ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Android εφαρμογής για διαδικτυακό φαρμακείο με την χρήση Firebase

Του φοιτητή
Αριστείδου Ραφαήλ
Αρ. Μητρώου: 154419

Επιβλέπων
Κεραμόπουλος Ευκλείδης
Αναπληρωτής Καθηγητής

Ημερομηνία 27-08-2022

Τίτλος Δ.Ε. Ανάπτυξη Android εφαρμογής για διαδικτυακό φαρμακείο με την χρήση Firebase.
Κωδικός Δ.Ε. 20221
Ονοματεπώνυμο φοιτητή: Αριστείδου Ραφαήλ
Ονοματεπώνυμο εισηγητή: Κεραμόπουλος Ευκλείδης
Ημερομηνία ανάληψης Δ.Ε. 30-10-2020
Ημερομηνία περάτωσης Δ.Ε. 27-08-2020

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως διπλωματική εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Στην οικογένεια μου και τους κοντινούς μου ανθρώπους

Πρόλογος

Κατά τη διάρκεια της πανδημίας όλοι οι καταναλωτές παγκοσμίως περιόρισαν τις μετακινήσεις τους, κυρίως από ευάλωτες ομάδες ανθρώπων που χρειάζονται φαρμακευτικά προϊόντα, με αποτέλεσμα η αγορά αυτών μέσω διαδικτύου να αποτελεί μονόδρομο. Με τη δημιουργία της παρούσας εργασίας προσπάθησα να δημιουργήσω μια εύκολη προς τον χρήστη εφαρμογή, που θα τον βοηθήσει να μειώσει όσο το δυνατόν μπορεί την έκθεση του ήδη αδύναμου του οργανισμού σε διάφορες άλλες ασθένειες .

Περίληψη

Στόχος της παρούσας διπλωματικής εργασίας είναι η δημιουργία μίας εφαρμογής ηλεκτρονικού φαρμακείου με εύχρηστο περιβάλλον για τον πελάτη όπου εκτός από αγορές θα του παρέχει και τη δυνατότητα να αποστέλλει ηλεκτρονικά μέσω της εφαρμογής, με τρεις διαφορετικούς τρόπους, τη συνταγή φαρμάκων που του έχει παραχωρήσει ο Γιατρός του. Επίσης, μέσω μιας δεύτερης εφαρμογής ο φαρμακοποιός θα μπορεί να διαχειριστεί τις παραγγελίες και τα προϊόντα του. Η χρήση της firebase ως μια real-time database μας βοηθά να παρέχουμε στον χρήστη μία μοναδική ταχύτητα αλληλεπίδρασης με τη βάση δεδομένων.

Development of Android application for an online pharmacy using Firebase

Rafail Aristeidou

Abstract

The aim of this thesis is to design a pharmacy application which is user friendly and practical to the customer, where in addition to online shopping, it will also provide him with the possibility to send electronically through the application the drug prescription provided to him by his doctor, in three different ways. Also, through a second application, the pharmacist will be able to manage his orders and the products. Using firebase as a real-time database help us to provide the user with a uniquely fast interaction with the database.

Ευχαριστίες

Θα ήθελα να εκφράσω τις θερμές μου ευχαριστίες στον κύριο Κεραμόπουλο που μου έδωσε τη δυνατότητα να συνεργαστούμε και με την υποστήριξη και την επίβλεψη του να ολοκληρώσω τη διπλωματική μου εργασία. Ακόμη θα ήθελα να ευχαριστήσω την οικογένεια μου και όλα τα κοντινά μου άτομα για την συνεχή στήριξη τους.

Περιεχόμενα

Πρόλογος v
Περίληψη vi
Abstract vii
Ευχαριστίες viii
Περιεχόμενα ix
Κατάλογος Σχημάτων xii
Συντομογραφίες xiv
Κεφάλαιο 1ο: Εισαγωγή 1
1.1 Εισαγωγή 1
1.2 Συνταγογραφούμενα φάρμακα 1
1.3 Στόχος 1
1.4 Δομή 1
1.5 Επίλογος 2
Κεφάλαιο 2ο: Firebase 3
2.1 Εισαγωγή 3
2.2 Authentication 3
2.3 Cloud Firestore 4
2.3.1 Indexes 5
2.4 Cloud Storage 5
2.5 Google Analytics 6
2.6 Επίλογος 7
Κεφάλαιο 3ο: Σχεδιασμός Android 9
3.1 Εισαγωγή 9
3.2 Android Studio 9
3.3 JAVA 9
3.4 Gradle 10
3.5 Δραστηριότητες 10
3.6 Fragments 11
3.7 Resources 12
3.8 Design Pattern 13

3.8.1 MVC 13
3.8.2 MVP 13
3.8.3 MVVM 13
3.9 Επίλογος 14
Κεφάλαιο 4ο: UI Design 15
4.1 Εισαγωγή 15
4.2 RecyclerView 15
4.3 TextView 15
4.4 EditText 15
4.5 Button 16
4.6 Linear Progress Indicator 16
4.7 Extended Floating Action Button 16
4.8 RadioGroup / RadioButton 17
4.9 ImageView 17
4.10 Chips 18
4.11 Bottom Navigation Bar 18
4.12 Επίλογος 18
Κεφάλαιο 5ο: Απαιτήσεις 19
5.1 Εισαγωγή 19
5.2 Λειτουργικές Απαιτήσεις 19
5.2.1 Χρήστης 19
5.2.2 Διαχειριστής 20
5.3 Μη Λειτουργικές Απαιτήσεις 21
5.4 Επίλογος 21
Κεφάλαιο 6ο: Λειτουργίες της εφαρμογής 23
6.1 Εισαγωγή 23
6.2 Απλός Χρήστης 23
6.2.1 Κεντρική Οθόνη 23
6.2.2 Προσθήκη στο Καλάθι 24
6.2.3 Καλάθι 25
6.2.4 Checkout 26
6.2.5 Αποστολή συνταγογραφούμενης παραγγελίας 27
6.2.6 Login 29
6.2.7 Sign Up 30
6.2.8 My Profile 31

6.2.9 My Orders ... 31
6.2.10 My Details ... 34
6.2.11 Change Password ... 35
6.2.12 Address book ... 36
6.2.13 Sign Out ... 37
6.3 Διαχειριστής ... 38
6.3.1 Προσθήκη κατηγορίας ή διαγραφή κατηγορίας ... 38
6.3.2 Δημιουργία προσφοράς ή επεξεργασία ήδη υπάρχουσας ... 39
6.3.3 Προσθήκη προϊόντος ... 39
6.3.4 Orders ... 40
6.3.5 Order Info ... 41
6.4 Επίλογος ... 42
Κεφάλαιο 7ο: Υλοποίηση ... 43
7.1 Εισαγωγή ... 43
7.2 Δομή εφαρμογής χρήστη ... 43
7.2.1 Μοντέλα ... 43
7.2.2 Πακέτα Διεπαφών Χρήστη ... 44
7.2.3 Κλάση Προβολής ... 45
7.2.4 Κλάση Μοντέλο Προβολής ... 46
7.2.5 Πακέτο util ... 47
7.2.6 MainActivity ... 47
7.3 Στοιχεία Αρχιτεκτονικής ... 48
7.3.1 LiveData ... 48
7.3.2 View Binding ... 49
7.3.3 RecyclerView ... 50
7.3.4 Glide ... 53
7.3.5 Navigation ... 53
7.3.6 SafeArgs Plugin ... 55
7.3.7 Intent ... 56
7.4 Επίλογος ... 57
Κεφάλαιο 8ο: Συμπεράσματα και προτάσεις βελτίωσης ... 59
ΒΙΒΛΙΟΓΡΑΦΙΑ ... 61

Κατάλογος Σχημάτων

Σχήμα 2.1: Αποθήκευση χρήστη στο Firebase 3
Σχήμα 2.2: Κώδικας μεθόδου πρόσβασης χρήστη 4
Σχήμα 2.3: Collection addresses της εφαρμογής με τα documents 4
Σχήμα 2.4: Δύο ευρετήρια 5
Σχήμα 2.5: Μέθοδος δημιουργίας και αποθήκευσης εγγράφου με ανέβασμα φωτογραφίας 6
Σχήμα 2.6: Παράδειγμα Analytics Events της εφαρμογής χρήστη 6
Σχήμα 2.7: Παράδειγμα Analytics της εφαρμογής χρήστη 7
Σχήμα 3.1: Μερίδιο αγοράς του λειτουργικού συστήματος smartphone το 2022 9
Σχήμα 3.2: Κύκλος ζωής μιας Δραστηριότητας σε επανεκκίνηση 11
Σχήμα 3.3: Κύκλος ζωής μιας Δραστηριότητας μαζί με το Fragment της 12
Σχήμα 3.4: Φάκελος Resouces από εφαρμογή χρήστη 12
Σχήμα 3.5: MVVM μοτίβο αρχιτεκτονικής λογισμικού[17] 14
Σχήμα 4.1: RecyclerView κατηγορίες 15
Σχήμα 4.2: Linear Progress Indicator καρτέλα παραγγελιών 16
Σχήμα 4.3: Linear Progress Indicator πληροφορίες παραγγελίας 16
Σχήμα 4.4: ExtendedFloatingActionButton 16
Σχήμα 4.5: RadioGroup με δυο RadioButtons 17
Σχήμα 4.6: Χρήση imageView για εμφάνιση εικόνας προϊόντος 17
Σχήμα 4.7: Chips για φίλτρο 18
Σχήμα 4.8: Navigation Bar τοποθετημένη στο κάτω μέρος της οθόνης 18
Σχήμα 6.1: Κεντρική οθόνη εφαρμογής χρήστη 23
Σχήμα 6.2: Εμφάνιση προϊόντων κατηγορίας “Βρέφος” 24
Σχήμα 6.3: Οθόνη λεπτομερειών για προϊών 24
Σχήμα 6.4: Επιβεβαίωση προσθήκης προϊόντος στο καλάθι 25
Σχήμα 6.5: Αριθμός προϊόντων που βρίσκονται στο καλάθι 25
Σχήμα 6.6: Προϊόντα που βρίσκονται στο καλάθι 26
Σχήμα 6.7: Φόρμα ολοκλήρωσης της παραγγελία μας 27
Σχήμα 6.8: Οθόνη επιβεβαίωσης παραγγελίας 27
Σχήμα 6.9: Οθόνη επιλογής τύπου για συνταγογραφούμενη παραγγελία 28
Σχήμα 6.10: Οθόνη αποστολής αριθμού ηλεκτρονικής συνταγής 28
Σχήμα 6.11: Οθόνη αποστολής φωτογραφία συνταγής 29
Σχήμα 6.12: Οθόνη αποστολής συνταγής αρχείου PDF 29
Σχήμα 6.13: Οθόνη εισόδου χρήστη στον λογαριασμό 30
Σχήμα 6.14: Οθόνη εγγράφεις νέου χρήστη 30
Σχήμα 6.15: Οθόνη προφίλ του χρήστη 31
Σχήμα 6.16: Λίστα παραγγελιών του χρήστη 32
Σχήμα 6.17: Πληροφορίες παραγγελίας 32
Σχήμα 6.18: Πληροφορίες παραγγελίας με φωτογραφία συνταγής 33
Σχήμα 6.19: Πληροφορίες παραγγελίας με αρχείο PDF συνταγής 34
Σχήμα 6.20: Οθόνη τροποποιήσεις πληροφοριών χρήστη 35
Σχήμα 6.21: Alert Dialog Box για επαλήθευση κωδικού πρόσβασης 35
Σχήμα 6.22: Οθόνη αλλαγής κωδικού πρόσβασης 36
Σχήμα 6.23: Μήνυμα λάθος κωδικού 36
Σχήμα 6.24: Λίστα αποθηκευμένων διευθύνσεων του χρήστη 37

Σχήμα 6.25: Οθόνη αποθήκευσες νέας διεύθυνσης 37
Σχήμα 6.26: Κεντρική οθόνη εφαρμογής διαχειριστή 38
Σχήμα 6.27: Οθόνη δημιουργίας νέας κατηγορίας..... 38
Σχήμα 6.28: Οθόνη διαγραφής υπάρχον κατηγορίας..... 39
Σχήμα 6.29: Οθόνη επεξεργασίας προσφοράς..... 39
Σχήμα 6.30: Οθόνη προσθήκης νέου προϊόντος 40
Σχήμα 6.31: Λίστα παραγγελιών..... 41
Σχήμα 6.32: Λεπτομέρειες παραγγελίας και επεξεργασία 41
Σχήμα 7.1: Πακέτο μοντέλων..... 43
Σχήμα 7.2: Κλάση SpecialOffers 44
Σχήμα 7.3: Πακέτο με κλάσεις της κεντρικής οθόνης 44
Σχήμα 7.4: Μέθοδος setupViews() της κλάσης MainStoreFragment 45
Σχήμα 7.5: Μέθοδος setupObservers() της κλάσης MainStoreFragment 46
Σχήμα 7.6: Μέθοδοι isLoggedIn() και fetchCategories() της κλάσης MainViewModel 47
Σχήμα 7.7: Πακέτο util..... 47
Σχήμα 7.8: Μέθοδοι onCreate() της κλάσης MainActivity..... 48
Σχήμα 7.9: LiveData Architecture Component[21] 49
Σχήμα 7.10: Μέθοδος fetchOrders() της κλάσης OredersViewModel..... 49
Σχήμα 7.11: Παραδιγμα χρησης του ViewBinding στην μεθοδο setupViews() της κλασεις RegisterFragment 50
Σχήμα 7.12: Κλάση CategoriesAdapter 52
Σχήμα 7.13: Κλάση CategoriesAdapterViewHolder μέσα στην κλάση CategoriesAdapter..... 52
Σχήμα 7.14: Χρήση μεθόδου glide..... 53
Σχήμα 7.15: Το γράφημα πλοήγησης nav_graph.xml εφαρμογής διαχειριστή..... 54
Σχήμα 7.16: Το γράφημα πλοήγησης nav_graph.xml εφαρμογής διαχειριστή σε xml..... 54
Σχήμα 7.17: activity_main.xml 55
Σχήμα 7.18: Κλάση MainActivity..... 55
Σχήμα 7.19: nav_graph.xml με παραμέτρους 56
Σχήμα 7.20: Διαδρομή με παράμετρο 56
Σχήμα 7.21: Χρήση Intent για επιλογή αρχείου τύπου PDF 56
Σχήμα 7.22: Μέθοδος onActivityResult() μετά την επιλογή αρχείου PDF 57

Συντομογραφίες

Δ.Ε.	Διπλωματική Εργασία
ΔΙΠΑΕ	Διεθνές Πανεπιστήμιο Ελλάδος
BaaS	Backend as a Service
MVVM	Model-View-ViewModel
MVP	Model-View-Presenter
MVC	Model-View-Controller
URL	Uniform Resource Locator
PDF	Portable Document Format
XML	Extensible markup language
DSL	Domain Specific Language
SDK	Software Development Kit
UI	User Interface

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Στο κεφάλαιο αυτό θα αναπτύξουμε τα κίνητρα που μας οδήγησαν στην δημιουργία αυτής της διπλωματικής εργασίας που έχει ως στόχο τη δημιουργία μιας Android εφαρμογής για ένα ηλεκτρονικό φαρμακείο. Θα αναλύσουμε το μέλλον που έχει ένα ηλεκτρονικό φαρμακείο και πως αυτό μπορεί να βοηθήσει στις μέρες μας.

Από τις αρχές του 1980 μέχρι και σήμερα ο τρόπος αγοράς προϊόντων και υπηρεσιών έχει αλλάξει κατά πολύ με τις ηλεκτρονικές αγορές. Η φαρμακευτική βιομηχανία δεν μπορούσε να μην επηρεαστεί. Εκτός από τα πλεονεκτήματα που παρέχει μία ηλεκτρονική αγορά από ένα φαρμακείο στις μέρες μας παρέχει και ασφάλεια στην προσωπική μας υγεία. Κατά τη διάρκεια των lockdown λόγω της πανδημίας όλοι οι καταναλωτές παγκοσμίως περιόρισαν τις μετακινήσεις τους, όπου αυτό είχε ως αποτέλεσμα την αύξηση αγορών φαρμάκων μέσω διαδικτύου προσπαθώντας να διασφαλίσουν την ασφάλεια της υγεία τους και να μειώσουν τον κίνδυνο να εκτεθούν στην ασθένεια Covid-19.

1.2 Συνταγογραφούμενα φάρμακα

Η συνταγογράφηση φαρμάκων σε μια κόλλα χαρτί ανέκαθεν αποτελούσε τον αποκλειστικό τρόπο συνταγογράφησης κάποιου φαρμάκου. Ωστόσο, λόγω της ραγδαίας αύξησης της τεχνολογίας, η συνταγογράφηση δεν θα μπορούσε να μην επηρεαστεί, με αποτέλεσμα στην Ελλάδα να έχει δημιουργηθεί η δυνατότητα ηλεκτρονικής συνταγογράφησης. Με την εγγραφή σας στην πλατφόρμα λαμβάνονται από τον γιατρό σας τα στοιχεία των συνταγογραφημένων φαρμάκων με μήνυμα στο κινητό σας τηλέφωνο η μέσω ηλεκτρονικού ταχυδρομείου. Ο φαρμακοποιός εκτελεί τη συνταγή φαρμάκων με τον αριθμό (barcode) που έχετε λάβει.

Ωστόσο, η αποστολή και η αγορά συνταγογραφούμενων φαρμάκων μέσω διαδικτύου απαγορεύεται στην Ελλάδα. Για αυτό τον λόγο η αγορά των συνταγογραφούμενων φαρμάκων μπορεί να γίνει μόνο με τη φυσική παραλαβή από το φαρμακείο. Αυτό όμως δεν μειώνει τα οφέλη που μπορεί να έχει μία online αποστολή της συνταγής στο φαρμακείο έτσι ώστε να είναι έτοιμη για παραλαβή και να μειώσει τον χρόνο αναμονής.

1.3 Στόχος

Η εφαρμογή έχει ως στόχο την εύκολη πρόσβαση και αγορά φαρμακευτικών προϊόντων από τον πελάτη και ταυτόχρονα του δίνει την ευκολία να αποστείλει ηλεκτρονικά τον αριθμό συνταγής του. Στόχος της παρούσας διπλωματικής είναι η δημιουργία ενός ηλεκτρονικού φαρμακείου το οποίο θα περιλαμβάνει δύο διαφορετικές εφαρμογές, μία εφαρμογή για τον καταναλωτή και μία για τον διαχειριστή.

Από τα παραπάνω μπορούμε να αντιληφθούμε ότι ο μόνος βιώσιμος τρόπος λειτουργίας ενός ηλεκτρονικού φαρμακείου είναι υβριδικός, παρέχοντας ένα παραδοσιακό κατάστημα φαρμακείου το οποίο ταυτόχρονα δέχεται παραγγελίες μέσω διαδικτύου.

1.4 Δομή

Στο δεύτερο κεφάλαιο αυτής της εργασίας παρουσιάζεται το Firebase που αποτελεί Backend κομμάτι των δυο εφαρμογών. Παρουσιάζονται κάποιες από τις κύριες υπηρεσίες που παρέχει και παραδείγματα του πως έχουν χρησιμοποιηθεί στις δυο εφαρμογές.

Το τρίτο κεφάλαιο περιέχει πληροφορίες για κάποια από τα βασικά στοιχεία που αποτελούν μια Android εφαρμογή. Επίσης, παρουσιάζονται τα τρία επικρατέστερα αρχιτεκτονικά μοτίβα.

Στο τέταρτο κεφάλαιο παρουσιάζονται τα στοιχεία διεπαφής χρήστη που χρησιμοποιήθηκαν στις εφαρμογές με σκοπό τη δημιουργία ενός απλού περιβάλλοντος εργασίας.

Το πέμπτο κεφάλαιο περιγράφει τις λειτουργικές απαιτήσεις των δυο εφαρμογών ξεχωριστά με μικρή περιγραφή για κάθε μια από αυτές. Στο ίδιο κεφάλαιο βρίσκονται και οι μη λειτουργικές απαιτήσεις που θα έχει το σύστημα.

Στο κεφάλαιο έξι της εργασίας περιγράφονται οι λειτουργίες των δυο εφαρμογών ξεχωριστά. Παρουσιάζονται εικόνες από τις δυο εφαρμογές με μερικά λόγια για το πως λειτουργεί η κάθε οθόνη.

Στο επόμενο κεφάλαιο περιγράφετε ο τρόπος υλοποίησης των εφαρμογών όπως η δομή της εφαρμογής και στοιχεία αρχιτεκτονικής καθώς και ο τρόπος που αυτά χρησιμοποιήθηκαν στις εφαρμογές.

Τέλος, ακολουθεί το συμπέρασμα καθώς και μελλοντικές προτάσεις βελτίωσης των δυο εφαρμογών.

1.5 Επίλογος

Σε αυτό το κεφάλαιο έχουν αναφερθεί τα κίνητρα με κάποιες γενικές πληροφορίες που ισχύουν για τα συνταγογραφούμενα φάρμακα στην Ελλάδα άλλα και τον στόχο που έχει η δημιουργία αυτής της Δ.Ε.

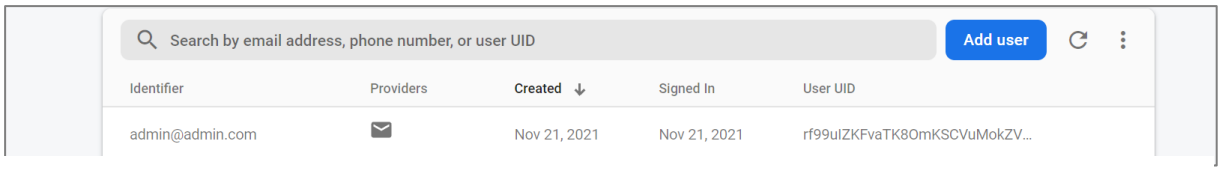
Κεφάλαιο 2ο: Firebase

2.1 Εισαγωγή

Ένας από τους στόχους μας και κύρια πρόκληση σε αυτήν τη διπλωματική ήταν να δούμε κατά πόσο είναι εφικτό οι δυο εφαρμογές να επικοινωνούν άμεσα αλλά και να αποθηκεύουν τα δεδομένα τους χωρίς την ανάγκη κάποιου server (serverless application). Αυτό έγινε εφικτό με τη βοήθεια της Google Firebase που θεωρείται και ως μια BaaS υπηρεσία (Backend as a Service), η οποία έχει σκοπό την απλοποίηση των ρυθμίσεων αλλά και των διαμορφώσεων που πρέπει να γίνουν στο Backend περιβάλλον. Η Firebase δημιουργήθηκε από μια start-up εταιρεία το 2011 και σκοπό είχε να παρέχει ένα API για το chatbot της σελίδας τους. Βλέποντας όμως τη δυνατότητά να μεταφέρουν και δεδομένα μέσω αυτού σε πραγματικό χρόνο αποφάσισαν να τη διαχωρίσουν και να τη δώσουν στο κοινό το 2012[1]. Το 2014 η εταιρία αγοράστηκε από την Google και από τότε η Google έχει επενδύσει αρκετά στη Firebase κάνοντας την κομμάτι του Google Cloud Platform, δίνοντας τη δυνατότητα να αλληλοεπιδρά τόσο με τα Google Service όσο και με πολλές υπηρεσίες τρίτων.

2.2 Authentication

Χρησιμοποιώντας το Firebase Authentication, μπορούμε να περιορίσουμε την πρόσβαση των χρηστών σε δεδομένα, παρέχοντας ασφάλεια στην εφαρμογή αλλά και στα δεδομένα του χρήστη. Στον χρήστη παρέχεται ασφαλή πρόσβαση στον λογαριασμό του με την χρήση κωδικού, email, αριθμό τηλεφώνου ακόμη και έλεγχο ταυτότητας μέσω των κοινωνικών του δικτύων. Συγκεκριμένα στην εφαρμογή μας παρέχουμε στον χρήστη για έλεγχο πρόσβασης, την χρήση του email του και κωδικό πρόσβασης όπως φαίνεται στο Σχήμα 2.1. Το email είναι μοναδικό για τον κάθε χρήστη και ακόμη παρέχει την δυνατότητα σε περίπτωση που έχει ξεχάσει των κωδικό του να του αποσταλεί email για την αλλαγή πρόσβασης στον λογαριασμό του. Για την αποθήκευση των κωδικών πρόσβασης, το Firebase Authentication χρησιμοποιεί μια τροποποιημένη έκδοση κατακερματισμού των κωδικών πρόσβασης[2].



Σχήμα 2.1: Αποθήκευση χρήστη στο Firebase

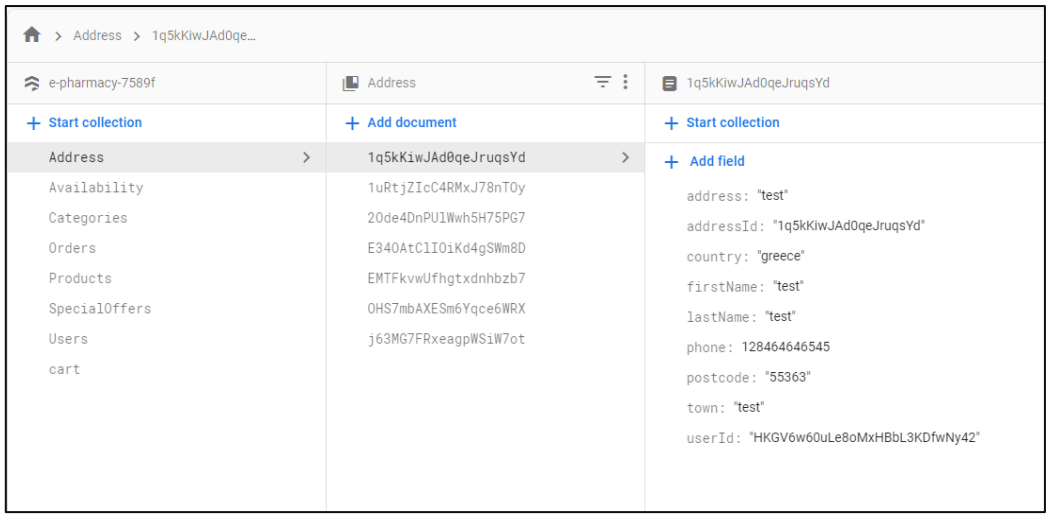
Στο Σχήμα 2.2 μπορούμε να δούμε την μέθοδο login(). Η μέθοδος login() έχει δυο παραμέτρους, το email και τον κωδικό πρόσβασης του χρήστη. Η επαλήθευση των στοιχείων του χρήστη γίνεται με την μέθοδο signInWithEmailAndPassword() η οποία παρέχεται από το Firebase.

```
public void login(String email, String password) {
      auth.signInWithEmailAndPassword(email, password).addOnCompleteListener(task -> {
          if (task.isSuccessful()) {
              set_user();
              _isSuccess.postValue(true);
          } else {
              _error.postValue(task.getException().getMessage());
          }
      });
  }
```

Σχήμα 2.2: Κώδικας μεθόδου πρόσβασης χρήστη

2.3 Cloud Firestore

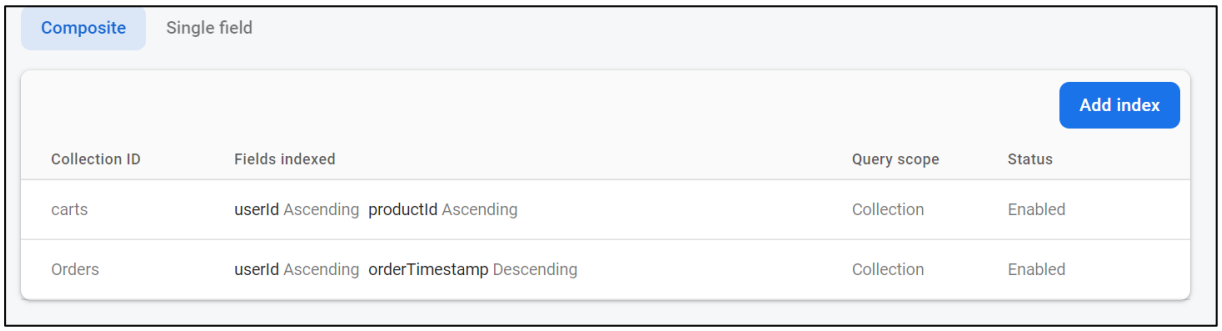
Η Firebase προσφέρει δύο τύπους μη σχεσιακών βάσεων δεδομένων: Firebase Realtime Database και Cloud Firestore. Και οι δύο βάσεις δεδομένων επιτρέπουν στους χρήστες να συγχρονίζουν δεδομένα μεταξύ τους σε πραγματικό χρόνο χρησιμοποιώντας listeners που καλούνται κάθε φορά που εντοπίζεται μια αλλαγή. Ο κύριος λόγος για τον οποίο επιλέχθηκε το Firestore Database ήταν επειδή τα δεδομένα είναι πιο δομημένα από την Realtime Database[3]. Στη Realtime Database, τα δεδομένα αποθηκεύονται σε ένα δέντρο JavaScript Object Notation (JSON), ενώ στην Firestore Database, τα δεδομένα οργανώνονται σε documents και collections που είναι πιο εύκολο να αναζητηθούν. Στο Σχήμα 2.3 βλέπουμε στα αριστερά την στήλη με τα collections, στην μεσαία στήλη τα documents που ανήκουν στο συγκεκριμένο collections, ενώ στην δεξιά στήλη είναι όλα τα πεδία που περιέχονται στο συγκεκριμένο document.



Σχήμα 2.3: Collection addresses της εφαρμογής με τα documents

2.3.1 Indexes

Το Firestore διασφαλίζει την απόδοση των ερωτημάτων (queries) δημιουργώντας αυτόματα για εμάς ένα ευρετήριο (index) για κάθε βασικό ερώτημα[4]. Για πιο σύνθετα ερωτήματα μπορούμε να δημιουργήσουμε επιπλέον ευρετήρια έτσι ώστε να διασφαλίσουμε την ταχύτητα εκτέλεσης τους. Στο Σχήμα 2.4 μπορούμε να δούμε δυο indexs τα οποία έχουμε δημιουργήσει για τις εφαρμογές. Για παράδειγμα το index orders μας επιστρέφει τις παραγγελίες του χρήστη με φθίνουσα σειρά αναλόγως με της ημερομηνίας που έχει καταχωρηθεί η παραγγελία.



Σχήμα 2.4: Δύο ευρετήρια

2.4 Cloud Storage

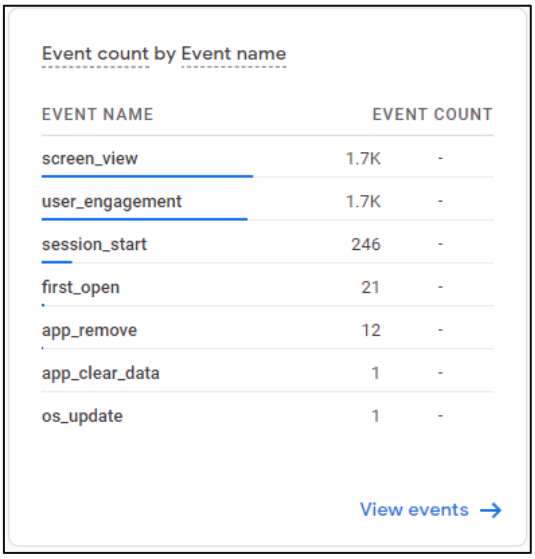
Το Cloud Storage έχει σχεδιαστεί για να ανεβάζουμε και να κατεβάζουμε αρχεία όπως εικόνες, ήχο, βίντεο και άλλο περιεχόμενο που έχει δημιουργηθεί από τον χρήστη. Στην περίπτωση της εφαρμογής μας, χρησιμοποιείται κατά κύριο λόγο κατά την υποβολή συνημμένης φωτογραφίας της ιατρικής συνταγής ή αρχείου τύπου PDF και από τον διαχειριστή για τις φωτογραφίες των προϊόντων, κατηγοριών και προσφορών. Όταν ανεβάζουμε ένα αρχείο, το Cloud Storage μας επιστρέφει την τοποθεσία όπου το αρχείο έχει αποθηκευτεί και στη συνέχεια αυτό το URL της φωτογραφίας εμείς το αποθηκεύουμε στο Firestore document του συγκεκριμένου προϊόντος ή παραγγελίας. Στο Σχήμα 2.5 μπορούμε να δούμε ένα παράδειγμα σε κώδικα της πιο πάνω διαδικασίας μέσω της μεθόδου uploadSpecialOffer() που χρησιμοποιεί ο διαχειριστής για να ανεβάσει νέα προσφορά. Μέσα στην uploadSpecialOffer() καλείτε η κλάση UploadTask η οποία είναι υπεύθυνη να ανεβάσει το αρχείο σε συγκεκριμένο σημείο του Cloud Storage που έχουμε δηλώσει. Όταν η ποιο πάνω διαδικασία τελειώσει επιτυχώς ανεβάζουμε το νέο document στο Firestore με τις πληροφορίες της προσφοράς αλλά και την URL διεύθυνση της εικόνας.

```
public void uploadSpecialOffer(SpecialOffers specialOffer,Uri url){
FirebaseStorage storage = FirebaseStorage.getInstance();
StorageReference storageRef = storage.getReference();
StorageReference imagesRef =
storageRef.child("SpecialOffers/"+UUID.randomUUID().toString());
UploadTask uploadTask = imagesRef.putFile(url);
uploadTask.addOnSuccessListener(taskSnapshot ->
imagesRef.getDownloadUrl().addOnSuccessListener(downloadUrl -> {
   SpecialOffers NewSpecialOffer = new SpecialOffers(specialOffer.getTitle(),
downloadUrl.toString(), specialOffer.getProductsCode());
   database.collection("SpecialOffers").add(NewSpecialOffer).addOnCompleteListener(task -
> {
       if (task.isSuccessful()) {
           _isSuccess.postValue(true);
       } else {
           _error.postValue(task.getException().getMessage());
       }
   });
}));
```

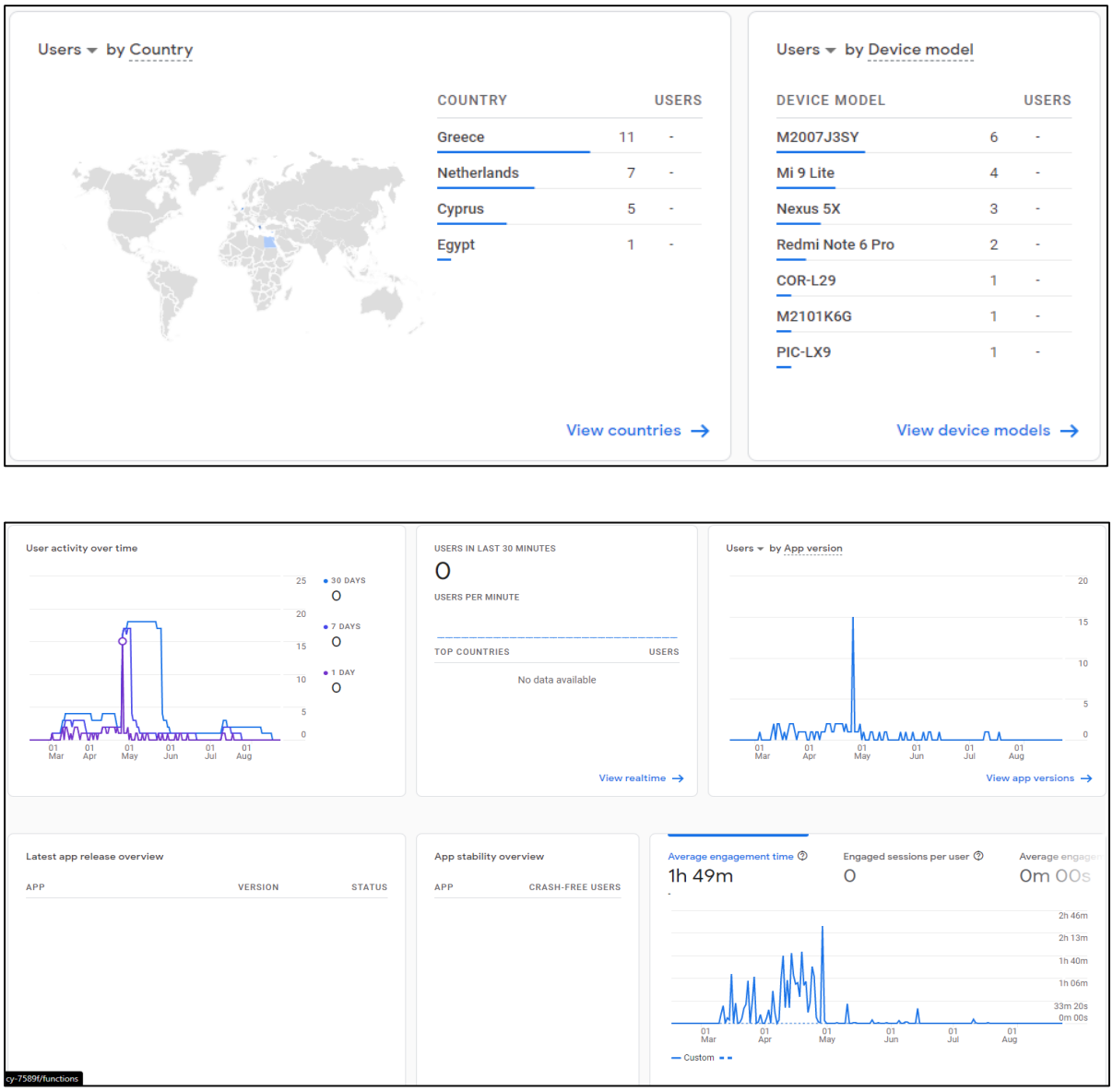
Σχήμα 2.5: Μέθοδος δημιουργίας και αποθήκευσης εγγράφου με ανέβασμα φωτογραφίας

2.5 Google Analytics

Με το Google Analytics στο Firebase, μπορούμε να καταγράψουμε συμβάντα στην εφαρμογή για να κατανοήσουμε το περιεχόμενο που παρακινεί το ενδιαφέρον του χρήστη και να προσδιορίσουμε ορισμένα χαρακτηριστικά που διακρίνουν διαφορετικούς τύπους χρηστών. Το Analytics παρέχει αυτές τις πληροφορίες, ώστε να μπορούν να αποστέλλονται στοχευμένες ειδοποιήσεις ή λειτουργίες σχετικά με τη βελτίωση της εφαρμογής. Τα στατιστικά στοιχεία καταγράφουν αυτόματα πολλές βασικές ιδιότητες ενός χρήστη, όπως γλώσσα, τοποθεσία, μοντέλο συσκευής και έκδοση λειτουργικού συστήματος όπως φαίνονται στο Σχήμα 2.7, ακόμα και διαφορετικά γεγονότα (Σχήμα 2.6).



Σχήμα 2.6: Παράδειγμα Analytics Events της εφαρμογής χρήστη



Σχήμα 2.7: Παράδειγμα Analytics της εφαρμογής χρήστη

2.6 Επίλογος

Στο πιο πάνω κεφάλαιο παρουσιάστηκε η Firebase οπού επιλέχθηκε σαν βάση για τις εφαρμογές μας και αποτέλεσε την BaaS των εφαρμογών μας. Αναφέρθηκαν μερικά από τα εργαλεία που μας παρέχει και μερικά μικρά κομμάτια κώδικα που χρησιμοποιήθηκαν στις εφαρμογές μας.

Κεφάλαιο 3ο: Σχεδιασμός Android

3.1 Εισαγωγή

Το Android είναι ένα λειτουργικό σύστημα που βασίζεται σε Linux και αναπτύχθηκε από την Google. Εκτός από φορητές συσκευές, τροφοδοτεί επίσης tablet, ρολόγια και ακόμη και ορισμένους τύπους αυτοκινήτων. Δεν υπάρχει λειτουργικό σύστημα smartphone που να είναι πιο δημοφιλή από το λειτουργικό σύστημα Android. Παρακάτω στο Σχήμα 3.1 επισημαίνεται το μερίδιο αγοράς του λειτουργικού συστήματος smartphone το 2022. Το λειτουργικό σύστημα Android της Google κυριαρχεί στην αγορά smartphone με μερίδιο αγοράς 70,97%, σύμφωνα με την Global Stats[5]. Έχουν υπάρξει πολλές ενημερώσεις Android που έχουν βελτιώσει σταδιακά το λειτουργικό σύστημα. Στην εφαρμογή έχουμε σαν ελάχιστη απαιτούμενη έκδοση το API 21: Android 5.0 (LOLLIPOP). Αυτό σημαίνει ότι το 98,4% των συσκευών Android που είναι ενεργές στο Google Play Store θα μπορούν να τρέξουν την εφαρμογή[6].

Πίνακας

S/N	Mobile OS	Market Share (%)
1	Android OS	70.94
2	iOS	28.29
3	Samsung	0.43
4	KaiOS	0.17
5	Unknown	0.1
6	Nokia Unknow	0.1

Σχήμα 3.1: Μερίδιο αγοράς του λειτουργικού συστήματος smartphone το 2022.

3.2 Android Studio

Μια εφαρμογή Android αναπτύσσεται στο Android studio, ένα σύγχρονο εργαλείο ανάπτυξης. Η Google διατηρεί αυτήν την εφαρμογή και βασίζεται στο IntelliJ IDEA IDE. Υπάρχουν πολλές δυνατότητες που παρέχονται από το Android studio, όπως πρότυπα κώδικα, εργαλεία Lint για ανάλυση κώδικα, πρόγραμμα επεξεργασίας WYSIWYG (What you see is what you get) για σχεδιασμό διάταξης κ.λπ.

3.3 JAVA

Η Java για πρώτη φορά κυκλοφόρησε το 1995 από την Sun Microsystems. Στο παρελθόν αποτελούσε την επίσημη γλώσσα για την ανάπτυξη Android εφαρμογών και πλέον έχει αντικατασταθεί από την Kotlin[7]. Οι περισσότερες εφαρμογές στο google play store είναι γραμμένες σε αυτήν. Λόγω της αντικειμενοστρέφειας της και άλλων χαρακτηριστικών όπως τα threads, exceptions, null pointers κ.τ.λ. αποτελεί μια πολύπλοκη γλώσσα για αρχάριους. Η Java δεν μεταγλωττίζει σε γλώσσα μηχανής αλλά μέσω του Javac δημιουργεί τα αρχεία σε Java bytecode, όπου αυτά τα αρχεία τρέχουν σε μια εικονική μηχανή. Με αυτό τον τρόπο οποιαδήποτε συσκευή ανεξαρτήτως αρχιτεκτονικής μπορεί να χρησιμοποιήσει την εικονική μηχανή που είναι ανεπτυγμένη για αυτήν και να τρέξει τα αρχεία Java bytecode στην ανάλογη γλώσσα μηχανής. Στην περίπτωση των Android, αρχικά μέχρι την εκδοχή του

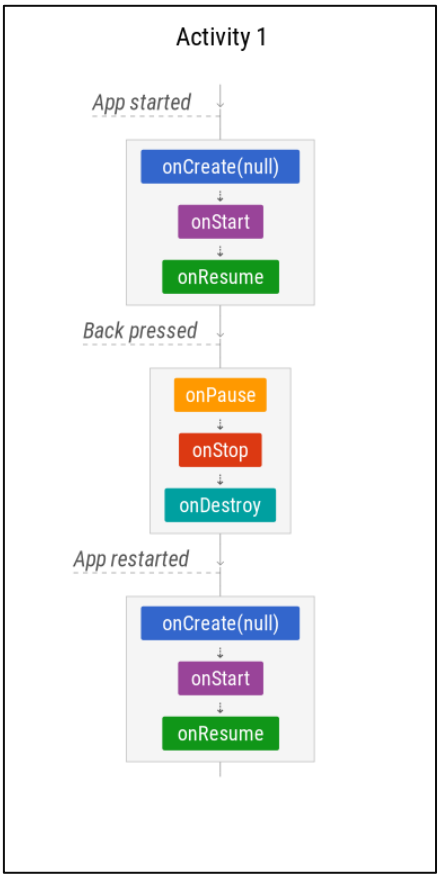
Android 4.4, η εικονική μηχανή που χρησιμοποιείται ήταν η Dalvik Virtual Machine(DVM), η οποία με την βοήθεια ενός εργαλείου μετέφραζε τα αρχεία από Java bytecode σε Dalvik bytecode και μετέπειτα το DVM σε γλώσσα μηχανής. Το 2014 η Google αντικατέστησε το Dalvik και κυκλοφόρησε το Android Runtime (ART) για το Android 5 το οποίο βελτίωσε την απόδοση και χρήσης μπαταρίας των εφαρμογών[8].

3.4 Gradle

Το Gradle είναι ένα εργαλείο που αυτοματοποιεί την διαδικασία δημιουργίας της εφαρμογής το οποίο είναι βασισμένο στο Groovy. Με την χρήση του Groovy το οποίο χρησιμοποιεί Domain Specific Language (DSL) μας δίνετε η δυνατότητα να χειριστούμε την λογική με την οποία θα δημιουργήσει η εφαρμογή[8]. Κάθε project όταν δημιουργείται, δημιουργούνται με αυτό δύο αρχεία build.Gradle. Το πρώτο αρχείο Gradle το συναντάμε στο root φάκελο όπου προσδιορίζει τον τρόπο κατασκευής για όλα τα modules. Ενώ το δεύτερο το βρίσκουμε στο φάκελο app, όπου αυτό το αρχείο είναι το σημείο όπου ορίζονται όλες οι εξαρτήσεις και όπου δηλώνονται οι εκδόσεις SDK[9]. Το πιο πάνω έχει πολλές λειτουργίες στην εφαρμογή όπως, το αναγνωριστικό της εφαρμογής (id), η έκδοση του SDK που χρησιμοποιεί, οι εξαρτήσεις από εξωτερικές βιβλιοθήκες, οι εκδόσεις του παρόντος λογισμικού.

3.5 Δραστηριότητες

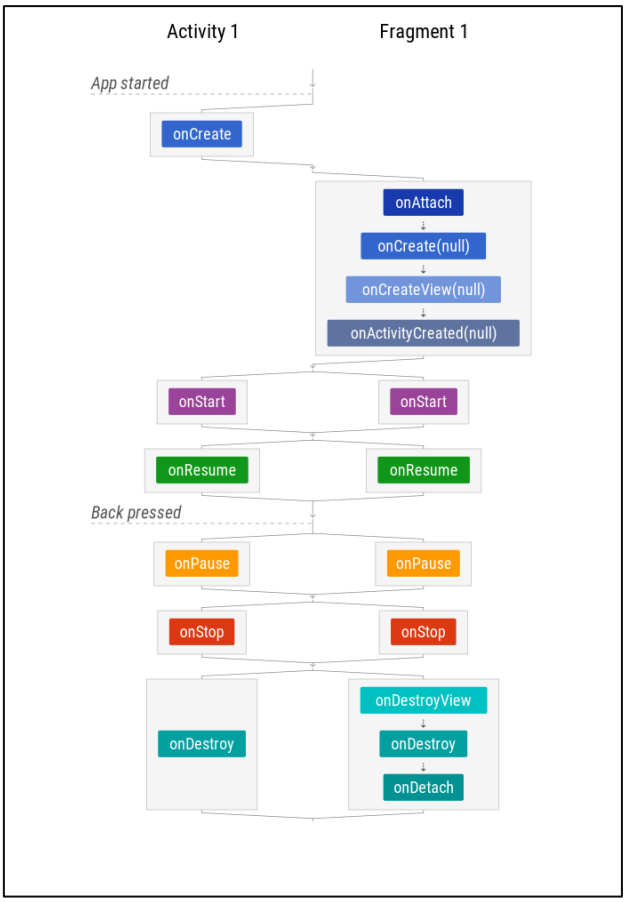
Τα δύο πιο σημαντικά δομικά στοιχεία μιας Android εφαρμογής είναι τα Activities και Fragments, τα οποία είναι θεμελιώδη μέρη μιας εφαρμογής. Αυτά είναι ουσιαστικά, αυτό που βλέπει ο χρήστης σε μια οθόνη και με το οποίο μπορεί να αλληλοεπιδράσει. Και τα δύο αυτά στοιχεία έχουν επίγνωση του κύκλου ζωής τους, πράγμα που σημαίνει ότι περνούν από πολλά συγκεκριμένα στάδια κατά τις αλλαγές στην εφαρμογή. Η κλάση Δραστηριότητα, παρέχει 6 μεθόδους επανάκλησης που επιτρέπουν στη δραστηριότητα να γνωρίζει ότι μια κατάσταση έχει αλλάξει. Αυτές οι μέθοδοι είναι η onCreate(), onStart(), onResume(), onPause(), onStop(), και onDestroy()[10]. Στο Σχήμα 3.2 , μπορούμε να δούμε ένα παράδειγμα κύκλου ζωής μιας Δραστηριότητας. Κάθε εφαρμογή χρειάζεται τουλάχιστον μια δραστηριότητα, όπου θα παρέχει το παράθυρο στο οποίο η εφαρμογή σχεδιάζει τη διεπαφή χρήστη της. Αυτό το παράθυρο συνήθως γεμίζει την οθόνη, αλλά μπορεί να είναι μικρότερο από την οθόνη και ουσιαστικά μια δραστηριότητα υλοποιεί μια οθόνη σε μια εφαρμογή. Ανοίγοντας την εφαρμογή φορτώνεται η κύρια δραστηριότητα(MainActivity). Καθώς ένας χρήστης πλοηγείται μέσα, έξω από και πίσω στην εφαρμογή σας, οι Δραστηριότητας στην εφαρμογή μεταβαίνουν σε διαφορετικές καταστάσεις στον κύκλο ζωής τους.



Σχήμα 3.2: Κύκλος ζωής μιας Δραστηριότητας σε επανεκκίνηση

3.6 Fragments

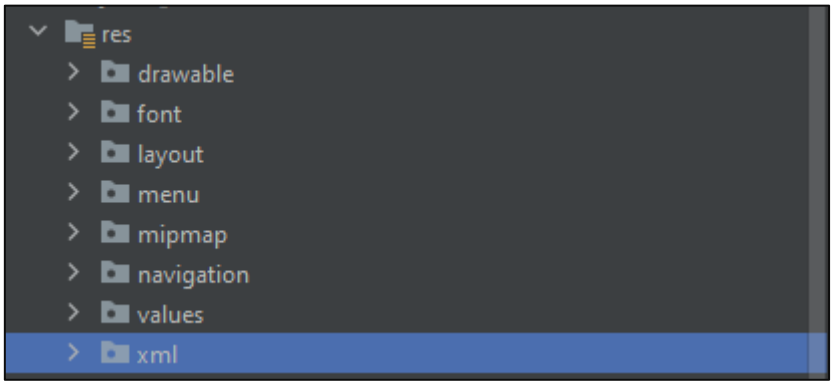
Τα Fragments μοιάζουν πολύ με τις Δραστηριότητες. Ωστόσο, ένα Fragment δεν μπορεί να ζήσει από μόνο του, πρέπει να "φιλοξενείται" σε ένα Activity ή σε ένα Fragment. Ένα Fragment αποτελεί ένα κομμάτι της διεπαφής χρήστη και περιέχει τον κώδικα που χρειάζεται για να κατασκευάσει το layout καθώς και να διαχειριστεί την αλληλεπίδραση του χρήστη με αυτήν. Ένα Fragment έχει και αυτό κύκλο ζωής σαν ένα Activity με τις δικές του μεθόδους επανάκλησης[11]. Στο Σχήμα 3.3 βλέπουμε τον κύκλο ζωής ενός Fragment να τρέχει παράλληλα με ένα κύκλο ζωής μιας Δραστηριότητας, αφού πρώτα δημιουργηθεί η Δραστηριότητα[12].



Σχήμα 3.3: Κύκλος ζωής μιας Δραστηριότητας μαζί με το Fragment της

3.7 Resources

Ο φάκελος πόρων σε κάθε project Android Studio επιτρέπει στον προγραμματιστή να οργανώνει αρχεία όπως εικόνες και συμβολοσειρές ξεχωριστά από τον πηγαίο κώδικα Java[13]. Με αυτό τον τρόπο δεν χρησιμοποιείται απευθείας αναφορά στα αρχεία, αλλά χρησιμοποιείται σχετική διαδρομή στον πηγαίο κώδικα και αυτό επιτρέπει στον προγραμματιστή να αλλάξει εύκολα όποια τιμή συμβολοσειράς επιθυμεί ή εικόνας, χωρίς να χρειάζεται να αναζητήσει και να αντικαταστήσει την συγκεκριμένη συμβολοσειρά ή εικόνα που επιθυμεί σε κάθε σημείο στον κώδικα ξεχωριστά (Σχήμα 3.4).



Σχήμα 3.4: Φάκελος Resouces από εφαρμογή χρήστη

3.8 Design Pattern

Η καλή αρχιτεκτονική λογισμικού είναι ένας από τους βασικούς παράγοντες που συμβάλλουν στην επιτυχία του λογισμικού. Υπάρχουν πολλά αρχιτεκτονικά μοτίβα που χρησιμοποιούνται στην ανάπτυξη Android. Μερικά από τα πιο γνωστά είναι το Model View Controller (MVC), Model View Presenter (MVP) και Model View ViewModel (MVVM). Επιλέγοντας μια σωστή αρχιτεκτονική λογισμικού για το πρόγραμμα μας, διασφαλίζουμε ότι αυτό θα είναι ευκολότερο να ελεγχθεί για τυχόν λάθη αλλά και να διορθωθούν ενώ ταυτόχρονα διασφαλίζει την ευκολότερη συντήρηση του. Τα τρία πιο πάνω έχουν σαν κοινό στόχο να αποσυνδέσουν την διεπαφή χρήστη από το μοντέλο της βάσης δεδομένων αλλά και την επιχειρηματική λογική όσο το δυνατόν περισσότερο.

3.8.1 MVC

Το MVC διαιρεί το σύστημα σε τρία στοιχεία, όπως ορίζει το όνομά του, μοντέλο (Model), προβολή (View) και ελεγκτής (Controller). Το μοντέλο (Model) είναι υπεύθυνο για το χειρισμό της λογικής του τομέα (πραγματικοί επιχειρηματικοί κανόνες) και την επικοινωνία με τη βάση δεδομένων και τα επίπεδα δικτύου[14]. Η προβολή (View) είναι το επίπεδο UI (User Interface) που περιέχει στοιχεία που είναι ορατά στην οθόνη. Επιπλέον, παρέχει την οπτικοποίηση των δεδομένων που είναι αποθηκευμένα στο Μοντέλο και προσφέρει αλληλεπίδραση στον χρήστη. Ο ελεγκτής (Controller) είναι το στοιχείο που καθορίζει τη σχέση μεταξύ της Προβολής και του Μοντέλου. Περιέχει τη βασική λογική της εφαρμογής και ενημερώνεται για τη συμπεριφορά του χρήστη και ενημερώνει το Μοντέλο ανάλογα με τις ανάγκες.

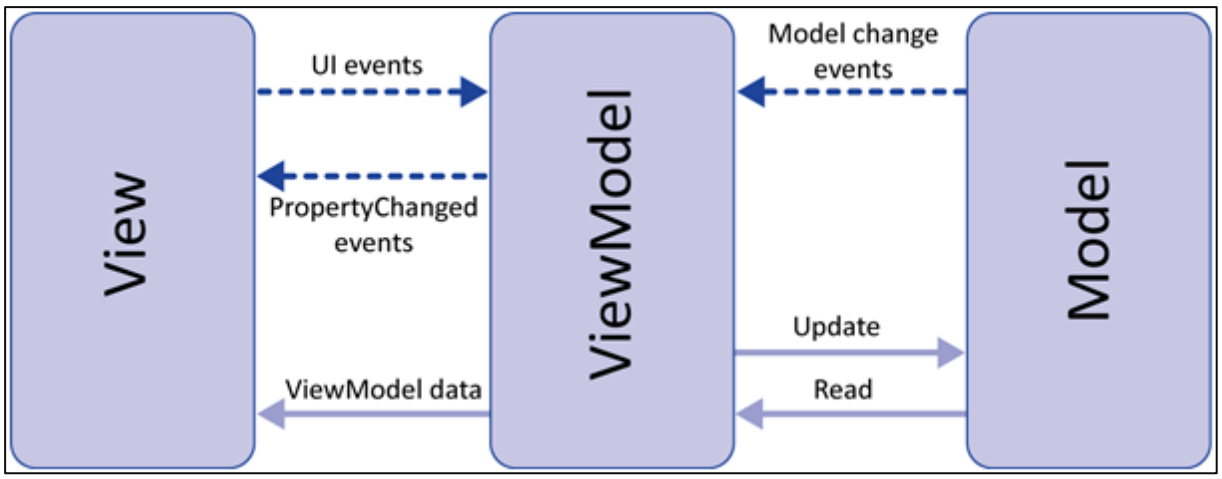
3.8.2 MVP

Το μοτίβο MVP όπως και το MVC διαιρεί το σύστημα σε τρία στοιχεία, όπως ορίζει το όνομά του, μοντέλο (Model), προβολή (View) και παρουσιαστής (Presenter). Το Μοντέλο και η προβολή έχουν τον ίδιο ρόλο όπως και στο MVC, το στοιχείο παρουσιαστή (Presenter) είναι το ίδιο με τον ελεγκτή στο MVC που στοχεύει να οργανώσει και να συντονίσει όλα τα ενδιάμεσα στοιχεία – αλληλεπίδραση, επιλογές και εντολές[15]. Η επικοινωνία προβολής (View) και του παρουσιαστή (Presenter) γίνεται με την χρήση διεπαφής (interface) όπου ο παρουσιαστής (Presenter) διαχειρίζεται μια προβολή (View) την φορά. Το ίδιο γίνεται και με την επικοινωνία του παρουσιαστή (Presenter) και του μοντέλου (Model). Σε αυτήν την αρχιτεκτονική το μοντέλο (Model) δεν επικοινωνεί ούτε αλληλοεπιδρά με κάποιο τρόπο με την προβολή (View)[16].

3.8.3 MVVM

Το Model — View — ViewModel (MVVM) είναι το πιο αναγνωρισμένο μοτίβο αρχιτεκτονικής λογισμικού στην ανάπτυξη εφαρμογών Android. Ξεπερνά όλα τα μειονεκτήματα των μοτίβων σχεδιασμού MVP και MVC[18]. Η αρχιτεκτονική MVVM έχει τρία στοιχεία, όπως δηλώνει το όνομά του, Model, View και view-model (Σχήμα 3.5). Το στοιχείο προβολής (View) εμφανίζει τη διεπαφή χρήστη της εφαρμογής. Το μοντέλο αντιπροσωπεύει τα δεδομένα, το ίδιο με το μοντέλο που εξηγείται στην προηγούμενη αρχιτεκτονική. Το View-Model, το οποίο σημαίνει μοντέλο προβολής, προορίζεται να διαχειριστεί την κατάσταση προβολής. Θα μεταβιβάσει τα δεδομένα και τις λειτουργίες για προβολή και επίσης θα διαχειριστεί τη λογική και τη συμπεριφορά της προβολής. Ένα καλό στοιχείο View-

Model δεν γνωρίζει την προβολή, αντίθετα μεταδίδει μόνο τις αλλαγές και οι μεμονωμένοι συνδρομητές θα χειρίζονται τα δεδομένα χωριστά. Αυτό σέβεται το κύκλο ζωής μεμονωμένων παρατηρητών. Επίσης, το MVVM υποστηρίζει αμφίδρομη σύνδεση δεδομένων μεταξύ των ιδιοτήτων προβολής και το μοντέλο, το οποίο οδηγεί σε άμεση ενημέρωση της διεπαφής χρήστη όταν τα δεδομένα έχουν αλλαγές. Για τους ποιο πάνω λόγους η ανάπτυξη και των δυο εφαρμογών έγινε με την χρήση του συγκεκριμένου αρχιτεκτονικού μοτίβου.



Σχήμα 3.5: MVVM μοτίβο αρχιτεκτονικής λογισμικού[17]

3.9 Επίλογος

Στο κεφάλαιο 3 αναφέρθηκαν κάποιες γενικές πληροφορίες για το Android λειτουργικό αλλά και βασικά στοιχεία που αποτελούν μια Android εφαρμογή. Τέλος, αναφέρθηκαν τα τρία σημαντικότερα μοντέλα αρχιτεκτονικής μιας Android εφαρμογής και ποιο επιλέχτηκε στη συγκεκριμένη Δ.Ε.

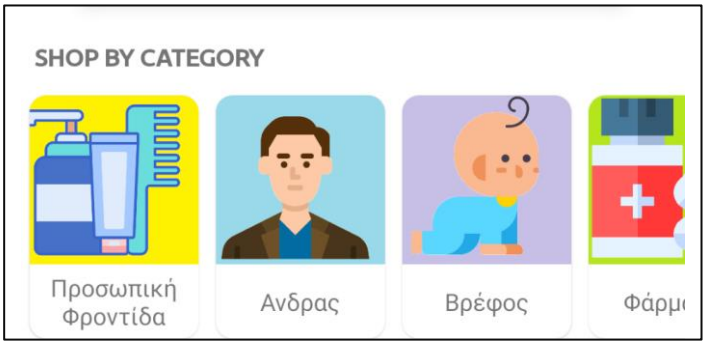
Κεφάλαιο 4ο: UI Design

4.1 Εισαγωγή

Το περιβάλλον εργασίας χρήστη (UI) πρέπει να είναι σαφές χωρίς περιττά αντικείμενα, αλλά το πιο σημαντικό, να είναι κατανοητό. Έχοντας αυτό κατά νου, το πρόγραμμα περιέχει μια πληθώρα απλών χρήσιμων γραφικών, ενώ έχουν παραληφθεί εξωτερικά στοιχεία που απλώς θα αποσπούσαν την προσοχή του χρήστη. Αυτό το κεφάλαιο παρέχει μια περιγραφή των στοιχείων που χρησιμοποιήθηκαν για τη δημιουργία διεπαφής της εφαρμογής.

4.2 RecyclerView

Το RecyclerView ανακυκλώνει στοιχεία όπου στην περίπτωση μας είναι ως συνήθως καρτέλες (cards). Οι καρτέλες είναι ένα ξεχωριστό layout όπου για παράδειγμα μπορεί να έχει τη φωτογραφία της κατηγορίας όπως στο Σχήμα 4.1. Όταν ο χρήστης κάνει κύλιση στην οθόνη μια καρτέλα μετακινείται από την οθόνη και το RecyclerView δεν καταστρέφει την προβολή του. Αντιθέτως, το RecyclerView επαναχρησιμοποιεί την προβολή και με την κύλιση στην οθόνη, τα νέα στοιχεία που υπάρχουν στο RecyclerView γίνονται ορατά στον χρήστη. Αυτή η επαναχρησιμοποίηση βελτιώνει σημαντικά την απόδοση, βελτιώνοντας την ανταπόκριση της εφαρμογής και μειώνοντας την κατανάλωση ενέργειας.



Σχήμα 4.1: RecyclerView κατηγορίες

4.3 TextView

Ένα TextView εμφανίζει στον χρήστη ένα κείμενο, ωστόσο η βασική κλάση έχει ρυθμιστεί ώστε να μην επιτρέπει την επεξεργασία.

4.4 EditText

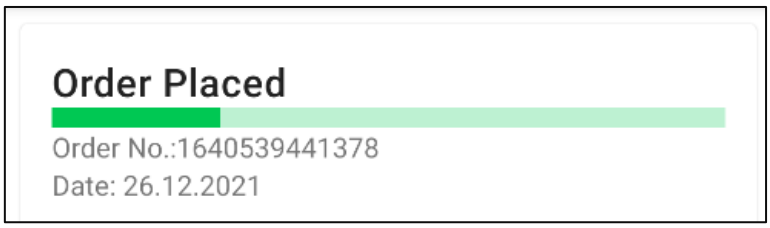
Σε αντίθεση με το TextView δίνει την δυνατότητα να διαμορφώνεται το κείμενο. Το EditText είναι μια υποκατηγορία του TextView με λειτουργίες επεξεργασίας κειμένου. Συχνά χρησιμοποιούμε το EditText στις εφαρμογές μας για να παρέχουμε ένα πεδίο εισαγωγής κειμένου.

4.5 Button

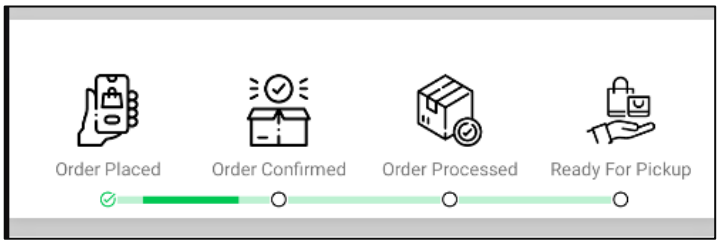
Τα κουμπιά επιτρέπουν στους χρήστες να κάνουν ενέργειες και να κάνουν επιλογές με ένα μόνο άγγιγμα.

4.6 Linear Progress Indicator

Τα LinearProgressIndicator ενημερώνουν τους χρήστες για την κατάσταση των διαδικασιών που βρίσκονται σε εξέλιξη, όπως η φόρτωση μιας εφαρμογής, η υποβολή μιας φόρμας ή η αποθήκευση ενημερώσεων. Στην εφαρμογή το χρησιμοποιούμε για να δείξουμε την πρόοδο μιας παραγγελίας με δυο διαφορετικούς (Σχήμα 4.2) (Σχήμα 4.3).



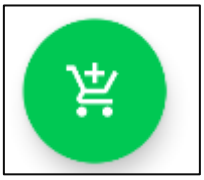
Σχήμα 4.2: Linear Progress Indicator καρτέλα παραγγελιών



Σχήμα 4.3: Linear Progress Indicator πληροφορίες παραγγελίας

4.7 Extended Floating Action Button

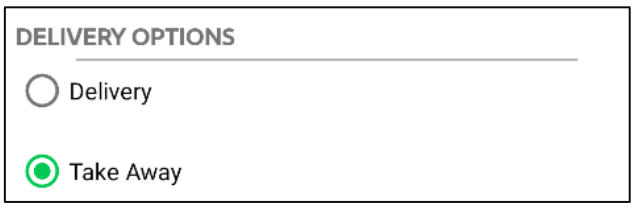
Ένα ExtendedFloatingActionButton εκτελεί την κύρια ενέργεια σε μια οθόνη π.χ. στην οθόνη που το έχουμε είναι στην προθήκη του προϊόντος, στο καλάθι του χρήστη (Σχήμα 4.4). Εμφανίζεται μπροστά από όλο το περιεχόμενο της οθόνης, συνήθως ως κυκλικό σχήμα με ένα εικονίδιο στο κέντρο του.



Σχήμα 4.4: ExtendedFloatingActionButton

4.8 RadioGroup / RadioButton

Στο RadioGroup, η επιλογή ενός RadioButton μεταξύ πολλών RadioButtons που έχουν προστεθεί σε RadioGroup, αυτόματα καταργεί όλες τις υπόλοιπες άλλες επιλογές. Δηλαδή σημαίνει ότι κάθε φορά μπορούμε να επιλέξουμε μόνο ένα κουμπί επιλογής από μια ομάδα RadioButtons που ανήκουν στην ίδια ομάδα RadioGroup (Σχήμα 4.5).



Σχήμα 4.5: RadioGroup με δυο RadioButtons

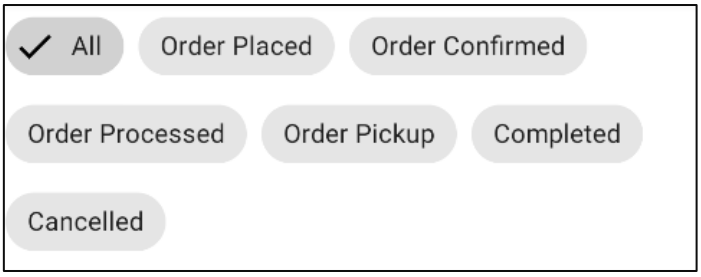
4.9 ImageView

Η κλάση imageView χρησιμοποιείται για την εμφάνιση οποιουδήποτε είδους εικόνας στην εφαρμογή Android, η οποία είτε μπορεί να είναι Bitmap είτε Drawable. Επιπλέον, προσφέρει την δυνατότητα να εφαρμόσουμε την απόχρωση σε μια εικόνα όπως επίσης και τον έλεγχο του μεγέθους και της κίνησης μιας εικόνας (Σχήμα 4.6).

Σχήμα 4.6: Χρήση imageView για εμφάνιση εικόνας προϊόντος

4.10 Chips

Ένα Chip αντιπροσωπεύει μια σύνθετη οντότητα σε ένα μικρό μπλοκ. Είναι ένα στρογγυλεμένο κουμπί που αποτελείται από μια ετικέτα, ένα προαιρετικό εικονίδιο και ένα προαιρετικό εικονίδιο κλεισίματος. Ένα τσιπ έχει πολλές χρήσεις, συγκεκριμένα στην περίπτωση μας όπως φαίνεται στο Σχήμα 4.7 η χρήση είναι σαν φίλτρο, όπου κάθε φορά που ο χρήστης επιλεγεί ένα από τα chips αυτό εφαρμόζει το ανάλογο φίλτρο.



Σχήμα 4.7: Chips για φίλτρο

4.11 Bottom Navigation Bar

Αυτό το στοιχείο περιέχει εικονίδια αλλά και κείμενο. Εμφανίζεται στο κάτω μέρος της οθόνης όπου επιτρέπει την πλοήγηση στις κύριες οθόνες της εφαρμογής, καθιστώντας την κάθε μια από αυτές προσβάσιμη από οποιοδήποτε σημείο στην εφαρμογή (Σχήμα 4.8).

Σχήμα 4.8: Navigation Bar τοποθετημένη στο κάτω μέρος της οθόνης

4.12 Επίλογος

Στο παρόν κεφάλαιο υπάρχουν με λεπτομέρεια τα στοιχεία διεπαφής χρήστη που έχουν επιλεχθεί για τις εφαρμογές. Επίσης, υπάρχουν λεπτομέρειες για το πως αυτά δουλεύουν αλλά και εικόνες με παραδείγματα από τα συγκεκριμένα στοιχεία στις εφαρμογές.

Κεφάλαιο 5ο: Απαιτήσεις

5.1 Εισαγωγή

Οι λειτουργικές απαιτήσεις περιγράφουν τα χαρακτηριστικά (λειτουργίες) της εφαρμογής[19]. Η λειτουργικότητα ενός συστήματος ορίζεται από τις απαιτήσεις του, οι οποίες καθορίζουν τι ακριβώς πρέπει να κάνει και πώς πρέπει να ανταποκρίνεται στις εντολές των χρηστών. Οι λειτουργικές απαιτήσεις καθορίζουν τους στόχους του λογισμικού, πράγμα που σημαίνει ότι εάν δεν εκπληρωθούν, το λογισμικό δεν θα λειτουργήσει. Οι μη λειτουργικές απαιτήσεις περιγράφουν τα χαρακτηριστικά ποιότητας ενός συστήματος λογισμικού[19].

5.2 Λειτουργικές Απαιτήσεις

5.2.1 Χρήστης

1. Εγγραφή νέου χρήστη

   Ο νέος χρήστης θα κάνει εγγραφή στη φόρμα εγγραφής, η οποία θα αποτελείται από μια οθόνη που θα περιέχει 5 πεδία όνομα, επίθετο, διεύθυνση ηλεκτρονικού ταχυδρομείου, κωδικό και αριθμού τηλεφώνου.

2. Αναζήτηση προϊόντος

   Μέσω της εφαρμογής θα δίνετε στον χρήστη η δυνατότητα να αναζητήσει ανάμεσα σε όλα τα προϊόντα που υπάρχουν στο κατάστημα.

3. Προσθήκη προϊόντος στο καλάθι

   Επιλέγοντας μια καρτέλα προϊόντος ο χρήστης θα έχει τη δυνατότητα να προσθέσει το προϊόν στο καλάθι.

4. Προβολή καλαθιού

   Στο καλάθι θα μπορούμε να δούμε τα προϊόντα που έχουμε προσθέσει σε αυτό το σύνολο της αξίας τους ή να αφαιρέσουμε ένα προϊόν από αυτό.

5. Ολοκλήρωση αγοράς

   Με την ολοκλήρωση της επιλογής των προϊόντων του ο χρήστης θα καταχωρεί την παραγγελία, συμπληρώνοντας όλα τα απαιτούμενα στοιχεία για αυτήν και θα λαμβάνει ένα μοναδικό αριθμό παραγγελίας.

6. Διαχείριση Διευθύνσεων

   Ο χρήστης θα έχει τη δυνατότητα να αποθηκεύει και να διαχειρίζεται πολλές διευθύνσεις όπου θα θέλει να γίνετε η αποστολή της παραγγελίας του.

7. Αλλαγή προσωπικών στοιχείων και κωδικού πρόσβασης

   Η εφαρμογή θα δίνει τη δυνατότητα στο χρήστη να επεξεργάζεται τα προσωπικά του στοιχεία (Όνομα, Επίθετο, τηλέφωνο, ηλεκτρονικό ταχυδρομείο) και να αλλάζει των κωδικό πρόσβασης του.

8. Ανάκτηση κωδικού πρόσβασης

   Σε περίπτωση που ο χρήστης έχει ξεχάσει των κωδικό του θα έχει τη δυνατότητα να ανακτήσει την πρόσβαση του χρησιμοποιώντας το ηλεκτρονικό του ταχυδρομείο.

9. Αποστολή Συνταγογράφησης

   Η εφαρμογή δίνει τη δυνατότητα στον χρήστη να αποστείλει μια συνταγογράφηση φαρμάκων δίνοντας του τρεις επιλογές

   - Αποστέλλοντας τον αριθμό E-Prescription
   - Αποστέλλοντας φωτογραφία της συνταγής
   - Αποστέλλοντας αρχείο τύπου PDF

10. Παρακολούθηση Παραγγελιών

    Ο χρήστης θα έχει τη δυνατότητα να βλέπει την πρόοδο της παραγγελίας του καθώς και παλιότερες παραγγελίες

5.2.2 Διαχειριστής

1. Προσθήκη προϊόντος

   Η εφαρμογή δίνει τη δυνατότητα στον διαχειριστή να προσθέσει ένα νέο προϊόν, να το επεξεργαστεί ή ακόμα και να το διαγράψει.

2. Δημιουργία κατηγορίας

   Ο διαχειριστής θα έχει τη δυνατότητα να προσθέσει μια νέα κατηγορία ή να τη διαγράψει.

3. Δημιουργία Προσφοράς

   Η εφαρμογή θα δίνει τη δυνατότητα στον διαχειριστή να προσθέσει μια νέα προσφορά, να την επεξεργαστεί και να την καταργήσει.

4. Παρακολούθηση Λίστας Παραγγελιών

   Ο διαχειριστής θα έχει πρόσβαση στις παραγγελίες και θα μπορεί να χρησιμοποιεί φίλτρο ώστε να εμφανίζονται ανάλογα με την κατάσταση παραγγελίας που βρίσκεται.

5. Αναζήτηση Παραγγελίας

   Η εφαρμογή θα δίνει τη δυνατότητα στο διαχειριστή να αναζητήσει μια παραγγελία χρησιμοποιώντας τον αριθμό παραγγελίας.

6. Ενημέρωση Παραγγελιών

   Στις παραγγελίες θα υπάρχει η δυνατότητα να επεξεργάζεται η κατάσταση τους, έτσι ώστε ο διαχειριστής να ενημερώνει την κατάσταση της παραγγελίας ανάλογα με το σε ποιο σημείο βρίσκετε.

5.3 Μη Λειτουργικές Απαιτήσεις

1. Η εφαρμογή πρέπει να παρέχει ασφάλεια στα δεδομένα του χρήστη και ταυτοποίηση του.
2. Και οι δυο εφαρμογές να είναι λειτουργικές πάντα.
3. Η εφαρμογή να παρέχει απλό και εύκολα κατανοητό προς τον χρήστη περιβάλλον διεπαφής.
4. Οι εφαρμογές να έχουν άμεση επικοινωνία με τη βάση και να ενημερώνονται για τυχόν αλλαγές που συμβαίνουν.
5. Οι εφαρμογές να μπορούν να τρέξουν σε πάνω από το 95% των συσκευών Android.

5.4 Επίλογος

Οι λειτουργικές απαιτήσεις και μη λειτουργικές, περιγράφονται και αναλύονται σε αυτό το κεφάλαιο. Αποτελούν τις βασικές λειτουργίες και απαιτήσεις που θα προσπαθήσουμε να καλύψουμε μέσα στις δυο εφαρμογές.

Κεφάλαιο 6ο: Λειτουργίες της εφαρμογής

6.1 Εισαγωγή

Στο παρακάτω κεφάλαιο θα δούμε τις οθόνες που έχουν οι 2 εφαρμογές. Πάνω από κάθε οθόνη περιγράφεται ο σκοπός της κάθε οθόνης και ποια λειτουργική απαίτηση εξυπηρετεί, αλλά και μερικά λόγια που περιγράφουν την κάθε οθόνη.

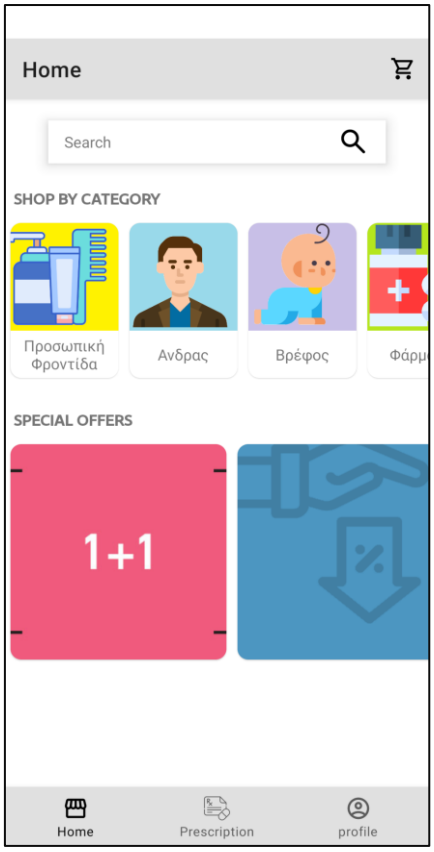
6.2 Απλός Χρήστης

Ένας απλός χρήστης ανοίγοντας την εφαρμογή το πρώτο πράγμα που βλέπει είναι την κεντρική οθόνη της εφαρμογής και δεν υπάρχει η ανάγκη να κάνει είσοδο σε λογαριασμό ή εγγραφή. Εγγραφή χρειάζεται μόνο όταν προσπαθήσει να τοποθετήσει οτιδήποτε στο καλάθι ή να προβεί σε παραγγελία. Η εφαρμογή χωρίζεται σε 3 κύριες οθόνες:

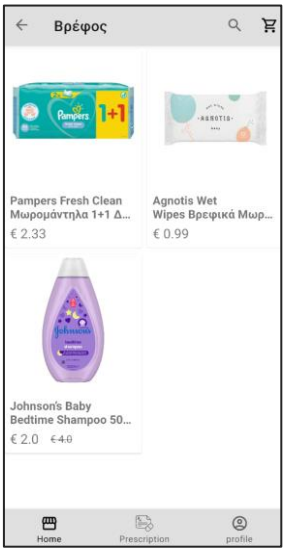
- Το ηλεκτρονικό κατάστημα
- Τη φόρμα αποστολής συνταγογραφημένης παραγγελίας
- Τη φόρμα προφίλ

6.2.1 Κεντρική Οθόνη

Στην κεντρική οθόνη όπως φαίνεται στο Σχήμα 6.1 ο χρήστης έχει τη δυνατότητα να πλοηγηθεί στο κατάστημα, να κάνει αναζήτηση για ένα προϊόν ή να ψάξει τις κατηγορίες (Σχήμα 6.2) ή και τις προσφορές.



Σχήμα 6.1: Κεντρική οθόνη εφαρμογής χρήστη

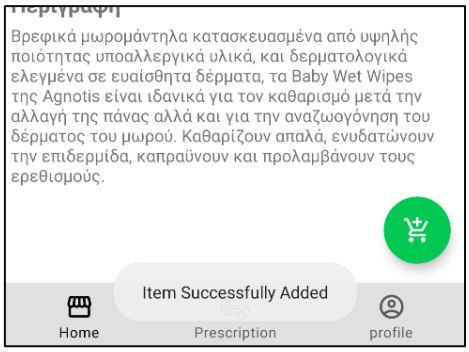


Σχήμα 6.2: Εμφάνιση προϊόντων κατηγορίας “Βρέφος”

6.2.2 Προσθήκη στο Καλάθι

Επιλέγοντας μια καρτέλα ενός προϊόντος ο χρήστης μπορεί να δει τη φωτογραφία του προϊόντος και τη περιγραφή (Σχήμα 6.3). Κάτω δεξιά υπάρχει ένα ExtendedFloatingActionButton όπου μας δίνει την δυνατότητα να προσθέσουμε το προϊόν στο καλάθι (Σχήμα 6.4).

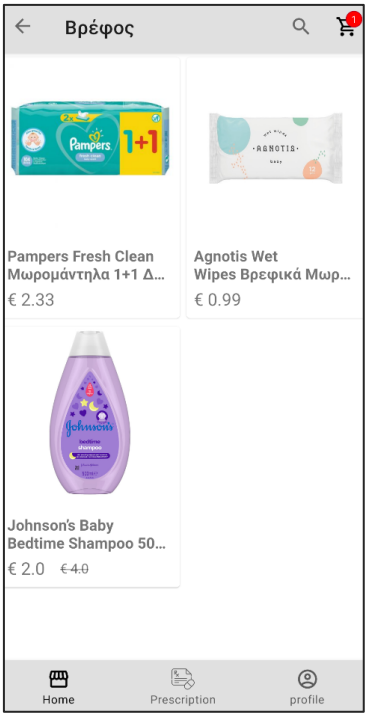
Σχήμα 6.3: Οθόνη λεπτομερειών για προϊών



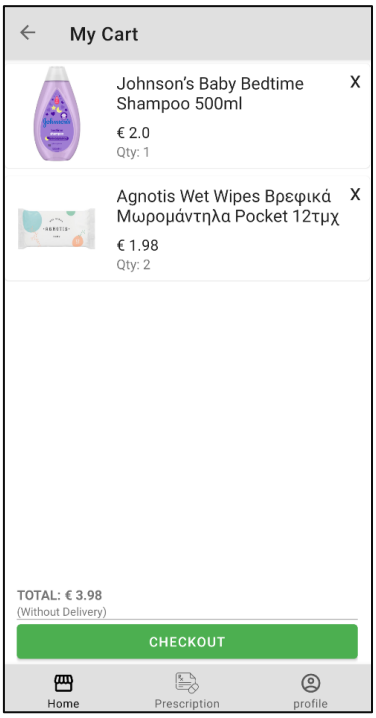
Σχήμα 6.4: Επιβεβαίωση προσθήκης προϊόντος στο καλάθι

6.2.3 Καλάθι

Όπως μπορούμε να δούμε στο Σχήμα 6.5 ο αριθμός των αντικειμένων αναγράφεται στο εικονίδιο του καλαθιού πάνω δεξιά. Πατώντας στο καλάθι, οδηγούμαστε σε αυτό όπου μπορούμε να δούμε τα προϊόντα που υπάρχουν σε αυτό, την ποσότητα τους αλλά και το συνολικό ποσό αξίας τους. Αφαιρώντας οποιοδήποτε αντικείμενο μπορούμε να δούμε ότι το ποσό ενημερώνεται αυτόματα αλλά και ο αριθμός των αντικειμένων που αναγράφεται στο καλάθι (Σχήμα 6.6).



Σχήμα 6.5: Αριθμός προϊόντων που βρίσκονται στο καλάθι

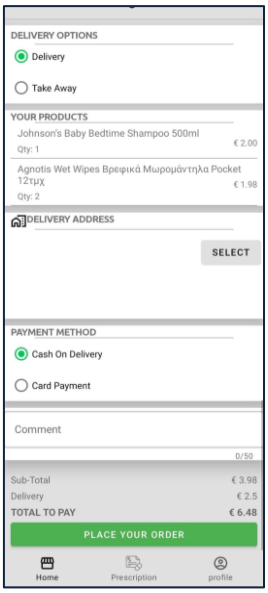


Σχήμα 6.6: Προϊόντα που βρίσκονται στο καλάθι

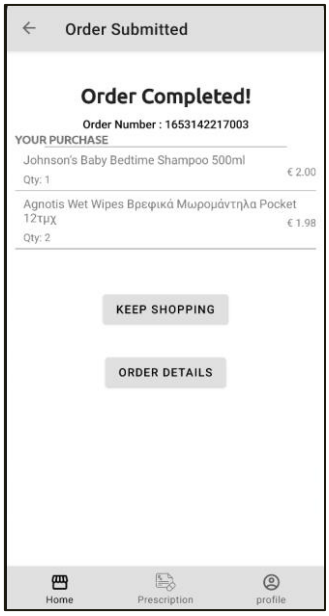
6.2.4 Checkout

Για να ολοκληρωθεί μια παραγγελία στο καλάθι πιέζουμε το κουμπί “CHECKOUT” όπου μας μεταφέρει σε μια νέα φόρμα για να ολοκληρώσουμε την παραγγελία μας. Σε αυτή την φόρμα επιλέγουμε τις λεπτομέρειες της παραγγελίας όπως το αν θα είναι με αποστολή ή παραλαβή από το κατάστημα, τη διεύθυνση αποστολής εάν πρόκειται για αποστολή, τον τρόπο πληρωμής και προαιρετικά κάποιο σχόλιο (Σχήμα 6.7).

Όταν συμπληρώσουμε όλα τα απαραίτητα στοιχεία πιέζουμε το “PLACE YOUR ORDER” που βρίσκεται στο κάτω μέρος της οθόνης και η παραγγελία καταγράφεται και μεταφερόμαστε σε μια νέα οθόνη που επιβεβαιώνει την παραγγελία και μας δίνει τον αριθμό παραγγελίας (Σχήμα 6.8).



Σχήμα 6.7: Φόρμα ολοκλήρωσης της παραγγελία μας

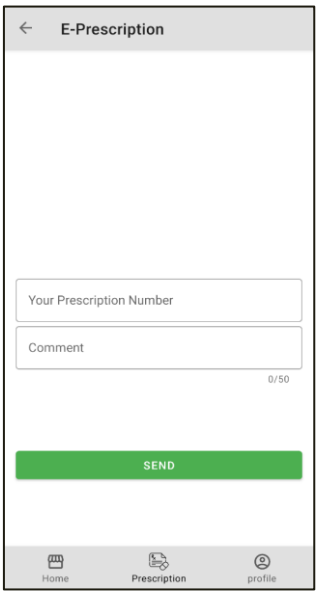


Σχήμα 6.8: Οθόνη επιβεβαίωσης παραγγελίας

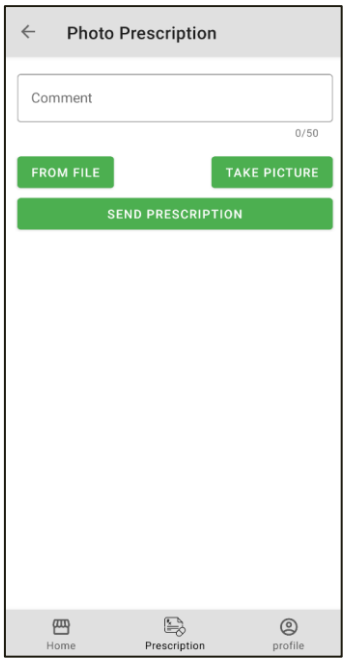
6.2.5 Αποστολή συνταγογραφούμενης παραγγελίας

Σε περίπτωση που θέλουμε να αποστείλουμε στο φαρμακείο μια συνταγογραφούμενη παραγγελία επιλέγουμε το μεσαίο εικονίδιο από την Navigation Bar στο κάτω μέρος της οθόνης. Αυτομάτως εμφανίζεται η παρακάτω οθόνη όπως βλέπουμε στο Σχήμα 6.9 . Σε αυτή την οθόνη μας δίνονται 3 επιλογές, η μια είναι η αποστολή του αριθμού της ηλεκτρονικής συνταγής (Σχήμα 6.10), την αποστολή μέσω φωτογραφίας (Σχήμα 6.11) και η τελευταία μέσω αποστολής αρχείου PDF (Σχήμα 6.12). Στη συνέχεια υπάρχει η επιλογή αναγραφής κάποιου σχολίου προς το φαρμακείο και να προστεθεί η συνταγή είτε μέσω φωτογραφίας είτε μέσω φόρτωσης από το αρχείο.

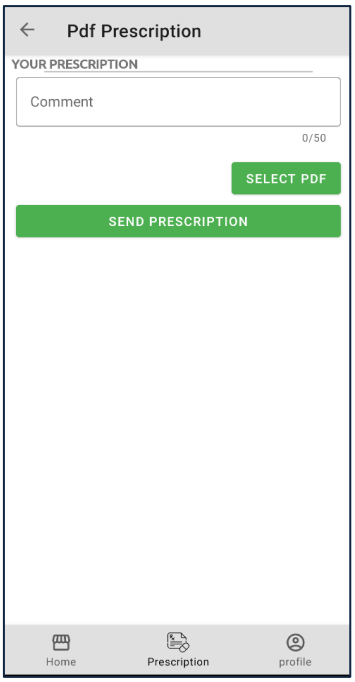
Σχήμα 6.9: Οθόνη επιλογής τύπου για συνταγογραφούμενη παραγγελία



Σχήμα 6.10: Οθόνη αποστολής αριθμού ηλεκτρονικής συνταγής



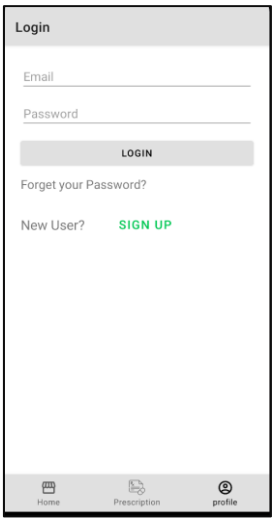
Σχήμα 6.11: Οθόνη αποστολής φωτογραφία συνταγής



Σχήμα 6.12: Οθόνη αποστολής συνταγής αρχείου PDF

6.2.6 Login

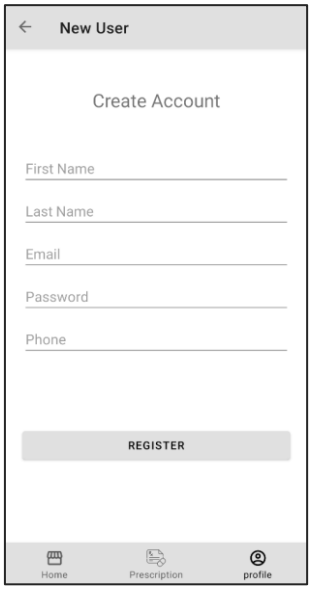
Για να προβεί ο χρήστης σε είσοδο στο λογαριασμό του, πιέζει το εικονίδιο κάτω δεξιά το οποίο αντιστοιχεί στην φόρμα πληροφοριών του χρήστη. Εάν ο χρήστης δεν είναι συνδεδεμένος, τότε εμφανίζεται η φόρμα εισόδου στο λογαριασμό (login) όπου καλείται να συμπληρώσει τα πεδία email και password και πατώντας το κουμπί “LOGIN” για να πιστοποιηθεί ότι πρόκειται για έγκυρο χρήστη (Σχήμα 6.13).



Σχήμα 6.13: Οθόνη εισόδου χρήστη στον λογαριασμό

6.2.7 Sign Up

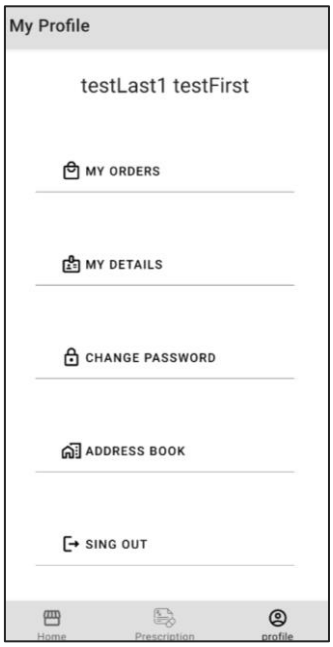
Στην περίπτωση που ο χρήστης δεν έχει ακόμα λογαριασμό στην εφαρμογή, πρέπει να κάνει εγγραφή πιέζοντας το κουμπί “SIGNUP” και αυτομάτως οδηγείται στη φόρμα εγγραφής, η φόρμα αποτελείται από μια μόνο οθόνη που φαίνεται στο Σχήμα 6.14. Η οθόνη περιέχει 5 πεδία όνομα, επίθετο, διεύθυνση ηλεκτρονικού ταχυδρομείου, κωδικό και αριθμό τηλεφώνου. Το κάθε πεδίο έχει τους απαραίτητους ελέγχους για να διασφαλιστεί η σωστή εγγραφή. Για παράδειγμα, στο πεδίο email ελέγχετε η εγκυρότητα της ηλεκτρονικής διεύθυνσης και αν ανήκει ήδη σε κάποιον άλλον λογαριασμό και στον κωδικό να περιέχονται τουλάχιστον 6 χαρακτήρες.



Σχήμα 6.14: Οθόνη εγγράφεις νέου χρήστη

6.2.8 My Profile

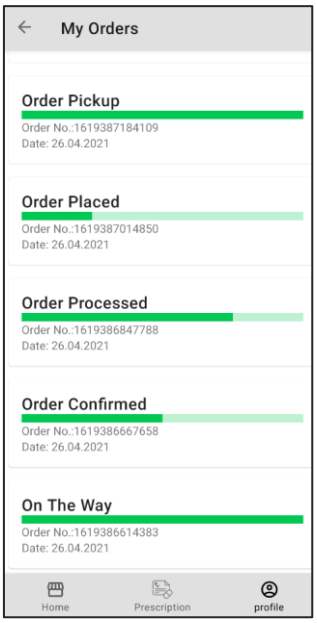
Εφόσον η είσοδος είναι έγκυρη από εδώ και πέρα, και καθώς ο χρήστης είναι συνδεδεμένος, η οθόνη προφίλ θα είναι όπως φαίνεται στο πιο κάτω (Σχήμα 6.15), δίνοντας την δυνατότητα στον χρήστη να δει τις παραγγελίες του, να επεξεργαστεί τα στοιχεία του, να αλλάξει κωδικό εισόδου και να επεξεργαστεί τη διεύθυνση του.



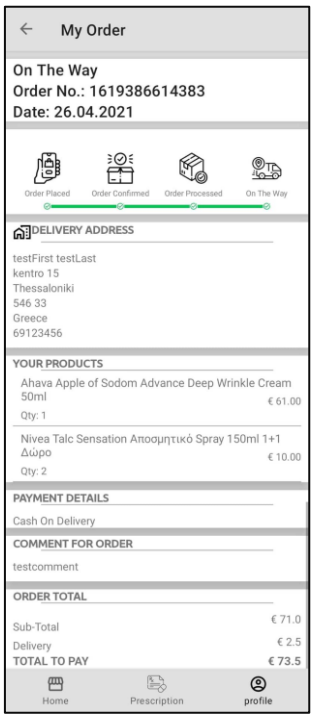
Σχήμα 6.15: Οθόνη προφίλ του χρήστη

6.2.9 My Orders

Ο χρήστης έχει την δυνατότητα να δει όλες τις παραγγελίες του και την κατάσταση που βρίσκονται με φθίνουσα σειρά, βάση της ημερομηνίας που πραγματοποιήθηκε η παραγγελία (Σχήμα 6.16). Στο κάτω μέρος της κάθε μιας από τις κάρτες παραγγελίας ένα πράσινο κουτί τύπου ProgressBar αντικατοπτρίζει την κατάσταση στην οποία βρίσκεται η παραγγελία. Επιλέγοντας μια καρτέλα παραγγελίας (Σχήμα 6.17), εμφανίζεται η φόρμα με τις λεπτομέρειες της παραγγελίας. Στο πάνω μέρος της φόρμας με λόγια εμφανίζεται η κατάσταση παραγγελίας, ο αριθμός παραγγελίας και η ημερομηνία που πραγματοποιήθηκε η παραγγελία. Έπειτα μπορούμε να δούμε ένα σχήμα με την κατάσταση της παραγγελίας. Στη συνέχεια εμφανίζεται η διεύθυνση αποστολής που ο χρήστης έχει δηλώσει, τα προϊόντα με την ποσότητα και την τιμή τους, η μέθοδος πληρωμής, το σχόλιο που ίσως άφησε ο χρήστης και τέλος το συνολικό ποσό της παραγγελίας.



Σχήμα 6.16: Λίστα παραγγελιών του χρήστη



Σχήμα 6.17: Πληροφορίες παραγγελίας

Η εμφάνιση της οθόνης παραγγελίας από φωτογραφία συνταγής διαφέρει από την οθόνη παραγγελίας προϊόντων. Όπως μπορούμε να δούμε στο πιο κάτω Σχήμα 6.18, την θέση της διεύθυνσης αποστολής και των προϊόντων αλλά και της μεθόδου πληρωμής παίρνει ένα πλαίσιο με την φωτογραφία που έχουμε αποστείλει.

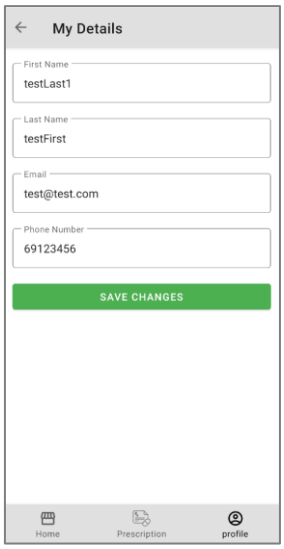
Σχήμα 6.18: Πληροφορίες παραγγελίας με φωτογραφία συνταγής

Η οθόνης παραγγελίας από συνταγή μέσω αρχείου PDF είναι όμοια με αυτήν της συνταγής από φωτογραφία (Σχήμα 6.19), με την διαφορά ότι στην θέση της φωτογραφίας έχουμε ένα εικονίδιο με την λέξη PDF, όταν ο χρήστης πιέσει το εικονίδιο άμεσα κατεβάζει το αρχείο PDF που είχε αποστείλει.

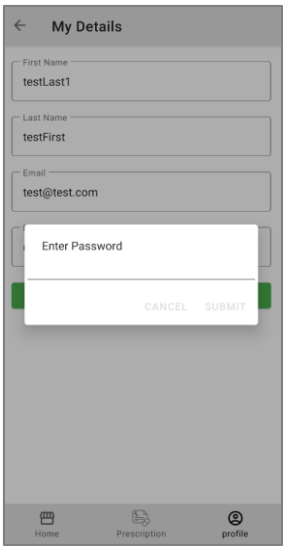
Σχήμα 6.19: Πληροφορίες παραγγελίας με αρχείο PDF συνταγής

6.2.10 My Details

Σε αυτή την φόρμα δίνετε η δυνατότητα στο χρήστη να δει και να τροποποιήσει τα προσωπικά του δεδομένα όπως το όνομα του, το επίθετο του, το email του αλλά και τον τηλεφωνικό του αριθμό (Σχήμα 6.20). Για λόγους ασφάλειας πριν την αποθήκευση των αλλαγών, ο χρήστης είναι υποχρεωμένος να καταχωρήσει τον κωδικό του σε ένα AlertDialog Box για επαλήθευση όπως μπορούμε να δούμε στο Σχήμα 6.21.



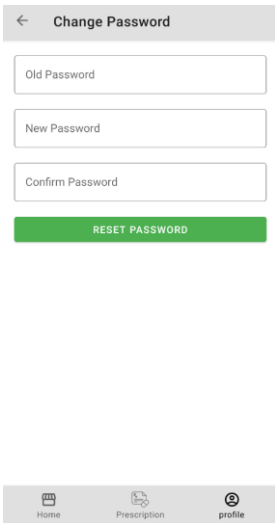
Σχήμα 6.20: Οθόνη τροποποιήσεις πληροφοριών χρήστη



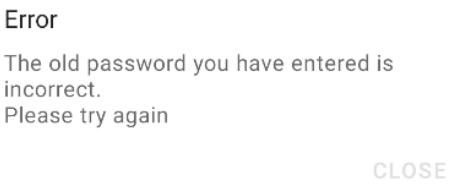
Σχήμα 6.21: Alert Dialog Box για επαλήθευση κωδικού πρόσβασης

6.2.11 Change Password

Στο χρήστη οποιαδήποτε στιγμή δίνετε η δυνατότητα να αλλάξει τον κωδικό πρόσβασης του. Για λόγους ασφάλειας πρέπει αρχικά να πληκτρολογήσει τον παλιό κωδικό και στη συνέχεια τον καινούριο δυο φορές όπου και συγκρίνονται μεταξύ τους για να αποφευχθούν τυχών λάθη από την πλευρά του χρήστη (Σχήμα 6.22). Στο Σχήμα 6.23 μπορούμε να δούμε το μήνυμα λάθους που εμφανίζετε εάν καταχώρηση λάθος κωδικό ο χρήστης.



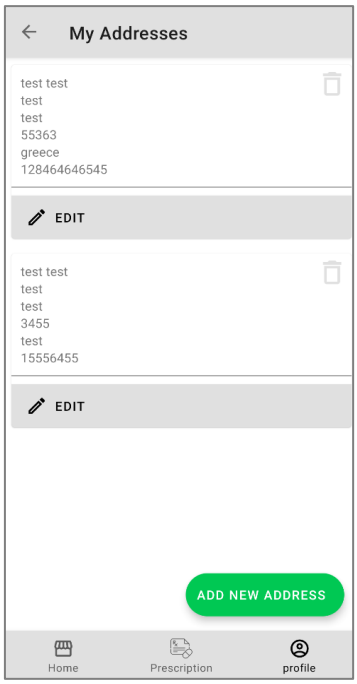
Σχήμα 6.22: Οθόνη αλλαγής κωδικού πρόσβασης



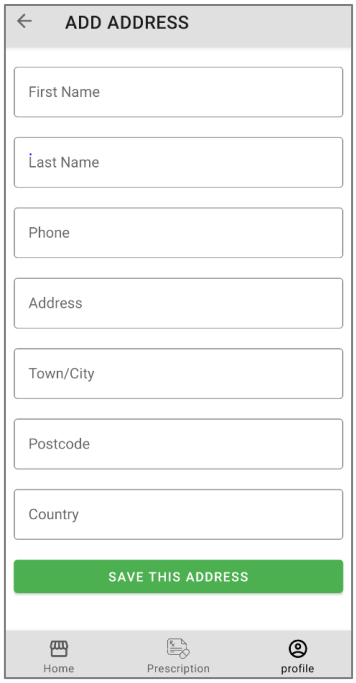
Σχήμα 6.23: Μήνυμα λάθος κωδικού

6.2.12 Address book

Η εφαρμογή δίνει στον χρήστη τη δυνατότητα να αποθηκεύει πολλές διαφορετικές διευθύνσεις και τις διαχειρίζεται από το συγκεκριμένο σημείο. Με το κουμπί “ADD NEW ADDRESS” προσθέτει μια νέα διεύθυνση και μέσω ενός RecycleViewer μπορούμε να τις δούμε αλλά και να κάνουμε αλλαγές σε αυτές και ακόμα να τις διαγράψουμε (Σχήμα 6.24). Στο Σχήμα 6.25 βλέπουμε τη φόρμα καταχώρησης μιας νέας διεύθυνσης που θέλει ο χρήστης.



Σχήμα 6.24: Λίστα αποθηκευμένων διευθύνσεων του χρήστη



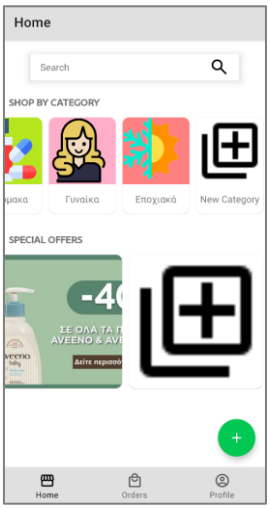
Σχήμα 6.25: Οθόνη αποθήκευσες νέας διεύθυνσης

6.2.13 Sign Out

Και τέλος, το Sign Out δίνει την δυνατότητα στον χρήστη να αποσυνδεθεί από τον λογαριασμό του.

6.3 Διαχειριστής

Όπως και στην εφαρμογή του χρήστη η αρχική οθόνη παραμένει σχεδόν η ίδια, έτσι ώστε ο διαχειριστής να μπορεί να έχει εικόνα του τι βλέπουν οι χρήστες της εφαρμογής. Οι αλλαγές που έχουν γίνει είναι η προσθήκη μιας νέας καρτέλας στο τέλος των δυο RecyclerView (Σχήμα 6.26) όπου δίνουν την δυνατότητα στον διαχειριστή να προσθέτει νέα κατηγορία ή προσφορά. Επίσης, έχει προστεθεί ένα επιπλέον κουμπί στο κάτω μέρος αριστερά που επιτρέπει την δημιουργία ενός νέου προϊόντος.

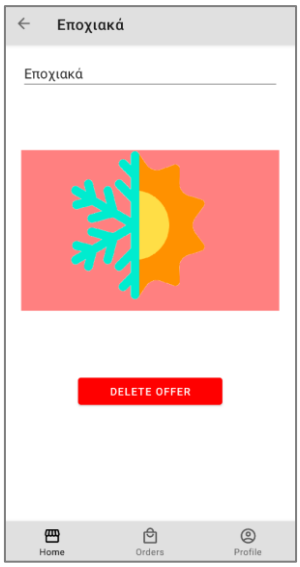


Σχήμα 6.26: Κεντρική οθόνη εφαρμογής διαχειριστή

6.3.1 Προσθήκη κατηγορίας ή διαγραφή κατηγορίας

Ο Διαχειριστής μπορεί να προσθέσει νέα κατηγορία προσθέτοντας το όνομα της και μια φωτογραφία (Σχήμα 6.27). Από την άλλη πιέζοντας σε μια ήδη υπάρχουσα κατηγορία μπορεί να διαγράψει μια κατηγορία (Σχήμα 6.28).

Σχήμα 6.27: Οθόνη δημιουργίας νέας κατηγορίας



Σχήμα 6.28: Οθόνη διαγραφής υπάρχον κατηγορίας

6.3.2 Δημιουργία προσφοράς ή επεξεργασία ήδη υπάρχουσας

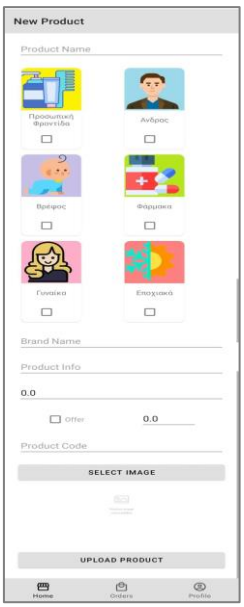
Η δημιουργία μιας νέας προσφοράς απαιτεί από το διαχειριστή να επιλέξει όλα τα προϊόντα που ανήκουν σε αυτή. Για αυτό τον λόγο σε σύγκριση με την δημιουργία νέας κατηγορίας στην φόρμα έχουμε προσθέσει ένα RecycleViewer που επιτρέπει στο χρήστη να επιλέξει τα προϊόντα που υπάρχουν σε αυτή (Σχήμα 6.29). Στην περίπτωση της επεξεργασίας μιας προσφοράς, τα προϊόντα που είναι ήδη στην προσφορά εμφανίζονται επιλεγμένα και ο διαχειριστής έχει την δυνατότητα να προσθέσει ή να αφαιρέσει προϊόντα.

Σχήμα 6.29: Οθόνη επεξεργασίας προσφοράς

6.3.3 Προσθήκη προϊόντος

Ο διαχειριστής έχει τη δυνατότητα να καταχωρήσει ένα νέο προϊόν στο κατάστημα. Για την καταχώρηση του προϊόντος πρέπει να εισάγει το όνομα του προϊόντος, να επιλέξει τις κατηγορίες στις

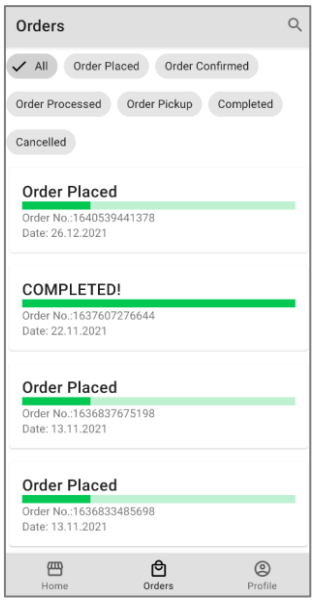
οποίες ανήκει, την μάρκα, πληροφορίες του προϊόντος, την τιμή, τον κωδικό του προϊόντος, την φωτογραφία και τέλος δίνεται η επιλογή στον διαχειριστή να προσθέσει μια νέα τιμή στο προϊόν αν αυτό βρίσκεται σε προσφορά. Αν ο διαχειριστής επιλέξει ένα ήδη υπάρχων προϊόν μπορεί να το επεξεργαστεί ή ακόμη και να το διαγράψει (Σχήμα 6.30).



Σχήμα 6.30: Οθόνη προσθήκης νέου προϊόντος

6.3.4 Orders

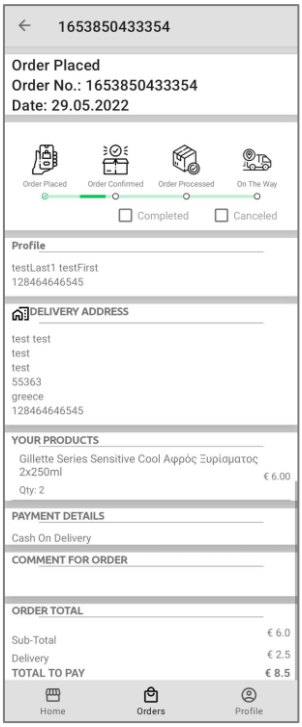
Επιλέγοντας το μεσαίο εικονίδιο από το Navigation Bar ο διαχειριστής βλέπει όλες τις παραγγελίες που έχουν γίνει στο κατάστημα του με φθίνοντα αριθμό ημερομηνίας που έχει καταχωρηθεί η παραγγελία. Επίσης, στην κορυφή της οθόνης έχει την δυνατότητα να αναζητήσει οποιαδήποτε παραγγελία βάσει του αριθμού παραγγελίας ή να ενεργοποιήσει ένα από τα φίλτρα που βλέπουμε σαν Chips όπου επιλέγουμε την κατάσταση της παραγγελίας (Σχήμα 6.31).



Σχήμα 6.31: Λίστα παραγγελιών

6.3.5 Order Info

Επιλέγοντας μια από τις παραγγελίες ο διαχειριστής μπορεί να δει λεπτομέρειες για την παραγγελία. Πιέζοντας σε ένα από τα τέσσερα εικονίδια του σχήματος που δείχνουν το στάδιο παραγγελίας, η παραγγελία αυτομάτως ενημερώνεται. Επίσης, δυο checkbox δίνουν την δυνατότητα στο διαχειριστή να ακυρώσει μια παραγγελία ή να την βάλει σε κατάσταση ολοκλήρωσης (Σχήμα 6.32).



Σχήμα 6.32: Λεπτομέρειες παραγγελίας και επεξεργασία

6.4 Επίλογος

Στο κεφάλαιο αυτό εμφανίστηκαν με λεπτομέρειες όλες οι οθόνες των δυο εφαρμογών μαζί με μια μικρή περιγραφή του τρόπου λειτουργίας τους και το που εξυπηρετεί η κάθε μια ξεχωριστά.

Κεφάλαιο 7ο: Υλοποίηση

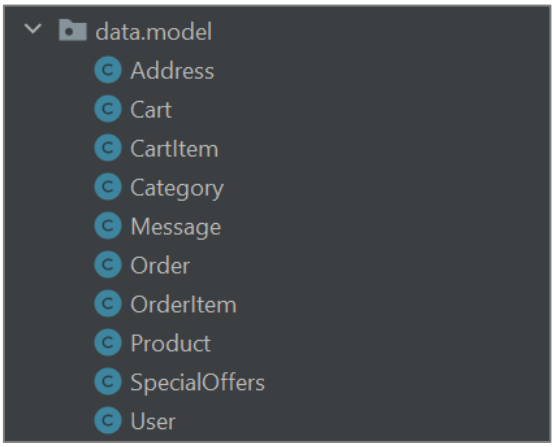
7.1 Εισαγωγή

Στο παρακάτω κεφάλαιο θα παρακολουθήσουμε την δομή και τα στοιχεία αρχιτεκτονικής που χρησιμοποιήθηκαν για την τελική ανάπτυξη της εφαρμογής. Σε γενικές γραμμές η εφαρμογή πραγματοποιήθηκε με στοιχεία αρχιτεκτονικής Android όπως το LiveData, το ViewBinding και με Navigation. Επίσης, σε αυτό το κεφάλαιο, θα δοθούν απαντήσεις στο πώς ενημερώνεται η διεπαφή χρήστη από την βάση δεδομένων και άλλες σημαντικές λεπτομέρειες.

7.2 Δομή εφαρμογής χρήστη

7.2.1 Μοντέλα

Στο πρώτο πακέτο συναντάμε τα μοντέλα, το συγκεκριμένο πακέτο περιέχει 10 ξεχωριστές κλάσεις, όπου το κάθε μοντέλο αντιπροσωπεύει ένα έγγραφο (document) από την βάση δεδομένων μας (Σχήμα 7.1).



Σχήμα 7.1: Πακέτο μοντέλων

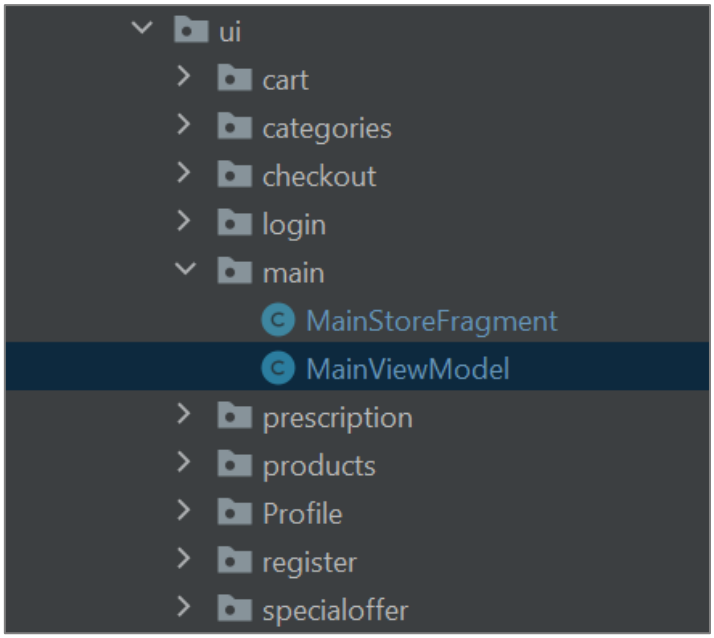
Στο ποιο κάτω Σχήμα 7.2, μπορούμε να δούμε το περιεχόμενο μιας μικρής κλάσης μοντέλο το SpecialOffers. Σε αυτή την κλάση υπάρχουν μόνο 3 μεταβλητές; ο τίτλος της προσφοράς, η διεύθυνση της φωτογραφίας που αντιστοιχεί στην προσφορά και ένας πίνακας με τους κωδικούς των προϊόντων που ανήκουν σε αυτή. Όπως μπορούμε να δούμε δεν υπάρχει κάποια λογική μέσα σε αυτές τις κλάσεις, παρά μόνο οι Constructor methods και οι Getter methods.

```java
public class SpecialOffers {
    private String title;
    private String photoUrl;
    private ArrayList<String> productsCode;

    public SpecialOffers() {
    }

    public SpecialOffers(String title, String photoUrl, ArrayList<String> productsCode) {
        this.title = title;
        this.photoUrl = photoUrl;
        this.productsCode = productsCode;
    }

    public String getTitle() { return title; }

    public String getPhotoUrl() { return photoUrl; }

    public ArrayList<String> getProductsCode() { return productsCode; }

}
```

Σχήμα 7.2: Κλάση SpecialOffers

7.2.2 Πακέτα Διεπαφών Χρήστη

Σε αυτό το πακέτο περιέχονται χωρισμένα σε υποπακέτα όλες οι κλάσεις χωρισμένες αναλόγως της διεπαφής. Στο κάθε πακέτο μπορούμε να βρούμε το λιγότερο δύο κλάσεις, μια κλάση όπου περιέχονται οι μέθοδοί δημιουργίας της κάθε διεπαφής και οι μέθοδοί παρατηρητών, ενώ στην άλλη κλάση ViewModel βρίσκονται οι μέθοδοί που περιέχουν λογική, και μεταβιβάζει τα δεδομένα και τις λειτουργίες για προβολής (Σχήμα 7.3).



Σχήμα 7.3: Πακέτο με κλάσεις της κεντρικής οθόνης

7.2.3 Κλάση Προβολής

Πιο συγκεκριμένα στο πιο κάτω πακέτο Σχήμα 7.4, μπορούμε να δούμε τις δυο κλάσεις για την κύρια οθόνη όπου η κλάση MainStoreFragment είναι η κλάση προβολής(View), περιέχει όλες τις μεθόδους για την δημιουργία αλλά και μεθόδους που σχετίζονται με την διεπαφή του χρήστη. Στην MainStoreFragment μπορούμε να βρούμε την μέθοδο setupViews(), η οποία είναι υπεύθυνη για να ετοιμάσει τα RecyclerView των κατηγοριών αλλά και των προσφορών και επίσης να το τοποθετήσει ένα onClickListener στο στοιχείο searchBar.

```
private void setupViews() {
    categoriesAdapter = new CategoriesAdapter( listener: this);

    binding.catRecyclerView.setHasFixedSize(true);
    binding.catRecyclerView.setAdapter(categoriesAdapter);

    specialOffersAdapter = new SpecialOffersAdapter( listener: this);

    binding.specialRecycler.setHasFixedSize(true);
    binding.specialRecycler.setAdapter(specialOffersAdapter);

    binding.searchBar.setOnClickListener(search -> {
        Navigation.findNavController(binding.getRoot()).navigate(
                MainStoreFragmentDirections.actionMainStoreFragmentToProductFragment( categoryName: "Search",  offersProducts: null));
    });
}
```

Σχήμα 7.4: Μέθοδος setupViews() της κλάσης MainStoreFragment

Επίσης, σε αυτήν την κλάση όπως και στις άλλες κλάσεις διεπαφής συναντάμε την μέθοδο setupObservers() (Σχήμα 7.5) όπου σκοπό έχουν να παρακολουθούν τα LiveData από την ViewModel κλάση καθ' όλη την διάρκεια που το Fragment είναι ζωντανό και σε τυχόν αλλαγές να ενημερώνουν την διεπαφή αναλόγως.

```
private void setupObservers() {
    mainViewModel.categories.observe(getViewLifecycleOwner(), categorieslist -> {
        categoriesAdapter.setData(categorieslist);
        binding.catRecyclerView.setVisibility(View.VISIBLE);
    });
    mainViewModel.specialOffers.observe(getViewLifecycleOwner(), specialofferslist -> {
        specialOffersAdapter.setData(specialofferslist);
        binding.specialRecycler.setVisibility(View.VISIBLE);
    });

    cartViewModel.itemsCount.observe(getViewLifecycleOwner(), itemCount -> {
        cartItemsCount = itemCount;
        Activity activity = getActivity();

        if (activity==null) throw new AssertionError( detailMessage: "Problem on loading please try again");
        activity.invalidateOptionsMenu();

    });

    productViewModel.products.observe(getViewLifecycleOwner(), Products -> {

        if (mainViewModel.isLoggedIn()) {
            cartViewModel.setProducts(Products);
            cartViewModel.fetchCart();
        }
    });

    productViewModel.fetchProducts();
    mainViewModel.fetchCategories();
    mainViewModel.fetchSpecialOffers();
}
```

Σχήμα 7.5: Μέθοδος setupObservers() της κλάσης MainStoreFragment

7.2.4 Κλάση Μοντέλο Προβολής

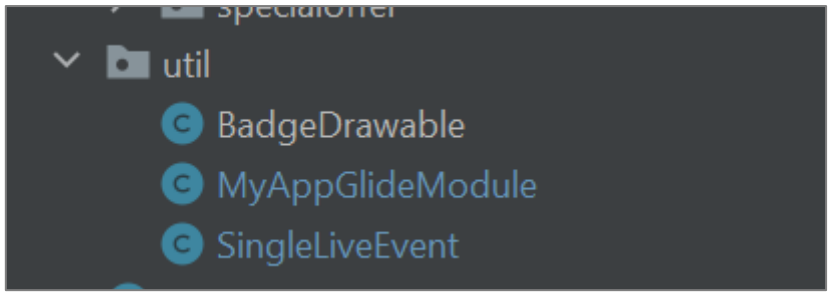
Στην κλάση μοντέλο προβολής MainViewModel περιέχονται οι μέθοδοι για την μεταβίβαση των δεδομένων στην κλάση διεπαφής(View) αλλά και σε κάποιες περιπτώσεις μέθοδοι με τη επιχειρηματική λογική. Στο ποιο κάτω Σχήμα 7.6 μπορούμε να δούμε την μέθοδο isLoggedIn() που ελέγχει αν ο χρήστης είναι συνδεδεμένος έτσι ώστε το MainStoreFragment να ενημερώνετε, αν θα πρέπει να εμφανίζεται στο καλάθι το σήμα με τα πόσα αντικείμενα περιέχονται σε αυτό. Επίσης, βλέπουμε την μέθοδο fetchCategories() όπου είναι υπεύθυνη να λαμβάνει από τη βάση τα documents στο collection Categories και να δημιουργεί με την βοήθεια του μοντέλου Categories μια LiveData λίστα με όλες τις κατηγορίες που υπάρχουν στην βάση.

```java
public boolean isLoggedIn() {
    if (auth.getCurrentUser() != null) {
        return true;
    } else {
        return false;
    }

}

public void fetchCategories() {
    database.collection( collectionPath: "Categories").addSnapshotListener((documentSnapshots, e) -> {
        if (e != null || documentSnapshots == null || documentSnapshots.getDocuments().size() == 0) {
            _error.postValue("Error or snapshot null");
            return;
        }

        List<Category> list = new ArrayList<>();

        for (DocumentSnapshot snapshot : documentSnapshots.getDocuments()) {
            Category category = snapshot.toObject(Category.class);

            if (category != null) {
                list.add(category);
            }
        }

        _categories.postValue(list);
        _isSuccess.postValue(true);
    });
}
```

Σχήμα 7.6: Μέθοδοι isLoggedIn() και fetchCategories() της κλάσης MainViewModel

7.2.5 Πακέτο util

Σε αυτό το πακέτο βρίσκουμε 3 κλάσεις που χρησιμοποιούνται σε διάφορες περιπτώσεις από πολλές κλάσεις όπως το SingleLiveEvent που περιέχει την μέθοδο που χρησιμοποιούμε για να συγκρίνουμε αν τα LiveData έχουν κάποια αλλαγή, ή ακόμα την BadgeDrawable όπου είναι υπεύθυνη να δημιουργεί το κόκκινο κύκλου στο εικονίδιο του καλαθιού όταν αυτό έχει αντικείμενα (Σχήμα 7.7).



Σχήμα 7.7: Πακέτο util

7.2.6 MainActivity

Η εφαρμογή αποτελείται από μόνο ένα Activity, όλες οι οθόνες είναι fragments τα οποία εμφανίζονται μέσα σε μια MainActivity στην οποία παρέχονται οι εντολές για την δημιουργία και την εγκατάσταση του Navigation της εφαρμογής (Σχήμα 7.8). Επίσης, υπάρχουν οι εντολές για την χρήση του binding στην εφαρμογή.

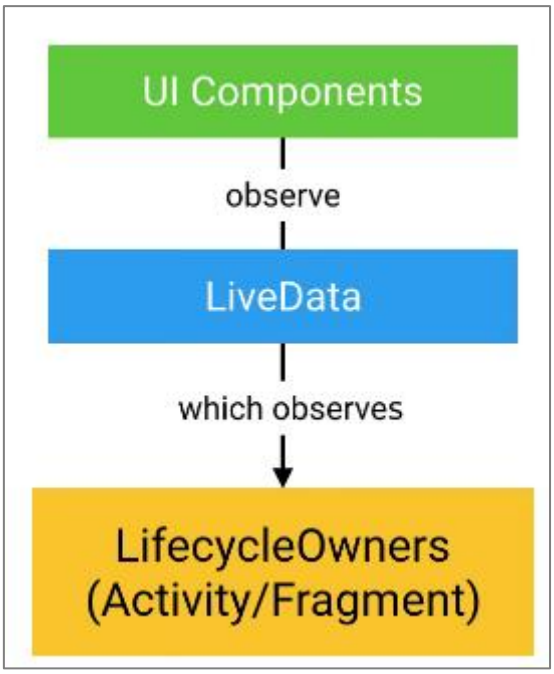
```java
public class MainActivity extends AppCompatActivity {


    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        ActivityMainBinding binding = ActivityMainBinding.inflate(getLayoutInflater());
        setContentView(binding.getRoot());

        AppCompatDelegate.setDefaultNightMode(AppCompatDelegate.MODE_NIGHT_NO);

        setSupportActionBar(binding.appToolbar);
        NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment);
        AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
                R.id.mainStoreFragment, R.id.prescriptionFragment, R.id.loginFragment, R.id.profileFragment).build();

        NavigationUI.setupWithNavController(binding.appToolbar, navController, appBarConfiguration);
        NavigationUI.setupWithNavController(binding.bottomNavigation, navController);
    }

}
```

Σχήμα 7.8: Μέθοδοι onCreate() της κλάσης MainActivity

7.3 Στοιχεία Αρχιτεκτονικής

7.3.1 LiveData

Μια κλάση LiveData είναι μια κλάση που κρατά δεδομένα. Παρόλα αυτά η κλάση LiveData έχει μια διαφορά από τις κλασικές κλάσεις που κρατούν δεδομένα, καθώς σέβεται τον κύκλο ζωής άλλων στοιχείων μιας εφαρμογής, όπως activity, fragments ή services[20]. Ως αποτέλεσμα, το LiveData ενημερώνει τους παρατηρητές στοιχείων εφαρμογής μόνο όταν βρίσκονται σε κατάσταση κύκλου ζωής (Σχήμα 7.9). Από προεπιλογή, το LiveData είναι αμετάβλητο. Τα δεδομένα μπορούν να παρατηρηθούν μόνο με τη χρήση του LiveData και δεν μπορούν να οριστούν. Για αυτό τον λόγο στην εφαρμογή χρησιμοποιείτε και η κλάση MutableLiveData, η οποία αποτελεί μια υποκλάση της κλάσης LiveData και μας επιτρέπει να παρατηρήσουμε και να ορίσουμε τις τιμές χρησιμοποιώντας μεθόδους postValue() και setValue(), έτσι ώστε να μπορούμε να αποστέλλουμε τιμές σε οποιονδήποτε ζωντανό ή ενεργό παρατηρητή. Στο Σχήμα 7.10, μπορούμε να δούμε πως μέσα από την κλάση OrdersViewModel, η μέθοδος fetchOrders() παίρνει τα στοιχεία που χρειάζεται από τη βάση και τα προσθέτει σε ένα αντικείμενο τύπου MutableLiveData όπου με τη σειρά του αυτό, τα μεταφέρει σε ένα αντικείμενο LiveData για να μπορεί να γίνεται παρακολούθηση (observe) στην κλάση του OrdersFragmnent.



Σχήμα 7.9: LiveData Architecture Component[21]

```
private final MutableLiveData<List<Order>> _orders = new MutableLiveData<>();
public LiveData<List<Order>> orders = _orders;


public void fetchOrders() {
    String time = "orderTimestamp";
    database.collection( collectionPath: "Orders").whereEqualTo( field: "userId", user.getUid()).orderBy(time, Query.Direction.DESCENDING)
            .addSnapshotListener((snapshots, e) -> {
        if (e != null || snapshots == null || snapshots.getDocuments().size() == 0) {
            return;
        }

        List<Order> list = new ArrayList<>();

        for (DocumentSnapshot snapshot : snapshots.getDocuments()) {
            Order item = snapshot.toObject(Order.class);

            if (item != null) {
                list.add(item);
            }
        }
        _orders.setValue(list);
    });
}
```

Σχήμα 7.10: Μέθοδος fetchOrders() της κλάσης OredersViewModel

7.3.2 View Binding

Η δυνατότητα View Binding διευκολύνει τη σύνταξη κώδικα που αλληλοεπιδρά με τις προβολές. Όταν η δυνατότητα View Binding είναι ενεργοποιημένη σε μια λειτουργική μονάδα, δημιουργεί μια κλάση δέσμευσης για κάθε αρχείο διάταξης XML που υπάρχει στη λειτουργική μονάδα[22]. Οι δεσμεύσεις περιέχουν άμεσες αναφορές σε όλες τις προβολές που έχουν αναγνωριστικά(ID) στη διάταξη στην

οποία ανήκουν. Με αυτό τον τρόπο ένα μεγάλο όπως μπορούμε να δούμε στο Σχήμα 7.11 μέρος του κώδικα αφαιρείται αποφεύγοντας την χρήση του findViewById.

```
public class RegisterFragment extends Fragment {
   private FragmentRegisterBinding binding;


   public View onCreateView(@NonNull LayoutInflater inflater, ViewGroup
container, Bundle savedInstanceState) {
       binding = FragmentRegisterBinding.inflate(inflater, container, false);
       viewModel = new
ViewModelProvider(this).get(LoginRegisterViewModel.class);

       return binding.getRoot();
   }
   private void setupViews() {
       binding.btnSingUp.setOnClickListener(view -> {
           String email = binding.txtEmail.getText().toString();
           String password = binding.txtPassword.getText().toString();
           String lastName = binding.textLastName.getText().toString();
           String firstName = binding.textFirstName.getText().toString();
           long phone = Long.parseLong(binding.textPhone.getText().toString());
       }
   }
```

Σχήμα 7.11: Παραδιγμα χρησης του ViewBinding στην μεθοδο setupViews() της κλασεις RegisterFragment

7.3.3 RecyclerView

Στο RecyclerView όπως έχει ειπωθεί πιο πάνω υπάρχει η δυνατότητα να προβληθούν αποτελεσματικά μεγάλα σύνολα δεδομένων. Παρέχει τα δεδομένα και καθορίζει πώς πρέπει να φαίνεται κάθε στοιχείο και η βιβλιοθήκη RecyclerView δημιουργεί τα στοιχεία δυναμικά όταν απαιτούνται. Επίσης, το πως πρέπει να φαίνονται αλλά και να αντιδρά το κάθε στοιχείο καθορίζεται από προσαρμοζόμενο Adapter που έχουμε δημιουργήσει. Στην παρούσα εργασία υπάρχουν συνολικά 6 διαφορετικές κλάσεις adapter, με την κάθε μια από αυτές να έχει το αντίστοιχο xml αρχείο οπού είναι υπεύθυνα για την εμφάνιση της κάθε κάρτας αντικειμένου στο κάθε RecyclerView. Οι 5 από αυτές είναι RecyclerView. Adapter ενώ η μια από αυτές είναι ListAdapter στην οποία έχουμε προσθέσει τη μέθοδο από RecyclerView.ViewHolder η οποία παρέχει την δυνατότητα στον Adaptor να επαναχρησιμοποιεί στοιχεία όπως το RecyclerView.

1. Η CartAdapter είναι η κλάση όπου έχει χρησιμοποιηθεί σαν ListAdapter, είναι υπεύθυνη για την εμφάνιση των προϊόντων που βρίσκονται στο καλάθι του χρήστη. Όταν ο observer που υπάρχει στην view κλάση του καλαθιού ενημερωθεί από την modelView κλάση με την λίστα από τα προϊόντα που υπάρχουν στο καλάθι αυτός στέλνει την λίστα στην adapter κλάση. Στη συνέχεια η λίστα περνά μέσα από μια κλάση DiffUtil.Callback όπου υπολογίζει αν υπάρχουν διαφορές στην νέα λίστα με την προηγουμένη αλλάζοντας μόνο τα στοιχεία που έχουν αλλάξει. Μέσω μιας Boolean μεταβλητής ελέγχει αν το προϊόν βρίσκεται σε προσφορά και εμφανίζει

την τιμή προσφοράς εάν όχι εμφανίζει την κανονική τιμή. Παράλληλα αντιστοιχίζει το όνομα του προϊόντος, την ποσότητα του συγκεκριμένου προϊόντος, την εικόνα αλλά και ένα clickListener ενός κουμπιού με εικονίδιο "Χ" όπου δίνει την δυνατότητα στον χρήστη να αφαιρέσει το προϊόν από το καλάθι του.

2. Η AddressAdaptor είναι υπεύθυνη για την εμφάνιση της διεύθυνσης που έχει ο χρήστης, παίρνοντας σαν παράμετρο όλες τις διευθύνσεις που αντιστοιχούν στο χρήστη μέσα σε μια λίστα. Μετέπειτα εμφανίζει το όνομα, την διεύθυνση, την πόλη, τον ταχυδρομικό κώδικα, χώρα και τηλεφωνικό αριθμό. Υπάρχει ένα clickListener ενός κουμπιού με εικονίδιο καλαθιού όπου δίνει την δυνατότητα στον χρήστη να αφαιρέσει την διεύθυνση, ένα για να τον μεταφέρει στην φόρμα επεξεργασίας της ύπαρχων διεύθυνσης και ακόμη ένα που αν ο χρήστης βρίσκεται στην διαδικασία checkout επιλέγετε η συγκεκριμένη διεύθυνση για την αποστολή.

3. Η CategoriesAdapter δέχεται μια λίστα με τις κατηγορίες και δημιουργεί καρτέλες με την κάθε κατηγορία με τον τίτλο και την εικόνα που της αντιστοιχεί (Σχήμα 7.12). Επίσης, περιέχεται ένας clickListener όπου όταν η καρτέλα πιεστεί, στέλνει μέσω του NavController τις πληροφορίες της συγκεκριμένης κατηγορίας στο επόμενο fragment (Σχήμα 7.13).

4. Η SpecialOfferAdapter λειτουργεί με τον ίδιο τρόπο όπως και η κλάση CategoriesAdapter, με την διαφορά ότι η λίστα που παίρνει η μεταβλητή αποτελείται από τις προσφορές. Έτσι, και αυτή η κλάση με την σειρά της διαθέτει ένα clickListener που αποστέλλει τις πληροφορίες στο επόμενο fragment.

5. Η ProductsAdapter κλάση δέχεται σε μορφή λίστας όλα τα προϊόντα που υπάρχουν σε κάθε καρτέλα. Μια Boolean μεταβλητή ελέγχει αν το προϊόν βρίσκεται σε προσφορά και εμφανίζει την τιμή προσφοράς, αλλά παράλληλα και την κανονική τιμή με μια διακριτική γραμμή η οποία θα διαγράφει την παλιά τιμή, εάν όχι εμφανίζει μόνο την κανονική τιμή. Παράλληλα εμφανίζει τον τίτλο του προϊόντος και την εικόνα. Ένας clickListener δίνει στον χρήστη την δυνατότητα, επιλέγοντας την καρτέλα να ανοίγει μια φόρμα με όλες τις πληροφορίες του προϊόντος. Στον συγκεκριμένο Adapter μαζί με την λίστα των προϊόντων στέλνονται και ακόμα 2 παράμετροι στην μέθοδο; ο τίτλος της κατηγορίας αν τα προϊόντα που θα εμφανιστούν θέλουμε να είναι από μια συγκεκριμένη κατηγορία ή μια λίστα από String όπου είναι τα productId των προϊόντων που αντιστοιχούν σε μια προσφορά. Για αυτό τον λόγο έχουμε 3 φίλτρα. Το ένα φίλτρο εμφανίζει μόνο τα προϊόντα της κατηγορίας εάν πρόκειται για εμφάνιση προϊόντων κατηγορίας, το δεύτερο το οποίο εμφανίζει τα προϊόντα που το productId τους περιέχεται στην λίστα με τα προϊόντα που ανήκουν στην συγκεκριμένη προσφορά που έχει επιλεχθεί και τέλος ένα τρίτο το οποίο χρησιμοποιείται για την αναζήτηση προϊόντων το οποίο καθώς πληκτρολογούμε κάτι στο κουτί αναζήτησης φιλτράρει παράλληλα όλα τα προϊόντα της λίστας για να βρει αυτά που ταιριάζουν στην αναζήτηση μας.

6. Η OrderAdapter κλάση δέχεται σε μορφή λίστας όλες τις παραγγελίες. Η λίστα αυτή έρχεται ταξινομημένη από την Firebase σε φθίνουσα σειρά βάση της ημερομηνίας υποβολής της. Στην

κάρτα κάθε προσφοράς στο RecyclerView εμφανίζεται η ημερομηνία που έχει γίνει η κάθε παραγγελία σε μορφή dd/mm/yyyy και ο αριθμός παραγγελίας. Με μεγάλα γράμματα αναγράφεται η κατάσταση κάθε παραγγελίας η οποία συνοδεύεται από μια progressbar που είναι συμπληρωμένη αναλόγως σε ποιο στάδιο βρίσκεται η παραγγελία.

```
public class CategoriesAdapter extends RecyclerView.Adapter<CategoriesAdapter.CategoriesAdapterViewHolder> {
    private List<Category> data;
    private CategoriesAdapter.CategoriesAdapterListener listener;

    public CategoriesAdapter(CategoriesAdapterListener listener) {

        this.listener = listener;
        data = new ArrayList<>();

    }

    public void setData(List<Category> category) {
        data.clear();
        data.addAll(category);

        notifyDataSetChanged();
    }

    @NonNull
    @Override
    public CategoriesAdapter.CategoriesAdapterViewHolder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        CategoryItemBinding binding = CategoryItemBinding.inflate(LayoutInflater.from(parent.getContext()), parent,  attachToParent: false);
        return new CategoriesAdapter.CategoriesAdapterViewHolder(binding);
    }

    @Override
    public void onBindViewHolder(@NonNull CategoriesAdapter.CategoriesAdapterViewHolder holder, int position) {
        holder.bind(data.get(position));
    }

    @Override
    public int getItemCount() { return data.size(); }
```

Σχήμα 7.12: Κλάση CategoriesAdapter

```
    class CategoriesAdapterViewHolder extends RecyclerView.ViewHolder {
        private final CategoryItemBinding binding;

        public CategoriesAdapterViewHolder(@NonNull CategoryItemBinding binding) {
            super(binding.getRoot());
            this.binding = binding;
        }

        public void bind(Category item) {
            binding.getRoot().setOnClickListener(v -> listener.onClicked(item));
            binding.titleTxt.setText(item.getTitle());
            Glide.with(binding.getRoot().getContext()) RequestManager
                    .load(item.getPhotoUrl()) RequestBuilder<Drawable>
                    .apply(new RequestOptions().placeholder(R.drawable.image_not_available).error(R.drawable.image_not_available))
                    .centerCrop()
                    .into(binding.categoryImg);
        }
    }

    public interface CategoriesAdapterListener {
        void onClicked(Category category);
    }
}
```

Σχήμα 7.13: Κλάση CategoriesAdapterViewHolder μέσα στην κλάση CategoriesAdapter

7.3.4 Glide

Για να είναι εφικτή η εμφάνιση των φωτογραφιών σε διαφορά σημεία του προγράμματος, έχει χρησιμοποιηθεί η Android βιβλιοθήκη Glide, η οποία παρέχει στον προγραμματιστή την δυνατότητα να ανακτήσει εικόνες, βίντεο και κινούμενα GIF και να τα εμφανίσει στην εφαρμογή του[23]. Στην εφαρμογή, χρησιμοποιείται ουσιαστικά για την εμφάνιση και την αλλαγή μεγέθους εικόνων που λαμβάνονται από το Firebase Storage, καθώς μπορούμε επίσης να προσθέσουμε μια εικόνα από resource όπου θα εμφανιστεί κατά την διάρκεια φόρτωσης είτε σε περίπτωση που η φόρτωση της εικόνας αποτύχει (Σχήμα 7.14).

```
Glide.with(binding.getRoot().getContext())
       .load(item.getPhotoUrl())
       .apply(new RequestOptions().placeholder(R.drawable.image_not_available)
       .error(R.drawable.image_not_available))
       .centerCrop()
       .into(binding.specialOfferImg);
```

Σχήμα 7.14: Χρήση μεθόδου glide

7.3.5 Navigation

Η πλοήγηση σε μια εφαρμογή (η μετακίνηση από τη μια οθόνη στην άλλη) είναι ένα απολύτως σημαντικό μέρος της ανάπτυξης μιας εφαρμογής Android. Όπως και οι συγγενευμένες δυο εφαρμογές αλλά και οι περισσότερες εφαρμογές Android περιλαμβάνουν πολλές οθόνες μέσω των οποίων ο χρήστης περιηγείται χρησιμοποιώντας κινήσεις οθόνης, κλικ κουμπιών και επιλογές μενού. Στις δυο εφαρμογές έχει χρησιμοποιηθεί το Android Jetpack Navigation Component οπού παρέχει μια αυτόματη διαδικασία κωδικοποίησης με εύκολη μέθοδο προβολής και οργάνωσης περίπλοκων διαδρομών πλοήγησης. Υπάρχουν τρία κύρια μέρη που συνεργάζονται αρμονικά για την επίτευξη των σκοπών του στοιχείου πλοήγησης (Navigation Component). Αυτά είναι το γράφημα πλοήγησης(Navigation graph), το NavHost και το NavController[24].

- Γράφημα πλοήγησης (Navigation graph): Αποτελείται από ένα μόνο αρχείο τύπου XML που περιέχει όλες τις πληροφορίες που σχετίζονται με την πλοήγηση. Ταυτόχρονα μέσω του XML έχουμε τη δυνατότητα να δούμε σε μορφή γραφήματος τις οθόνες και τις διαδρομές που μπορεί να ακολουθήσει ο χρήστης μέσα σε αυτήν (Σχήμα 7.15). Επίσης, μπορούμε να προσθέσουμε διάφορες παραμέτρους που θα αποσταλούν στην οθόνη προορισμού (Σχήμα 7.16).

Σχήμα 7.15: Το γράφημα πλοήγησης nav_graph.xml εφαρμογής διαχειριστή

```
<fragment
    android:id="@+id/specialOfferFragment"
    android:name="com.example.adminpharmacy.ui.specialoffer.addSpecialOffer.AddSpecialOfferFragment"
    android:label="{offerTitle}"
    tools:layout="@layout/fragment_add_special_offer">
    <argument
        android:name="offerTitle"
        app:argType="string"
        app:nullable="true" />
</fragment>
<fragment
    android:id="@+id/loginFragment"
    android:name="com.example.adminpharmacy.ui.login.LoginFragment"
    android:label="Login"
    tools:layout="@layout/fragment_login">

    <action
        android:id="@+id/action_loginFragment_to_profileFragment"
        app:destination="@id/profileFragment" />
</fragment>
```

Σχήμα 7.16: Το γράφημα πλοήγησης nav_graph.xml εφαρμογής διαχειριστή σε xml

- NavHost: Είναι ένα Layout ενός Fragment που χρησιμοποιείται με την προϋπόθεση ότι γίνετε πλοήγηση σε άλλα Fragments. Ουσιαστικά είναι ένα παράθυρο που τοποθετείται σε ένα Activity, εναλλάσσει και βγάζει διάφορους προορισμούς Fragments που περιλαμβάνονται στο γράφημα πλοήγησης (Navigation graph) (Σχήμα 7.17).

```
<fragment
    android:id="@+id/nav_host_fragment"
    android:name="androidx.navigation.fragment.NavHostFragment"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:defaultNavHost="true"
    app:layout_constraintBottom_toTopOf="@id/bottom_navigation"
    app:layout_constraintHorizontal_bias="0.0"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:layout_constraintTop_toBottomOf="@id/app_bar"
    app:layout_constraintVertical_bias="0.0"
    app:navGraph="@navigation/nav_graph" />
```

Σχήμα 7.17: activity_main.xml

- NavController: Είναι μια κλάση η οποία είναι γραμμένη σε Java/Kotlin και ακολουθεί πάντα ένα NavHost. Αυτό είναι που δίνει στην πραγματικότητα την εντολή να πραγματοποιηθεί η πλοήγηση.

Το στοιχείο Navigation περιλαμβάνει μια κλάση NavigationUI. Αυτή η κλάση περιέχει στατικές μεθόδους που διαχειρίζονται την πλοήγηση με την επάνω γραμμή της εφαρμογής, το navigation drawer και την κάτω γραμμή πλοήγηση (Σχήμα 7.18).

```
NavController navController=Navigation.findNavController( activity: this,R.id.nav_host_fragment);
AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(R.id.mainStoreFragment, R.id.ordersFragment, R.id.loginFragment, R.id.addProductFragment)

NavigationUI.setupWithNavController(binding.appToolbar, navController, appBarConfiguration);
NavigationUI.setupWithNavController(binding.bottomNavigation, navController);
```

Σχήμα 7.18: Κλάση MainActivity

7.3.6 SafeArgs Plugin

Στην πραγματικότητα, το πρόσθετο SafeArgs δημιουργεί κώδικα για να σας επιτρέψει να χρησιμοποιήσετε την ασφαλή πλοήγηση και τη μετάδοση επιχειρημάτων. Για να έχουμε τη δυνατότητα να το χρησιμοποιήσουμε θα πρέπει να προσθέσουμε το plugin στο αρχείο Gradle. Αυτό θα δημιουργήσει κλάσεις με βάση το γράφημα πλοήγησης (Σχήμα 7.19). Θα κάνει τις κλάσεις κατεύθυνσης για οποιονδήποτε προορισμό έχει ενέργειες και θα κάνει κλάσεις Args για οποιονδήποτε προορισμό με παραμέτρους (Σχήμα 7.20).

```
<fragment
    android:id="@+id/OrderDetailsFragment"
    android:name="com.example.myapplication.ui.Profile.Order.OrderDetailsFragment"
    android:label="My Order"
    tools:layout="@layout/fragment_order_details">
    <argument
        android:name="orderId"
        app:argType="string"
        app:nullable="true" />
</fragment>
```

Σχήμα 7.19: nav_graph.xml με παραμέτρους

```
binding.btnViewOrder.setOnClickListener(view -> {
    Navigation.findNavController(binding.getRoot()).navigate(OrderSubmittedFragmentDirections.actionOrderSubmittedFragmentToOrderDetailsFragment(orderId));
});
```

Σχήμα 7.20: Διαδρομή με παράμετρο

7.3.7 Intent

Η δυνατότητα που διαθέτει ένας χρήστης να μεταπηδήσει από μια εφαρμογή σε μια άλλη και μετά πίσω, στο Android λειτουργικό γίνεται με τη χρήση του Intent[25]. Το Intent στην εφαρμογή του χρήστη για παράδειγμα δίνει τη δυνατότητα στον χρήστη να ανοίξει την εφαρμογή φωτογραφίας για να μπορεί να φωτογραφήσει τη συνταγή του, ή το να μεταβεί στους φακέλους με τα αρχεία ή τις φωτογραφίες και να επιλέξει (Σχήμα 7.21).

```
binding.prescriptionFile.setOnClickListener(view -> {
    if (uri != null) {
        File file = new File(uri.toString());
        Intent intent = new Intent(Intent.ACTION_VIEW);
        intent.setDataAndType(Uri.parse(uri.toString()), type: "application/pdf");
        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
        intent.setFlags(Intent.FLAG_GRANT_READ_URI_PERMISSION);
        Intent newIntent = Intent.createChooser(intent, title: "Open File");
        try {
            startActivity(newIntent);
        } catch (ActivityNotFoundException e) {
            showToastError(e.getMessage());
        }
    }
});
```

Σχήμα 7.21: Χρήση Intent για επιλογή αρχείου τύπου PDF

Όταν ο χρήστης επιστρέψει στην εφαρμογή τότε η μέθοδος onActivityResult() αναλαμβάνει να χρησιμοποιήσει το αποτέλεσμα της επιλογής του χρήστη (Σχήμα 7.22).

```
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == PICK_PDF_CODE && resultCode == Activity.RESULT_OK) {
        uri = data.getData();
        String uriString = uri.toString();
        File myFile = new File(uriString);
        String path = myFile.getAbsolutePath();
        String displayName = null;

        if (uriString.startsWith("content://")) {
            Cursor cursor = null;
            try {
                cursor = getActivity().getContentResolver().query(uri,  projection: null,  selection: null,  selectionArgs: null,  sortOrder: null);
                if (cursor != null && cursor.moveToFirst()) {
                    displayName = cursor.getString(cursor.getColumnIndex(OpenableColumns.DISPLAY_NAME));
                }
            } finally {
                cursor.close();
            }
        } else if (uriString.startsWith("file://")) {
            displayName = myFile.getName();
        }
        binding.prescriptionFile.setVisibility(View.VISIBLE);
        binding.prescriptionFile.setText(displayName);
    }

}
```

Σχήμα 7.22: Μέθοδος onActivityResult() μετά την επιλογή αρχείου PDF

7.4 Επίλογος

Στο κεφάλαιο 7 είδαμε με παραδείγματα το πως έχουν δομηθεί οι δυο εφαρμογές αλλά και τα στοιχεία που χρησιμοποιήθηκαν και πως το κάθε ένα βοήθησε και με πιο τρόπο στη δημιουργία και ομαλή λειτουργία της εφαρμογής.

Κεφάλαιο 8ο: Συμπεράσματα και προτάσεις βελτίωσης

Με τη δημιουργία των δυο εφαρμογών έχει γίνει κατορθωτό να υλοποιηθούν όλες οι λειτουργικές απαιτήσεις των εφαρμογών που είχαμε θέσειωψ ψχ στην αρχή και να την καταστήσουν μια πλήρως λειτουργική εφαρμογή για τον χρήστη αλλά και για τον διαχειριστή. Με τη βοήθεια του Firebase και των δυνατοτήτων του, κατέστη δυνατόν οι δυο εφαρμογές να επικοινωνούν και να είναι λειτουργικές χωρίς την ανάγκη για δημιουργία κάποιου server, αφαιρώντας έξοδα διαχείρισης του αλλά και αρκετές ώρες δημιουργίας και συντήρησης του στο μέλλον καθώς και το ρίσκο τυχών προβλημάτων ασφαλείας του. Η αρχιτεκτονική MVVM βοήθησε ώστε ο κώδικας να μπορεί να ελεγχθεί ακόμα πιο εύκολα και χάρη στη δομή των πακέτων, είναι ακόμα πιο εύκολη η πλοήγηση στον κώδικα. Επίσης, η δυνατότητα που μας δίνει η κλάση Livedata να ενημερώνει άμεσα τα στοιχεία για τυχόν αλλαγές στα δεδομένα από το Firebase, αποτέλεσε τη βάση για να επιτευχθεί ο συγχρονισμός στην εφαρμογή. Με τη σειρά του, το ViewBinding ευκολύνει τη διαδικασία δημιουργίας του κώδικα, με ένα μεγάλο μέρος του να αφαιρείται αποφεύγοντας τη χρήση του findViewById. Όλα αυτά και ακόμα περισσότερα έχουν αναλυθεί στα πιο πάνω κεφάλαια, με παραδείγματα το πως έχει βοηθήσει το καθένα ξεχωριστά στην υλοποίηση της παρούσας Δ.Ε.

Στο μέλλον οι δυο εφαρμογές έχουν περιθώρια βελτίωσης. Πρώτον και βασικό είναι να διατηρηθεί και να διασφαλιστεί η σταθερότητα της εφαρμογής με την πάροδο του χρόνου. Επίσης, μια αρκετά καλή ιδέα θα ήταν η ενσωμάτωση ειδοποιήσεων που παρέχεται από το Firebase έτσι ο χρήστης θα ενημερώνεται άμεσα για μια αλλαγή στην κατάσταση παραγγελίας του. Ακόμα ένα σημαντικό εργαλείο από το Firebase, το Analyticsμ θα μπορούσε να ενσωματώσει κάποιες επιπλέον μεθόδους ώστε να προτείνει δημοφιλή προϊόντα στον χρήστη αλλά και να ενημερώνει τον διαχειριστή με στατιστικά στοιχεία για τα προϊόντα του.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] "What is Firebase? The complete story, abridged.", Medium, 2022. [Online]. Available: https://medium.com/firebase-developers/what-is-firebase-the-complete-story-abridged-bcc730c5f2c0. [Accessed: 10- Mar- 2022].

[2] "Firebase Authentication Password Hashing", Firebase Open Source, 2022. [Online]. Available: https://firebaseopensource.com/projects/firebase/scrypt/.[Accessed: 12- Mar- 2022].

[3] "Choose a Database: Cloud Firestore or Realtime Database | Firebase Realtime Database", Firebase, 2022. [Online]. Available: https://firebase.google.com/docs/database/rtdb-vs-firestore.[Accessed: 12- Mar- 2022].

[4] "Manage indexes in Cloud Firestore | Firebase", Firebase, 2022. [Online]. Available: https://firebase.google.com/docs/firestore/query-data/indexing.[Accessed: 01- Aug- 2022].

[5] "Mobile Operating System Market Share Worldwide | Statcounter Global Stats", StatCounter Global Stats, 2022. [Online]. Available: https://gs.statcounter.com/os-market-share/mobile/worldwide. [Accessed: 19- Mar- 2022].

[6] E. Belinski, "Android API Levels", Apilevels.com, 2022. [Online]. Available: https://apilevels.com/. [Accessed: 19- Mar- 2022].

[7] S. Bose, "A COMPARATIVE STUDY: JAVA VS KOTLIN PROGRAMMING IN ANDROID APPLICATION DEVELOPMENT", International Journal of Advanced Research in Computer Science, vol. 9, no. 3, pp. 41-45, 2018. Available: 10.26483/ijarcs.v9i3.5978.

[8] W. Jackson, Android apps for absolute beginners: Covering Android 7. California,USA: Apress, 2017.

[9]"Android | build.gradle - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/android-build-gradle/.[Accessed: 10- Apr- 2022].

[10] "The Activity Lifecycle | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/guide/components/activities/activity-lifecycle. [Accessed: 26- Apr- 2022].

[11] "Fragment lifecycle | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/guide/fragments/lifecycle. [Accessed: 26- Apr- 2022].

[12]"The Android Lifecycle cheat sheet—part I: Single Activities", Medium, 2022. [Online]. Available: https://medium.com/@JoseAlcerreca/the-android-lifecycle-cheat-sheet-part-i-single-activities-e49fd3d202ab. [Accessed: 18- Jul- 2022].

[13]"App resources overview | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/guide/topics/resources/providing-resources. [Accessed: 07- Aug- 2022].

[14]"MVC (Model View Controller) Architecture Pattern in Android with Example - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/mvc-model-view-controller-architecture-pattern-in-android-with-example/.[Accessed: 08- May- 2022].

[15] Prajesh, "Android: MVC, MVP, MVVM | TO THE NEW Blog", TO THE NEW BLOG, 2022. [Online]. Available: https://www.tothenew.com/blog/androidmvc-mvp-mvvm/. [Accessed: 01- May- 2022].

[16]"MVP (Model View Presenter) Architecture Pattern in Android with Example - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/mvp-model-view-presenter-architecture-pattern-in-android-with-example/.[Accessed: 10- May - 2022].

[17]"MVVM architecture, ViewModel and LiveData (Part 1)", Medium, 2022. [Online]. Available: https://proandroiddev.com/mvvm-architecture-viewmodel-and-livedata-part-1-604f50cda1. [Accessed: 29- Apr- 2022].

[18]"MVVM (Model View ViewModel) Architecture Pattern in Android - GeeksforGeeks", GeeksforGeeks,2022. [Online]. Available: https://www.geeksforgeeks.org/mvvm-model-view-viewmodel-architecture-pattern-in-android/. [Accessed: 18 - May- 2022].

[19]"Functional vs. Non-Functional Requirements: Why Are Both Important?", Uptech.team, 2022. [Online]. Available: https://www.uptech.team/blog/functional-vs-non-functional-requirements. [Accessed: 25- Jul- 2022].

[20]"LiveData overview | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/topic/libraries/architecture/livedata. [Accessed: 02- Aug- 2022].

[21]"Android Architecture Components — LiveData (Part 1)", Medium, 2022. [Online]. Available: https://medium.com/easyread/android-architecture-components-livedata-pt-1-133cad38e67e. [Accessed: 05- Aug- 2022].

[22]"View Binding | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/topic/libraries/view-binding. [Accessed: 10- Aug- 2022].

[23]"How to Use Glide Image Loader Library in Android Apps? - GeeksforGeeks", GeeksforGeeks, 2022. [Online]. Available: https://www.geeksforgeeks.org/image-loading-caching-library-android-set-2/. [Accessed: 10- Aug- 2022].

[24]"Using Android Jetpack Navigation Component", Medium, 2022. [Online]. Available: https://medium.com/kayvan-kaseb/using-android-jetpack-navigation-component-a3ed8ce4c8e8. [Accessed: 11- Aug- 2022].

[25]"Intent | Android Developers", Android Developers, 2022. [Online]. Available: https://developer.android.com/reference/android/content/Intent. [Accessed: 17- Aug- 2022].