοπική ζώνη ώρας της συσκευής του χρήστη. Όταν στη συνέχεια οι τιμές προβάλλονται από τις διάφορες λειτουργικότητες της GMA, η προβαλλόμενη χρονική στιγμή μετασχηματίζεται ώστε να είναι σύμφωνη με την τοπική ζώνη ώρας της συσκευής του χρήστη. Επίσης εφόσον ο χρήστης το επιθυμεί μπορεί

να επιλέξει την προβολή των τιμών γλυκόζης BG σε mmol/L μέσω της επιλογής BG Units στις ρυθμίσεις, ωστόσο οι τιμές BG αποθηκεύονται πάντα εκφρασμένες σε mg/dL και μετασχηματίζονται σε mmol/L κατά την προβολή τους. Η GMA έχει τη δυνατότητα είτε να λάβει τιμές BG είτε να παράξει εικονικές τιμές ανάλογα με επιλογή του χρήστη στο πεδίο Data Source των ρυθμίσεων. Ανεξάρτητα του τρόπου δημιουργίας των δεδομένων BG, η GMA αποθηκεύει άμεσα τα νέα δεδομένα BG στην τοπική βάση προτού πραγματοποιηθεί οποιαδήποτε περαιτέρω ενέργεια που αφορά τα δεδομένα. Αφότου οι τιμές BG αποθηκευτούν με επιτυχία στην τοπική βάση, είναι προσβάσιμες από το χρήστη με διάφορους τρόπους. Επιγραμματικά οι οθόνες και λειτουργίες που προβάλλουν ή χρησιμοποιούν τιμές BG εντός της GMA είναι: Η μόνιμη ειδοποίηση που διατηρεί η εφαρμογή μέσω του ForegroundService (υποενότητα 3.3.1) και η οποία δείχνει μονίμως την τελευταία τιμή BG, η αρχική οθόνη (υποενότητα 3.2.4) στην οποία υπάρχει διάγραμμα τιμών BG καθώς και η τελευταία χρονικά BG που έχει αποθηκευτεί στην τοπική βάση, η οθόνη ημερολογίου (υποενότητα 3.2.5) μέσω της οποίας ο χρήστης μπορεί να δει διαγράμματα και λίστες δεδομένων BG ανά ημερομηνία, η οθόνη δημιουργίας αναφορών (υποενότητα 3.2.8), μέσω της οποίας ο χρήστης μπορεί να λάβει αναλυτικές λίστες δεδομένων BG για οποιοδήποτε χρονικό διάστημα επιλέξει και τέλος, ο μηχανισμός ελέγχου συναγερμών ο οποίος κάθε φορά που αποθηκεύεται μια νέα τιμή BG ελέγχει εάν η νέα τιμή είναι εντός ορίων κάποιου συναγερμού ώστε να τον ενεργοποιήσει.

3.1.2 Δεδομένα Συναγερμών

Κατά τη διάρκεια της ημέρας και της νύχτας και ανάλογα με τα γεύματα και τις δραστηριότητες, η συγκέντρωση γλυκόζης στο αίμα του χρήστη παρουσιάζει διακυμάνσεις. Η εφαρμογή, λαμβάνοντας τα δεδομένα BG όπως περιγράφονται στην προηγούμενη υποενότητα παρέχει στο χρήστη τη δυνατότητα να ορίσει συναγερμούς οι οποίοι ειδοποιούν το χρήστη (μέσω ηχητικών σημάτων και δόνησης της συσκευής) όταν η ληφθείσα τιμή γλυκόζης BG είναι εκτός επιθυμητών ορίων τα οποία μπορεί ο ίδιος να ορίσει. Για να επιτευχθεί αυτή η λειτουργία, η GMA δημιουργεί δεδομένα συναγερμών. Τα δεδομένα αυτά αναπαρίστανται από ένα ζεύγος τιμών, την τιμή ορίου και τον τελεστή. Η τιμή ορίου είναι η τιμή συγκέντρωσης γλυκόζης με την οποία θα συγκριθεί η νέα τιμή γλυκόζης BG ενώ ο τελεστής ορίζει το είδος του ελέγχου που θα πραγματοποιηθεί. Η τιμή ορίου αποθηκεύεται πάντα εκφρασμένη σε mg/dL και θα πρέπει να είναι εντός του διαστήματος [40,400]. Ο τελεστής έχει δυαδικό σύνολο τιμών και μπορεί να είναι over ή under, δείχνοντας αν η τιμή ορίου που τον συνοδεύει είναι επιθυμητό να αποτελέσει άνω ή κάτω όριο. Οι Συναγερμοί αποθηκεύονται στην τοπική βάση της εφαρμογής και αναπαρίστανται από την κλάση AlertsModel που αναλύεται στο επόμενο κεφάλαιο. Με την εγκατάσταση της εφαρμογής, εισάγονται στην τοπική βάση δύο συναγερμοί. Ένας συναγερμός άνω ορίου και ένας συναγερμός κάτω ορίου με ζεύγη τιμών (180,over) και (70,under) αντίστοιχα. Ο χρήστης μπορεί να δει και να διαχειριστεί όλους τους συναγερμούς μέσω της Οθόνης διαχείρισης συναγερμών (υποενότητα 3.2.7). Κάθε φορά που μια νέα BG εισάγεται στην τοπική βάση, πραγματοποιείται έλεγχος για συναγερμούς, κατά τη διάρκεια του οποίου η νέα τιμή BG συγκρίνεται σύμφωνα με τον τελεστή με την οριακή τιμή κάθε συναγερμού και εφόσον είναι εκτός οριακής τιμής ενεργοποιείται η διαδικασία εκτέλεσης του συναγερμού.

3.1.3 Δεδομένα Σημειώσεων

Η συγκέντρωση της γλυκόζης στο αίμα των χρηστών επηρεάζεται από ένα σύνολο παραγόντων. Οι παράγοντες αυτοί είναι ποικίλης μορφής και μεταξύ άλλων δύναται να αφορούν γεύματα, σωματική άσκηση ή καταπόνηση, λήψη διαφόρων φαρμάκων και μεταβολές στη συναισθηματική κατάσταση. Προκειμένου να είναι ευκολότερο για τους χρήστες να καταγράφουν και να ανακαλούν συμβάντα που θεωρούν σημαντικά, η GMA επιτρέπει στο χρήστη να κρατά σημειώσεις οι οποίες αποθηκεύονται μόνιμα εντός της εφαρμογής. Τα δεδομένα σημειώσεων αναπαρίστανται από ένα ζεύγος τιμών, το κείμενο της σημείωσης το οποίο αποθηκεύεται ως συμβολοσειρά (Java String) και την χρονική στιγμή δημιουργίας της σημείωσης η οποία αποθηκεύεται ως Unix Time (Java long). Οι Σημειώσεις αποθηκεύονται στην τοπική βάση της εφαρμογής και αναπαρίστανται από την κλάση NotesModel που αναλύεται στο επόμενο κεφάλαιο. Ο χρήστης μπορεί να δημιουργήσει μια νέα σημείωση μέσω ενός αιωρούμενου κουμπιού ενεργειών (floating action button, υποενότητα 3.2.3) το οποίο υπάρχει σε όλες τις βασικές οθόνες της εφαρμογής εκτός της Οθόνης Ρυθμίσεων. Ο χρήστης μπορεί να ανατρέξει σε παλαιότερες σημειώσεις με δύο τρόπους. Το σύνολο των σημειώσεων που έχουν αποθηκευτεί εντός της εφαρμογής είναι προσβάσιμο μέσω της Οθόνης προβολής σημειώσεων (υποενότητα 3.2.10). Επίσης, ο χρήστης μπορεί να προβάλλει τις σημειώσεις ανά ημέρα μέσω της Οθόνης Ημερολογίου (υποενότητα 3.2.5).

3.1.4 Δεδομένα Λήψης Ινσουλίνης & Υδατανθράκων

Οι χρήστες της εφαρμογής, εάν ο βασικός ρυθμός έγχυσης ινσουλίνης δεν καλύπτει τις ανάγκες τους, δύναται να χρειαστεί να πραγματοποιήσουν κάποια έκτακτη εφάπαξ έγχυση ινσουλίνης. Όπως αναφέρθηκε και στο εισαγωγικό κεφάλαιο, οι εγχύσεις αυτές μπορεί να αφορούν Γευματική ή Διορθωτική ινσουλίνη και ονομάζονται Bolus. Η εφαρμογή δίνει στο χρήστη τη δυνατότητα μέσω της Οθόνης Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης (υποενότητα 3.2.6), να υπολογίσει την ποσότητα ινσουλίνης που χρειάζεται να λάβει ώστε να οδηγήσει τη συγκέντρωση γλυκόζης σε επιθυμητά για τον ίδιο επίπεδα.

Για τον υπολογισμό της ποσότητας ινσουλίνης που χρειάζεται να εγχυθεί χρησιμοποιείται ο τύπος:

$$Insulin = \frac{BG - Tar}{ISF} + \frac{Carbs + COB}{CR} - IOB$$

Όπου:

- **BG** είναι η τρέχουσα τιμή γλυκόζης
- **Tar** είναι ο μέσος όρος του εύρους επιθυμητών τιμών γλυκόζης (στόχος) του χρήστη
- **ISF** είναι ο συντελεστής ευαισθησίας του ασθενούς στην ινσουλίνη (Insulin Sensitivity Factor). Πρακτικά ορίζει το κατά πόσο θα μειωθεί το σάκχαρο του ασθενούς από μια μονάδα ινσουλίνης.
- **Carbs** είναι η ποσότητα υδατανθράκων που θα καταναλωθούν, εφόσον πρόκειται για Γευματική ινσουλίνη. Εάν ο υπολογισμός αφορά Διορθωτική ινσουλίνη ο όρος έχει μηδενική τιμή.
- **COB** είναι η ποσότητα ενεργών υδατανθράκων στον οργανισμό του ασθενούς (Carbs On Board)

- **CR** είναι η αναλογία υδατανθράκων, πρακτικά ορίζει πόσα γραμμάρια υδατάνθρακες αφομοιώνει μια μονάδα ινσουλίνης (Carbs Ratio)
- **IOB** είναι η ενεργή Ινσουλίνη στον οργανισμό του ασθενούς (Insulin On Board)

Ο όρος IOB υπολογίζεται αθροίζοντας την ποσότητα της εναπομένουσας ενεργής ινσουλίνης από όλες τις δόσεις που έχουν εγχυθεί το τελευταίο χρονικό διάστημα. Το χρονικό διάστημα αυτό ονομάζεται Διάρκεια Δράσης της Ινσουλίνης (Duration of Insulin Activity – **DIA**). Το DIA μπορεί να λάβει πολλές διαφορετικές τιμές σε ώρες, ανάλογα με τον τύπο της ινσουλίνης που χρησιμοποιείται αλλά και παράγοντες που σχετίζονται με τον οργανισμό στον οποίο εγχύεται [22]. Λόγω του εξατομικευμένου χαρακτήρα του μεγέθους, η εφαρμογή λαμβάνει την τιμή από το χρήστη και δύναται ο ίδιος να την αλλάξει μελλοντικά. Για κάθε έκτακτη δόση ινσουλίνης που εγχύθηκε κατά τις τελευταίες DIA ώρες, προστίθεται στο συνολικό IOB η ποσότητα που παραμένει ενεργή στον οργανισμό. Στα πλαίσια της εφαρμογής θεωρείται ότι η ενεργή ινσουλίνη μειώνεται γραμμικά και αναλογικά του χρόνου και σταματά να είναι ενεργή μετά το πέρας των DIA ωρών. Όμοια με τον όρο IOB, ο όρος COB υπολογίζεται ως άθροισμα των ενεργών υδατανθράκων που καταναλώθηκαν το τελευταίο χρονικό διάστημα και επηρεάζουν τα επίπεδα γλυκόζης. Το χρονικό διάστημα αυτό είναι η Διάρκεια Δράσης των Υδατανθράκων (Duration of Carbohydrate Activity – **DCA**) και μπορεί να λάβει διάφορες τιμές σε ώρες ανάλογα με τον τύπο υδατανθράκων που καταναλώθηκαν και άλλους παράγοντες. Όμοια με το DIA, το DCA παρέχεται στην εφαρμογή από το χρήστη και για κάθε ποσότητα υδατανθράκων που καταναλώθηκαν τις τελευταίες DCA ώρες προστίθεται στο COB η ενεργή ποσότητα υδατανθράκων που παραμένει στον οργανισμό. Στα πλαίσια της εφαρμογής θεωρείται ότι η ενεργή ποσότητα υδατανθράκων μειώνεται γραμμικά και αναλογικά με το χρόνο.

Προκειμένου να πραγματοποιηθεί ο υπολογισμός, κάποιες από τις τιμές που αναφέρονται παραπάνω, χρειάζεται να αποθηκεύονται μόνιμα από την εφαρμογή . Το εύρος τιμών-στόχου Tar αποθηκεύεται ως δύο ξεχωριστές τιμές, η τιμή κάτω ορίου (TARGET_BG_MINIMUM) και η τιμή άνω ορίου (TARGET_BG_MAXIMUM). Οι οριακές τιμές είναι ακέραιοι αριθμοί συγκέντρωσης γλυκόζης εκφρασμένη σε mg/dL και έχουν οι ίδιες οριακές τιμές τα 40 mg/dL και 400 mg/dL αντιστοίχως. Ο συντελεστής ευαισθησίας στην ινσουλίνη ISF, μεταβάλλεται ανάλογα με την ώρα της ημέρας. Για το λόγο αυτό, αποθηκεύεται από την εφαρμογή ως δισδιάστατος πίνακας χρονικής περιόδου της ημέρας – τιμής. Η πρώτη στήλη του πίνακα ISF περιέχει χρονικά διαστήματα της ημέρας εκφρασμένα σε ώρες αναφοράς ενώ η δεύτερη στήλη περιέχει την τιμή της ISF εκφρασμένη σε συγκέντρωση γλυκόζης ανά μονάδα ινσουλίνης (mg/dL/U) για το αντίστοιχο χρονικό διάστημα ως ακέραιο αριθμό. Με αντίστοιχο δισδιάστατο πίνακα αποθηκεύεται και η αναλογία υδατανθράκων CR. Η πρώτη στήλη του πίνακα CR περιέχει χρονικά διαστήματα της ημέρας εκφρασμένα σε ώρες αναφοράς ενώ η δεύτερη στήλη περιέχει την τιμή της CR εκφρασμένη σε γραμμάρια υδατάνθρακα ανά μονάδα ινσουλίνης (g/U) που αφορά το αντίστοιχο χρονικό διάστημα ως ακέραιο αριθμό. Οι παράγοντες DIA και DCA αποθηκεύονται ως ξεχωριστές δεκαδικές αριθμητικές τιμές εκφρασμένοι σε ώρες με ανώτατα όρια τις 10 και 7 ώρες αντιστοίχως. Εκτός από τα μεγέθη που χρησιμοποιούνται για τον υπολογισμό των έκτακτων δόσεων ινσουλίνης, η εφαρμογή αποθηκεύει και τους βασικούς ρυθμούς έγχυσης ινσουλίνης του χρήστη. Ομοίως με τις ISF και

CR, επειδή ο βασικός ρυθμός μεταβάλλεται κατά τη διάρκεια της ημέρας, η εφαρμογή αποθηκεύει ένα δισδιάστατο πίνακα στον οποίο η πρώτη στήλη περιέχει χρονικά διαστήματα της ημέρας εκφρασμένα σε ώρες αναφοράς ενώ η δεύτερη στήλη περιέχει τον βασικό ρυθμό εκφρασμένο σε μονάδες ινσουλίνης ανά ώρα (U/hr) για το αντίστοιχο χρονικό διάστημα. Όλα τα δεδομένα που αναφέρονται στην παρούσα υποενότητα, αποθηκεύονται από την εφαρμογή στο XML αρχείο Shared Preferences με τη μορφή κλειδιού-τιμής.

3.1.5 Δεδομένα Συμβάντων

Όπως έχει περιγραφεί στις προηγούμενες υποενότητες, η εφαρμογή δίνει στο χρήστη τη δυνατότητα να ορίζει συναγερμούς που ενεργοποιούνται όταν η συγκέντρωση γλυκόζης στο αίμα του είναι εκτός επιθυμητών ορίων, καθώς επίσης και να υπολογίζει την ποσότητα έκτακτων εφάπαξ δόσεων ινσουλίνης που πιθανόν να χρειαστεί να λάβει. Δεδομένης της σημαντικότητας τέτοιων συμβάντων, η GMA καταγράφει και αποθηκεύει μόνιμα τέτοια συμβάντα όταν λαμβάνουν χώρα. Τα δεδομένα συμβάντων αναπαρίστανται από τέσσερεις τιμές, τον τύπο του Συμβάντος, το μήνυμα περιγραφής του, τη χρονική στιγμή που έλαβε χώρα και την τιμή γλυκόζης BG του χρήστη την χρονική στιγμή του συμβάντος. Ο τύπος του συμβάντος έχει δυαδικό σύνολο τιμών και μπορεί να είναι TYPE_BOLUS ή TYPE_ALERT, που ορίζουν αν το συμβάν αφορά κάποια έκτακτη εφάπαξ έγχυση ινσουλίνης ή την ενεργοποίηση κάποιου συναγερμού αντίστοιχα. Το μήνυμα περιγραφής του συμβάντος αποθηκεύεται ως συμβολοσειρά (Java String) και περιέχει περισσότερες λεπτομέρειες για το συμβάν. Η χρονική στιγμή αποθηκεύεται ως Unix Time (Java long) με ζώνη ώρας UTC. Η τιμή γλυκόζης BG αποθηκεύεται εκφρασμένη ως mg/dL. Τα Συμβάντα αποθηκεύονται στην τοπική βάση της εφαρμογής και αναπαρίστανται από την κλάση EventsModel που αναλύεται στο επόμενο κεφάλαιο. Μέσω της Οθόνης Δημιουργίας Αναφορών (υποενότητα 3.2.8), ο χρήστης μπορεί μέσω της εφαρμογής να δημιουργήσει αρχείο αναφοράς τύπου pdf στο οποίο εμπεριέχονται αναλυτικά όλα τα συμβάντα που έχει αποθηκεύσει η εφαρμογή για το χρονικό διάστημα που επιλέγει ο χρήστης.

3.1.6 Δεδομένα Προτιμήσεων Χρήστη

Προκειμένου η εφαρμογή να είναι ευκολότερη στη χρήση και να εξυπηρετεί καλύτερα τις ανάγκες του χρήστη, έχει υλοποιηθεί ένας αριθμός ρυθμίσεων (υποενότητα 3.2.9) που αφορούν προτιμήσεις του χρήστη. Αυτές οι ρυθμίσεις περιλαμβάνουν επιλογή του τρόπου έκφρασης των τιμών BG παντού στη εφαρμογή, επιλογή του τρόπου λήψης τιμών BG, ενεργοποίηση ή απενεργοποίηση των συναγερμών, και προσωρινή σίγαση όλων των συναγερμών ή μόνο των συναγερμών υψηλής ή χαμηλής BG. Οι τιμές BG μπορεί να επιλεχθεί να προβάλλονται είτε ως mg/dL δηλαδή χιλιοστόγραμμα γλυκόζης ανά ένα δεκατόλιτρο αίματος είτε ως mmol/L δηλαδή χιλιοστομοίρια γλυκόζης ανά λίτρο αίματος. Η επιλογή αυτή αποθηκεύεται από την εφαρμογή στο XML αρχείο Shared Preferences, ως ζεύγος κλειδιού τιμής, με κλειδί τη συμβολοσειρά BGUNITS και τιμή το επιλεγμένο από το χρήστη μέγεθος έκφρασης της BG (mg/dL ή mmol/L). Η τιμή αυτή στη συνέχεια διαβάζεται σε όλα τα σημεία εντός της εφαρμογής που πρόκειται να προβάλλουν τιμή BG ώστε να μετατρέψουν την προβαλλόμενη τιμή στο επιλεγμένο μέγεθος. Η επιλογή του τρόπου λήψης τιμών BG, ορίζει πώς θα λαμβάνει η εφαρμογή νέες τιμές γλυκόζης. Η επιλογή αποθηκεύεται

ως ζεύγος κλειδιού-τιμής με κλειδί τη συμβολοσειρά DATA_SOURCE και έχει σύνολο τιμών (“xDrip+”,”Generated Locally”). Η προτίμηση αυτή διαβάζεται από το Service της εφαρμογής ώστε να αποθηκεύονται στη βάση μόνο οι τιμές που προκύπτουν από την επιλεγμένη πηγή.

Επιπλέον, στις ρυθμίσεις της εφαρμογής υπάρχουν ρυθμίσεις διαχείρισης συναγερμών. Η επιλογή ενεργοποίησης ή απενεργοποίησης των συναγερμών αποθηκεύεται στο αρχείο Shared Preferences, ως ζεύγος κλειδιού τιμής με κλειδί ENABLE_ALERTS και δυαδική τιμή λογικού τύπου (true,false). Οι επιλογές προσωρινής σίγασης των συναγερμών αποθηκεύονται με ίδιο τρόπο. Η επιλογή προσωρινής σίγασης όλων των συναγερμών έχει κλειδί τη συμβολοσειρά ALL_SNOOZE_TIME και τιμή την χρονική στιγμή μέχρι την οποία θα ισχύει σίγαση όλων των συναγερμών εκφρασμένη σε UNIX Time (Java long). Με τον ίδιο ακριβώς τρόπο αποθηκεύονται και οι επιλογές σίγασης μόνο των συναγερμών υψηλής και χαμηλής τιμής BG, με τη διαφορά ότι έχουν κλειδί τις συμβολοσειρές HIGH_SNOOZE_TIME και LOW_SNOOZE_TIME αντίστοιχα. Όλες οι τιμές οι οποίες αφορούν τη διαχείριση συναγερμών, διαβάζονται από την εφαρμογή κάθε φορά που ενεργοποιείται κάποιος συναγερμός ώστε να γίνει διαχείρισή του σύμφωνα με τις προτιμήσεις του χρήστη.

3.2 Διεπαφή Χρήστη

Σε αυτή την ενότητα αναλύονται όλες οι οθόνες που αποτελούν την διεπαφή χρήστη της εφαρμογής. Για κάθε οθόνη, περιγράφονται όλες οι περιπτώσεις χρήσης που οδηγούν στην εμφάνισή της, όλες οι λειτουργικότητες που επιτελούνται μέσω αυτής καθώς επίσης και ο τρόπος που λαμβάνονται και αποστέλλονται πληροφορίες από και προς το χρήστη αντίστοιχα.

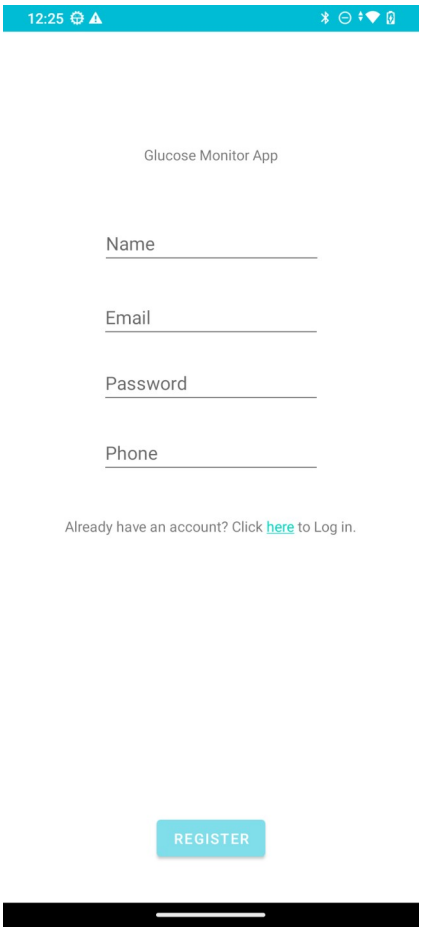
3.2.1 Οθόνη εγγραφής Χρήστη

Η οθόνη εγγραφής εμφανίζεται αποκλειστικά και μόνο εάν ο χρήστης το επιλέξει από την οθόνη εισόδου όπως αναλύεται στην υπόενότητα 3.2.2.

Η λειτουργικότητα που επιτελείται μέσω της οθόνης εγγραφής, είναι να πραγματοποιηθεί η μοναδική εγγραφή του χρήστη στη βάση δεδομένων του εξυπηρετητή (server) εάν αυτή δεν έχει ήδη πραγματοποιηθεί, διαφορετικά ο χρήστης οδηγείται στην οθόνη εισόδου προκειμένου να πραγματοποιήσει την είσοδό του στο σύστημα (log in). Ο server λειτουργεί διαδικτυακά και παράλληλα με την εφαρμογή και υποστηρίζει τη δυνατότητα εγγραφής χρηστών. Η λειτουργία αυτή έχει αναπτυχθεί με τη χρήση του εργαλείου Firebase και αναλύεται στην ενότητα 4. Ανάπτυξη Εφαρμογής. Η διαδικασία εγγραφής απαιτεί ο χρήστης να εισάγει τις προσωπικές του πληροφορίες όπως όνομα χρήστη, διεύθυνση ηλεκτρονικού ταχυδρομείου (e-mail), τον κωδικό με τον οποίο θα πραγματοποιεί την είσοδό του στο σύστημα (password) και τον αριθμό του τηλεφώνου του. Από αυτά τα τέσσερα πεδία τα πεδία e-mail και password είναι υποχρεωτικά και δεν θα πρέπει να είναι κενά καθώς είναι τα στοιχεία με τα οποία οι χρήστες πραγματοποιούν είσοδο μετά την αρχική εγγραφή.

Η οθόνη εγγραφής αποτελείται από εννέα στοιχεία. Αρχικά στο επάνω μέρος υπάρχει ένα πεδίο προβολής κειμένου (TextView) στο οποίο αναγράφεται το όνομα της εφαρμογής (Glucose Monitor App). Κάτω από αυτό υπάρχουν στοιχισμένα κάθετα, πέντε πεδία εισαγωγής κειμένου (EditText). Σε κάθε ένα από αυτά τα πεδία ο χρήστης μπορεί να εισάγει τις πληροφορίες όνομα χρήστη, e-mail, κωδικός χρήστη, επαλήθευση κωδικού και αριθμός

τηλεφώνου αντίστοιχα. Σε κάθε πεδίο αναγράφεται αρχικά μια υπόδειξη (hint) που αναφέρει τον τύπο πληροφορίας που δέχεται το εκάστοτε πεδίο και εξαφανίζεται όταν ο χρήστης ξεκινά να πληκτρολογεί στο συγκεκριμένο πεδίο. Επιπλέον, κάθε πεδίο δέχεται συγκεκριμένο τύπο συμβόλων που αντιστοιχούν στον τύπο της πληροφορίας. Για παράδειγμα το πεδίο εισαγωγής κειμένου που αφορά της εισαγωγή του τηλεφωνικού αριθμού δεν δέχεται πεζά ή κεφαλαία γράμματα και άλλους μη έγκυρους χαρακτήρες. Κάτω από το πεδίο εισαγωγής κωδικού υπάρχει ένα TextView στο οποίο αναφέρονται οι παράμετροι εγκυρότητας του κωδικού που ο χρήστης καλείται να δημιουργήσει. Κάτω από τα πεδία εισαγωγής κειμένου υπάρχει ένα τρίτο TextView το οποίο ενημερώνει τον χρήστη πως αν έχει ήδη πραγματοποιήσει εγγραφή μπορεί να πατήσει σε ένα συγκεκριμένο σημείο ώστε να μεταφερθεί στην οθόνη εισόδου χρήστη. Ακόμα πιο κάτω βρίσκεται ένα ψηφιακό κουμπί που αναγράφει Register. Όταν πατηθεί το συγκεκριμένο κουμπί γίνεται τοπικά έλεγχος της ορθότητας των πληροφοριών των τεσσάρων EditText. Για το πεδίο e-mail ελέγχεται αν η παρεχόμενη από το χρήστη εισαγωγή είναι έγκυρη και ακολουθεί τους κανόνες σύνταξης μιας διεύθυνσης e-mail δηλαδή περιέχει το σύμβολο @ , περιέχει έναν τομέα (π.χ .com) κ.λπ. Για το πεδίο password απαιτείται η συμβολοσειρά να αποτελείται από οκτώ έως δεκαέξι χαρακτήρες, να περιέχει τουλάχιστον ένα πεζό και τουλάχιστον ένα κεφαλαίο χαρακτήρα καθώς επίσης και τουλάχιστον έναν αριθμό. Εφόσον οι προαναφερθέντες έλεγχοι αποτύχουν η εφαρμογή προβάλλει ένα μήνυμα ενημέρωσης (Android Toast) και ενημερώνει το χρήστη για το ποιο πεδίο δεν έχει αποδεκτή τιμή και τον λόγο που η τιμή που εισήγαγε δεν είναι αποδεκτή. Παράλληλα η εφαρμογή μεταφέρει αυτόματα τον κέρσορα στο αντίστοιχο πεδίο. Αντίθετα, εφόσον ο έλεγχος των τιμών των πεδίων εισαγωγής είναι επιτυχημένος τότε η εφαρμογή εκκινεί στο παρασκήνιο την λειτουργία εγγραφής. Η λειτουργία αυτή μπορεί να λάβει από το server μήνυμα επιτυχίας (δηλαδή ο χρήστης έχει εγγραφεί στο σύστημα) ή αποτυχίας. Επειδή η συγκεκριμένη λειτουργία αποστέλλει και λαμβάνει πακέτα μέσω διαδικτύου εκτελείται ασύγχρονα, για το λόγο αυτό όταν εκκινείται και μέχρι να λάβει απάντηση η οθόνη εγγραφής προβάλλει μια κυκλική γραμμή προόδου ώστε να πληροφορήσει το χρήστη ότι εκτελείται μεταφορά δεδομένων. Τέλος εάν η λειτουργία λάβει μήνυμα επιτυχίας τότε ο χρήστης μεταφέρεται στην αρχική οθόνη της εφαρμογής. Σε αντίθετη περίπτωση η εφαρμογή εμφανίζει μήνυμα ενημέρωσης όπου πληροφορεί το χρήστη για το λόγο της αποτυχίας (π.χ. αδυναμία επικοινωνίας κ.α) και εμφανίζει ξανά την οθόνη εγγραφής.



Εικόνα 3.1 Στιγμιότυπο οθόνης κατά την προβολή της Οθόνης Εγγραφής Χρήστη.

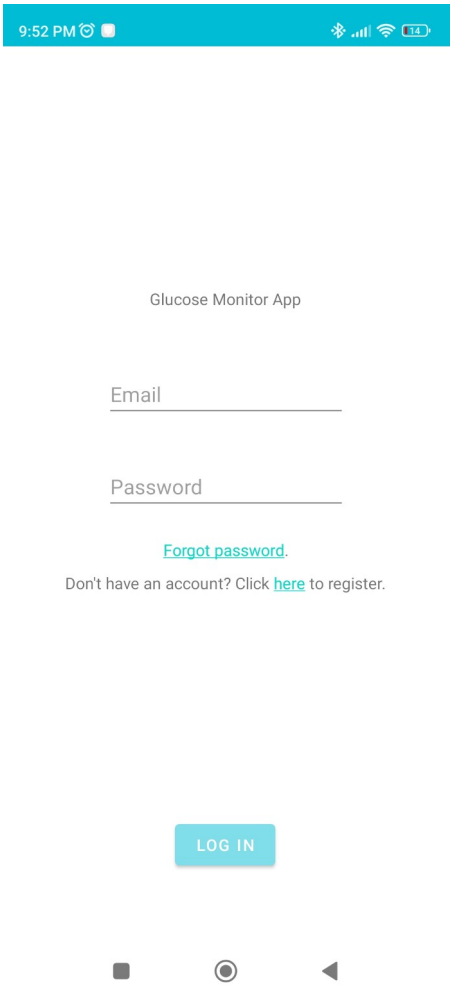
3.2.2 Οθόνη εισόδου Χρήστη

Κατά την πρώτη εκτέλεση της εφαρμογής μετά την εγκατάστασή της, η πρώτη οθόνη που παρουσιάζεται σε κάθε χρήστη είναι η Οθόνη εισόδου (Login Screen). Η οθόνη εισόδου χρήστη εμφανίζεται επίσης, εάν ο χρήστης το επιλέξει όσο προβάλλεται η οθόνη εγγραφής χρήστη, εάν ο χρήστης ανοίξει την εφαρμογή και δεν έχει πραγματοποιήσει είσοδο κατά το παρελθόν, ή εάν έχει πραγματοποιήσει έξοδο από την εφαρμογή. Εφόσον ο χρήστης έχει πραγματοποιήσει εγγραφή κατά το παρελθόν, τότε μπορεί να πραγματοποιήσει είσοδο στο σύστημα χρησιμοποιώντας το συνδυασμό e-mail και password τα οποία έθεσε κατά την εγγραφή του.

Οι λειτουργικότητες που επιτελούνται μέσω της οθόνης εισόδου είναι η πιστοποίηση των στοιχείων εισόδου (e-mail,password) του χρήστη από τη το server καθώς επίσης και η λειτουργία αλλαγής κωδικού στο server, σε περίπτωση που ο χρήστης δε θυμάται το συνθηματικό που έχει εισάγει κατά την εγγραφή του. Ο server λειτουργεί διαδικτυακά και παράλληλα με την εφαρμογή και υποστηρίζει τη δυνατότητα επαλήθευσης στοιχείων εισόδου και αλλαγής κωδικού χρηστών. Οι λειτουργικότητες εισόδου και αλλαγής κωδικού στο server έχουν αναπτυχθεί με τη χρήση του εργαλείου Firebase.

Η οθόνη εισόδου αποτελείται από έξι στοιχεία. Αρχικά στο επάνω μέρος υπάρχει ένα πεδίο προβολής κειμένου (TextView) στο οποίο αναγράφεται το όνομα της εφαρμογής (Glucose Monitor App). Κάτω από αυτό αυτό υπάρχουν στοιχισμένα κάθετα, δύο πεδία εισαγωγής

κειμένου (EditText). Τα πεδία εισαγωγής κειμένου αφορούν τη διεύθυνση e-mail και τον κωδικό εισόδου του χρήστη αντίστοιχα. Σε κάθε πεδίο αναγράφεται αρχικά μια υπόδειξη (hint) που αναφέρει τον τύπο πληροφορίας που δέχεται το εκάστοτε πεδίο και εξαφανίζεται όταν ο χρήστης ξεκινά να πληκτρολογεί στο συγκεκριμένο πεδίο. Επιπλέον κάθε πεδίο δέχεται συγκεκριμένο τύπο συμβόλων με περιορισμό χαρακτήρων, όπως έχει αναλυθεί στην υποενότητα 3.2.1. Κάτω από τα πεδία εισαγωγής κειμένου υπάρχει ένα δεύτερο TextView το οποίο δίνει στο χρήστη τη δυνατότητα να αλλάξει τον κωδικό του ενώ κάτω από αυτό υπάρχει ένα τρίτο TextView το οποίο ενημερώνει τον χρήστη πως αν δεν έχει πραγματοποιήσει εγγραφή μπορεί να πατήσει σε ένα συγκεκριμένο σημείο ώστε να μεταφερθεί στην οθόνη εγγραφής χρήστη. Εάν ο χρήστης επιλέξει να αλλάξει κωδικό, η εφαρμογή προβάλλει ένα αναδυόμενο παράθυρο. Το παράθυρο αυτό, αποτελείται από ένα πεδίο εισαγωγής e-mail και δύο ψηφιακά κουμπιά, ένα κουμπί ακύρωσης που πάνω του αναγράφει CANCEL και ένα κουμπί επικύρωσης που αναγράφει SEND RESET E-MAIL. Εφόσον ο χρήστης πατήσει το κουμπί ακύρωσης το αναδυόμενο παράθυρο κλείνει και η εφαρμογή προβάλλει την Οθόνη εισόδου. Εάν ο χρήστης επιλέξει το κουμπί επικύρωσης η εφαρμογή στέλνει στο server τη διεύθυνση e-mail την οποία ο χρήστης έχει πληκτρολογήσει στο πεδίο εισαγωγής e-mail. Εφόσον ο χρήστης πληκτρολογήσει ένα e-mail το οποίο υπάρχει στη βάση δεδομένων του server, τότε ο server αποστέλλει στην συγκεκριμένη διεύθυνση, ένα e-mail με έναν σύνδεσμο ο οποίος οδηγεί το χρήστη σε μια σελίδα μέσω της οποίας μπορεί να θέσει το νέο του συνθηματικό, ενώ εάν ο χρήστης πληκτρολογήσει μη έγκυρη διεύθυνση e-mail ή κάποιο e-mail που δεν υπάρχει στο server τότε η εφαρμογή εμφανίζει ένα μήνυμα που πληροφορεί το χρήστη για το σφάλμα εισαγωγής, κλείνει το αναδυόμενο παράθυρο και προβάλλει ξανά την Οθόνη εισόδου. Τέλος κάτω από τα δύο πεδία TextView βρίσκεται ένα ψηφιακό κουμπί που αναγράφει LOG IN. Όταν πατηθεί το συγκεκριμένο κουμπί γίνεται τοπικά έλεγχος της ορθότητας των πληροφοριών των δύο EditText που αφορούν την εισαγωγή e-mail και password. Εφόσον οι έλεγχοι αποτύχουν η εφαρμογή προβάλλει ένα μήνυμα ενημέρωσης (Android Toast) και ενημερώνει το χρήστη για το ποιο πεδίο δεν έχει αποδεκτή τιμή, τον λόγο που η τιμή που εισήγαγε δεν είναι αποδεκτή και η εφαρμογή μεταφέρει αυτόματα τον κέρσορα στο αντίστοιχο πεδίο. Αντίθετα εφόσον ο έλεγχος των τιμών των πεδίων εισαγωγής είναι επιτυχημένος τότε το η εφαρμογή εκκινεί στο παρασκήνιο την λειτουργία εισόδου. Η λειτουργία εισόδου αποστέλλει στο server διαδικτυακά τα στοιχεία e-mail και password που εισήγαγε ο χρήστης και μπορεί να λάβει από το server μήνυμα επιτυχίας ή αποτυχίας. Ομοίως με τη λειτουργία εγγραφής η εφαρμογή εκκινεί στο παρασκήνιο την λειτουργία εισόδου, και επειδή η συγκεκριμένη λειτουργία αποστέλλει και λαμβάνει πακέτα μέσω διαδικτύου εκτελείται ασύγχρονα, για το λόγο αυτό όταν εκκινείται και μέχρι να λάβει απάντηση η οθόνη εισόδου προβάλλει μια κυκλική γραμμή προόδου ώστε να πληροφορήσει το χρήστη ότι εκτελείται μεταφορά δεδομένων. Στην περίπτωση που η εφαρμογή λάβει μήνυμα επιτυχίας τότε ο χρήστης μεταφέρεται στην αρχική οθόνη της εφαρμογής. Αντίθετα σε περίπτωση αποτυχίας η εφαρμογή εμφανίζει μήνυμα ενημέρωσης όπου πληροφορεί το χρήστη για το λόγο της αποτυχίας και εμφανίζει ξανά την οθόνη εισόδου.



Εικόνα 3.2 Στιγμιότυπο οθόνης κατά την προβολή της Οθόνης Εισόδου Χρήστη.

3.2.3 Μενού και Έξοδος

Σε αυτή την υποενότητα θα αναλυθεί ο τρόπος με τον οποίο ο χρήστης πλοηγείται στις διάφορες οθόνες της εφαρμογής, καθώς επίσης και ο τρόπος που θέτει συγκεκριμένες ρυθμίσεις και προτιμήσεις χρήστη. Μετά την είσοδο ή εγγραφή του, ο χρήστης δύναται να χρησιμοποιήσει όλες τις λειτουργικότητες της εφαρμογής. Προκειμένου να είναι ευνόητη και χρηστική, η εφαρμογή διατηρεί την ίδια βασική δομή διεπαφής χρήστη σε όλες τις βασικές οθόνες πλην των οθονών εγγραφής, εισόδου και προβολής σημειώσεων. Η δομή αυτή ορίζει ότι σε κάθε οθόνη θα υπάρχει στο πάνω μέρος της μια επικεφαλίδα με τρία στοιχεία και στο κάτω μέρος αυτής ένα διαδραστικό πλωτό κουμπί (floating button). Αρχικά στο αριστερό μέρος της επικεφαλίδας υπάρχει ένα ψηφιακό κουμπί με σύμβολο τρεις οριζόντιες παράλληλες ευθείες γνωστό και ως burger button. Δεξιά από αυτό αναγράφεται ο τίτλος της οθόνης που προβάλλεται την συγκεκριμένη χρονική στιγμή και τέλος στο δεξιά μέρος της επικεφαλίδας υπάρχει ένα ακόμα ψηφιακό κουμπί με σύμβολο τρεις τελείες στοιχισμένες κάθετα, γνωστό και ως overflow button. Το burger button, όταν ενεργοποιείται από το χρήστη εμφανίζει στο αριστερό μέρος της οθόνης ένα συρόμενο μενού πλοήγησης (drawer menu) που περιέχει στο πάνω μέρος του το e-mail με το οποίο ο χρήστης έχει πραγματοποιήσει είσοδο και κάτω από αυτό ως επιλογές όλες τις βασικές οθόνες της εφαρμογής. Ονομαστικά στο drawer menu περιέχονται οι επιλογές: Home (Αρχική Οθόνη), Calendar (Οθόνη

Ημερολογίου), Bolus (Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης), Alerts (Οθόνη διαχείρισης συναγερμών), Reports (Οθόνη δημιουργίας αναφορών) και Settings (Οθόνη Ρυθμίσεων). Όταν επιλέγεται κάποια επιλογή τότε το drawer menu εξαφανίζεται και η εφαρμογή προβάλλει την αντίστοιχη οθόνη. Όταν ενεργοποιείται το overflow button, η εφαρμογή προβάλλει ένα μενού επιλογών το οποίο επικαλύπτει μέρος της οθόνης που προβάλλεται ως τότε. Το συγκεκριμένο μενού εμπεριέχει αποκλειστικά και μόνο την επιλογή εξόδου από το σύστημα (Log out). Όταν ενεργοποιείται η συγκεκριμένη επιλογή πραγματοποιείται έξοδος του χρήστη από το σύστημα και αφού ολοκληρωθεί, η εφαρμογή προβάλλει την Οθόνη Εισόδου Χρήστη. Σε κάθε βασική οθόνη της εφαρμογής πλην της Οθόνης Ρυθμίσεων, υπάρχει στο κάτω δεξιά μέρος της, ένα κυκλικό floating action button με σύμβολο ένα μολύβι. Όταν ενεργοποιείται αυτό το κουμπί παρέχεται στο χρήστη η δυνατότητα δημιουργίας μιας νέας σημείωσης. Εφόσον ο χρήστης επιλέξει τη δημιουργία νέας σημείωσης, η εφαρμογή εμφανίζει ένα αναδυόμενο παράθυρο το οποίο αποτελείται από επικεφαλίδα, πεδίο εισαγωγής κειμένου και δυο ψηφιακά κουμπιά. Η επικεφαλίδα του παραθύρου αναγράφει “New Note” και βρίσκεται στην κορυφή αυτού. Κάτω από την επικεφαλίδα στο κεντρικό τμήμα του παραθύρου υπάρχει ένα στοιχείο εισαγωγής κειμένου (EditText) το οποίο επιτρέπει στους χρήστες να πληκτρολογούν το κείμενο της σημείωσης. Κάτω από το EditText υπάρχουν τα ψηφιακά κουμπιά αποθήκευσης (SAVE) και ακύρωσης (CANCEL). Εάν ο χρήστης πατήσει Cancel, το αναδυόμενο παράθυρο δημιουργίας νέας σημείωσης κλείνει και η εφαρμογή προβάλλει την βασική οθόνη η οποία προβάλλονταν πριν το άνοιγμα του αναδυόμενου παραθύρου. Αντίθετα, εάν πατηθεί το κουμπί SAVE, η εφαρμογή αποθηκεύει παρασκηνιακά και ασύγχρονα τη νέα σημείωση, στην τοπική βάση δεδομένων.

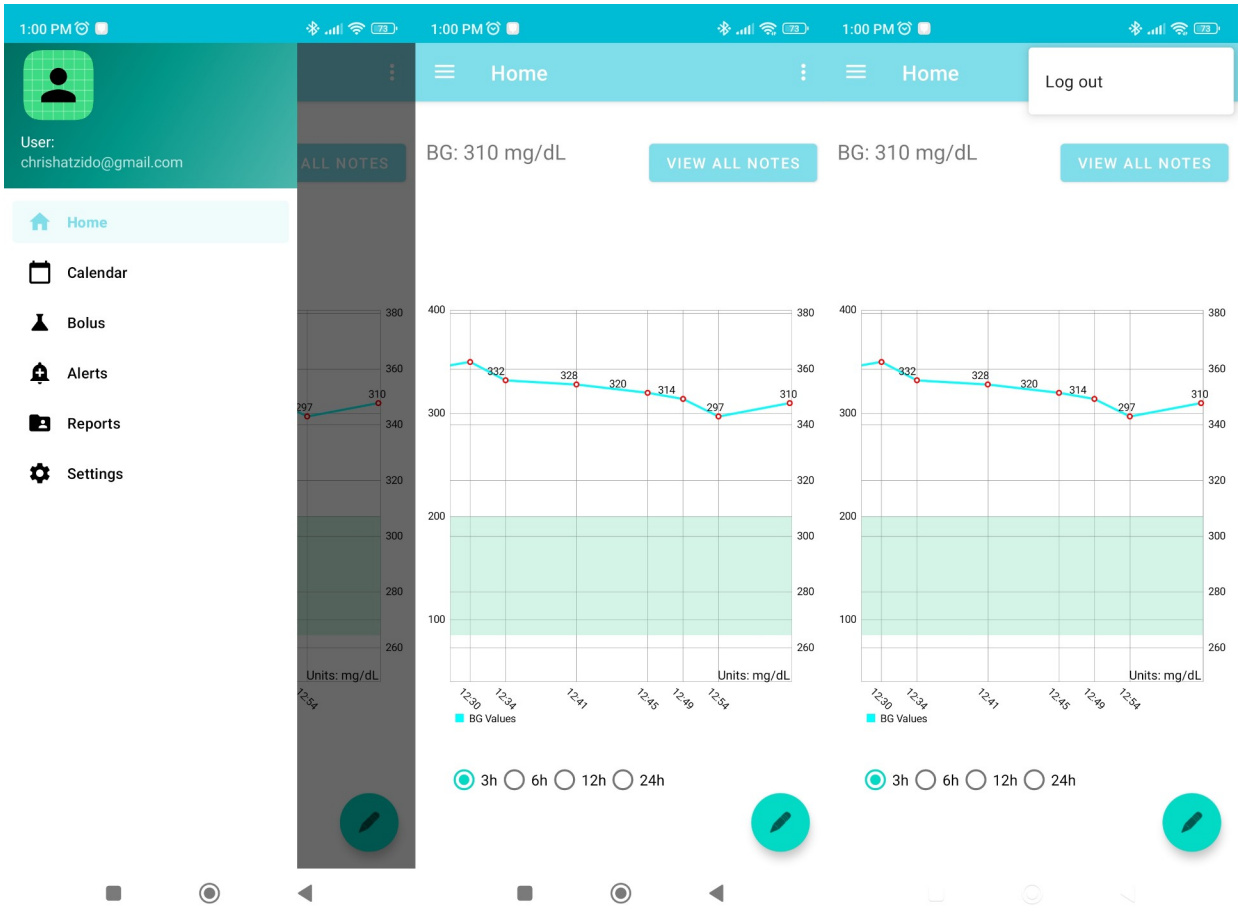
3.2.4 Αρχική οθόνη

Η Αρχική οθόνη της εφαρμογής με τίτλο Home περιέχει πληροφορίες σχετικά με την τελευταία μέτρηση γλυκόζης αίματος που έλαβε η εφαρμογή όπως επίσης και ένα αυτόματα ανανεώσιμο διαδραστικό διάγραμμα που περιέχει τις τιμές γλυκόζης των τελευταίων τριών, έξι, δώδεκα ή εικοσιτεσσάρων ωρών ανάλογα με το χρονικό διάστημα που έχει επιλέξει ο χρήστης. Επιπλέον παρέχει τη δυνατότητα να ανακατευθυνθεί ο χρήστης στην Οθόνη Προβολής Σημειώσεων (υποενότητα 3.2.10).

Η οθόνη Home είναι η προεπιλεγμένα αρχική οθόνη της εφαρμογής και εμφανίζεται κάθε φορά που ο χρήστης ή το λειτουργικό σύστημα ανοίγει την εφαρμογή εφόσον έχει πραγματοποιηθεί κατά το παρελθόν είσοδος του χρήστη η οποία δεν ακολουθείται από έξοδο. Επίσης, ο χρήστης μπορεί να ανακατευθυνθεί σε αυτή μέσω του drawer menu της εφαρμογής.

Η οθόνη Home είναι μια από τις βασικές οθόνες της εφαρμογής και διαθέτει επικεφαλίδα στο πάνω μέρος και πλωτό κουμπί στο κάτω μέρος της. Η Οθόνη αποτελείται από τέσσερα στοιχεία. Αρχικά στο πάνω αριστερά μέρος της οθόνης και κάτω από την επικεφαλίδα υπάρχει ένα TextView στο οποίο προβάλλεται η τιμή της τελευταίας μέτρησης γλυκόζης αίματος που έλαβε η εφαρμογή. Το πεδίο αυτό αρχικά προβάλλει την τιμή γλυκόζης αίματος με την πιο πρόσφατη χρονοσφραγίδα (υποενότητα 3.1.1), και ενημερώνεται αυτόματα κάθε φορά που η εφαρμογή δέχεται μια νέα τιμή. Δεξιά από το TextView προβολής της τελευταίας τιμής γλυκόζης αίματος υπάρχει ένα ψηφιακό κουμπί με τίτλο View All Notes, όταν αυτό

ενεργοποιείται, ο χρήστης ανακατευθύνεται στην Οθόνη προβολής σημειώσεων (υποενότητα 3.2.10). Κάτω από το κουμπί View All Notes και το TextView υπάρχει ένα διάγραμμα στο οποίο αποτυπώνονται οι τιμές της γλυκόζης αίματος που αποθήκευσε η εφαρμογή σε συνάρτηση με το χρόνο. Στον κάθετο άξονα του διαγράμματος αποτυπώνονται οι τιμές γλυκόζης αίματος ενώ στον οριζόντιο ο χρόνος. Το σύνολο τιμών του διαγράμματος δημιουργείται διαβάζοντας τις Τιμές BG (υποενότητα 3.1.1) από την τοπική βάση δεδομένων της εφαρμογής και διατηρώντας το υποσύνολο των Τιμών BG η χρονοσφραγίδα των οποίων είναι εντός του χρονικού ορίου που έχει επιλέξει ο χρήστης. Το σύνολο τιμών του διαγράμματος (Τιμή BG, Χρονοσφραγίδα) ενημερώνεται αυτόματα κάθε φορά που η εφαρμογή λαμβάνει μια νέα τιμή. Οι τιμές και οι μονάδες μέτρησης γλυκόζης αίματος του κάθετου άξονα του διαγράμματος όπως επίσης και η τελευταία τιμή γλυκόζης του TextView προβάλλονται σύμφωνα με τις μονάδες μέτρησης γλυκόζης αίματος που έχει επιλέξει ο χρήστης στην επιλογή Units της Οθόνης Ρυθμίσεων (υποενότητα 3.2.9) με προεπιλεγμένη την επιλογή mg/dL. Η εφαρμογή παρέχει τη δυνατότητα στο χρήστη να ορίσει το χρονικό διάστημα που αποτυπώνεται στο διάγραμμα μέσω μιας ομάδας κουμπιών μοναδικής επιλογής (radio group) που περιέχει τις επιλογές τριών, έξι, δώδεκα ή εικοσιτεσσάρων ωρών. Αυτή η ομάδα κουμπιών βρίσκεται ακριβώς κάτω από το διάγραμμα. Το διάγραμμα δεν υποστηρίζει λειτουργία zoom in και zoom out, διατηρώντας σταθερή την κλίμακα προβολής. Ωστόσο, επιτρέπει την οριζόντια μετατόπιση (scrolling), δίνοντας τη δυνατότητα στον χρήστη να περιηγηθεί σε παλαιότερες τιμές μετακινώντας το διάγραμμα προς τα δεξιά. Οι τιμές του διαγράμματος συνδέονται με ευθείες γραμμές από την παλαιότερη στην αμέσως επόμενη κ.ο.κ.



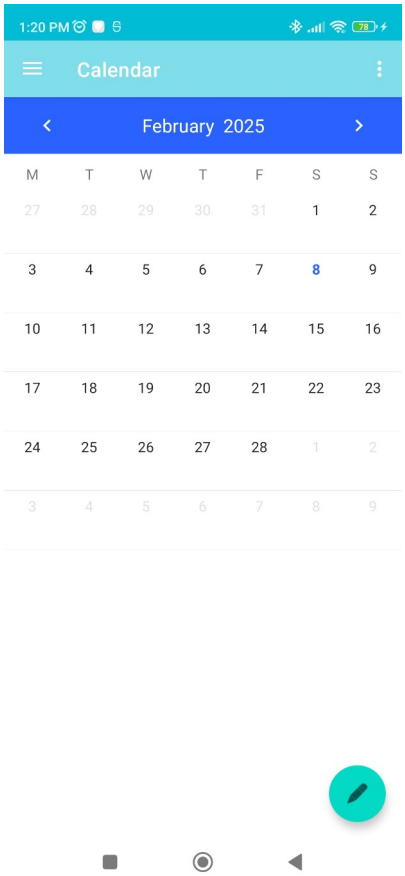
Εικόνα 3.3 Στιγμιότυπα της Αρχικής Οθόνης και των Drawer και Overflow Menus.

3.2.5 Οθόνη Ημερολογίου

Μια ακόμα βασική οθόνη της εφαρμογής είναι η Οθόνη Ημερολογίου με τίτλο Calendar. Η οθόνη Calendar προβάλλεται αποκλειστικά και μόνο αν ο χρήστης επιλέξει την αντίστοιχη επιλογή από το drawer menu. Η οθόνη Calendar ανήκει στις βασικές οθόνες της εφαρμογής και ως τέτοια διαθέτει επικεφαλίδα και πλωτό κουμπί.

Μέσω της οθόνης Calendar προσφέρεται στο χρήστη η δυνατότητα να ανατρέξει σε παρελθοντικές ημερομηνίες και να λάβει πληροφορίες. Πιο αναλυτικά ο χρήστης επιλέγει μια ημερομηνία μέσω ενός διαδραστικού ημερολογίου που διαθέτει η οθόνη. Στη συνέχεια, η εφαρμογή εμφανίζει ένα αναδυόμενο παράθυρο στο πάνω μέρος του οποίου αναγράφεται η επιλεγμένη ημερομηνία. Κάτω από αυτό υπάρχει μια λίστα με όλες τις τιμές γλυκόζης αίματος που έλαβε η εφαρμογή την συγκεκριμένη ημερομηνία συνοδευόμενες από την αντίστοιχη χρονοσφραγίδα, οι τιμές τις λίστας προβάλλονται από την παλαιότερη έως την πιο πρόσφατη. Κάτω από τη λίστα υπάρχουν τρία ψηφιακά κουμπιά με τίτλο Notes, Close και Graph. Το κουμπί Notes, όταν ενεργοποιείται, αντικαθιστά το τρέχον αναδυόμενο παράθυρο με ένα νέο που περιέχει όλα τα δεδομένα σημειώσεων (υποενότητα 3.1.3) που έχει αποθηκεύσει η εφαρμογή την συγκεκριμένη ημερομηνία. Στο συγκεκριμένο παράθυρο αρχικά προβάλλεται μια λίστα με χρονοσφραγίδες και όταν ο χρήστης επιλέξει μια χρονοσφραγίδα εμφανίζεται το κείμενο της αντίστοιχης σημείωσης. Το κουμπί Close κλείνει το αναδυόμενο παράθυρο και η εφαρμογή προβάλλει την Οθόνη ημερολογίου. Το κουμπί Graph αντικαθιστά το τρέχον αναδυόμενο παράθυρο με ένα νέο το οποίο περιέχει ένα διάγραμμα (Τιμών BG, χρόνου) αντίστοιχο με αυτό της οθόνης Home (υποενότητα 3.2.4) που όμως περιέχει το σύνολο τιμών BG η χρονοσφραγίδα των οποίων βρίσκεται εντός της επιλεγμένης ημέρας.

Η Οθόνη Ημερολογίου αποτελείται από ένα στοιχείο διεπαφής χρήστη. Είναι ένα στοιχείο τύπου CalendarView της βιβλιοθήκης Material (υποενότητα 4.2.2). Το συγκεκριμένο στοιχείο είναι ένα ψηφιακό ημερολόγιο στην επικεφαλίδα του οποίου μπορεί να γίνει επιλογή του μήνα και του έτους. Ως προεπιλογή στην επικεφαλίδα του CalendarView έχει τεθεί ο τρέχον μήνας του τρέχοντος έτους όπως τον λαμβάνει η εφαρμογή από το λειτουργικό σύστημα. Κάτω από την επικεφαλίδα υπάρχουν διαθέσιμες ως επιλογές όλες οι ημέρες του μήνα που έχει επιλεγεί στην επικεφαλίδα ενώ η τρέχουσα ημέρα έχει διαφορετικό χρώμα από τις υπόλοιπες. Κάθε φορά που ο χρήστης επιλέγει κάποια ημέρα η εφαρμογή δημιουργεί το αναδυόμενο παράθυρο που περιεγράφηκε στην προηγούμενη παράγραφο.



Εικόνα 3.4 Στιγμιότυπο οθόνης κατά την προβολή της Οθόνης Ημερολογίου.

3.2.6 Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης

Η Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης (Bolus), είναι μια από τις βασικές οθόνες της εφαρμογής. Εμφανίζεται μόνο εάν ο χρήστης επιλέξει να μεταφερθεί σε αυτή από του drawer menu. Μέσω αυτής, παρέχεται στο χρήστη η δυνατότητα να υπολογίσει ακριβώς την ποσότητα γευματικής ή διορθωτικής ινσουλίνης που χρειάζεται να εγχύσει, ώστε να βοηθήσει στο μεταβολισμό κάποιου γεύματος ή να διορθώσει κάποια υψηλή τιμή σακχάρου. Αυτό επιτυγχάνεται μέσω του υπολογισμού που περιγράφεται στην υποενότητα 3.1.4. Πιο συγκεκριμένα ο χρήστης εισάγει την ποσότητα υδατανθράκων (σε γραμμάρια) που πρόκειται να καταναλώσει αν πρόκειται για γευματική ινσουλίνη και στη συνέχεια η εφαρμογή αφού ανακτήσει όλες τις απαραίτητες τιμές από την τοπική βάση δεδομένων και το αρχείο Shared Preferences, υπολογίζει την ποσότητα ινσουλίνης που χρειάζεται να εγχυθεί προκειμένου η συγκέντρωση γλυκόζης να παραμείνει ή επανέλθει στα επιθυμητά επίπεδα που έχει ορίσει ο χρήστης στις ρυθμίσεις. Στη συνέχεια, προβάλλεται στο χρήστη η ποσότητα ινσουλίνης που χρειάζεται να εγχύσει καθώς και όλες οι τιμές που χρησιμοποιήθηκαν στον υπολογισμό της. Ακολούθως, παρασκηνιακά και ασύγχρονα, η εφαρμογή δημιουργεί τις απαραίτητες εγγραφές που αφορούν το συμβάν στην τοπική βάση δεδομένων. Οι εγγραφές αυτές περιλαμβάνουν εγγραφή νέου συμβάντος με τύπο TYPE_BOLUS έτσι ώστε να καταγραφεί η ποσότητα ινσουλίνης Bolus που εγχύθηκε, η χρονική στιγμή έγχυσης καθώς επίσης και η δημιουργία εγγραφής των υδατανθράκων που καταναλώθηκαν σε περίπτωση που το Bolus αφορούσε γευματική ινσουλίνη.

Η Οθόνη Bolus, διαθέτει επικεφαλίδα, πλωτό κουμπί και αποτελείται από 4 στοιχεία διεπαφής χρήστη. Στο επάνω μέρος αυτής υπάρχει ένα στοιχείο προβολής κειμένου (TextView) στο οποίο πριν τον υπολογισμό, αναγράφεται μήνυμα οδηγιών προς το χρήστη ενώ μετά τον υπολογισμό αναγράφονται όλες οι τιμές των παραγόντων που χρησιμοποιήθηκαν για τον υπολογισμό. Κάτω από αυτό, βρίσκονται δύο παράλληλα στοιχεία. Αριστερά, υπάρχει ένα στοιχείο εισαγωγής κειμένου (TextInputLayout) στο οποίο ο χρήστης εισάγει την ποσότητα υδατανθράκων που πρόκειται να καταναλωθούν, ενώ παράλληλα και δεξιά από αυτό υπάρχει ένα δεύτερο στοιχείο TextView το οποίο πριν τον υπολογισμό αναγράφει το μήνυμα “Bolus Intake: “ ενώ μετά τον υπολογισμό, σε αυτό το στοιχείο προβάλλεται η ποσότητα ινσουλίνης που χρειάζεται να εγχυθεί καθώς επίσης και οι μονάδες μέτρησης της. Τέλος, κάτω από τα δύο παράλληλα στοιχεία βρίσκεται ένα ψηφιακό κουμπί που αναγράφει “Calculate Bolus” και λειτουργεί ως κουμπί υποβολής προκειμένου να πραγματοποιηθεί ο υπολογισμός.

3.2.7 Οθόνη Διαχείρισης Συναγερμών

Η Οθόνη Διαχείρισης Συναγερμών (Alerts), είναι μια από τις βασικές οθόνες της εφαρμογής και διαθέτει επικεφαλίδα και πλωτό κουμπί. Εμφανίζεται αποκλειστικά και μόνο αν το επιλέξει ο χρήστης μέσω του drawer menu. Εντός της οθόνης, προβάλλονται όλα τα δεδομένα των ενεργών συναγερμών (υποενότητα 3.1.2) και δίνεται η δυνατότητα στο χρήστη να δημιουργεί νέους συναγερμούς και να επεξεργάζεται ή διαγράφει ήδη υπάρχοντες. Αποτελείται από τρία στοιχεία διεπαφής χρήστη. Στο πάνω μέρος της οθόνης υπάρχει ένα TextView, ακριβώς από κάτω βρίσκεται ένα στοιχείο προβολής λίστας (listView), ενώ ακόμη πιο κάτω υπάρχει ένα ψηφιακό κουμπί. Το TextView είναι η επικεφαλίδα της λίστας ενεργών συναγερμών που υπάρχει από κάτω του και αναγράφει “Active Alerts”. Το listView, είναι μια λίστα που περιέχει όλους τους ενεργούς συναγερμούς που έχουν αποθηκευτεί στην τοπική βάση δεδομένων της εφαρμογής. Για κάθε συναγερμό της λίστας, προβάλλεται το ζεύγος δεδομένων (τιμή BG, τελεστής) που τον αναπαριστούν. Η λίστα ενημερώνεται αυτόματα μετά από οποιαδήποτε αλλαγή στα δεδομένα της βάσης. Κάτω από τη λίστα, υπάρχει ένα ψηφιακό κουμπί που αναγράφει ADD. Μέσω του τελευταίου δίνεται η δυνατότητα να αποθηκευτούν νέοι συναγερμοί στην τοπική βάση δεδομένων.

Για να προσθέσει ο χρήστης ένα νέο συναγερμό χρειάζεται αρχικά να πατήσει το κουμπί ADD. Στη συνέχεια, εμφανίζεται ένα αναδυόμενο παράθυρο αποθήκευσης νέου συναγερμού, μέσω του οποίου αρχικά ορίζονται τα δεδομένα του νέου συναγερμού. Το αναδυόμενο παράθυρο καλύπτει μέρος της οθόνης και περιέχει έναν επιλογέα τελεστή, ένα πεδίο εισαγωγής αριθμών, ένα κουμπί ακύρωσης (CANCEL) και ένα κουμπί αποθήκευσης (SAVE). Ο επιλογέας μπορεί να έχει μία ενεργή επιλογή και περιέχει τις επιλογές άνω ορίου(over) και κάτω ορίου (under) οι οποίες αντιπροσωπεύουν τα δεδομένα τελεστή του νέου συναγερμού. Στο πεδίο εισαγωγής αριθμών εισάγεται από το χρήστη η τιμή γλυκόζης του συναγερμού. Εάν ο χρήστης πατήσει το κουμπί αποθήκευσης, αρχικά ελέγχεται εάν το πεδίο εισαγωγής αριθμών περιέχει έγκυρη τιμή BG. Εφόσον το πεδίο είναι κενό ή περιέχει κάποια μη θετική τιμή, η εφαρμογή εμφανίζει μήνυμα σφάλματος και το αναδυόμενο παράθυρο παραμένει ανοικτό. Σε αντίθετη περίπτωση, το αναδυόμενο παράθυρο κλείνει και η εφαρμογή αποθηκεύει παρασκηνιακά και ασύγχρονα τον νέο συναγερμό στην τοπική βάση δεδομένων.

Όταν ολοκληρωθεί η αποθήκευση του νέου συναγερμού η λίστα ενεργών συναγερμών ενημερώνεται αυτόματα. Εάν ο χρήστης πατήσει το κουμπί ακύρωσης το αναδυόμενο παράθυρο κλείνει και η εφαρμογή προβάλλει ξανά την Οθόνη Διαχείρισης Συναγερμών.

Για να γίνει επεξεργασία ή διαγραφή κάποιου ενεργού συναγερμού, ο χρήστης θα πρέπει αρχικά να πατήσει παρατεταμένα στην αντίστοιχη εγγραφή της λίστας. Το παρατεταμένο πάτημα σε κάποιο συναγερμό οδηγεί στην εμφάνιση ενός αναδυόμενου μενού επιλογών κάτω από την εγγραφή, το οποίο περιέχει τις επιλογές επεξεργασία (Edit) και διαγραφή (Delete). Εάν επιλεχθεί διαγραφή τότε η εφαρμογή διαγράφει παρασκηνιακά και συναγερμό από τη βάση. Εάν επιλεχθεί επεξεργασία τότε η εφαρμογή εμφανίζει ένα αναδυόμενο παράθυρο ακριβώς όμοιο με το παράθυρο αποθήκευσης νέου συναγερμού, με τη διαφορά ότι εάν ο χρήστης πατήσει το κουμπί SAVE, η εφαρμογή θα πραγματοποιήσει παρασκηνιακά και ασύγχρονα, ενημέρωση των δεδομένων του ήδη αποθηκευμένου στην βάση δεδομένων συναγερμού. Αφού ολοκληρωθούν οι διαδικασίες διαγραφής ή επεξεργασίας των συναγερμών, η λίστα ενεργών συναγερμών ενημερώνεται αυτόματα.

3.2.8 Οθόνη Δημιουργίας Αναφορών

Η Οθόνη Δημιουργίας αναφορών, επιτρέπει στους χρήστες να δημιουργούν και αποθηκεύουν αρχεία αναφορών τύπου PDF. Υπάρχουν δύο διαφορετικοί τύποι αναφορών που οι χρήστες μπορούν να δημιουργήσουν. Ο πρώτος τύπος αναφοράς, είναι αναφορά τιμών γλυκόζης και περιέχει αναλυτικά όλες τις τιμές γλυκόζης και τις αντίστοιχες χρονικές στιγμές στις οποίες ελήφθησαν και αποθηκεύτηκαν στην εφαρμογή, σε μια συγκεκριμένη χρονική περίοδο. Ο δεύτερος τύπος αναφοράς, είναι αναφορά συμβάντων και περιέχει αναλυτικά όλα τα δεδομένα συμβάντων (τύπος συμβάντος, χρονική στιγμή συμβάντος, μήνυμα περιγραφής συμβάντος) που κατέγραψε η εφαρμογή, σε μια συγκεκριμένη χρονική περίοδο. Η χρονική περίοδος κάθε τύπου αναφοράς επιλέγεται από το χρήστη και μπορεί να είναι κατ’ ελάχιστο μια ημερολογιακή ημέρα. Αφού ο χρήστης επιλέξει χρονική περίοδο και τύπο αναφοράς, η εφαρμογή δημιουργεί το αρχείο αναφοράς με τύπο PDF και στη συνέχεια το αποθηκεύει στον κοινόχρηστο φάκελο “Έγγραφα” (Documents) της συσκευής του.

Η Οθόνη Δημιουργίας αναφορών, ανήκει στις βασικές οθόνες της εφαρμογής, διαθέτει επικεφαλίδα, πλωτό κουμπί και αποτελείται από επτά στοιχεία διεπαφής χρήστη. Προκειμένου να παραχθεί ένα νέο αρχείο αναφοράς, χρειάζεται αρχικά να επιλεχθεί η χρονική περίοδος την οποία θα αφορά. Η χρονική περίοδος μπορεί να είναι κατ’ ελάχιστο μια ημερολογιακή ημέρα ενώ δεν υπάρχει κάποιος περιορισμός άνω ορίου. Για την επιλογή της χρονικής περιόδου χρησιμοποιούνται πέντε στοιχεία διεπαφής χρήστη. Στο πάνω μέρος της οθόνης υπάρχει ένα αρχείο προβολής κειμένου (TextView) και αναγράφει “Select Period”. Κάτω από αυτό υπάρχουν στοιχισμένα κάθετα, δύο ζεύγη στοιχείων TextView τα οποία εξυπηρετούν στην επιλογή των ημερομηνιών έναρξης και λήξης της χρονικής περιόδου της αναφοράς. Το πρώτο ζεύγος TextView αφορά την ημερομηνία εκκίνησης, το πρώτο στοιχείο αναγράφει την ένδειξη “From” ενώ δίπλα σε αυτό υπάρχει ένα TextView που αναγράφει “Click to select Date”. Εάν ο χρήστης πατήσει στην περιοχή του δεύτερου TextView, εμφανίζεται ένα αναδυόμενο παράθυρο επιλογής ημερομηνίας. Το αναδυόμενο παράθυρο είναι ένα παράθυρο διαλόγου τύπου DatePickerDialog και ανήκει στο πακέτο app του Android. Το DatePickerDialog, είναι παραμετροποιήσιμο και αποτελείται από τρία μέρη, Στο πάνω μέρος υπάρχει ο τίτλος του

παραθύρου, σαν τίτλος έχει επιλεχθεί να προβάλλεται η τρέχουσα ημερομηνία. Κάτω από τον τίτλο, υπάρχει ένα στοιχείο επιλογέα ημερομηνίας (Date Picker). Ο Date Picker, είναι το κεντρικό στοιχείο του διαλόγου και αποτελείται από μια κυλιόμενη λίστα (Spinner) στην οποία επιλέγεται ο μήνας και το έτος της ημερομηνίας, ενώ κάτω από αυτό υπάρχει μια διάταξη ημερολογίου (Calendar View), όπου εμφανίζεται ένα πλήρες ημερολόγιο του μήνα του έτους που είναι ανά πάσα στιγμή επιλεγμένος στο Spinner. Η τελική επιλογή, πραγματοποιείται πατώντας απευθείας πάνω στην ημερομηνία που υπάρχει στο Calendar View. Κάτω από τον επιλογέα ημερομηνίας, υπάρχουν δύο κουμπιά, ένα για επιβεβαίωση της επιλογής (OK), το οποίο κλείνει το διάλογο και επιστρέφει την επιλεγμένη ημερομηνία η οποία και προβάλλεται στο χρήστη, και ένα για ακύρωση (Cancel), το οποίο οδηγεί στο κλείσιμο του διαλόγου χωρίς να αλλάξει κάτι. Η εμφάνιση και η διάταξη του DatePickerDialog μπορεί να διαφέρει ανάλογα με την έκδοση του Android στην οποία εκτελείται η εφαρμογή. Σε παλαιότερες εκδόσεις, χρησιμοποιούνται περισσότερο οι κυλιόμενες λίστες, ενώ σε νεότερες, το ημερολόγιο είναι η προεπιλεγμένη μέθοδος επιλογής ημερομηνίας. Αφού επιλεχθεί κάποια ημερομηνία μέσω του αναδυόμενου παραθύρου, το παράθυρο κλείνει και το TextView που προηγουμένως έγραφε “Click to select Date” πλέον αναγράφει την επιλεχθείσα ημερομηνία. Με ακριβώς τον ίδιο τρόπο επιλέγεται η τελική ημερομηνία της χρονικής περιόδου μέσω του δεύτερου ζεύγους TextView με τη διαφορά ότι το πρώτο TextView του δεύτερου ζεύγους αναγράφει “Το”. Το τελικό βήμα για τη δημιουργία και αποθήκευση της εφαρμογής είναι να επιλεχθεί ο τύπος της. Αυτό επιτυγχάνεται μέσω δυο ψηφιακών κουμπιών. Τα κουμπιά βρίσκονται κάτω από τα TextView επιλογής χρονικής περιόδου και είναι στοιχισμένα παράλληλα μεταξύ τους και αναγράφουν “BG REPORT” και “BOLUS & ALERTS REPORT” αντίστοιχα. Εφόσον ο χρήστης έχει επιλέξει κάποια έγκυρη χρονική περίοδο και πατήσει οποιοδήποτε από τα κουμπιά, η εφαρμογή παράγει και αποθηκεύει παρασκηνιακά και ασύγχρονα, το αντίστοιχο αρχείο αναφοράς. Με τη δημιουργία κάποιου αρχείου αναφοράς, η επιλεγμένη χρονική περίοδος παραμένει, ώστε ο χρήστης να μπορεί να δημιουργήσει και άλλο τύπο αναφοράς για την συγκεκριμένη χρονική περίοδο.

3.2.9 Οθόνη Ρυθμίσεων

Η Οθόνη Ρυθμίσεων, είναι η τελευταία από τις βασικές οθόνες της εφαρμογής, διαθέτει επικεφαλίδα αλλά δεν διαθέτει πλωτό κουμπί δημιουργίας σημειώσεων. Ο λόγος απουσίας πλωτού κουμπιού στην εν λόγω οθόνη είναι ότι η Οθόνη Ρυθμίσεων αποτελείται από πολλά στοιχεία διεπαφής χρήστη τα οποία καθορίζουν διάφορες προεπιλογές και ρυθμίσεις. Η ύπαρξη πλωτού κουμπιού πάνω από αυτές θα λειτουργούσε αρνητικά στην εμπειρία του χρήστη καθώς και στην κατανόηση των στοιχείων διεπαφής χρήστη που αφορούν τις ρυθμίσεις. Μέσω της Οθόνης Ρυθμίσεων, ορίζονται τιμές και παράμετροι που επηρεάζουν ένα μεγάλο αριθμό παραγόντων οι οποίοι βρίσκονται σε διάφορα σημεία της εφαρμογής. Για το λόγο αυτό, έχει επιλεχθεί η Οθόνη Ρυθμίσεων να έχει κατακόρυφη σειριακή δομή όπου οι επιμέρους ρυθμίσεις διαθέτουν εξατομικευμένη επικεφαλίδα, περιοχή εισαγωγής τιμών και είναι στοιχισμένες κάθετα. Επιπλέον, οι ρυθμίσεις του μενού κατηγοριοποιούνται ανάλογα με το σκοπό που εξυπηρετούν και τα δεδομένα που αφορούν. Οι κατηγορίες ρυθμίσεων είναι: Ρυθμίσεις Τιμών Γλυκόζης, Ρυθμίσεις Συναγερμών και Ρυθμίσεις Θεραπείας. Η κάθε κατηγορία έχει τη δική της Επικεφαλίδα και όλες οι επιμέρους ρυθμίσεις τιμών που

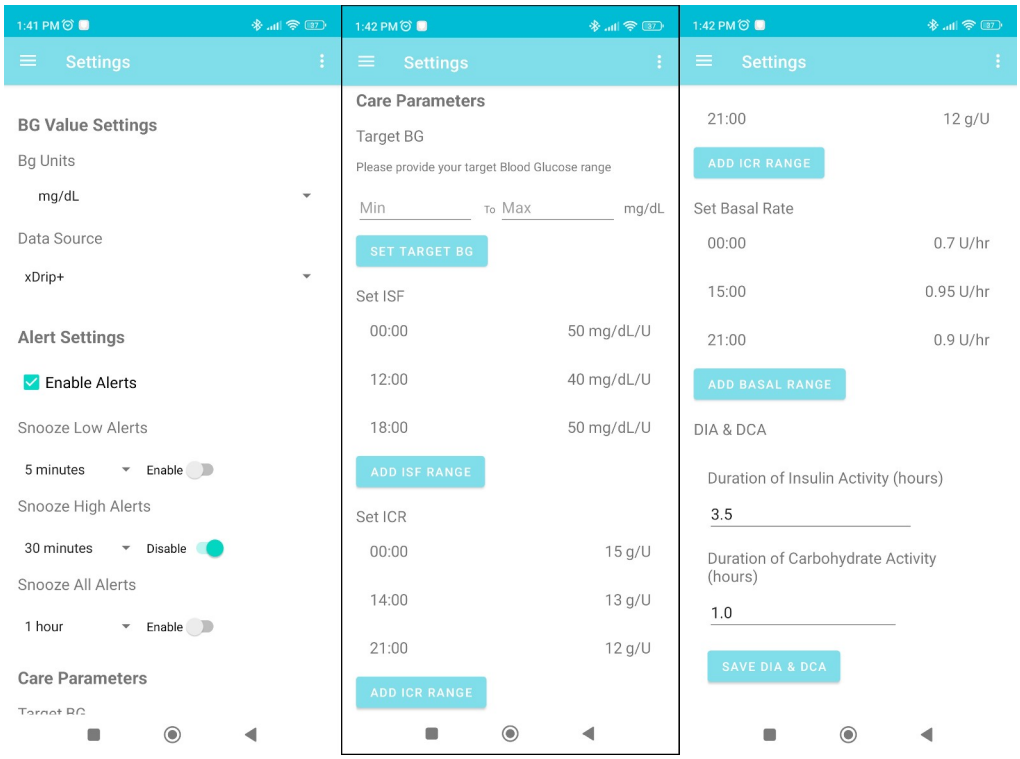
ανήκουν στην κατηγορία βρίσκονται κάτω από αυτήν την επικεφαλίδα. Οποτεδήποτε κάποια ρύθμιση αλλάξει, η εφαρμογή αποθηκεύει άμεσα τα νέα δεδομένα και ενημερώνει παρασκηνιακά όλες τις λειτουργίες που επηρεάζονται από την αλλαγή.

Η κατηγορία Ρυθμίσεις Τιμών Γλυκόζης (BG Value Settings), έχει δύο επιμέρους ρυθμίσεις, οι οποίες είναι: Μονάδες μέτρησης γλυκόζης αίματος (Bg Units) και Πηγή λήψης μονάδων γλυκόζης (Data Source). Η ρύθμιση Bg Units ορίζει τις μονάδες μέτρησης με βάση τις οποίες προβάλλονται οι τιμές BG σε όλη την εφαρμογή (διαγράμματα, λίστες, αναφορές, ειδοποιήσεις και συναγερμούς). Ο χρήστης μπορεί να επιλέξει την επιθυμητή μονάδα μέτρησης από μια αναπτυσσόμενη λίστα (drop down list). Η λίστα αυτή εμπεριέχει τις διαθέσιμες μονάδες μέτρησης γλυκόζης αίματος που υποστηρίζει η εφαρμογή και είναι είτε mg/dL είτε mmol/L. Η επιλογή αυτή αφορά μόνο τον τρόπο που παρουσιάζονται οι τιμές γλυκόζης αίματος και όχι τον τρόπο που αποθηκεύονται, καθώς η εφαρμογή αποθηκεύει πάντα τις τιμές γλυκόζης αίματος ως mg/dL και εφόσον ο χρήστης επιλέξει mmol/L πραγματοποιείται μετατροπή της τιμής κατά την προβολή της. Η ρύθμιση έχει ως προεπιλογή mg/dL. Η ρύθμιση Data Source αφορά τον τρόπο με τον οποίο η εφαρμογή θα λαμβάνει νέες τιμές γλυκόζης αίματος στο μέλλον. Οι επιλογές είναι δύο, η πρώτη είναι να λαμβάνει η εφαρμογή τιμές από την εφαρμογή xDrip$^+$ και η δεύτερη είναι να παράγονται από την εφαρμογή τοπικά εικονικές τιμές. Όμοια με την προηγούμενη ρύθμιση, ο επιθυμητός τρόπος λήψης τιμών επιλέγεται από μια drop down list. Οι επιλογές της λίστας είναι xDrip$^+$ και Generated Locally και ως προεπιλογή έχει οριστεί η xDrip$^+$.

Η κατηγορία Ρυθμίσεις Συναγερμών, έχει τέσσερεις επιμέρους ρυθμίσεις, οι οποίες είναι: Ενεργοποίηση Συναγερμών (Enable Alerts), Προσωρινή σίγαση συναγερμών χαμηλής γλυκόζης (Snooze Low Alerts), Προσωρινή σίγαση συναγερμών υψηλής γλυκόζης (Snooze High Alerts) και Προσωρινή σίγαση όλων των συναγερμών (Snooze All Alerts). Η ρύθμιση Enable Alerts είναι της μορφής πλαισίου ελέγχου και αφορά την γενική ενεργοποίηση ή απενεργοποίηση της λειτουργίας συναγερμών (υποενότητα 3.2.7). Ως προεπιλογή οι συναγερμοί είναι ενεργοί. Οι επόμενες τρεις ρυθμίσεις αφορούν σίγαση συναγερμών και έχουν όμοια δομή. Για να γίνει σίγαση κάποιας κατηγορίας ή όλων των συναγερμών χρειάζεται να επιλεχθεί το χρονικό διάστημα κατά το οποίο θα ισχύει η σίγαση, έτσι για κάθε ρύθμιση σίγασης υπάρχει μια drop down list με τα διαθέσιμα χρονικά διαστήματα σίγασης καθώς επίσης και ένας ψηφιακός διακόπτης (Switch) όπου ορίζει εάν η σίγαση για το επιλεγμένο χρονικό διάστημα είναι ενεργή ή όχι. Η κάθε κατηγορία σίγασης έχει διαφορετικά διαθέσιμα χρονικά διαστήματα και προεπιλογή αυτών στην αντίστοιχη αναπτυσσόμενη λίστα. Η σίγαση συναγερμών χαμηλής γλυκόζης έχει ως επιλογές 5, 10 ,15, 20 και 30 λεπτά και ως προεπιλογή τα 5 λεπτά. Η σίγαση συναγερμών υψηλής γλυκόζης έχει ως επιλογές 0.5, 1 ,1.5, 2, 2.5 και 3 ώρες και ως προεπιλογή τη μισή ώρα. Η σίγαση όλων των συναγερμών, έχει ως επιλογές από μια έως 24 ώρες (με βήμα 1 ώρα) και ως προεπιλογή τη 1 ώρα.

Η κατηγορία Ρυθμίσεις Θεραπείας, αφορά τιμές παραμέτρων που είναι σχετικές με την λήψη ινσουλίνης και υδατανθράκων και έχει πέντε επιμέρους ρυθμίσεις. Οι ρυθμίσεις αυτές είναι Ρύθμιση εύρους στόχου γλυκόζης (Target BG), Ρύθμιση συντελεστή ευαισθησίας ινσουλίνης (Set ISF), Ρύθμιση συντελεστή αναλογίας ινσουλίνης προς υδατάνθρακες(Set ICR), Ρύθμιση βασικού ρυθμού (Set Basal Rate) και Ρύθμιση Διάρκειας δράσης Ινσουλίνης & Υδατανθράκων (DIA & DCA). Η ρύθμιση Target BG, ορίζει ποιο είναι το επιθυμητό από το

χρήστη εύρος τιμών γλυκόζης και επηρεάζει το διάγραμμα της Αρχικής Οθόνης (υποενότητα 3.2.4) καθώς επίσης και τον υπολογισμό Bolus (υποενότητα 3.2.6). Το εύρος τιμών τίθεται από το χρήστη μέσω δύο στοιχείων εισαγωγής αριθμών και έχει περιορισμούς τιμών. Το εύρος δεν μπορεί να είναι εκτός του διαστήματος [40,400] και η τιμή κάτω ορίου θα πρέπει να είναι μικρότερη της τιμής άνω ορίου. Όταν ο χρήστης θέσει τις επιθυμητές οριακές τιμές του εύρους μπορεί να πατήσει το κουμπί SET TARGET BG που βρίσκεται κάτω από τα πεδία εισαγωγής των ανωτέρω τιμών ώστε να λάβει εφαρμογή η νέα ρύθμιση. Ως προεπιλογή έχει οριστεί το διάστημα [70,180]. Οι ρυθμίσεις Set ISF, Set ICR και Set Basal Rate, έχουν όμοια μορφή και επηρεάζουν τον υπολογισμό που περιγράφεται στην υποενότητα 3.1.4 και λαμβάνει χώρα στη Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης (υποενότητα 3.2.6). Τα μεγέθη ISF, ICR και Basal Rate, έχουν την ιδιαιτερότητα ότι αλλάζουν κατά τη διάρκεια της ημέρας, δίνονται στο χρήστη έπειτα από επικοινωνία με το γιατρό του και δεν αλλάζουν συχνά. Επειδή για κάθε μια από αυτές τις ρυθμίσεις οι χρήστες χρειάζεται να παρέχουν διάφορες τιμές για κάθε χρονική περίοδο της ημέρας οι οποίες θα καλύπτουν το εικοσιτετράωρο, το πεδίο ορισμού τιμών τους αποτελείται από μια λίστα που περιέχει όλα τα διαστήματα για τα οποία ο χρήστης έχει καταχωρίσει τιμή συνοδευόμενα από την αντίστοιχη τιμή, τις μονάδες μέτρησης του μεγέθους και ένα ψηφιακό κουμπί που επιτρέπει στους χρήστες να ορίσουν νέο διάστημα. Πατώντας το αντίστοιχο κουμπί προσθήκης διαστήματος, εμφανίζεται ένα αναδυόμενο παράθυρο στο οποίο ορίζονται η ώρα έναρξης του διαστήματος μέσω στοιχείου επιλογέα ώρας (TimePicker) καθώς επίσης και η εκάστοτε τιμή ISF, ICR ή Basal Rate μέσω πεδίων εισαγωγής αριθμών. Η ώρα λήξης του διαστήματος προσαρμόζεται αυτόματα και έχει τιμή, την ώρα έναρφης του επόμενου διαστήματος. Εφόσον το νέο διάστημα που όρισε ο χρήστης επικαλύπτει χρονικά άλλα ήδη ορισμένα διαστήματα, η εφαρμογή επιλύει την επικάλυψη θέτοντας πάντα την νέα τιμή στις ώρες επικάλυψης των διαστημάτων. Τα χρονικά διαστήματα που έχουν οριστεί, προβάλλονται από την εφαρμογή ταξινομημένα με βάση την ώρα έναρξης (από 00:00 έως 23:00). Η ρύθμιση ISF έχει ως προεπιλογή το διάστημα 00:00 – 23:59 και τιμή 50 mg/dL/U, η ρύθμιση ICR έχει ως προεπιλογή το διάστημα 00:00 – 23:59 και τιμή 14 g/U ενώ η ρύθμιση Basal Rate έχει ως προεπιλογή το διάστημα 00:00 – 23:59 και τιμή 0.5. Τέλος η ρύθμιση των DIA και DCA επηρεάζει επίσης τον υπολογισμό των IOB και COB του υπολογισμού που περιγράφεται στην υποενότητα 3.1.4 και λαμβάνει χώρα στη Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης. Οι Διάρκειες Δράσης Ινσουλίνης και Υδατανθράκων ορίζονται σε ώρες μέσω του αντίστοιχου πεδίου εισαγωγής αριθμών που βρίσκεται κάτω από την επικεφαλίδα της κάθε ρύθμισης. Η ρύθμιση DIA δεν μπορεί να υπερβαίνει τις 10 ώρες και η τιμή της DCA δεν μπορεί να υπερβαίνει τις 7 ώρες, αμφότερες τιμές θα πρέπει να είναι θετικοί αριθμοί. Η DIA έχει τιμή προεπιλογής τις 3,5 ώρες και η DCA τη 1 ώρα.



Εικόνα 3.5 Στιγμιότυπα οθόνης κατά την προβολή της Οθόνης Ρυθμίσεων.

3.2.10 Οθόνη προβολής σημειώσεων

Η Οθόνη προβολής σημειώσεων, παρέχει στο χρήστη τη δυνατότητα να δει όλες τις σημειώσεις που έχει δημιουργήσει εντός της εφαρμογής. Εμφανίζεται αποκλειστικά και μόνο αν το επιλέξει ο χρήστης μέσω της Αρχικής Οθόνης (υποενότητα 3.2.4). Αποτελείται από τρία στοιχεία διεπαφής χρήστη και δεν ανήκει στις βασικές οθόνες της εφαρμογής.

Στο Επάνω μέρος της οθόνης, υπάρχει ένα στοιχείο προβολής κειμένου (TextView) το οποίο αναγράφει “Notes” και αποτελεί τίτλο της οθόνης. Κάτω από αυτό, υπάρχει ένα στοιχείο προβολής λίστας το οποίο περιέχει όλες τις σημειώσεις που έχουν αποθηκευτεί στην τοπική βάση δεδομένων ταξινομημένες χρονικά από την πιο πρόσφατη στην παλαιότερη. Οι σημειώσεις στη λίστα δεν έχουν τίτλο και διακρίνονται από την ημερομηνία και ώρα δημιουργίας τους. Κάτω από την λίστα των σημειώσεων υπάρχει ένα ακόμη πεδίο προβολής κειμένου σημείωσης το οποίο αρχικά είναι κενό. Όταν επιλέγεται κάποια σημείωση από τη λίστα τότε το πεδίο προβολής κειμένου σημείωσης προβάλλει το κείμενο της αντίστοιχης σημείωσης.

3.3 Λειτουργία εφαρμογής στο παρασκήνιο

Εκτός από το γραφικό περιβάλλον της εφαρμογής, υπάρχουν αρκετές βασικές λειτουργικότητες αυτής, οι οποίες εκτελούνται στο παρασκήνιο χωρίς ο χρήστης να έχει άμεση αλληλεπίδραση με εκείνες. Επιπλέον κάποιες από αυτές, προϋποθέτουν τη συνεχή και απερίσπαστη λειτουργία της GMA στη συσκευή του χρήστη ώστε να εκτελεστούν σωστά. Οι βασικές λειτουργικότητες τέτοιου είδους αφορούν: Παραγωγή και λήψη νέων τιμών BG, Ενεργοποίηση συναγερμών, Δέκτες Εκπομπών (BroadCast Receivers), Αποθήκευση και

ανάκτηση δεδομένων από την τοπική βάση και Δοσοληψίες πακέτων με τον Παγκόσμιο Ιστό (Internet). Σε αυτή την ενότητα θα παρουσιαστεί, ο τρόπος με τον οποίο εκτελούνται αυτές οι λειτουργικότητες και πως επηρεάζουν το γραφικό περιβάλλον της εφαρμογής.

3.3.1 Υπηρεσίες Παρασκηνίου (Services)

Στις εφαρμογές Android, ένα βασικό στοιχείο που επιτρέπει την εκτέλεση εργασιών στο παρασκήνιο χωρίς την ανάγκη για διεπαφή χρήστη είναι τα Services (Υπηρεσίες). Χρησιμοποιούνται για εργασίες που πρέπει να συνεχίζουν να εκτελούνται ακόμα και όταν η εφαρμογή δεν είναι ενεργή ή όταν ο χρήστης αλλάξει σε άλλη εφαρμογή. Η GMA διαθέτει δύο τέτοια Services που καλύπτουν τις περισσότερες από τις λειτουργικότητες παρασκηνίου που περιλαμβάνει. Τα Services αυτά είναι, το Service παρακολούθησης γλυκόζης (GlucoseMonitorService), που είναι και το κύριο Service της εφαρμογής και αποτελεί μέρος διάφορων λειτουργιών και το Service εκτέλεσης συναγερμών (AlertService) το οποίο είναι μονοδιάστατο και χρησιμοποιείται μόνο στην λειτουργία συναγερμών. Η τεχνική υλοποίηση των Services αναλύεται στο επόμενο κεφάλαιο (υποενότητα 4.3.11).

Το GlucoseMonitorService, είναι Service της μορφής ForegroundService. Τα συγκεκριμένα Services απαιτείται να προβάλλουν καθ' όλη τη διάρκεια εκτέλεσής τους κάποια μόνιμη ειδοποίηση (Pertsistent Notification), ώστε ο χρήστης να είναι ενήμερος ότι βρίσκονται σε λειτουργία, αλλά και τι είδους λειτουργία εξυπηρετούν. Η μόνιμη ειδοποίηση παραμένει εμφανής ανεξαρτήτως του αν η εφαρμογή βρίσκεται στο προσκήνιο, παρασκήνιο ή είναι κλειστή και εξαφανίζεται μόνο αν σταματήσει να λειτουργεί το ForegroundService το οποίο και τη δημιούργησε. Αυτό καθιστά τα ForegroundServices να έχουν υψηλότερη προτεραιότητα σε επίπεδο λειτουργικού συστήματος και είναι λιγότερο πιθανό να τερματιστούν από αυτό. Τα ForegroundServices μπορούν να παραμένουν σε λειτουργία ακόμη και αν η συσκευή του χρήστη είναι σε κατάσταση αδράνειας. Το GlucoseMonitorService αν δεν εκτελείται ήδη, εκκινείτε πάντοτε μαζί με την εφαρμογή και η εκκίνησή του είναι μια από τις πρώτες ενέργειες που πραγματοποιεί η εφαρμογή κάθε φορά που η ίδια εκκινείται. Εφόσον κατά την εκκίνηση της εφαρμογής το GlucoseMonitorService βρίσκεται σε λειτουργία τότε δεν επανεκκινείται. Μια από τις πρώτες ενέργειες που εκτελεί το GlucoseMonitorService όταν εκκινείται, είναι η δημιουργία μόνιμης ειδοποίησης (Persistent Notification) με την οποία ενημερώνει το χρήστη ότι οι παρασκηνιακές υπηρεσίες της εφαρμογής βρίσκονται σε λειτουργία και πληροφορεί το χρήστη για την τελευταία τιμή γλυκόζης που αποθηκεύτηκε από την εφαρμογή, τη χρονική στιγμή λήψης αυτής και τις μονάδες μέτρησης βάσει των οποίων προβάλλεται η τιμή BG. Η μόνιμη ειδοποίηση δεν μπορεί να κρυφτεί από το χρήστη (συνήθως σύροντας εκτός οθόνης στο αντίστοιχο πεδίο ειδοποιήσεων των συσκευών Android) και εάν ο χρήστης πατήσει πάνω της τότε η εφαρμογή ανοίγει αν ήταν προηγουμένως κλειστή ή έρχεται στο προσκήνιο εάν προηγουμένως ήταν στο παρασκήνιο. Οι λειτουργικότητες τις οποίες το GlucoseMonitorService εξυπηρετεί είναι: η Υπηρεσία παραγωγής και λήψης γλυκόζης, η ενημέρωση του χρήστη για την τρέχουσα τιμή γλυκόζης αίματος, η λειτουργία συναγερμών και υπηρεσίες επανέναρξης των παρασκηνιακών λειτουργιών της εφαρμογής.

Όπως έχει αναφερθεί και σε προηγούμενες ενότητες η εφαρμογή δύναται να λαμβάνει τιμές γλυκόζης από την εφαρμογή xDrip^{+} ή να παράγει εικονικές τιμές γλυκόζης τοπικά ανάλογα με το την επιλογή του χρήστη στην Οθόνη Ρυθμίσεων (υποενότητα 3.2.9). Το

GlucoseMonitorService παρεμβάλλεται και στις δύο διαδικασίες. Στην περίπτωση που ο χρήστης επιλέξει η GMA να λαμβάνει τιμές BG μέσω της εφαρμογής xDrip+, το GlucoseMonitorService καταχωρεί τον αντίστοιχο Δέκτη Εκπομπών (υποενότητα 3.3.2), μέσω του οποίου η GMA δέχεται τιμές. Εάν ο χρήστης επιλέξει να παράγονται από την GMA εικονικές τιμές, τότε οι τιμές BG παράγονται μέσω του GlucoseMonitorService, το οποίο παράγει και αποθηκεύει στην τοπική βάση δεδομένων μια νέα εικονική τιμή BG κάθε ένα έως πέντε λεπτά, με περιορισμό η νέα τιμή γλυκόζης της νέας BG να μην απέχει πάνω από 20 mg/dL κατ' απόλυτη τιμή από την προηγούμενη, να μην είναι κάτω των 40 mg/dL και να μην υπερβαίνει την τιμή 400 mg/dL.

Κάθε φορά που αποθηκεύεται κάποια νέα τιμή BG στην τοπική βάση δεδομένων, ανεξαρτήτως πηγής, το GlucoseMonitorService ενημερώνεται παρασκηνιακά για τη νέα τιμή και αρχικά ενημερώνει τη μόνιμη ειδοποίηση ώστε η τελευταία να προβάλλει τα δεδομένα της νέας τιμής, στη συνέχεια το GlucoseMonitorService πραγματοποιεί έλεγχο για το αν η νέα τιμή χρειάζεται να προκαλέσει την ενεργοποίηση κάποιου συναγερμού. Η διαδικασία αυτή περιλαμβάνει και τον έλεγχο για το αν οι συναγερμοί είναι ενεργοί και αν είναι σε σίγαση (υποενότητα 3.2.9). Εφόσον κριθεί ότι χρειάζεται να ενεργοποιηθεί κάποιος συναγερμός, το GlucoseMonitorService πραγματοποιεί ένα Broadcast (μήνυμα Intent), το οποίο μέσω του αντίστοιχου δέκτη εκπομπής εκκινεί το AlertService. Το AlertService, είναι ένα τυπικό (Standard) Service του Android μέσω του οποίου εκτελείται η ενεργοποίηση συναγερμών. Όταν εκκινείται, δημιουργεί μια ειδοποίηση επικάλυψης (overlay notification) που εμφανίζεται πάνω από τις άλλες εφαρμογές, ενεργοποιεί τη λειτουργία δόνησης και αναπαράγει ηχητική ειδοποίηση. Παράλληλα, αποθηκεύει το συμβάν της ενεργοποίησης του συναγερμού στην τοπική βάση δεδομένων. Η ειδοποίηση επικάλυψης, εμφανίζεται πάντοτε πάνω από άλλες εφαρμογές και στην περίπτωση που η συσκευή βρίσκεται σε κατάσταση αδράνειας και η οθόνη είναι κλειστή, ξυπνά την οθόνη ώστε η ειδοποίηση να είναι εμφανής. Περιέχει 4 στοιχεία διεπαφής χρήστη τα οποία είναι: ένα εικονίδιο προειδοποίησης (τρίγωνο κόκκινου χρώματος με θαυμαστικό στο κέντρο αυτού), ένα TextView με την ένδειξη "Alert", ένα TextView που αναγράφει την τιμή της BG που προκάλεσε την ενεργοποίηση του συναγερμού και ένα TextView που αναγράφει την χρονική στιγμή λήψης της μέτρησης. Εάν ο χρήστης πατήσει σε οποιοδήποτε σημείο του γραφικού περιβάλλοντος της ειδοποίησης, η ειδοποίηση εξαφανίζεται, ο ήχος και η δόνηση σταματούν και η GMA ανοίγει εάν ήταν κλειστή ή έρχεται στο προσκήνιο εάν ήταν προηγουμένως στο παρασκήνιο. Με το κλείσιμο της ειδοποίησης σταματά και η εκτέλεση του AlertService.

Προκειμένου οι παρασκηνιακές υπηρεσίες της εφαρμογής να λειτουργούν απερίσπαστα, είναι αναγκαίο το GlucoseMonitorService να μην αφαιρείται από τη λίστα ενεργών παρασκηνιακών υπηρεσιών του λειτουργικού συστήματος. Αυτό προαπαιτεί από τα Services να εκκινούνται με συγκεκριμένο τρόπο και παραμέτρους, να ελαχιστοποιούν την ενέργεια (ποσοστό χρήσης της μπαταρίας της συσκευής) που καταναλώνουν και να λαμβάνουν άδεια από το λειτουργικό σύστημα κάθε φορά που πρόκειται να εκτελέσουν λειτουργίες την ώρα που η συσκευή βρίσκεται σε κατάσταση αναμονής. Για το λόγο αυτό, το GlucoseMonitorService είναι στρατηγικά σχεδιασμένο, ώστε να αδρανοποιείται (παραμένοντας ενεργό) όσο δεν απαιτείται να πραγματοποιεί ενέργειες και να επαναλειτουργεί πλήρως όταν χρειάζεται.

3.3.2 Δέκτες Εκπομπών

Οι Δέκτες Εκπομπών (BroadcastReceivers), είναι μία από τις βασικές συνιστώσες (components) του Android και χρησιμοποιούνται για τη λήψη και απόκριση σε broadcast μηνύματα (intents) που αποστέλλονται από το λειτουργικό σύστημα, άλλες εφαρμογές και εντός της ίδιας εφαρμογής. Ο κάθε BroadcastReceiver λαμβάνει και διαχειρίζεται συγκεκριμένα intents και μπορεί να σχετίζεται με προσκηνιακά και παρασκηνιακά τμήματα της εφαρμογής. Εντός της εφαρμογής υπάρχουν BroadcastReceivers οι οποίοι σχετίζονται με το λειτουργικό σύστημα, την εφαρμογή xDrip⁺ αλλά και λειτουργίες εντός εφαρμογής.

Το GlucoseMonitorService (υποενότητα 3.3.1) της εφαρμογής δύναται να εκκινείται αυτόματα κάθε φορά που η συσκευή του χρήστη ανοίγει (power on), εφόσον ο χρήστης και η έκδοση Android της συσκευής το επιτρέπουν. Για το σκοπό αυτό, έχει οριστεί εντός της GMA ο AutoStartReceiver, ο οποίος έχει σχεδιαστεί να διαχειρίζεται το Intent “android.intent.action.BOOT_COMPLETED” το οποίο αποστέλλεται από το λειτουργικό σύστημα όταν η συσκευή ανοίγει ενώ προηγουμένως ήταν κλειστή και το "android.intent.action.QUICKBOOT_POWERON" το οποίο αποστέλλεται από το λειτουργικό σύστημα όταν η συσκευή επανεκκινείται. Όταν ο AutoStartReceiver λάβει το intent επιχειρεί να εκκινήσει το GlucoseMonitorService.

Η λήψη τιμών BG από την εφαρμογή xDrip⁺ πραγματοποιείται μέσω Broadcast Intents. Στην εφαρμογή xDrip⁺, αφού δηλωθεί το όνομα πακέτου εφαρμογής της GMA ("com.cchatzidopavlakis.diplomatiki"), εκκινείται μια διαδικασία η οποία κάθε φορά που η εφαρμογή xDrip⁺ λαμβάνει κάποια νέα τιμή, πραγματοποιεί αποστολή συγκεκριμένου Intent το οποίο εμπεριέχει την νέα τιμή. Στην εφαρμογή GMA υπάρχει ο BGDataReceiver, ο οποίος λαμβάνει την τιμή. Αφού λάβει την τιμή και εφόσον ο χρήστης έχει επιλέξει να δέχεται η GMA τιμές με αυτόν τον τρόπο, την αποθηκεύει στην τοπική βάση δεδομένων.

Το GlucoseMonitorService είναι σχεδιασμένο να παραμένει αδρανές και να ενεργοποιείται ανά τρία έως πέντε λεπτά, παράλληλα εφόσον ο χρήστης έχει επιλέξει να παράγονται τοπικά εικονικές τιμές γλυκόζης η διαδικασία αυτή περιλαμβάνει και την παραγωγή και αποθήκευση νέων εικονικών τιμών BG. Αρχικά το GlucoseMonitorService ενημερώνει το σύστημα πως θα αφυπνιστεί σε τρία λεπτά, αυτό επιτυγχάνεται θέτοντας ένα PendingIntent το οποίο ενεργοποιείται μελλοντικά και συγκεκριμένα σε τρία λεπτά. Τα PendingIntents είναι μια ειδική κατηγορία του αντικειμένου Intent στο Android που επιτρέπει σε εφαρμογές ή το ίδιο το Android σύστημα να εκτελέσουν μια ενέργεια εκ μέρους της εφαρμογής που τα δημιουργεί, σε ένα μελλοντικό χρόνο. Όταν το PendingIntent ενεργοποιηθεί, η εφαρμογή το λαμβάνει και το διαχειρίζεται μέσω του MockValueReceiver. Όταν ο MockValueReceiver λάβει το PendingIntent αρχικά εκκινεί το GlucoseMonitorService εφόσον χρειάζεται και στη συνέχεια αποστέλλει ένα δεύτερο Intent με τύπο “ACTION_GENERATE_BG”. Το τελευταίο Intent λαμβάνεται από τοπικό BroadcastReceiver εντός του GlucoseMonitorService ο οποίος όταν το λάβει παράγει και αποθηκεύει στην τοπική βάση μια νέα εικονική τιμή γλυκόζης εφόσον ο χρήστης το επιλέξει στις Ρυθμίσεις και προγραμματίζει ένα νέο PendingIntent.

Ο τελευταίος BroadcastReceiver αφορά τους συναγερμούς. Στην προηγούμενη υποενότητα έχει αναφερθεί ότι το GlucoseMonitorService ενημερώνεται κάθε φορά που μια νέα τιμή BG αποθηκεύεται στη βάση δεδομένων και με τη σειρά του εκκινεί το AlertService το

οποίο και θέτει το συναγερμό σε λειτουργία. Αυτό επιτυγχάνεται μέσω του AlarmReceiver. Το GlucoseMonitorService, όταν χρειάζεται να ενεργοποιηθεί κάποιος συναγερμός, θέτει ένα νέο Intent το οποίο στη συνέχεια λαμβάνεται από τον AlarmReceiver και εκείνος με τη σειρά του εκκινεί το AlarmService.

3.3.3 Υπηρεσίες μεταφοράς δεδομένων

Οι υπηρεσίες Εγγραφής νέου χρήστη, Εισόδου χρήστη και Επαναφοράς κωδικού προϋποθέτουν την επικοινωνία της εφαρμογής με το Server μέσω διαδικτύου. Πιο συγκεκριμένα, αφού οι χρήστες συμπληρώσουν τις πληροφορίες τους στα αντίστοιχα πεδία της Οθόνης Εγγραφής (υποενότητα 3.2.1) ή Εισόδου (υποενότητα 3.2.2), η εφαρμογή αποστέλλει αυτές τις πληροφορίες στο Server προκειμένου να ολοκληρωθούν οι ανωτέρω λειτουργίες. Ο Server, έχει δημιουργηθεί με τη χρήση του εργαλείου Firebase (ενότητα 2.4) και για την πραγματοποίηση αυτών των διαδικασιών χρησιμοποιείται μια εξειδικευμένη κλάση της βιβλιοθήκης Firebase με όνομα FirebaseAuth. Οι μέθοδοι που αναλαμβάνουν τη μεταφορά δεδομένων λειτουργούν ασύγχρονα σε άλλο νήμα εκτέλεσης, επιτρέποντας στη διεπαφή χρήστη να συνεχίσει να εκτελείται όσο αναμένουν την απάντηση του server.

Εντός της εφαρμογής, όπως έχει αναφερθεί και σε προηγούμενες ενότητες δημιουργείται μια τοπική βάση δεδομένων στην οποία αποθηκεύονται διάφορες τιμές και δεδομένα της εφαρμογής. Τα δεδομένα αυτά μπορούν αφορούν και να προκύπτουν από το γραφικό περιβάλλον και τα Services της ίδιας της εφαρμογής ή εξωγενείς πηγές. Η αποθήκευση και ανάκτηση των δεδομένων αυτών όπως και οι δοσοληψίες δεδομένων μέσω διαδικτύου που περιγράφηκαν στην προηγούμενη παράγραφο, είναι πιθανό να διαρκέσουν αρκετά milliseconds ή και δευτερόλεπτα. Αυτό μπορεί να προκαλέσει διάφορα προβλήματα όπως, ο τερματισμός της εφαρμογής από το λειτουργικό καθώς το Android δύναται να θεωρήσει πως η εφαρμογή είναι μη αποκρίσιμη (ANR - Application Not Responding), η αρνητική εμπειρία του χρήστη καθώς η εφαρμογή θα έχει την εικόνα πως έχει “κολλήσει” καθώς επίσης και κακή χρήση των πόρων της συσκευής. Η ασύγχρονη φύση των μεθόδων της κλάσης FirebaseAuth αντιμετωπίζει το συγκεκριμένο θέμα όσον αφορά τις υπηρεσίες επικοινωνίας με το διαδίκτυο. Όσον αφορά τα δεδομένα της τοπικής βάσης, τα ερωτήματα SQL (SQL queries) που πραγματοποιούν αποθήκευση ή ανάκτηση αυτών, εκτελούνται με τη χρήση Java ExecutorService. Το ExecutorService είναι ένα framework της Java που χρησιμοποιείται για την εκτέλεση εργασιών σε νήματα (threads) ασύγχρονα και τερματίζει μετά την εκτέλεση αυτών. Η τεχνική υλοποίηση της μεταφοράς δεδομένων αναλύεται στο επόμενο κεφάλαιο.

3.4 Σενάρια Χρήσης

Σε αυτή την ενότητα θα παρουσιαστούν τρία σενάρια χρήσης διαφορετικών λειτουργιών της εφαρμογής. Για κάθε σενάριο χρήσης θα παρουσιαστεί: η περιγραφή του σεναρίου, οι προϋποθέσεις για την πραγματοποίησή του, οι συμμετέχοντες και ο ρόλος τους, η βασική ροή ενεργειών, οι εναλλακτικές ροές και το αναμενόμενο αποτέλεσμα.

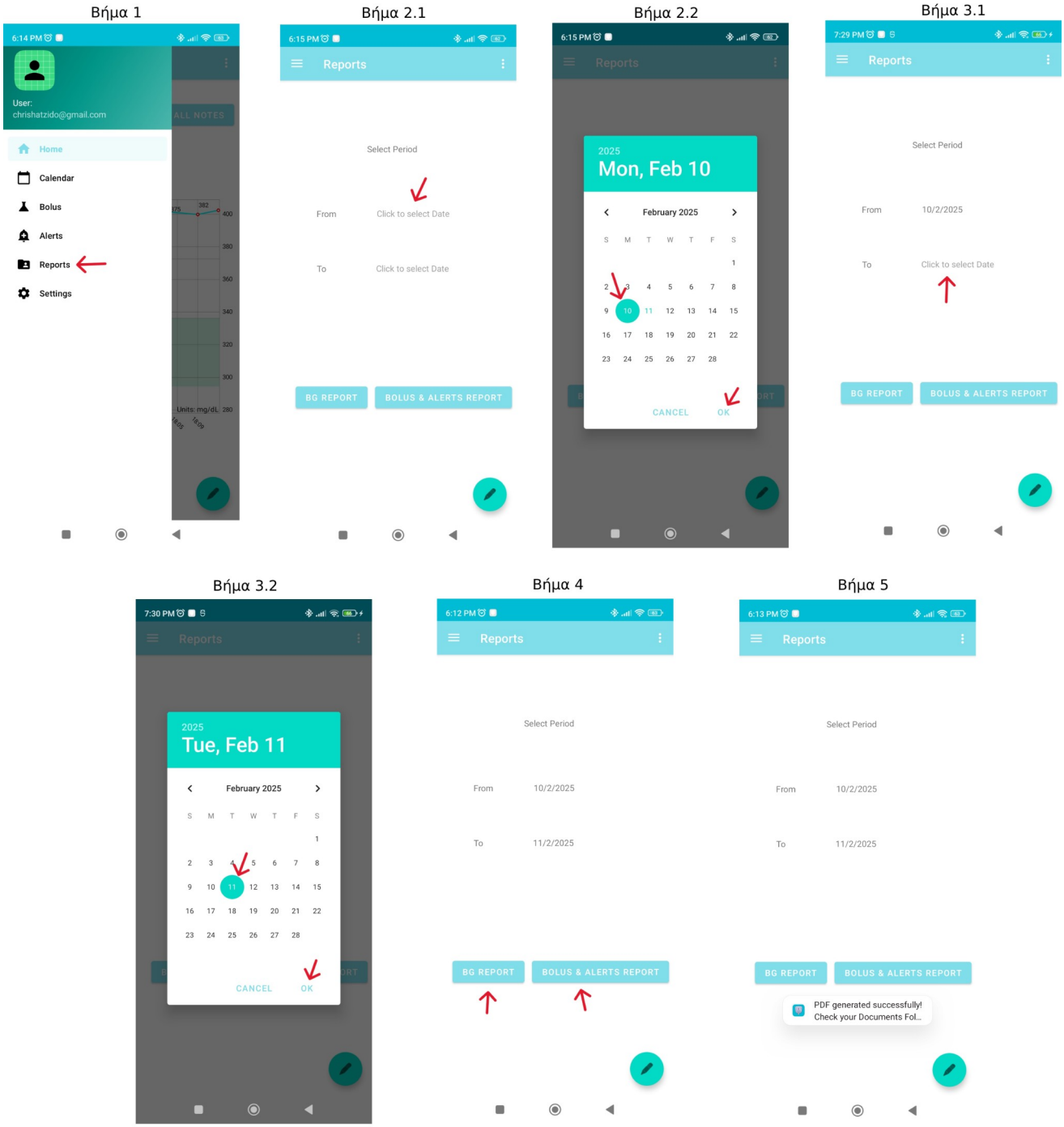
3.4.1 Σενάριο Δημιουργίας Αναφορών

Το σενάριο έχει ως στόχο τη δημιουργία και αποθήκευση αναφορών γλυκόζης αίματος (BG) και συμβάντων (Bolus & Alerts) από τον χρήστη, για χρονικό διάστημα που ο ίδιος επιλέγει, μέσω της Οθόνης Δημιουργίας Αναφορών της εφαρμογής. Η επίτευξη του στόχου του σεναρίου προϋποθέτει ο χρήστης να έχει παραχωρήσει τα απαραίτητα δικαιώματα στην GMA για την αποθήκευση αρχείων στον χώρο αποθήκευσης της συσκευής. Επιπλέον προϋπόθεση αποτελεί, η ύπαρξη αποθηκευμένων τιμών BG και δεδομένων συμβάντων στην τοπική βάση της εφαρμογής για το χρονικό διάστημα που επιλέγει ο χρήστης. Συμμετέχοντες στο σενάριο είναι ο Χρήστης και η εφαρμογή. Ο χρήστης, επιλέγει τη χρονική περίοδο και τους τύπους αναφορών που επιθυμεί να δημιουργήσει. Η εφαρμογή, παράγει και αποθηκεύει το αρχείο αναφοράς τύπου PDF στον φάκελο "Έγγραφα" της συσκευής. Αναμενόμενο αποτέλεσμα είναι η δημιουργία και αποθήκευση των αρχείων αναφοράς, πράγμα που επιτρέπει στο χρήστη να τα ανοίξει, να τα μοιραστεί ή να τα εκτυπώσει.

Η βασική ροή ενεργειών του σεναρίου αποτελείται από τα παρακάτω βήματα:

1 Ο χρήστης μεταβαίνει στην οθόνη Δημιουργίας Αναφορών μέσω του drawer menu της εφαρμογής.
2 Ο χρήστης επιλέγει την ημερομηνία έναρξης της χρονικής περιόδου της αναφοράς:
  2.1 Πατά πάνω στο TextView δίπλα στην αντίστοιχη ένδειξη (From).
  2.2 Επιλέγει την επιθυμητή ημέρα και στη συνέχεια πατάει OK.
3 Ο χρήστης επιλέγει την ημερομηνία λήξης της χρονικής περιόδου της αναφοράς.
  3.1 Πατά πάνω στο TextView δίπλα στην αντίστοιχη ένδειξη (To).
  3.2 Επιλέγει την επιθυμητή ημέρα και στη συνέχεια πατάει OK.
4 Επιλέγει τον τύπο αναφοράς πατώντας το αντίστοιχο κουμπί:
  4.1 BG Report για αναφορά τιμών γλυκόζης.
  4.2 Bolus & Alerts Report για αναφορά συμβάντων.
5 Η εφαρμογή δημιουργεί το αρχείο PDF και το αποθηκεύει στο φάκελο "Έγγραφα" της συσκευής.

Ακολουθεί η Εικόνα 3.6, στην εικόνα αυτή υπάρχουν στιγμιότυπα οθόνης για κάθε βήμα της βασικής ροής κατά την εκτέλεση του σεναρίου σε φυσική συσκευή.



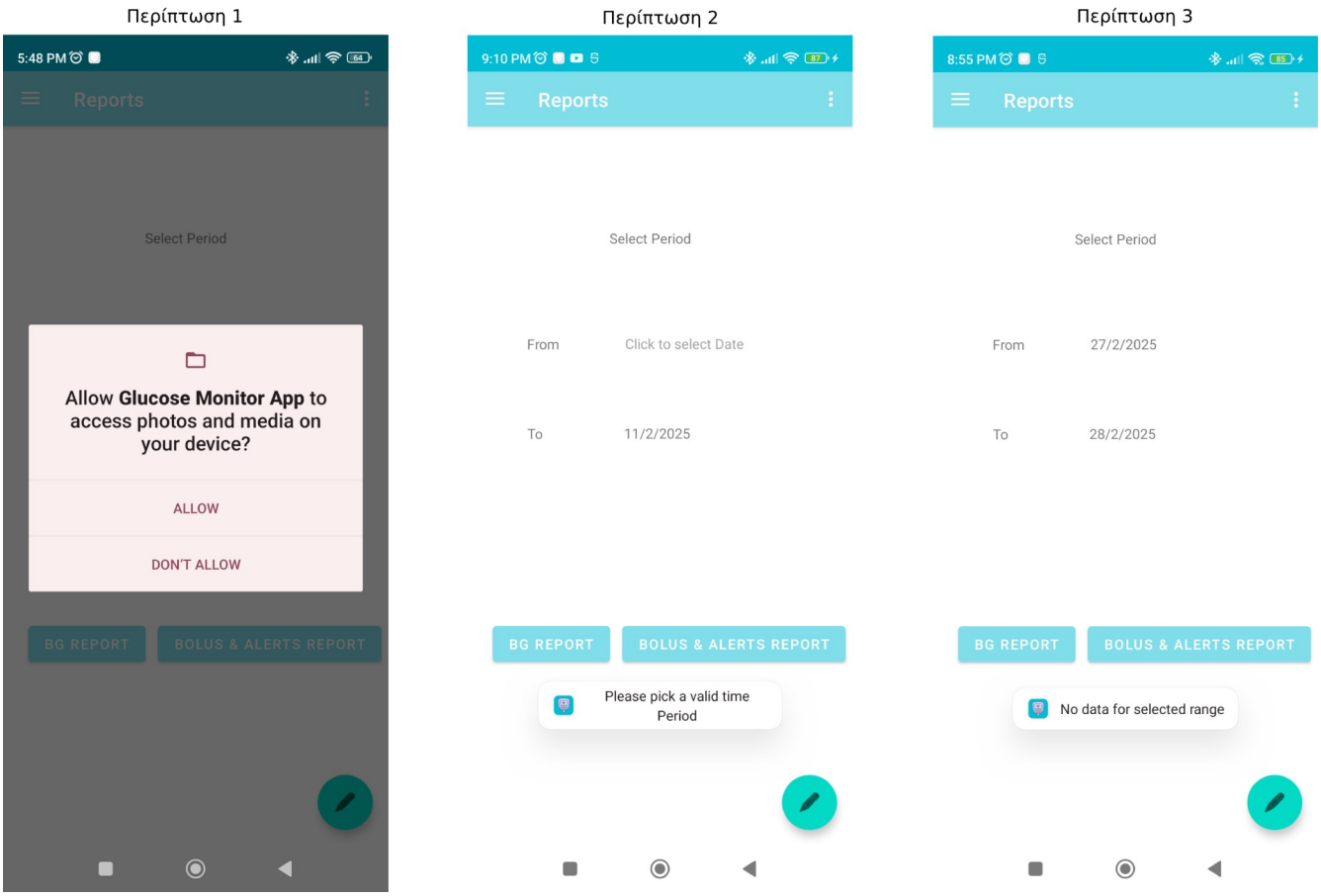
Εικόνα 3.6 Σενάριο Δημιουργίας Αναφορών - Βασική Ροή

Οι εναλλακτικές ροές του σεναρίου περιλαμβάνουν τις περιπτώσεις που:

1. Ο χρήστης προσπαθεί να δημιουργήσει αναφορά αλλά δεν έχει παραχωρήσει δικαιώματα αποθήκευσης. Στην περίπτωση αυτή, η εφαρμογή ζητά από το λειτουργικό σύστημα να εμφανίσει ένα αναδυόμενο παράθυρο μέσω του οποίου ο χρήστης μπορεί να δώσει την απαιτούμενη άδεια.
2. Ο χρήστης δεν επιλέγει σωστά χρονική περίοδο. Στην περίπτωση αυτή, η εφαρμογή εμφανίζει μήνυμα σφάλματος και δεν προχωρά στη δημιουργία της αναφοράς.

3. Δεν υπάρχει καμία αποθηκευμένη τιμή BG ή δεδομένων συμβάντων για το επιλεγμένο διάστημα. Στην περίπτωση αυτή, η εφαρμογή εμφανίζει μήνυμα σφάλματος και δεν προχωρά στη δημιουργία της αναφοράς.

Ακολουθεί η Εικόνα 3.7, στην οποία υπάρχουν στιγμιότυπα οθόνης για κάθε περίπτωση εναλλακτικής ροής κατά την εκτέλεση του σεναρίου σε φυσική συσκευή.



Εικόνα 3.7 Σενάριο Δημιουργίας Αναφορών - Εναλλακτικές Ροές

3.4.2 Σενάριο Επεξεργασίας Συναγερμού

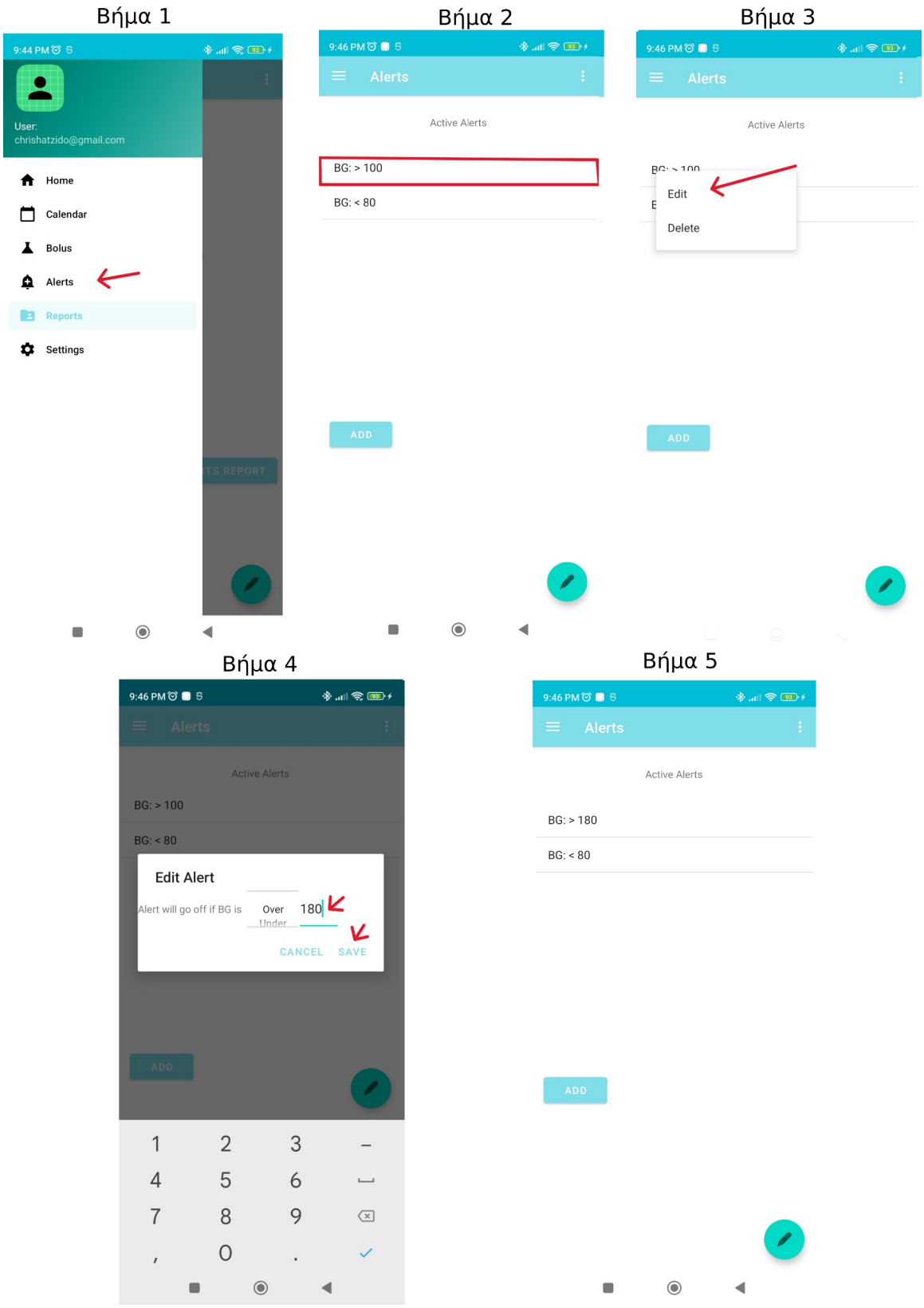
Το σενάριο έχει ως στόχο την τροποποίηση της τιμής γλυκόζης ορίου κάποιου συναγερμού από το χρήστη, μέσω τη Οθόνης Διαχείρισης Συναγερμών. Το σενάριο έχει την προϋπόθεση ότι ο χρήστης έχει ήδη αποθηκεύσει τουλάχιστον έναν συναγερμό στην εφαρμογή. Συμμετέχοντες στο σενάριο είναι ο Χρήστης ο οποίος επιθυμεί να επεξεργαστεί ένα συναγερμό και η GMA. Αναμενόμενο αποτέλεσμα είναι η αποθήκευση της τροποποίησης που πραγματοποιεί ο χρήστης και η προβολή της στη λίστα ενεργών συναγερμών της Οθόνης Διαχείρισης Συναγερμών.

Η βασική ροή ενεργειών του σεναρίου αποτελείται από τα παρακάτω βήματα:

1. Ο χρήστης μεταβαίνει στην Οθόνη Διαχείρισης Συναγερμών μέσω του drawer menu.

2. Ο χρήστης εντοπίζει τον συναγερμό που επιθυμεί να επεξεργαστεί και πατά παρατεταμένα πάνω στην αντίστοιχη εγγραφή της λίστας.
3. Εμφανίζεται ένα αναδυόμενο μενού επιλογών, στο οποίο ο χρήστης επιλέγει την ενέργεια "Επεξεργασία" (Edit).
4. Εμφανίζεται ένα αναδυόμενο παράθυρο επεξεργασίας συναγερμού. Ο χρήστης αλλάζει τις επιθυμητές ρυθμίσεις του συναγερμού και πατά το κουμπί "Αποθήκευση".
5. Η εφαρμογή αποθηκεύει τις νέες ρυθμίσεις στη βάση δεδομένων, το αναδυόμενο παράθυρο κλείνει και η λίστα ενεργών συναγερμών ανανεώνεται.

Εφόσον ολοκληρωθεί το Σενάριο Επεξεργασίας Συναγερμού, τα νέα δεδομένα αποθηκεύονται στην τοπική βάση δεδομένων της εφαρμογής καθώς πραγματοποιείται ενημέρωση (update) των δεδομένων του ήδη αποθηκευμένου συναγερμού. Ακολουθεί η Εικόνα 3.8, στην εικόνα αυτή υπάρχουν στιγμιότυπα οθόνης για κάθε βήμα της βασικής ροής κατά την εκτέλεση του σεναρίου σε φυσική συσκευή.



Εικόνα 3.8 Σενάριο Επεξεργασίας Συναγερμού – Βασικές Ροές

3.4.3 Σενάριο Υπολογισμού Bolus

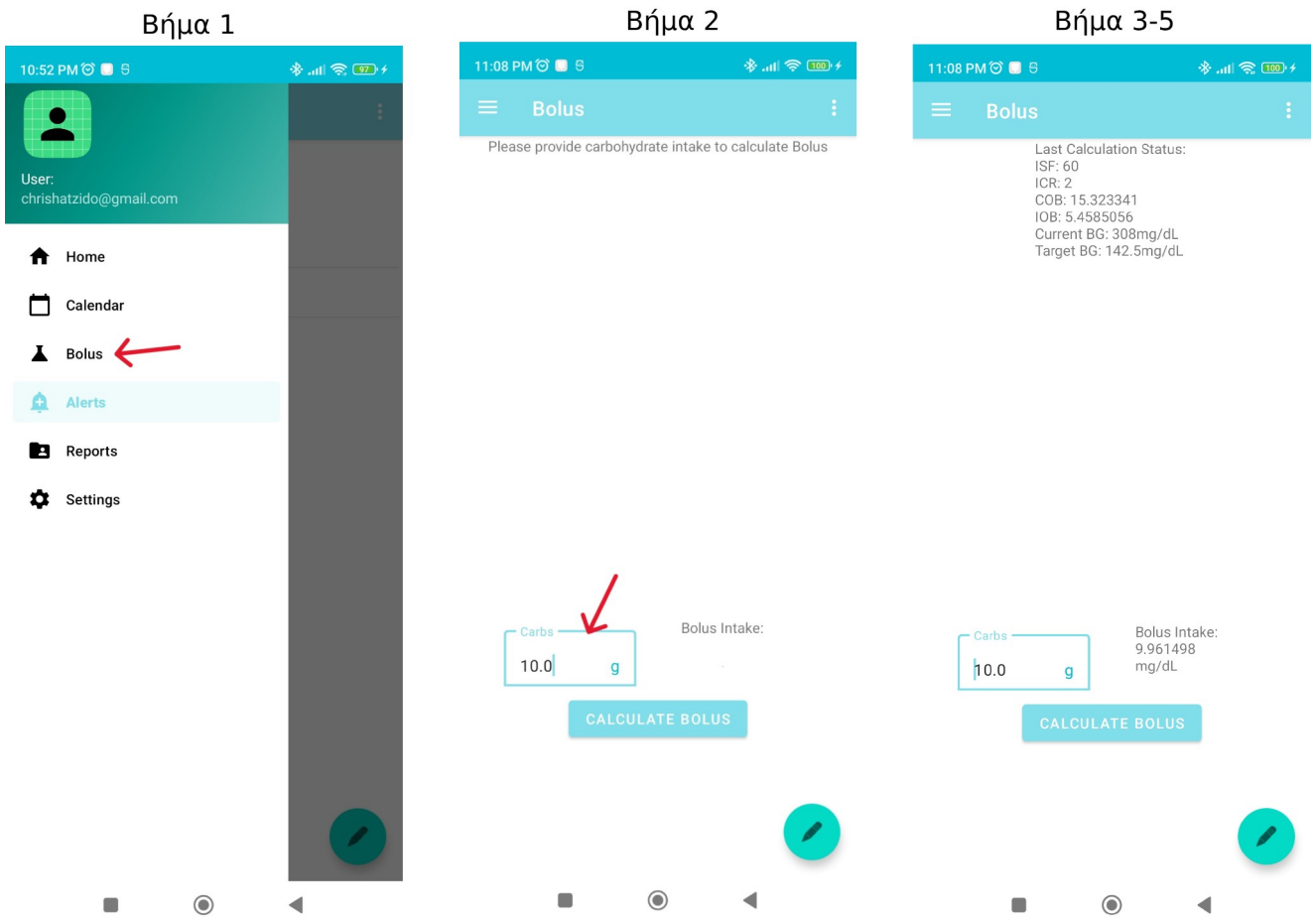
Το σενάριο αυτό έχει ως στόχο τον υπολογισμό της απαιτούμενης ποσότητας έκτακτης δόσης ινσουλίνης (Bolus) που πρέπει να εγχύσει ο χρήστης για μεταβολισμό γεύματος (10

γραμμάρια υδατανθράκων). Ο χρήστης εισάγει την ποσότητα υδατανθράκων που πρόκειται να καταναλώσει, και η εφαρμογή, αφού ανακτήσει τις απαραίτητες τιμές από τη βάση δεδομένων και τις ρυθμίσεις, υπολογίζει και εμφανίζει τη συνιστάμενη ποσότητα ινσουλίνης. Το σενάριο έχει την προϋπόθεση ότι ο χρήστης έχει ήδη καταχωρήσει τις απαραίτητες παραμέτρους στις ρυθμίσεις της εφαρμογής (π.χ. αναλογίες ινσουλίνης προς υδατάνθρακες, συντελεστής ευαισθησίας στην ινσουλίνη κ.λπ.). Αν πρόκειται για γευματική ινσουλίνη, το σενάριο προϋποθέτει πως ο χρήστης γνωρίζει την ποσότητα των υδατανθράκων που πρόκειται να καταναλώσει. Συμμετέχοντες στο σενάριο είναι ο χρήστης της εφαρμογής που επιθυμεί να υπολογίσει τη σωστή ποσότητα Bolus ινσουλίνης και η εφαρμογή που εκτελεί τους απαραίτητους υπολογισμούς και καταγράφει τα δεδομένα. Αναμενόμενο αποτέλεσμα για το σενάριο είναι η προβολή της ποσότητας ινσουλίνης που χρειάζεται να εγχυθεί. Ως μετασυνθήκες του σεναρίου έχουμε την καταγραφή και αποθήκευση του συμβάντος στην τοπική βάση δεδομένων. Καταγράφονται και αποθηκεύονται επίσης η ποσότητα ινσουλίνης και υδατανθράκων που εισήχθησαν στον οργανισμό του χρήστη προκειμένου να χρησιμοποιηθούν σε μελλοντικούς υπολογισμούς εφόσον χρειαστεί.

Η βασική ροή ενεργειών του σεναρίου αποτελείται από τα παρακάτω βήματα:

1. Ο χρήστης μεταβαίνει στην Οθόνη Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης μέσω του drawer menu.
2. Ο χρήστης εισάγει την ποσότητα υδατανθράκων που πρόκειται να καταναλώσει και πατάει το κουμπί "Calculate Bolus".
3. Η εφαρμογή ανακτά τις αποθηκευμένες παραμέτρους από τη βάση δεδομένων και το αρχείο Shared Preferences.
4. Η εφαρμογή υπολογίζει την απαιτούμενη ποσότητα ινσουλίνης με βάση τις καταχωρημένες τιμές του χρήστη.
5. Το αποτέλεσμα προβάλλεται στην οθόνη στο πεδίο "Bolus Intake", μαζί με όλες τις παραμέτρους που χρησιμοποιήθηκαν στον υπολογισμό.

Ακολουθεί η Εικόνα 3.9, στην εικόνα αυτή υπάρχουν στιγμιότυπα οθόνης για κάθε βήμα της βασικής ροής κατά την εκτέλεση του σεναρίου σε φυσική συσκευή.

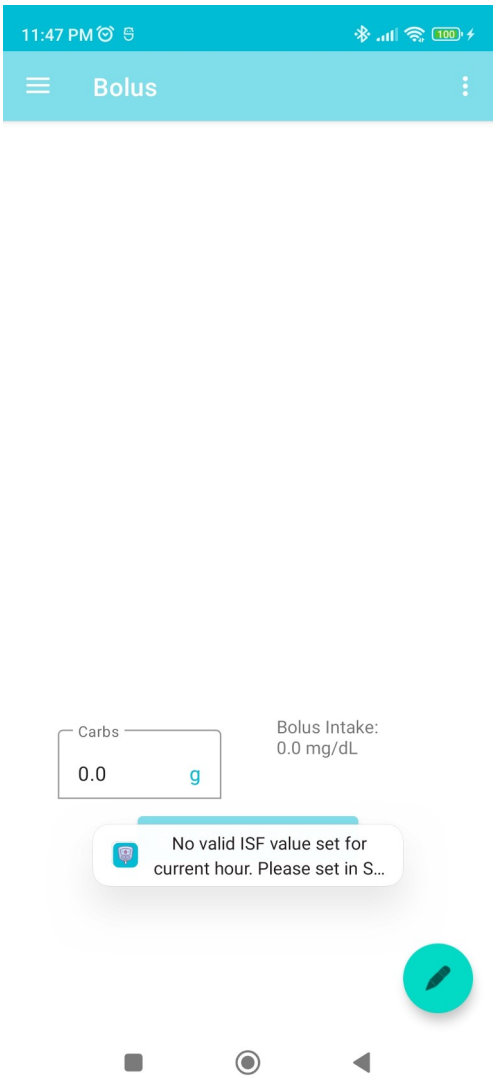


Εικόνα 3.9 Σενάριο Υπολογισμού Bolus – Βασικές Ροές

Οι εναλλακτικές ροές του σεναρίου περιλαμβάνουν τις περιπτώσεις που:

1. Ο χρήστης δεν έχει καταχωρίσει στις ρυθμίσεις τιμή DIA.
2. Ο χρήστης δεν έχει καταχωρίσει στις ρυθμίσεις τιμή DCA.
3. Η εφαρμογή δεν έχει λάβει κάποια πρόσφατη τιμή BG.
4. Ο χρήστης δεν έχει καταχωρίσει στις ρυθμίσεις τιμές εύρους στόχου BG

Για κάθε μία από τις ανωτέρω εναλλακτικές ροές η εφαρμογή εμφανίζει στο χρήστη κατάλληλο μήνυμα ώστε να τον πληροφορήσει τον λόγο που δεν μπορεί να πραγματοποιηθεί ο υπολογισμός. Στην Εικόνα 3.10 που ακολουθεί, παρουσιάζεται στιγμιότυπο οθόνης κατά την προβολή μηνύματος σφάλματος λόγω απουσίας τιμής ISF.



Εικόνα 3.10 Σενάριο Υπολογισμού Bolus – Μήνυμα Σφάλματος

3.5 Άδειες και Δικαιώματα

Το Android χρησιμοποιεί ένα σύστημα δικαιωμάτων (Permissions) για να προστατεύσει τα δεδομένα των χρηστών και να ελέγχει την πρόσβαση των εφαρμογών σε ευαίσθητες λειτουργίες της συσκευής, όπως τοποθεσία, κάμερα, επαφές, χώρος αποθήκευσης και άλλα. Τα δικαιώματα του Android για τρίτες εφαρμογές, χωρίζονται σε τρεις βασικές κατηγορίες [23]:

- Απλά Δικαιώματα (Normal Permissions)
- Δικαιώματα κατά τη λειτουργία (Runtime Permissions)
- Ειδικά Δικαιώματα (Special Permissions)

Τα Normal Permissions, είναι δικαιώματα τα οποία ζητούνται από τις εφαρμογές κατά τη διαδικασία εγκατάστασης αυτών και χορηγούνται αυτόματα από το σύστημα χωρίς την ανάγκη για ρητή έγκριση από τον χρήστη. Τέτοια δικαιώματα αφορούν λειτουργίες που δεν επηρεάζουν την ιδιωτικότητα του χρήστη όπως πρόσβαση στο διαδίκτυο, έλεγχος δόνησης και

άλλα. Η GMA ζητά από το λειτουργικό επτά Normal Permissions. Για να έχει η εφαρμογή πρόσβαση στο διαδίκτυο ζητά το “android.permission.INTERNET” Permission. Για να είναι δυνατό να ενεργοποιεί τη δόνηση κατά τη λειτουργία συναγερμών ζητείται το “android.permission.VIBRATE”. Για να ολοκληρωθούν ορισμένες διεργασίες παρασκηνίου της GMA είναι απαραίτητο να διατηρηθεί ενεργή η συσκευή όσο αυτές λαμβάνουν χώρα. Για αυτό το σκοπό ζητείται το “android.permission.WAKE_LOCK”. Για να διαθέτει η εφαρμογή ενεργή Foreground Service (GlucoseMonitorService) χρειάζεται να ζητηθεί το “android.permission. FOREGROUND_SERVICE”. Για τη χρήση πλήρους οθόνης κατά την προβολή ειδοποιήσεων ζητείται το “android.permission.USE_FULL_SCREEN_INTENT”.

Τα Runtime Permissions, αφορούν ευαίσθητα δεδομένα του χρήστη ή λειτουργίες που μπορούν να επηρεάσουν την ιδιωτικότητά του. Αυτά τα δικαιώματα δεν εγκρίνονται αυτόματα από το σύστημα. Προκειμένου να λάβουν έγκριση για αυτά, οι εφαρμογές ζητούν κατά τη διάρκεια της εκτέλεσης (runtime) από τους χρήστες να δώσουν την έγκρισή τους. Αυτό συνήθως γίνεται με αναδυόμενο παράθυρο στο οποίο υπάρχουν επιλογές allow και do not allow ώστε να ληφθεί η απόφαση του Χρήστη. Η GMA χρειάζεται δύο τέτοια Permissions, για την δυνατότητα εμφάνισης ειδοποιήσεων (Notifications) ζητείται το “android.permission. POST_NOTIFICATIONS”. Για να έχει η εφαρμογή πρόσβαση στα αρχεία του χρήστη ζητείται το “android.permission.READ_EXTERNAL_STORAGE”. Για κάθε ένα από τα προαναφερθέντα δικαιώματα η GMA ζητά από το λειτουργικό να εμφανίσει το αντίστοιχο αναδυόμενο παράθυρο ώστε ο χρήστης να εγκρίνει το Permission.

Τα Special Permissions, είναι τα πλέον ευαίσθητα δικαιώματα και δεν μπορούν να ζητηθούν μέσω του Runtime Permissions System. Αντίθετα, ο χρήστης πρέπει να τα ενεργοποιήσει χειροκίνητα από τις ρυθμίσεις της συσκευής. Οι Ρυθμίσεις αυτές είναι ομοιόμορφα υλοποιημένες σε όλες τις συσκευές Android και οι εφαρμογές δύναται να ανακατευθύνουν τους χρήστες στο αντίστοιχο πεδίο ρυθμίσεων της συσκευής τους ώστε οι χρήστες να εγκρίνουν τα συγκεκριμένα δικαιώματα για λογαριασμό της εκάστοτε εφαρμογής. Η GMA χρειάζεται τρία τέτοια δικαιώματα. Για να μπορεί η εφαρμογή να εμφανίζει στοιχεία γραφικού περιβάλλοντος πάνω από άλλες εφαρμογές (όταν ενεργοποιείται κάποιος συναγερμός), απαιτείται να έχει την ειδική άδεια “android.permission.SYSTEM_ALERT_WINDOW”. Για να μπορεί το GlucoseMonitorService να παραμένει ενεργό καθ΄ όλη τη διάρκεια της ημέρας απαιτείται η GMA να έχει έγκριση για το “android.permission.REQUEST_IGNORE_BATTERY_OPTIMIZATIONS”. Για να μπορεί η εφαρμογή να θέτει ακριβείς (χρονικά) ειδοποιήσεις, είναι απαραίτητο να εγκριθεί το “android.permission.SCHEDULE_EXACT_ALARM”. Για κάθε ένα από τα προαναφερθέντα δικαιώματα, η GMA ενημερώνει και ανακατευθύνει το χρήστη στην αντίστοιχη σελίδα ρυθμίσεων της συσκευής του.

Εκτός από τα προαναφερθέντα, προκειμένου η εφαρμογή να ενημερώνεται από το λειτουργικό κάθε φορά που η συσκευή του χρήστη ανοίγει (power-on) ή επανεκκινείται (restart) ώστε να μπορεί να εκκινείται παράλληλα και το GlucoseMonitorService, ζητούνται τα “android.permission.RECEIVE_BOOT_COMPLETED” και “android.permission.RECEIVE_ QUICKBOOT_POWERON”. Ωστόσο επειδή τα δικαιώματα αυτά ανήκουν στην κατηγορία Signature Permissions δεν διατίθενται σε τρίτες εφαρμογές παρά μόνο εάν πρόκειται για εφαρμογές της Google και των άμεσων συνεργατών της είτε εφαρμογές κατασκευαστών

συσκευών (OEM) όπως οι εταιρείες Samsung, Xiaomi και άλλες. Οι περισσότεροι κατασκευαστές παρέχουν στους χρήστες τη δυνατότητα να δώσουν τα παραπάνω δικαιώματα και σε τρίτες εφαρμογές μέσω των ρυθμίσεων της συσκευής (συνήθως ρυθμίσεις με όνομα Autostart ή συναφή). Επειδή κάθε κατασκευαστής που παρέχει τη ρύθμιση οργανώνει και ονοματίζει διαφορετικά ρυθμίσεις τέτοιου τύπου είναι αδύνατον η GMA να οδηγεί το χρήστη στην αντίστοιχη σελίδα ρυθμίσεων της συσκευής του (όπως συμβαίνει με τα Special Permissions) και απαιτείται από τους χρήστες να εντοπίσουν οι ίδιοι αν και πως δύναται να εγκρίνουν τα δικαιώματα αυτά για την GMA. Η τεχνική υλοποίηση του τρόπου με τον οποίο η GMA λαμβάνει τα απαραίτητα δικαιώματα αναλύεται στο επόμενο κεφάλαιο.

4. Ανάπτυξη Εφαρμογής

4.1 Η αρχιτεκτονική MVVM

Οι εφαρμογές Android εμπεριέχουν διάφορα θεμελιώδη δομικά στοιχεία (app components). Τα δομικά στοιχεία αυτά ανάλογα με το σκοπό που εξυπηρετούν, χωρίζονται σε τέσσερεις βασικές κατηγορίες: τα Στοιχεία γραφικού περιβάλλοντος, με πιο θεμελιώδες στις εφαρμογές Android τις Δραστηριότητες (Activities), τα στοιχεία υπηρεσιών (Services), τα στοιχεία Δεκτών Εκπομπών (Broadcast Receivers) και τα στοιχεία παροχών περιεχομένου (Content Providers) [24]. Η αρχιτεκτονική κάθε εφαρμογής ορίζει τον τρόπο κατά τον οποίο τα δομικά στοιχεία των εφαρμογών οργανώνονται και αλληλεπιδρούν. Η επιλογή κατάλληλης αρχιτεκτονικής είναι ζωτικής σημασίας για την ανάπτυξη, επεκτασιμότητα (scaling), δοκιμή (testing) και συντήρηση των εφαρμογών. Οι αρχιτεκτονικές που επιλέγονται είναι σημαντικό να ακολουθούν βασικές αρχές όπως η Διαχωρισμένη Αρχιτεκτονική (Separation of Concerns) και η Μοναδική Πηγή Αλήθειας (Single Source of Truth). Αυτό επιτυγχάνεται οργανώνοντας τον κώδικα των εφαρμογών με τέτοιο τρόπο ώστε τα διάφορα στρώματα της αρχιτεκτονικής (π.χ Στρώμα Δεδομένων, Στρώμα Διεπαφής Χρήστη κ.τ.λ.), να ορίζονται με σαφήνεια και να είναι όσο το δυνατό περισσότερο ανεξάρτητα μεταξύ τους.

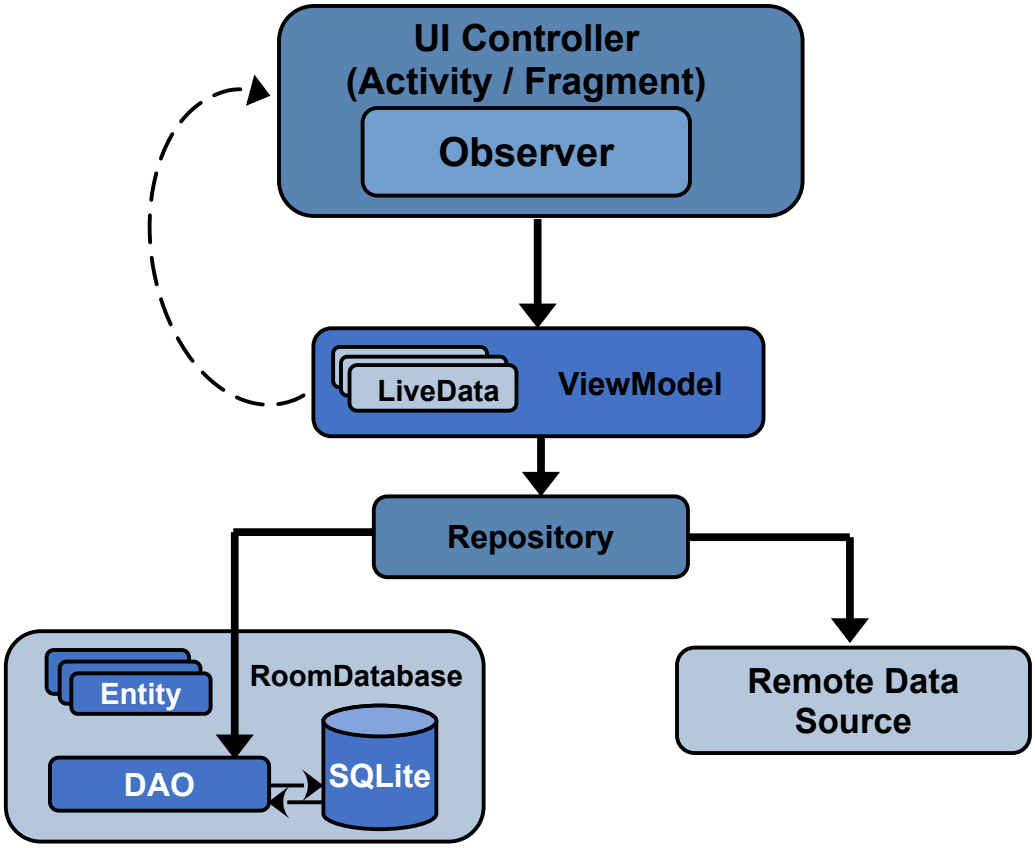
Τα πιο συνήθη μοτίβα αρχιτεκτονικής για εφαρμογές Android είναι τα:

- MVC (Model-View-Controller)
- MVP (Model-View-Presenter)
- MVVM (Model-View-ViewModel)

Το MVC είναι ένα από τα παλαιότερα μοτίβα αρχιτεκτονικής και διαχωρίζει τα στοιχεία της εφαρμογής στο Model το οποίο διαχειρίζεται τα δεδομένα και τη λογική της εφαρμογής, View το οποίο επιφορτίζεται με την εμφάνιση δεδομένων στο χρήστη και τον Controller ο οποίος χειρίζεται τις αλληλεπιδράσεις του χρήστη και ενημερώνει το Model και το View. Ένα βασικό μειονέκτημα του μοτίβου είναι πως ο Controller επιφορτίζεται με πολλές λειτουργικότητες, με αποτέλεσμα να γίνεται περίπλοκος και να δυσχεραίνει τη συντήρηση και δοκιμή του. Το MVP είναι μια βελτίωση του MVC που διαχωρίζει πιο καθαρά τη λογική από το UI. Ουσιαστικά αντικαθιστά τον Controller του MVC με τον Presenter ο οποίος δεν εξαρτάται από το Android framework, βελτιώνοντας τη διαδικασία δοκιμής καθώς επίσης μειώνει την εξάρτηση ανάμεσα σε στοιχεία UI και λογικής. Ωστόσο, καθώς οι εφαρμογές επεκτείνονται και μεγαλώνουν, ο

Presenter, εφόσον δεν οργανωθεί σωστά, γίνεται περίπλοκος με αποτέλεσμα να έχει αντίστοιχα μειονεκτήματα με τον Controller του MVC. Το MVVM, έχει αντίστοιχη δομή με το MVP καθώς ορίζει τα στρώματα : Model το οποίο αντιπροσωπεύει την πηγή των δεδομένων της εφαρμογής (API, Βάση Δεδομένων), View το οποίο αντιπροσωπεύει το UI της εφαρμογής και αλληλεπιδρά με τον χρήστη και το ViewModel το οποίο διαχειρίζεται τα δεδομένα που χρειάζονται για το UI και επικοινωνεί με το Model. Η ουσιαστική διαφορά ανάμεσα στα MVP και MVVM έγκειται στο ότι το ViewModel, διαφοροποιείται από τον Presenter ως προς την εξάρτησή του από τα στοιχεία του View καθώς επίσης και τον τρόπο με τον οποίο επικοινωνεί αλλαγές του επιπέδου Δεδομένων (Model) στο επίπεδο Διεπαφής Χρήστη (View). Σε αντίθεση με τους Presenters, τα ViewModels δεν διατηρούν αναφορές των αντίστοιχων Views αλλά επικοινωνούν με αυτά μέσω πακέτων Σύζευξης Δεδομένων (Data Binding) όπως τα δεδομένα τύπου LiveData του Android. Η απόζευξη του ViewModel από τα Views προσφέρει το πλεονέκτημα του να μην απαιτείται από το ViewModel να καλεί απευθείας μεθόδους του View (όπως συμβαίνει με τον Presenter) πράγμα που απλοποιεί τη διαδικασία δοκιμής και την επεκτασιμότητα της εφαρμογής ενώ παράλληλα αποσιωπεί κινδύνους σφαλμάτων και διαρροής μνήμης (memory leaks).

Η GMA έχει σχεδιαστεί και υλοποιηθεί με βάση το αρχιτεκτονικό μοτίβο MVVM. Πιο συγκεκριμένα, Το Model συμπεριλαμβάνει όλες τις κλάσεις οι οποίες αφορούν τη λογική των δεδομένων της εφαρμογής, την αποθήκευση και την ανάκτησή τους. Μέρος του Model, αποτελούν οι κατηγορίες κλάσεων: Οντοτήτων (Entities), Αντικειμένων Πρόσβασης Δεδομένων (Data Access Objects – DAOs) και Αποθετηρίων Δεδομένων (Data Repositories). Τα Entities, ορίζουν τις οντότητες της τοπικής βάσης δεδομένων και άλλους τύπους δεδομένων τη εφαρμογής. Τα DAOs, είναι κλάσεις που περιλαμβάνουν το σύνολο των ερωτημάτων (queries) της βάσης δεδομένων. Τα Repositories, πραγματοποιούν ασύγχρονα όλες τις προσβάσεις στα δεδομένα της τοπικής βάσης. Οποιαδήποτε αποθήκευση, ενημέρωση και ανάκτηση δεδομένων της βάσης χρειάζεται να γίνει, πραγματοποιείται πάντα μέσω της αντίστοιχης κλάσης Repository. Αυτό αφορά και τα δεδομένα που παράγονται μέσω της εφαρμογής αλλά και τα δεδομένα που λαμβάνονται από εξωγενείς πηγές. Με τον τρόπο αυτό διασφαλίζεται καλύτερα η πιστότητα των δεδομένων (Single source of truth). Το ViewModel συμπεριλαμβάνει κλάσεις οι οποίες παρακολουθούν (μέσω των Repositories) αλλαγές στα δεδομένα της εφαρμογής προκειμένου να εξυπηρετήσουν τις ανάγκες κλάσεων UI. Αυτό επιτυγχάνεται ορίζοντας εντός των ViewModels δεδομένα τύπου LiveData. Τα LiveData είναι μια ειδική κατηγορία παρατηρίσιμων δεδομένων. Σε αντίθεση με ένα κανονικό Observable, τα LiveData λειτουργούν συνειδητά ως προς τον κύκλο ζωής (lifecycle-aware) άλλων στοιχείων της εφαρμογής, όπως τα Activities, Fragments ή Services. Αυτή η ιδιότητα διασφαλίζει ότι τα LiveData ενημερώνουν μόνο τους παρατηρητές των στοιχείων της εφαρμογής που βρίσκονται σε ενεργή κατάσταση κύκλου ζωής [26]. Το View συμπεριλαμβάνει όλες τις κλάσεις που αφορούν τη Διεπαφή Χρήστη (UI) της εφαρμογής (Activities, Fragments, Dialogs). Οι κλάσεις αυτές περιέχουν αποκλειστικά τον κώδικα που αφορά τα αντίστοιχα στοιχεία UI και ενημερώνονται για αλλαγές στα δεδομένα που τις αφορούν μέσω παρατηρητών (Observers) οι οποίοι παρατηρούν συγκεκριμένα LiveData των ViewModels και ενημερώνονται αυτόματα όταν τα δεδομένα αυτά αλλάξουν, ενημερώνοντας παράλληλα το UI για την αλλαγή.



Σχήμα 4.1 Αρχιτεκτονικό μοτίβο Εφαρμογής

4.2 Βιβλιοθήκες Λογισμικού

Σε αυτή την ενότητα παρουσιάζονται οι βασικές βιβλιοθήκες λογισμικού, που χρησιμοποιήθηκαν στην ανάπτυξη της εφαρμογής. Για κάθε μια από αυτές θα περιγραφεί ο ρόλος, ο τρόπος λειτουργίας και ο τρόπος ενσωμάτωσης στην εφαρμογή.

4.2.1 Room

Το Room, είναι μια βιβλιοθήκη της πλατφόρμας Android Jetpack της Google και παρέχει ένα επίπεδο αφαίρεσης πάνω από την SQLite βάση δεδομένων. Διευκολύνει την πρόσβαση και διαχείριση των δεδομένων με έναν ευέλικτο και ασφαλή τρόπο. Η λειτουργία του Room βασίζεται σε τρία κύρια στοιχεία:

1. **Οντότητες (Entities)**: Αντιπροσωπεύουν τους πίνακες της βάσης δεδομένων. Κάθε κλάση που δημιουργείται με την επισημείωση (Annotation) @Entity αντιστοιχεί σε έναν πίνακα, και τα πεδία της κλάσης αντιστοιχούν σε στήλες του πίνακα.
2. **Αντικείμενα Πρόσβασης Δεδομένων (Data Access Objects - DAOs)**: Είναι διεπαφές Java (Java Interfaces) και επισημειώνονται με @Dao, περιέχουν τις μεθόδους για την πρόσβαση στη βάση δεδομένων, όπως εισαγωγή, διαγραφή και ανάκτηση δεδομένων. Οι μέθοδοι αυτές ορίζονται με SQL ερωτήματα και επισημειώνονται με κατάλληλα Annotations, όπως @Insert, @Delete και @Query.

3. **Κλάση Βάσης Δεδομένων (Database Class)**: Είναι η κύρια πρόσβαση στη βάση δεδομένων και ορίζεται με το Annotation @Database. Περιλαμβάνει τις οντότητες πινάκων συνοδευόμενες από τα αντίστοιχα DAOs και παρέχει σύνδεση με τη βάση δεδομένων.

Το βασικό πλεονέκτημα που προσφέρει η χρήση της βιβλιοθήκης Room είναι η επαλήθευση των ερωτημάτων SQL κατά το στάδιο μεταγλώττισης, γεγονός που ελαχιστοποιεί τα σφάλματα κατά την εκτέλεση. Επιπλέον, το Room ενσωματώνεται άμεσα με αρχιτεκτονικά στοιχεία του Android όπως το LiveData επιτρέποντας την παρακολούθηση αλλαγών των δεδομένων σε πραγματικό χρόνο και την εκτέλεση λειτουργιών σε ασύγχρονα νήματα εκτέλεσης [27].

Η βάση δεδομένων της GMA είναι υλοποιημένη με τη χρήση της βιβλιοθήκης Room. Στις υποενότητες 4.3.1 και 4.3.2 παρουσιάζονται αναλυτικά οι κλάσεις Entity και DAO της εφαρμογής και για την κάθε μία, παρουσιάζεται ποια Annotations της βιβλιοθήκης αξιοποιούν. Στην υποενότητα 4.3.3 αναλύεται η κλάση της βάσης δεδομένων της εφαρμογής, η οποία επισημειώνεται με το Annotation @Database και περιλαμβάνει τα Entities και DAOs.

4.2.2 Βιβλιοθήκες Διεπαφής Χρήστη

MPAndroidChart

Η MPAndroidChart, είναι μια δημοφιλής βιβλιοθήκη Android, που επιτρέπει τη δημιουργία διαδραστικών και οπτικά ελκυστικών γραφημάτων. Είναι ανοικτού κώδικα, διαθέσιμη μέσω του GitHub και διατίθεται ελεύθερα υπό την άδεια χρήσης Apache License, Version 2.0 [28]. Υποστηρίζει διάφορους τύπους γραφημάτων και είναι προσαρμόσιμη. Χρησιμοποιώντας την βιβλιοθήκη δημιουργείται και παραμετροποιείται το διάγραμμα τιμών γλυκόζης της Αρχικής Οθόνης της εφαρμογής.

Material Components for Android

Η βιβλιοθήκη Material, παρέχει μια συλλογή στοιχείων διεπαφής χρήστη που ακολουθούν τις αρχές του Material Design, και διευκολύνουν στη δημιουργία σύγχρονων και συνεπών διεπαφών χρήστη στις εφαρμογές. Εντός των διαφόρων Δραστηριοτήτων και Υποτμημάτων της εφαρμογής χρησιμοποιούνται διάφορα στοιχεία UI που παρέχει η βιβλιοθήκη όπως πεδία εισαγωγής κειμένου, στοιχεία πλοήγησης και άλλα.

4.2.3 Firebase

Όπως έχει αναφερθεί και σε προηγούμενα κεφάλαια, παράλληλα με την εφαρμογή λειτουργεί διαδικτυακά, Server που παρέχει τις δυνατότητες εγγραφής νέου χρήστη, επαλήθευσης στοιχείων εισόδου χρήστη και επαναφοράς κωδικού χρήστη. Για την επίτευξη αυτής της λειτουργικότητας έχει δημιουργηθεί μέσω της κονσόλας του Firebase μια διαδικτυακή εφαρμογή η οποία αξιοποιεί τη λειτουργικότητα Authentication που παρέχει το εργαλείο. Πιο συγκεκριμένα η εφαρμογή Firebase είναι παραμετροποιημένη έτσι ώστε να

ανταποκρίνεται μόνο σε requests τα οποία περιέχουν ένα συγκεκριμένο αναγνωριστικό το οποίο συνδέεται με το πακέτο της GMA διασφαλίζοντας πως η πρώτη θα είναι μη αποκρίσιμη σε άλλες εφαρμογές ή μεθόδους αποστολής request. Η εφαρμογή Firebase περιλαμβάνει τη λειτουργικότητα Authentication που προσφέρει η πλατφόρμα. Η λειτουργικότητα έχει παραμετροποιηθεί ώστε να είναι της μορφής email-password και να προσφέρει ανάκτηση κωδικού μέσω αποστολής ηλεκτρονικού μηνύματος για την επαναρύθμιση του κωδικού στο e-mail εγγραφής.

Η GMA αποστέλλει τα κατάλληλα request στην εφαρμογή Firebase μέσω του Firebase Authentication SDK με όνομα FirebaseAuth. Μέσω αυτής δίνεται πρόσβαση στις μεθόδους *createUserWithEmailAndPassword*, *signInWithEmailAndPassword* και *sendPasswordResetEmail*, οι οποίες εκτελούν τις διαδικασίες εγγραφής, εισόδου και επαναφοράς κωδικού αντίστοιχα. Αναλαμβάνουν την αποστολή των δεδομένων στο Server αλλά και την λήψη της απάντησης από αυτόν. Επειδή η αποστολή των δεδομένων στο Server και απάντηση αυτού δύναται να πάρουν αρκετό χρόνο αυτή η δοσοληψία πραγματοποιείται ασύγχρονα σε σχέση με την εκτέλεση της εφαρμογής στη συσκευή του χρήστη. Κατά την κλήση της κάθε μεθόδου ανταλλάσσονται με το Server 2 πακέτα, ένα πακέτο request(HTTPS POST) και ένα πακέτο response, με το τελευταίο να λαμβάνεται ασύγχρονα και να ενημερώνει το γραφικό περιβάλλον της εφαρμογής ανάλογα με την επιτυχία ή αποτυχία εκτέλεσης του request στο Server.

4.3 Ανάλυση Κώδικα Εφαρμογής

Σε αυτή την ενότητα αναλύεται ο κώδικας Android της GMA. Οι διάφορες κλάσεις οι οποίες απαρτίζουν την εφαρμογή, έχουν κατηγοριοποιηθεί ανάλογα με το είδος και τη λειτουργία τους. Σε κάθε υποενότητα αναλύεται μια διαφορετική κατηγορία κλάσεων. Για κάθε κλάση της εκάστοτε κατηγορίας αναλύεται η λειτουργία που επιτελεί και παρουσιάζονται τα αντίστοιχα κρίσιμα κομμάτια κώδικα.

4.3.1 Μοντέλα Δεδομένων

Σε αυτή την υποενότητα, αναλύονται οι κλάσεις που αναπαριστούν δεδομένα της εφαρμογής. Κάποια από αυτά τα δεδομένα αποθηκεύονται στην τοπική SQLite βάση δεδομένων (υποενότητα 4.3.3) ενώ κάποια άλλα αποθηκεύονται στο xml αρχείο κοινών προτιμήσεων (υποενότητα 4.3.9) της εφαρμογής. Αρχικά θα αναλυθούν οι κλάσεις που αναπαριστούν δεδομένα της βάσης και στη συνέχεια οι κλάσεις που αναπαριστούν δεδομένα κοινών προτιμήσεων. Όλες οι κλάσεις που παρουσιάζονται σε αυτή την υποενότητα διαθέτουν τον αντίστοιχο Constructor καθώς επίσης getter και setter μεθόδους για κάθε πεδίο, εκτός των πεδίων ID των κλάσεων Οντοτήτων Room.

Όπως έχει αναφερθεί και στην προηγούμενη ενότητα, η SQLite βάση δεδομένων έχει αναπτυχθεί με χρήση της βιβλιοθήκης Room. Ένα από τα δομικά στοιχεία της βάσης δεδομένων, είναι οι Οντότητες τις οποίες εμπεριέχει και η κάθε μια αποτελεί ένα ξεχωριστό πίνακα. Αυτό επιτυγχάνεται προσθέτοντας στις κλάσεις των δεδομένων συγκεκριμένα Annotations της βιβλιοθήκης Room προκειμένου να οριστούν οι απαραίτητες παράμετροι της οντότητας που η κάθε κλάση αναπαριστά. Με βάση τις παραμέτρους αυτές, η βάση

δημιουργεί και παραμετροποιεί τους αντίστοιχους πίνακες. Πιο συγκεκριμένα χρησιμοποιούνται τα Annotations:

- **@Entity(tableName)**: Το annotation Entity, τοποθετείται πάνω από τη δήλωση της κλάσης και ορίζει ότι θα δημιουργηθεί στη βάση δεδομένων πίνακας με στήλες τα πεδία της κλάσης και όνομα το όνομα της ιδιότητας (attribute) tableName.
- **@PrimaryKey(autoGenerate)**: Το annotation PrimaryKey, τοποθετείται πάνω από τη δήλωση του πεδίου της κλάσης, το οποίο επιθυμούμε να λειτουργεί ως πρωτεύον κλειδί (Primary Key) του πίνακα. Το attribute autoGenerate είναι τύπου Boolean και καθορίζει εάν η τιμή του πρωτεύοντος κλειδιού θα δημιουργείται αυτόματα από τη βάση δεδομένων. Αν η τιμή οριστεί σε true, η Room θα αυξάνει αυτόματα την τιμή του πεδίου σε κάθε νέα εγγραφή, όπως συμβαίνει με τα auto-increment πεδία στις βάσεις δεδομένων.
- **@ColumnInfo(name)**: Το annotation ColumnInfo, τοποθετείται πάνω από τα πεδία μεταβλητών μέλους (member variables) της κλάσης και μέσω του attribute name καθορίζει το όνομα της στήλης του πίνακα της βάσης δεδομένων που αντιστοιχεί στο member variable της κλάσης. Εάν δεν τοποθετηθεί το Annotation η στήλη θα ονοματιστεί με το ίδιο όνομα του member variable.

Οι κλάσεις που αναπαριστούν Οντότητες της βάσης δεδομένων είναι:

1. BGModel

Η κλάση BGModel, αναπαριστά τα δεδομένα γλυκόζης του χρήστη που περιγράφονται στην υποενότητα 3.1.1.

Ορίζει τον πίνακα με όνομα BGTable

```
@Entity(tableName = "BGTable")
public class BGModel
```

Έχει πρωτεύον κλειδί το πεδίο ID. Το κλειδί δημιουργείται και αυξάνεται αυτόματα από τη βάση δεδομένων

```
@PrimaryKey(autoGenerate = true)
private int ID;
```

Έχει πεδία-στήλες

```
private int ID;
private int value;
private Instant time;
```

2. AlertsModel

Η κλάση AlertsModel, αναπαριστά τα δεδομένα συναγερμών που περιγράφονται στην υποενότητα 3.1.2.

Ορίζει τον πίνακα με όνομα AlertsTable

```
@Entity(tableName = "AlertsTable")
public class AlertsModel
```

Έχει πρωτεύον κλειδί το πεδίο ID. Το κλειδί δημιουργείται και αυξάνεται αυτόματα από τη βάση δεδομένων

```
@PrimaryKey(autoGenerate = true)
private int ID;
```

Έχει πεδία-στήλες

```
private int ID;
private int value;
private String operand;
```

3. NotesModel

Η κλάση NotesModel, αναπαριστά τα δεδομένα σημειώσεων που περιγράφονται στην υποενότητα 3.1.3.

Ορίζει τον πίνακα με όνομα NotesTable

```
@Entity(tableName = "NotesTable")
public class NotesModel
```

Έχει πρωτεύον κλειδί το πεδίο ID. Το κλειδί δημιουργείται και αυξάνεται αυτόματα από τη βάση δεδομένων

```
@PrimaryKey(autoGenerate = true)
private int ID;
```

Έχει πεδία-στήλες

```
private int ID;
private String text;
private String time;
```

4. COBRecod

Η κλάση COBRecord, αναπαριστά τα δεδομένα ενεργών υδατανθράκων (Carbs On Board) που περιγράφονται στην υποενότητα 3.1.4.

Ορίζει τον πίνακα με όνομα COBRecordsTable

```
@Entity(tableName = "COBRecordsTable")
public class COBRecord
```

Έχει πρωτεύον κλειδί το πεδίο ID. Το κλειδί δημιουργείται και αυξάνεται αυτόματα από τη βάση δεδομένων.

```
@PrimaryKey(autoGenerate = true)
private int ID;
```

Έχει πεδία-στήλες

```
private int ID;
private double cob;
private long time;
```

5. EventsModel

Η κλάση EventsModel, αναπαριστά τα δεδομένα συμβάντων που περιγράφονται στην υποενότητα 3.1.5.

Ορίζει τον πίνακα με όνομα EventsTable

```
@Entity(tableName = "EventsTable")
public class EventsModel
```

Έχει πρωτεύον κλειδί το πεδίο ID. Το κλειδί δημιουργείται και αυξάνεται αυτόματα από τη βάση δεδομένων

```
@PrimaryKey(autoGenerate = true)
private int ID;
```

Έχει πεδία-στήλες

```
private int ID;
private String type;
private String message;
private Instant time;
private int bg;
```

Εκτός από τις κλάσεις Οντοτήτων της βάσης δεδομένων, που συμπεριλαμβάνουν τα Annotations της βιβλιοθήκης Room. Υπάρχουν και συμβατικές κλάσεις που αναπαριστούν δεδομένα της εφαρμογής που αποθηκεύονται στο αρχείο προτιμήσεων χρήστη. Αυτές είναι:

1. BasalRange

Η κλάση BasalRange, αναπαριστά δεδομένα βασικού ρυθμού έγχυσης ινσουλίνης του χρήστη (υποενότητα 3.1.4). Τα δεδομένα αυτά εισάγονται μέσω της οθόνης ρυθμίσεων και αφορούν την ποσότητα μονάδων ινσουλίνης ανά ώρα η οποία εγχύεται αυτόματα στο χρήστη για ένα συγκεκριμένο χρονικό διάστημα της ημέρας (σε ώρες).

Έχει τα πεδία

```
private long startHour;
private long endHour;
private double basalValue;
```

2. CRRange

Η κλάση CRRange, αναπαριστά δεδομένα αναλογίας ινσουλίνης προς υδατάνθρακες (CR- υποενότητα 3.1.4). Τα δεδομένα αυτά εισάγονται μέσω της οθόνης ρυθμίσεων και αφορούν την ποσότητα υδατανθράκων που καλύπτει μια μονάδα ινσουλίνης ανά για ένα συγκεκριμένο χρονικό διάστημα της ημέρας (σε ώρες).

Έχει τα πεδία

```
private long startHour;
private long endHour;
private int icrValue;
```

3. ISFRange

Η κλάση ISFRange, αναπαριστά δεδομένα ευαισθησίας στην ινσουλίνη (ISF- υποενότητα 3.1.4). Τα δεδομένα αυτά εισάγονται μέσω της οθόνης ρυθμίσεων και αφορούν την ποσότητα γλυκόζης στο αίμα που μειώνεται από μια μονάδα ινσουλίνης, για ένα συγκεκριμένο χρονικό διάστημα της ημέρας (σε ώρες).

Έχει τα πεδία

```
private long startHour;
private long endHour;
private int isfValue;
```

4.3.2 Αντικείμενα Πρόσβασης Δεδομένων

Σε αυτή την υποενότητα, αναλύονται οι διεπαφές (Interfaces) Αντικειμένων Πρόσβασης Δεδομένων (DAO) της βάσης δεδομένων (υποενότητα 4.3.3). Περιέχουν τις μεθόδους ερωτημάτων που πραγματοποιούνται στους πίνακες της βάσης. Όπως και στις κλάσεις Οντοτήτων που περιγράφηκαν στην προηγούμενη υποενότητα, έτσι και στα Interfaces των DAOs εισάγονται συγκεκριμένα Annotations της βιβλιοθήκης Room και αντιστοιχίζουν queries της βάσης με μεθόδους του Interface. Στα DAOs χρησιμοποιούνται τα Annotations:

- **@Dao**: Τοποθετείται πάνω από τη δήλωση του Interface και ορίζει ότι η διεπαφή είναι DAO. Χωρίς αυτό, το Room δεν αναγνωρίζει το αρχείο ως μέρος της βάσης δεδομένων.
- **@Insert**: Τοποθετείται πάνω από τη μέθοδο εισαγωγής εγγραφών στη βάση. Η μέθοδος λαμβάνει ως παράμετρο ένα αντικείμενο κάποιας κλάσης Οντότητας της

βάσης και εισάγει το αντικείμενο αυτό στον πίνακα που αντιστοιχεί στην κλάση Οντότητας του αντικειμένου. Σε ορισμένα @Insert έχει οριστεί συγκεκριμένη στρατηγική επίλυσης

- **@Update**: Τοποθετείται πάνω από τη μέθοδο ενημέρωσης εγγραφών της βάσης. Η μέθοδος λαμβάνει ώς παράμετρο ένα αντικείμενο κάποιας κλάσης Οντότητας της βάσης και ενημερώνει την εγγραφή με ίδιο PrimaryKey στον πίνακα που αντιστοιχεί στην κλάση Οντότητας του αντικειμένου.
- **@Delete**: Τοποθετείται πάνω από τη μέθοδο διαγραφής εγγραφών από τη βάση. Η μέθοδος λαμβάνει ως παράμετρο ένα αντικείμενο κάποιας κλάσης Οντότητας της βάσης και διαγράφει την εγγραφή με ίδιο PrimaryKey στον πίνακα που αντιστοιχεί στην κλάση Οντότητας του αντικειμένου.
- **@Query**: Τοποθετείται πάνω από μεθόδους προσαρμοσμένων SQL ερωτημάτων. Σε αντίθεση με τα προηγούμενα annotations, στα ερωτήματα του @Query παρέχεται ως παράμετρος η συμβολοσειρά με το SQL statement που είναι επιθυμητό να εκτελεστεί. Εάν το SQL statement δέχεται παραμέτρους τότε αυτές συμπεριλαμβάνονται στο SQL statement με σήμανση ":" ακολουθούμενη από το όνομα της παραμέτρου και αντιστοιχίζονται με τα αντίστοιχα ορίσματα στη μέθοδο της Interface.

Έχει επιλεχθεί να συνταχθεί ξεχωριστό DAO για κάθε πίνακα της βάσης δεδομένων. Τα DAOs της εφαρμογής είναι τα:

1. BGDao
   Συμπεριλαμβάνει τις μεθόδους ερωτημάτων στα δεδομένα του πίνακα BGTable της κλάσης BGModel. Επισημαίνεται με το @Dao annotation ώστε να οριστεί ότι η διεπαφή είναι DAO.

```
@Dao
public interface BGDao
```

Ορίζονται οι βασικές μέθοδοι εισαγωγής, ενημέρωσης και διαγραφής εγγραφών :

```
@Insert(onConflict = OnConflictStrategy.REPLACE)
void insert(BGModel bg);
@Update
void update(BGModel bg);
@Delete
void delete(BGModel bg);
```

και οι μέθοδοι προσαρμοσμένων SQL ερωτημάτων:

i. *deleteAllBG*: Διαγράφει όλες τις εγγραφές του πίνακα BGTable.

```
@Query("DELETE FROM BGTable")
   void deleteAllBG();
```

ii. *getAllBG*: Ανακτά όλες τις εγγραφές του πίνακα BGTable σε λίστα. Ταξινομημένες με βάση το ID σε φθίνουσα σειρά.

```
@Query("SELECT * FROM BGTable ORDER BY ID DESC")
   LiveData<List<BGModel>> getAllBG();
```

iii. *getLastBGQ*: Ανακτά την πιο πρόσφατη εγγραφή του πίνακα BGTable.

```
@Query("SELECT * FROM BGTable WHERE ID = (SELECT MAX(ID) FROM BGTable)")
   LiveData<BGModel> getLastBG();
```

iv. *findBGByDate*: Ανακτά όλες τις τιμές BG για μια συγκεκριμένη ημερομηνία. Δέχεται ως παραμέτρους τη χρονοσφραγίδα έναρξης της ημέρας και τη χρονοσφραγίδα λήξης της ημέρας. Με βάση αυτές τις χρονοσφραγίδες, επιστρέφει μια λίστα με όλες τις τιμές BG, η χρονοσφραγίδα των οποίων ανήκει στο διάστημα [Χρονοσφραγίδα έναρξης, Χρονοσφραγίδα λήξης]. Οι τιμές ταξινομούνται σε φθίνουσα σειρά με βάση το ID

```
@Query("SELECT * FROM BGTable WHERE time BETWEEN :startOfDay AND :endOfDay
   ORDER BY ID DESC"")
   List<BGModel> findBGByDate(Long startOfDay, Long endOfDay);
```

2. AlertsDao

Συμπεριλαμβάνει τις μεθόδους ερωτημάτων στα δεδομένα του πίνακα AlertsTable της κλάσης AlertModel. Επισημαίνεται με το @Dao annotation ώστε να οριστεί ότι η διεπαφή είναι DAO.

```
@Dao
public interface AlertsDao
```

Ορίζονται οι βασικές μέθοδοι εισαγωγής, ενημέρωσης και διαγραφής εγγραφών :

```
@Insert
void insert(AlertsModel alert);
@Update
void update(AlertsModel alert);
@Delete
void delete(AlertsModel alert);
```

και οι μέθοδοι προσαρμοσμένων SQL ερωτημάτων:

i. *deleteAllAlerts*: Διαγράφει όλες τις εγγραφές του πίνακα AlertsTable.

```
@Query("DELETE FROM AlertsTable")
void deleteAllAlerts();
```

ii. *getAllAlerts*: Ανακτά όλες τις εγγραφές του πίνακα AlertsTable σε λίστα. Ταξινομημένες με βάση το ID σε φθίνουσα σειρά.

```
@Query("SELECT * FROM AlertsTable ORDER BY ID DESC")
LiveData<List<AlertsModel>> getAllAlerts();
```

3. NotesDao

Συμπεριλαμβάνει τις μεθόδους ερωτημάτων στα δεδομένα του πίνακα NotesTable της κλάσης NotesModel. Επισημαίνεται με το @Dao annotation ώστε να οριστεί ότι η διεπαφή είναι DAO.

```
@Dao
public interface NotesDao
```

Ορίζονται οι βασικές μέθοδοι εισαγωγής, ενημέρωσης και διαγραφής εγγραφών :

```
@Insert
void insert(NotesModel note);
@Update
void update(NotesModel note);
@Delete
void delete(NotesModel note);
```

και οι μέθοδοι προσαρμοσμένων SQL ερωτημάτων:

i. *deleteAllNotes*: Διαγράφει όλες τις εγγραφές του πίνακα NotesTable.

```
@Query("DELETE FROM NotesTable")
void deleteAllNotes();
```

ii. *getAllNotes*: Ανακτά όλες τις εγγραφές του πίνακα NotesTable σε λίστα. Ταξινομημένες με βάση το ID σε φθίνουσα σειρά.

```
@Query("SELECT * FROM NotesTable ORDER BY ID DESC")
LiveData<List<AlertsModel>> getAllNotes();
```

iii. *findNoteByDate*: Επιστρέφει όλες τις εγγραφές μιας συγκεκριμένης ημερομηνίας σε λίστα, ταξινομημένες με βάση το ID σε φθίνουσα σειρά. Δέχεται ώς παράμετρο την ημερομηνία.

```
@Query("SELECT * FROM NotesTable WHERE time LIKE '%' || :queryDate || '%' ORDER BY
ID DESC")
List<NotesModel> findNoteByDate(String queryDate);
```

4. COBRecordDao

Συμπεριλαμβάνει τις μεθόδους ερωτημάτων στα δεδομένα του πίνακα COBRecordsTable της κλάσης COBRecord. Επισημαίνεται με το @Dao annotation ώστε να οριστεί ότι η διεπαφή είναι DAO.

```
@Dao
public interface COBRecordDao
```

Ορίζεται η βασική μέθοδος εισαγωγής δεδομένων:

```
@Insert
void insert(COBRecord record);
```

και οι μέθοδοι προσαρμοσμένων SQL ερωτημάτων:

i. *getRecordsFromTime*: Ανακτά όλες τις εγγραφές του πίνακα COBRecordsTable για ένα συγκεκριμένο χρονικό διάστημα. Δέχεται ώς παραμέτρους τη χρονοσφραγίδα έναρξης της ημέρας και τη χρονοσφραγίδα λήξης του διαστήματος. Με βάση αυτές τις χρονοσφραγίδες, επιστρέφει μια λίστα με όλες τις εγγραφές, η χρονοσφραγίδα των οποίων ανήκει στο διάστημα [Χρονοσφραγίδα έναρξης, Χρονοσφραγίδα λήξης]. Οι τιμές ταξινομούνται σε φθίνουσα σειρά με βάση το ID

```
@Query("SELECT * FROM COBRecordsTable WHERE time BETWEEN :startTime AND:endTime
ORDER BY ID DESC")
List<COBRecord> getRecordsFromTime(long startTime,long endTime);
```

ii. *deleteOldRecords*: Διαγράφει όλες τις εγγραφές του πίνακα COBRecordsTable η χρονοσφραγίδα των οποίων είναι παλαιότερες της χρονοσφρακίδας ορίου. Δέχεται σαν παράμετρο τη χρονοσφραγίδα ορίου.

```
@Query("DELETE FROM COBRecordsTable WHERE time < :thresholdTime")
void deleteOldRecords(long thresholdTime);
```

5. EventsDao

Συμπεριλαμβάνει τις μεθόδους ερωτημάτων στα δεδομένα του πίνακα EventsTable της κλάσης EventsModel. Επισημαίνεται με το @Dao annotation ώστε να οριστεί ότι η διεπαφή είναι DAO.

```
@Dao
public interface EventsDao
```

Ορίζονται οι βασικές μέθοδοι εισαγωγής, ενημέρωσης και διαγραφής εγγραφών :

```
@Insert(onConflict = OnConflictStrategy.REPLACE)
void insert(EventsModel event);
@Update
void update(EventsModel event);
@Delete
void delete(EventsModel event);
```

και οι μέθοδοι προσαρμοσμένων SQL ερωτημάτων:

i. *findEventByDate*: Ανακτά όλες τις εγγραφές του πίνακα EventsTable, για μια συγκεκριμένη ημέρα. Δέχεται ως παραμέτρους τις χρονοσφραγίδες έναρξης και λήξης της ημέρας. Επιστρέφει τις εγγραφές σε λίστα ταξινομημένες με βάση το ID με φθίνουσα σειρά.

```
@Query("SELECT * FROM EventsTable WHERE time BETWEEN :startOfDay AND :endOfDay
ORDER BY ID DESC")
List<EventsModel> findEventByDate(Long startOfDay, Long endOfDay);
```

ii. *findBolusEventsWithinDia*: Ανακτά όλες τις εγγραφές του πίνακα EventsTable, που έχουν συγκεκριμένο τύπο (TYPE_BOLUS ή TYPE_ALERT) και έλαβαν χώρα σε συγκεκριμένο χρονικό διάστημα. Ως παράμετροι δίνονται ο τύπος του συμβάντος που μας ενδιαφέρει, η χρονοσφραγίδα έναρξης του χρονικού διαστήματος και η χρονοσφραγίδα λήξης αυτού. Οι εγγραφές επιστρέφονται σε λίστα ταξινομημένες σε φθίνουσα σειρά με βάση το ID.

```
@Query("SELECT * FROM EventsTable WHERE type = :type AND time BETWEEN :startTime
AND :endTime ORDER BY ID DESC")
List<EventsModel> findBolusEventsWithinDia(String type, long startTime, long
endTime);
```

4.3.3 Βάση Δεδομένων

Η κλάση CGMDatabase, αποτελεί την κεντρική κλάση της τοπικής βάσης δεδομένων (με όνομα cgm_database). Μέσω αυτής δημιουργείται και παραμετροποιείται η βάση δεδομένων, ορίζονται οι πίνακες οντοτήτων και παρέχεται πρόσβαση στα DAOs που επιτρέπουν την αλληλεπίδραση με τον αποθηκευτικό χώρο. Η CGMDatabase δηλώνεται ως abstract και κληρονομεί από την RoomDatabase, η οποία είναι η βασική κλάση για όλες τις βάσεις δεδομένων Room και κάθε κλάση που επισημειώνεται με @Database θα πρέπει να κληρονομεί από αυτή. Η κλάση επισημειώνεται με:

```
@Database(entities = {AlertsModel.class, BGModel.class, NotesModel.class,
EventsModel.class, COBRecord.class}, version = 1)
@TypeConverters({Converters.class})
public abstract class CGMDatabase extends RoomDatabase
```

Το @Database ορίζει ότι η κλάση είναι μια Room Database. Έχει attributes τα entities και version. Μέσω του attribute entities ορίζονται οι κλάσεις Οντοτήτων (@Entity) που συμπεριλαμβάνονται και για κάθε μια από αυτές δημιουργείται ξεχωριστός πίνακας με στήλες τα πεδία της αντίστοιχης κλάσης. Μέσω του attribute version ορίζεται η τρέχουσα έκδοση της βάσης δεδομένων, ώστε να μπορεί να διαχειριστεί αλλαγές στη δομή της μέσω migrations. Με το Annotation @TypeConverters ορίζεται η κλάση με βάση την οποία μετατρέπονται τα σύνθετα αντικείμενα δεδομένων σε απλούς τύπους που η βάση μπορεί να αποθηκεύσει. Στην προκειμένη περίπτωση ορίζουμε την κλάση Converters, η οποία περιέχει όλες τις απαραίτητες μετατροπές και αναλύεται στην υποενότητα 4.3.9.

Η κλάση CGMDatabase ακολουθεί το Singleton Pattern για να διασφαλίσει ότι υπάρχει μόνο μία και μοναδική instance της βάσης δεδομένων κατά τη διάρκεια ζωής της εφαρμογής. Αυτό επιτυγχάνεται δηλώνοντας ένα instance ως static volatile (το volatile εξασφαλίζει ότι οι αλλαγές στην instance είναι άμεσα ορατές σε όλα τα threads). Το οποίο διαχειριζόμαστε μέσω της μεθόδου *getDatabase*:

```
private static volatile CGMDatabase instance;

        public static CGMDatabase getDatabase(final Context context) {
               if (instance == null) {
                   synchronized (CGMDatabase.class) {
                       if (instance == null) {
```

```
                    instance=Room.databaseBuilder(context.getApplicationCont
                ext(),CGMDatabase.class, "cgm_database")
                fallbackToDestructiveMigration().build();
                }
            }
        }
    return instance
}
```

Την πρώτη φορά που εκτελείται η *getDatabase*, δημιουργείται η βάση δεδομένων με όνομα cgm_database, και κάθε φορά που καλείται έκτοτε επιστρέφει το instance με την ήδη δημιουργημένη βάση δεδομένων. Αν η εφαρμογή κλείσει (και το GlucoseMonitorService), το instance γίνεται null και με νέο άνοιγμα της εφαρμογής ξαναεκτελείται η *Room.databaseBuilder* η οποία δεν ξαναδημιουργεί τη βάση αλλά επιστρέφει την ήδη δημιουργημένη βάση με όνομα cgm_database. Το τμήμα κώδικα που δημιουργεί το νέο instance εκτελείται σε συγχρονισμένο block (synchronized) προκειμένου να αποφευχθεί η ταυτόχρονη δημιουργία instance από πολλά Threads. Στη μέθοδο *Room.databaseBuilder*, δίνονται ως ορίσματα το context.getApplicationContext() με βάση το οποίο διασφαλίζεται οτι η βάση δεδομένων έχει διάρκεια ζωής εκτέλεσης όσο η εφαρμογή. Το CGMDatabase.class προσδιορίζει την κλάση της βάσης δεδομένων και η παράμετρος “cgm_database” προσδιορίζει το όνομα της βάσης. Η μέθοδος *fallbackToDestructiveMigration* προσδιορίζει τη διαχείριση των migrations και πιο συγκεκριμένα ορίζει στην εφαρμογή να διαγράψει τη βάση και να δημιουργήσει νέα, σε περίπτωση αλλαγής της τιμής της παραμέτρου version. Η μέθοδος *build* ολοκληρώνει τη δημιουργία και επιστρέφει το instance της βάσης. Η κλάση CGMDatabase συμπεριλαμβάνει τις εξής αφηρημένες μεθόδους:

```
public abstract AlertsDao alertsDao();
public abstract BGDao bgDao();
public abstract NotesDao notesDao();
public abstract EventsDao eventsDao();
public abstract COBRecordDao cobRecordDao();
```

Οι μέθοδοι *alertsDao*(), *bgDao*(), *notesDao*(), *eventsDao*() και *cobRecordDao*() είναι abstract μέθοδοι που επιστρέφουν τα αντίστοιχα DAO interfaces. Το Room δημιουργεί αυτόματα τις υλοποιήσεις τους κατά την εκτέλεση, επιτρέποντας την πρόσβαση στη βάση δεδομένων μέσω αυτών.

4.3.4 Αποθετήρια Δεδομένων

Όπως έχει αναλυθεί στην ενότητα 4.1, όλες οι δοσοληψίες δεδομένων της εφαρμογής με τη βάση δεδομένων πραγματοποιούνται μέσω κλάσεων Αποθετηρίων Δεδομένων (Data Repository). Έχει επιλεχθεί να υπάρχει μια ξεχωριστή κλάση Repository για κάθε DAO. Οι κλάσεις αυτές διατηρούν αναφορές των αντίστοιχων DAO (υποενότητα 4.3.2) τις οποίες αποκτούν μέσω του Instance της βάσης δεδομένων (υποενότητα 4.3.3). Χρησιμοποιώντας αυτές τις αναφορές ορίζονται μέθοδοι οι οποίες εκτελούν τις μεθόδους ερωτημάτων των DAOs και αλληλεπιδρούν με τη βάση δεδομένων. Επιπλέον, εντός των κλάσεων Repository, ορίζονται δεδομένα με wrapper την κλάση LiveData. Το LiveData επιτρέπει να ενημερώνονται τα δεδομένα του Repository όταν πραγματοποιείται αλλαγή των αντίστοιχων LiveData

δεδομένων του DAO και μπορούν με τη σειρά τους να παρατηρούνται από LiveData των ViewModels (υποενότητα 4.3.5). Για την εκτέλεση των μεθόδων ερωτημάτων μέσω της αναφοράς κάποιου DAO έχει γίνει χρήση του ExecutorService. Το ExecutorService είναι μια διεπαφή στην Java που παρέχει έναν μηχανισμό για την εκτέλεση εργασιών ασύγχρονα. Υπάρχουν οι κλάσεις Repository:

1. BGRepository

Διατηρεί αναφορά του BGDao και ορίζει τα παρακάτω LiveData:

```
private final BGDao mBGDao;

private final LiveData<BGModel> lastBg;
LiveData<List<BGModel>> lastNHoursBG;
private final LiveData<List<BGModel>> allBG;

      public BGRepository(Context application) {
         CGMDatabase db = CGMDatabase.getDatabase(application);
         mBGDao = db.bgDao();
         lastBg = mBGDao.getLastBG();
         allBG = mBGDao.getAllBG();
         lastNHoursBG =
         getLastNHoursBG((sharedPreferences.getInt(CHART_SELECTED_INTERVAL,12));
      }
```

Ορίζει ExecutorService που διαχειρίζεται 4 νήματα και χρησιμοποιείται για την εκτέλεση λειτουργιών της βάσης δεδομένων.

```
private static final int NUMBER_OF_THREADS = 4;
private static final ExecutorService databaseWriteExecutor =
Executors.newFixedThreadPool(NUMBER_OF_THREADS);
```

Διαθέτει τις μεθόδους:

```
public void insert(BGModel bg)
public void update(BGModel bg)
public void delete(BGModel bg)
public ArrayList<BGModel> getBGByDateStatic(Instant queryDate)
public void getBGByDate(Instant queryDate)
public LiveData<List<BGModel>> getLastNHoursBG(int hours)
```

Η κάθε μία από τις οποίες εκτελεί την αντίστοιχη μέθοδο ερωτήματος του BGDao σύμφωνα με την παρακάτω αντιστοίχιση

Μέθοδος BGRepository		Μέθοδος BGDao
insert	→	*insert*
update	→	*update*
delete	→	*delete*

getBGByDate	►	*findBGByDate*

2. AlertsRepository

Διατηρεί αναφορά του AlertsDao και ορίζει τα παρακάτω LiveData:

```
private final AlertsDao alertsDao;
private final LiveData<List<AlertsModel>> allAlerts;

public AlertsRepository(Application application) {
 CGMDatabase db = CGMDatabase.getDatabase(application);
 alertsDao = db.alertsDao();
 allAlerts = alertsDao.getAllAlerts();
}
```

Ορίζει ExecutorService που διαχειρίζεται 4 νήματα και χρησιμοποιείται για την εκτέλεση λειτουργιών της βάσης δεδομένων.

```
private static final int NUMBER_OF_THREADS = 4;
private static final ExecutorService databaseWriteExecutor =
Executors.newFixedThreadPool(NUMBER_OF_THREADS);
```

Διαθέτει τις μεθόδους:

```
public void insert(AlertsModel alert)
public void update(AlertsModel alert)
public void delete(AlertsModel alert)
public void deleteAllAlerts()
public LiveData<List<AlertsModel>> getAllAlerts()
```

Η κάθε μία από τις οποίες εκτελεί την αντίστοιχη μέθοδο ερωτήματος του AlertsDao σύμφωνα με την παρακάτω αντιστοίχιση

Μέθοδος AlertsRepository		Μέθοδος AlertsDao
insert	►	*insert*
update	►	*update*
delete	►	*delete*
deleteAllAlerts	►	*deleteAllAlerts*
getAllAlerts	►	*getAllAlerts*

3. NotesRepository

Διατηρεί αναφορά του NotesDao και ορίζει τα παρακάτω LiveData:

```
private NotesDao notesDao;
```

```
private LiveData<List<NotesModel>> allNotes;

public NotesRepository(Application application) {
 CGMDatabase db = CGMDatabase.getDatabase(application);
 notesDao = db.notesDao();
 allNotes = notesDao.getAllNotes();
}
```

Ορίζει ExecutorService που διαχειρίζεται 4 νήματα και χρησιμοποιείται για την εκτέλεση λειτουργιών της βάσης δεδομένων.

```
private static final int NUMBER_OF_THREADS = 4;
   private static final ExecutorService databaseWriteExecutor =
Executors.newFixedThreadPool(NUMBER_OF_THREADS);
```

Διαθέτει τις μεθόδους:

```
public void insert(NotesModel note)
public void update(NotesModel note)
public void delete(NotesModel note)
public void deleteAllNotes()
public void getNotesByDate(String queryDate)
```

Η κάθε μία από τις οποίες εκτελεί την αντίστοιχη μέθοδο ερωτήματος του NotesDao σύμφωνα με την παρακάτω αντιστοίχιση

Μέθοδος NotesRepository		Μέθοδος NotesDao
insert	→	*insert*
update	→	*update*
delete	→	*delete*
deleteAllNotes	→	*deleteAllAlerts*
getNotesByDate	→	*findNoteByDate*

4. COBRecordRepository

Διατηρεί αναφορά του COBRecordDao και δεν έχει LiveData:

```
private final COBRecordDao cobRecordDao;
    public COBRecordRepository(Context context) {
 CGMDatabase db = CGMDatabase.getDatabase(context);
 cobRecordDao = db.cobRecordDao();
}
```

Ορίζει ExecutorService που διαχειρίζεται 4 νήματα και χρησιμοποιείται για την εκτέλεση λειτουργιών της βάσης δεδομένων.

```
private static final int NUMBER_OF_THREADS = 4;
   private static final ExecutorService databaseWriteExecutor =
Executors.newFixedThreadPool(NUMBER_OF_THREADS);
```

Διαθέτει τις μεθόδους:

```
public void insert(COBRecord record)
public ArrayList<COBRecord> getRecordsFromTime(float dca)
public void deleteOldRecords(long thresholdTime)
```

Η κάθε μία από τις οποίες εκτελεί την αντίστοιχη μέθοδο ερωτήματος του COBRecordDao σύμφωνα με την παρακάτω αντιστοίχιση

Μέθοδος COBRecordRepository		Μέθοδος COBRecordDao
insert	→	*insert*
getRecordsFromTime	→	*getRecordsFromTime*
deleteOldRecords	→	*deleteOldRecords*

5. EventsRepoitory

Διατηρεί αναφορά του EventsDao και δεν έχει LiveData:

```
private final EventsDao eventsDao;
public EventsRepoitory(Context application) {CGMDatabase db =
CGMDatabase.getDatabase(application);
this.eventsDao = db.eventsDao();
}
```

Ορίζει ExecutorService που διαχειρίζεται 4 νήματα και χρησιμοποιείται για την εκτέλεση λειτουργιών της βάσης δεδομένων.

```
private static final int NUMBER_OF_THREADS = 4;

private static final ExecutorService databaseWriteExecutor =
Executors.newFixedThreadPool(NUMBER_OF_THREADS);
```

Διαθέτει τις μεθόδους:

```
public void insert(EventsModel event)
public void update(EventsModel event)
public void delete(EventsModel event)
public ArrayList<EventsModel> getEventByDateStatic(Instant queryDate)
public ArrayList<EventsModel> getBolusEventsWithinDia(float diaInHours)
```

Η κάθε μία από τις οποίες εκτελεί την αντίστοιχη μέθοδο ερωτήματος του EventsDao σύμφωνα με την παρακάτω αντιστοίχιση

Μέθοδος EventsRepoitory		Μέθοδος EventsDao
insert	→	*insert*
update	→	*update*
delete	→	*delete*
getEventByDate	→	*findEventByDate*
getBolusEventsWithinDia	→	*findBolusEventsWithinDia*

4.3.5 Μοντέλα προβολής

Οι κλάσεις μοντέλων προβολής (ViewModel) της εφαρμογής είναι υπεύθυνες για τη διαχείριση των δεδομένων που σχετίζονται με το UI. Όλες οι κλάσεις ViewModel που έχει η εφαρμογή κληρονομούν από την κλάση AndroidViewModel. Η κλάση αυτή είναι ένα αρχιτεκτονικό συστατικό (Architecture Component) της Android Jetpack, το οποίο έχει σχεδιαστεί για να διατηρεί και να διαχειρίζεται δεδομένα σχετιζόμενα με το UI, με έναν τρόπο που επιβιώνει από τις αλλαγές του κύκλου ζωής. Κάθε κλάση ViewModel καλύπτει τις ανάγκες του αντίστοιχου υποτμήματος (Fragment) (υποενότητα 4.3.7) και διατηρούν Instances όλων των Repository κλάσεων που χρειάζεται προκειμένου να αλληλεπιδράσουν με τα αντίστοιχα δεδομένα της βάσης και ορίζουν LiveData τα οποία ενημερώνονται από τα αντίστοιχα LiveData των Repository και είναι παρατηρήσιμα από τις κλάσεις των Fragments. Επιπλέον ορίζονται μέθοδοι οι οποίες αλληλεπιδρούν με τη βάση δεδομένων πάντα μέσω της αντίστοιχης κλάσης Repository. Η εφαρμογή έχει τα εξής ViewModels:

1. HomeViewModel

Διαχειρίζεται τα δεδομένα του HomeFragment της Αρχικής Οθόνης (Home). Μέσω του BGRepository παρέχει τα δεδομένα του διαγράμματος της αρχικής οθόνης (chartBG) και την πιο πρόσφατη τιμή BG (lastBG):

```
private final BGRepository bgRepository;
private final LiveData<BGModel> lastBG;
private LiveData<List<BGModel>> chartBG;

      public HomeViewModel(@NonNull Application application) {
         super(application);
         bgRepository = new BGRepository(application);
         lastBG = bgRepository.getLastBG();
         chartBG = bgRepository.getLastNHoursBG(savedInterval)
}
      public LiveData<List<BGModel>> getChartBG() {return chartBG;}
      public LiveData<BGModel> getLastBG() {return lastBG;}
```

2. CalendarViewModel

Διαχειρίζεται τα δεδομένα του CalendarFragment της Οθόνης Ημερολογίου (Calendar). Ορίζει μεθόδους οι οποίες μέσω των BGRepository και NotesRepository λαμβάνουν δεδομένα από τη βάση δεδομένων. Πιο συγκεκριμένα η *fetchBGByDate* καλεί την *getBGByDate* του BGRepository και η *fetchNotesByDate* καλεί την *getNotesByDate* του NotesRepository.

```
private BGRepository bgRepository;
private NotesRepository notesRepository;


        public CalendarViewModel(@NonNull Application application) {
        super(application);
        bgRepository = new BGRepository(application);
        notesRepository = new NotesRepository(application);
        }
    public void fetchBGByDate(Instant selectedDate) {
        bgRepository.getBGByDate(selectedDate);}
    public void fetchNotesByDate(String selectedDate) {
        notesRepository.getNotesByDate(selectedDate);}
```

3. BolusViewModel

Διαχειρίζεται τα δεδομένα του BolusFragment της Οθόνης Υπολογισμού έκτακτων Δόσεων Ινσουλίνης (Bolus). Μέσω των SharedPreferences (υποενότητα 4.3.10), EventsRepoitory, COBRecordRepository και BGRepository, ανακτά όλα τα απαραίτητα δεδομένα για να πραγματοποιηθεί ο υπολογισμός bolus που είναι διαθέσιμος μέσω της οθόνης Bolus και πραγματοποιεί τον υπολογισμό παρέχοντας στο UI την τιμή του αποτελέσματος. Η τιμή BG λαμβάνεται μέσω LiveData.

```
private final MutableLiveData<Float> insulin = new MutableLiveData<>();
private final MutableLiveData<Float> carbs = new MutableLiveData<>();
private final LiveData<BGModel> lastBG;
private EventsRepoitory eventsRepoitory;
private COBRecordRepository cobRecordRepository;
private BGRepository bgRepository;
private final SharedPreferences sharedPreferences;

    public MetricsViewModel(@NonNull Application application) {
        super(application);
        sharedPreferences = application.getSharedPreferences(SHARED_PREFS,
        Context.MODE_PRIVATE);
        bgRepository = new BGRepository(application);
        lastBG = bgRepository.getLastBG();
        eventsRepoitory = new EventsRepoitory(application);
        cobRecordRepository = new COBRecordRepository(application);
            }
private float calculateCOB()
private float calculateIOB()
private float getTargetBGValue()
private int getCurrentISFValue()
private int getCurrentICRValue()
public void calculateInsulin(final int bg)
```

Η calculateCOB υπολογίζει πόσα γραμμαρια ενεργών υδατανθράκων υπάρχουν στον οργανισμό του χρήστη, προκειμένου να το πετύχει αυτό καλεί τη μέθοδο *getRecordsFromTime* του CobRecordRepository καθώς και τιμές SharedPreferences

Η *calculateIOB* ανακτά τιμές SharedPreferences και υπολογίζει την ποσότητα ενεργής ινσουλίνης από προηγούμενα Bolus.

Η *getTargetBGV*alue, ανακτά το εύρος των τιμών γλυκόζης στόχου από τα SharedPreferences και υπολογίζει την τιμή στόχου ως τη μέση τιμή του εύρους τιμών.

Η getCurrentISFVAlue, ανακτά τις αποθηκευμένες στα SharedPreferences ISF και βρίσκει ποια από αυτές αντιστοιχεί στην ώρα της ημέρας κατά την οποία πραγματοποιείται ο υπολογισμός.

Η getCurrentICRVAlue, ανακτά τις αποθηκευμένες στα SharedPreferences ICR και βρίσκει ποια από αυτές αντιστοιχεί στην ώρα της ημέρας κατά την οποία πραγματοποιείται ο υπολογισμός

Η calculateInsulin, πραγματοποιεί τον υπολογισμό και παρέχει στο BolusFragment το αποτέλεσμα. Μετά τον υπολογισμό καλούνται οι insert μέθοδοι των CobRecordRepository και EventsRepoitory προκειμένου να αποθηκευτούν τα δεδομένα συμβάντος TYPE_BOLUS αλλά και η ποσότητα υδατανθράκων που εισήλθε στον οργανισμό.

4. AlertsViewModel

Διαχειρίζεται τα δεδομένα του AlertsFragment της Οθόνης Διαχείρισης Συναγερμών (Alerts). Μέσω του AlertsRepository παρέχει τα δεδομένα όλων των συναγερμών (allAlerts) της λίστας ενεργών συναγερμών:

```
private final AlertsRepository alertsRepository;
private final LiveData<List<AlertsModel>> allAlerts;

       public AlertsViewModel(@NonNull Application application) {
            super(application);
            alertsRepository = new AlertsRepository(application);
            allAlerts = alertsRepository.getAllAlerts();
            }
       LiveData<List<AlertsModel>> getAllAlerts(){return allAlerts;}
```

Ορίζει μεθόδους οι οποίες μέσω του AlertsRepository εισάγουν, ενημερώνουν και διαγράφουν εγγραφές στον πίνακα AlertsTable:

```
public void insert (AlertsModel alert) {alertsRepository.insert(alert);}
public void update(AlertsModel alert) {alertsRepository.update(alert);}
public void delete(AlertsModel alert) {alertsRepository.delete(alert);}
```

5. ReportsViewModel

Διαχειρίζεται τα δεδομένα του ReportsFragment της Οθόνης Δημιουργίας Αναφορών (Reports). Μέσω των SharedPreferences, EventsRepoitory και BGRepository, ανακτά όλα τα απαραίτητα δεδομένα και δημιουργεί τα αρχεία αναφορών που είναι διαθέσιμα μέσω της οθόνης Reports.

```
private BGRepository bgRepository;
private EventsRepoitory eventsRepoitory;

      public ReportsViewModel(@NonNull Application application) {
            super(application);
            bgRepository = new BGRepository(application);
            eventsRepoitory = new EventsRepoitory(application);
            }
private void drawPageBG(Canvas newCanvas, PdfDocument pdfDocument,
ArrayList<BGModel> temp, PdfDocument.Page myPage, Paint title, boolean isFirstPage,
String units)

public void generateBGPDF(Context context, Date startDate, Date endDate)
private void drawPageEvents(Canvas newCanvas, PdfDocument pdfDocument,
ArrayList<EventsModel> tempEvents, PdfDocument.Page myPage, Paint title, boolean
isFirstPage, String units)
      public void generateEventsPDF(Context context, Date fromDate, Date toDate)
```

Η *draw*PageBG, καλείται κάθε φορά που χρειάζεται να δημιουργηθεί στο αρχείο αναφοράς BG μια νέα σελίδα.

Η generateBGPDF, δημιουργεί μια νέα αναφορά τιμών BG, εδώ καλείται η getBGByDate του BGRepository.

Η drawPageEvents, καλείται κάθε φορά που χρειάζεται να δημιουργηθεί στο αρχείο αναφοράς Bolus & Alerts μια νέα σελίδα.

Η generateEventsPDF, δημιουργεί μια νέα αναφορά τιμών BG, εδώ καλείται η getEv*entByDate* του EventsRepository.

6. NotesViewModel

Διαχειρίζεται τα δεδομένα του NotesActivity της Οθόνης Προβολής Σημειώσεων (Notes). Μέσω του NotesRepository παρέχει τα δεδομένα όλων των σημειώσεων (allNotes) της λίστας αποθηκευμένων σημειώσεων:

```
private NotesRepository notesRepository;
private LiveData<List<NotesModel>> allNotes;

      public NotesViewModel(@NonNull Application application) {
            super(application);
            notesRepository = new NotesRepository(application);
            allNotes = notesRepository.getAllNotes();
      }
   public LiveData<List<NotesModel>> getAllNotes() {return allNotes;}
```

Ορίζει μεθόδους οι οποίες μέσω του NotesRepository εισάγουν, ενημερώνουν και διαγράφουν εγγραφές στον πίνακα NotesTable:

```
   public void insert(NotesModel note) {notesRepository.insert(note);}
public void update(NotesModel note) {notesRepository.update(note);}
public void delete(NotesModel note) {notesRepository.delete(note);}
```

4.3.6 Δραστηριότητες

Στο Android, οι Δραστηριότητες (Activities) είναι οι βασικές δομικές μονάδες του UI (User Interface) μιας εφαρμογής. Συνήθως, κάθε Activity αντιπροσωπεύει μια οθόνη με την οποία ο χρήστης μπορεί να αλληλεπιδράσει. Η εκτέλεση μιας εφαρμογής ξεκινά συνήθως από ένα main Activity, το οποίο μπορεί να εκκινεί άλλα Activities μέσω Intents. Κάθε Activity περνά από διαφορετικές καταστάσεις κύκλου ζωής (lifecycle), όπως onCreate(), onStart(), onResume(), onPause(), onStop(), και onDestroy(), οι οποίες καθορίζουν τη συμπεριφορά του Activity και την αλληλεπίδρασή του με τον χρήστη στα αντίστοιχα στάδια του κύκλου ζωής του. Κάθε Activity δηλώνεται στο AndroidManifest αρχείο με το ειδικό XML tag <activity>-</activity>. Εντός του tag προσδιορίζονται διάφορες παράμετροι του Activity όπως το όνομά, το θέμα (UI Theme) και άλλα. Επιπλέον σε αυτό το σημείο ορίζεται ποιο από τα Activities θα είναι το κύριο σημείο εισόδου της εφαρμογής, δηλαδή σηματοδοτεί ποιο Activity θα εκτελείται πρώτο, όταν ανοίγει η εφαρμογή. Κάθε μια από τις κλάσεις Activity της GMA κληρονομεί από την κλάση AppCompatActivity, η οποία είναι μια θεμελιώδης κλάση του Android SDK, και χρησιμοποιείται για τη διαχείριση της διεπαφής χρήστη μιας εφαρμογής. Επιπλέον κάθε κλάση Activity διαχειρίζεται κάποια διεπαφή χρήστη. Η διεπαφή χρήστη για όλα τα Activiteis και Fragments της εφαρμογής υπάρχει στο αντίστοιχο XML αρχείο διάταξης (Layout) (υποενότητα 4.3.8). Στην GMA υπάρχουν οι παρακάτω κλάσεις Activity:

1. RegisterActivity

Διαχειρίζεται τη διεπαφή χρήστη που αφορά την οθόνη Register (υποενότητα 3.2.1) και έχει αρχείο Layout το αρχείο activity_register.xml. Περιέχει μεταβλητές με βάση τις οποίες διαχειρίζεται τα στοιχεία διεπαφής χρήστη του activity_register layout και διατηρεί αναφορά σε Instance τύπου FirebaseAuth μέσω του οποίου εκτελείται η *createUserWithEmailAndPassword* η οποία επικοινωνεί με το Server ασύγχρονα και επιστρέφει το αποτέλεσμα.

```
EditText name,email,password,verifyPassword,phone;
Button RegisterButton;
TextView loginbtn, passwordFeedback;
FirebaseAuth fAuth;
ProgressBar pbar;


@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
            ⋮


   fAuth = FirebaseAuth.getInstance();
   setContentView(R.layout.activity_register);
            ⋮
   }
private boolean validatePassword()
private boolean passwordsMatch()
```

Μέσω των μεθόδων *validatePassword* και *passwordsMatch* πραγματοποιείται έλεγχος των συμβολοσειρών κωδικού που έχει εισάγει ο χρήστης στα πεδία εισαγωγής κωδικού και επαλήθευσης κωδικού, προκειμένου να ελεγχθεί εάν οι κωδικοί ταιριάζουν και εάν πληρούν όλα τα κριτήρια εγκυρότητας προτού αποσταλούν στο Server. Η Δραστηριότητα δηλώνεται στο AndroidManifest με παραμετροποιήσεις ώστε να επιτρέπεται μόνο ένα instance του Activity καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής και να εμφανίζεται πάντα σε κατακόρυφη (portrait) κατεύθυνση.

```
<activity
            android:name=".RegisterActivity"
            android:launchMode="singleTask"
            android:screenOrientation="portrait"
            android:theme="@style/Theme.Diplomatiki.NoActionBar"
            android:exported="true"/>
```

2. LoginActivity

Διαχειρίζεται τη διεπαφή χρήστη που αφορά την οθόνη Login (υποενότητα 3.2.2) και έχει αρχείο Layout το αρχείο activity_login.xml. Περιέχει μεταβλητές με βάση τις οποίες διαχειρίζεται τα στοιχεία διεπαφής χρήστη του activity_login layout και διατηρεί αναφορά σε Instance τύπου FirebaseAuth μέσω του οποίου εκτελούνται οι *signInWithEmailAndPassword* και *sendPasswordResetEmail* οι οποίες επικοινωνούν με το Server ασύγχρονα και επιστρέφουν το αποτέλεσμα. Η LoginActivity, είναι το προεπιλεγμένα κύριο σημείο εισόδου της εφαρμογής και κάθε φορά που η εφαρμογή ανοίγει εκτελείται το LoginActivity. Η πρώτη λειτουργία που εκτελείται καθώς "ανοίγει" το LoginActivity είναι ο έλεγχος για το αν έχει ήδη πραγματοποιηθεί ενεργή είσοδος του χρήστη στην εφαρμογή. Στην περίπτωση που έχει πραγματοποιηθεί είσοδος, η εφαρμογή εκκινεί την MainActivity χωρίς να εμφανιστεί το UI της LoginActivity η οποία και σταματά να εκτελείται.

```
   EditText email,password;
Button LoginButton;
TextView regbtn, tvForgotPassword ;
FirebaseAuth fAuth;
ProgressBar pbar;
SharedPreferences sharedPreferences;

@Override
protected void onCreate(@Nullable Bundle savedInstanceState) {
          ⋮
          setContentView(R.layout.activity_login);
          fAuth = FirebaseAuth.getInstance();
          sharedPreferences = getSharedPreferences(SHARED_PREFS,MODE_PRIVATE);
          ⋮
          if (fAuth.getCurrentUser()!=null){
            SharedPreferences.Editor editor = sharedPreferences.edit();
                editor.putString(EMAIL,fAuth.getCurrentUser().getEmail());
                editor.apply();
                startActivity(new
```

```
                Intent(getApplicationContext(),MainActivity.class));
                finish();
                }
        ⋮
}
private void sendPasswordResetEmail(String email)
```

Η Δραστηριότητα δηλώνεται στο AndroidManifest με παραμετροποιήσεις ώστε να είναι το κύριο σημείο εισόδου της εφαρμογής, να επιτρέπεται μόνο ένα instance του Activity καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής και να εμφανίζεται πάντα σε κατακόρυφη (portrait) κατεύθυνση.

```
<activity
     android:name=".LoginActivity"
     android:label="Login"
     android:launchMode="singleTask"
     android:screenOrientation="portrait"
     android:theme="@style/Theme.Diplomatiki.NoActionBar"
     android:exported="true">
     <intent-filter
       <action android:name="android.intent.action.MAIN" />
       <category android:name="android.intent.category.LAUNCHER" />
      </intent-filter>
</activity>
```

3. MainActivity

Διαχειρίζεται τη διεπαφή χρήστη που αφορά το κεντρικό μενού πλοήγησης(drawer menu) της εφαρμογής το οποίο διαχειρίζεται τις Οθόνες της εφαρμογής τις οποίες διαχειρίζονται τα Fragments (υποενότητα 4.3.7). Έχει αρχείο Layout το αρχείο activity_main.xml. Περιέχει μεταβλητές με βάση τις οποίες διαχειρίζεται το στοιχεία διεπαφής χρήστη του activity_ main layout δηλαδή των drawer menu, overflow button και του πλωτού κουμπιού τα οποία παρουσιάζονται στην υποενότητα 3.2.3. Επιπλέον μέσω του MainActivity πραγματοποιείται έλεγχος για το αν η εφαρμογή έχει τα απαραίτητα Special Permissions που αναφέρονται στην ενότητα 4.4. Σε περίπτωση που κάποια από τα Permissions δεν έχουν εγκριθεί, η MainActivity, ενημερώνει και οδηγεί το χρήστη στις ρυθμίσεις ώστε να παραχωρήσει τις απαιτούμενες άδειες. Μια ακόμη βασική ενέργεια που πραγματοποιείται κατά την εκτέλεση της lifecycle μεθόδου *onStart,* είναι η πραγματοποίηση προσκόλλησης (bind) στην υπηρεσία GlucoseMonitorService (υποενότητα 4.3.11) και αντίστοιχα αποκόλλησης (unbind) από το Service. Η προσκόλληση πραγματοποιείται μέσω της μεθόδου *bindService*, η οποία πραγματοποιεί απευθείας την σύνδεση σε περίπτωση που το GlucoseMonitorService είναι ενεργό είτε δημιουργεί ένα νέο instance του Service και πραγματοποιεί προσκόλληση σε αυτό. Η μέθοδος *onStart* εκτελείται τη στιγμή που ξεκινά να προβάλλεται το layout του Activity, ενώ η onStop όταν σταματά να προβάλλεται το layout του Activity.

```
GlucoseMonitorService glucoseMonitorService;
FloatingActionButton fab;
boolean isBound = false;

@Override
protected void onCreate(Bundle savedInstanceState) {
```

```
        ⋮
setContentView(R.layout.activity_main);
        ⋮
if(!checkPermissions()){
        requestPermissions();
}
        ⋮
}
@Override
protected void onStart() {
        super.onStart();
        Intent intent = new Intent(this, GlucoseMonitorService.class);
           bindService(intent, serviceConnection, Context.BIND_AUTO_CREATE);
}
@Override
protected void onStop() {
        super.onStop();
           if (isBound) {
            unbindService(serviceConnection);
               isBound = false;
           }
   }
```

Η Δραστηριότητα δηλώνεται στο AndroidManifest με παραμετροποιήσεις ώστε να είναι το κύριο σημείο εισόδου της εφαρμογής, να επιτρέπεται μόνο ένα instance του Activity καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής και να εμφανίζεται πάντα σε κατακόρυφη (portrait) κατεύθυνση.

```
      <activity
           android:name=".MainActivity"
           android:label="Home"
           android:noHistory="true"
           android:screenOrientation="portrait"
           android:theme="@style/Theme.Diplomatiki.NoActionBar" />
```

4. NotesActivity

Διαχειρίζεται τη διεπαφή χρήστη που αφορά την οθόνη Notes (υποενότητα 3.2.10) και έχει αρχείο Layout το αρχειό activity_notes.xml. Περιέχει μεταβλητές με βάση τις οποίες διαχειρίζεται τα στοιχεία διεπαφής χρήστη του activity_notes layout και αναφορά της κλάσης NotesViewModel μέσω της οποίας αντλεί τα απαραίτητα δεδομένα σημειώσεων μέσω LiveData.

```
private NotesViewModel notesViewModel;
private ArrayAdapter<NotesModel> adapter;
private TextView notesTextView;
@Override
protected void onCreate(Bundle savedInstanceState) {
        ⋮
        setContentView(R.layout.activity_notes);
        setContentView(R.layout.activity_notes);
```

```
        notesViewModel.getAllNotes().observe( . . .);
        ⋮
    }
@Override
public void onBackPressed()
```

Όταν ο χρήστης πατήσει το κουμπί Back των βασικών κουμπιών πλοήγησης των συσκευών Android τότε εκτελείται η lifecycle μέθοδος *onBackPressed*, μέσω της οποίας σταματά ο κύκλος ζωής της NotesActivity και η εφαρμογή εκκινεί την MainActivity.Η Δραστηριότητα δηλώνεται στο AndroidManifest με παραμετροποιήσεις ώστε να επιτρέπεται μόνο ένα instance του Activity καθ' όλη τη διάρκεια εκτέλεσης της εφαρμογής και να εμφανίζεται πάντα σε κατακόρυφη (portrait) κατεύθυνση.

```
<activity
        android:name=".ui.notes.NotesActivity"
        android:label="Notes"
        android:noHistory="true"
        android:screenOrientation="portrait"
        android:theme="@style/Theme.Diplomatiki.NoActionBar" />
```

4.3.7 Υποτμήματα

Ορισμένες από τις οθόνες της GMA, προβάλλονται μέσω του drawer menu της εφαρμογής. Η διαχείριση αυτών των οθονών πραγματοποιείται με τη χρήση κλάσεων Υποτμημάτων (Fragment classes). Τα Fragments στο Android είναι υποτμήματα μιας δραστηριότητας (Activity), τα οποία επιτρέπουν την οργάνωση της διεπαφής χρήστη (UI) και της λογικής της εφαρμογής σε μικρότερα, πιο διαχειρίσιμα μέρη. Είναι ουσιαστικά "μικρές δραστηριότητες" που μπορούν να ενσωματωθούν σε μία δραστηριότητα (Activity) και να έχουν την δική τους UI, αλλά και τον δικό τους κύκλο ζωής (lifecycle). Όλα τα Fragments της GMA ενσωματώνονται στην MainActivity και το καθ' ένα από αυτά διαχειρίζεται τη διεπαφή χρήστη των οθονών που αντιστοιχούν στις επιλογές του drawer menu. Όλες οι δοσοληψίες δεδομένων με τη βάση δεδομένων, που είναι απαραίτητες σε κάθε κλάση Fragment πραγματοποιούνται μέσω των αντίστοιχων ViewModel (υποενότητα 4.3.5), τα οποία εμπεριέχουν και όλα τα παρατηρήσιμα δεδομένα LiveData που είναι απαραίτητο να ενημερώνονται αυτόματα κατά τη διάρκεια του κύκλου ζωής του Fragment. Όλες οι κλάσεις Fragment της εφαρμογής, κληρονομούν από την κλάση Fragment, η οποία είναι μέρος του component UI του Android και αποτελεί βασική κλάση για τη δημιουργία κλάσεων Fragment. Κάθε Fragment της εφαρμογής εμπεριέχει όλο τον κώδικα που απαιτείται για τη διαχείριση κάποιου layout διεπαφής χρήστη (υποενότητα 4.3.8). Ο κώδικας αυτός αφορά τη συμπεριφορά κάθε στοιχείου διεπαφής χρήστη που συμπεριλαμβάνει το layout, καθώς επίσης και όλες τις μεθόδους παραμετροποίησης που το κάθε στοιχείο UI περιλαμβάνει (π.χ μέθοδοι onClick κ.τ.λ.). Η GMA έχει τις εξής κλάσεις Fragment:

1. HomeFragment

Διαχειρίζεται το layout που ορίζεται στο fragment_home, της Αρχικής Οθόνης (υποενότητα 3.2.4) και λαμβάνει δεδομένα μέσω του HomeViewModel. Είναι η πρώτη οθόνη που προβάλλεται μέσω του στοιχείου NavigationView (μενού πλοήγησης). Παρακολουθεί τα LiveData, chartBG και lastBG του HomeViewModel.

```
   HomeViewModel homeViewModel;
@Override
public void onCreate(@Nullable Bundle savedInstanceState) {

         ⋮
         homeViewModel = new ViewModelProvider(this).get(HomeViewModel.class);
            homeViewModel.getLastBG().observe(this,bgModel -> {
                        if(bgModel!=null)
                  changeText(String.valueOf(bgModel.getValue()));
      });
            homeViewModel.getChartBG().observe(this, bgModels -> {
            if (bgModels != null) {
                  loadChartData(bgModels);
                  }
      });
         ⋮
   }

public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
         ⋮
            View root = inflater.inflate(R.layout.fragment_home, container, false);
         ⋮
      }
```

2. CalendarFragment

Διαχειρίζεται το layout που ορίζεται στο fragment_calendar, της Οθόνης Ημερολογίου και λαμβάνει δεδομένα μέσω του CalendarViewModel.

```
CalendarViewModel calendarViewModel;
⋮
public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
         ⋮
            View root = inflater.inflate(R.layout.fragment_calendar, container,
            false);
            calendarViewModel = new
            ViewModelProvider(this).get(CalendarViewModel.class);
         ⋮
   }
private void getDayDataFromDb(Instant selectedDate)
private void getDayNotesFromDb(String selectedDate)
```

Μέσω των *getDayDataFromDb* και *getDayNotesFromDb*, εκτελούνται οι μέθοδοι του *fetchBGByDate* και *fetchNotesByDate* του CalendarViewModel αντίστοιχα.

3. BolusFragment

Διαχειρίζεται το layout που ορίζεται στο αρχείο fragment_bolus, της οθόνης Bolus και λαμβάνει δεδομένα μέσω του BolusViewModel. Παρακολουθεί τα δεδομένα LiveData carbs, insulin και lastBG του BolusViewModel.

```
private BolusViewModel bolusViewModel;
private BGModel bgModel;
        ⋮
public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup container,
Bundle savedInstanceState) {
        ⋮
        View root = inflater.inflate(R.layout.fragment_metrics, container, false);
        ⋮
           bolusViewModel = new ViewModelProvider(this).get(BolusViewModel.class);
           bolusViewModel.getCarbs().observe(getViewLifecycleOwner(),carbs ->
           CarbsTextLayout.getEditText().setText(String.valueOf(carbs)));
           bolusViewModel.getInsulin().observe(getViewLifecycleOwner(), insulin ->
           insulinTextView.setText("Bolus Intake: " + insulin +" "+getUnits()));
           bolusViewModel.getLastBG().observe(getViewLifecycleOwner(),bgModel -> {
           if(bgModel!=null)
               this.bgModel = bgModel;
           });
        ⋮
   }
```

4. AlertsFragment

Διαχειρίζεται το layout που ορίζεται στο αρχείο fragment_alerts, της οθόνης Alerts και λαμβάνει δεδομένα μέσω του AlertsViewModel. Παρακολουθεί τα δεδομένα LiveData allAlerts του AlertsViewModel.

```
AlertsViewModel alertsViewModel;
ArrayAdapter arrayAdapter;
public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
        View root = inflater.inflate(R.layout.fragment_alerts, container, false);
           alertsViewModel.getAllAlerts().observe(getViewLifecycleOwner(), alerts
           -> {
               if(!alerts.isEmpty()) {
                   for (AlertsModel alertsModel : alerts){
                      if(!alertsModel.getOperand().contains("BG:"))
                         alertsModel.setOperand("BG: "+ alertsModel.getOperand());
                   }
                   checkAlerts(alerts);
                   arrayAdapter.clear();
                   arrayAdapter.addAll(alerts);
               }
           });
     }
```

5. ReportsFragment

Διαχειρίζεται το layout που ορίζεται στο αρχείο fragment_reports, της οθόνης Reports και λαμβάνει δεδομένα μέσω του ReportsViewModel, μέσω του οποίου δημιουργείται και το αρχείο αναφορών σε περίπτωση που ο χρήστης το επιλέξει.

```
private ReportsViewModel reportsViewModel;

public View onCreateView(@NonNull LayoutInflater inflater,
ViewGroup container, Bundle savedInstanceState) {
            reportsViewModel = new
            ViewModelProvider(this).get(ReportsViewModel.class);
View root = inflater.inflate(R.layout.fragment_reports, container, false);
      }
```

Σε περίπτωση που ο χρήστης επιλέξει τη δημιουργία κάποιου αρχείου αναφοράς από το UI τότε καλούνται αντίστοιχα οι *generateBGPDF* και *generateEventsPDF* τουReportsViewModel.

6. SettingsFragment

Διαχειρίζεται το layout που ορίζεται στο αρχείο fragment_settings, της οθόνης Settings. Δεν χρειάζεται να αλληλεπιδράσει με τη βάση δεδομένων καθώς όλα τα δεδομένα που χρειάζεται αποθηκεύονται στο αρχείο κοινών προτιμήσεων.

4.3.8 Διεπαφή Χρήστη

Σε αυτή την υποενότητα, θα αναλυθούν τα αρχεία layout που ορίζουν τη μορφή των διάφορων οθονών της εφαρμογής. Τα αρχεία αυτά είναι τύπου XML, ονομάζονται αρχεία πόρων διάταξης (layout resource files) και αποθηκεύονται στον φάκελο res/layout της εφαρμογής. Είναι γραμμένα με χρήση της γλώσσας σήμανσης XML και ορίζουν τη δομή και τον σχεδιασμό στοιχείων διεπαφής χρήστη. Όλα τα αρχεία layout ξεκινούν με ένα θεμελιώδες στοιχείο (root element), το οποίο είναι ένας τύπος ViewGroup που περιέχει όλα τα υπόλοιπα στοιχεία της διάταξης. Ο τύπος του θεμελιώδους στοιχείου καθορίζει το σχήμα και κάποια βασικά χαρακτηριστικά της διάταξης. Στη συνέχεια και εντός του tag του θεμελιώδους στοιχείου προστίθενται όλα τα στοιχεία διεπαφής χρήστη που περιλαμβάνει η διάταξη καθώς επίσης και οι σχέσεις και περιορισμοί (Constraints) του εκάστοτε στοιχείου με όλα τα υπόλοιπα. Για κάθε κλάση Δραστηριότητας ή Υποτμήματος, υπάρχει και το αντίστοιχο layout αρχείο. Τα αρχεία layout της εφαρμογής είναι:

1. activity_register

Ορίζει το UI της Οθόνης Εγγραφής, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.1

2. activity_login

Ορίζει το UI της Οθόνης Εισόδου Χρήστη, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.2

3. activity_main

Είναι το αρχείο xml της MainActivity, διαφέρει από όλα τα υπόλοιπα αρχεία layout καθώς δεν ορίζει το UI κάποιας μεμονωμένης οθόνης αλλά ορίζει το βασικό μενού πλοήγησης της εφαρμογής. Έχει ως θεμελιώδες στοιχείο ένα DrawerLayout, το οποίο είναι ένα ειδικό ViewGroup που επιτρέπει την προσθήκη “συρταριού” πλοήγησης (navigation drawer). Εντός του DrawerLayout ορίζεται ένα στοιχείο τύπου NavigationView μέσω του οποίου ορίζονται οι επιλογές του μενού και άλλες παράμετροι αυτού

```
  <androidx.drawerlayout.widget.DrawerLayout
xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:app="http://schemas.android.com/apk/res-auto"
xmlns:tools="http://schemas.android.com/tools"
android:id="@+id/drawer_layout"
android:layout_width="match_parent"
android:layout_height="match_parent"
android:fitsSystemWindows="true"
tools:openDrawer="start">

    <include
    layout="@layout/app_bar_main"
    android:layout_width="match_parent"
    android:layout_height="match_parent" />


        <com.google.android.material.navigation.NavigationView
            android:id="@+id/nav_view"
            android:layout_width="wrap_content"
            android:layout_height="match_parent"
            android:layout_gravity="start"
            android:fitsSystemWindows="true"
            app:headerLayout="@layout/nav_header_main"
            app:menu="@menu/activity_main_drawer" />
</androidx.drawerlayout.widget.DrawerLayout>
```

Το include tag, χρησιμοποιείται για να προστεθούν στο DrawerLayout τα στοιχεία που βρίσκονται στο αρχείο app_bar_main. Σε αυτό το αρχείο βρίσκονται το πλωτό κουμπί καθώς επίσης και η επικεφαλίδα με το overflow button. Μέσω του στοιχείου NavigationView καθορίζονται οι παράμετροι του στοιχείου. Η παράμετρος headerLayout περιέχει τα στοιχεία UI που εμφανίζονται στο επάνω μέρος του menu και είναι συγκεκριμένα:

```
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
  xmlns:app="http://schemas.android.com/apk/res-auto"
  android:layout_width="match_parent"
  android:layout_height="@dimen/nav_header_height"
```

```
    android:background="@drawable/side_nav_bar"
    android:gravity="bottom"
    android:orientation="vertical"
    android:paddingLeft="@dimen/activity_horizontal_margin"
    android:paddingTop="@dimen/activity_vertical_margin"
    android:paddingRight="@dimen/activity_horizontal_margin"
    android:paddingBottom="@dimen/activity_vertical_margin"
    android:theme="@style/ThemeOverlay.AppCompat.Dark">

        <ImageView
            android:id="@+id/imageView"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:contentDescription="@string/nav_header_desc"
            android:paddingTop="@dimen/nav_header_vertical_spacing"
            app:srcCompat="@mipmap/ic_user_round" />

        <TextView
            android:layout_width="match_parent"
            android:layout_height="wrap_content"
            android:paddingTop="@dimen/nav_header_vertical_spacing"
            android:text="@string/nav_header_title"
            android:textAppearance="@style/TextAppearance.AppCompat.Body1" />

        <TextView
            android:id="@+id/textViewUserName"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="@string/nav_header_subtitle" />
</LinearLayout>
```

Η παράμετρος menu καθορίζει το αρχείο που εμπεριέχει τις επιλογές του menu. Το αρχείο menu του NavigationView περιέχει ένα group με όλες τις επιλογές του μενού και συγκεκριμένα:

```
<menu xmlns:android="http://schemas.android.com/apk/res/android"
xmlns:tools="http://schemas.android.com/tools"
tools:showIn="navigation_view">

        <group android:checkableBehavior="single">
            <item
                android:id="@+id/nav_home"
                android:icon="@drawable/ic_baseline_home_24"
                android:title="@string/menu_home" />
            <item
                android:id="@+id/nav_calendar"
                android:icon="@drawable/ic_baseline_calendar_today_24"
                android:title="@string/menu_calendar" />
            <item
                android:id="@+id/nav_metrics"
                android:icon="@drawable/ic_baseline_science_24"
                android:title="@string/menu_metrics" />
            <item
```

```
            android:id="@+id/nav_alerts"
            android:icon="@drawable/ic_baseline_add_alert_24"
            android:title="@string/menu_alerts" />
        <item
            android:id="@+id/nav_reports"
            android:icon="@drawable/ic_baseline_folder_shared_24"
            android:title="@string/menu_reports" />
        <item
            android:id="@+id/nav_settings"
            android:icon="@drawable/baseline_settings_24"
            android:title="@string/menu_settings" />

    </group>
</menu>
```

4. activity_notes

Ορίζει το UI της Οθόνης Προβολής Σημειώσεων, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.10

5. fragment_home

Ορίζει το UI της Αρχικής Οθόνης, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.4. Το στοιχείο διεπαφής χρήστη που αφορά το γράφημα είναι τύπου mikephil.charting.charts.LineChart και εισάγεται στην εφαρμογή μέσω της βιβλιοθήκης MPAndroidChart (υποενότητα 4.2.2).

6. fragment_calendar

Ορίζει το UI της Οθόνης Ημερολογίου, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.5

7. fragment_bolus

Ορίζει το UI της Οθόνης Υπολογισμού Έκτακτων Δόσεων Ινσουλίνης, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.6

8. fragment_alerts

Ορίζει το UI της Οθόνης Διαχείρισης Συναερμών, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.7

9. fragment_reports

Ορίζει το UI της Οθόνης Δημιουργίας Αναφορών, έχει θεμελιώδες στοιχείο τύπου ConstraintLayout το οποίο περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.8

10. fragment_settings

Ορίζει το UI της Οθόνης Ρυθμίσεων, έχει θεμελιώδες στοιχείο τύπου NestedScrollView καθώς θέλουμε οι διάφορες ρυθμίσεις να είναι στοιχισμένες κάθετα. Το NestedScrollView, περιέχει όλα τα στοιχεία διεπαφής χρήστη που περιγράφονται στην υποενότητα 3.2.9

4.3.9 Βοηθητικές κλάσεις

Κάποιες λειτουργίες εντός της εφαρμογής είναι κοινές και επαναλαμβάνονται. Οι λειτουργίες αυτές έχουν υλοποιηθεί σε βοηθητικές κλάσεις, στις οποίες πραγματοποιείται διαχωρισμένη και οργανωμένη υλοποίηση κοινών λειτουργιών. Προκειμένου να μην επαναλαμβάνεται ο ίδιος κώδικας σε διαφορετικά σημεία. Σε αυτή την υποενότητα, θα παρουσιαστούν οι βασικές βοηθητικές κλάσεις που έχουν υλοποιηθεί στην εφαρμογή, εξηγώντας τον ρόλο και τη λειτουργικότητά τους:

1. BGUnitTransformations

Η βοηθητική κλάση BGUnitTransformations, περιέχει μεθόδους οι οποίες πραγματοποιούν μετατροπή των τιμών γλυκόζης ανάλογα με τη προτίμηση του χρήστη. Πιο συγκεκριμένα, στην GMA οι τιμές γλυκόζης μπορούν να προβάλλονται είτε σε mg/dL είτε σε mmol/L και αποθηκεύονται πάντα σε mg/dL. Σε επίπεδο πηγαίου κώδικα θεωρείται ότι οι μονάδες md/dL είναι οι πρωτεύουσες μονάδες προβολής (Primary BG Units) ενώ οι mmol/L είναι οι δευτερεύουσες (Secondary BG Units). Η μετατροπή από mg/dL σε mmol/L πραγματοποιείται με την πράξη:

$$mmol/L = \frac{mg/dL}{18.18}$$

Στην BGUnitTransformations ορίζονται δύο μέθοδοι οι οποίες πραγματοποιούν τις μετατροπές από mg/dL σε mmol/L και το αντίστροφο.Η μέθοδος *primaryToSecondary* δέχεται ως όρισμα την τιμή BG σε mg/dL και επιστρέφει την τιμή σε mmol/L ενώ το αντίστροφο συμβαίνει με την *secondaryToPrimary.*

```
public static double primaryToSecondary(int bg)
public static int secondaryToPrimary(double bg)
```

2. Converters

Όπως είχε αναφερθεί και στην υποενότητα 4.3.3 η βάση δεδομένων υποστηρίζει μόνο απλούς τύπους δεδομένων όπως int, String, boolean και long. Όταν θέλουμε να αποθηκεύσουμε πιο σύνθετους τύπους δεδομένων, όπως Instant (που χρησιμοποιείται για την αναπαράσταση χρονικών στιγμών), χρειαζόμαστε TypeConverters. Οι TypeConverters βοηθούν στη μετατροπή τέτοιων δεδομένων πριν την αποθήκευση και μετά την ανάκτηση από τη βάση. Όταν μια βάση δεδομένων Room χρειάζεται τέτοιεςς μετατροπές τότε δηλώνεται η κλάση που τις περιέχει σαν παράμετρος του annotation @TypeConverter. Στην προκειμένη περίπτωση, σε κάποιους από τους πίνακες η αποθήκευση χρονικών στιγμών γίνεται με την αποθήκευση αντικειμένων τύπου Instant. Η κλάση Converters, δηλώνεται στο @Typeconverter annotation της βάσης δεδομένων και περιέχει τις μεθόδους μετατροπής *instantFromTimestamp* και *instantToTimestamp.* Η πρώτη μετατρέπει τιμές long σε Instant ενώ η δεύτερη το αντίστροφο. Οι μέθοδοι επισημειώνονται με @TypeConverter προκειμένου το Room να χρησιμοποιήσει αυτόματα τις μεθόδους για τις απαραίτητες μετατροπές.

```
@TypeConverter
public static Instant fromTimestamp(Long value)
@TypeConverter
public static Long instantToTimestamp(Instant instant)
```

3. DateTimeUtils

Η βοηθητική κλάση DateTimeUtils, περιέχει μεθόδους οι οποίες αφορούν τη διαχείριση ζώνης ώρας. Όλες οι χρονικές στιγμές που αποθηκεύονται στη βάση δεδομένων αποθηκεύονται σε ζώνη ώρας UTC. Η UTC (Coordinated Universal Time - Συντονισμένος Παγκόσμιος Χρόνος) είναι η πρωτεύουσα χρονική κλίμακα που χρησιμοποιείται διεθνώς ως σημείο αναφοράς για τον προσδιορισμό της ώρας σε όλο τον κόσμο. Ωστόσο η εφαρμογή χρειάζεται να προβάλλει τα δεδομένα στην τοπική ζώνη ώρας του χρήστη. Για το λόγο αυτό, η κλάση DateTimeUtils περιέχει τις μεθόδους*convertUTCToLocal*, *convertLocalToUTC* και *floatToLongHours.* Η *convertUTCToLocal* μετατρέπει την UTC ώρα (σε μορφή Instant) σε τοπική ζώνη ώρας (σε μορφή ZonedDateTime) ενώ η *convertLocalToUTC* πραγματοποιεί την αντίστροφη μετατροπή. Η *floatToLongHours* μετατρέπει μια χρονική διάρκεια από ώρες (σε δεκαδική μορφή) σε χιλιοστά του δευτερολέπτου (milliseconds).

```
public static ZonedDateTime convertUTCToLocal(Instant utcTime)
public static Instant convertLocalToUTC(ZonedDateTime localTime)
public static long floatToLongHours(float floatHours)
```

4.3.10 Κοινές προτιμήσεις

Οι Κοινές προτιμήσεις (Shared Preferences) αποτελούν έναν μηχανισμό αποθήκευσης δεδομένων σε εφαρμογές Android, επιτρέποντας τη διατήρηση μικρών συνόλων δεδομένων με τη μορφή ζευγών κλειδιού-τιμής (key-value). Χρησιμοποιούνται κυρίως για την αποθήκευση απλών ρυθμίσεων, προτιμήσεων του χρήστη και κατάστασης της εφαρμογής, εξασφαλίζοντας ότι αυτά τα δεδομένα θα είναι διαθέσιμα ακόμα και μετά το κλείσιμο και την επανεκκίνηση της εφαρμογής. Η αποθήκευση δεδομένων στις Shared Preferences πραγματοποιείται σε ένα αρχείο XML που είναι αποθηκευμένο στον εσωτερικό αποθηκευτικό χώρο της εφαρμογής. Το αρχείο αυτό είναι προσβάσιμο μόνο από την ίδια την εφαρμογή, διασφαλίζοντας έτσι την ασφάλεια των δεδομένων. Η αποθήκευση και επεξεργασία ζευγών κλειδιού-τιμής στο το αρχείο XML όπως και ανάκτηση τιμών από αυτό, γίνεται με χρήση της κλάσης SharedPreferences. Αρχικά μέσω της μεθόδου *getSharedPreferences* η οποία δέχεται ως ορίσματα το όνομα του αρχείου κοινών προτιμήσεων και την σταθερά προσβασιμότητας, επιστρέφει ένα instance της κλάσης SharedPreferences μέσω του οποίου στη συνέχεια μπορούν να αποθηκευτούν και ανακτηθούν δεδομένα στο αρχείο. Στα πλαίσια της GMA έχει επιλεχθεί, όλες οι Shared Preferences της εφαρμογής να αποθηκεύονται στο αρχείο με όνομα “SHARED_PREFS” και ως σταθερά προσβασιμότητας η “Context.MODE_PRIVATE“ η οποία ορίζει ότι το αρχείο θα είναι διαθέσιμο μόνο στην ίδια την εφαρμογή και δεν μπορεί να διαβαστεί ή να τροποποιηθεί από άλλες εφαρμογές.

```
SharedPreferences sharedPreferences =getApplication().getSharedPreferences
```

```
(SHARED_PREFS, Context.MODE_PRIVATE);
```

Στη συνέχεια μέσω μεθόδων τύπου get λαμβάνουμε την αποθηκευμένη τιμή με βάση το αντίστοιχο κλειδί. Στις συναρτήσεις τύπου get δίνονται σαν ορίσματα το κλειδί του ζεύγους και μια τιμή προεπιλογής που επιστρέφεται από τη μέθοδο σε περίπτωση που το αρχείο δεν περιέχει εγγραφή με το συγκεκριμένο κλειδί.

```
sharedPreferences.getString(DATA_SOURCE, "xDrip+");
```

Για την αποθήκευση ζεύγους χρησιμοποιείται το αντικείμενο Editor της SharedPreferences, μέσω του οποίου δίνεται πρόσβαση σε μεθόδους τύπου put που πραγματοποιούν την αποθήκευση. Σαν ορίσματα στις μεθόδους put δίνεται το κλειδί του ζεύγους και η επιθυμητή τιμή. Για την επεξεργασία τιμών που υπάρχουν ήδη στο αρχείο, πραγματοποιείται αποθήκευση της νέας τιμής με το ίδιο κλειδί.

```
SharedPreferences.Editor editor = sharedPreferences.edit();
editor.putString(preferenceKey, selectedValue);
editor.apply();
```

4.3.11 Υπηρεσίες Παρασκηνίου

Όπως έχει αναφερθεί και στην υποενότητα 3.3.1, η GMA συμπεριλαμβάνει δύο κλάσεις υπηρεσιών (Services). Συγκεκριμένα, τα services αυτά είναι το GlucoseMonitorService και το AlertService. Το πρώτο αποτελεί μέρος διάφορων λειτουργιών της εφαρμογής καθώς παρακολουθεί την διακύμανση των τιμών BG του χρήστη και τον ενημερώνει μέσω Persistent Notification, ελέγχει αν οι νέα τιμή γλυκόζης είναι εκτός ορίου κάποιου συναγερμού και σηματοδοτεί την εκκίνηση του AlertService όταν είναι απαραίτητο. Είναι σχεδιασμένο να “επιβιώνει” το κλείσιμο της εφαρμογής και να επανεκκινείται από το λειτουργικό σύστημα σε περίπτωση που σταματήσει να εκτελείται. Επιπλέον, εφόσον ο χρήστης και το λειτουργικό σύστημα το επιτρέπουν μπορεί να εκκινείται αυτόματα μετά το άνοιγμα της συσκευής του χρήστη. Το AlertService όταν εκκινείται ενεργοποιεί τη λειτουργία συναγερμού κατά την οποία ο χρήστης ειδοποιείται για την τιμή BG μέσω ήχου, δόνησης και ειδικής ένδειξης στην οθόνη της συσκευής του. Παρακάτω αναλύεται η βασική υλοποίηση κάθε κλάσης:

1. GlucoseMonitorService

Τα Services μιας εφαρμογής όπως και οι Δραστηριότητες, χρειάζεται να δηλώνονται μαζί με τις παραμέτρους λειτουργίας τους στο AndroidManifest αρχείο της εφαρμογής. Κατά τη δήλωση του GlucoseMonitorService προστίθενται οι παράμετροι stopWithTask="false" και foregroundServiceType="health". Η πρώτη παράμετρος καθορίζει αν η υπηρεσία θα σταματήσει αυτόματα όταν σταματήσει η δραστηριότητα που την έχει ξεκινήσει και έχει οριστεί ως false προκειμένου να μην σταματάει το service όταν σταματήσει να εκτελείται η εφαρμογή. Η δεύτερη παράμετρος χρησιμοποιείται για να καθορίσει τον τύπο της υπηρεσίας foreground,

δηλώνει ότι η υπηρεσία είναι τύπου υγείας και εκτελεί εργασίες που σχετίζονται με την παρακολούθηση ή καταγραφή δεδομένων υγείας.

```
<service
      android:name=".GlucoseMonitorService"
      android:permission="android.permission.FOREGROUND_SERVICE"
      android:enabled="true"
      android:stopWithTask="false"
      android:foregroundServiceType="health"/>
```

Η κλάση GlucoseMonitorService, κληρονομεί από την ForegroundService. Ένα Foreground Service είναι μια υπηρεσία που εκτελείται στο παρασκήνιο, αλλά εμφανίζει μια ειδοποίηση στον χρήστη για να τον ενημερώνει ότι η εφαρμογή ή η υπηρεσία είναι ενεργή. Αυτή η ειδοποίηση δίνει προτεραιότητα στη λειτουργία της υπηρεσίας και αποτρέπει το σύστημα από το να την κλείσει λόγω περιορισμών πόρων. Επιπλέον για να διασφαλιστεί ότι το service θα επανεκκινηθεί σε περίπτωση που τερματιστεί από το λειτουργικό η *onStartCommand* επιστρέφει START_STICKY, η οποία είναι μια σταθερά του Android και ζητά από το λειτουργικό σύστημα την επανεκκίνηση της υπηρεσίας σε περίπτωση που τερματιστεί από το ίδιο. Η μέθοδος *onStartCommand,* εκτελείται κάθε φορά που το service καλείται να εκκινήσει ανεξαρτήτως αν είναι ήδη ενεργό. Στην *onStartCommand* του GlucoseMonitorService αφού πραγματοποιηθεί έλεγχος για το αν το service είναι ήδη ενεργό, αρχικά δημιουργείται το Persistent Notification του service μέσω της μεθόδου *createInitialForegroundNotification*. Στη συνέχεια, γίνεται το register του BGDataReceiver ο οποίος λαμβάνει τιμές BG από την εφαρμογή xDrip+. Τέλος μέσω της μεθόδου *observeLastBG* παρακολουθούνται μέσω του BGRepository οι αλλαγές της τιμής γλυκόζης του χρήστη. Όταν παρατηρηθεί αλλαγή τιμής καλούνται οι *checkAlarm* και *updateNotificationWithBGValue.* Η *checkAlarm* ελέγχει αν θα πρέπει να ενεργοποιηθεί κάποιος συναγερμός και στέλνει το Intent το οποίο εκκινεί το AlarmService μέσω του AlarmReceiver. Η *updateNotificationWithBGValue*, ενημερώνει το Persistent Notification με την νέα τιμή γλυκόζης και τη χρονική στιγμή λήψης αυτής. Τέλος, η *setExactAlarm,* εφόσον έχει επιλεχθεί η παραγωγή εικονικών τιμών εκκινεί τη διαδικασία παραγωγής εικονικών τιμών στέλνοντας το κατάλληλο intent το οποίο διαχειρίζεται ο MockValueReceiver.

```
public class GlucoseMonitorService extends Service

@Override
public int onStartCommand(Intent intent, int flags, int startId) {
      if (!isServiceRunning) {
          isServiceRunning = true;
          createInitialForegroundNotification();

          BGDataReceiver bgDataReceiver = new BGDataReceiver();
          IntentFilter intentFilter = new IntentFilter();
          intentFilter.addAction("com.eveningoutpost.dexdrip.Intents.ACTION_N
          EW_BG_ESTIMATE");
          intentFilter.addAction("com.eveningoutpost.dexdrip.BgEstimate");
```

```
            if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
                registerReceiver(bgDataReceiver,intentFilter,Context.RECEIVER_N
                OT_EXPORTED);
            } else {
                registerReceiver(bgDataReceiver, intentFilter);
            }
            observeLastBG();
            setExactAlarm(this);
        }
        return START_STICKY;
    }

    private void observeLastBG() {
        bgRepository.getLastBG().observeForever(bgModel -> {
            if (bgModel != null) {
                ZonedDateTime localTime =
                DateTimeUtils.convertUTCToLocal(bgModel.getTime());
                BGModel adjustedModel = new
                BGModel(bgModel.getValue(),localTime.toInstant());
                checkAlarm(adjustedModel, getApplicationContext());
                updateNotificationWithBGValue(adjustedModel);
            }
        });
    }
```

2. AlertService

Το AlertService υλοποιεί την λειτουργία συναγερμού. Δηλώνεται στο AndroidManifest της εφαρμογής και κληρονομεί από την κλάση Service. Η Service είναι μια component class του Android που χρησιμοποιείται για την εκτέλεση λειτουργιών στο παρασκήνιο. Στην *onStartCommand* του AlertService πραγματοποιούνται τρεις ενέργειες. Αρχικά, μέσω του EventsRepository γίνεται εισαγωγή των δεδομένων του συμβάντος συναγερμού στη βάση δεδομένων, στη συνέχεια παραμετροποιεί και προβάλλει το γραφικό περιβάλλον της ειδοποίησης του συναγερμού (overlayView) θέτοντας την τιμή BG που προκάλεσε το συναγερμό αλλά και τη χρονική στιγμή λήψης. Τέλος ξεκινά την αναπαραγωγή του ηχητικού μηνύματος του συναγερμού και τη δόνηση της συσκευής η *onStartCommand* του AlertService επιστρέφει START_NOT_STICKY, η οποία είναι μια σταθερά του Android και ζητά από το λειτουργικό σύστημα να μην επανεκκινήσει την υπηρεσία σε περίπτωση που τερματιστεί από το ίδιο.

```
<service android:name=".AlertService" android:exported="false"/>

public class AlertService extends Service

@Override
public int onStartCommand(Intent intent, int flags, int startId) {

    ⋮
```

```
        EventsModel event = new EventsModel(TYPE_ALERT, msg,eventntInstant);
            eventsRepoitory.insert(event);

            TextView alertMessage = overlayView.findViewById(R.id.alertMessage);
            TextView alertValue = overlayView.findViewById(R.id.alertValue);
            ImageView alertIcon = overlayView.findViewById(R.id.alertIcon);

            if (message != null) {
                alertMessage.setText("Time of Measurement: " + message);
            }else{
                alertMessage.setText(R.string.overlay_alert_time);
            }

            alertValue.setText("Your glucose level is: " + intValue);
            alertIcon.setImageResource(R.drawable.ic_baseline_warning_24_red);

            overlayView.setOnClickListener(v →
                stopSelf());
            }
            vibrate()
        ⋮
            return START_NOT_STICKY;
    }
```

Όταν ο χρήστης κλείσει το συναγερμό εκτελείται η *onDestory* μέθοδος του AlertService η οποία κλείνει το παράθυρο της ειδοποίησης, σταματάει τον ήχο και τη δόνηση και ανοίγει το MainActivity της εφαρμογής

```
    @Override
    public void onDestroy() {
            super.onDestroy();
            if (overlayView != null) {
                windowManager.removeView(overlayView);
            }
            stopVibrationAndSound();
            Intent alertIntent = new
            Intent(getApplicationContext(),MainActivity.class);
            alertIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK |
            Intent.FLAG_ACTIVITY_CLEAR_TASK);
            getApplicationContext().startActivity(alertIntent);
    }
```

4.3.12 Κώδικας δεκτών συμβάντων

Σε αυτή την υποενότητα θα αναλυθούν όλες οι κλάσεις BroadcastReceiver που ορίζονται από την εφαρμογή. Οι BroadcastReceivers είναι ένας μηχανισμός στο Android που επιτρέπει στην εφαρμογή να λαμβάνει και διαχειρίζεται γεγονότα του συστήματος ή προσαρμοσμένα broadcasts από την ίδια ή άλλες εφαρμογές. Είναι χρήσιμοι για την επικοινωνία μεταξύ των components μιας εφαρμογής και την εκτέλεση συγκεκριμένων ενεργειών όταν συμβαίνουν προκαθορισμένα γεγονότα. Όλες οι κλάσεις Δεκτών Συμβάντων της εφαρμογής κληρονομούν από την κλάση BroadcastReceiver όπου είναι η βασική κλάση για την λήψη μηνυμάτων Intent.

Η λειτουργία των BroadcastReceivers περιλαμβάνει τρία στάδια. Αρχικά ο Receiver θα πρέπει να δηλωθεί (register), στη συνέχεια όταν το συμβάν το οποίο διαχειρίζεται λάβει χώρα ο BroadcastReceiver ενημερώνεται (receive) και διαχειρίζεται το συμβάν (handle). Υπάρχουν δύο τρόποι να δηλωθεί ένας BroadcastReceiver ο πρώτος είναι να δηλωθεί στο αρχείο AndroidManifest της εφαρμογής και ο δεύτερος είναι να δηλωθεί δυναμικά κατά την εκτέλεση της εφαρμογής, μέσω της μεθόδου *registerReceiver*. Όταν δηλώνεται κάποιος BrtoadcastReceiver, δηλώνονται όλα τα Intents τα οποία ο Receiver λαμβάνει. Στο σώμα της μεθόδου *onReceive* ορίζεται ο τρόπος με τον οποίο ο Receiver διαχειρίζεται το Intent. Στην GMA υπάρχουν οι εξής κλάσεις Receiver:

1. AutoStartReceiver

Δηλώνεται μέσω του AndroidManifest της εφαρμογής και λαμβάνει τα Intents BOOT_COMPLETED και QUICKBOOT_POWERON τα οποία είναι μηνύματα του λειτουργικού και γίνονται broadcast όταν η συσκευή ανοίγει:

```
<receiver android:
    <intent-fienabled="true" android:name=".Receivers.AutostartReceiver"
    android:exported="true">lter>
       <action android:name="android.intent.action.BOOT_COMPLETED"/>
       <action android:name="android.intent.action.QUICKBOOT_POWERON"/>
       <category android:name="android.intent.category.DEFAULT" />
    </intent-filter>
</receiver>
```

Όταν ο AutostartReceiver λάβει το Broadcast ξεκινά το GlucoseMonitorService

```
@Override
public void onReceive(Context context, Intent intent) {
      if(Intent.ACTION_BOOT_COMPLETED.equals(intent.getAction())) {
         Intent serviceIntent = new
      Intent(context,GlucoseMonitorService.class ,context.startForegroundServ
      ice(serviceIntent);
      }
}
```

2. BGDataReceiver

Δηλώνεται με κλήση της μεθόδου *registerReceiver* και δηλώνεται κατά την εκτέλεση της lifecycle μεθόδου *onStartCommand* του GlucoseMonitorService. Λαμβάνει τα Intents

com.eveningoutpost.dexdrip.Intents.ACTION_NEW_BG_ESTIMATE και com.eveningoutpost.dexdrip.BgEstimate τα οποία γίνονται broadcast από την εφαρμογή xDrip⁺.

```
@Override
public int onStartCommand(Intent intent, int flags, int startId) {
   ⋮
      BGDataReceiver bgDataReceiver = new BGDataReceiver();
      IntentFilter intentFilter = new IntentFilter();
```

```
        intentFilter.addAction("com.eveningoutpost.dexdrip.Intents.ACTION_NEW_BG_ESTIMATE");
        intentFilter.addAction("com.eveningoutpost.dexdrip.BgEstimate");

        if (Build.VERSION.SDK_INT >= Build.VERSION_CODES.TIRAMISU) {
           registerReceiver(bgDataReceiver,intentFilter,Context.RECEIVER_NOT_EXPORTED);
        } else {
           registerReceiver(bgDataReceiver, intentFilter);
        }
    ⋮
}
```

Στην *onReceive* όταν λαμβάνει το Intent, αποθηκεύει μέσω του BGRepository τη νέα τιμή στη βάση.

```
@Override
public void onReceive(Context context, Intent intent) {
    ⋮
        BGRepository bgRepository = new BGRepository(context);
        try {
           bgRepository.insert(bgModel);
        } catch (Exception ignored) {
        }
    ⋮
}
```

3. MockValueReceiver

Δηλώνεται μέσω του AndroidManifest της εφαρμογής και λαμβάνει το Intent ACTION_GENERATE_BG το οποίο είναι μήνυμα της εφαρμογής και εξυπηρετεί στη λειτουργία παραγωγής εικονικών τιμών BG:

```
<receiver
android:name="com.cchatzidopavlakis.diplomatiki.Receivers.MockValueReceiver"
android:exported="true"/>
```

Στην *onReceive* όταν λαμβάνει το Intent παράγει μια νέα τιμή BG και δρομολογεί νέο alarm το οποίο θα οδηγήσει στο επόμενο broadcast του “ACTION_GENERATE_BG “

```
if ("ACTION_GENERATE_BG".equals(intent.getAction())) {
        generateBG();
        rescheduleAlarm(context);
}
```

4. AlarmReceiver

Δηλώνεται μέσω του AndroidManifest της εφαρμογής και λαμβάνει το Explicit Intent :

```
Intent intent = new Intent(context, AlarmReceiver.class);

<receiver android:name=".Receivers.AlarmReceiver"
android:exported="true"/>
```

Στην *onReceive* όταν λάβει το Intent ξεκινά το AlarmService μέσω του οποίου εκτελείται η ενεργοποίηση των συναγερμών.

```
@Override
public void onReceive(Context context, Intent intent) {
      PowerManager powerManager = (PowerManager)
      context.getSystemService(Context.POWER_SERVICE);
      PowerManager.WakeLock wakeLock =
      powerManager.newWakeLock(PowerManager.FULL_WAKE_LOCK |
      PowerManager.ACQUIRE_CAUSES_WAKEUP , "GMA:AlertReceiverWakeLock");
      wakeLock.acquire(4 * 60 * 1000L );
      try {
           Intent serviceIntent = new Intent(context, AlertService.class);
           int bgValue = intent.getIntExtra("bgValue",0);
           String time = intent.getStringExtra("time");
           serviceIntent.putExtra("bgValue",bgValue);
           serviceIntent.putExtra("time",time);
           context.startService(serviceIntent);
      }finally {
           if(wakeLock.isHeld())
           wakeLock.release();
       }
}
```

4.4 Άδειες και Δικαιώματα

Όπως έχει αναλυθεί και στην ενότητα 3.5, η GMA ζητά από το λειτουργικό σύστημα και τους χρήστες συγκεκριμένα Permissions. Τα Permissions αυτά μπορεί να είναι απλά Normal, Runtime ή Special Permissions. Ανεξαρτήτως κατηγορίας όλα τα Permissions που ζητά η εφαρμογή συμπεριλαμβάνονται στο αρχείο AndroidManifest.xml. Το AndroidManifest.xml είναι το αρχείο διαμόρφωσης της εφαρμογής και περιέχει πληροφορίες που το Android χρειάζεται για να εκτελέσει και να διαχειριστεί σωστά την εφαρμογή. Μια από αυτές τις πληροφορίες είναι και τα Permissions που χρειάζεται η εφαρμογή ώστε να λειτουργεί αποτελεσματικά. Το AndroidManifest της GMA περιέχει τα Permissions:

```
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.USE_FULL_SCREEN_INTENT" />
<uses-permission android:name="android.permission.POST_NOTIFICATIONS" />
<uses-permission android:name="android.permission.SCHEDULE_EXACT_ALARM" />
<uses-permission android:name="android.permission.VIBRATE" />
<uses-permission android:name="android.permission.WAKE_LOCK" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE" />
<uses-permission android:name="android.permission.FOREGROUND_SERVICE_HEALTH" />
```

```
<uses-permission android:name="android.permission.SYSTEM_ALERT_WINDOW" />
<uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.READ_EXTERNAL_STORAGE" />
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.ACCESS_NOTIFICATION_POLICY"/>
<uses-permission android:name="android.permission.POST_NOTIFICATIONS"/>
<uses-permissionandroid:name="android.permission.
REQUEST_IGNORE_BATTERY_OPTIMIZATIONS"/>
<uses-permission android:name="android.permission.RECEIVE_BOOT_COMPLETED" />
<uses-permission android:name="android.permission.RECEIVE_QUICKBOOT_POWERON" />
```

Όλα τα Runtime Permissions, χρειάζονται να λάβουν την άδεια του χρήστη τουλάχιστον μια φορά και κατά την εκτέλεση της εφαρμογής. Αυτό επιτυγχάνεται με ελέγχους για το αν η GMA έχει ήδη λάβει το αντίστοιχο Permission μέσω της μεθόδου *ContextCompat.checkSelfPermission* , η οποία ελέγχει και επιστρέφει αν η εφαρμογή έχει λάβει κάποιο συγκεκριμένο Permission. Εφόσον η εφαρμογή δεν έχει το απαιτούμενο Permission για την εκτέλεση κάποιας λειτουργίας, το ζητά από το χρήστη χρησιμοποιώντας τη μέθοδο *ActivityCompat.requestPermissions.* Η οποία ζητά από το λειτουργικό να εμφανίσει το αναδυόμενο παράθυρο μέσω του οποίου ο χρήστης μπορεί να παραχωρήσει το ζητούμενο Runtime Permission.

Τα Special Permissions, είναι απαραίτητο να τα παραχωρήσει ο χρήστης από τις ρυθμίσεις της συσκευής του. Για το σκοπό αυτό και προκειμένου να είναι πιο εύκολο για τους χρήστες να παραχωρήσουν τα απαραίτητα Permissions, η εφαρμογή κάθε φορά που ανοίγει ελέγχει αν διαθέτει κάθε ένα από τα Special Permissions που χρειάζεται. Εφόσον κάποιο Permission δεν έχει εγκριθεί, η εφαρμογή εμφανίζει στο χρήστη ένα μήνυμα που τον ενημερώνει για το απαραίτητο Permission και του προτείνει να τον ανακατευθύνει η ίδια εκεί. Εφόσον ο χρήστης το επιλέξει, η GMA ανακατευθύνει το χρήστη στο σημείο της εκάστοτε ρύθμισης μέσω Intent. Πιο συγκεκριμένα για τα Special Permissions SCHEDULE_EXACT_ALARM, REQUEST_IGNORE_BATTERY_OPTIMIZATIONS, ACCESS_NOTIFICATION_POLICY και SYSTEM_ALERT_WINDOW χρησιμοποιούνται τα Intents:

```
Intent alarmIntent = new Intent(Settings.ACTION_REQUEST_SCHEDULE_EXACT_ALARM);

Intent batteryIntent = new Intent();
batteryIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
batteryIntent.setAction(ACTION_REQUEST_IGNORE_BATTERY_OPTIMIZATIONS);
batteryIntent.setData(Uri.parse("package:" + packageName));

Intent notificationPolicyIntent = new Intent(android.provider.Settings.
ACTION_NOTIFICATION_POLICY_ACCESS_SETTINGS);

Intent overlayIntent = new Intent(Settings.ACTION_MANAGE_OVERLAY_PERMISSION,
Uri.parse("package:" + context.getPackageName()));
overlayIntent.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
```

5. Επίλογος

5.1 Συμπεράσματα

Ο σχεδιασμός και η υλοποίηση της εφαρμογής Glucose Monitor App μας οδήγησε σε συγκεκριμένα συμπεράσματα όσον αφορά τη δημιουργία μιας εφαρμογής που αποτελεί μέρος συστήματος “Τεχνητού Παγκρέατος” και την μετέπειτα χρήση της από τον τελικό χρήστη.

Όσον αφορά εφαρμογές σαν την GMA και τα συστήματα “Τεχνητού Παγκρέατος” γενικότερα, είναι προφανές ότι βοηθούν με πολλούς τρόπους τους ασθενείς που ακολουθούν ινσουλινοθεραπεία, συμβάλλει ουσιαστικά στη θεραπευτική αντιμετώπιση του διαβήτη και τη βελτίωση της ποιότητας ζωής των ασθενών. Ο ουσιαστικός σκοπός τέτοιων θεραπειών είναι η μεγιστοποίηση των χρονικών περιόδων κατά τις οποίες το σάκχαρο του ασθενούς βρίσκεται εντός στόχου και η ελαχιστοποίηση των χρονικών περιόδων κατά τις οποίες παρουσιάζει υπεργλυκαιμία ή υπογλυκαιμία. Η δυνατότητα παρακολούθησης του σακχάρου καθ’ όλη τη διάρκεια της ημέρας, συνοδευόμενη από τους συναγερμούς, τις αναφορές και όλα τα υπόλοιπα εργαλεία που προσφέρει η GMA απλοποιεί κατά πολύ την καθημερινότητα των ασθενών. Ωστόσο, ο σημαντικότερος παράγοντας που επηρεάζει τη διαχείριση και την καλύτερη εφαρμογή της ινσουλινοθεραπείας, είναι η εκπαίδευση του ασθενούς όσον αφορά τη νόσο, τη θεραπεία και τους παράγοντες της καθημερινότητάς του που τις επηρεάζουν.

Η σοβαρότητα της νόσου και ο αυξανόμενος αριθμός ανθρώπων που νοσούν παγκοσμίως, οδηγούν ολοένα και περισσότερους κερδοσκοπικούς ή μη οργανισμούς να δραστηριοποιούνται στο πεδίο της αντιμετώπισης του διαβήτη με χρήση συστημάτων τεχνητού παγκρέατος. Εταιρείες από όλο τον κόσμο δημιουργούν και εξελίσσουν συστήματα CGM και αντλίες ινσουλίνης, ενώ παράλληλα αναπτύσσονται βελτιωμένα λογισμικά ελέγχου και διαχείρισής τους.

5.2 Μελλοντικές επεκτάσεις

Με βάση τις παρατηρήσεις και τα συμπεράσματα που διατυπώθηκαν προηγουμένως, μπορούμε να αναφέρουμε ορισμένες πιθανές μελλοντικές επεκτάσεις της εφαρμογής Glucose Monitor App:

- Η προσθήκη περισσότερων πηγών από τις οποίες λαμβάνονται δεδομένα γλυκόζης. Στο εμπόριο υπάρχουν πολλές συσκευές συνεχούς καταγραφής γλυκόζης οι οποίες θα μπορούσαν να υποστηρίζονται από την εφαρμογή.
- Η επέκταση των δυνατοτήτων του server. Ο server θα μπορούσε να παρέχει λειτουργικότητες όπως ο αυτόματος συγχρονισμός τιμών γλυκόζης, η δημιουργία online αναφορών και η δυνατότητα ενημέρωσης τρίτων σε περίπτωση ενεργοποίησης κάποιου συναγερμού.
- Η υποστήριξη σύνδεσης με αντλίες ή πένες έγχυσης ινσουλίνης προκειμένου ο χρήστης να μπορεί να πραγματοποιεί εγχύσεις ινσουλίνης μέσω της εφαρμογής.

- Η βελτιστοποίηση του αλγορίθμου υπολογισμού έκτακτων δόσεων ινσουλίνης ώστε συνυπολογίζοντας περισσότερους παράγοντες, να παρέχει περισσότερες και πιο ολοκληρωμένες προτάσεις θεραπείας.

6. Βιβλιογραφία

[1] World Health Organization, "Overview," *WHO*, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes [Accessed: Jan. 22, 2024].

[2] Mayo Clinic, "Overview," *Mayo Clinic*, [Online]. Available: https://www.mayoclinic.org/diseases-conditions/type-1-diabetes/symptoms-causes/syc-20353011 [Accessed: Jan. 22, 2024].

[3] Mayo Clinic, "Overview," *Mayo Clinic*, [Online]. Available: https://www.mayoclinic.org/diseases-conditions/type-2-diabetes/symptoms-causes/syc-20351193 [Accessed: Jan. 22, 2024].

[4] World Health Organization, "Gestational diabetes," *WHO*, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes [Accessed: Jan. 22, 2025].

[5] World Health Organization, "Key facts," *WHO*, [Online]. Available: https://www.who.int/news-room/fact-sheets/detail/diabetes [Accessed: Jan. 22, 2025].

[6] Υπουργείο Υγείας Ελλάδας, "Διαβήτης," *Υπουργείο Υγείας Ελλάδας*, [Online]. Available: https://www.moh.gov.gr/articles/news/848-enhmerwtiko-deltio-toy-pagkosmioy-organismoy-ygeias-poy-gia-to-diabhth [Accessed: Jan. 28, 2025].

[7] National Institute of Diabetes and Digestive and Kidney Diseases, "Artificial Pancreas," *NIDDK*, [Online]. Available: https://www.niddk.nih.gov/health-information/diabetes/overview/managing-diabetes/artificial-pancreas#artificial [Accessed: Jan. 28, 2025].

[8] Bekiari, E., Kitsios, K., Thabit, H., Tauschmann, M., Athanasiadou, E., Karagiannis, T., Haidich, A. B., Hovorka, R., & Tsapas, A. (2018). Artificial pancreas treatment for outpatients with type 1 diabetes: systematic review and meta-analysis. BMJ (Clinical Research Ed.), 361. https://doi.org/10.1136/BMJ.K1310

[9] Britannica, "Android operating system," *Britannica*, [Online]. Available: https://www.britannica.com/technology/Android-operating-system [Accessed: Oct. 20, 2024].

[10] Google, "How Google Play works," *Google*, [Online]. Available: https://play.google/howplaywork [Accessed: Oct. 20, 2024].

[11] Business of Apps, "Google Play Store Statistics (2024)," *Business of Apps*, [Online]. Available: https://www.businessofapps.com/data/google-play-statistics [Accessed: Oct. 20, 2024].

[12] Google Developers, "Meet Android Studio," *Google Developers*, [Online]. Available: https://developer.android.com/studio/intro [Accessed: Oct. 21, 2024].

[13] Amazon Web Services, "What is Java," *AWS*, [Online]. Available: https://aws.amazon.com/what-is/java [Accessed: Oct. 21, 2024].

[14] IBM, "Java Development Kit," *IBM*, [Online]. Available: https://www.ibm.com/docs/en/i/7.4?topic=platform-java-development-kit [Accessed: Oct. 21, 2024].

[15] Δ. Σαούγκος, *Συμπίεση JAVA BYTECODE*, Πανεπιστήμιο Ιωαννίνων, 2006. [Online]. Available: https://www.cse.uoi.gr/wp-content/uploads/publications/Thesis.pdf

[16] JavaTpoint, "JVM (Java Virtual Machine) Architecture," *JavaTpoint*, [Online]. Available: https://www.javatpoint.com/jvm-java-virtual-machine [Accessed: Oct. 21, 2024].

[17] TechTarget, "What is Google Firebase," *TechTarget*, [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase [Accessed: Oct. 28, 2024].

[18] TechTarget, "Is Google Firebase secure?" *TechTarget*, [Online]. Available: https://www.techtarget.com/searchmobilecomputing/definition/Google-Firebase [Accessed: Oct. 28, 2024].

[19] Nightscout Foundation, "README," *GitHub*, [Online]. Available: https://github.com/NightscoutFoundation/xDrip?tab=readme-ov-file

[20] Nightscout Foundation, "About," *Nightscout Foundation*, [Online]. Available: https://www.nightscoutfoundation.org/about

[21] Epoch Converter, "What is epoch time?" *Epoch Converter*, [Online]. Available: https://www.epochconverter.com [Accessed: Nov. 2, 2024].

[22] Cleveland Clinic, "Injectable Insulin Medications," *Cleveland Clinic*, [Online]. Available: https://my.clevelandclinic.org/health/drugs/13902-injectable-insulin-medications [Accessed: Feb. 5, 2025].

[23] Google Developers, "Types of permissions," *Google Developers*, [Online]. Available: https://developer.android.com/guide/topics/permissions/overview [Accessed: Feb. 10, 2025].

[24] Google Developers, "App Components," *Google Developers*, [Online]. Available: https://developer.android.com/guide/components/fundamentals#components [Accessed: Feb. 11, 2025].

[25] GeeksforGeeks, "MVVM (Model View ViewModel) Architecture Pattern in Android," *GeeksforGeeks*, [Online]. Available: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android [Accessed: Feb. 11, 2025].

[26] Google Developers, "LiveData Overview," *Google Developers*, [Online]. Available: https://developer.android.com/topic/libraries/architecture/livedata#:~:text=LiveData%20is%20a%20wrapper%20that,Kotlin%20Java [Accessed: Feb. 12, 2025].

[27] Google Developers, "Save data in a local database using Room," *Google Developers*, [Online]. Available: https://developer.android.com/training/data-storage/room [Accessed: Feb. 12, 2025].

[28] GitHub, "LICENSE," *GitHub*, [Online]. Available: https://github.com/PhilJay/MPAndroidChart/blob/master/LICENSE [Accessed: Feb. 13, 2025].

Συντομογραφίες - Αρκτικόλεξα- Ακρωνύμια

κ.α.	και άλλα
κ.λπ	και λοιπά
κ.ο.κ.	και ούτω καθεξής
π.χ	παραδείγματος χάρη
APK	Android Package Kit
BG	Blood Glucose
CGMS	Continuous Glucose Monitoring System
COB	Carbs On Board
CR	Carbohydrate Ratio
DCA	Duration of Carbohydrate Activity
DIA	Duration of Insulin Activity
GMA	Glucose Monitor App
IDE	Integrated Development Environment
IOB	Insulin On Board
ISF	Insulin Sensitivity Factor
JDK	Java Development Kit
JRE	Java Runtime Environment
JVM	Java Virtual Machine
MVVM	Model View ViewModel
UTC	Coordinated Universal Time

Απόδοση Ξενόγλωσσων Όρων

Ξενόγλωσσος Όρος	Απόδοση
Reports	Αναφορές
Insulin Pump	Αντλία έγχυσης Ινσουλίνης
Debugging	Αποσφαλμάτωση
Data Repositories	Αποθετήρια Δεδομένων
Wirelessly	Ασύρματα
Blood Glucose	Γλυκόζη Αίματος
Activity	Δραστηριότητα
Documents	Έγγραφα
Emulator	Εξομοιωτής
Smartphones	Έξυπνα Κινητά Τηλέφωνα
Server	Εξυπηρετητής
Service	Υπηρεσία
Code Editor	Επεξεργαστής Κώδικα
Shared Preferences	Κοινές Προτιμήσεις
Password	Κωδικός
Internet	Παγκόσμιος Ιστός
Source Code	Πηγαίος Κώδικας
Alerts	Συναγερμοί
Coordinated Universal Time	Συντονισμένη Παγκόσμια Ώρα
Drawer Menu	Συρόμενο Μενού
Cloud	Υπολογιστικό νέφος
Fragment	Υποτμήμα
Wearable Device	Φορέσιμη Συσκευή