ΑΛΕΞΑΝΔΡΕΙΟ Τ.Ε.Ι. ΘΕΣΣΑΛΟΝΙΚΗΣ

ΣΧΟΛΗ ΤΕΧΝΟΛΟΓΙΚΩΝ ΕΦΑΡΜΟΓΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ Τ.Ε.

Τμήμα Μηχανικών Πληροφορικής ΑΤΕΙΘ

Πτυχιακή Εργασία

Δημιουργία εφαρμογής διαχείρισης εικονικών ουρών σε Android (IKA)

Του φοιτητή

Κυριάκου Καραγκιοζίδη

Αρ. Μητρώου: 123894

Επιβλέπων καθηγητής

Κεραμόπουλος Ευκλείδης

Θεσσαλονίκη 2017

Πρόλογος

Καθώς η καθημερινότητα έχει πολύ γρήγορους ρυθμούς και το κινητό τηλέφωνο έχει μετατραπεί σε κάτι πολύ περισσότερο από ένα εργαλείο επικοινωνίας και ψυχαγωγίας, πολλές φορές αποτελεί ένα μέσο για την επίλυση διαφόρων προβλημάτων. Η φορητότητα και το μικρό μέγεθος αποτελούν τα κύρια πλεονεκτήματά του, αφού επιτρέπει στον χρήστη να κάνει παρόμοιες λειτουργίες με αυτές των ηλεκτρονικών υπολογιστών. Ο συνδυασμός όλων αυτών των χαρακτηριστικών κάνει το κινητό τηλέφωνο (smartphone) το κατάλληλο εργαλείο για να λύσει τα καθημερινά προβλήματα που παρουσιάζονται. Αυτό έχει σαν αποτέλεσμα, να κεντρίσει το ενδιαφέρον πολλών προγραμματιστών ώστε να αναπτύξουν τις ιδέες/λύσεις τους σε εφαρμογές για κινητές συσκευές.

Λόγω όλων των παραπάνω, η εφαρμογή της παρούσας πτυχιακής εργασίας επιλέχθηκε να αναπτυχθεί για κινητά τηλέφωνα και πιο συγκεκριμένα για το λογισμικό Android.

Περίληψη

Η παρούσα πτυχιακή εργασία έχει ως κύριο στόχο την όσο το δυνατόν καλύτερη εξυπηρέτηση των πελατών που επισκέπτονται τα υποκαταστήματα Ι.Κ.Α. του Νομού Θεσσαλονίκης, περιμένοντας στην ουρά αναμονής για τα διάφορα τμήματα. Η εφαρμογή επιτρέπει στους χρήστες να εκδίδουν εισιτήρια για τα γραφεία/τμήματα που επιθυμούν, καθώς επίσης και στους υπαλλήλους των εκάστοτε γραφείων να ενημερώνουν τους χρήστες για την πορεία της αναμονής τους, κάθε φορά που εξυπηρετούν έναν πελάτη.

Τα βασικά στοιχεία αυτής της πτυχιακής εργασίας αφορούν κυρίως δύο μέρη. Το πρώτο, είναι ο εξυπηρετητής (server) και οι λειτουργίες που επιτελεί για την εφαρμογή, όπως η αποθήκευση της βάσης δεδομένων και των διάφορων υπολογισμών, ούτως ώστε να παρουσιαστούν στον χρήστη (client), δηλαδή στην Android εφαρμογή. Η Android εφαρμογή, που αποτελεί και το δεύτερο μέρος της πτυχιακής εργασίας, ασχολείται με την εμφάνιση των καταστημάτων/τμημάτων και την δυνατότητα έκδοσης εισιτηρίου σε ένα ή περισσότερα από αυτά, από την πλευρά του πελάτη, καθώς και με τη δυνατότητα των υπαλλήλων να ενημερώνουν τους χρήστες για την ολοκλήρωση της εξυπηρέτησης τους.

Συμπερασματικά, αυτή η εφαρμογή στοχεύει να βοηθήσει τον εκάστοτε χρήστη που επιθυμεί να επισκεφθεί οποιοδήποτε κατάστημα Ι.Κ.Α., χωρίς να χρειάζεται η φυσική του παρουσία στον χώρο αναμονής, παρά μόνο κατά το χρονικό διάστημα που πρόκειται να εξυπηρετηθεί. Επίσης, από τη σκοπιά των τεχνολογιών, στοχεύει σε νέες και δημοφιλείς τεχνολογίες, οι οποίες αποτελούν ένα ενδιαφέρον κομμάτι για μελέτη.

Abstract

The current thesis aims to provide the best possible service to the clients visiting the IKA branches of Thessaloniki, while they are queuing for the various departments. The application allows users to issue tickets in any office / department, as well as office employees to inform users of their waiting time, whenever they serve a customer.

This dissertation focuses in two main parts. The first one is the server and the functions it performs for the application, such as the storage of the database and the various calculations, so that they are presented to the user, i.e. the Android application. The Android application, which is the second part, deals with the appearance of branches / departments and the possibility of issuing a ticket to one or more of them, on the client's side, as well as the ability of employees to inform the users after the completion of their service.

In conclusion, this application aims to help users who wish to visit any of the IKA branches without being physically present in the queue unless it is time to be served. As far as technologies are concerned, the application aims at new and popular ones, which are an interesting part to study.

Ευρετήριο Περιεχομένων

ΕΥΡΕΤΗΡΙΟ ΣΧΗΜΑΤΩΝ ΚΑΙ ΠΙΝΑΚΩΝ

Εισαγωγή

Η παρούσα πτυχιακή εργασία έχει σαν σκοπό τον σχεδιασμό και τη δημιουργία μίας εφαρμογής με κάποιες από τις πιο σύγχρονες και δημοφιλείς τεχνολογίες που χρησιμοποιούνται σήμερα για την ανάπτυξη Android εφαρμογών.

Πρόκειται για μια client-server (πελάτης-εξυπηρετητής) εφαρμογή, άρα αποτελείται από δύο μέρη. Τα πρώτα κεφάλαια (1^{ο} και 2^{ο}) αναφέρονται στο κομμάτι του εξυπηρετητή και τα υπόλοιπα (3^{ο}, 4^{ο}, 5^{ο} και 6^{ο}) στο κομμάτι του πελάτη (Android εφαρμογή). Και στα δύο μέρη περιγράφονται με λεπτομέρεια οι τεχνολογίες που χρησιμοποιήθηκαν ούτως ώστε να ολοκληρωθεί η εφαρμογή.

Πιο συγκεκριμένα, στο 1^{ο} κεφάλαιο περιγράφονται οι τεχνολογίες και τα προγραμματιστικά περιβάλλοντα που χρησιμοποιήθηκαν από τη μεριά του εξυπηρετητή (server). Στη συνέχεια, στο 2^{ο} κεφάλαιο αναλύεται η βάση δεδομένων πάνω στην οποία βασίστηκε η εφαρμογή καθώς και όλες οι λειτουργίες που επιτελεί ο server, παραθέτοντας κομμάτια κώδικα από το backend.

Περνώντας στο δεύτερο μέρος της εργασίας που αφορά τον πελάτη (client), το 3^{ο} κεφάλαιο εισάγει τον αναγνώστη στις βασικές έννοιες του λειτουργικού συστήματος Android. Συνεχίζοντας, το 4^{ο} κεφάλαιο αναλύει τα βασικά στοιχεία από τα οποία αποτελείται μια εφαρμογή Android. Επιπρόσθετα, στο 5^{ο} κεφάλαιο παρουσιάζονται όλες οι τεχνολογίες και τα βασικά εργαλεία που χρησιμοποιήθηκαν για την υλοποίηση της παρούσας εφαρμογής. Τέλος, στο 6^{ο} κεφάλαιο περιγράφεται και αναλύεται εκτενώς η λειτουργία και η χρήση της εφαρμογής, παραθέτοντας, όπου χρειάζεται, επεξηγηματικά στιγμιότυπα οθόνης (screenshots).

Καλή ανάγνωση!

1. Τεχνολογίες που χρησιμοποιήθηκαν

1.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει μία αναλυτική περιγραφή των τεχνολογιών που χρησιμοποιήθηκαν όσων αφορά τη μεριά του εξυπηρετητή (Server Side – Back end), προκειμένου να γίνει κατανοητό στον αναγνώστη τόσο το θεωρητικό, όσο και το πρακτικό μέρος.

1.2 Τεχνολογίες

1.2.1 PHP

Η PHP είναι μία γλώσσα προγραμματισμού που χρησιμοποιείται από διακομιστές/εξυπηρετητές και έχει σχεδιαστεί κυρίως για web development, για τη δημιουργία δυναμικών και διαδραστικών ιστοσελίδων, αλλά χρησιμοποιείται επίσης και ως γενική γλώσσα προγραμματισμού. Τα αρχικά PHP στα πρώτα χρόνια «ζωής» της, σήμαιναν “Personal Home Page” (Προσωπική Αρχική Σελίδα), αλλά τώρα πια αντιπροσωπεύει το αναδρομικό ακρωνύμιο “PHP: Hypertext Preprocessor”.

Ο κώδικας PHP μπορεί να ενσωματωθεί σε σήμανση HTML (πλέον HTML 5) ή μπορεί να χρησιμοποιηθεί σε συνδυασμό με διάφορα συστήματα προτύπων και πλαισίων για web. Ο κώδικας PHP επεξεργάζεται συνήθως από έναν διερμηνέα PHP που υλοποιείται συνήθως ως πρόσθετο στον διακομιστή ιστού ή ως εκτελέσιμη διεπαφή Common Gateway Interface (CGI) [2]. Το λογισμικό του εξυπηρετητή συνδυάζει τα αποτελέσματα του διερμηνέα και του εκτελέσιμου κώδικα PHP, ο οποίος μπορεί να είναι οποιοσδήποτε τύπος δεδομένων, συμπεριλαμβανομένων εικόνων, με την παραγόμενη ιστοσελίδα. Ο κώδικας PHP μπορεί επίσης να εκτελεστεί μέσω διεπαφής γραμμής εντολών (Command-line Interface – CLI) και να χρησιμοποιηθεί για την υλοποίηση αυτόνομων γραφικών εφαρμογών. Επίσης, είναι σημαντικό να αναφερθεί, ότι εφόσον η PHP είναι ιδιαίτερα κατάλληλη για την ανάπτυξη web από την πλευράς διακομιστή, τρέχει και εκτελείται στον ίδιο τον διακομιστή.

Ο βασικός διερμηνέας PHP, που δημιουργήθηκε από την Zend Engine, είναι δωρεάν λογισμικό που εκδίδεται με τη σύμφωνη γνώμη της ομάδας ανάπτυξης της PHP. Πλέον, η PHP έχει διαδοθεί ευρέως σε όλο τον κόσμο και μπορεί να εγκατασταθεί στους περισσότερους διακομιστές ιστού σε κάθε σχεδόν λειτουργικό σύστημα και πλατφόρμα, εντελώς δωρεάν [1].

Το παρακάτω (κλασσικό) παράδειγμα “Hello World!” είναι γραμμένο σε κώδικα PHP και είναι ενσωματωμένο σε ένα έγγραφο HTML:

```
<!DOCTYPE html>
<html>
   <head>
      <title>PHP Test</title>
   </head>
   <body>
      <?php echo '<p>Hello World</p>'; ?>
   </body>
</html>
```

Ενώ δεν υπάρχει καμία προϋπόθεση για την ενσωμάτωση κώδικα PHP σε HTML, μια πιο απλή έκδοση του εν λόγω παραδείγματος μπορεί να γραφτεί όπως παρακάτω, με την ετικέτα κλεισίματος (closing tag) να παραλείπεται, όπως ακριβώς συμβαίνει σε αρχεία που περιέχουν καθαρό κώδικα PHP.

```
<?="Hello, world";
```

Τέλος, αξιοσημείωτο είναι το γεγονός ότι ενώ υπήρχε μία βασική λειτουργικότητα αντικειμενοστρεφούς προγραμματισμού στις εκδόσεις 3 και 4, η ομάδα ανάπτυξης της PHP έγραψε από την αρχή τον χειρισμό αντικειμένων για την έκδοση 5, επεκτείνοντας το σύνολο των χαρακτηριστικών και βελτιώνοντας αισθητά την απόδοση. Πλέον, με τη νέα προσέγγιση, τα αντικείμενα σχετίζονται από τον χειρισμό και όχι από την τιμή τους.

1.2.2 MySQL

Η MySQL είναι ένα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων ανοιχτού κώδικα (Relational DataBase Management System – RDBMS). Το όνομα της είναι ένας συνδυασμός του “My”, το όνομα της κόρης του συνιδρυτή Michael Widenius και της “SQL”, τη συντομογραφία δηλαδή του “Structured Query Language” (Δομημένη Γλώσσα Ερωτημάτων). Η MySQL ανήκε και χρηματοδοτήθηκε από μία ενιαία κερδοσκοπική εταιρεία, τη σουηδική MySQL AB, που τώρα ανήκει στην Oracle Corporation. Για ιδιόκτητη χρήση, διατίθενται πολλές αμειβόμενες εκδόσεις της και προσφέρουν επιπλέον διαθέσιμες λειτουργίες.

Η MySQL είναι το κεντρικό στοιχείο της στοίβας λογισμικού ανοιχτού κώδικα LAMP. Το αρκτικόλεξο LAMP σημαίνει “Linux, Apache, MySQL, Perl / PHP / Python”. Μερικές από τις εφαρμογές που χρησιμοποιούν τη βάση δεδομένων MySQL είναι: TYPO3, MODx, Joomla, WordPress, phpDB, MyDB και Drupal. Η MySQL χρησιμοποιείται επίσης και σε πολύ γνωστές ιστοσελίδες, όπως της Google, του Facebook, του Twitter, του Flickr και του YouTube.

1.2.2.1 MariaDB

Στον εξυπηρετητή που χρησιμοποίησα για τη δημιουργία του back-end της εφαρμογής, σαν προεπιλεγμένο τύπο για τη διαχείριση της βάσης δεδομένων που θα χρησιμοποιούσα ήταν η MariaDB. Η MariaDB είναι ουσιαστικά μια διακλάδωση (fork) του project της MySQL, που σημαίνει ότι οι προγραμματιστές παίρνουν ένα αντίγραφο του πηγαίου κώδικα και ξεκινούν την ανεξάρτητη ανάπτυξη του, δημιουργώντας ένα ξεχωριστό κομμάτι λογισμικού – μία νέα έκδοση (third-party). Μία από αυτές τις διακλαδώσεις, λοιπόν, είναι και η MariaDB, που η ανάπτυξή της, αξίζει να σημειωθεί, ότι ηγείται από τους αρχικούς προγραμματιστές της MySQL, οι οποίοι, έχοντας ανησυχίες για το project τους, που αποκτήθηκε από την Oracle, αποφάσισαν να τη δημιουργήσουν, έχοντας ωστόσο διατηρήσει υψηλή συμβατότητα με την MySQL.

Η MariaDB μετατρέπει τα δεδομένα σε δομημένες πληροφορίες σε ένα ευρύ φάσμα εφαρμογών, που κυμαίνονται από τραπεζικές συναλλαγές έως απλές ιστοσελίδες. Είναι, όπως ισχυρίζεται, μία βελτιωμένη αντικατάσταση της MySQL. Χρησιμοποιείται επειδή είναι γρήγορη, κλιμακωτή και ισχυρή, με ένα πλούσιο οικοσύστημα μηχανών αποθήκευσης, plug-ins και πολλών άλλων εργαλείων, καθιστώντας το πολύ πιο ευέλικτο και με μεγάλη ποικιλία για όλες τις περιπτώσεις χρήσης της.

1.2.3 JSON

Το JSON (JavaScript Object Notation) είναι βασισμένο σε ένα υποσύνολο της γλώσσας προγραμματισμού JavaScript και είναι ένα πρότυπο κειμένου που χρησιμοποιείται για αποθήκευση και ανταλλαγή δεδομένων. Η πιο συνήθης χρήση του σήμερα, είναι η μετάδοση δεδομένων ανάμεσα σε εφαρμογές. Είναι ελαφρύ, απλό, εύκολα κατανοητό από τους ανθρώπους και δεν εξαρτάται από καμιά γλώσσα προγραμματισμού, χαρακτηριστικά που το έχουν κάνει ευρέως διαδεδομένο.

1.2.3.1 Διάφορες μορφές JSON

Ένα JSON ουσιαστικά πρόκειται για κείμενο γραμμένο με σημειογραφία ενός αντικειμένου JavaScript, που είναι της μορφής «κλειδί/τιμή» (key/value).

Κλειδί (Key): Ένα κλειδί είναι πάντοτε μια συμβολοσειρά (string) που περικλείεται σε εισαγωγικά.

Τιμή (Value): Μία τιμή μπορεί να είναι μια συμβολοσειρά, ένας αριθμός, μία έκφραση Boolean, ένας πίνακας ή ακόμη και ένα αντικείμενο.

Ζεύγος κλειδί/τιμή (Key/Value Pair): Ένα τέτοιο ζεύγος ακολουθεί μια συγκεκριμένη σύνταξη, με το κλειδί να ακολουθείται από άνω κάτω τελεία, ακολουθούμενο από την τιμή. Στην περίπτωση που υπάρχουν πολλαπλά ζεύγη, διαχωρίζονται μεταξύ τους με κόμμα. Η σύνταξη του είναι όπως η παρακάτω:

```
{
    "κλειδί" : "τιμή"
}
```

Σαν κλειδί μπορεί να είναι οποιοδήποτε αλφαριθμητικό και σαν τιμή υποστηρίζει τους παρακάτω τύπους:

α) Αλφαριθμητικό (String)

```
{
    “store_name” : “ΙΚΑ Καλαμαριάς”
}
```

β) Αριθμός

```
{
    “active_tickets” : 11
}
```

γ) Λογικές τιμές (true – false)

```
{
    “queue_is_empty” : true
}
```

δ) Πίνακας (Array)

```
{
    “coordinates” : [ 40.45124, 22.25987 ]
}
```

ε) Αντικείμενο (Object)

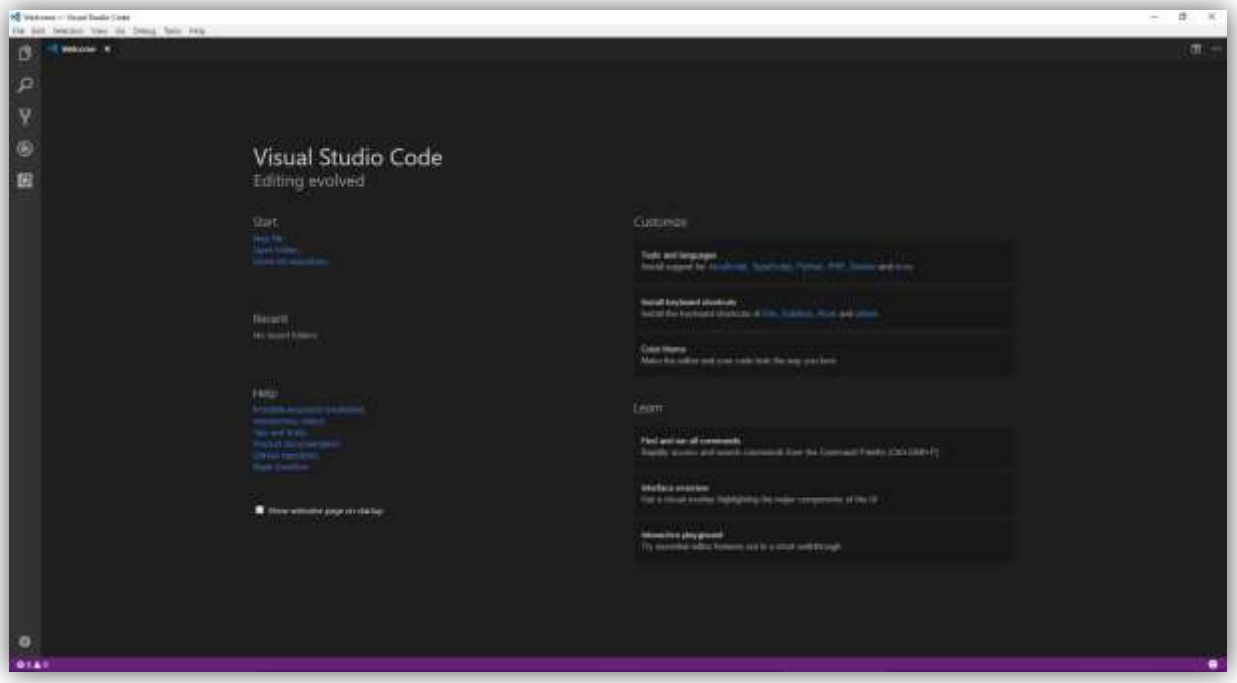
```
{
    “store” : {
                “store_name” : “ΙΚΑ Θεσσαλονίκης”,
                “address” : “Αριστοτέλους 15 - 17”
            }
}
```

1.3 Προγραμματιστικά Περιβάλλοντα

1.3.1 Visual Studio Code

Το Visual Studio Code είναι ένας από τους δεκάδες επεξεργαστές κώδικα που αναπτύχθηκε από τη Microsoft για Windows, Linux και MacOS. Περιλαμβάνει υποστήριξη για εντοπισμό σφαλμάτων, ενσωματωμένο έλεγχο για Git, διάφορες επισημάνσεις και χρώματα ανάλογα με τη γλώσσα προγραμματισμού, βοηθό για τυχόν λάθη και επεξηγήσεις του κώδικα και διάφορα πρόσθετα (plug-ins). Είναι επίσης πολύ προσαρμόσιμο στις ανάγκες του εκάστοτε χρήστη, έχοντας τη δυνατότητα να αλλάξουν τα χρώματα και το θέμα του προγράμματος, τις συντομεύσεις πληκτρολογίου, αλλά και πολλές άλλες ρυθμίσεις και προτιμήσεις. Πρόκειται για δωρεάν και ανοικτού κώδικα πρόγραμμα, παρόλο που ακριβώς πριν το κατέβασμα του εκτελέσιμου αρχείου, η εταιρία ενημερώνει ότι χρειάζεται άδεια ιδιοκτησίας.

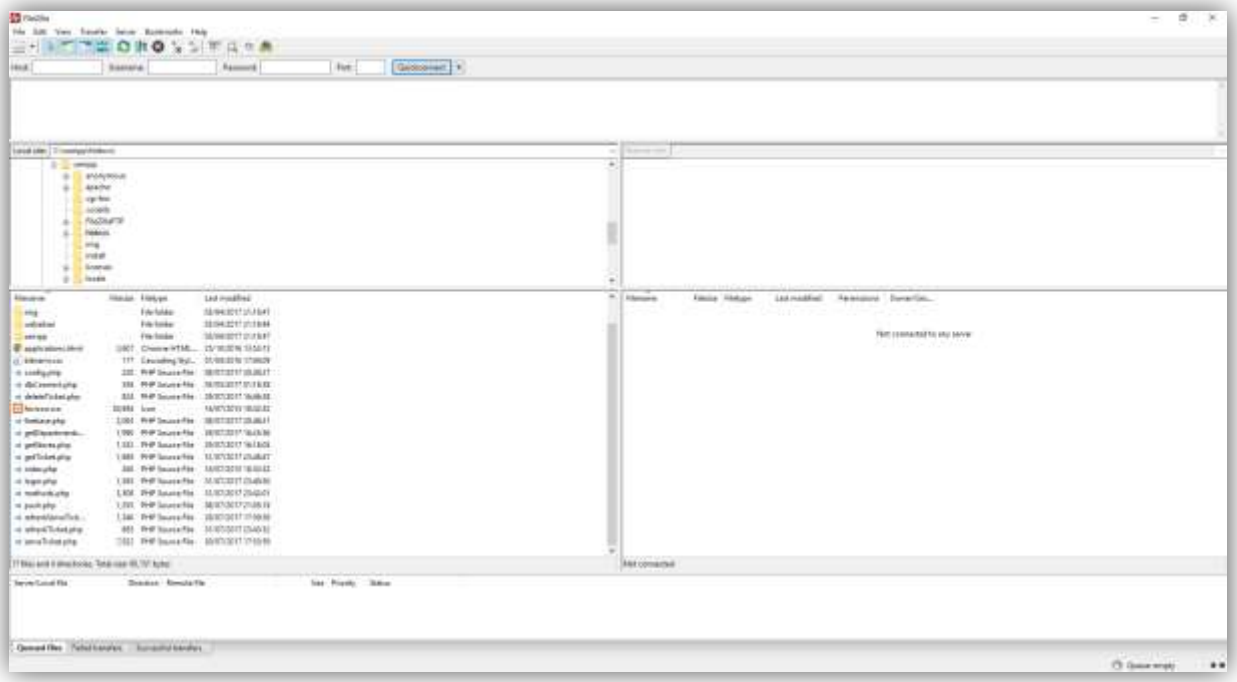
Ακολουθεί μία εικόνα που παρουσιάζει την κύρια οθόνη του. Για δοκιμή και περισσότερες πληροφορίες [3].



Εικόνα 1: Visual Studio Code περιβάλλον

1.3.2 FileZilla

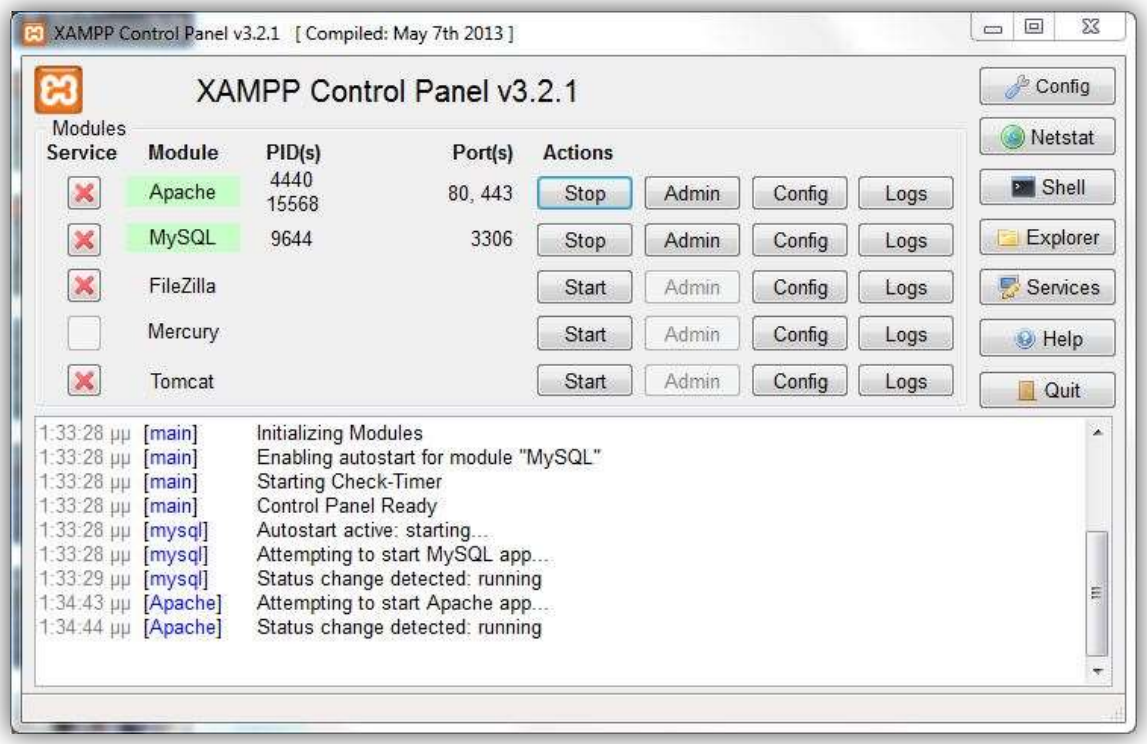
Πρόκειται για έναν FTP πελάτη τον οποίο μπορούμε να χρησιμοποιήσουμε για να ανεβάσουμε αρχεία σε κάποιον εξυπηρετητή. Υπάρχουν διαθέσιμες εκδόσεις για Linux, Windows και MacOS. Ο πελάτης υποστηρίζει FTP, SFTP και FTPS (FTP μέσω SSL / TLS). Είναι ένα ελεύθερο λογισμικό [4] και το περιεχόμενο της αρχικής οθόνης του φαίνεται στη παρακάτω εικόνα.



Εικόνα 2: FileZilla περιβάλλον

1.3.3 XAMPP

Συνήθως η PHP και η MySQL εγκαθίστανται σε συνδυασμό με κάποιον http-server. Η πιο απλή και ολοκληρωμένη λύση η εγκατάσταση του συγκεκριμένου λογισμικού, το οποίο προσφέρει σε πακέτο τον Apache http-server, τη γλώσσα PHP, και τη βάση MySQL. Η διάθεση και η χρήση του είναι δωρεάν [5] και η επιλογή της έκδοσης εξαρτάται από την πλατφόρμα του λειτουργικού συστήματος που έχει ο server. Στην παρακάτω εικόνα, παρατίθεται ο «πίνακας ελέγχου» (control panel) του λογισμικού.



Εικόνα 3: Πίνακας Ελέγχου XAMPP

1.3.4 Επίλογος

Στο παρόν κεφάλαιο έγινε μια προσπάθεια να περιγραφούν αναλυτικά όλες οι τεχνολογίες που χρησιμοποιήθηκαν, καθώς και τα προγραμματιστικά περιβάλλοντα ανάπτυξης του back-end της εφαρμογής.

Είμαστε έτοιμοι λοιπόν να αναλύσουμε εκτενώς όλες τις λειτουργίες που επιτελεί ο εξυπηρετητής προς την Android εφαρμογή.

2. Περιγραφή του back-end συστήματος

2.1 Περιγραφή της βάσης δεδομένων

Η βάση δεδομένων που δημιουργήθηκε για την παρούσα εφαρμογή αποτελείται από 12 πίνακες: 1 πίνακας “stores” με τα στοιχεία των καταστημάτων του ΙΚΑ στην περιοχή του Νομού Θεσσαλονίκης, 1 πίνακας “departments” με τα ονόματα όλων των τμημάτων που υπάρχουν στα παραπάνω υποκαταστήματα ΙΚΑ, 1 πίνακας “users” που περιέχει τα στοιχεία των υπαλλήλων του ΙΚΑ, 1 πίνακας “store_departments” με τις συσχετίσεις μεταξύ των καταστημάτων και των τμημάτων (ουσιαστικά συσχετίζει τα τμήματα που υπάρχουν σε κάθε υποκατάστημα ΙΚΑ) και τέλος 8 πίνακες “queue_table” που περιλαμβάνουν όλα τα στοιχεία των εισιτηρίων που κόβουν οι χρήστες της εφαρμογής σε κάθε κατάστημα. (Υπάρχουν 8 καταστήματα, οπότε θα πρέπει να υπάρξουν και 8 queuing πίνακες).

2.1.1 Πίνακας “stores”

Στον πίνακα “*stores*” αποθηκεύουμε όλες τις πληροφορίες που είναι απαραίτητες για την επικοινωνία με οποιονδήποτε τρόπο με το κατάστημα, αλλά και εκείνες που μας επιτρέπουν να βρούμε την τοποθεσία της στον χάρτη. Τα πεδία του πίνακα λοιπόν είναι τα εξής: *name* (όνομα), *address* (διεύθυνση), *postal_code* (ταχυδρομικός κώδικας – ΤΚ), *email*, *phone* (τηλέφωνο), *latitude* (γεωγραφικό πλάτος), *longitude* (γεωγραφικό μήκος).

2.1.2 Πίνακας “departments”

Ο πίνακας “*departments*” έχει περισσότερο βοηθητικό χαρακτήρα που θα αναλυθεί παρακάτω, οπότε το μόνο πεδίο που περιέχει είναι το όνομα (name) του τμήματος.

2.1.3 Πίνακας “store_departments”

Ο πίνακας “*store_departments*” ουσιαστικά αποτελεί τον τρόπο συσχέτισης των πινάκων “*stores*” και “*departments*”, όπως αφήνει να εννοηθεί από το όνομά του. Κάνοντας JOIN τους 2 προηγούμενους πίνακες, μέσω του “*store_departments*” και των συσχετίσεων που υπάρχουν αποθηκευμένες εκεί, έχει σαν αποτέλεσμα να μπορούμε να συνδέσουμε το κάθε κατάστημα με τα τμήματα που περιέχονται σε αυτό.

2.1.4 Πίνακας “users”

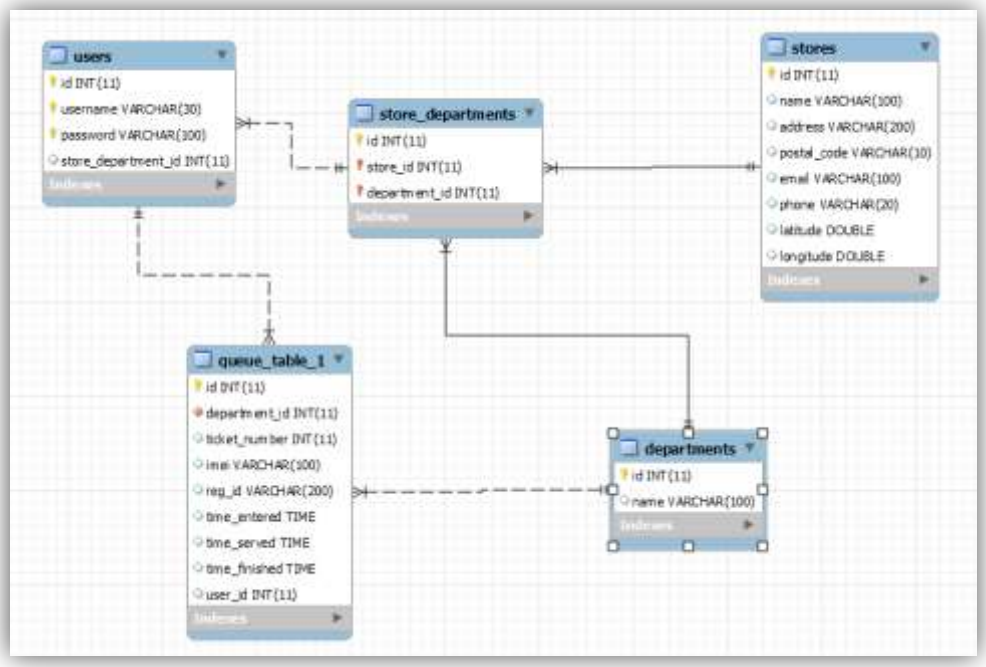
Ο πίνακας “*users*” είναι αρκετά απλοϊκός στην υλοποίηση του, έχοντας σαν πεδία ένα *username* (όνομα χρήστη) και ένα *password* (κωδικός πρόσβασης), καθώς και ένα “*store_department_id*” το οποίο είναι απαραίτητο για την συσχέτιση του εκάστοτε υπαλλήλου με την ουρά αναμονής και ειδικότερα με τον χρήστη που θα εξυπηρετήσει. (Περισσότερες λεπτομέρειες παρακάτω)

2.1.5 Πίνακες “queue_table_”

Σε αυτούς τους 8 πίνακες αποθηκεύουμε όλες τις πληροφορίες που είναι απαραίτητες τη στιγμή που ένας χρήστης αιτείται εισιτήριο σε μία οποιαδήποτε ουρά, οποιουδήποτε τμήματος σε καθένα από τα 8 καταστήματα που υπάρχουν στην εφαρμογή. Αυτό σημαίνει ότι όλες οι πληροφορίες των εισιτηρίων όλων των ουρών αναμονής κάθε υποκαταστήματος του ΙΚΑ, αποθηκεύονται σε έναν μόνο πίνακα, αυτόν που αντιστοιχεί στο κάθε κατάστημα. Οι ουρές διαχωρίζονται μεταξύ τους ανάλογα με το τμήμα (*department_id*) που είναι και ένα από τα πεδία του πίνακα. Τα υπόλοιπα είναι: *ticket_number* (αριθμός εισιτηρίου), *imei* (International Mobile Equipment Identity / Διεθνής Ταυτότητα Κινητού Εξοπλισμού [6] – χρησιμοποιείται ούτως ώστε ο κάθε χρήστης να είναι μοναδικός και να μην μπορεί να εκδίδει παραπάνω από ένα εισιτήριο ανά ουρά αναμονής), *reg_id* (registration id / id εγγραφής – χρησιμοποιείται για να υπάρξει η δυνατότητα αποστολής ειδοποιήσεων /

notifications), *time_entered* (ώρα έκδοσης εισιτηρίου), *time_served* (ώρα εισόδου στο γραφείο ώστε να εξυπηρετηθεί), *time_finished* (ώρα που τελείωσε η εξυπηρέτηση του χρήστη από τον υπάλληλο).

Το διάγραμμα ER της βάσης δεδομένων που χρησιμοποιήθηκε στην παρούσα πτυχιακή εργασία είναι το εξής (για εξοικονόμηση χώρου, για το διάγραμμα χρησιμοποιήθηκε μόνο ο πίνακας "*queue_table_1*", καθώς όλοι οι υπόλοιποι είναι παρόμοιοι):



2.2 Αναλυτική περιγραφή του συστήματος

Σε αυτό το υποκεφάλαιο θα περιγράψουμε αναλυτικά τις λειτουργίες που παρέχει το back-end σύστημα και τις πληροφορίες που μας επιστρέφουν τα αρχεία PHP όταν τα καλούμε μέσα από την εφαρμογή.

2.2.1 Δημιουργία connection

Δίνοντας 4 συγκεκριμένα στοιχεία της βάσης δεδομένων που χρησιμοποιούμε, μπορούμε να τη συνδέσουμε μεταξύ της εφαρμογής και του εξυπηρετητή που έχουμε στη διάθεση μας. Αυτά είναι: 1) το όνομα του server που χρησιμοποιούμε, 2) το όνομα χρήστη (username) της βάσης δεδομένων, 3) τον κωδικό πρόσβασης (password) της βάσης δεδομένων (αν υπάρχει) και 4) την ονομασία της βάσης δεδομένων. Η σύνδεση αυτή γίνεται πολύ εύκολα με την παρακάτω γραμμή κώδικα PHP:

```
$conn = mysqli_connect($servername, $username, $password, $dbname);
```

2.2.2 Εμφάνιση υποκαταστημάτων ΙΚΑ

Για να εμφανίσουμε όλα τα καταστήματα που περιέχονται στη βάση δεδομένων μας, χρησιμοποιούμε ένα απλό SELECT SQL ερώτημα:

```
$sql = "SELECT * FROM stores";
```

Από το αποτέλεσμα αυτού του ερωτήματος, δημιουργείται ένας πίνακας με όλα τα στοιχεία των καταστημάτων που υπάρχουν αποθηκευμένα. Το κάθε κατάστημα μετατρέπεται σε ένα JSON αντικείμενο (*JSON Object*), και στη συνέχεια όλα τα καταστήματα σε ένα JSON πίνακα (*JSON Array*). Ένα μέρος του αποτελέσματος της εν λόγω μετατροπής είναι το εξής:

```
{
    "stores": [
        {
            "id": "2",
            "name": "ΙΚΑ Ευόσμου",
            "address": "Καραολή Δημητρίου 12",
            "postal_code": "56224",
            "email": "ika504@otenet.gr",
```

```
            "phone": null,
            "latitude": "40.6640291",
            "longitude": "22.9022909"
        },
        {
            "id": "3",
            "name": "ΙΚΑ Θεσσαλονίκης",
            "address": "Αριστοτέλους 15 - 17",
            "postal_code": "54624",
            "email": "ypok501@otenet.gr",
            "phone": "2310294700",
            "latitude": "40.6344775",
            "longitude": "22.9426441"
        }
    ]
}
```


2.2.3 Εμφάνιση τμημάτων ανά υποκατάστημα ΙΚΑ

Μόλις ο χρήστης επιλέξει ένα κατάστημα, θα εμφανιστεί μία νέα λίστα με τα τμήματα που περιέχει το εν λόγω υποκατάστημα. Για να γίνει αυτή η εμφάνιση, γίνεται πρώτα μια προεργασία στη βάση δεδομένων. Όπως περιγράφτηκε παραπάνω, έχουμε δημιουργήσει έναν πίνακα “*store_departments*” και έναν πίνακα “*departments*”. Μαζί με το *store_id* (ουσιαστικά το πεδίο *id* του πίνακα “*stores*”), οι εγγραφές του πίνακα “*store_departments*” περιέχουν στη μία μεριά το *store_id* και στην άλλη το *department_id* (ουσιαστικά το πεδίο *id* του πίνακα “*departments*”). Με αυτόν τον τρόπο, φτιάχνουμε έναν συσχετιστικό πίνακα με όλα τα τμήματα του κάθε υποκαταστήματος ΙΚΑ. Αυτό, θα γίνει με τον κατάλληλο κώδικα SQL, χρησιμοποιώντας INNER JOIN μεταξύ των πινάκων “*stores*” και “*departments*”:

```
// finds which departments are in the selected store
$sql = "SELECT departments.id, departments.name
```

```
FROM ((store_departments
INNER JOIN stores ON stores.id=store_departments.store_id)
INNER JOIN departments ON departments.id=store_departments.department_id)
WHERE store_id = " . $store_id;
```

Αν, για παράδειγμα, είχαμε επιλέξει το “ΙΚΑ Πύλης Αξιού”, ο εξυπηρετητής θα επέστρεφε το ακόλουθο JSON Array:

```
{
    "departments": [
        {
            "department_id": "4",
            "department_name": "Τμήμα Εσόδων - Οικοδ. Έργων",
            "active_tickets": "5",
            "waiting_time": " 5’ 50" "
        },
        {
            "department_id": "8",
            "department_name": "Τμήμα Παροχών Ασθενείας",
            "active_tickets": "2",
            "waiting_time": " 2’ 45" "
        },
        {
            "department_id": "6",
            "department_name": "Τμήμα Οικονομικό",
            "active_tickets": "1",
            "waiting_time": " 4’ 30" "
        },
        {
            "department_id": "1",
            "department_name": "Τμήμα Διοικητικό",
            "active_tickets": "10",
```

```
                "waiting_time": " 8’ 40" "
            }
        ]
}
```

Τα πεδία “*active_tickets*” και “*waiting_time*” υπολογίζονται από τον εξυπηρετητή και θα αναλυθούν περαιτέρω σε παρακάτω υποκεφάλαιο.

2.2.4 Έκδοση εισιτηρίου

Την στιγμή που ο χρήστης επιλέξει τμήμα, ο εξυπηρετητής ελέγχει εάν έχει ήδη βγάλει εισιτήριο στο συγκεκριμένο τμήμα/ουρά, για να εμφανίσει τις κατάλληλες πληροφορίες. Αυτό είναι πολύ εύκολο να το δούμε, αφού απλώς ελέγχουμε εάν το πεδίο “*time_finished*” είναι null, εάν δηλαδή ακόμη ο χρήστης δεν έχει εξυπηρετηθεί.

```
// finds if the user has already ticket in a specific department
$sql_search = "SELECT *
            FROM queue_table_". $store_id ."
            WHERE imei = ". $imei ."
            AND department_id = ". $department_id ."
            AND time_finished IS NULL";
```

Η μοναδικότητα του χρήστη, πιστοποιείται από τον αριθμό IMEI, που είναι μοναδικός για κάθε συσκευή, γι’ αυτό και χρησιμοποιείται στο SQL ερώτημα.

Εφόσον, λοιπόν, ο χρήστης δεν έχει βγάλει εισιτήριο, και επιλέξει να βγάλει καινούριο, υπολογίζουμε τον μέγιστο αριθμό εισιτηρίου που έχει εκδοθεί μέχρι εκείνη τη στιγμή και προσθέτουμε + 1, το καινούριο δηλαδή.

```
// finds the last ticket that has been issued and issues the next one to the user
$sql_max_ticket = "SELECT MAX(ticket_number) AS ticket_number
            FROM queue_table_". $store_id ."
            WHERE department_id = ". $department_id;
$ticket_number = $ticket_row['ticket_number'] + 1;
```

Η έτοιμη μέθοδος *MAX()* επιστρέφει τη μέγιστη τιμή ενός ερωτήματος SQL.

Στη συνέχεια, εισάγουμε στη βάση τα στοιχεία για το εισιτήριο που μόλις εκδόθηκε:

```
// inserts the issued ticket to the database
$sql = "INSERT INTO queue_table_". $store_id .
      "(department_id, ticket_number, imei, reg_id, time_entered)
       VALUES (". $department_id .", ". $ticket_number .", '". $imei ."', '". $reg_id ."', CURTIME())";
```

Το πεδίο “reg_id” είναι το registration_id που δημιουργείται αυτόματα κατά την πρώτη εκκίνηση της εφαρμογής στο κινητό, αποθηκεύεται στη βάση, και χρησιμοποιείται όταν το σύστημα στέλνει ειδοποίηση για την πορεία της ουράς και την τρέχουσα θέση του χρήστη σε αυτή. Περισσότερες λεπτομέρειες γι’ αυτή τη λειτουργία θα υπάρξουν σε επόμενο υποκεφάλαιο. Επίσης, η μέθοδος *CURTIME()* (υπάρχει και σαν CURRENT_TIME()) είναι μία από τις πολλές έτοιμες μεθόδους που προσφέρει η MySQL και επιστρέφει την τρέχουσα ώρα σε μορφή HH:MM:SS.

Τέλος, αυτό που επιστρέφει στην εφαρμογή, η συγκεκριμένη λειτουργία έκδοσης εισιτηρίου είναι το ακόλουθο JSON Object:

```
{
      "status":"200",
      "ticket_number":9,
      "active_tickets":"6",
       "waiting_time":" 1' 3" "
}
```

2.2.5 Login

Για να εκτελεστεί η συγκεκριμένη λειτουργία, χρειάζονται 2 παράμετροι: το *όνομα χρήστη* (username) και ο *κωδικός πρόσβασης* (password) του υπαλλήλου που επιχειρεί να κάνει login στο σύστημα. Έπειτα, ελέγχουμε εάν ό,τι εισήγαγε ο χρήστης είναι σωστό, αν δηλαδή υπάρχει χρήστης στη βάση δεδομένων με τα παραπάνω στοιχεία. Ο απαιτούμενος κώδικας SQL είναι ο εξής:

```
// finds if the username and password that the agent entered are correct

$sql = "SELECT users.id AS user_id, stores.id AS store_id, stores.name AS store_name, departments.id AS department_id, departments.name AS department_name

    FROM (((users

    INNER JOIN store_departments ON users.store_department_id = store_departments.id)

    INNER JOIN stores ON stores.id = store_departments.store_id)

    INNER JOIN departments ON departments.id = store_departments.department_id)

    WHERE users.username ='" . $username . "' AND users.password ='" . $password . "'";
```

Όπως φαίνεται και από τον κώδικα, κρατάμε πληροφορίες τόσο για το όνομα του καταστήματος, όσο και του τμήματος, που θα μας χρησιμεύσουν για να τα εμφανίσουμε σε επόμενη οθόνη της εφαρμογής. Η απάντηση (response) του εξυπηρετητή στην κλήση αυτής της λειτουργίας είναι το παρακάτω JSON Object:

```
{

    "status": "200",

    "user_id": "5",

     "store_id": "1",

    "store_name": "ΙΚΑ 25ης Μαρτίου",

    "department_id": "1",

    "department_name": "Τμήμα Διοικητικό",

    "active_tickets": "6",

     "serving_time": " 21" "

}
```

2.2.6 Εξυπηρέτηση εισιτηρίου

Με αυτή τη λειτουργία, ο υπάλληλος του εκάστοτε καταστήματος ΙΚΑ έχει τη δυνατότητα να εξυπηρετήσει τους χρήστες που έχουν εισέλθει στο τμήμα/ουρά που εξυπηρετεί. Αρχικά το σύστημα ψάχνει να βρει τον χρήστη με το μικρότερο αριθμό εισιτηρίου:

```
// finds the next user to serve, and creates a timestamp
$sql_min_ticket = "SELECT id, MIN(ticket_number) AS ticket_number
                   FROM queue_table_". $store_id ."
                   WHERE time_served IS NULL
                   AND department_id =". $department_id;
```

Η μέθοδος *MIN()* είναι ακόμη μία από τις έτοιμες μεθόδους της MySQL και επιστρέφει την ελάχιστη τιμή μιας έκφρασης, ενός ερωτήματος SQL.

Αυτό γίνεται, ούτως ώστε να αποθηκευτεί στη βάση δεδομένων ένα ακριβές αποτύπωμα χρόνου (timestamp) με την ακριβή ώρα που ξεκίνησε η εξυπηρέτηση του χρήστη. Αυτό γίνεται με τον ακόλουθο κώδικα SQL:

```
$sql_time_served = "UPDATE queue_table_". $store_id ."
                    SET time_served = CURTIME(), user_id = ". $user_id ."
                   WHERE id = ". $id;
```

Η PHP μεταβλητή “user_id” χρησιμοποιείται από τη βάση, για να προσδιορίσει το id, την ταυτότητα του υπαλλήλου που εξυπηρετεί.

Την επόμενη φορά που θα ζητηθεί από την εφαρμογή η συγκεκριμένη λειτουργία το σύστημα θα κάνει αναζήτηση για τον χρήστη που εξυπηρετεί αυτή τη στιγμή ο υπάλληλος με ένα ορισμένο *user_id*. Δηλαδή, ψάχνει στη βάση δεδομένων, για κάποιον χρήστη, για τον οποίο έχει οριστεί το *time_served*, που αντιστοιχεί στη χρονική στιγμή που ξεκίνησε να εξυπηρετείται και ταυτόχρονα δεν έχει οριστεί ακόμη το πεδίο *time_finished*, που αντιστοιχεί στη χρονική στιγμή που ο υπάλληλος τελείωσε με την εξυπηρέτηση του συγκεκριμένου πελάτη και πίεσε το κουμπί για να προσέλθει ο επόμενος προς εξυπηρέτηση.

```
// finds the correct ticket and creates a timestamp for 'time_finished', when the agent finishes with serving
$sql = "SELECT id
        FROM queue_table_". $store_id ."
        WHERE time_finished IS NULL
        AND user_id = ". $user_id;
```

Αφού βρεθεί, λοιπόν, ο χρήστης, θα πρέπει να ενημερωθεί η βάση δεδομένων με το timestamp εκείνης της στιγμής που ο υπάλληλος πάτησε το κουμπί.

```
$sql_time_finished = "UPDATE queue_table_". $store_id ."
                     SET time_finished = CURTIME()
                     WHERE id = ". $id;
```

Με αυτόν τον τρόπο, κλείνει ο κύκλος εξυπηρέτησης ενός πελάτη, αφού και τα 3 πεδία που χρησιμοποιούνται στη βάση δεδομένων για τον εντοπισμό της κατάστασης του χρήστη (ώρα εισαγωγής στην ουρά/τμήμα – time_entered, ώρα που ξεκίνησε η εξυπηρέτηση – time_served, ώρα που τελείωσε η εξυπηρέτηση – time_finished) έχουν τιμές timestamp. Αυτό σημαίνει ότι έχουν πλέον εξυπηρετηθεί και αποτελούν παρελθόν για την ουρά αναμονής. Αυτό που επιστρέφεται στην εφαρμογή, από τη συγκεκριμένη λειτουργία είναι το ακόλουθο JSON Object:

```
{
   "status":"200",
   "current_ticket":"4",
   "next_ticket":5,
   "active_tickets":"5",
   "serving_time":"1h 45' 43""
}
```

Το πεδίο “*status*” υπάρχει για να πιστοποιηθεί στην εφαρμογή ότι η λειτουργία εκτελέστηκε σωστά και επιστρέφει αποτέλεσμα. Επίσης, σημαντικό πεδίο είναι και το “*next_ticket*”, που αποτελεί τον δείκτη για την εφαρμογή, ούτως ώστε να γνωρίζει το id του επόμενου χρήστη προς εξυπηρέτηση.

2.2.7 Notifications

Η συγκεκριμένη λειτουργία, αποτελεί μία υπολειτουργία της παραπάνω, αφού υλοποιείται και εκτελείται τη στιγμή που ο υπάλληλος του ΙΚΑ πατήσει το κουμπί για την εξυπηρέτηση του επόμενου πελάτη. Αρχικά παρατίθεται ο κώδικας PHP μαζί με

ερωτήματα SQL που περιέχουν την υλοποίησή της και στη συνέχεια θα σχολιαστεί και θα επεξηγηθεί εκτενώς:

```
$sql_tickets = "SELECT reg_id FROM queue_table_". $store_id ."
                 WHERE department_id =". $department_id . "
                AND time_served IS NULL";
$tickets = array();
if (mysqli_num_rows($notif_tickets) > 0) {
   while($row = mysqli_fetch_array($notif_tickets)) {
      $ticket['reg_id'] = $row['reg_id'];
      array_push($tickets, $ticket['reg_id']);
   }
}

for ($i=0; $i < count($tickets); $i++) {
   switch ($i+1) {
      case 1:
      case 2:
      case 3:
      case 4:
      case 5:
      case 10:
         $title = $store_name ." - ". $department_name;
         $message = "Είστε ο ". ($i + 1) ."ος στην ουρά.";
         if($i + 1 == 1) {
            $message = "Είστε ο επόμενος προς εξυπηρέτηση!";
         }

         $push->setTitle($title);
         $push->setMessage($message);
         $push->setIsBackground(FALSE);
```

```
        $push->setPayload($payload);


        $json = '';
        $response = '';


        $json = $push->getPush();
        $response = $firebase->send($tickets[$i], $json);
        break;
    }
}
```

Στο πρώτο μέρος του κώδικα, εκτελούμε ένα ερώτημα SQL για να βρούμε όλα τα *registration_id* , που όπως έχει αναφερθεί το πεδίο αυτό δημιουργείται αυτόματα μετά την πρώτη εκτέλεση της εφαρμογής. Εφόσον υπάρχει αποτέλεσμα σε αυτό το ερώτημα, τα αποθηκεύουμε σε έναν πίνακα που θα χρησιμοποιηθεί παρακάτω με την εντολή *array_push($tickets, $ticket['reg_id']);* .

Στο αμέσως επόμενο κομμάτι κώδικα, έχει δημιουργηθεί ένας βρόγχος επανάληψης, ο οποίος διαπερνά όλα τα στοιχεία του πίνακα που δημιουργήθηκε προηγουμένως. Με το switch – case, ουσιαστικά απομονώνουμε τα *reg_id* στις θέσεις 1,2,3,4,5 και 10, για να αποστείλουμε ειδοποιήσεις στους χρήστες που εκείνη τη στιγμή κατέχουν τις παραπάνω θέσεις στην ουρά. Η ειδοποίηση περιέχει τις πλέον απαραίτητες πληροφορίες που χρειάζεται ο χρήστης, με τίτλο το όνομα του υποκαταστήματος ακολουθούμενο από το τμήμα και σαν κύριο μήνυμα είτε το κείμενο: «Είστε ο επόμενος προς εξυπηρέτηση», είτε το: «Είστε ο Χ^{ος} στην ουρά» σε οποιαδήποτε άλλη περίπτωση.

2.2.8 Βοηθητικές μέθοδοι

Σε αυτό το υποκεφάλαιο παρατίθενται 3 βασικές μέθοδοι που χρησιμοποιήθηκαν στη μεριά του εξυπηρετητή και βοήθησαν στον υπολογισμό των

ενεργών εισιτηρίων ανά ουρά, όπως επίσης και στον υπολογισμό των μέσων χρόνων αναμονής και εξυπηρέτησης (waitingTime και servingTime αντίστοιχα).

2.2.8.1 activeTickets()

```
// returns the active tickets that has been issued in a specific department at a store
function activeTickets($store_id, $department_id) {
    global $conn;
    $sql = "SELECT COUNT(id) AS size FROM queue_table_". $store_id ."
             WHERE time_served IS NULL
            AND time_finished IS NULL
           AND department_id =". $department_id;

    $result = mysqli_query($conn, $sql);

    if ($result) {
        while($row = mysqli_fetch_row($result)) {
            return $row[0];
        }
    } else {
        return 0;
    }
}
```

Στην παραπάνω μέθοδο, περνάμε σαν παραμέτρους το *store_id* και το *department_id* για να μπορέσουμε να προσδιορίσουμε, στο ερώτημα SQL που θα εκτελέσουμε, τον αριθμό των ενεργών εισιτηρίων στην συγκεκριμένη ουρά. Αυτό γίνεται απλά με την χρήση της SQL μεθόδου *COUNT()*, η οποία επιστρέφει τον αριθμό των εγγραφών που ταιριάζουν με τα κριτήρια που θέλουμε. Στην προκειμένη περίπτωση θέλουμε να μας επιστραφεί ο αριθμός των εισιτηρίων που δεν έχουν ακόμη εξυπηρετηθεί από κάποιον υπάλληλο, γι’ αυτό και χρησιμοποιούμε το κριτήριο *time_served IS NULL*.

2.2.8.2 waitingTime()

```
// returns the actual waiting time, according to the average waiting time of each user
function waitingTime($store_id, $department_id) {

    global $conn;
    $sql = "SELECT name, AVG(temp.waitingTime) AS theResult FROM (
                SELECT 'test' AS name,
                UNIX_TIMESTAMP(time_finished) - UNIX_TIMESTAMP(time_entered) AS waitingTime
               FROM queue_table_" . $store_id . "
               WHERE department_id = ". $department_id ." AND time_served IS NOT NULL
               AND time_finished IS NOT NULL) temp GROUP BY temp.name";

    $result = mysqli_query($conn, $sql);
    if($result) {
        $row = mysqli_fetch_assoc($result);

        $waiting = gmdate("H:i:s:", $row['theResult']);
        $waiting = explode(':', $waiting);
        $value = "";
        if($waiting[0] != "00")
            $value .= intval($waiting[0])."h";

        if($waiting[1] != "00")
            $value .= " ".intval($waiting[1])." ' ";

        if($waiting[2] != "00")
            $value .= " ".intval($waiting[2])." '' ";

        if($value == "")
            $value = "0"";
```

```
        return $value;
    }
}
```

2.2.8.3 servingTime()

```
// returns the actual waiting time, according to the average serving time on each user
function servingTime($store_id, $department_id) {
            global $conn;
            $sql = "SELECT name, AVG(temp.serviceTime) AS theResult FROM (
              SELECT 'test' AS name,
              UNIX_TIMESTAMP(time_finished) - UNIX_TIMESTAMP(time_served) AS serviceTime
             FROM queue_table_" . $store_id . "
             WHERE department_id = ". $department_id ." AND time_served IS NOT NULL
             AND time_finished IS NOT NULL) temp GROUP BY temp.name";

    $result = mysqli_query($conn, $sql);
    if ($result) {
        $row = mysqli_fetch_assoc($result);

        $waiting = gmdate("H:i:s:", $row['theResult']);
        $waiting = explode(':', $waiting);
        $value = "";
        if($waiting[0] != "00")
            $value .= intval($waiting[0])."h";

        if($waiting[1] != "00")
            $value .= " ".intval($waiting[1])." ' ";

        if($waiting[2] != "00")
```

```
            $value .= " ".intval($waiting[2])." '' ";

        if($value == "")
            $value = "0'";

        return $value;
    }
}
```

Θα σχολιάσουμε συνολικά και τις 2 μεθόδους, αφού ουσιαστικά το μεγαλύτερο μέρος του κώδικα είναι παρόμοιο και το μόνο που αλλάζει είναι το SQL ερώτημα που εκτελεί ο εξυπηρετητής.

Όπως βλέπουμε, περνάμε ξανά σαν παραμέτρους το *store_id* και το *department_id*, για να προσδιορίσουμε το ακριβές υποκατάστημα και τμήμα για το οποίο θα υπολογιστούν οι μέσοι χρόνοι αναμονής/εξυπηρέτησης. Στο SQL ερώτημα που χρησιμοποιούμε για να υπολογίσουμε τους μέσους χρόνους, αρχικά, δημιουργείται ένας πίνακας temp, ο οποίος απλώς περιέχει τους χρόνους αναμονής/εξυπηρέτησης του κάθε χρήστη. Ο χρόνος αναμονής υπολογίζεται κάνοντας την αφαίρεση του timestamp που περιέχει τον χρόνο εξόδου (*time_finished*) από την ουρά μείον το timestamp που περιέχει τον χρόνο εισόδου (*time_entered*) στην ουρά. Αντίστοιχα, ο χρόνος εξυπηρέτησης υπολογίζεται κάνοντας την αφαίρεση του timestamp που περιέχει τον χρόνο εξόδου (*time_finished*) από την ουρά μείον τη χρονική στιγμή που τελείωσε η εξυπηρέτηση (*time_served*). Στη συνέχεια, το ερώτημα ουσιαστικά υπολογίζει με την SQL μέθοδο *AVG()* τον μέσο όρο όλων (average) των εγγραφών που περιέχει ο προσωρινός πίνακας temp.

Αφού αποθηκευτεί το παραπάνω αποτέλεσμα σε έναν πίνακα, σειρά παίρνει ο κώδικας PHP που προσπαθεί με συγκεκριμένες μεθόδους να δημιουργήσει το αποτέλεσμα που θα επιστρέψει η κάθε μέθοδος. Αρχικά με την PHP μέθοδο *gmdate()*, μετατρέπεται το timestamp που έχουμε αποθηκευμένο σε String με συγκεκριμένη μορφή που ορίζουμε εμείς, που στην περίπτωση μας είναι ”H:m:s”. Στη συνέχεια, χρησιμοποιώντας την PHP μέθοδο *explode()* δημιουργείται ένας ακόμη

προσωρινός πίνακας που σε κάθε κελί περιέχει το String που υπάρχει ακριβώς πριν τον χαρακτήρα που ορίσαμε στο πρώτο όρισμα της μεθόδου, στην περίπτωση μας την άνω κάτω τελεία. Έτσι, δημιουργείται ένας πίνακας με κελιά που αντιστοιχούν στις ώρες (αν υπάρχουν), στα λεπτά (αν υπάρχουν) και τα δευτερόλεπτα.

2.2.9 Events

Μια πολύ σημαντική λειτουργία που προσφέρει η MySQL είναι η δημιουργία συμβάντων (events). Μέσω αυτών, μπορεί να εκτελείται κάποιο προγραμματισμένο χρονικά ερώτημα SQL, οποιαδήποτε στιγμή το ορίσει ο χρήστης. Στη δική μας περίπτωση, οι ενέργειες που πραγματοποιούν τα events, είναι ο καθαρισμός (*TRUNCATE*) της βάσης δεδομένων, και πιο συγκεκριμένα των πινάκων "*queue_table_*" τα μεσάνυχτα κάθε ημέρας. Η εντολή αυτή, χρησιμοποιείται για να διαγράψει τα δεδομένα μέσα σε έναν πίνακα, χωρίς όμως να διαγράψει τον ίδιο τον πίνακα. Ο λόγος που γίνεται αυτή η λειτουργία είναι προφανής, που δεν είναι άλλος από το να καθαριστεί η βάση από τα δεδομένα και να ξεκινήσει ο εξυπηρετητής να δημιουργεί καινούργια στατιστικά στοιχεία για τον μέσο χρόνο αναμονής και εξυπηρέτησης.

2.2.10 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια προσπάθεια να αναλυθούν όλες οι λειτουργίες που μπορεί να επιτελέσει ο εξυπηρετητής (server) με τη χρήση κώδικα PHP και MySQL. Με το τέλος αυτού του κεφαλαίου, ολοκληρώνεται η αναφορά και ανάλυση του πρώτου μέρους της παρούσας πτυχιακής εργασίας.

Τα επόμενα κεφάλαια που ακολουθούν, αναφέρονται στο επόμενο κομμάτι, τον πελάτη (client), και πιο συγκεκριμένα στο Android.

3. Εισαγωγή στο λειτουργικό σύστημα Android

3.1 Εισαγωγή

Σε αυτό το εισαγωγικό κεφάλαιο επεξηγείται το τι ακριβώς είναι το λειτουργικό σύστημα Android, πάνω στο οποίο αναπτύχθηκε η εφαρμογή της παρούσας πτυχιακής εργασίας. Επιπρόσθετα, θα αναλυθεί η αρχιτεκτονική του, αλλά και εκείνα τα χαρακτηριστικά τα οποία το έχουν οδηγήσει να βρίσκεται στη κορυφή των πιο δημοφιλών λειτουργικών συστημάτων.

3.2 Τι είναι το Android

Το Android είναι ένα λειτουργικό σύστημα ανοικτού κώδικα για κινητές συσκευές, βασισμένο στον Linux πυρήνα και σχεδιασμένο αρχικά για συσκευές με οθόνες αφής, όπως έξυπνα τηλέφωνα (Smartphones) και ταμπλέτες (Tablets). Στη συνέχεια, με τη φήμη που απέκτησε στην αγορά, προσαρμόστηκε να χρησιμοποιείται και σε ρολόγια (Android Wear), τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) καθώς και σε συσκευές διαδικτύου των πραγμάτων (Internet of Things), έχοντας το καθένα το δικό του εξειδικευμένο περιβάλλον εργασίας (User Interface).

Το Android αρχικά αναπτύχθηκε από την Android Inc., την οποία αγόρασε η Google το 2005, και κυκλοφόρησε το 2007, μαζί με την ίδρυση του Open Handset Alliance - μιας κοινοπραξίας εταιρειών υλικού, λογισμικού και τηλεπικοινωνιών που αφιερώνεται στην προώθηση ανοικτών προτύπων για κινητές συσκευές. Αυτή τη στιγμή πρόκειται για το πιο ευρέως διαδεδομένο λογισμικό στο κόσμο. Οι εφαρμογές Android μπορούν να εγκατασταθούν από το Google Play το οποίο διαθέτει πάνω από 2,7 εκατομμύρια εφαρμογές (μέχρι τον Φεβρουάριο του 2017). Το Android είναι το λειτουργικό σύστημα με τις καλύτερες πωλήσεις σε tablets από το 2013 και τρέχει στη συντριπτική πλειοψηφία των smartphones παγκοσμίως. Επίσης, τον Μάιο του 2017, η Google ενημέρωσε ότι το Android έχει δύο δισεκατομμύρια μηνιαίους ενεργούς χρήστες. Η τελευταία έκδοση η οποία είναι διαθέσιμη τη παρούσα χρονική στιγμή είναι η 7.1.2 με κωδική ονομασία Nougat.

Ο πηγαίος κώδικας (source code) του Android κυκλοφορεί από την Google με άδεια ανοικτού κώδικα, αν και οι περισσότερες συσκευές Android κυκλοφορούν τελικά με ένα συνδυασμό ελεύθερου/ανοικτού λογισμικού και ιδιόκτητου λογισμικού, ούτως ώστε η κάθε εταιρία να εξειδικεύει την εμπειρία χρήσης του λειτουργικού και να διαχωρίζεται από τις υπόλοιπες. Για την ανάπτυξή του, η Google επιτρέπει στους κατασκευαστές να συνθέσουν κώδικα με τη χρήση της γλώσσας προγραμματισμού Java και C++ μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Γενικότερα, το Android είναι δημοφιλές σε εταιρείες τεχνολογίας που απαιτούν ένα έτοιμο, χαμηλού κόστους και προσαρμόσιμο λειτουργικό σύστημα για συσκευές υψηλής τεχνολογίας.

3.3 Κύρια χαρακτηριστικά

Τα κυριότερα χαρακτηριστικά που οδήγησαν το Android να γίνει τόσο διαδεδομένο και νούμερο ένα στις πωλήσεις κινητών συσκευών ακολουθούν παρακάτω:

3.3.1 Δωρεάν και ανοιχτή πηγή

Το Android είναι μια πλατφόρμα ανοικτού κώδικα και δωρεάν για χρήση. Πιο συγκεκριμένα το λειτουργικό σύστημα έχει κατοχυρωθεί με την άδεια χρήσης GNU General Public Licence Version 2 (GPLv2) η οποία υποχρεώνει βελτιώσεις τρίτων να εξακολουθούν να εμπίπτουν στους όρους των προτύπων ανοικτού κώδικα. Αυτό σημαίνει ότι μπορεί οποιοσδήποτε να κλωνοποιήσει τον πρωτότυπο κώδικα του λειτουργικού συστήματος, να τον τροποποιήσει και να διαθέσει μια δικιά του έκδοση με ότι επιπλέον χαρακτηριστικά έχει προσθέσει, αρκεί να εξακολουθεί να παραμένει υπό τις άδειες του ανοικτού κώδικα. Το περιβάλλον δημιουργίας του Android διανέμεται μέσω της άδειας Apache Software Licence η οποία επιτρέπει προσθήκες κλειστού κώδικα. Αυτό σημαίνει ότι μπορεί μια εταιρεία να χρησιμοποιήσει τη πλατφόρμα και να της προσθέσει επιπλέον χαρακτηριστικά χωρίς να υποχρεούται να παρέχει το τελικό προϊόν με άδεια ανοικτού κώδικα, κάτι που συμβαίνει αρκετά συχνά θέλοντας έτσι η κάθε εταιρεία να διαφοροποιηθεί από τους υπόλοιπους ανταγωνιστές.

Το λειτουργικό σύστημα βέβαια παραμένει να είναι το Android. Οι αλλαγές αυτές συμβαίνουν σε υψηλό επίπεδο και συνήθως στοχεύουν στη διεπαφή χρήστη (User Interface - UI).

3.3.2 Δωρεάν διαθέσιμα εργαλεία ανάπτυξης λογισμικού

Όλα τα εργαλεία καθώς και το SDK - NDK (οι βιβλιοθήκες προγραμματισμού για την ανάπτυξη εφαρμογών σε Android) που θα χρειαστεί κάποιος για να αναπτύξει μια Android εφαρμογή παρέχονται δωρεάν για όλα τα γνωστά λειτουργικά συστήματα (Linux, Windows, MacOS).

3.3.3 Γνωστές γλώσσες προγραμματισμού

Η ανάπτυξη μιας Android εφαρμογής γίνεται είτε μέσω του SDK (Software Development Kit) το οποίο χρησιμοποιεί τη γλώσσα προγραμματισμού Java, είτε μέσω του NDK (Native Development Kit) το οποίο χρησιμοποιεί τη γλώσσα προγραμματισμού C++. Και οι δύο είναι πολύ γνωστές και δημοφιλείς έτσι ώστε δεν απαιτείται από τον προγραμματιστή να μάθει μια τελείως καινούργια γλώσσα η οποία θα χρησιμοποιείται αποκλειστικά και μόνο για το Android.

3.3.4 Google Play Store

Πρόκειται για την επίσημη ψηφιακή πλατφόρμα διανομής εφαρμογών για το Android. Δημιουργήθηκε και λειτουργεί από τη Google, επιτρέποντας στους χρήστες να περιηγηθούν και να κατεβάσουν εφαρμογές και στους προγραμματιστές να ανεβάσουν τις εφαρμογές τους, τις οποίες μπορούν να διαθέσουν είτε επί πληρωμής είτε δωρεάν. Για να ανεβάσει κάποιος μια εφαρμογή στο Google Play store θα πρέπει αρχικά να εγγραφεί σε αυτό, κάτι το οποίο κοστίζει 20$. Όταν μια καινούργια εφαρμογή ανέβει για πρώτη φορά, το Google Play store πραγματοποιεί κάποιους

ελέγχους για κακόβουλο λογισμικό, έτσι ώστε οι χρήστες να είναι σίγουροι ότι χρησιμοποιούν ασφαλείς και χωρίς ιούς εφαρμογές.

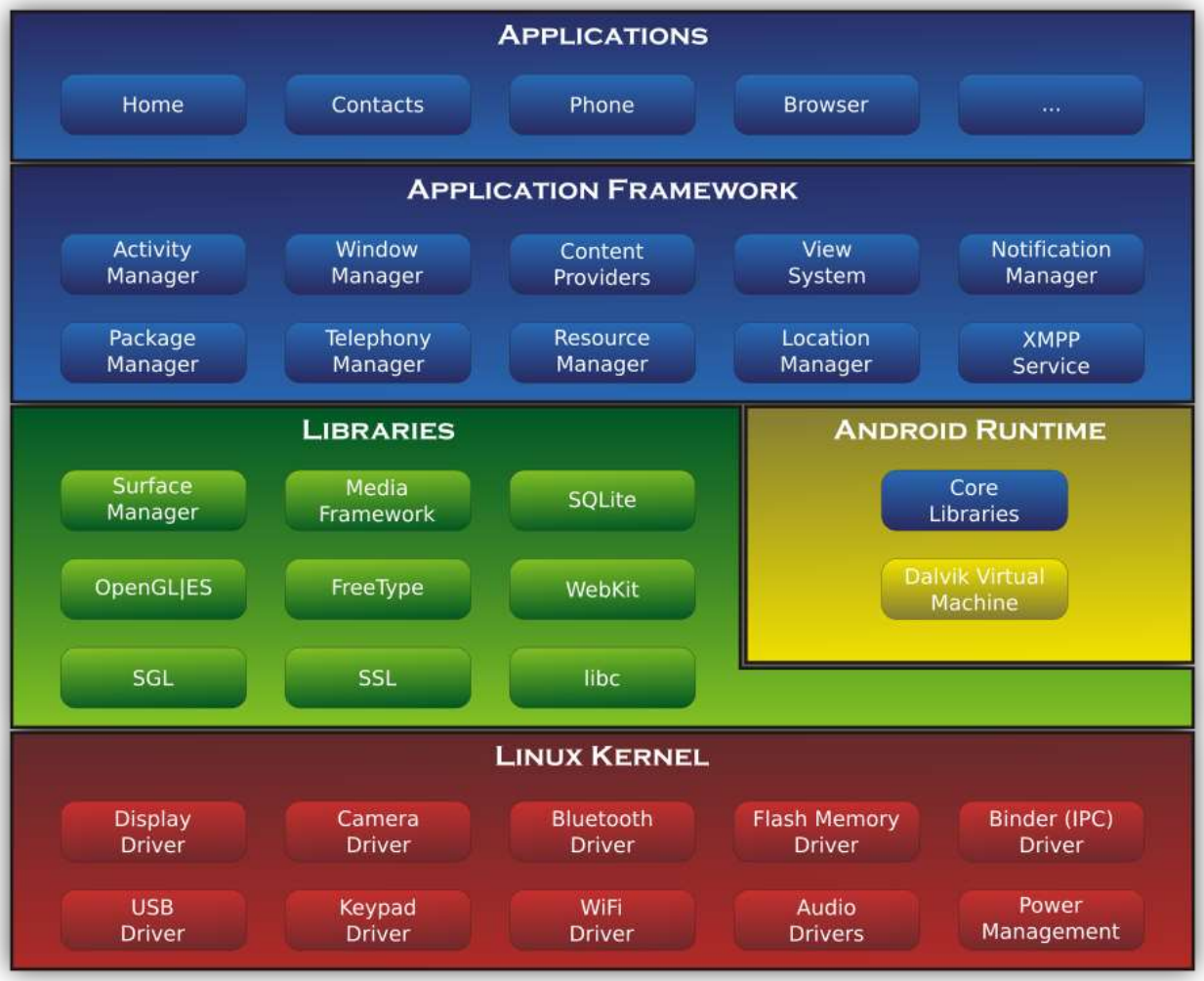
3.3.5 Material Design

Τον Νοέμβριου του 2014 η Google ανακοίνωσε την έκδοση 5.0 του Android με κωδική ονομασία Lollipop. Μεταξύ άλλων χαρακτηριστικών προσέφερε και το Material Design, το οποίο απέφερε δραστικές αλλαγές στον όλο σχεδιασμό της διεπαφής χρήστη (UI) στο Android. Η επιτυχία του Material Design ήταν τόσο μεγάλη, έχοντας σαν αποτέλεσμα να υιοθετηθεί από πολλές εφαρμογές ανεξαρτήτου πλατφόρμας και λειτουργικού συστήματος. Πρόκειται για ένα σύνολο σχεδιαστικών κανόνων επεκτείνοντας το μοντέλο των "καρτών", που είναι σχεδιασμός βασισμένος σε διάταξη πίνακα με κινούμενα σχέδια και μεταβάσεις που ανταποκρίνονται στις κινήσεις των χρηστών, καθώς και εφέ βάθους με κατάλληλο φωτισμό και σκίαση αντικειμένων. Ο σχεδιαστής του, Matias Duarte είχε δηλώσει «το Material Design έχει φυσικές επιφάνειες και άκρα. Οι σκιές δίνουν νόημα σε αυτό που αγγίζει ο χρήστης». Το σύνολο των κανόνων και των τεχνικών για έναν γραφιστικό σχεδιασμό μιας εφαρμογής στο Material design μπορούν να βρεθούν στην επίσημη ιστοσελίδα του [7]. Παρακάτω παρατίθεται μια εικόνα που παρουσιάζει ένα μέρος μιας εφαρμογής η οποία χρησιμοποιεί Material Design.

Εικόνα 4: Παράδειγμα Material Design [7]

3.4 Αρχιτεκτονική του Android

Για να καταλάβει κάποιος τον τρόπο με τον οποίο δουλεύει το Android αρκεί να μελετήσει την εικόνα 4 στην οποία φαίνονται τα διαφορετικά στρώματα από τα οποία αποτελείται το λειτουργικό σύστημα του Android. Αρχικά παρατίθεται η εικόνα και στην συνέχεια αναλύονται οι βασικοί τομείς στους οποίους χωρίζεται το λειτουργικό σύστημα.



Εικόνα 5: Αρχιτεκτονική του λειτουργικού Android

Όπως φαίνεται το λειτουργικό σύστημα χωρίζεται σε 6 τομείς και 5 στρώματα. Αναφέρονται συνοπτικά οι 6 βασικοί τομείς και η λειτουργία που επιτελεί ο καθένας.

- **Linux Kernel**

  Είναι ο πυρήνας στον οποίο βασίζεται το λειτουργικό σύστημα Android. Περιέχει όλους τους χαμηλού επιπέδου οδηγούς συσκευών (low level device drivers) που χρειάζονται για τα διάφορα υλικά μέρη της συσκευής καθώς και υποκείμενες λειτουργίες όπως διαχείριση νημάτων (threads) ή μνήμης χαμηλού επιπέδου.

- **Android Runtime**

  Παρέχει ένα σύνολο από βασικές βιβλιοθήκες που δίνουν τη δυνατότητα στους προγραμματιστές να γράψουν εφαρμογές για το Android σε γλώσσα προγραμματισμού Java. Ακόμη, περιέχει την εικονική μηχανή (virtual machine) ART (για εκδόσεις του Android >= 5.0) ή Dalvik (για εκδόσεις του android < 5.0) οι οποίες επιτρέπουν κάθε εφαρμογή να τρέξει σε δική της διαδικασία, με δική της ξεχωριστή μνήμη. Και οι 2 είναι ειδικές εκδόσεις του JVM (Java Virtual Machine) τροποποιημένες για κινητές συσκευές. Η ART περιέχει παραπάνω χαρακτηριστικά από τη Dalvik και πετυχαίνει καλύτερες αποδόσεις.

- **Native C/C++ Libraries**

  Εδώ βρίσκεται ο κώδικας όλων των χαρακτηριστικών του Android. Πολλά βασικά χαρακτηριστικά και υπηρεσίες του είναι γραμμένες σε C/C++. Το Android παρέχει ένα Java API (Application Programming Interface) για να προσφέρει τη λειτουργία αυτών των χαρακτηριστικών στις εφαρμογές.

- **Java Application Framework**

  Όλα τα χαρακτηριστικά του Android (οι προηγούμενες βιβλιοθήκες) γραμμένα στη γλώσσα προγραμματισμού Java. Ό,τι χρειάζεται ένας προγραμματιστής για να αναπτύξει μια εφαρμογή βρίσκεται εδώ.

- **System Applications**

  Εδώ βρίσκονται όλες οι εφαρμογές που είναι εγκατεστημένες στη κινητή συσκευή ή πρόκειται να εγκατασταθούν σε αυτή. Οι εφαρμογές μπορούν να

αλληλεπιδρούν μεταξύ τους, για παράδειγμα εάν μια εφαρμογή χρειάζεται να στείλει κάποιο SMS μπορεί να καλέσει την ήδη εγκατεστημένη SMS εφαρμογή για να το στείλει.

3.5 Επίλογος

Συνοψίζοντας, σε αυτό το κεφάλαιο έγινε μια εμπεριστατωμένη εισαγωγή στο λειτουργικό σύστημα Android προκειμένου ο αναγνώστης να κατανοήσει τι είναι το Android με πολλαπλές αναφορές σε ορισμούς, στα χαρακτηριστικά του αλλά και στην αρχιτεκτονική του από την οποία απαρτίζεται.

4. Προγραμματισμός στο Android

4.1 Εισαγωγή

Σε αυτό το κεφάλαιο θα γίνει μια περιγραφή για το τι κατηγορίες εφαρμογών μπορούν να υπάρξουν στο Android καθώς και μια ανάλυση των συστατικών στοιχείων που μπορούν να χρησιμοποιηθούν σε μια Android εφαρμογή. Πρόκειται για θεμελιώδεις έννοιες με τις οποίες εάν εξοικειωθεί ο αναγνώστης, θα είναι σε θέση να ξεκινήσει να προγραμματίζει στο Android.

4.2 Κατηγορίες εφαρμογών

Οι τύποι εφαρμογών που μπορούν να αναπτυχθούν στο Android είναι τέσσερεις και αναλύονται στη συνέχεια.

4.2.1 Εφαρμογές Προσκηνίου (Foreground Applications)

Πρόκειται για το πιο συνηθισμένο είδος εφαρμογών. Ξεκινούν τη λειτουργία τους όποτε ο χρήστης τις ανοίξει (όπου και ξεκινούν να εκτελούνται στο προσκήνιο) και η λειτουργία τους αναστέλλεται όταν δεν είναι πλέον ορατές, όταν δηλαδή ο χρήστης μεταφερθεί σε μια άλλη εφαρμογή ή στη κύρια οθόνη (Home Screen). Όταν μια τέτοια εφαρμογή αναστέλλει τη λειτουργία της, σταματάει και ο κύκλος ζωής της, κάτι που σημαίνει πως όταν ο χρήστης ξανά ανοίξει την εφαρμογή αυτή είναι πιθανό να μην βρίσκεται στην ίδια κατάσταση με αυτή που βρισκόταν προτού ανασταλεί. Οπότε, ο προγραμματιστής είναι υπεύθυνος να αποθηκεύει την κατάσταση της πριν η εφαρμογή απομακρυνθεί από το προσκήνιο ώστε αν χρειαστεί, να επανέλθει στην ίδια κατάσταση με πριν. Τέλος, εφόσον πρόκειται για εφαρμογές προσκηνίου θα πρέπει να διαθέτουν μια ευπαρουσίαστη διεπαφή χρήστη ώστε η χρήση τους να είναι ευχάριστη/εύκολη.

4.2.2 Εφαρμογές Παρασκηνίου (Background Applications)

Οι εφαρμογές αυτής της κατηγορίας εκτελούνται στο παρασκήνιο και δεν αλληλεπιδρούν σχεδόν καθόλου με το χρήστη. Η χρήση τους αφορά τον έλεγχο για διάφορα μηνύματα από ενέργειες που εκτελούνται από το υλικό, το σύστημα ή άλλες

εφαρμογές. Παραδείγματος χάριν, υπάρχει μια εφαρμογή παρασκηνίου η οποία ελέγχει/περιμένει για εισερχόμενες κλήσεις προς τη κινητή συσκευή.

4.2.3 Διακοπτόμενες εφαρμογές (Always On Applications)

Εδώ κατατάσσονται όλες οι εφαρμογές οι οποίες χρειάζεται και να αλληλεπιδρούν με το χρήστη μέσω μιας διεπαφής αλλά και να αντιδρούν σε γεγονότα όταν δεν βρίσκονται στο προσκήνιο. Έτσι, πρόκειται για μια ένωση εφαρμογών προσκηνίου- παρασκηνίου. Τέτοιες εφαρμογές θα πρέπει να ενημερώνουν πάντοτε το χρήστη, σε όποια κατάσταση και αν βρίσκονται. Είτε με ενημέρωση του UI εάν βρίσκονται στο προσκήνιο, είτε με κάποια ειδοποίηση εάν βρίσκονται στο παρασκήνιο. Παραδείγματα τέτοιων εφαρμογών είναι αυτές του τύπου “email”.

4.2.4 Widgets

Τέτοιες εφαρμογές παρέχουν διαδραστικά γραφικά στοιχεία τα οποία μπορούν να εισαχθούν στην αρχική οθόνη (Home Screen), ώστε να ενημερώνουν τον χρήστη χωρίς να χρειάζεται να εισέλθει στην εφαρμογή. Για παράδειγμα, η στάθμη της μπαταρίας ή η πρόβλεψη του καιρού είναι τέτοιες εφαρμογές.

4.3 Βασικά συστατικά στοιχεία μιας εφαρμογής

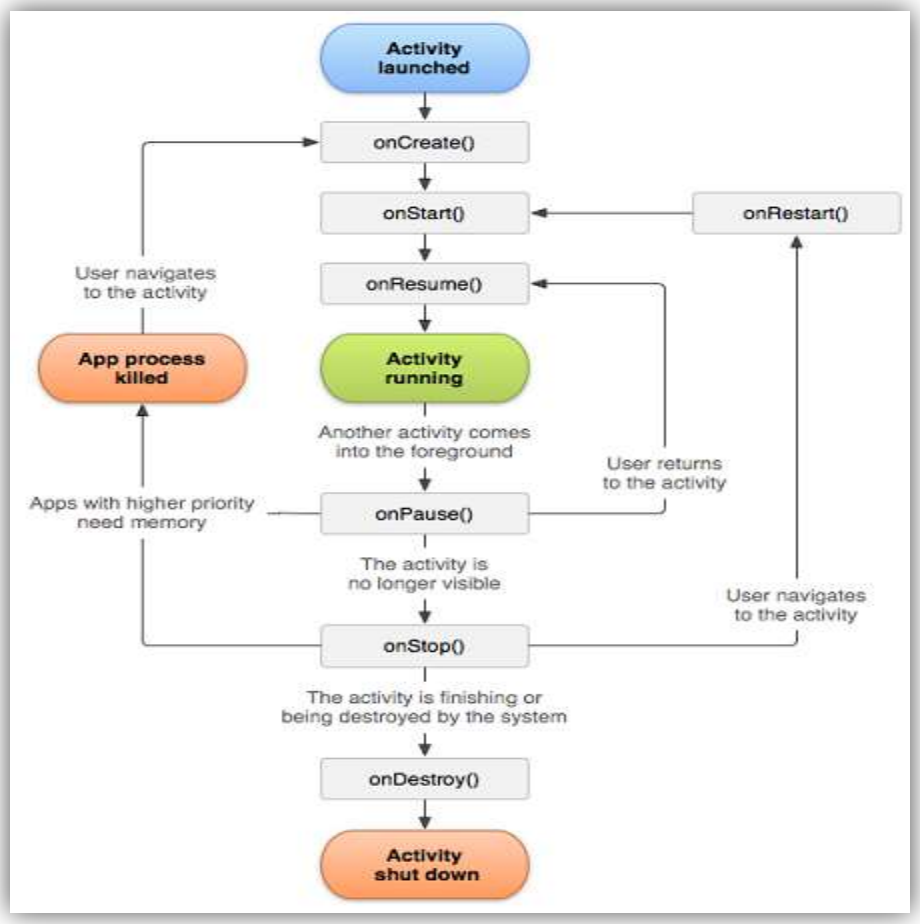
Μετά τους διάφορους τύπους των εφαρμογών στο Android, θα ήταν σκόπιμο να αναλυθούν τα κυριότερα στοιχεία που χρησιμοποιούνται σε μια Android εφαρμογή καθώς και η διατύπωση των όρων.

4.3.1 Context

Πρόκειται για μια διεπαφή (interface) που παρέχει γενικές πληροφορίες σχετικά με το περιβάλλον της εφαρμογής. Είναι μια αφηρημένη κλάση, η εφαρμογή της οποίας παρέχεται από το Android. Επιτρέπει την πρόσβαση σε συγκεκριμένους πόρους και κλάσεις της εφαρμογής, όπως επίσης και κλήσεις ανωτέρου επιπέδου (επιπέδου εφαρμογής), όπως άνοιγμα *activities*, εκπομπή και λήψη *intents* κτλ (Οι δύο αυτοί όροι θα αναλυθούν εκτενέστερα παρακάτω). Κάθε εφαρμογή διατηρεί τις κλάσεις και τους πόρους της σε ένα συγκεκριμένο περιβάλλον, οπότε χρησιμοποιώντας το Context μπορούμε να έχουμε πρόσβαση σε αυτά.

4.3.2 Activity

Μια δραστηριότητα (activity), είναι ένα συστατικό της εφαρμογής που παρέχει μια οθόνη με την οποία οι χρήστες μπορούν να αλληλεπιδρούν με σκοπό να κάνει κάτι, όπως να καλέσετε κάποιο τηλέφωνο, να τραβήξετε μια φωτογραφία, να στείλετε ένα email ή να δείτε ένα χάρτη. Μια εφαρμογή αποτελείται συνήθως από πολλαπλές δραστηριότητες που είναι συνδεδεμένες η μια με την άλλη. Συνήθως, μια δραστηριότητα σε μια εφαρμογή έχει οριστεί ως η «κύρια» δραστηριότητα (Main Activity), η οποία παρουσιάζεται στο χρήστη κατά την εκκίνηση της εφαρμογής. Κάθε δραστηριότητα μπορεί στη συνέχεια να αρχίσει μια άλλη δραστηριότητα, προκειμένου να εκτελέσει διάφορες ενέργειες. Κάθε φορά που μια νέα δραστηριότητα ξεκινά, η προηγούμενη δραστηριότητα έχει σταματήσει, αλλά το σύστημα διατηρεί τη δραστηριότητα σε μια στοίβα.



Εικόνα 6: Κύκλος ζωής μιας δραστηριότητας (activity). [8]

Παραπάνω στην εικόνα παρουσιάζονται οι μέθοδοι που καλούνται κατά την διάρκεια ζωής μιας δραστηριότητας τα οποία και θα αναλύσουμε.

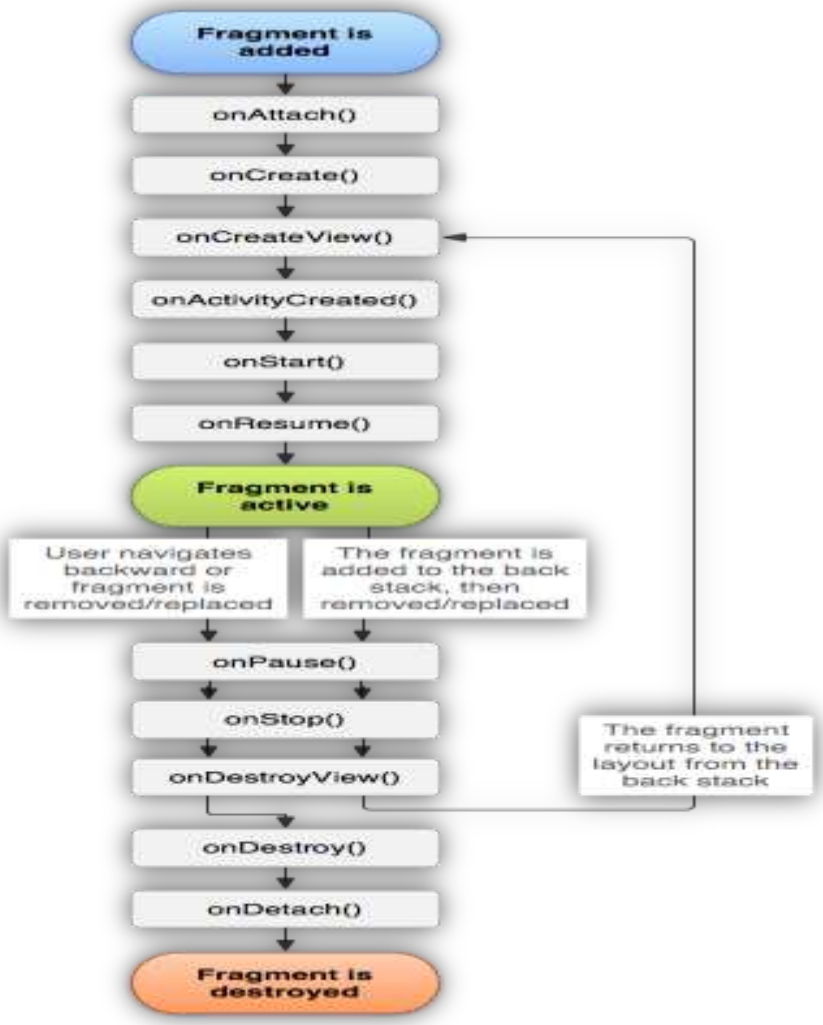
- Η μέθοδος onCreate() καλείται όταν η δραστηριότητα δημιουργείται για πρώτη φορά. Εδώ κάνουμε όλες τις αρχικοποιήσεις και πάντα ακολουθεί η μέθοδος OnStart().
- Η μέθοδος onRestart() καλείται μετά που η δραστηριότητα έχει σταματήσει, ακριβώς πριν ξεκινήσει ξανά και πάντα ακολουθεί η μέθοδος OnStart().
- Η μέθοδος OnStart() καλείται ακριβώς πριν η δραστηριότητα εμφανιστεί στον χρήστη. Ακολουθεί η μέθοδος onResume() αν η δραστηριότητα γίνεται στο προσκήνιο, ή η μέθοδος onStop() αν γίνεται κρυφά.

- Η μέθοδος onResume() καλείται πριν αρχίσει η αλληλεπίδραση της δραστηριότητας με τον χρήστη και πάντα ακολουθεί η onPause().
- Η μέθοδος onPause() καλείται όταν το σύστημα είναι έτοιμο να ξεκινήσει την επανάληψη μιας άλλης δραστηριότητας. Σταματάει οτιδήποτε περιττό καταναλώνει πόρους από την CPU και αν η δραστηριότητα επιστρέφει στο προσκήνιο ακολουθεί η μέθοδος onResume(), αλλιώς αν παραμείνει αόρατη προς τον χρήστη καλείται η μέθοδος onStop().
- Η μέθοδος onStop() καλείται όταν η δραστηριότητα δεν είναι πλέον ορατή στον χρήστη, αυτό μπορεί να συμβεί είτε διότι καταστρέφεται είτε επειδή μια άλλη δραστηριότητα έχει επαναληφθεί και την καλύπτει. Ακολουθεί η μέθοδος onRestart() αν η δραστηριότητα επιστρέφει για να αλληλοεπιδράσει με το χρήστη ή ακολουθεί η μέθοδος onDestroy() αν πρόκειται να τερματιστεί.

4.3.3 Fragment

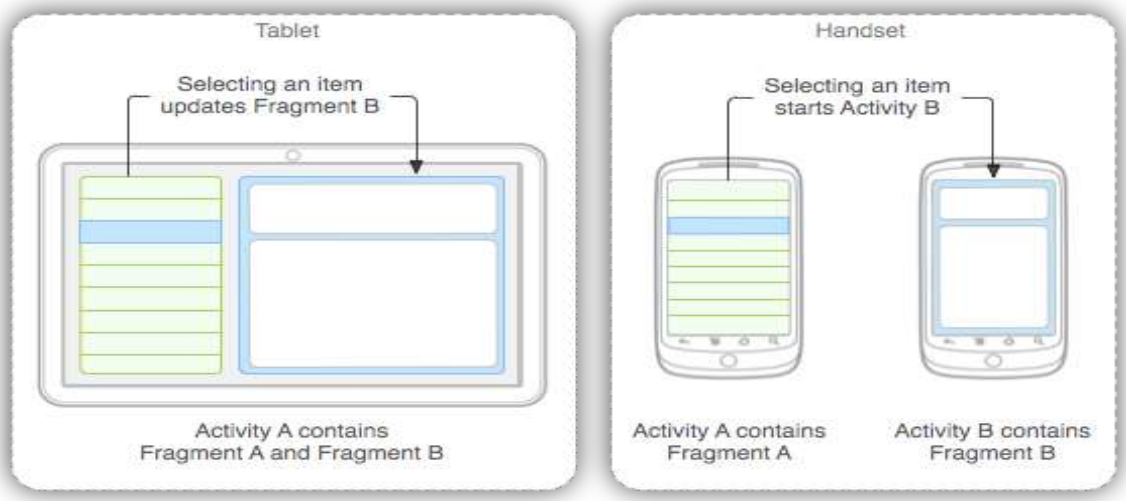
Τα Fragment παρόλο που είναι ένα εξειδικευμένο στοιχείο στην ανάπτυξη εφαρμογών Android, είναι αναγκαίο να αναφερθούν στην παρούσα πτυχιακή εργασία, διότι έχουν χρησιμοποιηθεί στην εφαρμογή που υλοποιήσαμε. Τα Fragment είναι διαθέσιμα από την έκδοση Android 3.x.x και έπειτα, και χρησιμοποιούνται ευρύτατα από τους προγραμματιστές.

Ένα Fragment πρέπει πάντα να είναι ενσωματωμένο μέσα σε μια δραστηριότητα και ο κύκλος ζωής του επηρεάζεται άμεσα από τον κύκλο ζωής αυτής της δραστηριότητας. Παρακάτω φαίνεται ο κύκλος ζωής ενός Fragment.



Εικόνα 7: Κύκλος ζωής ενός fragment. [9]

Το Android εισήγαγε τα fragments για να υποστηρίξει πιο δυναμικά και ευέλικτα τη διεπαφή χρήστη (UI - User Interface) κυρίως σε μεγάλες οθόνες. Τα fragments επιτρέπουν το σχεδιασμό χωρίς να μπει ο προγραμματιστής σε περίπλοκες αλλαγές. Για να γίνει κατανοητό, μια εφαρμογή μπορεί να χρησιμοποιήσει ένα fragment για να παρουσιάσει μια λίστα με άρθρα ή επιλογές στα αριστερά και ένα άλλο Fragment για να δείξει το άρθρο ή την επιλογή στα δεξιά. Και τα δύο αυτά fragments είναι εμφανή μέσα στην ίδια δραστηριότητα και το καθένα έχει το δικό του κύκλο ζωής. Παρακάτω παρουσιάζεται μια εικόνα που δείχνει πώς δύο γραφικά στοιχεία που ορίζονται από fragments μπορούν να συνδυαστούν και να εμφανιστούν σε μια δραστηριότητα σχεδιασμένη για tablet και πως μπορούν να χωριστούν όταν σχεδιάζονται για κινητό.



Εικόνα 8: Ένα παράδειγμα που δείχνει πως δύο γραφικά στοιχεία που ορίζονται από fragments μπορούν να συνδυαστούν σε ένα activity για σχεδιασμό σε tablet και να χωριστούν για σχεδιασμό σε κινητό. [9]

4.3.4 Manifest

Όλες οι Android εφαρμογές είναι υποχρεωτικό να έχουν ένα αρχείο Android Manifest (AndroidManifest.xml). Το συγκεκριμένο XML αρχείο παρέχει ουσιώδεις πληροφορίες για την εφαρμογή μας στο Android σύστημα που πρέπει να γνωρίζει πριν τρέξει την εφαρμογή. Το Android Manifest περιλαμβάνει τα ονόματα των Java πακέτων της εφαρμογής, περιγράφει τα εξαρτήματα της εφαρμογής όπως είναι τα Activities, τα Services, Broadcast Receiver και άλλα, δηλώνει τις άδειες που απαιτούνται από την εφαρμογή μας, τις βιβλιοθήκες που χρησιμοποιούνται και τέλος δηλώνει το ελάχιστο επίπεδο του Android API το οποίο απαιτεί η εφαρμογή.

4.3.5 Intent

Το Android χρησιμοποιεί ένα σύγχρονο μηχανισμό αποστολής/παραλαβής μηνυμάτων ώστε να ανταπεξέλθει σε αιτήσεις εργασιών με το κατάλληλο Activity. Κάθε αίτηση συσκευάζεται ως ένα Intent. Αυτό σημαίνει πως κάθε αίτηση δηλώνει μια πρόθεση να γίνει κάτι, για παράδειγμα εάν από ένα Activity θέλουμε να ξεκινήσουμε κάποιο άλλο, αυτό προγραμματιστικά θα γίνει με την χρήση Intent.

4.3.6 Service

Ένα Service μπορεί να εκτελέσει μακροχρόνιες λειτουργίες στο παρασκήνιο χωρίς να παρέχει τη δυνατότητα για αλληλεπίδραση με τη διεπαφή χρήστη (UI). Συστατικά μέρη μιας εφαρμογής μπορούν να ξεκινήσουν ένα service και αυτό να εξακολουθεί να εκτελείται ακόμη και αν ο χρήστης εγκαταλείψει τη συγκεκριμένη εφαρμογή (που ξεκίνησε το service). Ένα service για παράδειγμα μπορεί να παίζει μουσική ή να εκτελεί διαδικτυακές συναλλαγές.

4.3.7 Broadcast

Το Android μπορεί να στέλνει ή να δέχεται μηνύματα από το λειτουργικό σύστημα ή από εφαρμογές. Αυτά τα μηνύματα στέλνονται όταν προκύψουν συγκεκριμένα γεγονότα. Για παράδειγμα το λειτουργικό σύστημα στέλνει ένα τέτοιο μήνυμα όταν η συσκευή ξεκινά να φορτίζεται. Οι εφαρμογές μπορούν να στείλουν δικά τους τέτοια μηνύματα για να ενημερώσουν για παράδειγμα άλλες εφαρμογές που ενδιαφέρονται γι’ αυτά.

4.3.8 Layouts

Ένα layout ορίζει την οπτική δομή για μια διεπαφή χρήστη (UI) σε μια εφαρμογή. Ένα layout μπορεί να οριστεί με δύο τρόπους, είτε μέσω XML αρχείων είτε προγραμματιστικά (κατά τη δημιουργία ενός Activity). Το πλεονέκτημα να οριστούν τα layout από αρχεία XML είναι ότι κερδίζουμε την αποσύνδεση του layout από τον κώδικα της εφαρμογής. Παρακάτω αναφέρονται μερικά από τα βασικότερα σχεδιαστικά πρότυπα που χρησιμοποιούνται στην ανάπτυξη μιας Android εφαρμογής.

4.3.8.1 LinearLayout

Μια προβολή LinearLayout στοιχίζει όλα τα στοιχεία της προς μια κατεύθυνση,

οριζόντια ή κάθετα. Για παράδειγμα η εικόνα 9 παρουσιάζει ένα LinearLayout που έχει στοιχισμένα τα στοιχεία του οριζόντια το ένα μετά το άλλο.

Εικόνα 9: Παράδειγμα διάταξης LinearLayout

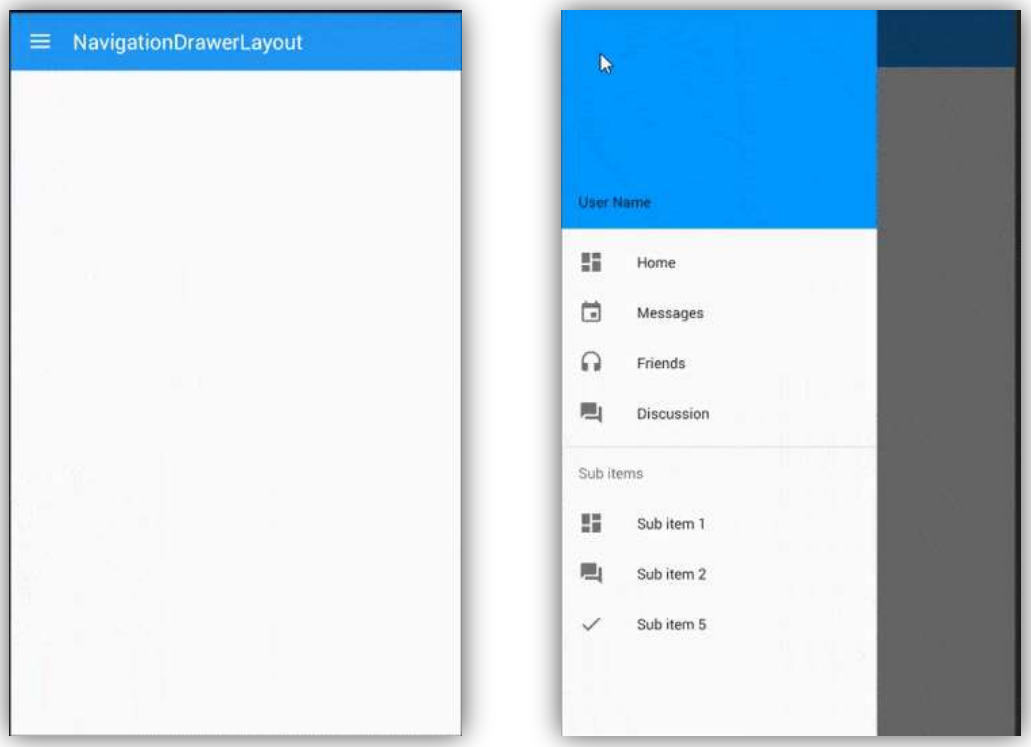
4.3.8.2 RelativeLayout

Μια προβολή RelativeLayout στοιχίζει τα στοιχεία της σε σχετικές θέσεις. Η θέση του κάθε στοιχείου μπορεί να προσδιοριστεί σε σχέση με οποιοδήποτε άλλο στοιχείο της προβολής. Για παράδειγμα "στα αριστερά" (left-of) ή "κάτω" (below) από κάποιο συγκεκριμένο στοιχείο. Επίσης οι θέσεις των στοιχείων μπορούν να σχετίζονται και με όλη τη προβολή, για παράδειγμα ευθυγράμμιση προς τα κάτω (aligned to the bottom). Στη εικόνα 10 που ακολουθεί φαίνεται ένα παράδειγμα RelativeLayout.

Εικόνα 10: Παράδειγμα διάταξης RelativeLayout

4.3.8.3 DrawerLayout

Το DrawerLayout επιτρέπει την διαδραστική προβολή ενός «συρταριού» από μία ή και τις δύο κάθετες άκρες του παραθύρου. Η τοποθέτηση και η διάταξή του ελέγχεται χρησιμοποιώντας ένα συγκεκριμένο χαρακτηριστικό (layout_gravity) που μεταβάλλεται ανάλογα από την πλευρά του παραθύρου που επιθυμούμε να βγει το «συρτάρι»: αριστερά ή δεξιά. Ουσιαστικά πρόκειται για ένα μενού πλοήγησης που εμφανίζει τις κύριες επιλογές στην άκρη της οθόνης. Είναι κρυμμένο από προεπιλογή, και εμφανίζεται είτε όταν ο χρήστης σύρει απαλά το δάχτυλό του από την άκρη της οθόνης προς το κέντρο, είτε όταν πατήσει το εικονίδιο της εφαρμογής στο πάνω μέρος της οθόνης (toolbar/action bar). Παρακάτω παρατίθεται ένα παράδειγμα ενός DrawerLayout.



Εικόνα 11: Παράδειγμα κλειστού / ανοικτού DrawerLayout

4.3.9 Dialogs

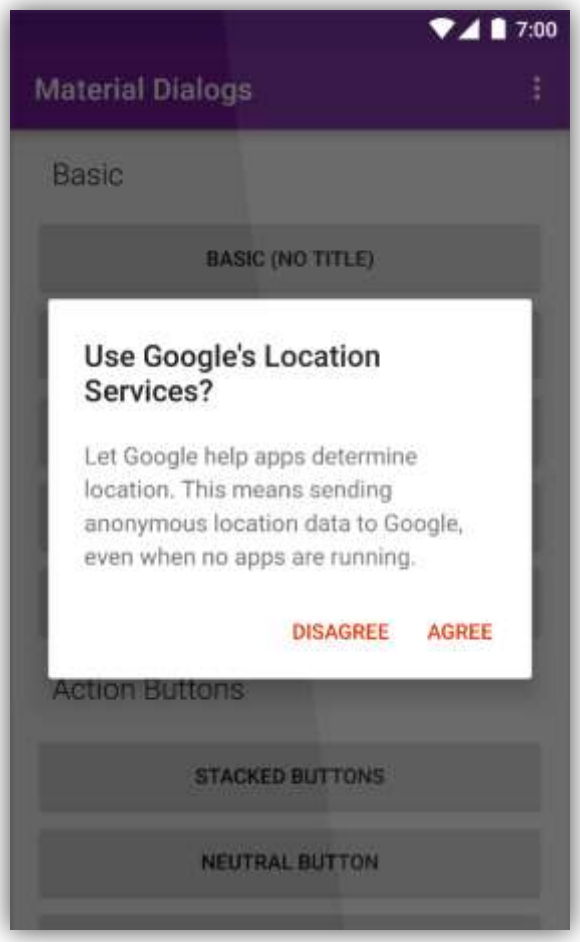
Ένα Dialog (διάλογος) είναι ένα μικρό παράθυρο το οποίο προτρέπει τον χρήστη να πάρει μια απόφαση ή να εισάγει επιπρόσθετες πληροφορίες. Ένα Dialog δεν γεμίζει όλη την οθόνη και συνήθως χρησιμοποιείται όταν ο χρήστης πρέπει να κάνει κάποια ενέργεια για να μπορέσει να συνεχίσει. Τα πιο βασικά είδη διαλόγου είναι:

- **AlertDialog:** ένα dialog που περιέχει έναν τίτλο, μέχρι τρία κουμπιά, μια λίστα με κάποια επιλεγμένα στοιχεία ή μια τελείως προσαρμοσμένη διάταξη (custom layout).

- **DatePickerDialog:** Ένα dialog με μια προκαθορισμένη διεπαφή χρήστη που επιτρέπει στον χρήστη να επιλέξει ημερομηνία ή ώρα.

- **ProgressDialog:** Ένα Progress Dialog χρησιμοποιείται για να δείξει στον χρήστη την πρόοδο μιας διαδικασίας. Όταν εκτελούνται χρονοβόρες διαδικασίες για να μη βλέπει ο χρήστης μια παγωμένη οθόνη, χρησιμοποιείται το Progress Dialog που ενημερώνει τον χρήστη για την πρόοδο της διαδικασίας.

Στην εικόνα 12 που ακολουθεί παρουσιάζεται ένα AlertDialog.



Εικόνα 12: Παράδειγμα Alert Dialog

4.3.10 Toolbar

Το toolbar είναι ένα στοιχείο που προσδιορίζει τη θέση του χρήστη και περιέχει διαθέσιμες ενέργειες και τρόπους πλοήγησης. Χρησιμοποιώντας ένα toolbar, προσφέρεται στους χρήστες ένα οικείο περιβάλλον στην χρήση της εφαρμογής, που προσαρμόζεται χάρη στις διαφορετικές διαμορφώσεις οθόνης. Εκτελεί σημαντικές ενέργειες, οι οποίες είναι εμφανέστατες και προσβάσιμες με έναν προβλέψιμο τρόπο. Στην εικόνα 13 που ακολουθεί παρουσιάζεται ένα παράδειγμα ενός toolbar.

Εικόνα 13: Παράδειγμα Toolbar

4.4 Επίλογος

Συνοψίζοντας, έγινε μια προσπάθεια ο αναγνώστης να κατανοήσει τα βασικά συστατικά στοιχεία μιας Android εφαρμογής τα οποία αποτελούν το θεμέλιο για να ξεκινήσει κάποιος να προγραμματίζει στο Android.
Παρακάτω θα δούμε την αρχιτεκτονική και τις τεχνολογίες που χρησιμοποιήθηκαν για τη εφαρμογή της πτυχιακής εργασίας όπως και την αναλυτική περιγραφή της.

5. Τεχνολογίες που χρησιμοποιήθηκαν

5.1 Εισαγωγή

Στο παρόν κεφάλαιο θα γίνει λεπτομερής περιγραφή της δομής των φακέλων μαζί με τα αρχεία που περιέχουν καθώς και των τεχνολογιών (γλώσσες προγραμματισμού, βιβλιοθήκες λογισμικού, προγραμματιστικά περιβάλλοντα) που χρησιμοποιήθηκαν για να υλοποιηθεί η Android εφαρμογή.

5.2 Google Cloud Messaging – GCM

Το Google Cloud Messaging είναι μια υπηρεσία ειδοποιήσεων (notifications) για κινητά τηλέφωνα που αναπτύχθηκε από την Google. Επιτρέπει σε προγραμματιστές που αναπτύσσουν εφαρμογές για τρίτους (third-party applications) να στέλνουν δεδομένα μέσω ειδοποιήσεων σε εφαρμογές που «τρέχουν» λειτουργικό Android, καθώς και για επεκτάσεις που έχουν αναπτυχθεί για τον περιηγητή Google Chrome. Η παραπάνω υπηρεσία πρόσφατα αντικαταστάθηκε από τη Firebase Cloud Messaging (FCM) της Google και διατίθεται δωρεάν στους προγραμματιστές [11]. Παρακάτω περιγράφονται τα βήματα που πρέπει να ακολουθήσουμε για να χρησιμοποιήσουμε την υπηρεσία στην παρούσα εφαρμογή.

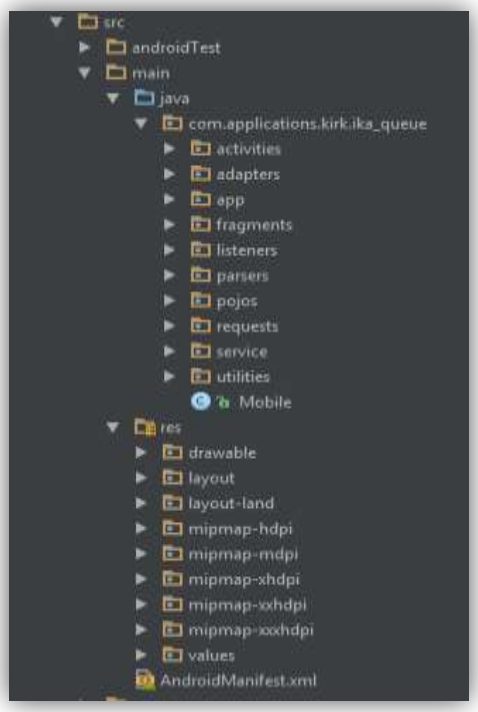
5.2.1 Προσθήκη του GCM στην εφαρμογή

Υπάρχουν 2 τρόποι για να προσθέσουμε την λειτουργία GCM στην εφαρμογής μας. Είτε χειροκίνητα, προσθέτοντας μόνοι μας τις κατάλληλες εξαρτήσεις (dependencies) στο project μας, είτε (εάν διαθέτουμε την τελευταία έκδοση του λογισμικού Android Studio) με τη χρήση ενός βοηθού (Firebase Assistant), ούτως ώστε να συνδεθεί η εφαρμογή με το Firebase. Στην περίπτωση μας, χρησιμοποιήσαμε τον βοηθό και προσθέτοντας απλά τα στοιχεία του Google λογαριασμού μας, προστέθηκε αυτόματα ο απαραίτητος κώδικας για να χρησιμοποιήσουμε το Firebase.

Τέλος, αφού έχουμε εγγραφεί πλέον στο Firebase Cloud Messaging, επισκεπτόμαστε την ιστοσελίδα <https://console.firebase.google.com>, επιλέγουμε το project στο οποίο μόλις το προσθέσαμε και αντιγράφουμε το API Key που είναι μοναδικό για κάθε εφαρμογή και που θα χρειαστεί για να χρησιμοποιηθεί στο back-end του συστήματος μας, για να μπορέσουμε να στείλουμε επιτυχώς ειδοποιήσεις στους χρήστες της εφαρμογής.

5.3 Δομή του project στο Android Studio

Η δομή της εφαρμογής αποτελείται από φακέλους και διάφορα αρχεία σε μορφή δέντρου, όπως σε κάθε Android εφαρμογή. Η ταξινόμηση των αρχείων στους φακέλους έγινε, κατά το δυνατόν, με βάση τα χαρακτηριστικά τους.
Η συγκεκριμένη εφαρμογή αναπτύχθηκε στο περιβάλλον Android Studio και η δομή της απεικονίζεται παρακάτω:



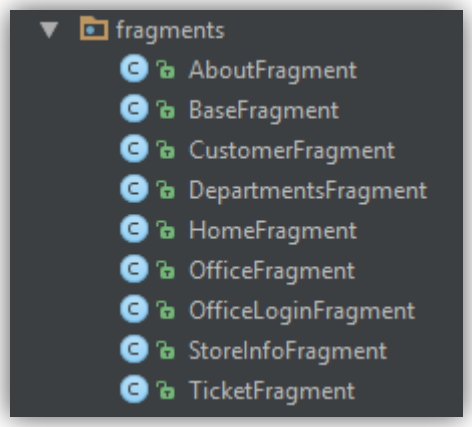
Εικόνα 14: Δομή της εφαρμογής στο Android Studio

Συγκεκριμένα, βλέπουμε τους εξής φακέλους (πακέτα):

- ***/src:***

  Ριζικός φάκελος που περιλαμβάνει όλους τους υποφακέλους του project.
- ***/src/main/java:***

  Περιέχει τους φακέλους με τις κλάσεις της εφαρμογής. Ο πρώτος φάκελος με όνομα "*com.applications.kirk.ika_queue*" περιλαμβάνει όλη τη λειτουργικότητα της εφαρμογής. Αποτελείται από επιμέρους φακέλους, που ο καθένας αφορά και μια συγκεκριμένη λειτουργία. Για παράδειγμα, αν επεκτείνουμε τον φάκελο "*fragments*", θα δούμε όλα τα fragments που δημιουργήθηκαν για τη συγκεκριμένη εφαρμογή.



Εικόνα 15: Λίστα των fragments που χρησιμοποιεί η εφαρμογή

- ***/src/res***

  Στον συγκεκριμένο φάκελο αποθηκεύονται όλοι οι πόροι (resources) της εφαρμογής.
- ***/src/res/drawable***

  Σε αυτό το φάκελο αποθηκεύονται όλα τα γραφικά της εφαρμογής. Με τον όρο γραφικά, αναφερόμαστε σε κάθε τύπου εικόνας αλλά και σε αρχεία XML που αφορούν τα γραφικά συστατικά της εφαρμογής.
- ***src/res/layout***

  Σε αυτόν τον φάκελο αποθηκεύονται αρχεία XML τα οποία σχετίζονται με το γραφικό περιβάλλον της εφαρμογής.

- ***src/res/values***

  Στον φάκελο values αποθηκεύονται κατάλληλα αρχεία που περιέχουν ειδικές μεταβλητές που χρησιμοποιεί η εφαρμογή, όπως τιμές χρωμάτων και αλφαριθμητικών με συγκεκριμένα ονόματα ή αρχεία με σύνολα κανόνων (styles) για την εμφάνιση συγκεκριμένων γραφικών στοιχείων.

5.4 Τεχνολογίες

Σε αυτό το σημείο, θα αναπτυχθούν αναλυτικά οι γλώσσες προγραμματισμού, τα προγραμματιστικά περιβάλλοντα, καθώς και διάφορες τεχνολογίες που χρησιμοποιήθηκαν για τη δημιουργία της εφαρμογής.

5.4.1 Java

Η Java είναι μια αντικειμενοστραφής γλώσσα προγραμματισμού που σχεδιάστηκε από την εταιρεία πληροφορικής Sun Microsystems. Σκοπός της ήταν να επιτρέψει στους προγραμματιστές να γράφουν μια φορά το πρόγραμμα και να το τρέξουν οπουδήποτε (υπήρχε το μότο: “Write once, run anywhere - WORA), πράγμα που σημαίνει ότι ο μεταγλωττισμένος κώδικας Java μπορεί να τρέξει σε οποιαδήποτε πλατφόρμα (Windows, Linux, Unix και MacOS) που υποστηρίζει Java, χωρίς να χρειάζεται να επαναμεταγλωττιστεί (compiling) ή να αλλάξει ο πηγαίος κώδικας για κάθε διαφορετικό λειτουργικό σύστημα.

Για να επιτευχθεί όμως αυτό, χρειαζόταν κάποιος τρόπος έτσι ώστε τα προγράμματα γραμμένα σε Java να μπορούν να είναι «κατανοητά» από κάθε υπολογιστή ανεξάρτητα του είδους επεξεργαστή (Intel x86, IBM, Sun SPARC, Motorola) αλλά και λειτουργικού συστήματος. Ο λόγος είναι ότι οι εφαρμογές γραμμένες σε Java, τυπικά μεταγλωττίζονται σε γλώσσα μηχανής, η οποία είναι διαφορετική σε κάθε λειτουργικό σύστημα. Ο συμβολικός κώδικας (assembly) που μεταφράζεται και εκτελείται σε Windows είναι διαφορετικός από αυτόν που μεταφράζεται και εκτελείται σε έναν υπολογιστή Macintosh.

Η λύση δόθηκε με την ανάπτυξη της Εικονικής Μηχανής (Java Virtual Machine ή JVM). Οι εφαρμογές γραμμένες σε Java, μεταγλωττίζονται μέσω του μεταγλωττιστή javac, ο οποίος παράγει έναν αριθμό από αρχεία .class (κώδικας byte ή bytecode).

Όταν πρόκειται να εκτελεστεί η εφαρμογή σε ένα μηχάνημα, το Java Virtual Machine, που θα πρέπει να είναι εγκατεστημένο σε αυτό, θα αναλάβει να διαβάσει τα αρχεία .class. Στη συνέχεια, τα μεταφράζει σε γλώσσα μηχανής που να υποστηρίζεται από το λειτουργικό σύστημα και τον επεξεργαστή, έτσι ώστε να εκτελεστεί (έτσι λειτουργεί η «παραδοσιακή» Εικονική Μηχανή. Σε πιο σύγχρονες εκδόσεις, μπορεί και μεταγλωττίζει εκ των προτέρων τμήματα bytecode απευθείας σε κώδικα μηχανής (native code) με αποτέλεσμα να βελτιώνεται η ταχύτητα). Χωρίς αυτό δε θα ήταν δυνατή η εκτέλεση λογισμικού γραμμένου σε Java.

Τέλος, για να γράψει κάποιος κώδικα Java δε χρειάζεται τίποτα άλλο παρά έναν απλό επεξεργαστή κειμένου. Παρ' όλ' αυτά, ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) είναι ιδιαίτερα βοηθητικό, ειδικά στον εντοπισμό σφαλμάτων (debugging). Υπάρχουν αρκετά διαθέσιμα, ενώ πολλά από αυτά είναι δωρεάν.

5.4.2 XML

Η XML (eXtensible Markup Language) είναι μία γλώσσα σήμανσης, που περιέχει ένα σύνολο κανόνων για την ηλεκτρονική κωδικοποίηση κειμένων, σε μορφή που είναι ταυτόχρονα αναγνώσιμη τόσο από τον άνθρωπο, όσο κι από μηχανή. Ορίζεται, κυρίως, στην προδιαγραφή XML 1.0 (XML 1.0 Specification), που δημιούργησε ο διεθνής οργανισμός προτύπων W3C (World Wide Web Consortium), αλλά και σε διάφορες άλλες σχετικές προδιαγραφές ανοιχτών προτύπων.

Η XML σχεδιάστηκε δίνοντας έμφαση στην απλότητα, τη γενικότητα και τη χρηστικότητα στο Διαδίκτυο. Είναι μία μορφοποίηση δεδομένων κειμένου, με ισχυρή υποστήριξη Unicode για όλες τις γλώσσες του κόσμου. Αν και ο σχεδιασμός της XML επικεντρώνεται στα κείμενα, χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων, όπως για παράδειγμα αυτών που προκύπτουν στις υπηρεσίες ιστού. Υπάρχει μία ποικιλία διεπαφών προγραμματισμού εφαρμογών, που μπορούν να χρησιμοποιούν οι προγραμματιστές, για να προσπελαύνουν δεδομένα XML, αλλά και διάφορα συστήματα σχημάτων XML, τα οποία είναι σχεδιασμένα για να βοηθούν στον ορισμό γλωσσών, που προκύπτουν από την XML.

5.4.2.1 Βασική ορολογία XML

Τα βασικά στοιχεία που θα συναντήσει κάποιος στην χρήση της XML είναι τα παρακάτω:

- **Χαρακτήρας Unicode**

  Εξ ορισμού, ένα κείμενο XML είναι μία ακολουθία χαρακτήρων. Σχεδόν κάθε χαρακτήρας Unicode μπορεί να εμφανίζεται σε ένα κείμενο XML.

- **Επεξεργαστής και εφαρμογή**

  Είναι το λογισμικό που επεξεργάζεται ένα κείμενο XML. Ο επεξεργαστής, ουσιαστικά, αναλύει τη σήμανση και μεταφέρει δομημένες πληροφορίες σε μια εφαρμογή. Υπάρχουν μερικές πολύ συγκεκριμένες απαιτήσεις, σχετικά με το τι μπορεί και τι δεν μπορεί να κάνει ένας επεξεργαστής XML, αλλά καμία, όσον αφορά στη συμπεριφορά της εφαρμογής. Ο επεξεργαστής (όπως ονοματίζεται από την προδιαγραφή), αναφέρεται συχνά, με τον αγγλικό όρο XML parser.

- **Σήμανση και περιεχόμενο**

  Οι χαρακτήρες που συνθέτουν ένα έγγραφο XML αποτελούν είτε τη σήμανση είτε το περιεχόμενό του, τα οποία μπορούν να διακριθούν και να επισημανθούν με την εφαρμογή κάποιων απλών συντακτικών κανόνων. Γενικότερα, οι συμβολοσειρές που αποτελούν τη σήμανση είτε ξεκινούν με το χαρακτήρα **“<”** και καταλήγουν με το χαρακτήρα **“>”**, είτε αρχίζουν με τον χαρακτήρα **“&”** και τελειώνουν με τον χαρακτήρα **“;”**. Οι ακολουθίες χαρακτήρων που δε συνιστούν τη σήμανση, αποτελούν το περιεχόμενο ενός κειμένου XML.

- **Ετικέτα**

  Ένα στοιχείο σήμανσης που ξεκινά με το χαρακτήρα **“<”** και καταλήγει στο χαρακτήρα **“>”**. Υπάρχουν τρία είδη ετικέτας: *ετικέτες-αρχής (start-tag)*, για παράδειγμα <section>, *ετικέτες-τέλους (end-tag)*, για παράδειγμα </section>, και *ετικέτες-χωρίς-περιεχόμενο (empty-element tag)*, για παράδειγμα <line-break/>.

- **Στοιχείο**

  Είναι ένα λογικό απόσπασμα ενός κειμένου, που είτε ξεκινά με μία ετικέτα-αρχής και καταλήγει σε μία ετικέτα-τέλους, είτε αποτελείται μόνο από μία ετικέτα-χωρίς-περιεχόμενο. Οι χαρακτήρες που υπάρχουν, αν υπάρχουν, μεταξύ μιας ετικέτας-αρχής και μιας ετικέτας-τέλους, αποτελούν το περιεχόμενο του στοιχείου, το οποίο μπορεί να περιέχει σήμανση, συμπεριλαμβανομένων και άλλων στοιχείων, που ονομάζονται στοιχεία-παιδιά. Ένα παράδειγμα ενός στοιχείου είναι το *<Greeting>Hello, world.</Greeting>*.

- **Χαρακτηριστικό**

  Είναι ένα στοιχείο σήμανσης που αποτελείται από ένα ζευγάρι όνομα/τιμή, το οποίο υπάρχει μέσα σε μία ετικέτα-αρχής ή σε μία ετικέτα-χωρίς-περιεχόμενο. Στο παρακάτω παράδειγμα, το στοιχείο img έχει δύο χαρακτηριστικά, τα src και alt: *< img src=”myImage.jpg” alt=’by Chris’ />*. Ένα άλλο παράδειγμα θα ήταν το *<step number=”3”>Connect A to B.</step>*, όπου το όνομα του χαρακτηριστικού είναι “number” και η τιμή του είναι “3”.

- **Δήλωση XML**

  Τα κείμενα XML μπορούν να αρχίζουν, με τη δήλωση κάποιων πληροφοριών σχετικών με αυτά, όπως στο ακόλουθο παράδειγμα:
  <?xml version="1.0" encoding="UTF-8"?>

Παράδειγμα:

```
<?xml version="1.0" encoding="UTF-8"?>
<note>
  <to>Tom</to>
  <from>Jennifer</from>
  <heading>Reminder</heading>
  <body>Don't forget me this weekend!</body>
</note>
```

5.4.3 Βοηθητική βιβλιοθήκη για ασύγχρονη φόρτωση εικόνων

Σε ένα συγκεκριμένο σημείο της παρούσας εφαρμογής, χρειάστηκε η φόρτωση του χάρτη του εκάστοτε καταστήματος ΙΚΑ (θα αναλυθεί περαιτέρω παρακάτω). Σε περιπτώσεις όπου ο προγραμματιστής θέλει απλώς να φορτώσει μία εικόνα σε ένα συγκεκριμένο activity/fragment και για οποιονδήποτε λόγο, δεν επιθυμεί να είναι αποθηκευμένη στο project, χρειάζεται μια βιβλιοθήκη όπως την συγκεκριμένη, με το όνομα Ion. Η βοηθητική αυτή βιβλιοθήκη έχει πολλά χαρακτηριστικά, όπως:

- Ασύγχρονο download:
    - Εικόνων (για προσθήκη σε ImageViews και Bitmaps)
    - JSON
    - Αλφαριθμητικών (Strings)
    - Αρχείων
- Εύκολο στη χρήση API, σχεδιασμένο ειδικά για Android
    - Ακυρώνει αυτόματα οποιαδήποτε λειτουργία, όταν ο χρήστης επιστρέφει στο προηγούμενο Activity.
    - Όλες οι λειτουργίες της βιβλιοθήκης επιστρέφουν κάτι, κι έτσι μπορούν να ακυρωθούν.
- Υποστήριξη HTTP POST/PUT.

Στη συγκεκριμένη εφαρμογή, χρησιμοποιήθηκε μόνο το πρώτο χαρακτηριστικό, αυτό της ασύγχρονης φόρτωσης εικόνων και της προσθήκης τους σε ένα ImageView. Ο κώδικας που χρειάζεται για αυτή τη λειτουργία είναι εξαιρετικά απλός:

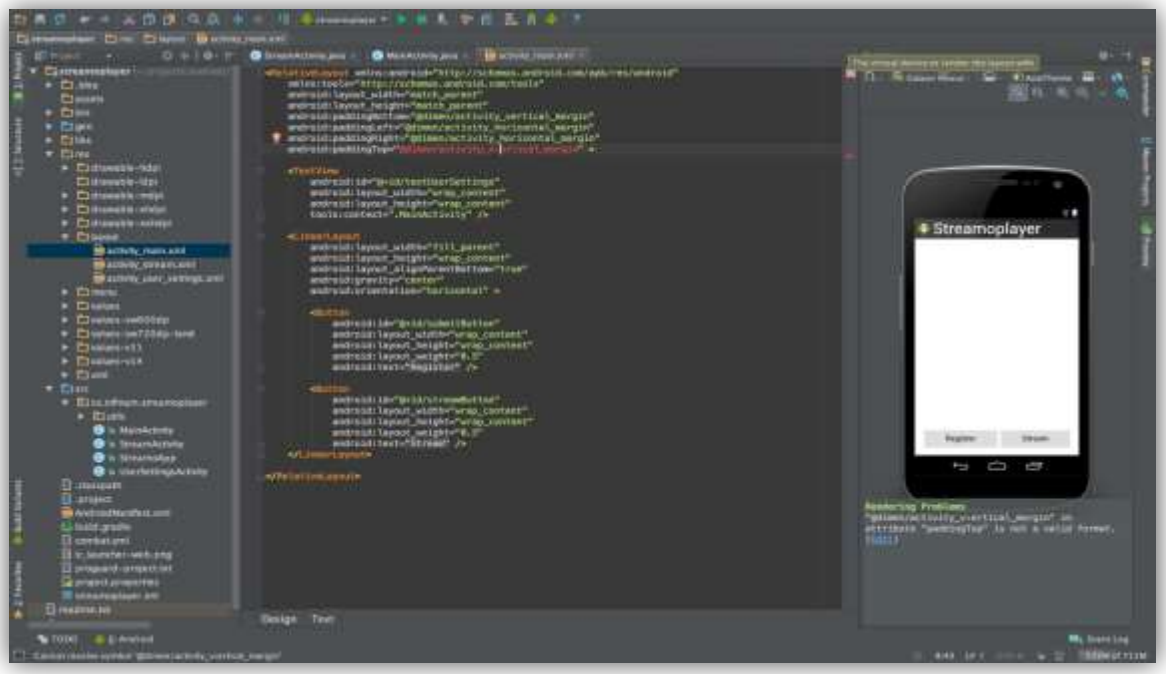
```
Ion.with(imageView)
 .placeholder(R.drawable.placeholder_image)
 .error(R.drawable.error_image)
 .animateLoad(spinAnimation)
 .animateIn(fadeInAnimation)
 .load("http://example.com/image.png");
```

Η βιβλιοθήκη Ion είναι ουσιαστικά ένα GitHub project, το οποίο είναι δωρεάν προς όλους. [10]

5.4.4 Android Studio

Το Android Studio είναι ένα ολοκληρωμένο προγραμματιστικό περιβάλλον (Integrated Development Environment - IDE) για ανάπτυξη εφαρμογών στην πλατφόρμα Android, βασισμένο στο Intellij IDEA. Διατίθεται από τη Google [12] και είναι το IDE το οποίο προτείνεται και επίσημα από την ίδια για την ανάπτυξη εφαρμογών Android. Προσφέρει χαρακτηριστικά όπως:

- Υποστήριξη Gradle
- Επεξεργαστή κειμένου με πλούσιες δυνατότητες
- Ενσωματωμένη υποστήριξη για την Cloud πλατφόρμα της Google.
- Πλούσιο επεξεργαστή διάταξης (layout) με επιλογή για προεπισκόπηση της διάταξης σε ποικίλες διαστάσεις οθονών.
- Διευρυμένη υποστήριξη προτύπων για τις υπηρεσίες της Google και για διάφορους τύπους συσκευών.
- Ύπαρξη Εικονικής Android συσκευής (Emulator) για εκτέλεση και αποσφαλμάτωση (debugging) εφαρμογών.



Εικόνα 16: Στιγμιότυπο οθόνης του προγράμματος Android Studio

5.5 Επίλογος

Συνοψίζοντας, σε αυτό το κεφάλαιο έγινε μια προσπάθεια να περιγραφούν αναλυτικά όλες οι τεχνολογίες που χρησιμοποιήθηκαν προκειμένου να ολοκληρωθεί επιτυχώς η Android εφαρμογή. Λαμβάνοντας υπόψιν λοιπόν όλα τα παραπάνω, είμαστε έτοιμοι να περιγράψουμε την αναλυτική λειτουργία της.

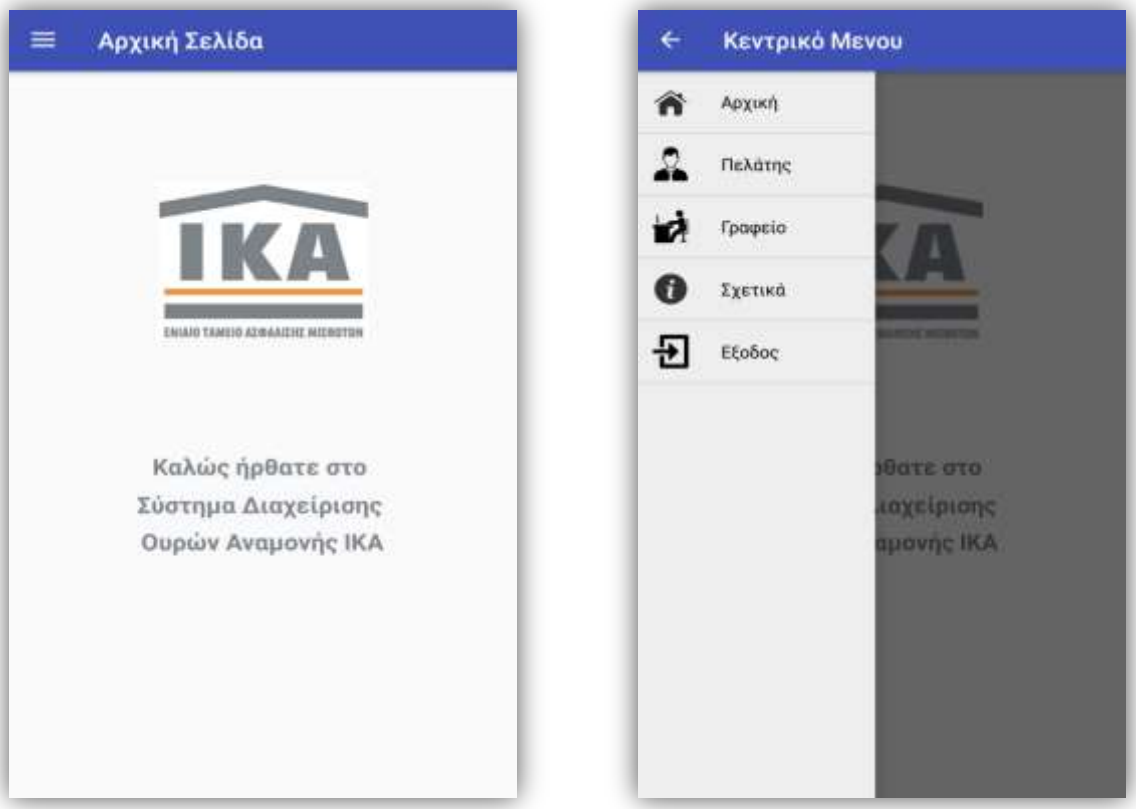
6. Αναλυτική περιγραφή και χρήση της εφαρμογής

6.1 Εισαγωγή

Στο τελευταίο αυτό κεφάλαιο της παρούσας πτυχιακής εργασίας θα περιγραφεί η αναλυτική λειτουργία της εφαρμογής με τη χρήση στιγμιοτύπων οθόνης (screenshots), ώστε να γίνει όσο το δυνατόν πιο κατανοητή στον αναγνώστη.

6.2 Αρχική Οθόνη

Μόλις ο χρήστης ανοίξει την εφαρμογή, εμφανίζεται το εισαγωγικό activity, το οποίο περιέχει το εικονίδιο (logo) της εφαρμογής, που είναι ουσιαστικά το logo του ΙΚΑ και επίσης ένα μήνυμα καλωσορίσματος. Τελευταίο και κυριότερο, το εικονίδιο πάνω αριστερά στην οθόνη, όπου πατώντας το, εμφανίζεται ένα κρυφό μενού, που περιέχει τις κύριες λειτουργίες της εφαρμογής.



Εικόνα 17-18: Αριστερά η αρχική σελίδα – Δεξιά το κρυφό drawer menu

6.3 Πελάτης

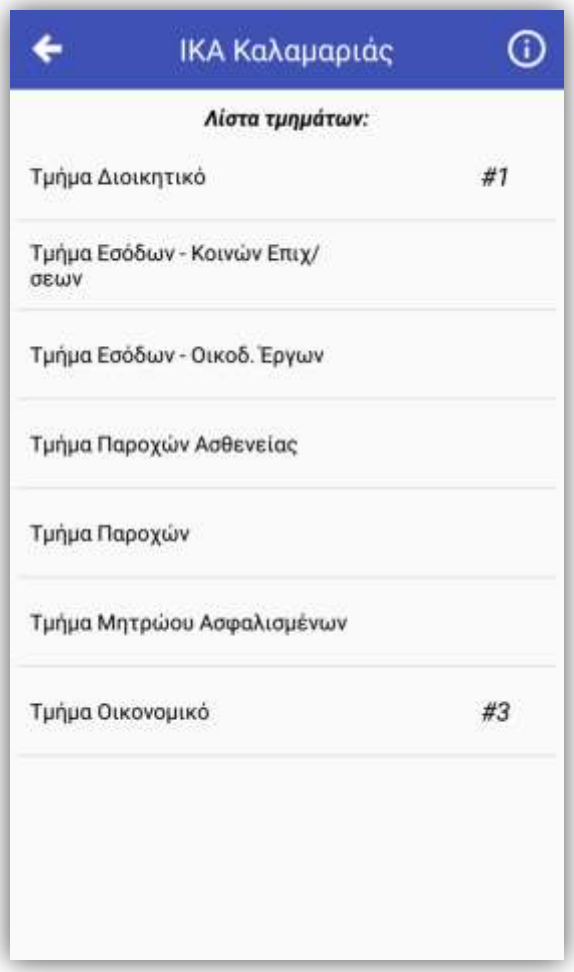
6.3.1 Λίστα καταστημάτων ΙΚΑ

Στο συγκεκριμένο fragment, φορτώνεται μια απλή λίστα με όλα τα υποκαταστήματα ΙΚΑ που υπάρχουν στην ευρύτερη περιοχή του Νομού Θεσσαλονίκης. Στο δεξί μέρος του κάθε στοιχείου της λίστας, εμφανίζεται ο αριθμός των ανοιχτών εισιτηρίων στις διάφορες ουρές/τμήματα του κάθε υποκαταστήματος, στην περίπτωση φυσικά που υπάρχουν, αλλιώς δεν εμφανίζεται τίποτα. Όπως φαίνεται, για παράδειγμα, στην παρακάτω εικόνα, ο χρήστης φαίνεται να έχει 2 ανοιχτά εισιτήρια προς εξυπηρέτηση σε 2 διαφορετικές ουρές του καταστήματος Καλαμαριάς.

Εικόνα 19: Λίστα καταστημάτων ΙΚΑ

6.3.2 Λίστα τμημάτων ΙΚΑ

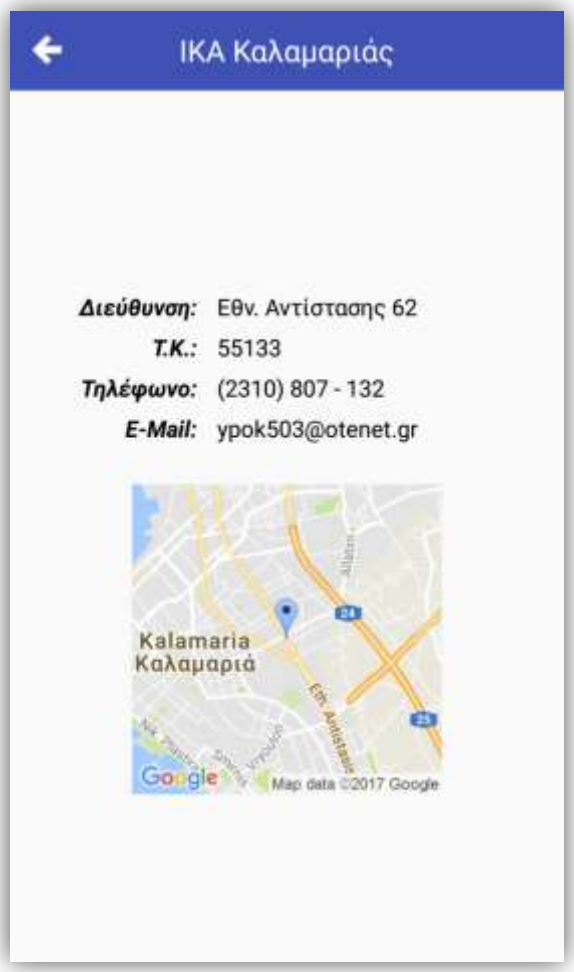
Πρόκειται για την συνέχεια του προηγούμενου fragment, όπου φορτώνεται από τη βάση μία λίστα που περιέχει τα αντίστοιχα τμήματα που υπάρχουν σε κάθε υποκατάστημα ΙΚΑ. Όπως και προηγουμένως, στο δεξιό μέρος του κάθε τμήματος, εμφανίζεται, αν υπάρχει, ο αριθμός του ενεργού εισιτηρίου που έχει στην κατοχή του ο χρήστης. Στην προκειμένη περίπτωση, ο χρήστης κατέχει το νούμερο #1 στο Τμήμα Διοικητικό και το νούμερο #3 στο Τμήμα Οικονομικών στο κατάστημα ΙΚΑ Καλαμαριάς. Τέλος, υπάρχει ένα εικονίδιο στο πάνω δεξί μέρος της οθόνης, που παραπέμπει στο αμέσως επόμενο προς περιγραφή fragment, που περιέχει τις διάφορες πληροφορίες του καταστήματος.



Εικόνα 20: Λίστα τμημάτων του εκάστοτε καταστήματος ΙΚΑ

6.3.3 Πληροφορίες καταστήματος ΙΚΑ

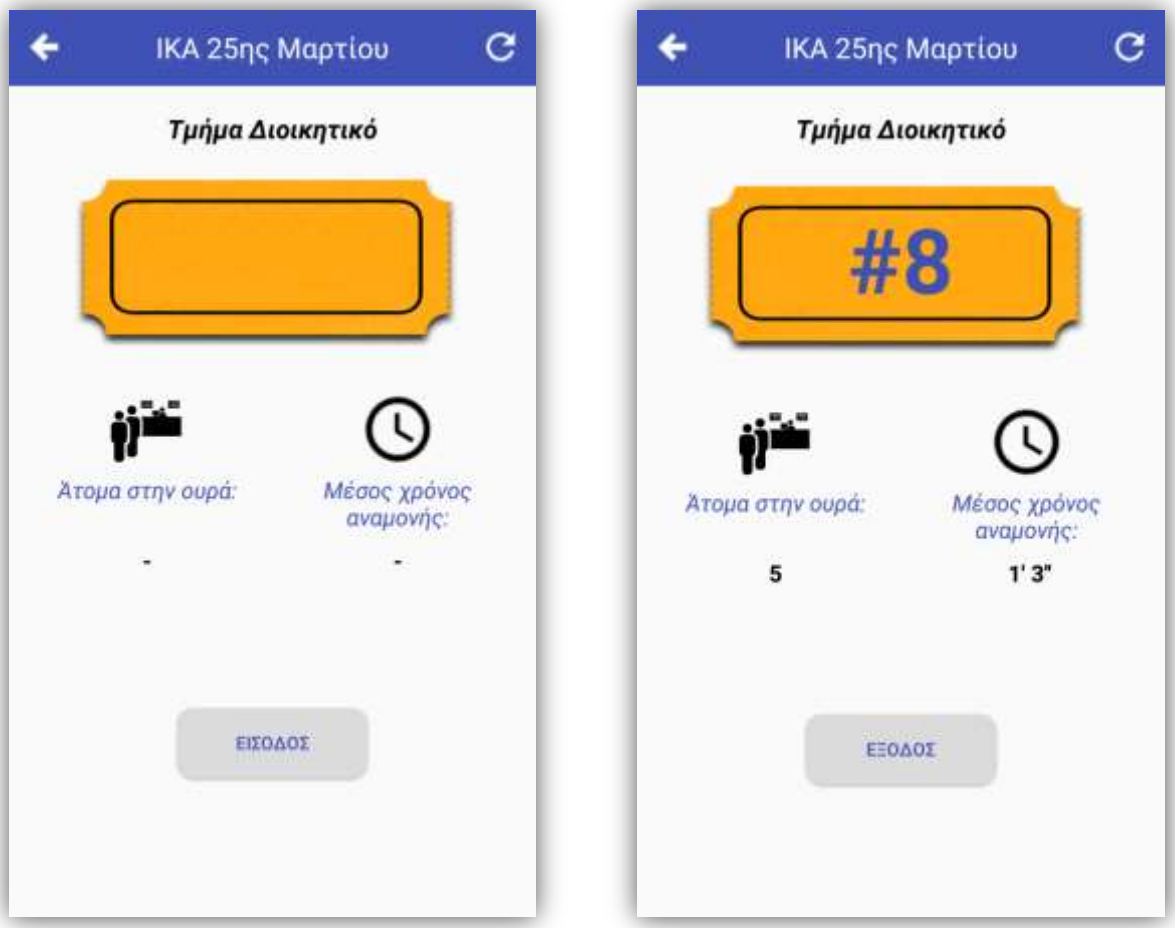
Σε αυτό το fragment περιέχονται όλες οι πληροφορίες του εκάστοτε καταστήματος που χρειάζεται ο χρήστης, όπως τηλέφωνο, διεύθυνση. Πιο αναλυτικά, όπως φαίνεται και στην παρακάτω εικόνα, εμφανίζονται κατά σειρά η διεύθυνση, ο ταχυδρομικός κώδικας, το τηλέφωνο επικοινωνίας (αν υπάρχει) και το e-mail. Επίσης, κάτω από αυτά τα πεδία, υπάρχει ένας χάρτης, που αν ο χρήστης πατήσει επάνω του, μεταφέρεται αυτόματα στους Χάρτες Google (Google Maps), έχοντας τη δυνατότητα να ζητήσει οδηγίες μετάβασης προς το συγκεκριμένο κατάστημα. Τέλος, έχει υλοποιηθεί η κλήση με το άγγιγμα του πεδίου του τηλεφώνου από τον χρήστη, όπως επίσης και του ανοίγματος της εφαρμογής Gmail για σύνθεση e-mail, πατώντας στο πεδίο e-mail.



Εικόνα 21: Πληροφορίες καταστήματος ΙΚΑ

6.3.4 Οθόνη έκδοσης εισιτηρίου

Στο συγκεκριμένο fragment της εφαρμογής, ο χρήστης έχει τη δυνατότητα να εκδώσει εισιτήριο σε οποιοδήποτε από τα τμήματα του υποκαταστήματος ΙΚΑ στο οποίο βρίσκεται. Πατώντας το κουμπί «*Είσοδος*», εμφανίζονται άμεσα στην οθόνη κάποιες βασικές πληροφορίες για τη συγκεκριμένη ουρά, όπως τον αριθμό εισιτηρίου που κατέχει, τον αριθμό των ατόμων που περιμένουν στην ουρά και τέλος τον μέσο όρο αναμονής. Επίσης, στο πάνω δεξί μέρος της οθόνης, υπάρχει ένα κουμπί ανανέωσης, όπου πατώντας το, ενημερώνονται εκ νέου τα στοιχεία για την ουρά. Η ίδια λειτουργία μπορεί να εκτελεστεί, εάν ο χρήστης αγγίξει ένα σημείο της οθόνης και απαλά σύρει το δάχτυλο προς τα κάτω. Τέλος, αν, για οποιοδήποτε λόγο, ο χρήστης θέλει να εξέλθει από την ουρά, μπορεί να πατήσει το κουμπί «*Έξοδος*», στο κάτω μέρος της οθόνης.

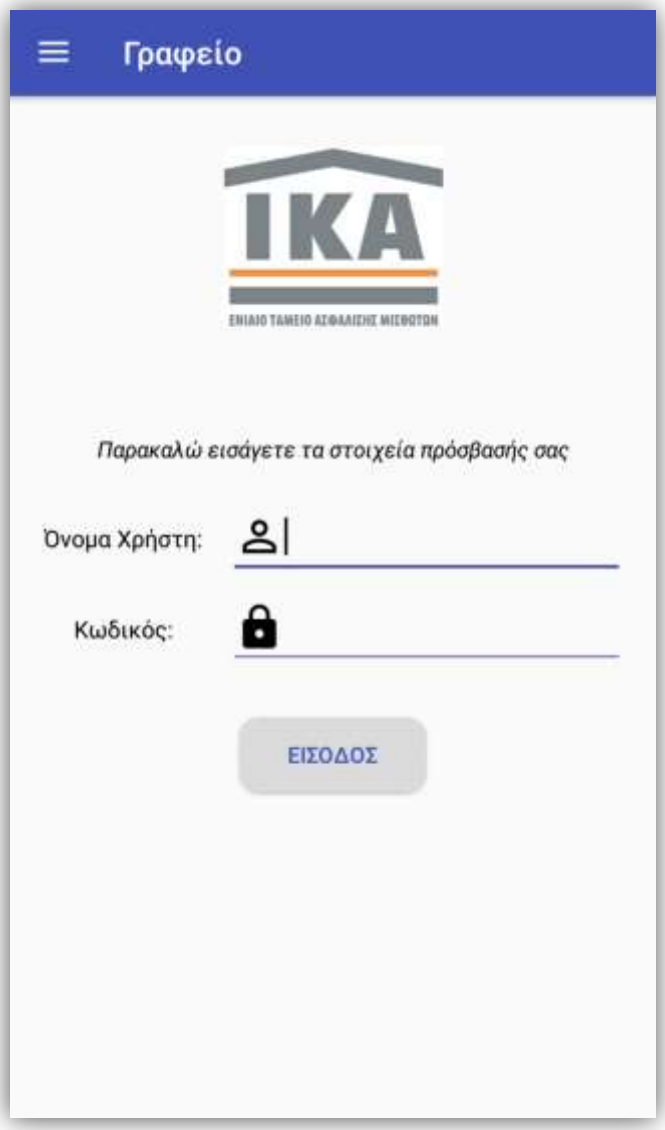


Εικόνα 22-23: Αριστερά η εμφάνιση του fragment πριν την έκδοση εισιτηρίου, δεξιά η εμφάνιση μετά την έκδοση

6.4 Γραφείο

6.4.1 Login υπαλλήλου

Στο παρών fragment, ο εκάστοτε υπάλληλος έχει παραλάβει ένα συγκεκριμένο όνομα χρήστη (username) και κωδικό πρόσβασης (password). Με αυτά τα στοιχεία εισέρχεται στην «υπηρεσία γραφείου» της εφαρμογής. Φυσικά, εάν εισάγει λάθος στοιχεία εμφανίζεται μήνυμα λάθους.



Εικόνα 24: Login Υπαλλήλου

6.4.2 Εξυπηρέτηση πελατών

Σε αυτό το fragment, εμφανίζονται όλες οι πληροφορίες που χρειάζεται ο κάθε υπάλληλος, ούτως ώστε να γνωρίζει τι συμβαίνει στην ουρά που εξυπηρετεί. Στο πάνω μέρος της οθόνης εμφανίζεται το τμήμα στο οποίο ανήκει και από κάτω 3 σημαντικά πεδία: 1) *Τρέχων εισιτήριο*, που αποτελεί τον αριθμό εισιτηρίου που εξυπηρετεί αυτή τη στιγμή ο υπάλληλος, 2) *Άτομα στην ουρά*, που αποτελεί τον αριθμό των εναπομείναντων πελατών που αναμένουν να εξυπηρετηθούν και 3) *Μέσος χρόνος εξυπηρέτησης*, που αποτελεί τον μέσο όρο που χρειάζεται ο υπάλληλος για να παρέχει τις διάφορες υπηρεσίες. Κάθε φορά που ο υπάλληλος τελειώνει με την παροχή υπηρεσιών σε κάποιο πελάτη, πιέζει το κουμπί «Επόμενο εισιτήριο», και αυτόματα, ενημερώνονται όλα τα παραπάνω στοιχεία. Τέλος, υπάρχει η δυνατότητα (όπως και στην οθόνη για έκδοση εισιτηρίου) ανανέωσης των πληροφοριών, με σκοπό να γνωρίζει ο υπάλληλος τον ρυθμό με τον οποίο εισέρχονται νέοι πελάτες/χρήστες στη συγκεκριμένη ουρά.

Εικόνα 25: Εξυπηρέτηση Πελατών

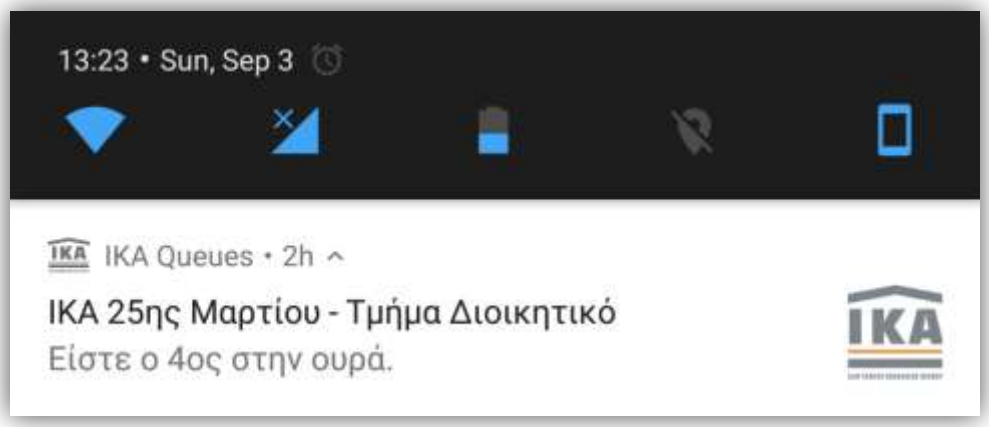
6.5 Σχετικά με την εφαρμογή

Πρόκειται για ένα απλό fragment που δίνει πληροφορίες για την εφαρμογή.

Εικόνα 26: Σχετικά με την εφαρμογή

6.6 Ειδοποιήσεις – Notifications

Τη στιγμή που ο υπάλληλος πιέζει το κουμπί «Επόμενο εισιτήριο», όπως περιγράφτηκε παραπάνω, ο εξυπηρετητής στέλνει ειδοποίηση (notification) στους χρήστες για την πρόοδο της τρέχουσας κατάστασης της ουράς. Ειδικότερα την στέλνει στον $10^{ο}$, $5^{ο}$, $4^{ο}$, $3^{ο}$, $2^{ο}$ και τέλος στον επόμενο προς εξυπηρέτηση. Παρακάτω παρατίθεται ένα στιγμιότυπο ειδοποίησης της εφαρμογής.



Εικόνα 27: Ειδοποίηση - Notification

6.7 Επίλογος

Σε αυτό το κεφάλαιο έγινε μια αναλυτική παρουσίαση της λειτουργίας και χρήσης της εφαρμογής με χρήση στιγμιοτύπων οθόνης (screenshots) και αναλυτική περιγραφή του καθενός. Ακολουθούν τα συμπεράσματα και οι μελλοντικές επεκτάσεις της εφαρμογής.

7. Επίλογος

7.1 Σύνοψη και συμπεράσματα

Στην παρούσα πτυχιακή εργασία, αναλύθηκε λεπτομερώς μια εφαρμογή για κινητά τηλέφωνα (smartphones), τα οποία χρησιμοποιούν το λειτουργικό σύστημα Android και η οποία επικοινωνεί με έναν εξυπηρετητή (server) που περιέχει μια βάση δεδομένων (database). Αποκτήθηκαν σημαντικές γνώσεις, τόσο για την σχεδίαση και για τον προγραμματισμό Android εφαρμογών, όσο και για τον τρόπο επικοινωνίας τους με τον server.

Τα κινητά καθιερώνονται όλο και περισσότερο στην καθημερινότητα των ανθρώπων, με αποτέλεσμα η δημιουργία εφαρμογών που θα κάνουν ακόμα πιο εύκολη τη ζωή μας, να κερδίζει δημοτικότητα και να αποτελεί άκρως δημοφιλές και ενδιαφέρον τομέα, που ένας προγραμματιστής αξίζει να επενδύσει τον χρόνο του και να ασχοληθεί.

7.2 Μελλοντικές επεκτάσεις

Ο τρόπος με τον οποίο έχει υλοποιηθεί η εν λόγω εφαρμογή, επιτρέπει την προσθήκη νέων χαρακτηριστικών.

Θα μπορούσαν να προστεθούν τα υποκαταστήματα ΙΚΑ ολόκληρης της Ελλάδας, όπως επίσης θα μπορούσε να χρησιμοποιηθεί ο πομπός GPS που υπάρχει σε όλα πλέον τα κινητά τηλέφωνα (smartphones), για να εντοπιστεί το κοντινότερο κατάστημα. Τέλος, θα ήταν χρήσιμη η λειτουργία για αναφορά ή και απλό σχολιασμό είτε για το υποκατάστημα είτε για την ποιότητα της εξυπηρέτησης του κάθε χρήστη από τον εκάστοτε υπάλληλο.

8. Βιβλιογραφία και εξωτερικοί σύνδεσμοι

8.1 Εξωτερικοί σύνδεσμοι

[1] http://php.net/downloads.php [Accessed: 16/09/2017]

[2] https://en.wikipedia.org/wiki/Common_Gateway_Interface [Accessed: 16/09/2017]

[3] https://code.visualstudio.com/ [Accessed: 16/09/2017]

[4] https://filezilla-project.org/download.php [Accessed: 16/09/2017]

[5] https://www.apachefriends.org/index.html [Accessed: 16/09/2017]

[6] https://en.wikipedia.org/wiki/International_Mobile_Equipment_Identity [Accessed: 16/09/2017]

[7] https://developer.android.com/design/material/index.html [Accessed: 16/09/2017]

[8] https://developer.android.com/guide/components/activities/activity-lifecycle.html [Accessed: 16/09/2017]

[9] https://developer.android.com/guide/components/fragments.html

[10] https://github.com/koush/ion [Accessed: 16/09/2017]

[11] https://firebase.google.com/docs/cloud-messaging/ [Accessed: 16/09/2017]

[12] https://developer.android.com/studio/index.html [Accessed: 16/09/2017]

8.2 Βιβλιογραφία

Burnette (2010), Hello, Android: Introducing Google's Mobile Development Platform, Pragmatic Bookshelf, Raleigh, North Carolina, United States

Deitel, A., Deitel, H., Deitel, P. (2014), Android How to Program, Pearson, London, UK

Delessio, C., Darcey, L., Conder S. (2013), Android Application Development in 24 Hours, Sams Publishing, Indianapolis, Indiana, United States

Jackson (2017), Android Apps for Absolute Beginners, Apress, New York City, New York, United States

Lockhart (2015), Modern PHP: New Features and Good Practices, O'Reilly Media, Sebastopol, California, United States

Meier (2010), Professional Android 2 Application Development, Wrox, Birmingham, UK

Welling L., Thomson L. (2009), PHP and MySQL Web Development, Pearson, London, UK