ΔΙΕΘΝΕΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ
ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ
ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη εφαρμογής για την αντιμετώπιση της υπερφόρτωσης των κούριερ»

Της φοιτήτριας
Σίντου Κασσιανής
Αρ. Μητρώου: 164741

Επιβλέπων
Καζακόπουλος Αριστοτέλης
Καθηγητής

18 Ιανουαρίου 2023

Ανάπτυξη εφαρμογής για την αντιμετώπιση της υπερφόρτωσης των κούριερ.
21329
Φοιτήτρια: Σίντου Κασσιανή
Ονοματεπώνυμο εισηγητή: Καζακόπουλος Αριστοτέλης
Ημερομηνία ανάληψης Δ.Ε. 08-10-2021
Ημερομηνία περάτωσης Δ.Ε. 18-01-2023

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.

«Για τους αγαπημένους μου ανθρώπους»

Πρόλογος

Τα τελευταία χρόνια, η τεχνολογική ανάπτυξη αποκτά ραγδαίους ρυθμούς. Σε πάρα πολλούς τομείς, όπως στην ιατρική, η είσοδος και η χρήση των εξελιγμένων τεχνολογικών μέσων είναι αναπόφευκτη. Παράλληλα, καταγράφεται σημαντική αύξηση στο ηλεκτρονικό εμπόριο, με αξιοσημείωτη συνέπεια την υπερφόρτωση των εταιριών μεταφοράς δεμάτων. Δεν είναι λίγε οι φορές , τις οποίες ένα δέμα καθυστερεί αρκετά να φτάσει στον προορισμό του. Η χρήση τεχνολογικών μέσων για την επίλυση του προβλήματος αυτού, δεν είναι τόσο ανεπτυγμένη. Με αφορμή αυτό, η συγκεκριμένη διπλωματική εργασία επικεντρώνεται στην ανάπτυξη εφαρμογής για την καλύτερη διαχείριση μεταφοράς δεμάτων.

Περίληψη

Η εφαρμογή της παρούσας εργασίας προσφέρει λύση, αξιοποιώντας τα ήδη δρομολόγια που γίνονται καθημερινά από τους πολίτες μιας περιοχής. Απευθύνετε σε εκείνους που έχουν την επιθυμία αποστολής ενός δέματος καθώς και σε εκείνους με την θέληση και την δυνατότητα μεταφοράς. Ύστερα από έρευνα, σχεδιάστηκε έτσι ώστε να είναι εύχρηστη για την οριζόντια θέση της συσκευής και όσο το δυνατόν πιο φιλική γίνεται και να κάνει τον χρήστη να νιώθει άνετα, χρησιμοποιώντας την με το ένα χέρι. Η πλοήγηση στην εφαρμογή είναι εύκολη καθώς ο χρήστης αλληλεπιδρά με οθόνες που προβάλουν το κείμενο τους, στην γλώσσα που επιθυμεί. Είναι διαθέσιμη και απευθύνεται στο ευρύ κοινό. Θα μπορεί ο οποιοσδήποτε με τις αντίστοιχες προϋποθέσεις να αναλαμβάνει και να παραδίδει πακέτα που ο προορισμός τους βρίσκεται στον δρόμο του. Έτσι, θα συμβάλει γενικά στο έργο της παράδοσης, βοηθώντας στην ταχύτερη μεταφορά του πακέτου. Βεβαίως, η διαφορά θα είναι αισθητή μόλις υπάρξει ένας ικανοποιητικός αριθμός χρηστών που παραδίδουν, συνολικά μεγάλο αριθμό δεμάτων. Στην περίπτωση αυτή, θα είναι μία υπολογίσιμη δύναμη και η προσφερόμενη βοήθεια θα είναι σωτήρια. Τέλος, η εφαρμογή αυτή με τις άπειρες μελλοντικές επεκτάσεις και βελτιώσεις, μπορεί να θέσει τα θεμέλια για ένα καινούργιο μέλλον στον τρόπο διαχείριση μεταφοράς δεμάτων.

«Application development to deal with courier overload»

«Sintou Kassiani»

Abstract

The application of this project offers a solution, making useful the existing routes that are made daily by the citizens of a place. Is aimed those who have the desire to send a package as well as those with the will and ability to transport it. After research, the application was designed to be easy to use in the horizontal position of the device and as friendly as possible to make the user feel comfortable using it with one hand. Navigating the application is easy as the user interacts with screens that display their text, in the language of user's choice. It is available and aimed the public. Anyone with the corresponding conditions will be able to take over and deliver packages whose destination is on their way. Thus, the user will generally contribute to the delivery task by helping to move the package faster. Of course, the difference will be noticeable once there is a huge number of users, delivering, in total a large number of packages. In this case, the help they offer will be lifesaving. Finally, this application with its infinite future expansions and improvements, can make the foundation for a new future about the way packages transport is managed.

Περιεχόμενα

Κατάλογος Σχημάτων

Κατάλογος Πινάκων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η ανάγκη που οδήγησε στην ανάπτυξη της εφαρμογής για την καλύτερη και αποτελεσματικότερη διαχείριση μεταφοράς δεμάτων, αφορά την χρονική στιγμή που έγινε η συγγραφή της διπλωματικής εργασίας. Είναι πολύ πιθανών στο άμεσο μέλλον, με τους ταχύτατους ρυθμούς εξέλιξης, να υπάρξουν μικρές ή μεγάλες διαφοροποιήσεις, . Παρακάτω, θα αναπτυχθούν πληροφορίες σχετικά με την συνηθισμένη διαδικασία που ακολουθεί κάποιος για την αποστολή ενός δέματος.

1.2 Σύστημα μεταφοράς δεμάτων

Η αποστολή ενός δέματος επιτυγχάνεται κατά κύριο λόγω μέσω μιας μεταφορικής εταιρίας δεμάτων. Η διαδικασία της παραλαβής και αποστολής, εξαρτάτε από την εκάστοτε εταιρία. Με εξαίρεση, μεμονωμένων εταιριών, κοινό σημείο αποτελεί το γεγονός, πως όλα τα δέματα έχουν αφετηρία ένα φυσικό κατάστημα της εταιρίας. Μόλις παραδοθεί το δέμα στο φυσικό κατάστημα από τον αποστολέα, κοστολογείται ανάλογα τα χαρακτηριστικά του. Το ογκομετρικό βάρος και η περιοχή τελικού προορισμού του δέματος επηρεάζουν το κόστος. Σημαντικό ρόλο παίζει αν το δέμα θα το παραλάβει κάποιος από το κατάστημα η αν θα πρέπει να παραδοθεί στην ακριβής διεύθυνση του τελικού παραλήπτη. Σε δυσπρόσιτες περιοχές ή νησιωτικές περιοχές το κόστος αυξάνεται σημαντικά. Ακόμη, ένα δέμα έχει επιβαρυμένο κόστος όταν είναι με αντικαταβολή ή αποστέλλεται στο εξωτερικό. Η επιθυμία άμεσης παράδοσης, σε συγκεκριμένο και ορισμένο χρονικό διάστημα, κοστολογείται επιπλέον και εξαρτάται , αν η ημέρα παράδοσης είναι Σάββατο η είναι ημέρα αργίας . Όλα τα παραπάνω, συνυπολογίζονται και σχηματίζουν, τελικώς, τα έξοδα αποστολής, καθώς και τον εκτιμώμενο χρόνο αποστολής. Με την ολοκλήρωση της κοστολόγησης, και της εκτίμησης του χρόνου αποστολής, ξεκινά η αποστολή. Διαθέσιμοι υπάλληλοι αναλαμβάνουν την παράδοση των δεμάτων προς τις τελικές διευθύνσεις, στην περιοχή που εξυπηρετούν ή την προώθηση τους προς τα καταστήματα των άλλων περιοχών.

Από τα όσα αναφέρθηκαν προηγουμένως, συμπεραίνουμε πως δύο είναι οι κύριες μεταβλητές στην όλη διαδικασία. Η πρώτη από αυτές είναι το χρήμα. Από την στιγμή που θα υπολογιστεί μετατρέπεται σε σταθερά και παραμένει ως έχει μέχρι το τέλος. Η δεύτερη, είναι ο χρόνος. Μετά τον υπολογισμό του, παραμένει ως μεταβλητή. Μάλιστα μετατρέπεται σε ευαίσθητη μεταβλητή και επηρεάζεται από φυσικά γεγονότα που εξελίσσονται παράλληλα. Λίγες είναι οι φορές που ένα δέμα, θα φτάσει στον τελικό προορισμό συντομότερα από τον αναμενόμενο. Σύνηθες φαινόμενο είναι η καθυστερημένη παράδοση. Τον συχνότερο λόγο, που έχει ως αποτέλεσμα, την προσθήκη των επιπλέον καθυστερήσεων, αποτελεί το γεγονός της υπερφόρτωσης, από δέματα, της εταιρίας μεταφοράς δεμάτων.

1.3 Η ανάγκη

Καθώς ο τρόπος διαχείρισης, τελικά, μίας αποστολής είναι αρκετά σύνθετος και κατ’ επέκταση γίνεται ακόμη πιο σύνθετη η σύγκριση μεταξύ ταχυμεταφορών [1], δημιουργήθηκε η ανάγκη μιας πιο απλουστευμένης και εύχρηστης μορφής αποστολής η οποία θα είναι προσβάσιμη από το ευρύ κοινό.

Στόχος αυτής της εργασίας είναι η ανάπτυξη μίας ολοκληρωμένης Android εφαρμογής η οποία θα εκμεταλλεύεται τα δρομολόγια που γίνονται καθημερινά από τους πολίτες μιας περιοχής θα παρουσιάζει, με έναν εύχρηστο και φιλικό προς τον χρήστη τρόπο, όλα τα απαραίτητα δεδομένα των δεμάτων και θα διαχειρίζεται τις αποστολές αυτών.

1.4 Επίλογος

Στο συγκεκριμένο κεφάλαιο της διπλωματικής εργασίας, έγινε αναφορά στο σύστημα διαχείρισης των εταιριών μεταφοράς δεμάτων, στον τρόπο και στην μεθοδικότητα που ακολουθούν για την ολοκλήρωση μιας αποστολής, καθώς και στην ανάγκη που οδήγησε στην εκπόνηση της εργασίας. Στα επόμενα κεφάλαια θα περιγραφεί το τεχνολογικό περιβάλλον που χρησιμοποιήθηκε, θα γίνει ανάλυση των απαιτήσεων της εφαρμογής, θα σχολιαστεί μέρος του κώδικα που αφορά τον σχεδιασμό και την ανάπτυξη της εφαρμογής και θα παρουσιαστεί αναλυτικά η εφαρμογή. Στο τελευταίο κεφάλαιο, θα αναφερθούν κάποια συμπεράσματα και κάποιες πιθανές, μελλοντικές, βελτιστοποιήσεις της εφαρμογής.

Κεφάλαιο 2ο: Περιγραφή του τεχνολογικού περιβάλλοντος που χρησιμοποιήθηκε

2.1 Εισαγωγή

Στο κεφάλαιο αυτό θα παρουσιαστούν σύντομα οι τεχνολογίες, τα εργαλεία και η αρχιτεκτονική που χρησιμοποιήθηκαν και βοήθησαν στην ανάπτυξη και στην λειτουργία της εφαρμογής.

2.2 Android Studio

Επιλέχθηκε το περιβάλλον του Android Studio [2] [3] και η χρήση της γλώσσας java, καθώς είναι επιθυμητό η εφαρμογή να τρέχει σε ένα ικανοποιητικό αριθμό κινητών συσκευών, να είναι εύχρηστη και προσιτή. Με τις δυνατότητες και τα εργαλεία που παρέχει το Android Studio σε συνδυασμό με την Java, η επιθυμία έγινε πραγματικότητα.

Βασισμένο στο λογισμικό της JetBrains IntelliJ IDEA, το Android Studio , ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (IDE), σχεδιάστηκε αποκλειστικά για προγραμματισμό Android. Είναι διαθέσιμο για Windows, Mac OS X και Linux , ενώ αντικατέστησε τα Eclipse Android Development Tools (ADT) ως το κύριο IDE της Google για ανάπτυξη εφαρμογών Android.

Δύο είναι οι γλώσσες προγραμματισμού που υποστηρίζει [4]. Μια από αυτές είναι η Kotlin, που είναι μια νέα γλώσσα προγραμματισμού που αναπτύχθηκε από προγραμματιστές από το IDE της εταιρίας Jet Brains και εμφανίστηκε για πρώτη φορά το 2011 με την επίσημη κυκλοφορία της να είναι τον Φεβρουάριο του 2016. Η Java, μία γλώσσα αντικειμενοστραφούς προγραμματισμού (OOP) που τέθηκε σε χρήση το 1995. Αναπτύχθηκε από την εταιρία SUN Microsystems, την οποία αργότερα απέκτησε η εταιρία Oracle, είναι η δεύτερη γλώσσα που υποστηρίζεται. Αξίζει να αναφερθεί ,πως ο κώδικας Java μπορεί να μετατρέπεται σε κώδικα Kotlin και αντίστροφα, ενώ ένα Android project μπορεί να περιέχει κώδικα Java και Kotlin ταυτόχρονα.

Τέλος, το Android Studio χρησιμοποιεί το Gradle [5], ένα προηγμένο και προεγκατεστημένο σύστημα κατασκευής (build system), παρόμοιο με το Apache Ant και το Apache Maven, βασιζόμενο στο Groovy για την κατασκευή και τη διαχείριση Java projects. Στην ουσία, συγκεντρώνει και συνδυάζει τους πόρους, τον πηγαίο κώδικα και τις εξωτερικές βιβλιοθήκες σε ένα τελικό APK.

2.3 Αρχιτεκτονική MVVM

Είναι επιθυμητό, κάθε εφαρμογή, κάθε project, να ακολουθεί μια αρχιτεκτονική για να αποφεύγει φαινόμενα όπως αυτό του "spaghetti code", όπου ο κώδικας είναι τόσο μπερδεμένος , όπως ένα πιάτο με spaghetti. Για την αποφυγή του προηγούμενου φαινομένου, την διατήρηση καθαρού κώδικα και για την γενικότερη ευελιξία επιλέχθηκε η αρχιτεκτονική MVVM (Model View View Model) [6]. Ένα από τα πιο συνηθισμένα λάθη στη δημιουργία Android εφαρμογής είναι η ενσωμάτωση όλου του κώδικα και της λειτουργικότητας στις κλάσεις των Activities και των Fragments, ενώ θα έπρεπε να περιέχουν κώδικα που αφορά μόνο την διεπαφή με τον χρήστη (UI) και την παρουσίαση των εκάστοτε δεδομένων.

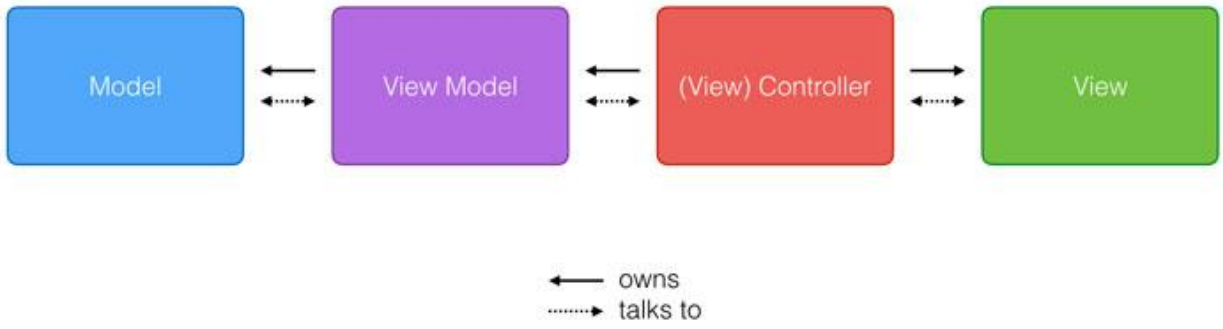
2.3.1 Πλεονεκτήματα

Συχνά ένας προγραμματιστής ακολουθώντας μια αρχιτεκτονική, έρχεται αντιμέτωπος με το ερώτημα 'ποιο κομμάτι κώδικα ταιριάζει και που'. Στην MVVM αρχιτεκτονική αυτό, είναι αρκετά πιο

ξεκάθαρο. Τα τέσσερα επίπεδα, Μοντέλο, Μοντέλο Προβολής, Ελεγκτής και Προβολή είναι διαχωρισμένα. Ο διαχωρισμός αυτός οδηγεί στην καλύτερη διαχείριση τους, κάτι που χαρακτηρίζει και ξεχωρίζει την αρχιτεκτονική αυτή από τις άλλες. Έτσι, λόγω της διαφανής επικοινωνίας που υπάρχει μεταξύ των τεσσάρων επιπέδων, ο κώδικας βρίσκεται στο επίπεδο και στην κλάση που πρέπει να βρίσκεται, είναι ανεξάρτητος καθώς και εύκολος σε οποιοδήποτε τεστ.

2.3.2 Κανόνες

Υπάρχουν 6 άτυποι κανόνες που πρέπει να γνωρίζει κάποιος, όταν επιλέξει να ακολουθήσει την αρχιτεκτονική MVVM. Οι παρακάτω κανόνες αν και μπορούν να τροποποιηθούν , λειτουργούν ως κατευθυντήριες γραμμές και αποτελούν την βάση για την εφαρμογή της αρχιτεκτονικής αυτής.



Εικόνα 2.1: MVVM Αρχιτεκτονική

Όπως αναπαριστάται γραφικά στην Εικόνα 2.1 , παρακάτω σχολιάζονται οι 6 άτυποι αυτοί κανόνες.

- Η προβολή(View) δεν γνωρίζει για τον ελεγκτή(Controller) που ανήκει. Γενικά, οι προβολές (Views) γνωρίζουν μόνο πώς να παρουσιάσουν τα δεδομένα προς τον χρήστη.
- Ο ελεγκτής(Controller) δεν γνωρίζει για το μοντέλο(Model). Αυτό είναι κάτι που ξεχωρίζει την MVVM αρχιτεκτονική από την MVC(Model-View-Controller).
- Το μοντέλο(Model) δεν γνωρίζει για το μοντέλο προβολής(View Model) στο οποίο ανήκει.
- Το μοντέλο(Model) ανήκει στο μοντέλο προβολής(View Model). Όταν χρησιμοποιείτε η MVC, το μοντέλο(Model) ανήκει συνήθως στον ελεγκτή(Controller).
- Η προβολή(View) ανήκει στον ελεγκτή(Controller).
- Ο ελεγκτής(Controller) κατέχει το μοντέλο προβολής(View Model). Αλληλεπιδρά με το επίπεδο του μοντέλου(Model) μέσω ενός η περισσοτέρων μοντέλων προβολής(View Models).

Όπως προαναφέρθηκε οι κανόνες αυτοί είναι άτυποι. Μπορούν δηλαδή να παραβιαστούν η να τροποποιηθούν, με κίνδυνο, βέβαια, την διατάραξη της σωστής εφαρμογής και της καλής απόδοσης της αρχιτεκτονικής.

2.4 Firebase

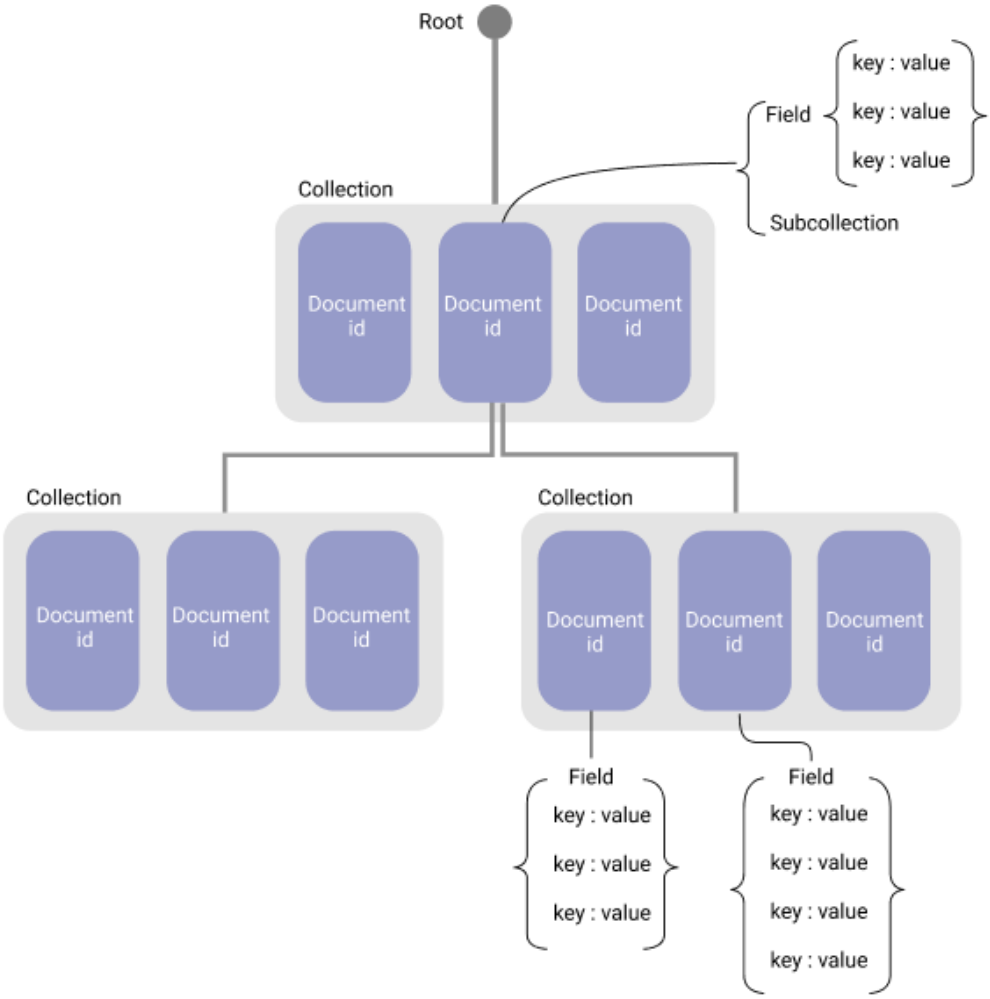
Το Firebase [7] είναι μια πλατφόρμα που αναπτύχθηκε από την Google για τη δημιουργία εφαρμογών για κινητά και ιστούς . Αρχικά ήταν μια ανεξάρτητη εταιρεία που ιδρύθηκε το 2011 από την Envolve.Το 2014, η Google απέκτησε την πλατφόρμα και είναι πλέον η κορυφαία προσφορά της όσον αφορά την ανάπτυξη εφαρμογών. Το πρώτο προϊόν του Firebase ήταν το Firebase Realtime Database, ένα API που συγχρονίζει δεδομένα εφαρμογών σε συσκευές iOS, Android και Web και τα αποθηκεύει στο cloud του Firebase.

2.4.1 Authentication

Μία από τις κυριότερες λειτουργίες μιας εφαρμογής, είναι ο έλεγχος της ταυτότητας. Ο έλεγχος ταυτότητας της Firebase [8] στοχεύει να διευκολύνει τη δημιουργία ασφαλών συστημάτων ελέγχου, βελτιώνοντας παράλληλα την εμπειρία σύνδεσης του τελικού χρήστη. Παρέχει ,μια από άκρο σε άκρο (end-to-end)και εύκολα υλοποιήσιμη, ταυτοποίηση, υποστηρίζοντας λογαριασμούς email και κωδικούς πρόσβασης, έλεγχο ταυτότητας και σύνδεση μέσω αριθμού τηλεφώνου, καθώς και σύνδεση μέσω Google, Twitter, Facebook και Github. Η ολοκληρωμένη ασφάλεια που προσφέρει η firebase είναι κατασκευασμένη από την ίδια ομάδα που ανέπτυξε το Google Sign-in και το Chrome Password Manager. Εκμεταλλεύτηκε και εφαρμόζει την τεχνογνωσία της Google πάνω στην διαχείριση μιας από τις μεγαλύτερες βάσεις δεδομένων λογαριασμών στον κόσμο.

2.4.2 Cloud Firestore

Το Cloud Firestore [9]είναι μια NoSQL βάση δεδομένων, που επιτρέπει την αποθήκευση δεδομένων, τον συγχρονισμό και την αναζήτηση δεδομένων, για την εκάστοτε εφαρμογή, σε παγκόσμια κλίμακα. Κατά την αποθήκευση, η πληροφορία αποθηκεύεται σε μορφή εγγράφου document και συλλογής collection Εικόνα 2.2. Έτσι μπορούν να δημιουργηθούν ιεραρχίες με σχετικά δεδομένα, εύκολα προσβάσιμα. Τέλος, με το Cloud Firestore και με τον συγχρονισμός των δεδομένων, μεταξύ πολλών συσκευών, η εφαρμογή κατατάσσεται στις εφαρμογές πραγματικού χρόνου (real time).



Εικόνα 2.2: Firestore data organization

2.5 Google Materials

Η βιβλιοθήκη Material είναι ένα σύστημα σχεδίασης που αναπτύχτηκε από την Google για να βοηθήσει τους προγραμματιστές να δημιουργούν υψηλής ποιότητας, ψηφιακές εμπειρίες για Android, iOS, Flutter και το Web [10]. Προσφέρει μεγάλη ποικιλία από έτοιμα συστατικά, για την σχεδίαση της διεπαφής χρήστη, όπως TextViews, Data Pickers, Toolbars . Προσφέρει ακόμη και αρκετές χρήσιμες σχεδιαστικές οδηγίες(guidelines).

2.6 ProGuard

Κατά την ολοκλήρωση μιας εφαρμογής, για την ελαχιστοποίηση του μεγέθους της είναι αναγκαίο να συρρικνωθεί. Με τον όρο αυτό, αναφερόμαστε στην κατάργηση περιττού κώδικα και πόρων που δεν χρησιμοποιούνται καθώς και στην συμπίεση του αρχείου. Για να επιτευχθεί το παραπάνω, χρησιμοποιείται το ProGuard [11], ένα δωρεάν εργαλείο, ανοιχτού κώδικα, για εφαρμογές Java και Kotlin. Μπορεί να επιτύχει πάνω από το 90% σμίκρυνση, 20% αύξηση της ταχύτητας μιας εφαρμογής και ταυτόχρονα να παρέχει μια βασική άμυνα, απέναντι στην κακοπροαίρετη αντιστροφή του τελικού αρχείου, με την τεχνική αλλαγής ονομάτων κλάσεων, μεθόδων και μεταβλητών (code obfuscation).

Κεφάλαιο 3ο: Ανάλυση Εφαρμογής

3.1 Εισαγωγή

Στο κεφάλαιο αυτό αναλύεται ο σκοπός και οι απαιτήσεις που καλείται να καλύψει η εφαρμογή. Ένα πίνακας με τις λειτουργίες της εφαρμογής μαζί με ένα διάγραμμα κλάσεων αποσκοπούν στην καλύτερη κατανόηση του τρόπου λειτουργίας της εφαρμογής αλλά και στην καλύτερης και αποδοτικότερης αλληλεπίδρασης με αυτήν.

3.2 Σκοπός της εφαρμογής

Δεν είναι λίγες οι φορές, στις οποίες ένα δέμα έχει καθυστερήσει αρκετά να φτάσει στον προορισμό του, λόγω υπερφόρτωσης της εκάστοτε εταιρίας μεταφοράς δεμάτων [12]. Σκοπός της εφαρμογής είναι να βοηθήσει στην καλύτερη διακίνηση και παράδοση. Το πρόβλημα της γρήγορης μεταφοράς των δεμάτων ήδη απασχολεί εταιρίες ταχυμεταφοράς δεμάτων, παγκοσμίως. Οι λύσεις που καταφεύγουν να ακολουθήσουν όμως, είναι προσωρινές χωρίς να λύνουν ουσιαστικά το πρόβλημα. Καλούνται να προσλάβουν επιπλέον προσωρινό προσωπικό, κάτι που για οικονομικούς λόγους, κυρίως, δεν εφαρμόζεται. Αντ' αυτού, πιέζουν το ήδη υπάρχων προσωπικό για γρηγορότερες παραδόσεις και γρηγορότερες μετακινήσεις. Αυτό έχει ως αποτέλεσμα, για την μεταφορά, αυξημένο κίνδυνο για τους οδηγούς, ενδεχομένως, λόγω αυξημένης ταχύτητας και για την παράδοση, αυξημένο κίνδυνο κλοπής η καταστροφής των δεμάτων από καιρικές συνθήκες, καθώς, τα πακέτα αφήνονται στην πόρτα και γενικά σε εξωτερικό χώρο, χωρίς να τα έχει παραλάβει κάποιο φυσικό πρόσωπο. Συμπερασματικά, ακολουθώντας την τακτική αυτή και πάλι δεν μπορεί να έρθει ισορροπία ως προς το αποτέλεσμα. Μπορεί, εν τέλει να παραδίδονται ταχύτερα τα δέματα, όμως με τις ανάλογες συνέπιες κάθε φορά. Αξίζει, να θυσιαστεί η ποιότητα και η αξιοπιστία, για την ταχύτητα;

Το πρόβλημα, λοιπόν, θα έρθει να το διαχειριστεί και να επιλύσει η εφαρμογή, αξιοποιώντας τα ήδη δρομολόγια που γίνονται καθημερινά από τους πολίτες μιας περιοχής. Μέσω της εφαρμογής θα μπορεί ο οποιοσδήποτε να επιλέγει και να παραδίδει δέματα που ο προορισμός τους βρίσκεται στον δρόμο του. Έστω ένας οδηγός χ, χρήστης της εφαρμογής, κάνει καθημερινά μια διαδρομή από το σπίτι του, στην δουλειά του. Αν υπάρχει δέμα στην εφαρμογή, με σημείο παραλαβής και προορισμού, ανάμεσα, η κοντά στην διαδρομή του οδηγού, τότε θα μπορεί να το αναλάβει να το παραδώσει στον τελικό προορισμό του χωρίς κάποιον ιδιαίτερο κόπο. Αντίστοιχα, μπορεί να γίνει και για προορισμούς που δύσκολα θα έφτανε μια εταιρία. Έστω ότι κάποιος θέλει να στείλει δέμα, σε ένα νησί, η σε μια δυσπρόσιτη περιοχή. Είναι γνωστό, πως οι εκτιμώμενες ημέρες, για την παράδοση από μια εταιρία μεταφοράς δεμάτων, είναι αυξημένες σε τέτοιες περιπτώσεις, πόσο μάλλον αν η εταιρία αντιμετωπίζει ήδη πρόβλημα υπερφόρτωσης και διαχείρισης. Βάζοντάς το στην εφαρμογή, ένας χρήστης- οδηγός, που θα έχει προγραμματισμένο ταξίδι με τον ίδιο προορισμό με το δέμα, και έτσι και αλλιώς θα εκτελέσει την διαδρομή αυτή, γιατί να μην μεταφέρει και το δέμα ως εκεί. Θα μπορεί, λοιπόν, να το επιλέξει, να το μεταφέρει και να το παραδώσει.

Συνοψίζοντας, και στις δύο παραπάνω περιπτώσεις ο οδηγός θα έχει συμβάλει γενικά στο έργο της παράδοσης, βοηθώντας στην ταχύτερη μεταφορά, δίχως να θέτει σε κίνδυνο τον ίδιο και την ακεραιότητα του δέματος. Σημαντικό είναι, πως ο σκοπός της εφαρμογής θα έχει επιτευχθεί. Ένα δέμα θα έχει φτάσει στον τελικό προορισμό του, χωρίς αυξημένες καθυστερήσεις και κυρίως αξιοποιώντας μια προγραμματισμένη διαδρομή, χωρίς επιπλέον επιβάρυνση.

3.3 Ανάλυση απαιτήσεων

Είναι πολύ σημαντική η φάση κατά την οποία, συγκεντρώνονται οι απαραίτητες πληροφορίες για τον σχεδιασμό της βάσης δεδομένων και τον καθορισμό των προδιαγραφών της εφαρμογής [13]. Η φάση αυτή ονομάζεται ανάλυση απαιτήσεων , γίνεται κατά την ανάπτυξη μιας εφαρμογής και δίνει στον προγραμματιστή την δυνατότητα εκτίμησης του χρονοδιαγράμματος του συνολικού έργου.

3.3.1 Λειτουργίες εφαρμογής

Από την στιγμή που καταγράφηκαν τα δεδομένα και οι απαραίτητες πληροφορίες, δημιουργήθηκε ο παρακάτω πίνακας με τις βασικές λειτουργίες της εφαρμογής.

Πίνακας 1 : Λειτουργίες εφαρμογής

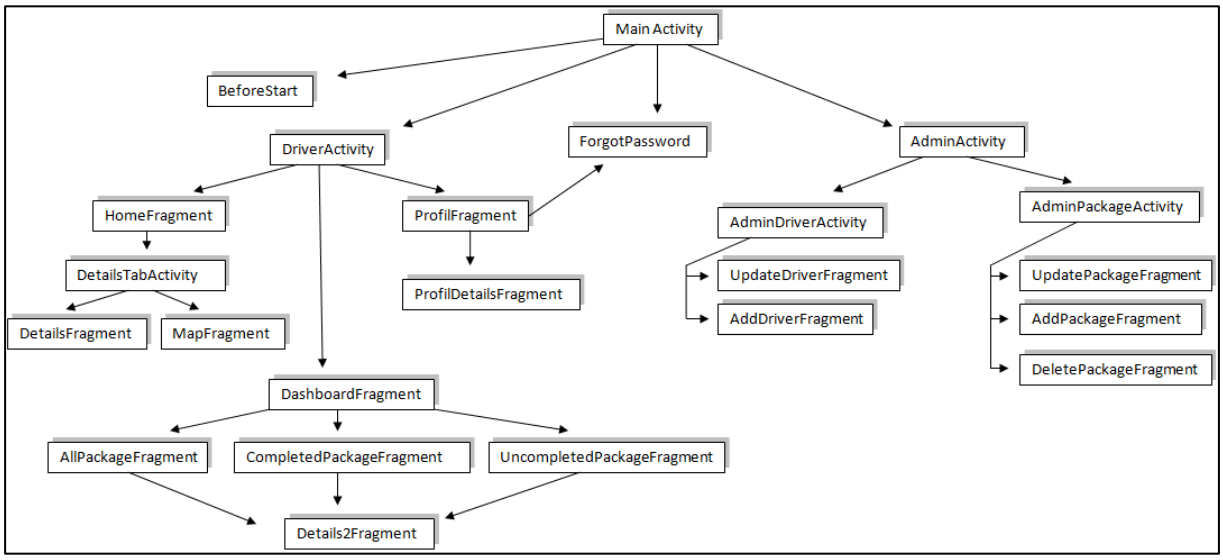
Ενέργεια	**Περιγραφή απαίτησης**	**Περιγραφή ενέργειας του χρήστη**
Δημιουργία Λογαριασμού - Εγγραφή	Επικοινωνία με την βάση δεδομένων για την δημιουργία νέου λογαριασμού. Μήνυμα λάθους σε περίπτωση μη αποδεκτών στοιχείων.	Συμπλήρωση στοιχείων και πάτημα του κουμπιού «εγγραφή»
Είσοδος -Σύνδεση	Έλεγχος στοιχείων που πληκτρολόγησε ο χρήστης. Αν είναι σωστά και υπάρχει λογαριασμός, γίνεται είσοδος στην εφαρμογή. Αν ο λογαριασμός είναι λογαριασμός με δικαιώματα επιβεβαιωμένου οδηγού, τότε εισέρχεται στην εφαρμογή, διαφορετικά, μεταβαίνει σε μια οθόνη με πληροφορίες για το πώς θα γίνει επιβεβαιωμένος οδηγός. Διαφορετικά αν δεν υπάρχει καθόλου λογαριασμός εμφανίζεται μήνυμα λάθους.	Συμπλήρωση στοιχείων και πάτημα του κουμπιού «είσοδος»
Είσοδος- Σύνδεση με Google λογαριασμό	Αν υπάρχει λογαριασμός Google τότε ο χρήστης, χρησιμοποιώντας τον, μπορεί να συνδεθεί στην εφαρμογή. Αλλιώς εμφανίζεται μήνυμα λάθους	Συμπλήρωση email και πάτημα του κουμπιού «Σύνδεση» με Google λογαριασμό
Έξοδος- Αποσύνδεση	Ο χρήστης- οδηγός αποσυνδέεται και δεν έχει πλέον πρόσβαση στις λειτουργίες.	Πάτημα του κουμπιού «Αποσύνδεση»
Αλλαγή κωδικού	Αποστέλλεται ένα email αλλαγής κωδικού στο email που είναι συνδεδεμένος ο	Πάτημα του κειμένου «Αλλαγή κωδικού»

	χρήστης.	
Ξέχασα τον κωδικό	Μετά την πληκτρολόγηση του email από τον χρήστη, αποστέλλεται ένα email επαναφοράς, στο email που πληκτρολόγησε.	Πάτημα του κειμένου «Forgot Password»
Επιλογή δέματος	Αφού πατηθεί ένα δέμα από τον χρήστη – οδηγό και αφού πατηθεί το πλήκτρο «επιλογή», το δέμα προστίθεται στον χρήστη. Τέλος ενημερώνονται οι αντίστοιχοι πίνακες στην βάση.	Επιλογή δέματος και πάτημα του κουμπιού «Επιλογή».
Ακύρωση επιλογής δέματος	Στο μενού με τα επιλεγμένα δέματα, όταν επιλεχθεί ένα δέμα, με το πάτημα του κουμπιού «Ακύρωση». Το δέμα γίνεται ξανά διαθέσιμο στους υπόλοιπους χρήστες, και ενημερώνεται η βάση.	Επιλογή δέματος και πάτημα του κουμπιού «Ακύρωση».
Ολοκλήρωση παράδοσης	Αφού πατηθεί ένα δέμα που έχει αναλάβει ήδη ,από τον χρήστη – οδηγό και αφού πατηθεί το πλήκτρο «ολοκλήρωση», ενημερώνονται οι αντίστοιχοι πίνακες στην βάση και το δέμα έχει παραδοθεί.	Επιλογή δέματος και πάτημα του κουμπιού «Ολοκλήρωση».
Πληροφορίες για το δέμα	Πατώντας πάνω σε ένα δέμα ο χρήστης-οδηγός, εμφανίζονται επιπλέον πληροφορίες για αυτό. Συγκεκριμένα μια καρτέλα με τον χάρτη και μια με τις πληροφορίες.	Επιλογή ενός δέματος
Ενεργοποίηση-Απενεργοποίηση ειδοποιήσεων	Επικοινωνία με την βάση, εγγραφή-απεγγραφή του χρήστη και ενεργοποίηση-απενεργοποίηση ειδοποιήσεων για την προσθήκη δέματος. Κάθε φορά που προστίθεται ένα δέμα, ο χρήστης-οδηγός θα λαμβάνει μια ειδοποίηση.	Πάτημα στον διακόπτη «Ειδοποιήσεις»
Προσθήκη χρήστη/οδηγού	Ο χρήστης- διαχειριστής μπορεί να προσθέσει από την αρχή έναν νέο χρήστη οδηγό. Συμπληρώνοντας όλα τα απαραίτητα στοιχεία, και πατώντας, προσθήκη και	Πάτημα του κουμπιού «Προσθήκη χρήστη/οδηγού». Συμπλήρωση στοιχείων. Πάτημα του κουμπιού «Προσθήκη».

	ολοκλήρωση, ένας νέος χρήστης δημιουργείται και προστίθεται στην βάση. Εμφανίζεται ενημερωτικό μήνυμα επυτιχίας/αποτυχίας	Τσεκάρισμα των κουτιών «Διαχειριστής» και «Με άδεια οδηγού» , αν χρειάζεται. Πάτημα του κουμπιού «Ολοκλήρωση».
Επεξεργασία Χρήστη/Οδηγού	Ο χρήστης- διαχειριστής μπορεί να επεξεργαστεί τα στοιχεία ενός άλλου χρήστη. Αναζητώντας τον χρήστη με το email,έχει πρόσβαση στις πληροφορίες του χρήστη. Με το πάτημα του κουμπιού «αποθήκευση» αποθηκεύονται οι αλλαγές και ενημερώνεται η βάση.	Πληκτρολόγηση του email και πάτημα του κουμπιού «Αναζήτηση». Αλλαγή των στοιχείων του χρήστη. Πάτημα του κουμπιού «Αποθήκευση»
Προσθήκη δέματος	Ο χρήστης- διαχειριστής μπορεί να προσθέσει από την αρχή ένα νέο δέμα. Συμπληρώνοντας όλα τα απαραίτητα στοιχεία, πατώντας προσθήκη και ολοκλήρωση , ένα νέο δέμα δημιουργείται, προστίθεται στην βάση. Μετά την ολοκλήρωση στην οθόνη εμφανίζεται το id του δέματος. Και το σύστημα ειδοποιεί όλους του χρήστες για την προσθήκη νέου δέματος.	Πάτημα του κουμπιού «Προσθήκη δέματος». Συμπλήρωση στοιχείων. Πάτημα του κουμπιού «Προσθήκη». Τσεκάρισμα του κουτιού «Εύθραυστο» , αν χρειάζεται. Πάτημα του κουμπιού «Ολοκλήρωση».
Επεξεργασία δέματος	Ο χρήστης- διαχειριστής μπορεί να επεξεργαστεί τα στοιχεία από ένα δέμα. Αναζητώντας το δέμα με το id του, έχει πρόσβαση στις πληροφορίες. Με το πάτημα του κουμπιού «αποθήκευση» αποθηκεύονται οι αλλαγές και ενημερώνεται η βάση.	Πληκτρολόγηση του id και πάτημα του κουμπιού «Αναζήτηση». Αλλαγή των στοιχείων του δέματος. Πάτημα του κουμπιού «Αποθήκευση»
Διαγραφή δέματος	Ο χρήστης- διαχειριστής μπορεί να διαγράψει ένα δέμα. Αναζητώντας το δέμα με το id του, και πατώντας το κουμπί «διαγραφή» διαγράφεται το δέμα από την βάση, αν αυτό επιτρέπεται	Πληκτρολόγηση του id και πάτημα του κουμπιού «Αναζήτηση». Πάτημα του κουμπιού «Διαγραφή»

3.3.2 Διάγραμμα κλάσεων

Στην συνέχεια στην Εικόνα 3.1 παρατίθεται ένα διάγραμμα κλάσεων που δείχνει τις συσχετίσεις μεταξύ των κλάσεων και των λειτουργιών της εφαρμογής.



Εικόνα 3.1: Διάγραμμα Κλάσεων

Στο παραπάνω διάγραμμα, απεικονίζονται με ορθογώνια παραλληλόγραμμα οι κλάσεις, ενώ τα βελάκια δείχνουν την κατεύθυνση και την διαδρομή δημιουργίας και επικοινωνίας των αντίστοιχων κλάσεων. Αρχικά, όταν ανοίγει η εφαρμογή ο χρήστης βρίσκεται στην οθόνη εισόδου ή εγγραφής της εφαρμογής, ενώ αλληλεπιδρά με την κλάση MainActivity. Υπάρχουν 4 πιθανές διαδρομές. Οι 3 από αυτές είναι μετά την επιτυχημένη είσοδο του χρήστη. Στην περίπτωση που ο χρήστης έχει λογαριασμό και δεν μπορεί να συνδεθεί του δίνεται η δυνατότητα για επαναφορά κωδικού με την κλάση ForgotPasswordActivity.

Οι άλλες 3 περιπτώσει αντιστοιχούν και στους 3 διαφορετικούς ρόλους χρηστών που μπορεί να έχει η εφαρμογή. Ο πρώτος αντιπροσωπεύει τον χρήστη ,που ναι μεν έχει κάνει εγγραφή, αλλά δεν έχει ολοκληρώσει με τα απαραίτητα έγγραφα, που θα του δώσουν την άδεια για την πλήρη είσοδο στις λειτουργίες της εφαρμογής. Όπως φαίνεται και στο διάγραμμα από την αντίστοιχη κλάση BeforeStart δεν υπάρχει επικοινωνία με τις υπόλοιπες κλάσεις. Επομένως , μόνο και εφόσον ο χρήστης έχει προσκομίσει όλα τα απαραίτητα έγγραφα, ο λογαριασμός του θα αναβαθμιστεί από τον διαχειριστή και ο ρόλος του θα είναι πλέον εκείνος του χρήστη-οδηγού.

Εισερχόμενος ως χρήστης-οδηγός, η εφαρμογή καλεί την κλάση DriverActivity και ανοίγει οθόνη που εμπεριέχει ένα μενού. Το μενού αποτελείται από 3 πιθανές επιλογές, η κάθε μία επιλογή επικοινωνεί με την αντίστοιχη δική της κλάση(HomeFragment, DashboardFragment, ProfilFragment) και σχηματίζει το δικό της περιεχόμενο. Αξιοσημείωτο είναι, πως με την υλοποίηση του μενού, ο χρήστης σε αυτό το σημείο, μπορεί να πηγαινοέρχεται από την μια επιλογή στην άλλη. Συνεχίζοντας, οι 2 πρώτες επιλογές (HomeFragment, DashboardFragment) έχουν να κάνουν με το κύριο θέμα της εφαρμογής, την διαχείριση δηλαδή των δεμάτων. Οι κλάσεις, που βρίσκονται από κάτω, σύμφωνα με το διάγραμμα (DetailsTabActivity, DetailsFragment, MapFragment, AllPackagesFragment, CompletedPackagesFragment, UncompletedPackagesFragment, Details2Fragment), διασχίζονται όταν ο χρήστης θέλει να δει τα διαθέσιμα δέματα , δέματα που έχει παραδώσει, δέματα που έχει αναλάβει αλλά δεν έχει παραδώσει ακόμη, καθώς και επιπλέον πληροφορίες για αυτά. Η Τρίτη επιλογή έχει να

κάνει με το προφίλ του χρήστη. Η διαχείριση των προσωπικών του στοιχείων γίνεται μέσω της κλάσης ProfilDetailsFragment. Από εδώ, αν ο χρήστης επιθυμεί να αλλάξει τον κωδικό πρόσβασης του, έρχεται σε επικοινωνία με την κλάση ForgotPasswordActivity.

Προηγουμένως έγινε γρήγορη αναφορά στον ρόλο διαχειριστή, και στην αναβάθμιση ενός λογαριασμού. Συνδεόμενος κάποιος με δικαιώματα διαχειριστή, καλείται η κλάση AdminActivity, σχηματίζοντας παράλληλα μια οθόνη με δύο κουμπιά. Το πρώτο κουμπί έχει να κάνει με τον λογαριασμό κάποιου χρήστη/οδηγού. Σε περίπτωση επιλογής του καλείται η κλάση AdminDriverActivity, η οποία με την σειρά της σχηματίζει οθόνη ξανά με 2 κουμπιά, το ένα για αναβάθμιση και επεξεργασία λογαριασμού, που καλεί την κλάση UpdateDriverFragment , και το άλλο, για την προσθήκη ενός καινούργιου χρήστη-οδηγού που καλεί την κλάση AddDriverFragment. Πηγαίνοντας πίσω στην αρχική οθόνη του διαχειριστή, αν επιλεχθεί το δεύτερο κουμπί ενεργοποιείται η κλάση AdminPackageActivity. Ακριβώς όπως πριν, σχηματίζεται μια νέα οθόνη με τα δικά της κουμπιά. Το πρώτο αντιστοιχεί στην κλάση UpdatePackageFragment για την επεξεργασία των πληροφοριών ενός δέματος, ενώ το δεύτερο αντιστοιχεί στην κλάση AddPackageFragment που μέσω αυτής μπορεί να προστεθεί ένα νέο διαθέσιμο δέμα στου χρήστες-οδηγούς. Υπάρχει ακόμη ένα κουμπί για την διαγραφή εξ ολοκλήρου ενός δέματος και η κλάση που ενεργοποιείται κατά το πάτημα του είναι η DeletePackageFragment.

Κεφάλαιο 4ο: Σχεδιασμός και ανάπτυξη εφαρμογής

4.1 Εισαγωγή

Στο κεφάλαιο αυτό θα περιγραφεί ο σχεδιασμός και θα σχολιαστούν κύρια συστατικά και επιπρόσθετα θα δοθούν κομμάτια κώδικα που γράφθηκαν για την ανάπτυξη του έργου.

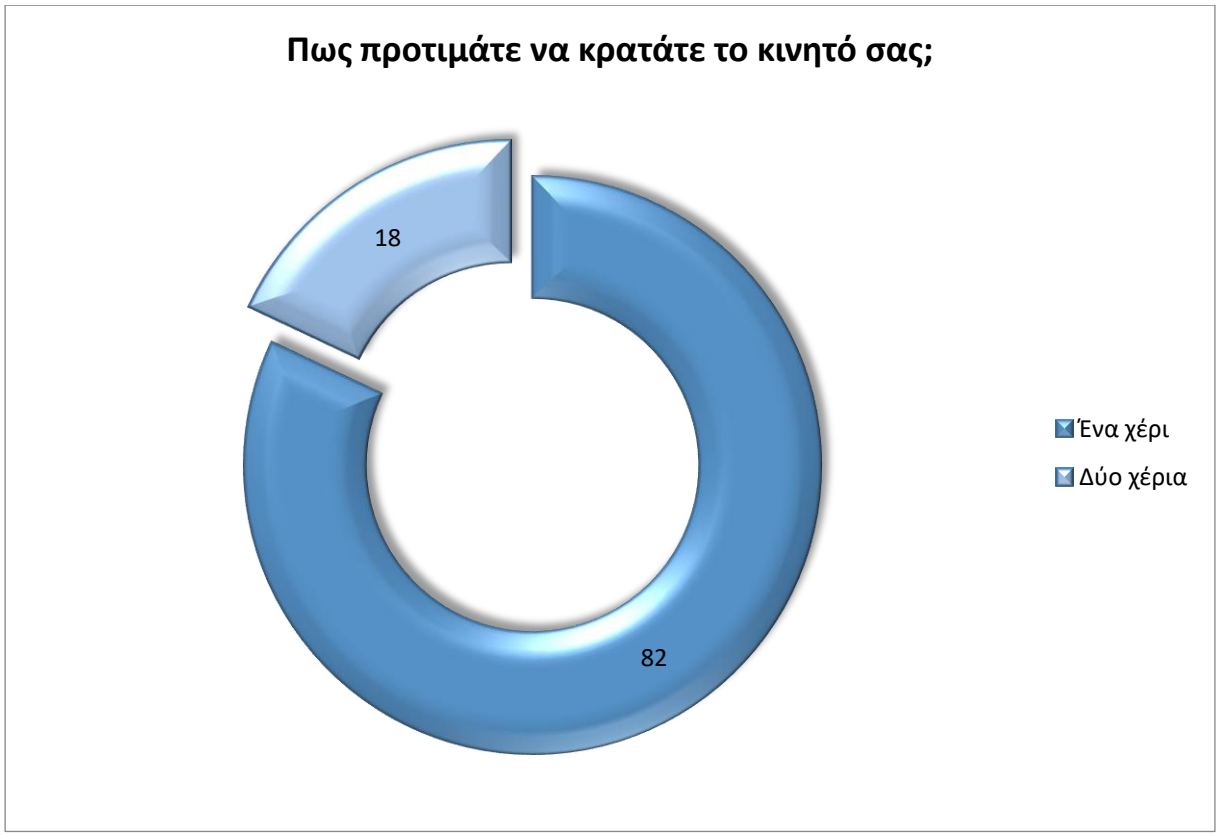
4.2 Σχεδιασμός

Όσον αφορά τον σχεδιασμό της εφαρμογής, όλες οι δραστηριότητές της (συμπεριλαμβανομένων και των οθονών όπως log in, log out και register) ξεκίνησαν ως Empty Activity. Κατά τη δημιουργία ενός νέου activity δημιουργούνται αυτόματα μία Java κλάση και ένα XML αρχείο όπου η κλάση είναι υπεύθυνη για το προγραμματιστικό κομμάτι της δραστηριότητας (κώδικας) ενώ το XML αρχείο είναι υπεύθυνο για τη διαμόρφωση της εμφάνισης της δραστηριότητας (Activity). Δεν θα έπρεπε να παραλειφθεί όμως το γεγονός ότι τη δεδομένη χρονική στιγμή υπάρχει ένα μεγάλο εύρος κινητών συσκευών (τόσο smartphones όσο και tablet) με συνέπεια να υπάρχει πληθώρα επιλογών τόσο στις αναλύσεις των οθονών όσο και στο μέγεθός τους. Επιπλέον, σημαντικό στοιχείο για τον σχεδιασμό μιας εφαρμογής αποτελεί και η προτίμηση στο κράτημα μιας συσκευής. Πως κρατάει ο χρήστης το κινητό του συνήθως; Με ένα χέρι, η με δύο ; Σε οριζόντια η σε κάθετη θέση; Για την επίλυση των ερωτημάτων αυτών, διεξήχθη μια έρευνα.

4.2.1 Η έρευνα

Για τον σκοπό της έρευνας αναπτύχθηκε ένα ερωτηματολόγιο. Το ερωτηματολόγιο απάντησαν σύνολο 100 άτομα, οι ηλικίες των οποίων ήταν από 20 έως 50. Ερωτήθηκαν φοιτητές, δημόσιοι και ιδιωτικοί υπάλληλοι, καθώς και ελεύθεροι επαγγελματίες που ανήκουν και στα δύο φύλα. Οι ερωτήσεις που απάντησαν ήταν γενικού περιεχομένου για τις προτιμήσεις και τις συνήθειες όσον αφορά την αλληλεπίδραση τους με την κινητή συσκευή τους. Πιο συγκεκριμένα αντιστοιχούν σε 3 κατηγορίες και θα αναλυθούν μαζί με τα αποτελέσματα παρακάτω.

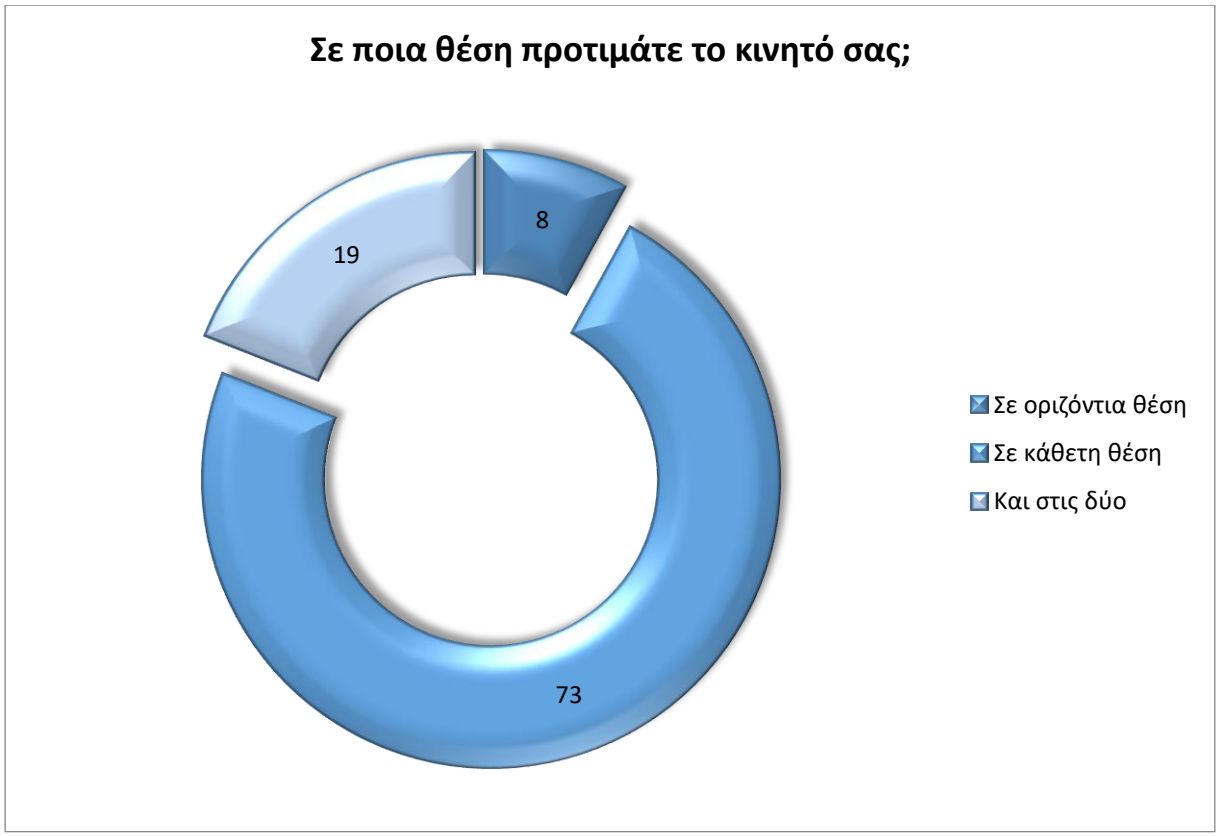
4.2.1.1 Κράτημα



Εικόνα 4.1: Ερώτηση 1: Πως προτιμάτε να κρατάτε το κινητό σας;

Η ερώτηση που τέθηκε σχετικά με το κράτημα της συσκευής, είναι μείζονος σημασίας για τον σχεδιασμό εύχρηστης εφαρμογής. Με βάση τα αποτελέσματα που φαίνονται στην Εικόνα 4.1 ήταν αρκετά ξεκάθαρο, πως η εφαρμογή πρέπει να σχεδιαστεί ,όσο το δυνατόν γίνεται, πιο φιλική για την χρήση της με το ένα χέρι. Φαίνεται πως ξεπερνούν το 80 % εκείνοι που προτιμούν να κάνουν χρήση του κινητού με το ένα χέρι.

4.2.1.2 Θέση



Εικόνα 4.2: Ερώτηση 2: Πως προτιμάτε να κρατάτε το κινητό σας;

Κατά την χρήση του κινητού, ανάλογα την αλληλεπίδραση με αυτό, προτιμάτε και διαφορετική θέση. Πιο συγκεκριμένα στις Android συσκευές υπάρχει η δυνατότητα για προβολή της εφαρμογής είτε σε κάθετη θέση (portrait), είτε σε οριζόντια θέση (landscape). Είναι εφικτό μια εφαρμογή να σχεδιαστεί και για τις δύο περιπτώσεις ή μόνο για μία από τις δύο. Συνεπώς, τέθηκε το ερώτημα, ποια θέση προτιμάτε. Ως απαντήσεις υπήρχαν 3 επιλογές. Η πρώτη είναι για την οριζόντια θέση, η δεύτερη για την κάθετη θέση, ενώ η τρίτη είναι για εκείνους που δεν έχουν σαφής απάντηση. Τα αποτελέσματα παρουσιάζονται στην Εικόνα 4.2. Το 73% προτιμά ξεκάθαρα την κάθετη θέση και μόλις το 8% ξεκάθαρα την οριζόντια θέση, ενώ υπάρχει και ένα 19% που αρέσκεται και με τις δύο θέσεις. Ερμηνεύοντας τα παραπάνω, η απόφαση ήταν μονόδρομος προς την κάθετη θέση(portrait).

4.2.1.3 Γλώσσα

Εικόνα 4.3: Ερώτηση 3: Σε ποια γλώσσα προτιμάτε να είναι οι εφαρμογές σας;

Το τρίτο και τελευταίο ερώτημα που τέθηκε ήταν σχετικό με την γλώσσα που θα επιλεχθεί. Είναι από τα σημαντικότερα ερωτήματα που πρέπει να απαντηθούν πριν τον σχεδιασμό μιας εφαρμογής. Στην Εικόνα 4.3 βλέπουμε πως τα αποτελέσματα είναι αρκετά κοντά. Δεν υπήρχε ξεκάθαρη κλήση σε μια από τις δύο απαντήσεις της ερώτησης. Με βάση αυτό και την ευκολία που παρέχει το Android Studio, για την ανάπτυξη της εφαρμογής σε πολλαπλές γλώσσες, αποφασίστηκε η ανάπτυξη της εφαρμογής και στην Ελληνική και στην Αγγλική γλώσσα.

4.2.2 Ο κώδικας

Σύμφωνα με την σχετική έρευνα, που αναλύθηκε προηγουμένως καταβλήθηκε προσπάθεια ώστε η εφαρμογή πρώτον, να είναι εύχρηστη έχοντας την συσκευή σε οριζόντια θέση και δεύτερον, ο χρήστης να νιώθει άνετα, χρησιμοποιώντας την με το ένα χέρι και διαβάζοντας την στην γλώσσα που επιθυμεί.

Με την προσθήκη της γραμμής που φαίνεται στην Εικόνα 4.4, σε κάθε activity που υπάρχει στο αρχείο AndroidManifest.xml αναγκάζουμε την εφαρμογή να είναι μόνιμα σε κάθετη θέση(portrait).

```
android:screenOrientation="portrait"
```

Εικόνα 4.4: Εντολή για την κάθετη θέση της εφαρμογής

Για τον προγραμματισμό της γλώσσας της εφαρμογής, χρειαζόμαστε τα αρχεία πόρου strings (string recourses). Ένα τέτοιο αρχείο παρέχει συμβολοσειρές κειμένου για την εφαρμογή σας με προαιρετικό στυλ και μορφοποίηση κειμένου. Στην Εικόνα 4.5 φαίνεται μέρος από το αρχείο με το κείμενο στην αγγλική γλώσσα ,ενώ στην Εικόνα 4.6 βλέπουμε μέρος από το αρχείο με το κείμενο στην ελληνική

γλώσσα. Ανάλογα με την γλώσσα της συσκευής, το σύστημα την αναγνωρίζει και φορτώνει το αντίστοιχο αρχείο. Η default γλώσσα της εφαρμογής είναι η Ελληνική.

```
<resources>
    <string name="app_name">Cdouk</string>
    <string name="title_activity_driver_main">MainActivity</string>
    <string name="title_home">Home</string>
    <string name="title_dashboard">My packages</string>
    <string name="title_notifications">Profile</string>
    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string-array name="textAddDriver">
        <item>email</item>
        <item>Password</item>
        <item>First Name</item>
        <item>Last Name</item>
        <item>Address</item>
        <item>Number</item>
    </string-array>
    <string-array name="textAddPackage">
        <item>Name</item>
        <item>DeliveryFrom-Address</item>
        <item>ShippingTo-Address</item>
        <item>Weight</item>
        <item>Price</item>
        <item>Length</item>
        <item>Width</item>
        <item>Heigt</item>
    </string-array>
    <string name="title_activity_details">DetailsActivity</string>
    <string name="hello_world">Hello Square World!</string>
    <string name="title_activity_maps">Map</string>
    <string name="before_start">Access Denied.
        To complete the registration, you must bring all the documents required to our offices. </string>
    <string name="password">Below is the email to which we will send a link to change your password</string>
    <string name="driver">Driver</string>
    <string name="packagee">Package</string>
    <string name="exit">Exit</string>
    <string name="update">Update</string>
    <string name="add">Add</string>
    <string name="change">Change</string>
    <string name="delete">Delete</string>
```

Εικόνα 4.5: strings.xml (En)

```xml
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="app_name">Cdouk</string>
    <string name="title_activity_driver_main">MainActivity</string>
    <string name="title_home">Αρχική</string>
    <string name="title_dashboard">Τα Δέματα μου</string>
    <string name="title_notifications">Προφίλ</string>
    <string name="hello_blank_fragment">Hello blank fragment</string>
    <string-array name="textAddDriver">
        <item>email</item>
        <item>Κωδικός</item>
        <item>Όνομα</item>
        <item>Επίθετο</item>
        <item>Διεύθυνση</item>
        <item>Τηλέφωνο</item>
    </string-array>
    <string-array name="textAddPackage">
        <item>Όνομα Παραλήπτη</item>
        <item>Διεύθυνση Παραλαβής</item>
        <item>Διεύθυνση Αποστολής</item>
        <item>Βάρος</item>
        <item>Κόστος</item>
        <item>Μήκος</item>
        <item>Πλάτος</item>
        <item>Υψος</item>
    </string-array>
    <string name="title_activity_details">DetailsActivity</string>
    <string name="hello_world">Hello Square World!</string>
    <string name="title_activity_maps">Map</string>
    <string name="before_start">Δεν επιτρέπεται ακόμη η πρόσβαση στην εφαρμογή.
        Για την ολοκλήρωση της εγγραφής πρέπει να παρουσιαστείται με τα απουτούμε έγγραφα στα γραφεία μας. </string>
    <string name="password">Παρακάτω είναι το email στο οποίο θα στείλουμε link για την αλλαγή του κωδικού σας</string>
    <string name="driver">Οδηγός</string>
    <string name="packagee">Δέμα</string>
    <string name="exit">Εξοδος</string>
    <string name="update">Αναβαθμιση</string>
    <string name="add">Προσθήκη</string>
    <string name="change">Επεξεργασία</string>
    <string name="delete">Διαγραφή</string>
```

Εικόνα 4.6: strings.xml (El)

4.3 Ανάπτυξη εφαρμογής

4.3.1 Gradle

Το Android Studio χρησιμοποιεί το Gradle [5], ένα προηγμένο kit εργαλείων κατασκευής, για την αυτοματοποίηση και τη διαχείριση της διαδικασίας build. Κάθε διαμόρφωση της διαδικασίας μπορεί να ορίσει το δικό της σύνολο κώδικα και πόρων, ενώ μπορεί να επαναχρησιμοποιεί τα κοινά μέρη σε όλες τις εκδόσεις της εφαρμογής. Στο Grandle πραγματοποιούνται διάφορες ρυθμίσεις, όπως η ελάχιστη, μέγιστη και στοχευμένη έκδοση SDK που θα τρέχει η αντίστοιχη εφαρμογή, καθώς και η έκδοση της JAVA. Επίσης, στο Grandle λαμβάνει χώρα ο κώδικας για την προσθήκη εξωτερικής βιβλιοθήκης και για τον ορισμός του ProGuard [11], που είναι υπεύθυνο για την αφαίρεση αχρησιμοποίητων κλάσεων, μεθόδων, πεδίων και παραμέτρων για την διαμόρφωση της release έκδοσης της εφαρμογής.

```
dependencies {
    //By using the Firebase Android BoM, your app will always use
    // compatible versions of the Firebase Android libraries.
    implementation platform('com.google.firebase:firebase-bom:31.1.1')
    implementation 'com.google.firebase:firebase-firestore'
    implementation 'com.google.firebase:firebase-auth'
    implementation 'com.google.firebase:firebase-messaging'
    implementation 'com.google.firebase:firebase-core'
    implementation 'com.google.firebase:firebase-functions'
    implementation 'com.google.firebase:firebase-storage'
```

Εικόνα 4.7: Grandle. Εξωτερική βιβλιοθήκη

Στην Εικόνα 4.7 είναι εμφανές πως η δήλωση μιας εξωτερικής βιβλιοθήκης είναι αρκετά εύκολη. Αρκεί η εντολή implementation σε σειρά με το αντίστοιχο πακέτο και την επιθυμητή έκδοση της βιβλιοθήκης. Στην Εικόνα 4.8 φαίνεται ο απαιτούμενος κώδικας για τον ορισμό του ProGuard.

```
buildTypes {
    release {
        minifyEnabled false
        proguardFiles getDefaultProguardFile('proguard-android-optimize.txt'), 'proguard-rules.pro'
    }
}
```

Εικόνα 4.8 : Grandle. ProGuard

4.3.2 Android Manifest

Κάθε Android εφαρμογή πρέπει να έχει ένα AndroidManifest.xml αρχείο με ακριβώς αυτό το όνομα. Το αρχείο αυτό, περιγράφει βασικές πληροφορίες που αφορούν την εφαρμογή σας, στα Android εργαλεία , στο λειτουργικό σύστημα Android και στο Google Play. Στο Android Studio κατά την δημιουργία της εφαρμογής, το αρχείο δημιουργείται αυτόματα και τα περισσότερα από τα βασικά στοιχεία του προστίθενται κατά την διάρκεια δημιουργίας και διαμόρφωσης της εφαρμογή.

Μεταξύ πολλών άλλων πραγμάτων, το αρχείο απαιτείται να δηλώνει τα εξής:

- Τα στοιχεία της εφαρμογής. Περιλαμβάνει ονομαστικά όλα τα activities και ορίζει το αρχικό activity όταν ανοίγει η εφαρμογή και τις υπηρεσίες. Κάθε στοιχείο πρέπει να ορίζει βασικές ιδιότητες, όπως φαίνεται και στην Εικόνα 4.9 το όνομα της κλάσης (name).
- Τις άδειες και τα δικαιώματα που χρειάζεται η εφαρμογή για πρόσβαση σε προστατευμένα μέρη του συστήματος ή άλλων εφαρμογών. Αφορούν, κυρίως, τη χρήση διαδικτύου, κάμερας, μικροφώνου κ.α. Δηλώνει επίσης τυχόν δικαιώματα που πρέπει να έχουν άλλες εφαρμογές εάν θέλουν να έχουν πρόσβαση σε περιεχόμενο από αυτήν την εφαρμογή. Στη συγκεκριμένη εφαρμογή οι άδειες που χρειάστηκαν φαίνονται στην Εικόνα 4.9 στις σειρές 4,5,6,7,8.
- Τα χαρακτηριστικά υλικού και λογισμικού που απαιτεί η εφαρμογή, τα οποία επηρεάζουν τις συσκευές που μπορούν να εγκαταστήσουν την εφαρμογή από το Google Play

```xml
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:tools="http://schemas.android.com/tools"
    package="com.example.cdouk">
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.WAKE_LOCK" />
    <uses-permission android:name="android.permission.VIBRATE" />
    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Cdouk"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/Theme.Cdouk">
        <activity android:name=".ui.driverUi.home.DetailsTabActivity" android:screenOrientation="portrait"></activity>
        <meta-data...>
        <uses-library...>
        <meta-data...>
        <activity android:name=".ForgotPasswordActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.adminUi.driver.AdminDriverActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.adminUi.driver.ContainerDriverActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.adminUi.packagee.AdminPackageActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.adminUi.packagee.ContainerPackageActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.adminUi.AdminActivity" android:screenOrientation="portrait"/>
        <activity android:name=".ui.driverUi.DriverActivity" android:label="MainActivity"
            android:screenOrientation="portrait"/>
        <activity android:name=".BeforeStart" />
        <activity android:name=".MainActivity" android:screenOrientation="portrait">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <service
            android:name=".notifications.FirebaseMessagingService"
            android:permission="TODO"
            tools:ignore="ExportedService">
            <intent-filter...>
        </service>
    </application>
</manifest>
```

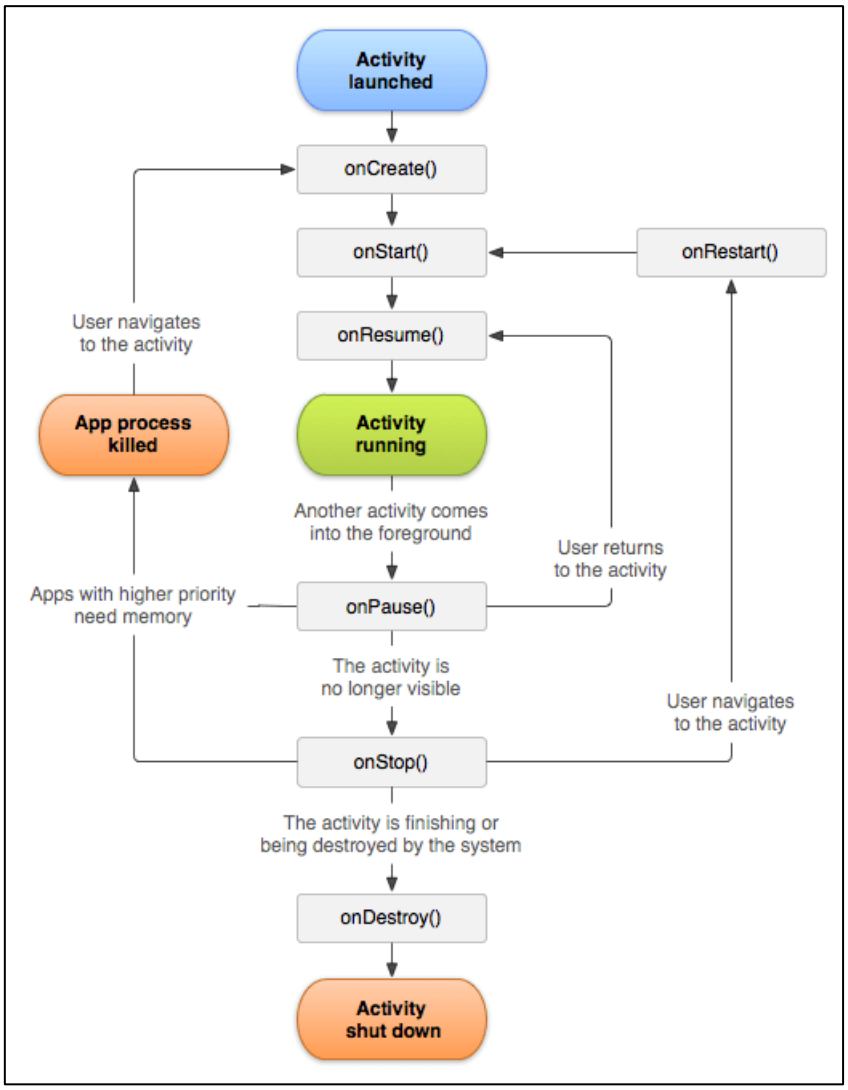
Εικόνα 4.9: AndroidManifest.xml

4.3.3 Activity και Fragment

4.3.3.1 Activity

Η κλάση Activity είναι ένα κρίσιμο συστατικό μιας εφαρμογής Android [14]. Ο τρόπος με τον οποίο ξεκινούν να συνδυάζονται και να αλληλεπιδρούν τα Activities μεταξύ τους, αποτελεί θεμελιώδες μέρος του μοντέλου εφαρμογής της πλατφόρμας.

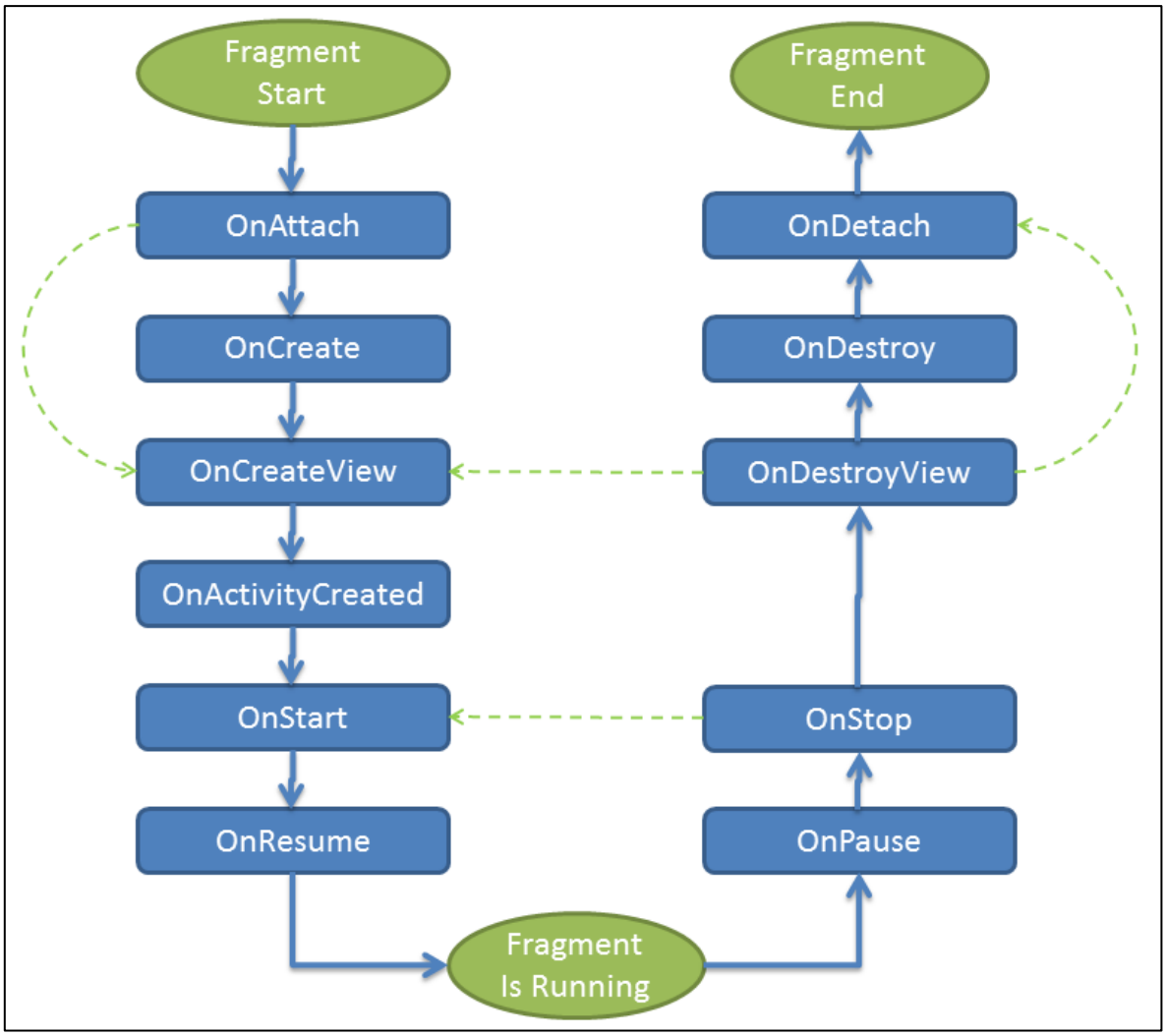
Σε αντίθεση με άλλα προγραμματιστικά παραδείγματα στα οποία οι εφαρμογές ξεκινούν με μια κύρια μέθοδο main(),μία Android εφαρμογή ξεκινά καλώντας τον κώδικα από ένα Activity. Καλεί συγκεκριμένες μεθόδους που αντιστοιχούν σε συγκεκριμένα στάδια του κύκλου ζωής ενός Activity (Εικόνα 4.10).



Εικόνα 4.10: Android Activity Lifecycle

4.3.3.2 Fragment

Ένα Fragment αντιπροσωπεύει ένα επαναχρησιμοποιήσιμο τμήμα της διεπαφής χρήστη (UI) μιας εφαρμογής [14]. Καθορίζει και διαχειρίζεται τη δική του διεπαφή χρήστη, έχει τον δικό του κύκλο ζωής (lifecycle) (Εικόνα 4.11) και μπορεί να χειριστεί δικά του συμβάντα εισόδου. Τα Fragments δεν μπορούν να υπάρξουν μόνα τους, πρέπει να «φιλοξενούνται» από ένα Activity ή από ένα άλλο Fragment.



Εικόνα 4.11: Android Fragment Lifecycle

Για την επαναχρησιμοποίηση των fragments, πρέπει να δημιουργηθούν το καθένα ως ένα εντελώς αυτοτελές στοιχείο που καθορίζει τη δική του διάταξη και συμπεριφορά. Αφού καθοριστούν, στην συνέχεια μπορούν να συσχετιστούν με ένα activity και εν τέλει να ακολουθούν τη λογική της εφαρμογής, ώστε να ολοκληρωθεί η συνολική σύνθετη διεπαφή χρήστη.

4.3.3.3 Τρόποι επικοινωνίας

Συχνά χρειάζεται η μεταφορά δεδομένων και η επικοινωνία μεταξύ activity και των fragments του ή μεταξύ δύο η περισσοτέρων fragments. Για να παραμείνουν όλα τα στοιχεία αυτοτελή δεν πρέπει να γίνεται απευθείας επικοινωνία. Λύση έρχεται να δώσει η βιβλιοθήκη Fragment που παρέχει την επιλογή, ενός κοινόχρηστου ViewModel. Η χρήση του Bundle προσφέρει ακόμη μια λύση. Η προτεινόμενη επιλογή εξαρτάται από την περίπτωση χρήσης.

```
5   public class ProfilDetailsViewModel extends ViewModel {
6       private String onoma;
7       private String epitheto;
8       private String dieuthinsh;
9       private String tel;
10      private int vathmos;
11      private boolean admin;
12      private boolean adeia;
13      private String email;
14      public ProfilDetailsViewModel() {
15      }
16      //Getters
17      public String getOnoma() { return onoma; }
20      public String getEpitheto() { return epitheto; }
23      public String getDieuthinsh() { return dieuthinsh; }
26      public String getTel() { return tel; }
29      public int getVathmos() { return vathmos; }
32      public boolean isAdmin() { return admin; }
35      public boolean isAdeia() { return adeia; }
38      public String getEmail() { return email; }
41      //Setters
42      public void setOnoma(String onoma) { this.onoma = onoma; }
45      public void setEpitheto(String epitheto) { this.epitheto = epitheto; }
48      public void setDieuthinsh(String dieuthinsh) { this.dieuthinsh = dieuthinsh; }
51      public void setTel(String tel) { this.tel = tel; }
54      public void setVathmos(int vathmos) { this.vathmos = vathmos; }
57      public void setAdmin(boolean admin) { this.admin = admin; }
60      public void setAdeia(boolean adeia) { this.adeia = adeia; }
63      public void setEmail(String email) { this.email = email; }
66
67  }
```

Εικόνα 4.12: Παράδειγμα υλοποίησης ViewModel

Σε ορισμένες περιπτώσεις, ίσως χρειαστεί να μοιραστούν δεδομένα μεταξύ των fragments και του activity. Για παράδειγμα, μπορεί να χρειάζεται να αλλαχθεί ένα στοιχείο καθολικής διεπαφής χρήστη με βάση μια αλληλεπίδραση μέσα σε ένα τμήμα. Στο παράδειγμα της Εικόνα 4.12 τα δεδομένα αποθηκεύονται σε ιδιωτικές μεταβλητές στις γραμμές 6 έως και 13. Η πρόσβαση σε αυτά γίνεται μέσω των μεθόδων getters και setters που υλοποιούνται στις γραμμές 17 έως και 38 για τις getters και 42 έως και 63 για τις setters αντίστοιχα.

```
ProfilDetailsViewModel profilDetailsViewModel= new ViewModelProvider( owner: this).get(ProfilDetailsViewModel.class);
```

Εικόνα 4.13 :Παράδειγμα δημιουργίας Αντικειμένου View Model

```
98      profilDetailsViewModel.setOnoma(user.getOnoma());
99      onoma.setText(profilDetailsViewModel.getOnoma());
100
```

Εικόνα 4.14: Παράδειγμα χρήσης ViewModel

Με την εντολή της Εικόνα 4.13, γραμμένη σε κλάση που επεκτείνει την κλάση Fragment, στην μέθοδο onViewCreated() δημιουργείται ένα αντικείμενο της κλάσης ProfilDetailsViewModel. Στην συνέχεια χρησιμοποιώντας τις setters του αντικειμένου ,όπως στην Εικόνα 4.14, γραμμή 98 (setOnoma(…)), γεμίζουν τα πεδία του ViewModel με τα αντίστοιχα δεδομένα. Έπειτα, καλώντας τις getters του, όπως στην γραμμή 99 (getOnoma()), τα αποθηκευμένα δεδομένα χρησιμοποιούνται για την παρουσία τους. Το κύριο πλεονέκτημά του ViewModel είναι ότι αποθηκεύει προσωρινά τα δεδομένα που εμφανίζονται στην διεπαφή και τα διατηρεί κατά την πλοήγηση μεταξύ activities και fragments. Αυτό σημαίνει ότι η διεπαφή χρήστη δεν χρειάζεται να ανακτήσει ξανά τα δεδομένα για παράδειγμα κατά την περιστροφή της οθόνης.

Σε κάποιες περιπτώσεις χρειάζεται να μεταφερθεί μια τιμή, για μια μόνο φορά. Τότε, είναι πιο εύκολη και εύχρηστη η επιλογή του Bundle. Είναι ένας τρόπος αποθήκευσης ζευγών σε μορφή κλειδί/τιμή.

```
125         DetailsFragment detailsFragment = new DetailsFragment();
126         Bundle args = new Bundle();
127         args.putParcelable("package", homeViewModel.getClickedItem());
128         args.putString("id",id);
129         detailsFragment.setArguments(args);
```

Εικόνα 4.15: Παράδειγμα χρήσης Bundle. Αποστολή δεδομένων

Στην Εικόνα 4.15, στην γραμμή 125 δημιουργείται το Fragment που θέλουμε να στείλουμε τα δεδομένα. Στην γραμμή 126 δημιουργείται το στιγμιότυπο του Bundle. Στην 127 και 128 δημιουργούνται δύο ζευγάρια της μορφής κλειδί/τιμή τα δεδομένα. Στην 129, στέλνονται στο επιθυμητό fragment.

```
87          Bundle args= this.getArguments();
88          if(args!=null) {
89              Packagee p = args.getParcelable( key: "package");
90              String id = args.getString( key: "id");
91              fillTexts(p,id);
92          }
```

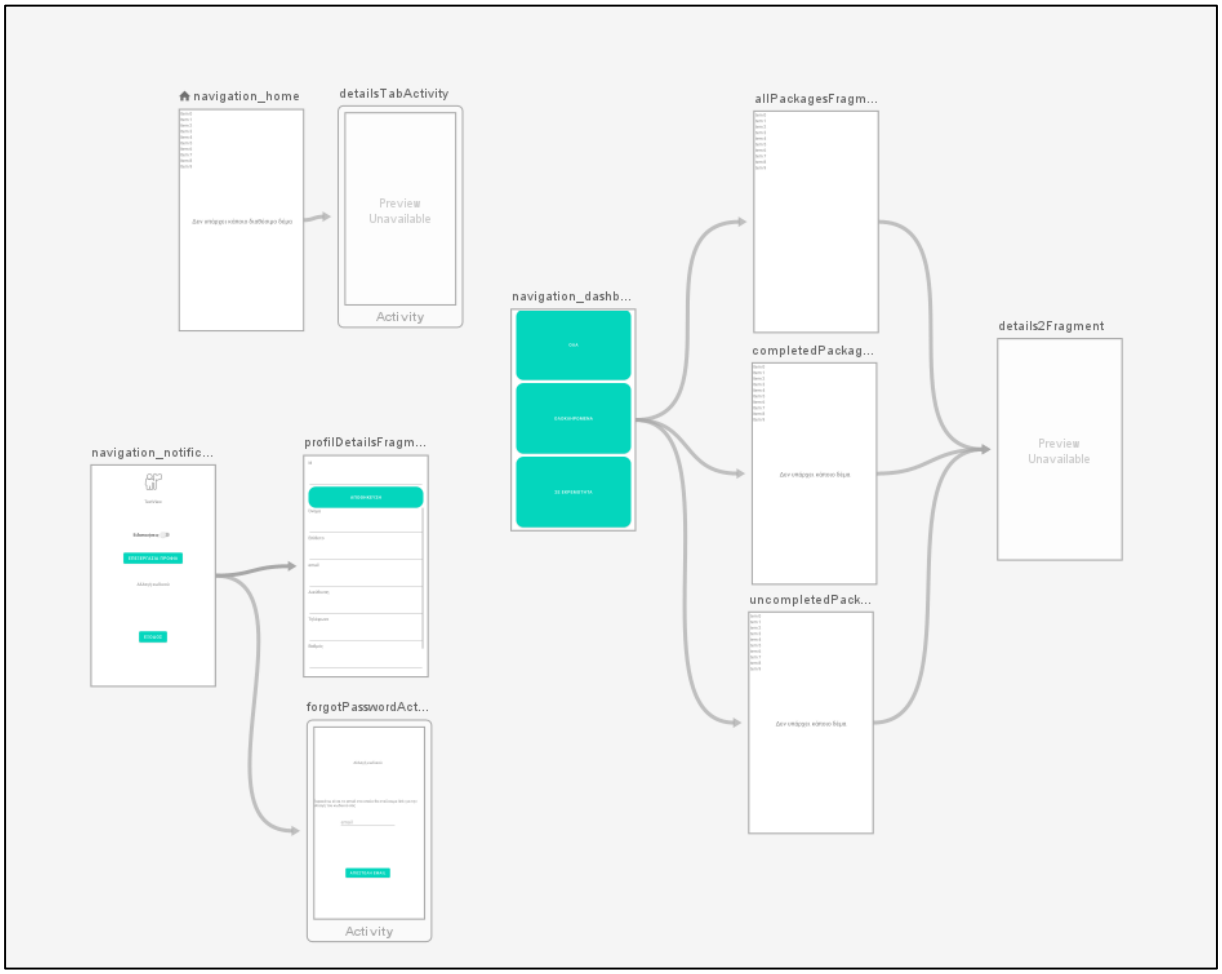
Εικόνα 4.16: Παράδειγμα χρήσης Bundle. Παραλαβή δεδομένων

Στην Εικόνα 4.16, στην γραμμή 87 δημιουργείται το αντίστοιχο στιγμιότυπο του Bundle. Αφού πρώτα ελεγχθεί αν το στιγμιότυπο είναι κενό, τότε στις γραμμές 89 και 90, παραλαμβάνονται οι αντίστοιχες τιμές στις μεταβλητές.

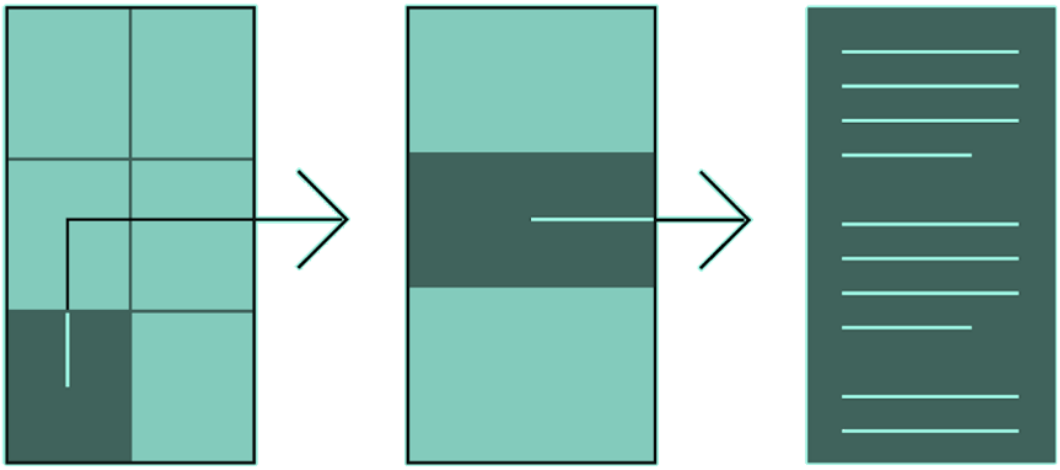
4.3.4 Navigation Component

Το Navigation Component [15] αναφέρεται στις αλληλεπιδράσεις που επιτρέπουν στους χρήστες να πλοηγούνται σε διάφορα σημεία μιας εφαρμογής. Αποτελείται από τρία βασικά μέρη, το Navigation Graph, το NavigationHost και το NavigationController.

4.3.4.1 Navigation Graph



Εικόνα 4.17: NavGraph



Εικόνα 4.18: Navigation Component

Στην Εικόνα 4.17 φαίνεται ένα από τα NavGraph της εφαρμογής. Το NavGraph είναι ένα αρχείο πόρων που περιέχει όλους τους προορισμούς και τις ενέργειές. Το γράφημα αντιπροσωπεύει όλες τις πιθανές διαδρομές πλοήγησης της εφαρμογής μεταξύ αυτών των fragments. Είναι αναγκαίο να οριστεί αρχικός προορισμός. Κάθε προορισμός αντιπροσωπεύεται από μια μικρογραφία προεπισκόπησης και οι ενέργειες σύνδεσης αντιπροσωπεύονται από βέλη που δείχνουν πώς οι χρήστες μπορούν να πλοηγηθούν από τον έναν προορισμό στον άλλο. Μέσω του Navigation Graph είναι εφικτό να προσθέσουμε κι άλλες λεπτομέρειες ,όπως το επιθυμητό animation κατά την μετάβαση από ένα Fragment σε ένα άλλο. Για την καλύτερη κατανόηση, στην Εικόνα 4.18, απεικονίζεται η μορφή που έχει κάθε πλοήγησης, που ξεκινάει από ένα αρχικός προορισμός και καταλήγει μέσω ενεργειών σύνδεσης, σε έναν τελικό προορισμό.

4.3.4.2 NavigationHost

Ένα από τα βασικά μέρη του Navigation Component είναι το NavHost. Το NavHost είναι ένα κενό κοντέινερ που εμφανίζει τους προορισμούς από το Navigation Graph. Όπως φαίνεται και στην Εικόνα 4.19 αρκεί ο ορισμός των πεδίων name με το NavHostFragment και navGraph με το επιθυμητό Nav Graph.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    android:id="@+id/container"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.material.bottomnavigation.BottomNavigationView
        android:id="@+id/nav_view"
        android:layout_width="0dp"
        android:layout_height="wrap_content"
        android:layout_marginStart="0dp"
        android:layout_marginEnd="0dp"
        android:background="?android:attr/windowBackground"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:menu="@menu/bottom_nav_menu" />

    <androidx.fragment.app.FragmentContainerView
        android:id="@+id/nav_host_fragment"
        android:name="androidx.navigation.fragment.NavHostFragment"
        android:layout_width="match_parent"
        android:layout_height="547dp"
        app:defaultNavHost="true"
        app:layout_constraintBottom_toTopOf="@id/nav_view"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintLeft_toLeftOf="parent"
        app:layout_constraintRight_toRightOf="parent"
        app:layout_constraintTop_toTopOf="parent"
        app:navGraph="@navigation/mobile_navigation" />

</androidx.constraintlayout.widget.ConstraintLayout>
```

Εικόνα 4.19: ActivityDriverUi

4.3.4.3 NavigationController

Τέλος, για την ολοκλήρωση του Navigation Component, απομένει η δημιουργία του NavController όπως βλέπουμε στην Εικόνα 4.20. Είναι το αντικείμενο που θα μας δώσει πρόσβαση προγραμματιστικά για την διαχείριση των μεταβάσεων μεταξύ των προορισμών.

Στην γραμμή 27 δημιουργήτε ένα νέο Builder χρησιμοποιώντας ένα μενού που περιέχει όλους τους προορισμούς ανώτατου επιπέδου. Στην γραμμή 30 δημιουργείται το αντικείμενο navController. Η γραμμή 31 ρυθμίζει το ActionBar που επιστρέφεται από το AppCompatActivity.getSupportActionBar για χρήση με το navController. Στην γραμμή 32 ρυθμίζεται ένα αντικείμενο BottomNavigation(navView) για χρήση με το αντικείμενο navController.

```
23        BottomNavigationView navView = findViewById(R.id.nav_view);
24        // Passing each menu ID as a set of Ids because each
25        // menu should be considered as top level destinations.
26
27        AppBarConfiguration appBarConfiguration = new AppBarConfiguration.Builder(
28                R.id.navigation_home, R.id.navigation_dashboard, R.id.navigation_notifications)
29                .build();
30        NavController navController = Navigation.findNavController( activity: this, R.id.nav_host_fragment);
31        NavigationUI.setupActionBarWithNavController( activity: this, navController, appBarConfiguration);
32        NavigationUI.setupWithNavController(navView, navController);
```

Εικόνα 4.20: NavController και Bottom Navigation

4.3.5 Bottom Navigation

Το Bottom Navigation αντιπροσωπεύει μια τυπική γραμμή πλοήγησης (toolbar) στο κάτω μέρος της οθόνης. Το Bottom Navigation διευκολύνει την εξερεύνηση και την περιήγηση στην εφαρμογή μεταξύ των κυρίων κατηγοριών με ένα μόνο πάτημα. Ο κάθε προορισμός της εφαρμογής αντιπροσωπεύεται από το αντίστοιχο αντικείμενο (item). Στην Εικόνα 4.20, στην γραμμή 23, δημιουργείτε το αντικείμενο navView και με την χρήση του findViewById θα αλληλεπιδρά με το στοιχείο διεπαφής με αναγνωριστικό όνομα ‘nav_view’. Στις επόμενες γραμμές, όπως αναφέρθηκε και σε προηγούμενο κεφάλαιο, συνδυάζονται Bottom Navigation με έναν NavController.

```
2   <menu xmlns:android="http://schemas.android.com/apk/res/android">
3
4       <item
5           android:id="@+id/navigation_home"
6           android:icon="@drawable/ic_home_black_24dp"
7           android:title="Αρχική" />
8
9       <item
10          android:id="@+id/navigation_dashboard"
11          android:icon="@drawable/ic_dashboard_black_24dp"
12          android:title="Τα Δέματα μου" />
13
14      <item
15          android:id="@+id/navigation_notifications"
16          android:icon="@drawable/ic_notifications_black_24dp"
17          android:title="Προφίλ" />
18
19  </menu>
```

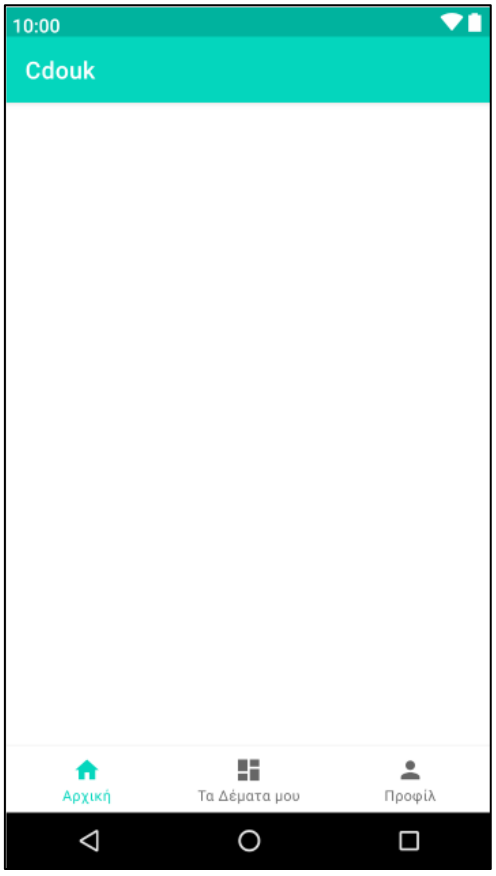
Εικόνα 4.21: Παράδειγμα δημιουργίας Μενού

```
<com.google.android.material.bottomnavigation.BottomNavigationView
    android:id="@+id/nav_view"
    android:layout_width="0dp"
    android:layout_height="wrap_content"
    android:layout_marginStart="0dp"
    android:layout_marginEnd="0dp"
    android:background="?android:attr/windowBackground"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintLeft_toLeftOf="parent"
    app:layout_constraintRight_toRightOf="parent"
    app:menu="@menu/bottom_nav_menu" />
```

Εικόνα 4.22: Παράδειγμα BottomNavigation

Ο τρόπος προσθήκης ενός προορισμού (item) είναι ίδιος με αυτόν ενός toolbar. Δημιουργείτε πρώτα ένα menu (Εικόνα 4.21), και έπειτα συσχετίζεται με το BottomNavigation. Στην Εικόνα 4.22, στην τελευταία γραμμή του κώδικα φαίνεται η αντίστοιχη συσχέτιση.

Έπειτα από τα παραπάνω, το τελικό αποτέλεσμα του Bottom Navigation της εφαρμογής φαίνεται στην Εικόνα 4.23



Εικόνα 4.23: Διεπαφή χρήστη του Bottom Navigation

4.3.6 RecyclerView

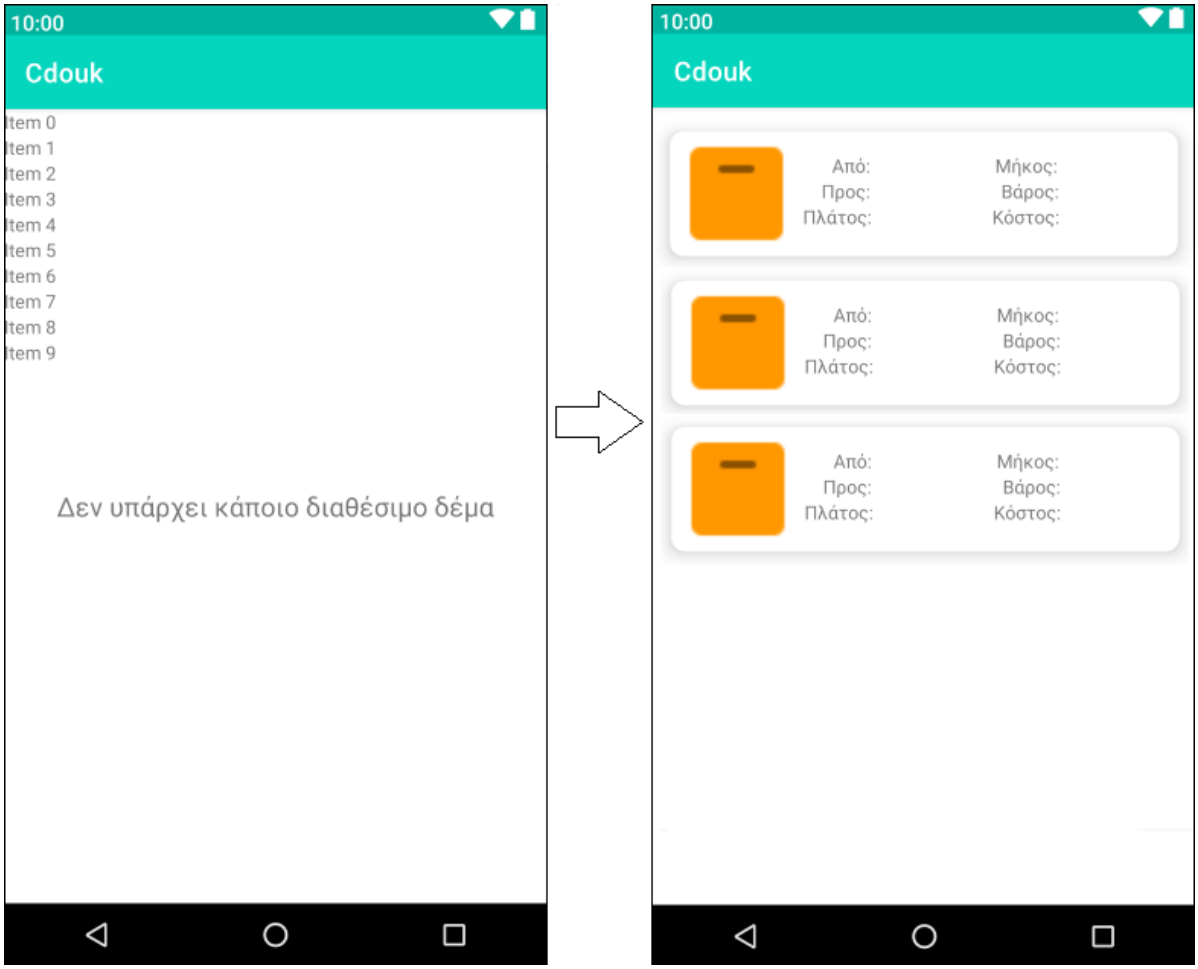
Το RecyclerView [16] είναι ένα ViewGroup που περιέχει προβολές που αντιστοιχούν σε δεδομένα. Είναι μια προβολή από μόνη του, επομένως προστίθεται στη διάταξή, με τον τρόπο που θα γινόταν η προσθήκη οποιουδήποτε άλλου στοιχείου διεπαφής χρήστη. Κάθε μεμονωμένο στοιχείο στη λίστα του recyclerView ορίζεται από ένα αντικείμενο viewHolder. Όπως προδίδει και το όνομα του, το RecyclerView ανακυκλώνει τα μεμονωμένα στοιχεία του. Στην περίπτωση ,δηλαδή, που ένα στοιχείο μετακινηθεί εκτός οθόνης, το RecyclerView δεν καταστρέφει την προβολή του, αλλά επαναχρησιμοποιεί την προβολή για να προβάλει τα νέα στοιχεία που έχουν μετακινηθεί στην οθόνη.

```
17  public class HomeRecyclerAdapter extends RecyclerView.Adapter<HomeRecyclerAdapter.Holder> {
18
19      Context context;
20      private ArrayList<Packagee> packages = new ArrayList<>();
21      private ItemClickListener itemClickListener;
22
23      public HomeRecyclerAdapter(Context context, ArrayList<Packagee> p, ItemClickListener itemClickListener) {...}
28
29      @NonNull
30      @Override
31      public Holder onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
32          LayoutInflater inflater = LayoutInflater.from(context);
33          View view = inflater.inflate(R.layout.my_row, parent,  attachToRoot: false);
34          return new Holder(view);
35      }
36
37      @Override
38      public void onBindViewHolder(@NonNull Holder holder, int position) {
39
40          holder.txt1.setText(packages.get(position).getDieuthinsh_paralavhs());
41          holder.txt2.setText(packages.get(position).getDieuthinsh_apostolhs());
42          holder.txt3.setText(""+packages.get(position).getKostos());
43
44          holder.itemView.setOnClickListener(view -> {
45              itemClickListener.onItemClicked(packages.get(position),position);
46          });
47      }
48
49      @Override
50      public int getItemCount() { return packages.size(); }
53
54      public interface ItemClickListener{
55          void onItemClicked(Packagee p,int position);
56      }
57      public class Holder extends RecyclerView.ViewHolder {
58          TextView txt1, txt2, txt3;
59          public Holder(@NonNull View itemView) {...}
65      }
66  }
```

Εικόνα 4.24: Παράδειγμα κώδικα RecyclreViewAdapter

Ο ορισμός των δεδομένων στον RecyclerView πραγματοποιείται μέσω ενός RecyclerView.Adapter. Οι μέθοδοι onCreateViewHolder() και onBindViewHolder(), που φαίνονται στην Εικόνα 4.24 είναι σημαντικοί μέθοδοι και είναι αναγκαίο να υλοποιηθούν. Στην onCreateViewHolder() επιστρέφεται το ViewHolder αφού έχει δημιουργηθεί πρώτα το επιθυμητό view, στην γραμμή 33, το οποίο περνιέται στο ViewHolder ως παράμετρος. Η κλάση τύπου ViewHolder υλοποιείται μέσα στον ίδιο τον adapter (γραμμές 57 έως και 65). Στο παράδειγμα, το κάθε στοιχείο του recycler View λειτουργεί σαν κουμπί. Για τον λόγω αυτόν, ήταν απαραίτητη η χρήση του ClickListener. Η μέθοδος onBindViewHolder(),

γεμίζει κάποια πεδία του στιγμιότυπου Holder και θέτει τον ClickListener που είναι υπεύθυνος για την κλήση των απαραίτητων μεθόδων στην περίπτωση πατήματος ενός στοιχείου του RyceclerView. Η υλοποίηση του listener που πρέπει να επικοινωνούν με το Fragment από το οποίο καλέστηκε το adapter πραγματοποιείται μέσω interface και δίνετε στον RecyclerView.Adapter όταν δημιουργείτε μέσω παραμέτρου. Στην γραμμή 45 καλείτε η κατάλληλη μέθοδος του interface όπως και όπου χρειάζεται.



Εικόνα 4.25 Μορφή RecyclerView

Στην Εικόνα 4.25 απεικονίζεται στο αριστερό μέρος το στοιχείο του RecyclerView, όταν είναι άδειο, ενώ στο δεξί μέρος, όταν γεμίζει από τον κώδικα.

4.3.7 MapView

Ένα View που εμφανίζει έναν χάρτη (με δεδομένα που λαμβάνονται από την υπηρεσία Χαρτών Google). Το MapsSDK για Android παρέχει ποικιλία κλάσεων που μπορεί να χρησιμοποιήσει η εφαρμογή για τη καλύτερη δυνατή διαχείριση του κύκλου ζωής [17], της λειτουργικότητας και των δεδομένων ενός χάρτη. Οι κλάσεις υποστηρίζουν τις αλληλεπιδράσεις των χρηστών με βάση το μοντέλο διεπαφής χρήστη, όπως ο ορισμός της αρχικής κατάστασης του χάρτη και η ανταπόκριση στην εισαγωγή χειρονομιών από τον χρήστη κατά τη διάρκεια εκτέλεσης. Κατά την εστίαση σε αυτό, καταγράφει τα πλήκτρα και τις κινήσεις αφής για την μετακίνηση και πλοήγηση στον χάρτη.

Για την χρήση ενός αντικειμένου GoogleMap στην εφαρμογή, πρέπει να χρησιμοποιηθεί είτε ένα αντικείμενο SupportMapFragment είτε ένα αντικείμενο MapView, όπως στην Εικόνα 4.26, ως

αντικείμενο κοντέινερ για τον χάρτη και, στη συνέχεια, να ανακτηθεί το αντικείμενο GoogleMap από το κοντέινερ. Επειδή οι κλάσεις κοντέινερ προέρχονται είτε από ένα Android fragment η ένα view, παρέχουν στον χάρτη τη διαχείριση του κύκλου ζωής. Για την χρήση μιας τέτοιας κλάσης πρέπει να προωθηθούν όλες οι μέθοδοι κύκλου ζωής από το activity ή το fragment που περιέχει αυτήν την προβολή στις αντίστοιχες σε αυτήν την κλάση. Στο παράδειγμα της Εικόνα 4.27 φαίνεται καθαρά ο κώδικας από τις κάποιες από τις μεθόδους κύκλου ζωής.

```
<androidx.constraintlayout.widget.ConstraintLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    xmlns:app="http://schemas.android.com/apk/res-auto"
    xmlns:tools="http://schemas.android.com/tools"
    android:layout_width="match_parent"
    android:layout_height="match_parent">

    <com.google.android.gms.maps.MapView
        android:id="@+id/mapView"
        android:layout_width="match_parent"
        android:layout_height="match_parent"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</androidx.constraintlayout.widget.ConstraintLayout>
```

Εικόνα 4.26: Χρήση MapView

```
156     @Override
157     public void onResume() {
158         super.onResume();
159         mapView.onResume();
160     }
161
162     @Override
163     public void onStart() {
164         super.onStart();
165         mapView.onStart();
166     }
167
168     @Override
169     public void onStop() {
170         super.onStop();
171         mapView.onStop();
172     }
173
174
175     @Override
176     public void onPause() {
177         mapView.onPause();
178         super.onPause();
179     }
180
181     @Override
182     public void onDestroy() {
183         mapView.onDestroy();
184         super.onDestroy();
185     }
186
```

Εικόνα 4.27: Παράδειγμα Μεθόδων Κύκλου ζωής

Ένα GoogleMap πρέπει να αποκτηθεί χρησιμοποιώντας το getMapAsync(OnMapReady Callback). Το MapView αρχικοποιεί αυτόματα το σύστημα χαρτών και την προβολή.

Μόλις οριστεί μια παρουσία αυτής της διεπαφής, σε ένα αντικείμενο MapFragment ή MapView, ενεργοποιείται η μέθοδος onMapReady(GoogleMap) όταν ο χάρτης είναι έτοιμος για χρήση και παρέχει μια μη μηδενική παρουσία του GoogleMap. Εάν οι υπηρεσίες Google Play δεν είναι εγκατεστημένες στη συσκευή, θα ζητηθεί από τον χρήστη να την εγκαταστήσει και η μέθοδος onMapReady(GoogleMap) θα ενεργοποιηθεί μόνο όταν ο χρήστης την εγκαταστήσει και επιστρέψει στην εφαρμογή.

Στην Εικόνα 4.28, στην γραμμή 35, η κλάση κάνει implement την διεπαφή OnMapReadyCallBack. Στις γραμμές 80 έως και 102, παρουσιάζεται το σώμα της μεθόδου onMapReady(GoogleMap). Αφού πρώτα ελέγχει τις απαραίτητες άδειες, στην συνέχεια προσθέτει στον χάρτη 2 μαρκαρισμένες διευθύνσεις και τέλος τον σχηματίζει.

```java
public class MapsFragment extends Fragment implements OnMapReadyCallback {
    private GoogleMap mMap;
    private MapView mapView;
    private String dieuthinsh_paralavhs,dieuthinsh_apostolhs;
    private int height = 100;
    private int width = 100;
    Bitmap smallMarker2; Bitmap smallMarker;
    private static final String MAPVIEW_BUNDLE_KEY = "MapViewBundleKey";

    @Override
    public void onAttach(@NonNull Context context) { super.onAttach(context); }

    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle savedInstanceState) {...}

    @Override
    public void onMapReady(GoogleMap googleMap) {
        if (ActivityCompat.checkSelfPermission(getActivity(), Manifest.permission.ACCESS_FINE_LOCATION)
                != PackageManager.PERMISSION_GRANTED
                && ActivityCompat.checkSelfPermission(getActivity(), Manifest.permission.ACCESS_COARSE_LOCATION)
                != PackageManager.PERMISSION_GRANTED) {
            return;
        }
        mMap = googleMap;
        // Add markers
        LatLng paralavh = getLocationFromAddress(getActivity(), dieuthinsh_paralavhs);
        mMap.addMarker(new MarkerOptions().position(paralavh).title("Παραλαβή από"));
        mMap.moveCamera(CameraUpdateFactory.newLatLng(paralavh));

        LatLng apostolh = getLocationFromAddress(getActivity(), dieuthinsh_apostolhs);
        mMap.addMarker(new MarkerOptions().position(apostolh).title("Αποστολή σε"));

        LatLngBounds.Builder builder = new LatLngBounds.Builder();
        builder.include(paralavh);
        builder.include(apostolh);
        LatLngBounds bounds = builder.build();
        googleMap.animateCamera(CameraUpdateFactory.newLatLngBounds(bounds,  padding: 200));
        googleMap.setMyLocationEnabled(true);
    }
```

Εικόνα 4.28: OnMapReadyCallback

4.3.8 ViewPagerAdapter

Ο ViewPagerAdapter είναι ένας adapter για την διαχείριση διάταξης που επιτρέπει στο χρήστη να περιηγηθεί αριστερά και δεξιά σε σελίδες δεδομένων [18]. Το ViewPager χρησιμοποιείται συχνότερα σε συνδυασμό με το android.app.Fragment, το οποίο είναι ένας βολικός τρόπος παροχής και διαχείρισης του κύκλου ζωής κάθε σελίδας. Υπάρχουν adapters που εφαρμόζονται για τη χρήση fragments με το ViewPager, οι οποίοι καλύπτουν τις πιο συνηθισμένες περιπτώσεις χρήσης. Αυτά είναι τα androidx.fragment.app.FragmentPagerAdapter και androidx.fragment.app.FragmentStatePagerAdapter. Καθεμία από αυτές τις κλάσεις έχει απλό κώδικα που δείχνει την δημιουργία μιας πλήρη διεπαφής χρήστη.

Για να ρυθμιστεί μια διάταξή με το ViewPager, πρέπει να προστεθεί το στοιχείο <ViewPager> στη διάταξη XML. Στην εφαρμογή μας, ο κώδικας του XML αρχείου φαίνεται στην Εικόνα 4.29.

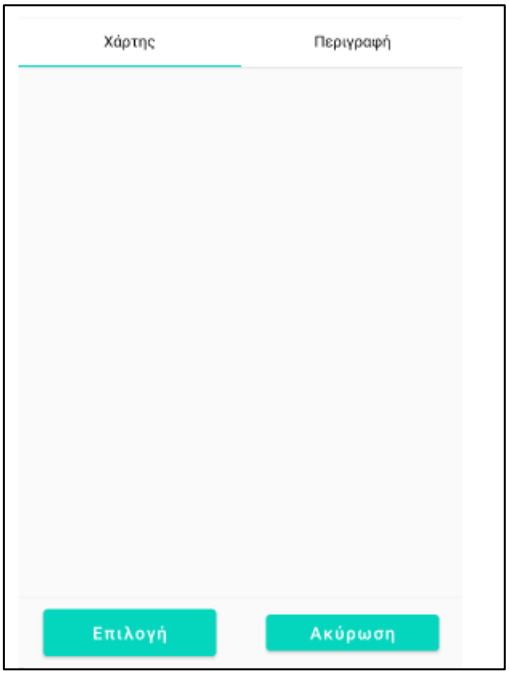
```
<androidx.viewpager.widget.ViewPager
    android:id="@+id/viewPager"
    android:layout_width="0dp"
    android:layout_height="0dp"
    app:layout_constraintBottom_toTopOf="@+id/linear_id"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintStart_toStartOf="parent"
    app:layout_constraintTop_toBottomOf="@+id/tabLayout2" >

</androidx.viewpager.widget.ViewPager>
```

Εικόνα 4.29: Παράδειγμα ViewPager

Στον παρακάτω κώδικα, Εικόνα 4.30, δημιουργείτε ένας ViewPagerAdapter που αποτελείται από δύο καρτέλες. Στις γραμμές 96 και 97, δημιουργούνται δύο στιγμιότυπα από δύο fragments και προστίθενται στον adapter. Ο κώδικας της μεθόδου addFragment βρίσκεται στην κλάση του ViewPagerAdapter και είναι αυτός στην Εικόνα 4.32. Τέλος, συσχετίζεται ο adapter με το στοιχείο διεπαφής viewPager, και μετά με το στοιχείο διεπαφής tabLayout και δημιουργείται το τελικό αποτέλεσμα που φαίνεται στην Εικόνα 4.31.

```
91      final ViewPagerAdapter viewPagerAdapter = new ViewPagerAdapter(getSupportFragmentManager()
92
93      new Handler().post(new Runnable() {
94          @Override
95          public void run() {
96              viewPagerAdapter.addFragment(MapsFragment.getInstance(), title: "Χάρτης");
97              viewPagerAdapter.addFragment(DetailsFragment.getInstance(), title: "Περιγραφή");
98              viewPager.setAdapter(viewPagerAdapter);
99              tabLayout.setupWithViewPager(viewPager);
100         }
101     });
```

Εικόνα 4.30: Παράδειγμα ViewAdapter με καρτέλες



Εικόνα 4.31: Διεπαφή χρήστη του viewPager με δύο καρτέλες

```
public void addFragment(Fragment fragment, String title){
    fragmentList.add(fragment);
    stringList.add(title);
}
```

Εικόνα 4.32: Μέθοδος στην κλάση ViewPagerAdapter

4.3.9 Push Notifications

Μια ειδοποίηση push είναι ένα μήνυμα που εμφανίζεται σε μια κινητή συσκευή [19]. Οι υπεύθυνοι εφαρμογών μπορούν να τις στείλουν ανά πάσα στιγμή, καθώς οι χρήστες δεν χρειάζεται να βρίσκονται στην εφαρμογή ή να χρησιμοποιούν τις συσκευές τους για να τις λαμβάνουν. Οι ειδοποιήσεις push μοιάζουν με μηνύματα κειμένου SMS και ειδοποιήσεις για κινητά, αλλά τις λαμβάνουν μόνο οι χρήστες που έχουν εγκαταστήσει την εκάστοτε εφαρμογή. Όλες οι πλατφόρμες κινητής τηλεφωνίας – iOS, Android, Fire OS, Windows και BlackBerry – έχουν τις δικές τους υπηρεσίες για υποστήριξη push ειδοποιήσεων.

Στην συγκεκριμένη εφαρμογή, χρησιμοποιήθηκαν οι ειδοποιήσεις push ,ώστε να ενημερώνεται ο χρήστης κάθε φορά που προστίθεται ένα δέμα στην βάση της εφαρμογής. Μέσω της εφαρμογής παρέχεται στον χρήστη η δυνατότητα να ενεργοποιεί και να απενεργοποιεί τις ειδοποιήσεις τύπου push. Στην Εικόνα 4.33 παρουσιάζεται ο κώδικας για την ενεργοποίηση και απενεργοποίηση των push ειδοποιήσεων.

```
private void buttonClicked(View view) {
    switch (view.getId()) {
        case R.id.allowNotifications: {
            if (allow.isChecked()) {
                FirebaseMessaging.getInstance().subscribeToTopic("NewPackage").addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText(getActivity(), text: "Εγγραφή : επιτυχής", Toast.LENGTH_SHORT).show();
                    }
                }
                );
                profilViewModel.setAllow(true);
            } else {
                FirebaseMessaging.getInstance().unsubscribeFromTopic("NewPackage").addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText(getActivity(), text: "Απεγγραφή : επιτυχής", Toast.LENGTH_SHORT).show();
                    }
                }
                );
                profilViewModel.setAllow(false);
            }
```

Εικόνα 4.33: Εγγραφή/Απεγγραφή στα push notifications

Για να γίνει εφικτό το παραπάνω πρέπει να συνδεθεί η εφαρμογή με την βάση δεδομένων Firebase και να γίνει το κατάλληλο set up για το cloud messaging.

Παρακάτω, στην Εικόνα 4.34 και στην Εικόνα 4.35 φαίνεται ο κώδικας που εκτελείται κάθε φορά που ένα δέμα προστίθεται στην βάση δεδομένων, για τον σχηματισμό και την αποστολή pushNotification.

```
private void sendMessage(){
    String title="Νέο δέμα";
    String body="Ένα νεο δέμα είναι τώρα διαθέσιμο ";
   FcmNotificationsSender notificationSender =
            new FcmNotificationsSender( userFcmToken: "/topics/NewPackage",title,body,getContext(),getActivity());
   notificationSender.SendNotifications();
}
```

Εικόνα 4.34: Μέθοδος για την αποστολή pushNotification

```
public void SendNotifications() {
    requestQueue = Volley.newRequestQueue(mActivity);
    JSONObject mainObj = new JSONObject();
    try {
        mainObj.put( name: "to", userFcmToken);
        JSONObject notiObject = new JSONObject();
        notiObject.put( name: "title", title);
        notiObject.put( name: "body", body);
        notiObject.put( name: "icon", R.drawable.new_package); // enter icon that exists in drawable only
        mainObj.put( name: "notification", notiObject);
        JsonObjectRequest request = new JsonObjectRequest(Request.Method.POST, postUrl, mainObj, new Response.Listener<JSONObject>() {
            @Override
            public void onResponse(JSONObject response) {
                // code run is got response
            }
        }, new Response.ErrorListener() {
            @Override
            public void onErrorResponse(VolleyError error) {
                // code run is got error
            }
        }) {
            @Override
            public Map<String, String> getHeaders() throws AuthFailureError {
                Map<String, String> header = new HashMap<>();
                header.put( k: "content-type",  v: "application/json");
                header.put( k: "authorization",  v: "key=" + fcmServerKey);
                return header;
            }
        };
        requestQueue.add(request);

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

}
```

Εικόνα 4.35: Δημιουργία και αποστολή pushNotification

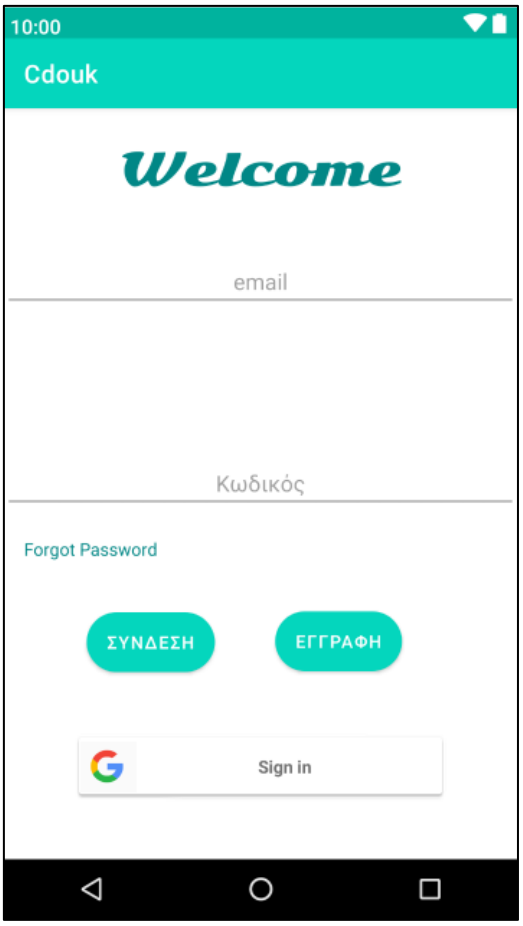
Κεφάλαιο 5ο: Παρουσίαση εφαρμογής

5.1 Εισαγωγή

Στο κεφάλαιο αυτό θα πραγματοποιηθεί η παρουσίαση της εφαρμογής. Η εφαρμογή αποτελείται από δύο ξεχωριστά μέρη. Το ένα μέρος, έχει να κάνει με λειτουργίες που αφορούν μόνο έναν διαχειριστή. Το άλλο μέρος της εφαρμογής αλληλεπιδρά με ένα ταυτοποιημένο χρήστη της εφαρμογής και περιέχει λειτουργίες που τον αφορούν.

5.2 Σύνδεση και Δημιουργία λογαριασμού

Κατά την σύνδεση, ενός χρήστη στην εφαρμογή γίνεται έλεγχος, αν έχει δικαιώματα διαχειριστή η όχι και φορτώνεται τη κατάλληλη διεπαφή χρήστη. Στην Εικόνα 5.1 βλέπουμε την απλή διαδικασία της σύνδεσης η της εγγραφής.



Εικόνα 5.1: Login και Register

Ο χρήστης καλείται να πληκτρολογήσει τα στοιχεία σύνδεσής του, δηλαδή το Email του και τον κωδικό πρόσβασης. Έπειτα γίνεται ταυτοποίηση του χρήστη με την Firestore και στο σενάριο της επιτυχούς σύνδεσης ο χρήστης μεταβαίνει αυτόματα στο αρχικό μενού, σε διαφορετική περίπτωση λαμβάνει ένα μήνυμα λάθους. Επιπλέον ο χρήστης έχει την δυνατότητα να δημιουργήσει νέο λογαριασμό , με το κουμπί «Εγγραφή». Σε περίπτωση λάθους η εφαρμογή δεν επιτρέπει στο χρήστη να προχωρήσει στη δημιουργία λογαριασμού αν δεν τηρούνται τα παρακάτω κριτήρια:

- Συμπληρωμένο όνομα χρήστη

- Συμπληρωμένο Email με επιβεβαιωμένη σωστή μορφή
- Ο κωδικό να έχει τουλάχιστον 6 χαρακτήρες

Στην Εικόνα 5.2 , βλέπουμε την μέθοδο που εκτελείται κατά την σύνδεση ενός χρήστη, ενώ στην Εικόνα 5.3, βλέπουμε την μέθοδο που εκτελείται κατά την δημιουργία ενός λογαριασμού.

```
private void loginUser(String email, String password) {
    auth.signInWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
        @Override
        public void onSuccess(AuthResult authResult) {
            Toast.makeText( context: MainActivity.this, text: "Σύνδεση : επιτυχής", Toast.LENGTH_SHORT).show();
            String id = auth.getUid();
            dr= db.collection( collectionPath: "users").document( documentPath: ""+id);
            dr.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    if(documentSnapshot.exists()){
                        if(documentSnapshot.getBoolean( field: "admin")){
                            startActivity(new Intent( packageContext: MainActivity.this, AdminActivity.class));
                            finish();
                        }else
                        if(documentSnapshot.getBoolean( field: "adeia")){
                            startActivity(new Intent( packageContext: MainActivity.this, DriverActivity.class));
                            finish();
                        }
                        else{
                            startActivity(new Intent( packageContext: MainActivity.this,BeforeStart.class));
                            finish();
                        }
                    }
                }
            })
            .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(MainActivity.this, "Αποτυχία Σύνδεσης", Toast.LENGTH_SHORT).show();
                            Toast.makeText( context: MainActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
                        }
            });
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(MainActivity.this, "Αποτυχία Σύνδεσης", Toast.LENGTH_SHORT).show();
            Toast.makeText( context: MainActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
        }
    });
}
```

Εικόνα 5.2: Μέθοδος που εκτελείται για την σύνδεση

```
public void RegisterUser(String email, String password){
    if(email!=null && password!=null) {
        auth.createUserWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
            @Override
            public void onSuccess(AuthResult authResult) {
                User user = new User();
                user.setAdmin(false);
                user.setAdeia(false);
                user.setOnoma("");
                user.setEmail(email);
                String id = auth.getUid();
                db.collection( collectionPath: "users").document( documentPath: "" + id).set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                    @Override
                    public void onSuccess(Void aVoid) {
                        Toast.makeText( context: MainActivity.this,  text: "Σύνδεση : επιτυχής", Toast.LENGTH_SHORT).show();
                        startActivity(new Intent( packageContext: MainActivity.this, BeforeStart.class));
                        finish();
                    }
                })
                        .addOnFailureListener(new OnFailureListener() {
                            @Override
                            public void onFailure(@NonNull Exception e) {
                                Toast.makeText( context: MainActivity.this,  text: "Σύνδεση : απέτυχε! Κάτι πήγε στραβά με το id του document", Toast.LENGTH_SHORT).show();
                                Toast.makeText( context: MainActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
                            }
                        });
            }
        })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText( context: MainActivity.this,  text: "Σύνδεση : απέτυχε! Κάτι πήγε στραβά με το email. Ίσως ο χρήστης υπάρχει ήδη", Toast.LENGTH_SHORT).show()
                        Toast.makeText( context: MainActivity.this, e.toString(), Toast.LENGTH_SHORT).show();
                    }
                });
    }
    else{
        Toast.makeText( context: MainActivity.this,  text: "Σύνδεση : απέτυχε! Δεν μπορεί να είναι κενά το email η ο κωδικός", Toast.LENGTH_SHORT).show();
    }

}
```

Εικόνα 5.3: Μέθοδος που εκτελείται για την δημιουργία ενός λογαριασμού

Αξιοσημείωτο είναι πως στην περίπτωση που ο χρήστης έχει συνδεθεί προηγουμένως στην εφαρμογή και δεν έχει κάνει σωστή αποσύνδεση. Με το άνοιγα ξανά της εφαρμογή, το σύστημα συνδέει αυτόματα τον χρήστη και παραλείπει την οθόνη της σύνδεσης. Στην Εικόνα 5.4 φαίνεται ο κώδικας που ελέγχει αν υπάρχει ήδη συνδεδεμένος χρήστης.

```
protected void onCreate(Bundle savedInstanceState) {
    auth = FirebaseAuth.getInstance();
    FirebaseUser fuser=auth.getCurrentUser();
    if (fuser!=null){
        String id = auth.getUid();
        dr= db.collection( collectionPath: "users").document( documentPath: ""+id);
        dr.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
            @Override
            public void onSuccess(DocumentSnapshot documentSnapshot) {
                if(documentSnapshot.exists()){
                    if(documentSnapshot.getBoolean( field: "admin")){
                        startActivity(new Intent( packageContext: MainActivity.this, AdminActivity.class));
                        finish();
                    }else
                    if(documentSnapshot.getBoolean( field: "adeia")){
                        startActivity(new Intent( packageContext: MainActivity.this, DriverActivity.class));
                        finish();
                    }
                    else{
                        startActivity(new Intent( packageContext: MainActivity.this,BeforeStart.class));
                        finish();
                    }
                }
            }
        });
    }
```

Εικόνα 5.4: Έλεγχος για συνδεδεμένο χρήστη

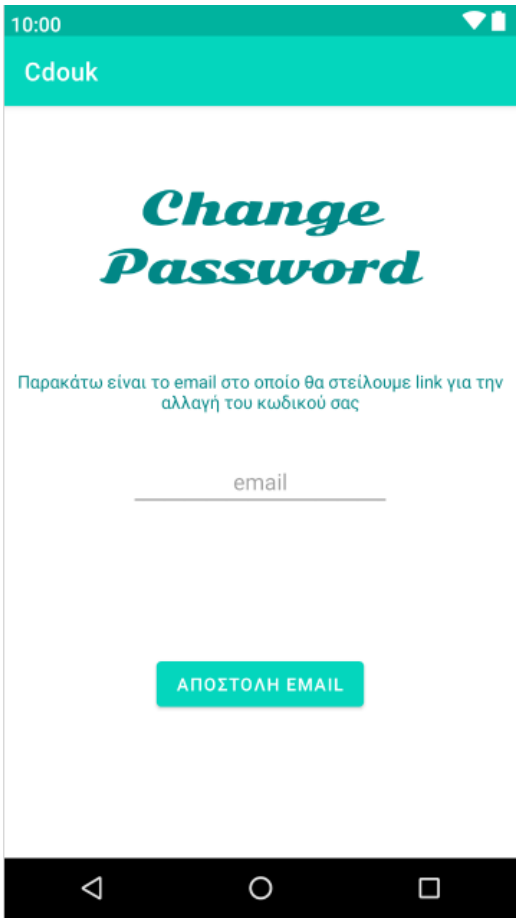
Τέλος, υπάρχει το κουμπί που επιτρέπει την είσοδο στην εφαρμογή με λογαριασμό Google. Ο κώδικας που εκτελείται είναι στην Εικόνα 5.5.

```
private void signIn(){
    Intent signInIntent = mGoogleSignInClient.getSignInIntent();
    startActivityForResult(signInIntent, RC_SIGN_IN);
}
@Override
public void onActivityResult(int requestCode, int resultCode, Intent data) {
    super.onActivityResult(requestCode, resultCode, data);
    if (requestCode == RC_SIGN_IN) {
        Task<GoogleSignInAccount> task = GoogleSignIn.getSignedInAccountFromIntent(data);
        try {
            GoogleSignInAccount account = task.getResult(ApiException.class);
            firebaseAuthWithGoogle(account);
        } catch (ApiException e) {
            Toast.makeText( context: this,  text: "Σύνδεση μέσω Google: απέτυχε!", Toast.LENGTH_SHORT).show();
        }
    }
}

private void firebaseAuthWithGoogle(GoogleSignInAccount account) {
    AuthCredential credential = GoogleAuthProvider.getCredential(account.getIdToken(),  accessToken: null);
    CollectionReference cr;
    cr = db.collection( collectionPath: "users");


    auth.signInWithCredential(credential).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
        @Override
        public void onSuccess(AuthResult authResult) {
            String id = auth.getUid();
            db.collection( collectionPath: "users").document( documentPath: "" + id).get().
                    addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
                @Override
                public void onSuccess(DocumentSnapshot documentSnapshot) {
                    if (documentSnapshot.exists()) {...}
                    else {...}
                }
            });


        }
    });
}
```

Εικόνα 5.5: Μέθοδος για Google Sign in

5.2.1 ForgotPassword

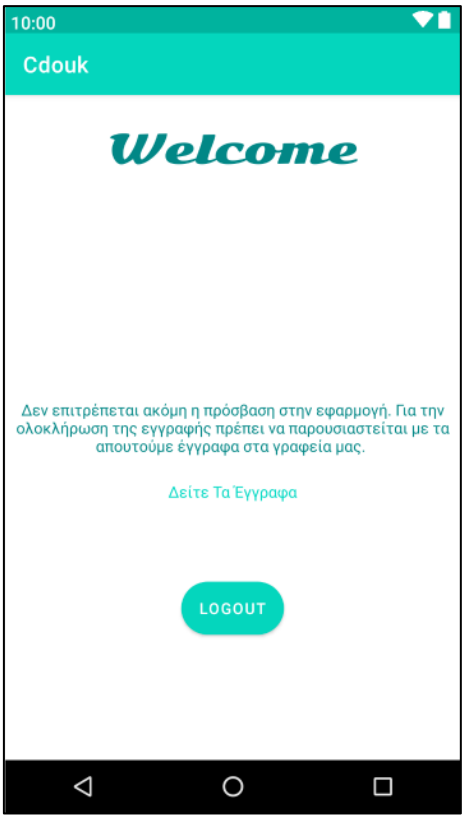
Στην περίπτωση που ένας ήδη εγγεγραμμένος χρήστης έχει ξεχάσει τον κωδικό σύνδεσής του υπάρχει η δυνατότητα ανάκτησης κωδικού. Η ενέργεια αυτή, παρέχεται με το πάτημα του κειμένου 'Forgot Password'. Ο χρήστης μεταφέρεται στην οθόνη που φαίνεται στην Εικόνα 5.6 για την ολοκλήρωση της διαδικασίας. Θα κληθεί να πληκτρολογήσει το Email που είχε δηλώσει κατά την εγγραφή του. Έπειτα θα του αποσταλεί ένα Email ανάκτησης κωδικού για να δημιουργήσει έναν καινούριο κωδικό.



Εικόνα 5.6: Αλλαγή κωδικού

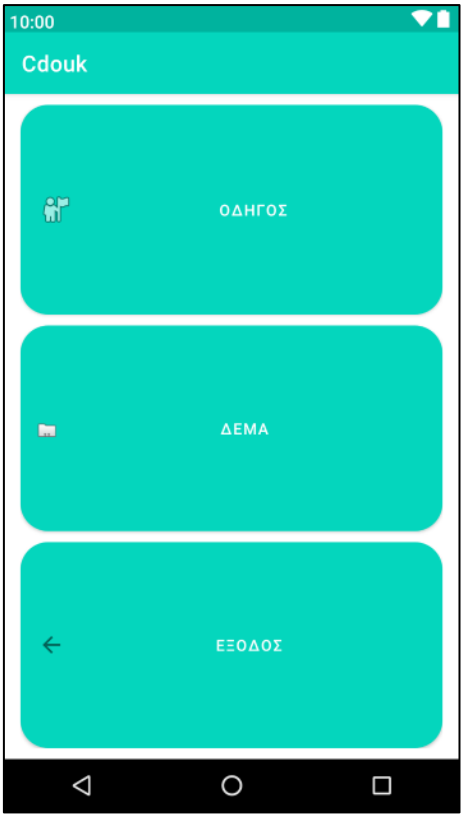
5.3 Μη ολοκληρωμένη εγγραφή

Για την πρόσβαση στην εφαρμογή και στις αντίστοιχες λειτουργίες της, θα πρέπει να έχει ολοκληρώσει την απαιτούμενη διαδικασίας για την σωστή εγγραφή. Ο λογαριασμός ελέγχεται και στην περίπτωση που δεν έχει άδεια πρόσβασης ο χρήστης μεταφέρεται στην οθόνη στην Εικόνα 5.7.



Εικόνα 5.7: Οθόνη πριν την ολοκλήρωση της εγγραφής

5.4 Διαχειριστής



Εικόνα 5.8: Μενού που εμφανίζεται μόνο στον διαχειριστή

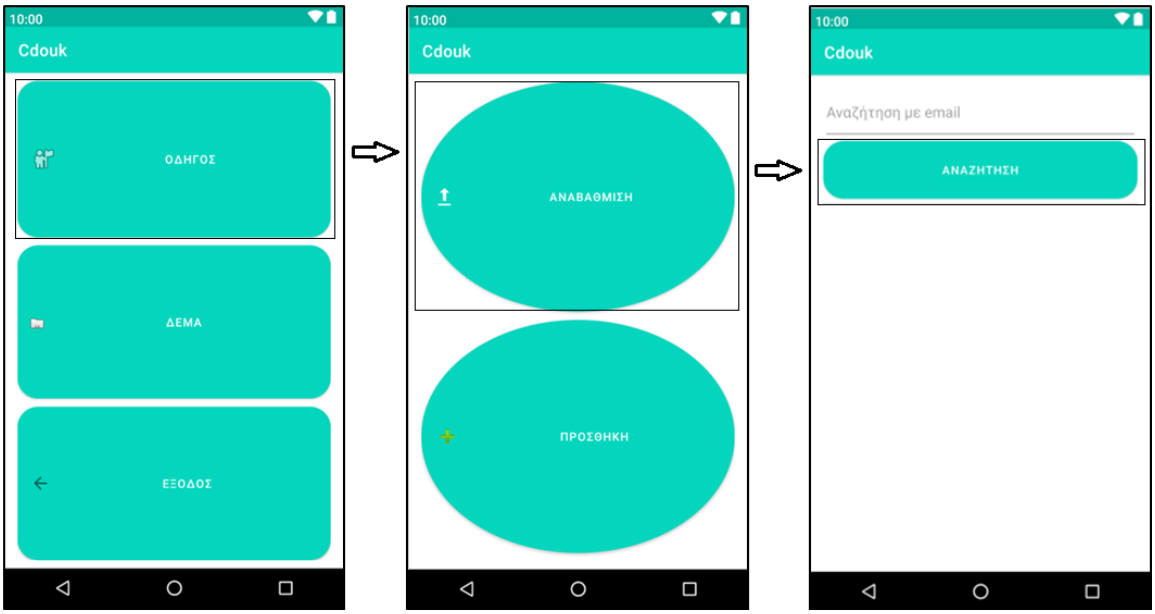
Ο διαχειριστής της εφαρμογής, έχει πρόσβαση στις πληροφορίες όλων των χρηστών και τον δεμάτων που ανήκουν στην εφαρμογή. Μέσω τους Menu που εμφανίζεται στην Εικόνα 5.8 διαχειρίζεται και επεξεργάζεται όλα τις αντίστοιχες πληροφορίες.

5.4.1 Οδηγός

Με την επιλογή του κουμπιού «Οδηγός» μπορεί να αναβαθμίσει ένα υπάρχον λογαριασμό η να προσθέσει έναν καινούργιο.

5.4.1.1 Αναβάθμιση

Στην Εικόνα 5.9 βλέπουμε τα διαδοχικά βήματα που κάνει ο διαχειριστής. Στο τελευταίο μέρος με το πάτημα του κουμπιού «Αναζήτηση» η εφαρμογή ψάχνει στην βάση δεδομένων για τον χρήστη με το email που έχει δοθεί στο πλαίσιο κειμένου και έπειτα, εμφανίζει όλα του τα στοιχεία και ένα κουμπί «Αποθήκευση». Ο διαχειριστής είναι σε θέση να ολοκληρώσει τις επιθυμητές αλλαγές. Στην Εικόνα 5.10 βλέπουμε τον κώδικα που εκτελείται.

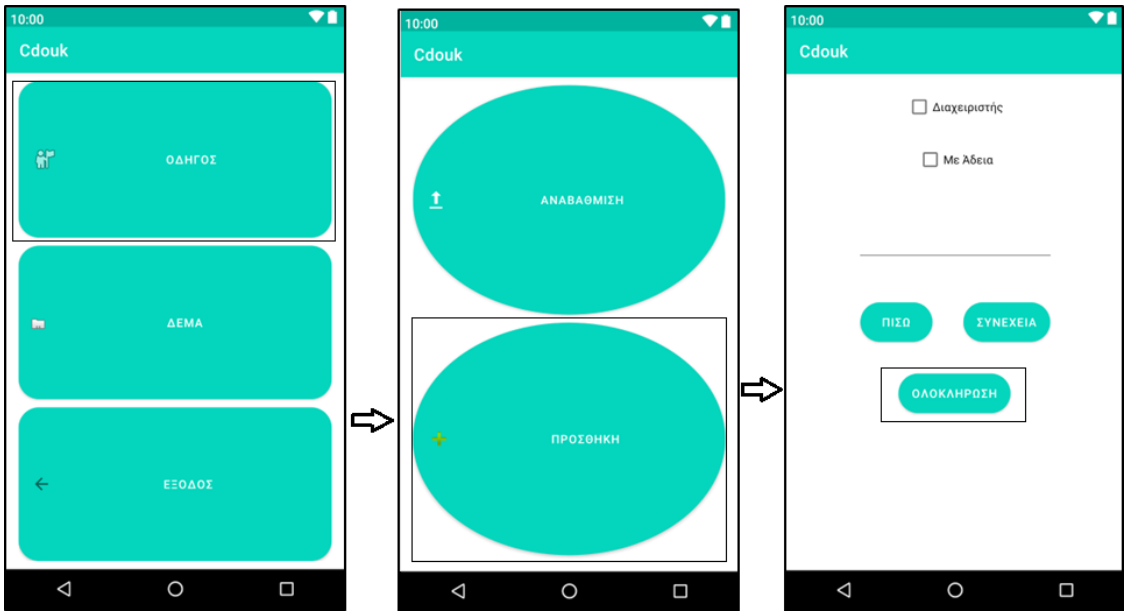


Εικόνα 5.9: Βήματα για την αναβάθμιση ενός χρήστη/οδηγού

```
public void updateUser(String email){
    CollectionReference cr= db.collection( collectionPath: "users");
    flagForSave=false;

    cr.get().addOnSuccessListener(new OnSuccessListener<QuerySnapshot>() {
        @Override
        public void onSuccess(QuerySnapshot queryDocumentSnapshots) {
            for(QueryDocumentSnapshot documentSnapshot: queryDocumentSnapshots){
                if (documentSnapshot.getString( field: "email").equals(email)){
                    try{
                        cr.document(documentSnapshot.getId()).update( field: "onoma", value: ""+onoma.getText());
                        cr.document(documentSnapshot.getId()).update( field: "epitheto", value: ""+epitheto.getText());
                        cr.document(documentSnapshot.getId()).update( field: "dieuthinsh", value: ""+dieuthinsh.getText());
                        cr.document(documentSnapshot.getId()).update( field: "tel", value: ""+tel.getText());
                        cr.document(documentSnapshot.getId()).update( field: "vathmos",Integer.valueOf(vathmos.getText().toString()));
                        cr.document(documentSnapshot.getId()).update( field: "admin",admin.isChecked());
                        cr.document(documentSnapshot.getId()).update( field: "adeia",adeia.isChecked());
                        Toast.makeText(getActivity(), text: "Επιτυχής Αποθήκευση", Toast.LENGTH_SHORT).show();
                        flagForSave=true;

                    }catch (Exception e){
                        Toast.makeText(getActivity(), text: ""+e, Toast.LENGTH_SHORT).show();
                    }
                }
            }
            if(!flagForSave){
                Toast.makeText(getActivity(), text: "Κάτι πήγε στραβά", Toast.LENGTH_SHORT).show();
            }
        }
    })
    .addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getActivity(), text: ""+e.toString(), Toast.LENGTH_SHORT).show();
        }
    });

}
```

Εικόνα 5.10: Μέθοδος αναβάθμισης χρήστη

5.4.1.2 Προσθήκη



Εικόνα 5.11: Βήματα για την προσθήκη ενός χρήστη

Στην Εικόνα 5.11 βλέπουμε τα διαδοχικά βήματα που κάνει ο διαχειριστής για να προσθέσει ένα χρήστη. Στο τελευταίο μέρος, μόλις συμπληρώσει τα στοιχεία, που εμφανίζονται διαδοχικά στο πλαίσιο κειμένου, πάνω από τα κουμπιά «Πίσω» και «Συνέχεια», με το πάτημα του κουμπιού «Ολοκλήρωση» η εφαρμογή εκτελεί τον κώδικα που παρατίθεται στην Εικόνα 5.12.

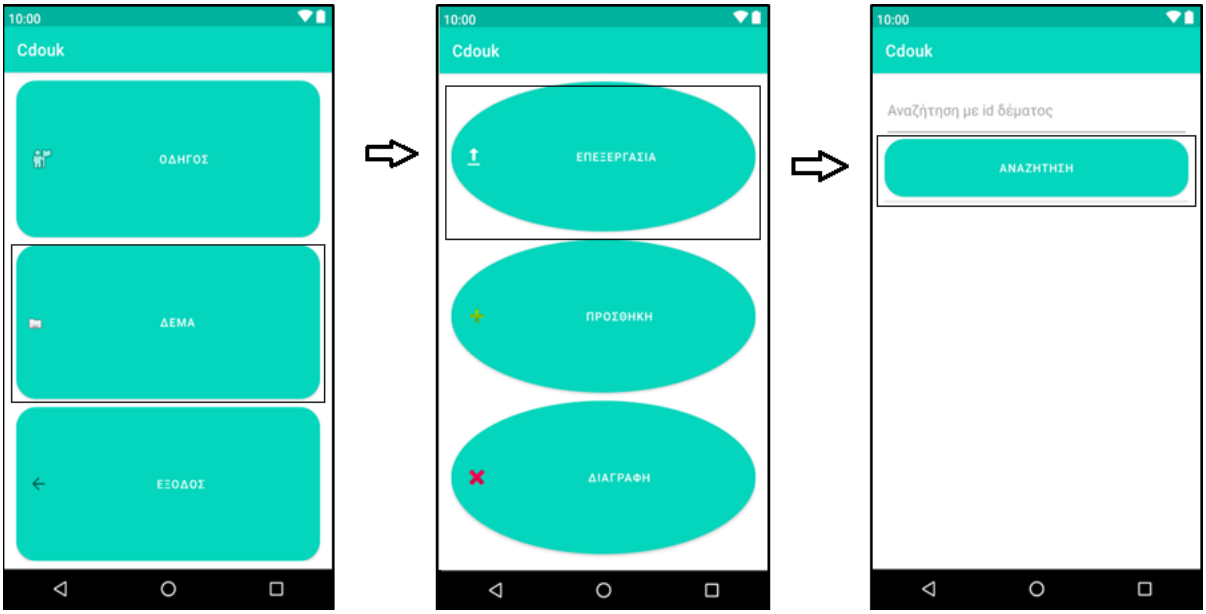
```
public void addUser(String email, String password, String onoma, String epitheto, String dieuthinsh, String tel, Boolean adeia, Boolean admin){

    auth.createUserWithEmailAndPassword(email, password).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
        @Override
        public void onSuccess(AuthResult authResult) {
            User user= new User();
            user.setAdmin(admin);
            user.setAdeia(adeia);
            user.setOnoma(onoma);
            user.setEpitheto(epitheto);
            user.setEmail(email);
            user.setDieuthinsh(dieuthinsh);
            user.setTel(tel);
            String id = authResult.getUser().getUid();
            db.collection( collectionPath: "users").document( documentPath: ""+id).set(user).addOnSuccessListener(new OnSuccessListener<Void>() {
                @Override
                public void onSuccess(Void aVoid) {
                    Toast.makeText(getActivity(), text: "Επιτυχής προσθήκη", Toast.LENGTH_SHORT).show();
                }
            })
                    .addOnFailureListener(new OnFailureListener() {
                        @Override
                        public void onFailure(@NonNull Exception e) {
                            Toast.makeText(getActivity(), text: "Η προσθήκη δεν έγινε", Toast.LENGTH_SHORT).show();
                            Toast.makeText(getActivity(), text: ""+e.toString(), Toast.LENGTH_SHORT).show();


                        }
                    });
        }
    })
            .addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    Toast.makeText(getActivity(), text: "Η εγγραφή και η προσθήκη δεν έγινε", Toast.LENGTH_SHORT).show();
                    Toast.makeText(getActivity(), text: ""+e.toString(), Toast.LENGTH_SHORT).show();
                }
            });

}
```

Εικόνα 5.12: Μέθοδος προσθήκης ενός χρήστη

5.4.2 Δέμα

Με την επιλογή του κουμπιού «Δέμα» δίνεται η δυνατότητα επεξεργασίας, προσθήκης ή διαγραφής ενός δέματος.

5.4.2.1 Επεξεργασία



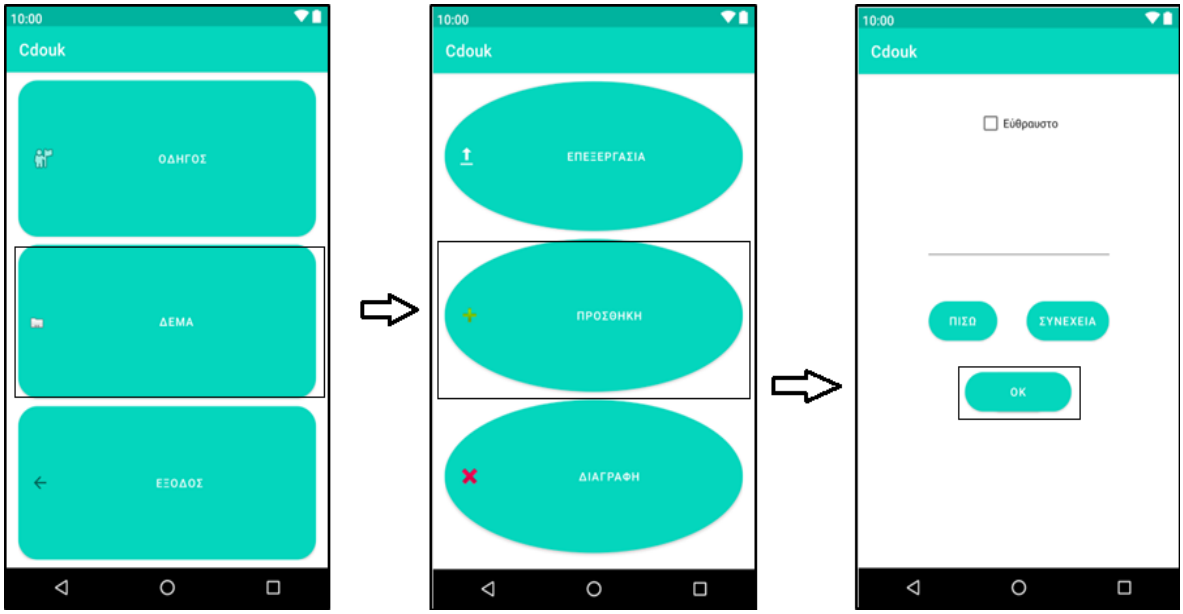
Εικόνα 5.13: Βήματα για την επεξεργασία ενός δέματος

Στην Εικόνα 5.13 βλέπουμε τα διαδοχικά βήματα που κάνει ο διαχειριστής. Στο τελευταίο μέρος με το πάτημα του κουμπιού «Αναζήτηση» η εφαρμογή ψάχνει στην βάση δεδομένων το δέμα με το μοναδικό χαρακτηριστικό του που έχει δοθεί στο πλαίσιο κειμένου και έπειτα, εμφανίζει όλα του τα στοιχεία και ένα κουμπί «Αποθήκευση». Ο διαχειριστής είναι σε θέση να επεξεργαστεί και να ολοκληρώσει τις επιθυμητές αλλαγές. Ο κώδικας που εκτελείται παρουσιάζεται στην Εικόνα 5.14.

```
public void updatePackage(String id){
    CollectionReference cr= db.collection( collectionPath: "packages");
    flagForSave=false;
    try{
        cr.document( documentPath: ""+id).update( field: "onoma_paralhpth", value: ""+onoma_paralhpth.getText());
        cr.document( documentPath: ""+id).update( field: "dieuthinsh_paralavhs", value: ""+dieuthinsh_paralavhs.getText());
        cr.document( documentPath: ""+id).update( field: "dieuthinsh_apostolhs", value: ""+dieuthinsh_apostolhs.getText());
        cr.document( documentPath: ""+id).update( field: "varos",Float.valueOf(varos.getText().toString()));
        cr.document( documentPath: ""+id).update( field: "kostos",Float.valueOf(kostos.getText().toString()));
        cr.document( documentPath: ""+id).update( field: "mhkos",Float.valueOf(mhkos.getText().toString()));
        cr.document( documentPath: ""+id).update( field: "platos",Float.valueOf(platos.getText().toString()));
        cr.document( documentPath: ""+id).update( field: "upsos",Float.valueOf(upsos.getText().toString()));
        cr.document( documentPath: ""+id).update( field: "euthrasto",euthrasto.isChecked());
        cr.document( documentPath: ""+id).update( field: "me_odhgo",me_odhgo.isChecked());
        Toast.makeText(getActivity(), text: "Επιτυχής Αποθήκευση", Toast.LENGTH_SHORT).show();
        flagForSave=true;

    }catch (Exception e){
        Toast.makeText(getActivity(), text: ""+e, Toast.LENGTH_SHORT).show();
    }
    if(!flagForSave){
        Toast.makeText(getActivity(), text: "Κάτι πήγε στραβά με την αποθήκευση", Toast.LENGTH_SHORT).show();
    }
}
```

Εικόνα 5.14: Μέθοδος για την ολοκλήρωση της επεξεργασίας ενός δέματος

5.4.2.2 Προσθήκη



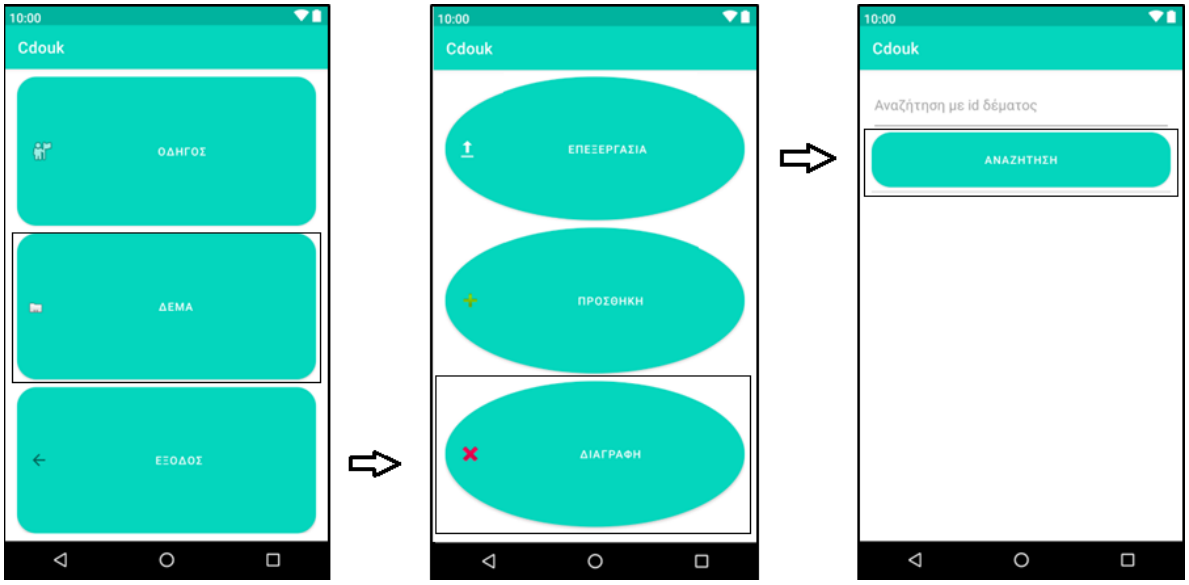
Εικόνα 5.15: Βήματα για την προσθήκη ενός δέματος

Στην Εικόνα 5.15 παρουσιάζονται τα διαδοχικά βήματα που κάνει ο διαχειριστής για να προσθέσει ένα δέμα. Στο τελευταίο μέρος, μόλις συμπληρώσει τα στοιχεία, που εμφανίζονται διαδοχικά στο πλαίσιο κειμένου, πάνω από τα κουμπιά «Πίσω» και «Συνέχεια», με το πάτημα του κουμπιού 'Ολοκλήρωση' η εφαρμογή εκτελεί τον κώδικα της Εικόνα 5.16. Η μέθοδος sendMessage() είναι η μέθοδος για την αποστολή push notification. Σχετικές λεπτομέρειες βρίσκονται στο αντίστοιχο κεφάλαιο (4.3.9)

```
public void addPackage(Packagee p){
    db.collection( collectionPath: "packages").add(p).addOnSuccessListener(new OnSuccessListener<DocumentReference>() {
        @Override
        public void onSuccess(DocumentReference documentReference) {
            finish.setVisibility(View.INVISIBLE);
            ok.setVisibility(View.VISIBLE);
            text.setVisibility(View.VISIBLE);
            editText.setVisibility(View.VISIBLE);
            Toast.makeText(getActivity(), text: "Επιτυχής προσθήκη δέματος", Toast.LENGTH_SHORT).show();
            text.setText("Id δέματος");
            editText.setText(documentReference.getId());
            sendMessage();
        }
    }).addOnFailureListener(new OnFailureListener() {
        @Override
        public void onFailure(@NonNull Exception e) {
            Toast.makeText(getActivity(), text: "Η προσθήκη δεν έγινε", Toast.LENGTH_SHORT).show();
            Toast.makeText(getActivity(), text: ""+e.toString(), Toast.LENGTH_SHORT).show();

        }
    });

}
```

Εικόνα 5.16: Μέθοδος για την προσθήκη δέματος

5.4.2.3 Διαγραφή



Εικόνα 5.17: Βήματα για την διαγραφή ενός δέματος

Στην Εικόνα 5.17 φαίνεται η διαδικασία για την διαγραφή ενός δέματος. Στο τελευταίο μέρος με το πάτημα του κουμπιού «Αναζήτηση» η εφαρμογή ψάχνει στην βάση δεδομένων το δέμα με το μοναδικό χαρακτηριστικό του που έχει δοθεί στο πλαίσιο κειμένου και έπειτα, εμφανίζει όλα του τα στοιχεία και ένα κουμπί «Διαγραφή». Ο διαχειριστής είναι σε θέση να διαγράψει οριστικά το δέμα.

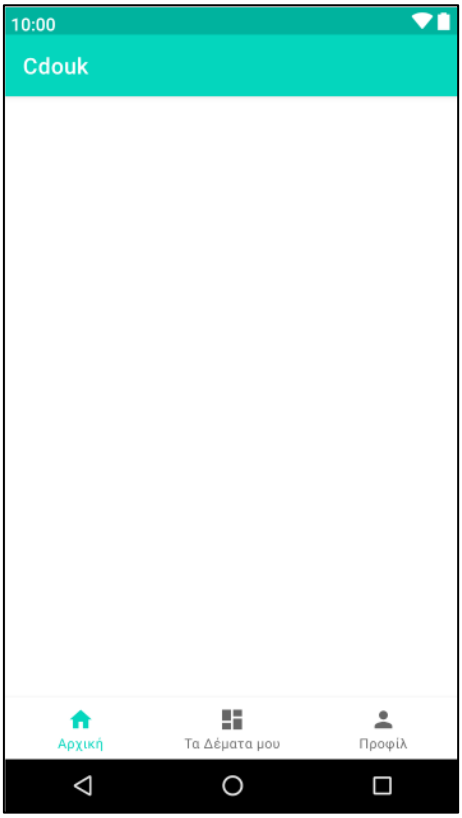
```
public void deletePackage(String id){
    CollectionReference cr= db.collection( collectionPath: "packages");
    cr.document( documentPath: ""+id).get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            Packagee p =documentSnapshot.toObject(Packagee.class);
            if( p!=null) {
                if(!p.isMe_odhgo()){
                    cr.document(id).delete().addOnSuccessListener(new OnSuccessListener<Void>() {
                        @Override
                        public void onSuccess(Void aVoid) {
                            Toast.makeText(getActivity(), text: "Επιτυχής Διαγραφή", Toast.LENGTH_SHORT).show();
                            scrollView.setVisibility(View.INVISIBLE);
                        }
                    })
                            .addOnFailureListener(new OnFailureListener() {
                                @Override
                                public void onFailure(@NonNull Exception e) {
                                    Toast.makeText(getActivity(), text: ""+e.toString(), Toast.LENGTH_SHORT).show();
                                }
                            });
                }
                else{
                    Toast.makeText(getActivity(), text: "Αποτυχία Διαφραφής", Toast.LENGTH_SHORT).show();
                    Toast.makeText(getActivity(), text: "Το δέμα έχει ανατεθεί σε οδηγό", Toast.LENGTH_SHORT).show();
                }
            }
        }
    });
```

Εικόνα 5.18: Μέθοδος για την διαγραφή ενός δέματος

Από τον κώδικα στην Εικόνα 5.18, συμπεραίνουμε πως η διαγραφή θα είναι επιτυχής μόνο στην περίπτωση που το δέμα δεν το έχει αναλάβει κάποιος οδηγός. Διαφορετικά εμφανίζει μήνυμα λάθους. Στην περίπτωση που ο διαχειριστής επιθυμεί ακόμη να διαγράψει το δέμα, θα πρέπει να μπει στο menu επεξεργασίας δέματος , να αλλάξει το χαρακτηριστικό που ορίζει αν στο έχει αναλάβει κάποιος οδηγός και έπειτα να ξαναεπιστρέψει στο menu της διαγραφής για να επιτύχει την αρχική του επιθυμία.

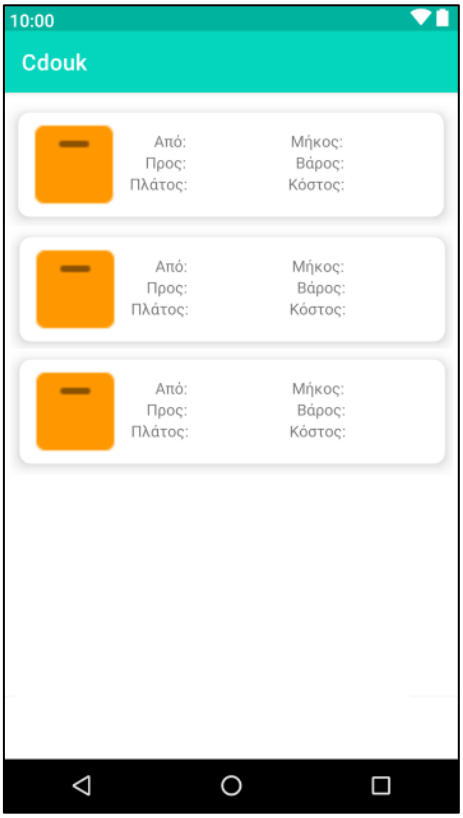
5.5 Χρήστης/Οδηγός

Ο χρήστης της εφαρμογής με ολοκληρωμένη πρόσβαση σε αυτήν, μπορεί να αξιοποιήσει πλήρως τις δυνατότητες και τις λειτουργίες της εφαρμογής, με την περιήγησή του στην εφαρμογή, μέσω του Menu που παρουσιάζεται στην Εικόνα 5.19



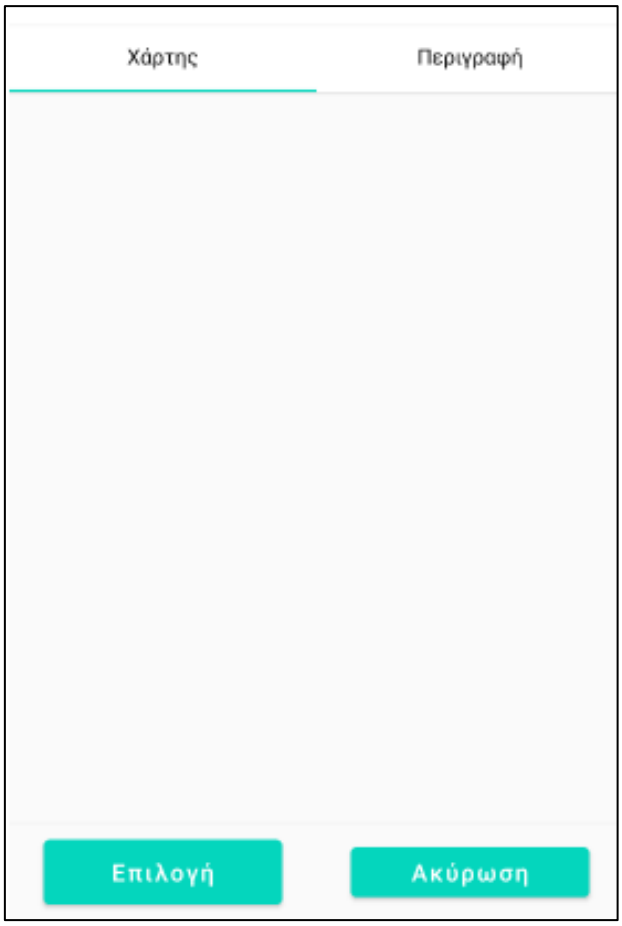
Εικόνα 5.19: Οθόνη του Χρήστη/Οδηγού

5.5.1 Αρχική



Εικόνα 5.20: Αρχική οθόνη

Στην Αρχική οθόνη που έχει μορφή σαν αυτήν στην Εικόνα 5.20, ο χρήστης πλέον μπορεί να δει όλα τα διαθέσιμα προς αυτόν δέματα. Πατώντας πάνω στο επιθυμητό δέμα μεταφέρεται σε οθόνη (Εικόνα 5.21) που εμφανίζονται αναλυτικότερα οι πληροφορίες. Επίσης, η παρουσία ενός χάρτη, με μαρκαρισμένες τις διευθύνσεις του δέματος, του επιτρέπει να πλοηγηθεί σε αυτόν. Στο σημείο εκείνο ο χρήστης αποφασίζει αν θα αναλάβει το δέμα η όχι, με το κλικ στο κουμπί «Επιλογή» η στο κουμπί «Ακύρωση». Στην Εικόνα 5.22, φαίνεται ο κώδικας που εκτελείται με το πάτημα του κουμπιού «Επιλογή». Στην περίπτωση που το έχει προλάβει άλλος, το σύστημα δεν επιτρέπει την επιλογή του δέματος και εμφανίζει μήνυμα λάθους. Στην Εικόνα 5.23, παρουσιάζεται ο κώδικας για το κουμπί «Ακύρωση».



Εικόνα 5.21: Οθόνη για τις πληροφορίες ενός δέματος

```
private void chooseButtonClicked(){ //prosthetei to epilegmemo paketo ston xrhsth
    FirebaseAuth auth = FirebaseAuth.getInstance();
    String id =auth.getUid();
    DocumentReference dr = db.collection( collectionPath: "packages").document( documentPath: ""+packageId);
    dr.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
        @Override
        public void onSuccess(DocumentSnapshot documentSnapshot) {
            if(!documentSnapshot.getBoolean( field: "me_odhgo")){
                db.collection( collectionPath: "users").document( documentPath: ""+id).update( field: "Allpackages", FieldValue.arrayUnion(dr));
                db.collection( collectionPath: "users").document( documentPath: ""+id).update( field: "Uncompletedpackages", FieldValue.arrayUnion(dr));
                db.collection( collectionPath: "packages").document( documentPath: ""+packageId).update( field: "me_odhgo", value: true);
                Toast.makeText( context: DetailsTabActivity.this,  text: "Προστέθηκε", Toast.LENGTH_SHORT).show();
            }
            else{
                Toast.makeText( context: DetailsTabActivity.this,  text: "Αποτυχία! Το έχει αναλάβει άλλος", Toast.LENGTH_SHORT).show();
            }
        }
    });

}
```

Εικόνα 5.22: Μέθοδος για την επιλογή ενός δέματος

```
@Override
public void onBackPressed() {
    this.finish();
    overridePendingTransition(R.anim.corner_down_right,R.anim.slide_out_right);
}


public void cancelButtonClicked() { onBackPressed(); }
```

Εικόνα 5.23: Μέθοδος για την ακύρωση και επιστροφή

5.5.2 Τα Δέματα μου

Εικόνα 5.24: Οθόνη 'Τα Δέματα μου'

Με την επιλογή «Τα Δέματα μου» από το μενού, εμφανίζεται η οθόνη στην Εικόνα 5.24. Μέσω των τριών κουμπιών, ο χρήστης μπορεί να δει όλα του τα δέματα, και ποια από αυτά έχουν ολοκληρωθεί η όχι. Στο σημείο αυτό ο χρήστης μπορεί να ολοκληρώσει μια παράδοση ή να αποδεσμευτεί από ένα δέμα που έχει αναλάβει προηγουμένως και να το κάνει πάλι διαθέσιμο σε όλους τους χρήστες.

```
private void completeButtonClicked(){
    FirebaseAuth auth = FirebaseAuth.getInstance();
    String id =auth.getUid();
    String packageId= this.packageId.getText().toString();
     try {
         DocumentReference dr = db.collection( collectionPath: "packages").document( documentPath: ""+packageId);
         db.collection( collectionPath: "users").document( documentPath: ""+id).update( field: "CompletedPackages", FieldValue.arrayUnion(dr));
         db.collection( collectionPath: "packages").document( documentPath: ""+packageId).update( field: "oloklhromeno", value: true);
         Toast.makeText(getActivity(), text: "Παραδόθηκε επιτυχώς", Toast.LENGTH_SHORT).show();
     }catch (Exception e){
         Toast.makeText(getActivity(), text: "Κάτι πήγε λάθος! "+e.toString(), Toast.LENGTH_SHORT).show();
     }
}
```

Εικόνα 5.25: Μέθοδος για την ολοκλήρωση παράδοσης

Στην παραπάνω Εικόνα 5.25, φαίνεται ο κώδικας για την ολοκλήρωση μιας παράδοσης. Αξίζει να σημειωθεί πως δεν επιτρέπει το σύστημα να ολοκληρωθεί για δεύτερη φορά η παράδοση σε ένα δέμα. Στην Εικόνα 5.26, ο κώδικας αποδεσμεύει έναν οδηγό από ένα δέμα και το κάνει ξανά διαθέσιμο. Άξιο σημείωσης είναι το πώς, δεν μπορεί να αποδεσμευτεί κάποιος οδηγός από δέμα που έχει παραδοθεί και έχει ολοκληρωθεί.

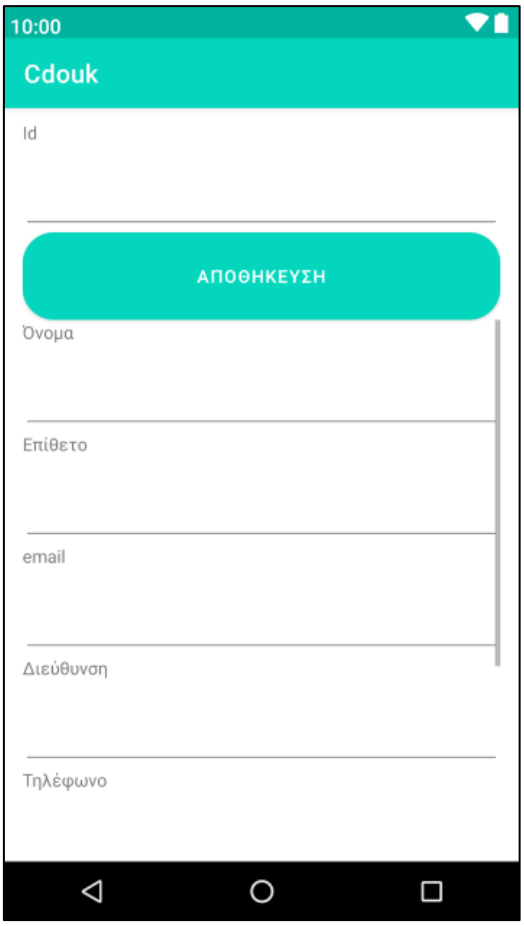
```
private void cancelButtonClicked(){ //diagrafei to  paketo ston xrhsth kai to kanei ksana diathesimo
    FirebaseAuth auth = FirebaseAuth.getInstance();
    String id =auth.getUid();
    String packageId= this.packageId.getText().toString();
    try {
        DocumentReference dr = db.collection( collectionPath: "packages").document( documentPath: "" + packageId);
        db.collection( collectionPath: "users").document( documentPath: "" + id).update( field: "Allpackages", FieldValue.arrayRemove(dr));
        db.collection( collectionPath: "users").document( documentPath: "" + id).update( field: "Uncompletedpackages", FieldValue.arrayRemove(dr));
        db.collection( collectionPath: "packages").document( documentPath: "" + packageId).update( field: "me_odhgo",  value: false);

    }catch (Exception e){
        Toast.makeText(getActivity(), text: "Κάτι πήγε λάθος! "+e.toString(), Toast.LENGTH_SHORT).show();
    }
    Toast.makeText(getActivity(), text: "Ακυρώθηκε επιτυχώς", Toast.LENGTH_SHORT).show();
}
```

Εικόνα 5.26: Μέθοδος για την αποδέσμευση μιας παράδοσης

5.5.3 Προφίλ

Εικόνα 5.27: Οθόνη ‘Προφίλ’

Μεταφερόμενος ο χρήστης στο Προφίλ, (Εικόνα 5.27) μπορεί να επεξεργαστεί κάποιες από τις πληροφορίες που τον αφορούν. Η οθόνη που εμφανίζεται στην Εικόνα 5.28 δίνει πρόσβαση για αλλαγή στα προσωπικά δεδομένα του χρήστη.



Εικόνα 5.28: Οθόνη επεξεργασίας Προφίλ

Με την ολοκλήρωση της επεξεργασίας εκτελείτε ο κώδικας στην Εικόνα 5.29. Επίσης, μπορεί να αλλάξει τον κωδικό του, με την γνωστή διαδικασία που έχει αναφερθεί στην αρχή του κεφαλαίου στο κεφάλαιο 5.2.1., το οποίο παρέχεται με το πάτημα του κειμένου ‘Αλλαγή κωδικού’. Μπορεί, επιπλέον, να ενεργοποιήσει και να απενεργοποιήσει τις ειδοποιήσεις τύπου push. Σχετικές λεπτομέρειες βρίσκονται στο αντίστοιχο κεφάλαιο (4.3.9). Τέλος, από το κουμπί «Έξοδος» μπορεί να εξέλθει με ασφάλεια από την εφαρμογή. Έτσι την επόμενη φορά που θα εισέλθει στο σύστημα της εφαρμογής θα πρέπει να εισάγει εκ νέου τα στοιχεία σύνδεσης του. Το σύστημα δεν θα τον εισάγει αυτόματα. Ο αντίστοιχος κώδικας βρίσκεται στην Εικόνα 5.30

```
public void updateUser(){

    FirebaseAuth auth;
    DocumentReference dr;
    auth = FirebaseAuth.getInstance();
    FirebaseUser fuser=auth.getCurrentUser();
    //Blepei ama uparxei user

    if (fuser!=null) {
        String id = fuser.getUid();
        dr = db.collection( collectionPath: "users").document( documentPath: "" + id);
        dr.get().addOnSuccessListener(new OnSuccessListener<DocumentSnapshot>() {
            @Override
            public void onSuccess(DocumentSnapshot documentSnapshot) {
                try {
                    dr.update( field: "onoma",  value: "" + onoma.getText());
                    dr.update( field: "epitheto",  value: "" + epitheto.getText());
                    dr.update( field: "dieuthinsh",  value: "" + dieuthinsh.getText());
                    dr.update( field: "tel",  value: "" + tel.getText());
                    dr.update( field: "admin", admin.isChecked());
                    dr.update( field: "adeia", adeia.isChecked());
                    Toast.makeText(getActivity(),  text: "Επιτυχής Αποθήκευση", Toast.LENGTH_SHORT).show();

                } catch (Exception e) {
                    Toast.makeText(getActivity(),  text: "" + e, Toast.LENGTH_SHORT).show();
                }
            }
        })
                .addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
                        Toast.makeText(getActivity(),  text: "" + e.toString(), Toast.LENGTH_SHORT).show();
                    }
                });
    }

}
```

Εικόνα 5.29: Μέθοδος για την επεξεργασία των στοιχείων του Προφίλ

```
FirebaseAuth.getInstance().signOut();
Toast.makeText(this.getActivity(),  text: "Έξοδος", Toast.LENGTH_SHORT).show();
startActivity(new Intent(this.getActivity(), MainActivity.class));
```

Εικόνα 5.30: Έξοδος από την εφαρμογή

Κεφάλαιο 6ο: Συμπεράσματα και προτάσεις βελτίωσης

6.1 Εισαγωγή

Στο κεφάλαιο αυτό παρατίθενται τα συμπεράσματα. Επίσης συζητιέται η πρωτοτυπία της εργασίας, οι περιορισμοί που υπήρξαν και διατυπώνονται προτάσεις βελτίωσης για μελλοντικές επεκτάσεις της εφαρμογής.

6.2 Συμπεράσματα

Ο σημαντικότερος στόχος, όπως έχει αναφερθεί, είναι η εφαρμογή να βοηθήσει και να συμβάλει στην γενική διαχείριση των δεμάτων. Για να επιτευχθεί αυτός ο στόχος θα πρέπει να δοθεί βαρύτητα στην ευχρηστία της εφαρμογής για να μπορεί να χρησιμοποιηθεί από το ευρύ κοινό. Σκοπός της εργασίας αυτής, ήταν η ανάπτυξη μιας Android εφαρμογής η οποία θα προσφέρει διάφορα δεδομένα και πληροφορίες, σχετικά με τα διαθέσιμα δέματα, για όλους όσους τους ενδιαφέρει η διαδικασία της μεταφοράς και παράδοσης δεμάτων. Έτσι, η ιδέα ενός πιο ολοκληρωμένου και έμπιστου περιβάλλοντος και η ανάγκη προστασίας ευαίσθητων πληροφοριών από ανεπιθύμητους χρήστες, οδήγησε στον σχεδιασμό και στην ανάπτυξη εφαρμογής, εύχρηστη κα φιλική στον χρήστη, με σύστημα ταυτοποίησης και σύνδεσης. Τέλος, η ύπαρξη του ρόλου διαχειριστή, κρίθηκε απαραίτητη, ώστε τα δεδομένα των εγγεγραμμένων χρηστών να είναι διαχειρήσιμα και οι πληροφορίες των αποθηκευμένων δεμάτων να είναι ελεγχόμενες.

6.3 Συζήτηση

Στα πλαίσια της διερεύνησης για αντίστοιχες έρευνες και εργασίες που ήδη έχουν διεξαχθεί, δεν υπήρξαν ευρήματα άξια αναφοράς τους. Αυτό, είχε ως αποτέλεσμα, την διαπίστωση του βαθμού πρωτοτυπίας και του περιορισμού στην διαθέσιμη βιβλιογραφία.

6.4 Προτάσεις βελτίωσης

Καθώς η ιδέα της εφαρμογής βρίσκεται ακόμη στην αρχή της εξέλιξής της, θα μπορούσαν να πραγματοποιηθούν διάφορες βελτιώσεις και επεκτάσεις είτε τεχνικού θέματος πάνω στην εφαρμογή, είτε περιεχομένου.

6.4.1 Επεκτάσεις τεχνικού θέματος

Σχετικά με τα διαθέσιμα δέματα, θα ήταν καλό να προστεθεί αναζήτηση μαζί με φίλτρα, για την καλύτερη αναζήτηση των χρηστών. Θα μπορούσαν να προσφέρονται φίλτρα σχετικά με τις τοποθεσίες του κάθε δέματος. Ο χρήστης θα μπορούσε να δίνει μια ακτίνα χιλιομέτρων με το οποίο θα περιόριζε την αναζήτηση του μόνο σε δέματα που έχουν σημεία παραλαβής και παράδοσης εντός της ακτίνας αυτής. Ακόμη, θα μπορούσε να προστεθεί, σαν φίλτρο στην αναζήτηση και η ταξινόμηση. Μια πιθανή ταξινόμηση θα μπορούσε να γίνει με βάση της απόστασης του χρήστη από την τοποθεσία του δέματος. Ακόμη θα μπορούσε να γίνει ταξινόμηση με βάση τα φυσικά χαρακτηριστικά του δέματος. Για παράδειγμα με βάση το βάρος, η το ύψος. Φυσικά, πρέπει να τονιστεί πως για τις παραπάνω προσθήκες θα πρέπει να υπάρξουν και αντίστοιχες αναβαθμίσεις στην βάση δεδομένων και στον κώδικα της εφαρμογής.

Αρκετά εύχρηστη θα γινόταν η εφαρμογή στην περίπτωση που ο κάθε χρήστη θα είχε την δυνατότητα να προσθέτει μόνος ου τα δικά του δέματα, χωρίς να πρέπει να επικοινωνήσεις με τον διαχειριστή.

Ένα κουμπί στην αρχική οθόνη , μαζί με τον κατάλληλο κώδικα, θα αναβάθμιζε σημαντικά την εφαρμογή.

Σχετικά με την επικοινωνία μεταξύ οδηγού, αποστολέα και παραλήπτη, θα ήταν χρήσιμη, η δυνατότητα για chat. Οι εμπλεκόμενοι θα μπορούσαν να συνεννοηθούν πιο εύκολα μεταξύ τους, για την καλύτερη εξυπηρέτηση του απώτερου σκοπού της εφαρμογής.

6.4.2 Βελτιώσεις περιεχομένου

Το ενδεχόμενο ύπαρξης διαθέσιμου ιστορικού αναζήτησης ή λίστα αγαπημένων (συχνότερων) τοποθεσιών θα έδινε την δυνατότητα για αποθήκευση τοποθεσιών και διαδρομών , ώστε να γίνεται ευκολότερος ο τρόπος αναζήτησης και εύρεσης δέματος, καθώς και την δυνατότητα εύκολης και γρήγορης εύρεσης τοποθεσιών και διαδρομών η άλλων χαρακτηριστικών που έχουν αναζητηθεί στο παρελθόν.

6.4.3 Επίλογος

Εν κατακλείδι, η συγκεκριμένη εφαρμογή θα μπορούσε να αποτελέσει την αρχή ενός νέου τρόπου διαχείρισης και μεταφοράς δεμάτων. Πέρα από τις επεκτάσεις και τις βελτιώσει που αναφέρθηκαν, με λίγη φαντασία, μπορεί να σκεφτεί κανείς άπειρες ακόμη. Το θέμα της εφαρμογής είναι διαχρονικό, χωρίς άμεση ημερομηνία λήξης. Πάντα θα υπάρχει η ανάγκη μεταφοράς και αποστολής δεμάτων. Είναι στο χέρι του καθενός, πλέον, η κάλυψη της.

ΒΙΒΛΙΟΓΡΑΦΙΑ

Κεφάλαιο 7ο: Έργα που αναφέρονται

[1] S. Mahamad, S. Sulaiman και W. Y. Leng, «An Integrated Courier Services Application: A New User Experience,» σε *IEEE Conference on e-Learning, e-Management and e-Services (IC3e)*, Langkawi, Malaysia, 2018.

[2] P. J. Deitel, C How to Program, 2nd Edition, Prentice Hall, 1994.

[3] T. Hagos, «Android Studio,» σε *Learn Android Studio 3*, Apress, Berkeley, CA, 2018, pp. 5-17.

[4] S. Bose, «A comparative study: java vs kotlin programming in android application development,» International Journal of Advanced Research in Computer Science, 2018, pp. 41-45.

[5] T. Hagos, «Gradle,» σε *Android Studio IDE Quick Reference*, Apress, Berkeley, CA, 2019, p. 83–93.

[6] A. Mishra, «The MVVM Architectural Pattern,» σε *iOS Code Testing*, Apress, Berkeley, CA, 2017, p. 43–60.

[7] L. Moroney, «The Firebase Realtime Database,» σε *The Definitive Guide to Firebase*, Apress, Berkeley, CA, 2017, p. 51–71.

[8] L. Moroney, «Using Authentication in Firebase,» σε *The Definitive Guide to Firebase*, Apress, Berkeley, CA, 2017, p. 25–50.

[9] F. Dahunsi, A. Joseph, O. Sarumi και O. Obe, σε *Database management system for mobile crowdsourcing applications*, Nigerian Journal of Technology, 2021, pp. 713-727.

[10] K. Mew, Mastering Android Studio 3, Packt Publishing Ltd, 2017.

[11] Y. Piao, J.-H. Jung και J. H. Yi, «Structural and Functional Analyses of ProGuard Obfuscation Tool,» The Journal of Korean Institute of Communications and Information Sciences, 2013, pp. 654-662.

[12] R. L. E. Dones και M. N. Young, «Demand on the of Courier Services during COVID-19 Pandemic in the Philippines,» Singapore, IEEE, 2020, pp. 131-134.

[13] W. B. A. Hatem Herchi, From user requirements to UML class diagram, arXiv preprint arXiv:1211.0713, 2012.

[14] Y. W. Syaifudin, N. Funabiki, I. Mu'aasyiqiin και D. C. Wijaya, «An Implementation of Multiple Activities Topic for Learning Intent and Fragment in Android Programming Learning Assistance System,» Okayama, Japan, International Conference on Information and Education Technology (ICIET), 2021, pp. 6-13.

[15] T. Hagos, «Navigation,» σε *In Learn Android Studio 4*, Apress, Berkeley, CA, 2020, pp. 111-130.

[16] N. S. Fatima, D. Steffy, D. Stella και S. N. Devi, «Enhanced Performance of Android Application Using RecyclerView,» σε *Advanced Computing and Intelligent Engineering*, Springer, Singapore, 2020, p. 189–199.

[17] R. Saborido, R. Morales, F. Khomh, Y.-G. Guéhéneuc και G. Antoniol, Getting the most from map data structures in Android, Empirical Software Engineering, 2018.

[18] W. Jackson, «Android's ViewPager Class: Using ViewPager to Navigate Horizontally,» σε *Pro Android UI*, Apress, Berkeley, CA, 2014, pp. 497-516.

[19] K. Fraser, B. Yousuf και O. Conlan, «Scrutable and Persuasive Push-Notifications,» σε *International Conference on Persuasive Technology*, Springer, Cham, 2019, pp. 67-73.