ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΑΝΑΠΤΥΞΗ ΕΦΑΡΜΟΓΗΣ ΗΛΕΚΤΡΟΝΙΚΟΥ ΠΑΡΟΥΣΙΟΛΟΓΙΟΥ ΜΕ ΧΡΗΣΗ ΔΙΑΔΙΚΤΥΑΚΩΝ ΤΕΧΝΟΛΟΓΙΩΝ ΜΕΣΩ ΤΟΥ ΣΥΣΤΗΜΑΤΟΣ ANDROID

ΖΑΧΑΡΟΠΟΥΛΟΣ ΑΘΑΝΑΣΙΟΣ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

……..ΚΟΛΟΜΒΑΤΣΟΣ ΚΩΝΣΤΑΝΤΙΝΟΣ……

……Επίκουρος καθηγητής ΠΘ……

UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

DEVELOPMENT OF ELECTRONIC PRESENTATION APPLICATION USING INTERNET TECHNOLOGIES THROUGH ANDROID SYSTEM

….. ZACHAROPOULOS ATHANASIOS ……

ADVISOR

……..KOLOMVATSOS KONSTANTINOS……..

……Assistant Professor……

«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [1], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:

1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί ***χωρ ς να τα περικλείω σε εισαγωγικά*** *και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

2. Δέχομαι ότι η αυτολεξεί ***παράθεση χωρίς εισαγωγικά****, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κ.λπ.), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια

4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.

Ημερομηνία: …21…/..10../2020…

Ο Δηλών.

ΖΑΧΑΡΟΠΟΥΛΟΣ ΑΘΑΝΑΣΙΟΣ

(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»

ΠΕΡΙΛΗΨΗ

Η πτυχιακή εργασία που έχει υλοποιηθεί έχει να κάνει με τη δημιουργία μιας android εφαρμογής για την λήψη απουσιολογίου μέσω κινητού και ειδικότερα μέσω Wi-Fi peer to peer ή Wi-Fi direct. Ο μαθητής ορίζεται από την εφαρμογή ως (client) και ο καθηγητής ως (host).Εφόσον γίνει η σύνδεση των δυο μέσω του Wi-Fi, μόνο τότε ο μαθητής έχει τη δυνατότητα να προσθέσει τη παρουσία του στη βάση δεδομένων. Σε περίπτωση που δε γίνει η σύνδεση με το καθηγητή, ο μαθητής δεν έχει το δικαίωμα εγράφης στη βάση δεδομένων. Αντίστοιχα ο καθηγητής αφού οριστεί σαν host θα έχει τη δυνατότητα πρόσβασης και επεξεργασίας της βάσης. Η απουσία ενός μαθητή εξαρτάται από το αν έχει κάνει σύνδεση με το καθηγητή ή όχι, καθώς αν δεν έχει περάσει τη παρουσία του στη βάση δεν θα ενημερωθεί η ημερομηνία της τελευταίας παρουσίας οπότε μόλις ο καθηγητής πατήσει το κουμπί για να προστεθούν οι απουσίες στους μαθητές που δεν ήταν εκεί ή δεν έκαναν σύνδεση με αυτόν, η εφαρμογή θα εισάγει στη βάση απουσία παρόλο που αυτός μπορεί να βρισκόταν στην αίθουσα. Μόλις ολοκληρωθεί η διαδικασία σύνδεσης ενός μαθητή ακολουθεί ο επόμενος. Εδώ πρέπει να τονιστεί πως για να περαστούν οι απουσίες χρειάζεται κάθε μαθητής να κάνει σύνδεση με το καθηγητή, αφού η εφαρμογή είναι βασισμένη σε μοντέλο σύνδεσης peer-to-peer(άτομο με άτομο). Αντίθετα για την υποβολή των απουσιών χρειάζεται απλά ο καθηγητής να κάνει εκκίνηση της ενημέρωσης της βάσης μέσω του κουμπιού για τις απουσίες και η εφαρμογή αυτόματα θα περάσει απουσία σε όσους δεν έχουν κάνει σύνδεση με το καθηγητή την ημέρα εκείνη.

Table of Contents

ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

ΓΕΝΙΚΟ ΠΛΑΝΟ ΕΦΑΡΜΟΓΗΣ

ΓΕΝΙΚΗ ΙΔΕΑ

Στην εφαρμογή που ακολουθεί σαν έχει τεθεί να δημιουργηθεί μια βάση δεδομένων με τα ονόματα των μαθητών μιας τάξης καθώς και με τα στοιχεία κάθε μαθητή έτσι ώστε ο καθένας να έχει τη δική του μοναδική ταυτότητα και αυτό είναι και ένα από τα προβλήματα που θα κλεισθούμε να αντιμετωπίσουμε στη συνέχεια. Ο κάθε μαθητής θα έχει τη δυνατότητα να εισάγει το ονοματεπώνυμο του τον αριθμό μητρώου του το email του και τα υπόλοιπα στοιχεία που θα προσθέσουμε έτσι ώστε να εξασφαλίσουμε τη μοναδικότητα του. Από την άλλη πλευρά ο καθηγητής θα πρέπει να έχει πρόσβαση στη βάση δεδομένων προκειμένου να παρατηρεί τις παρουσίες -απουσίες των μαθητών του καθώς και αν είναι σωστά τα στοιχεία κάθε μαθητή. Οπότε θα πρέπει να υπάρχει μια οθόνη για τον μαθητή και μια οθόνη για το καθηγητή με διαφορετικές λειτουργίες και δυνατότητες για τη κάθε πλευρά, αλλά παράλληλα θα πρέπει να κάνουν εγγραφή και διαχείριση αντίστοιχα στην ιδιά λίστα (βάση δεδομένων).

ΠΡΟΒΛΗΜΑΤΑ ΚΑΙ ΛΥΣΕΙΣ

Τα προβλήματα που υπάρχουν είναι τα εξής :

1. Η επαλήθευση των στοιχείων του μαθητή
2. Η ύπαρξη του μαθητή στη βάση
3. Η μοναδική ύπαρξη ενός μαθητή για αποφυγή λαθών στη βάση
4. Ο τρόπος σύνδεσης μαθητή και καθηγητή
5. Η αποφυγή κάποιου μαθητή να μπορεί να προσθέτει παρουσίες σε παραπάνω από ένα άτομο η αντίστοιχα να μη μπορεί να έχει πρόσβαση στη τροποποίηση της βάσης για τοίχων διαγραφή άλλων μαθητών η αλλοίωση της βάσης δεδομένων.
6. Να μην υπάρχουν δυο μαθητές με τα ίδια στοιχεία για παράδειγμα το email.

Το πρώτο από τα προβλήματα μπορούμε να το αποφύγουμε με έναν έλεγχο ταυτοποίησης στοιχείων η αυθεντικότητας του χρήστη. Ένας τρόπος για την επίλυση θα ήταν από το κάθε φοιτητή όταν κάνει εγγραφή να μπορούμε μέσω της εφαρμογής να αποθηκεύουμε ένα μοναδικό για αυτόν στοιχείο όπως για παράδειγμα την ip του κινητού του. Με μια δεύτερη ματιά όμως αυτό θα μπορούσε να μας δημιουργήσει πρόβλημα γιατί η ip του δεν θα ήταν σταθερή κάθε φορά που θα ερχόταν για σύνδεση στη βάση και δεν θα υπήρχε ταυτοποίηση. Γι' αυτό ο έλεγχος που θα γίνεται για να εξακριβώσουμε αν ο μαθητής υπάρχει μέσα στη βάση η είναι η πρώτη φορά που κάνει είσοδο σε αυτή να γίνεται με το email του.

Αυτό βέβαια σχετίζεται με ένα ακόμα πρόβλημα, το να μην υπάρχουν δυο η παραπάνω μαθητές με το ίδιο email έτσι ώστε όταν κάνουμε αναζήτηση με αυτό να μπορούμε να πάρουμε μόνο ένα μαθητή για τη διαχείριση των στοιχείων του. Προκειμένου να λυθεί αυτό το πρόβλημα θα πρέπει να συνδέσουμε το email κάθε μαθητή με ένα μοναδικό στοιχείο του κινητού του μαθητή όπως για παράδειγμα τον αριθμό κατασκευής του η τον IMEI. Οπότε σε κάθε αναζήτηση θα υπάρχει πιο αυστηρός έλεγχος στα στοιχεία για την ταυτοποίηση για να πάρουμε κάθε φορά το επιθυμητό αποτέλεσμα και να μπορούμε

να περάσουμε κάθε φορά μόνο σε έναν μαθητή τα στοιχεία για παράδειγμα τη παρουσία του στη βάση δεδομένων.

Όπως αναφέραμε και πιο πάνω ένα ακόμα σημαντικό θέμα είναι να μη μπορεί ένας μαθητής να χειριστεί τις απουσίες άλλων μαθητών που συνδέεται με την απαίτηση κάθε μαθητής να έχει διαφορετικό email για τη σωστή διαχείριση των στοιχείων του. Αυτό μπορεί να επιτευχθεί με τη διαχείριση της εγγραφής κάθε μαθητή στη βάση δεδομένων. Για παράδειγμα όταν ένας νέος μαθητής κατεβάσει την εφαρμογή και πληκτρολογήσει το email του να γίνεται μια αναζήτηση των στοιχείων του στη βάση δεδομένων και αν δεν υπάρχει να του βγάζει μια κατάλληλη φόρμα εγγραφής. Με αυτό το τρόπο κάθε μαθητής που θα εγγράφεται στη βάση θα έχει ξεχωριστό email. Από την άλλη πλευρά όταν θα έχουμε είσοδο στην εφαρμογή για υποβολή παρουσίας θα πρέπει να ελέγχουμε το άτομο στο οποίο θέλουμε να περάσουμε τη παρουσία ή την απουσία αντίστοιχα. Έτσι κρατά τις δύο μεταβλητές μοναδικότητας χρήστη και επαλήθευσης. Η πρώτη μεταβλητή μας βοηθά να μπορεί ο χρήστης να καταχωρήσει την παρουσία του στη σωστή θέση της βάσης που αντιστοιχεί στα στοιχεία του. Για την προκείμενη περίπτωση θα χρησιμοποιήσουμε το IMEI του κινητού του χρήστη και σε σύνδεση με το email του θα έχουμε εξασφαλίσει τη μοναδικότητα του, καθώς ο IMEI είναι διαφορετικός για κάθε συσκευή.

ΤΟ ΑΠΟΤΕΛΕΣΜΑ

Με βάση λοιπόν τα παραπάνω στοιχεία καθώς και το λόγο δημιουργίας της εφαρμογής, το αποτέλεσμα που θα πρέπει να έχουμε θα είναι μια εφαρμογή βασισμένη στη διευκόλυνση του καθηγητή και του μαθητή. Παράλληλα θα πρέπει να κρατήσουμε την εφαρμογή όσο πιο απλή και κατανοητή γίνεται για να μπορεί οποιασδήποτε να έχει πρόσβαση σε αυτή αλλά να μπορεί να τη χειριστεί και με ευκολία.

Για να το πετύχουμε αυτό δεν αρκεί μόνο να δημιουργήσουμε μια καλή εφαρμογή, θα πρέπει να κατασκευάσουμε και κατάλληλες ειδοποιήσεις η μηνύματα σε οποίο σημείο χρειάζεται για να μπορεί ο χρήστης να τη χειρίζεται με μεγαλύτερη ευκολία και κατανόηση.

ΚΕΦΑΛΑΙΟ 2 Επισκόπηση

ΠΡΟΓΡΑΜΜΑΤΑ ΥΛΟΠΟΙΗΣΗΣ

Στο κεφάλαιο που ακολουθεί θα παρουσιαστούν τα εργαλεία - προγράμματα που χρησιμοποιήθηκαν για την υλοποίηση και την υποστήριξη της ομαλής λειτουργίας της πτυχιακής, σε ένα κινητό με android λειτουργικό σύστημα οποιασδήποτε έκδοσης μεγαλύτερης της 4.4 KitKat. Τα προγράμματα αυτά είναι το android studio 4.1, καθώς και το firebase-Realtime database.

Για να μπορέσουμε όμως να κατανοήσουμε τη λειτουργία του android studio πρέπει πρώτα να κάνουμε μια αναφορά στα integrated development environments (IDE) προγράμματα, αφού το android studio αποτελεί μέλος αυτής της οικογενείας προγραμμάτων.

Ένα ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) είναι μια εφαρμογή λογισμικού που παρέχει ολοκληρωμένες εγκαταστάσεις σε προγραμματιστές υπολογιστών για ανάπτυξη λογισμικού. Ένα IDE αποτελείται συνήθως από τουλάχιστον έναν επεξεργαστή πηγαίου κώδικα, εργαλεία αυτοματισμού κατασκευής και ένα debugger ή αλλιώς ένα εργαλείο για τον εντοπισμό σφαλμάτων. Κάποια παραδείγματα από τέτοιου είδους IDE είναι τα εξής :

1. NetBeans: είναι ένα από τα πιο γνωστά προγράμματα σε αυτή τη κατηγορία καθώς μπορεί να υποστηρίξει πλήθος γλωσσών, μια από τις οποίες είναι η java.
2. Eclipse: είναι ένα επίσης ευρέως γνωστό πρόγραμμα το όποιο μπορεί να ανταπεξέλθει σε πολλές γλώσσες προγραμματισμού, μια από τις οποίες είναι και η C.
3. Aptana: είναι ένα πρόγραμμα το όποιο απευθύνεται κυρίως σε προγραμματισμό που έχει να κάνουν με το web development όπως κατασκευή ιστοσελίδων.
4. Android Studio: είναι ένα περιβάλλον κατάλληλο για την αναβάθμιση και τη δημιουργία εφαρμογών από συσκευές android, με πολλές δυνατότητες. Εμείς θα αναλύσουμε περεταίρω το συγκεκριμένο περιβάλλον καθώς αυτό είναι το βασικό πρόγραμμα στο όποιο υλοποιήθηκε η εφαρμογή.

ANDROID STUDIO 4.1

-ΤΙ ΕΙΝΑΙ ΤΟ ANDROID STUDIO;

Το Android Studio είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για το λειτουργικό σύστημα Android της Google, το οποίο βασίζεται στο λογισμικό IntelliJ IDEA της JetBrains και έχει σχεδιαστεί ειδικά για την ανάπτυξη Android. Είναι διαθέσιμο για λήψη σε λειτουργικά συστήματα που βασίζονται σε Windows, macOS και Linux ή ως συνδρομητική υπηρεσία το 2020.

-ΤΙ ΠΡΟΣΦΕΡΕΙ ΤΟ ANDROID STUDIO;

Το Android Studio προσφέρει:

- Ένα ευέλικτο σύστημα κατασκευής
- Ένα γρήγορο εξομοιωτή με πλήθος από χαρακτηριστικά
- Ένα περιβάλλον όπου μπορεί να αναπτυχθεί μια εφαρμογή ή προγράμματα για όλες τις συσκευές Android

- Έχει τη δυνατότητα της εφαρμογής αλλαγών στο πρόγραμμα που τρέχει σε πραγματικό χρόνο χωρίς να είναι αναγκαία η επανεκκίνηση του
- Περιέχει πρότυπα κώδικα που βοηθούν να δημιουργηθούν κοινές δυνατότητες εφαρμογών και να εισαχθούν δείγματα κώδικα
- Εκτεταμένα εργαλεία δοκιμής και frameworks για μεγαλύτερη ευκολία στην ανάπτυξη και στη διόρθωση του κώδικα
- Εργαλεία για την απόδοση, τη χρηστικότητα, τη συμβατότητα έκδοσης και άλλα για την αντιμετώπιση προβλημάτων
- Υποστήριξη C ++ και NDK
- Τη δυνατότητα έκδοσης της εφαρμογής που έχετε δημιουργήσει είτε μέσω του play store είτε προωθώντας τη οι ίδιοι.

FIREBASE REALTIME DATABASE

-ΤΙ ΕΙΝΑΙ ΤΟ FIREBASE REALTIME DATABASE;

Αρχικά θα πρέπει να μελετήσουμε την έννοια της βάσης δεδομένων. Τι είναι λοιπόν η βάση δεδομένων; Είναι μια οργανωμένη συλλογή δεδομένων, γενικά αποθηκευμένη και προσβάσιμη ηλεκτρονικά από ένα σύστημα υπολογιστή.

Πιο συγκεκριμένα η firebase Realtime database είναι μια βάση δεδομένων που φιλοξενείται στο cloud. Τα δεδομένα αποθηκεύονται ως JSON αντικείμενα και συγχρονίζονται σε πραγματικό χρόνο με κάθε συνδεδεμένο χρήστη. Το Firebase Realtime Database χρησιμοποιεί συγχρονισμό δεδομένων. Κάθε φορά που υπάρχει τροποποίηση στα δεδομένα της, οποιαδήποτε συνδεδεμένη συσκευή λαμβάνει αυτήν την ενημέρωση σε λιγότερο από ένα δευτερόλεπτο και ανανεώνει τα στοιχεία της. Οι εφαρμογές Firebase μπορούν να ανταπεξέλθουν ακόμη και όταν είναι εκτός σύνδεσης, επειδή το SDK Firebase Realtime Database αποθηκεύει τα δεδομένα στο δίσκο. Μόλις αποκατασταθεί η συνδεσιμότητα, η συσκευή από τη πλευρά του client λαμβάνει τις αλλαγές που έχασε, συγχρονίζοντάς την με την τρέχουσα κατάσταση της συσκευής από τη πλευρά του host. Η πρόσβαση στη Βάση δεδομένων Firebase Realtime μπορεί να γίνει είτε από μια κινητή συσκευή είτε από ένα πρόγραμμα περιήγησης ιστού. Η ασφάλεια και η επικύρωση των δεδομένων, διατίθενται μέσω των κανόνων ασφαλείας του Firebase Realtime Database, που εκτελούνται όταν διαβάζονται ή γράφονται τα δεδομένα.

-ΠΩΣ ΛΕΙΤΟΥΡΓΕΙ;

Η βάση δεδομένων Firebase Realtime υποστηρίζει τη δημιουργία εφαρμογών επιτρέποντας την ασφαλή πρόσβαση στη βάση δεδομένων. Τα δεδομένα διατηρούνται τοπικά και ακόμη και όταν είναι εκτός σύνδεσης. Όταν η συσκευή ανακτήσει τη σύνδεση, η βάση δεδομένων συγχρονίζει τις τοπικές αλλαγές δεδομένων με τις απομακρυσμένες ενημερώσεις.

Η βάση δεδομένων Realtime παρέχει μια ευέλικτη γλώσσα κανόνων που βασίζεται στον τρόπο δομής των δεδομένων και πότε μπορούν να διαβαστούν ή να γραφτούν τα δεδομένα. Όταν ενσωματωθούν, οι προγραμματιστές μπορούν να ορίσουν ποιος έχει πρόσβαση σε ποια δεδομένα και πώς μπορούν να έχουν πρόσβαση σε αυτά.

Η βάση δεδομένων Realtime είναι μια βάση δεδομένων NoSQL. Μια τέτοια βάση δεδομένων παρέχει έναν μηχανισμό αποθήκευσης και ανάκτησης δεδομένων, ως εκ τούτου έχει διαφορετικές βελτιστοποιήσεις και λειτουργικότητα σε σύγκριση με μια σχεσιακή βάση δεδομένων. Το Realtime Database API έχει σχεδιαστεί για να επιτρέπει μόνο λειτουργίες που μπορούν να εκτελεστούν γρήγορα. Αυτό μας δίνει τη δυνατότητα να δημιουργήσουμε μια εφαρμογή που σε πραγματικό χρόνο να μπορεί να εξυπηρετήσει πολλούς χρήστες χωρίς να υπάρξει πρόβλημα στην ανταπόκριση.

ΛΕΙΤΟΥΡΓΙΚΟ ANDROID

ΤΙ ΕΙΝΑΙ ΤΟ ANDROID

Το Android είναι ένα λειτουργικό σύστημα για συσκευές κινητής τηλεφωνίας το οποίο τρέχει τον πυρήνα του λειτουργικού Linux. Αρχικά αναπτύχθηκε από την Google και αργότερα από την Open Handset Alliance. Επιτρέπει στους κατασκευαστές λογισμικού να συνθέτουν κώδικα με την χρήση της γλώσσας προγραμματισμού Java, ελέγχοντας την συσκευή μέσω βιβλιοθηκών λογισμικού ανεπτυγμένων από την Google. Το Android είναι κατά κύριο λόγο σχεδιασμένο για συσκευές με οθόνη αφής, όπως τα έξυπνα τηλέφωνα και τα τάμπλετ, με διαφορετικό περιβάλλον χρήσης για τηλεοράσεις (Android TV), αυτοκίνητα (Android Auto) και ρολόγια χειρός (Android Wear). Παρόλο που έχει αναπτυχθεί για συσκευές με οθόνη αφής, έχει χρησιμοποιηθεί σε κονσόλες παιχνιδιών, ψηφιακές φωτογραφικές μηχανές και σε άλλες ηλεκτρονικές συσκευές.

ΧΑΡΑΚΤΗΡΙΣΤΙΚΑ ΤΟΥ ANDROID

Το λειτουργικό σύστημα android από τη δημιουργία του έως σήμερα έχει αναπτυχθεί και έχει εξελιχθεί με αποτέλεσμα πολλά από τα χαρακτηριστικά του να έχουν αλλάξει. Στις πιο πρόσφατες εκδόσεις του λειτουργικού τα χαρακτηριστικά που μπορεί κανείς να συναντήσει είναι τα εξής :

- Η πλατφόρμα μπορεί να υποστηριχθεί από πολλές αναλύσεις οθόνης (από VGA μέχρι 4K).
- Γίνεται χρήση της βάσης δεδομένων SQLite για τις ανάγκες αποθήκευσης.
- Το Android υποστηρίζει τεχνολογίες συνδεσιμότητας συμπεριλαμβανομένου GSM/EDGE/UMTS/HSPA/HSPA+/LTE, 3G, 4G, CDMA, EV-DO, Bluetooth, NFC, και Wi-Fi.
- Οι τρόποι ανταλλαγής μηνυμάτων σε αυτό γίνετε μέσω SMS και MMS.
- Δυνατότητα περιήγησης στο ίντερνετ μέσω του ANDROID browser, ή μέσω άλλων από το play store που μπορούν να τον αντικαταστήσουν.
- Έχει υποστήριξη σε ένα ευρύ φάσμα από μορφές ήχου, καθώς και μορφές στατικής και κινουμένης εικόνας.
- Το λειτουργικό Android μπορεί να λειτουργήσει με πολλά εξαρτήματα hardware όπως για παράδειγμα κάμερες στατικής ή κινούμενης εικόνας, οθόνες αφής, GPS, αισθητήρες επιτάχυνσης και άλλα.
- Το περιβάλλον ανάπτυξης του λογισμικού περιέχει προσομοιωτή συσκευής που δίνει τη δυνατότητα άμεσης προβολής της εφαρμογής σε αναπαράσταση κινητού, καθώς και πολλά εργαλεία για τη διόρθωση λαθών στο κώδικα και όχι μόνο πχ απόδοση του εκτελέσιμου λογισμικού, ανάλυση της εφαρμογής κ.λπ.
- Διαθέτει ένα μεγάλο κατάλογο εφαρμογών (google play) από τον οποίο μπορούν να μεταφορτωθούν και να εγκατασταθούν πολλές εφαρμογές στη συσκευή χωρίς τη χρήση υπολογιστή.

ΣΤΟΧΟΣ ΠΤΥΧΙΑΚΗΣ

Αυτά τα δυο προγράμματα λοιπόν μπορούν να συνεργαστούν άψογα και να δημιουργήσουν, με το κατάλληλο κάθε φορά κώδικα, εφαρμογές για τη διαχείριση και την ενημέρωση μιας λίστας δεδομένων. Στη σημερινή εποχή υπάρχει πληθώρα τέτοιου είδους εφαρμογών κυρίως για την εξυπηρέτηση online καταστημάτων, αποθηκών προϊόντων και άλλων πολλών επιχειρήσεων. Ο στόχος που θέσαμε στη πτυχιακή είναι η υλοποίηση ενός παρουσιολογίου, χρησιμοποιώντας τους μαθητές - φοιτητές ως δεδομένα σε μια λίστα με τα στοιχεία που τους περιγράφουν πχ όνομα, αριθμό μητρώου κλπ., έτσι ώστε σαν αποτέλεσμα να έχουμε μια βάση δεδομένων η οποία θα ενημερώνεται κάθε φορά σε πραγματικό χρόνο. Με αυτό το τρόπο κάθε μαθητής θα μπορεί να εισάγει τη παρουσία του στη βάση δεδομένων και στη συνέχεια ο καθηγητής να μπορεί να ελέγχει τις παρουσίες κάθε μαθητή και μέσω της εφαρμογής να ενημερώνει τις απουσίες από τους μαθητές που δε παρουσιάστηκαν.

Για να υλοποιηθεί η εφαρμογή έπρεπε να υπάρχει ένας τρόπος σύνδεσης αναμεσά στο καθηγητή και το μαθητή. Το αντικείμενο που συνδέει αυτούς τους δυο είναι το Wi-Fi και πιο συγκεκριμένα το Wi-Fi direct. Το Wi-Fi direct είναι ένα πρότυπο Wi-Fi για ασύρματες συνδέσεις peer-to-peer που επιτρέπει σε δύο συσκευές να δημιουργούν απευθείας σύνδεση Wi-Fi χωρίς ενδιάμεσο σημείο ασύρματης πρόσβασης, δρομολογητή ή σύνδεση στο Διαδίκτυο.

ΓΙΑΤΙ WIFI DIRECT ΚΑΙ ΟΧΙ BLUETOOTH Η NFC;

	Bluetooth 5.0	WiFi Direct
Peer-to-peer sharing	Yes	Yes
Speeds	1-3 Mbit/s	>54Mbits/s
Range	100m	46-100m
Energy consumption	0.01–1.0 W	2 to 20 watts
Frequency	2.4Ghz	2.4 or 5.0Ghz
Service discovery	Yes	Yes
Supported devices	Smartphones Smart TVs Laptops Smartwatches	Smartphones Smart TVs Laptops Smartwatches Desktop computers

Εικόνα 1

Όπως μπορούμε να δούμε και από την εικόνα το Wi-Fi direct έχει πολύ μεγαλύτερες ταχύτητες μεταφοράς δεδομένων από ότι το Bluetooth, ενώ μπορεί να είναι συμβατό και με σταθερούς υπολογιστές. Επιπλέον οι παλιότερες συσκευές με την έκδοση Bluetooth 4.0 έχουν μικρότερη εμβέλεια στο σήμα σύνδεσης, οπότε και για λογούς απόστασης μέσα σε μια μεγάλη αίθουσα η αμφιθέατρο επιλέχθηκε το Wi-Fi direct. Αυτό μπορούμε να το δούμε και στην Εικόνα 1.

Σε αντίθεση με το NFC το Wi-Fi direct υποστηρίζεται από όλες τις συσκευές android συνεπώς το κάνει πιο προσιτό για το σχεδιασμό μια βασικής εφαρμογής παρουσιολογίου με ανταπόκριση στο όλο το ποσοστό των κινητών με λειτουργικό android.

ΚΕΦΑΛΑΙΟ 3 FRAMEWORKS

ΕΙΣΑΓΩΓΗ ΣΤΑ FRAMEWORKS

Ο ΟΡΟΣ FRAMEWORK

Στον προγραμματισμό υπολογιστών, ένα framework αποτελεί ένα πλήθος δομημένων αρχείων κώδικα σε φακέλους κατά το οποίο το λογισμικό μπορεί να αλλάξει με πρόσθετο κωδικό χρήστη που έχει σαν αποτέλεσμα τη δημιουργία εφαρμογών. Τα frameworks μπορεί να περιλαμβάνουν προγράμματα υποστήριξης, μεταγλωττιστές, βιβλιοθήκες κώδικα, σύνολα εργαλείων και διεπαφές προγραμματισμού εφαρμογών (API) που συγκεντρώνουν όλα τα διαφορετικά στοιχεία για την ανάπτυξη ενός έργου ή συστήματος. Χρησιμοποιούνται έτσι ώστε ο προγραμματιστής να μπορεί να αναπτύξει ένα υπάρχων κομμάτι κώδικα και να μη χρειάζεται να το δημιουργήσει από την αρχή. Αυτό επιτυγχάνεται μέσω της επαναχρησιμοποίησης του κώδικα που μας προσφέρουν τα frameworks.

ΒΑΣΙΚΗ ΔΙΑΦΟΡΑ FRAMEWORK ΚΑΙ ΕΝΟΣ LIBRARY

Για να αναλύσουμε τη διαφορά μεταξύ τους πρέπει να δούμε τι περιέχει το καθένα. Μόλις εγκατασταθεί και διαμορφωθεί ένα framework, δημιουργεί μια δομή καταλόγου. Κάθε ένας από αυτούς τους φακέλους θα μπορούσε να έχει επιπλέον καταλόγους. Οι κατάλογοι μπορούν περαιτέρω να έχουν αρχεία, τάξεις, ρουτίνες δοκιμών, πρότυπα και άλλα. Οι βιβλιοθήκες είναι APIs(διεπαφές προγραμματισμού εφαρμογών) υψηλού επιπέδου που μπορούν να δημιουργήσουν ή να μετατρέψουν κώδικα BYTE.

Κάποιοι μπορεί να υποθέσουν ότι ένα framework είναι μια συλλογή βιβλιοθηκών, όπως ακριβώς και οι βιβλιοθήκες είναι μια συλλογή προ-μεταγλωττισμένων ρουτίνων. Ωστόσο, αυτό δεν ισχύει καθώς δεν χρησιμοποιούνται ή δεν εξαρτώνται όλα τα frameworks από βιβλιοθήκες. Η διαφορά μεταξύ μιας βιβλιοθήκης και ενός πλαισίου λογισμικού(framework) είναι ότι το framework καλεί τον κώδικα. Σε αντίθεση με αυτό, ο κώδικας καλεί τη βιβλιοθήκη λογισμικού.

ΟΙ ΚΑΤΗΓΟΡΙΕΣ ΤΩΝ FRAMEWORK

Υπάρχουν πολλοί τύποι πλαισίων λογισμικού που διευκολύνουν την ανάπτυξη εφαρμογών για ένα ευρύ φάσμα τομέων ανάπτυξης εφαρμογών. Κάποια από τα σημεία που θα συναντήσουμε τα frameworks είναι σε έναν ιστότοπο, επιστήμη δεδομένων, διαχείριση βάσεων δεδομένων ή εφαρμογές για κινητά. Τα frameworks χωρίζονται σε τρείς βασικές κατηγορίες :

1. Web Application Frameworks
2. Data Science Frameworks
3. Mobile Development Frameworks

Κάθε μια από αυτές τις κατηγορίες έχει τα δικά της εργαλεία και προγράμματα που βασίζονται πάνω σε αυτά για την δημιουργία διαφόρων μορφών κώδικα. Για παράδειγμα στη πρώτη κατηγορία υπάρχουν τα εργαλεία όπως το Laravel και Angular.

- Το Angular είναι ένα framework JavaScript ανοιχτού κώδικα που διευκολύνει τη δημιουργία εφαρμογών στον ιστό για συσκευές κινητών και υπολογιστών.
  https://angular.io/
- Το Laravel είναι ένα framework εφαρμογών ιστού βασισμένο σε PHP με ευέλικτη σύνταξη.
  https://laravel.com/

Στη δεύτερη κατηγορία έχουμε τα εργαλεία Apache Spark, PyTorch και TensorFlow που σχετίζονται με τα Data Science Frameworks.

- Το Apache Spark είναι μια μηχανή ανάλυσης για επεξεργασία δεδομένων μεγάλης κλίμακας. https://spark.apache.org/
- Το PyTorch είναι ένα framework μηχανικής εκμάθησης ανοιχτού κώδικα που επιταχύνει τη διαδικασία από την έρευνα και το πρωτότυπο έως την ανάπτυξη παραγωγής. https://pytorch.org/
- Το TensorFlow είναι ένα ολοκληρωμένο framework ανοιχτού κώδικα για μηχανική μάθηση (ML/ machine learning). https://www.tensorflow.org/

Στη τρίτη κατηγορία έχουμε τα εργαλεία με frameworks που σχετίζονται με την ανάπτυξη εφαρμογών κινητού, όπως το Ionic, Xamarin και το Flutter.

- Το Ionic είναι ένα δωρεάν εργαλείο UI(user interface) ανοιχτού κώδικα για ανάπτυξη υψηλής ποιότητας εγγενών εφαρμογών πολλαπλών πλατφορμών για Android, iOS και Web. https://ionicframework.com/
- Το Xamarin είναι μια δωρεάν πλατφόρμα ανάπτυξης εφαρμογών για τη δημιουργία εφαρμογών Android, iOS με .NET και C #. https://dotnet.microsoft.com/apps/xamarin
- Το Flutter είναι το εργαλείο της Google για τη δημιουργία εγγενώς μεταγλωττισμένων εφαρμογών για κινητά, ιστούς και υπολογιστές από μία μόνο βάση κώδικα. https://flutter.dev/

Υπάρχουν πάρα πολλά framework σε κάθε μια από αυτές τις τρεις κατηγορίες. Αυτά που αναφέραμε είναι ένα παράδειγμα για να καταλάβουμε το πλήθος τους καθώς και τις διαφορετικές λειτουργίες που μπορεί να έχει το καθένα από αυτά σε σύγκριση με κάποιο άλλο framework της ίδιας ή και άλλης κατηγορίας.

ΘΕΤΙΚΑ ΚΑΙ ΑΡΝΙΤΙΚΑ ΑΠΟ ΤΗ ΧΡΗΣΗ ΤΩΝ FRAMEWORK

Όπως κάθε στοιχείο έτσι και τα framework ανάλογα με το τρόπο ή το σκοπό χρήσης τους έχουν κάποια αρνητικά η θετικά στοιχεία. Ένα από τα θετικά στοιχεία είναι ότι η βασική δομή του κώδικα έχει ήδη δημιουργηθεί, άρα θα χρειαστεί λιγότερος χρόνος για τη δημιουργία ή τη τροποποίηση ενός λογισμικού πάνω σε αυτή τη δομή κώδικα που υπάρχει μέσω του framework. Αυτό βοηθάει στη βελτίωση της αποτελεσματικότητας αφού εξοικονομείται χρόνος για τη δημιουργία μιας νέας εφαρμογής. Ένας ακόμη σημαντικός λόγος για τη χρήση των framework είναι ότι εξοικονομείτε χρόνος και έξοδα για την ανάπτυξη μιας εφαρμογής καθώς δεν χρειάζεται η δημιουργία βασικών τμημάτων του κώδικα μιας που τα frameworks παρέχουν αρκετά από αυτά. Τα πιο δημοφιλή framework είναι δωρεάν. Με βάση το προηγούμενο στοιχείο υπάρχει μια μεγάλη κοινότητα προγραμματιστών που υποστηρίζει τη σωστή λειτουργία των framework και αν εντοπιστεί κάποιο πρόβλημα αντιμετωπίζεται συνήθως άμεσα. Επιπλέον υπάρχουν έγγραφα υποστήριξης καθώς και πολλές ομάδες που μπορούν να δώσουν λύσεις σε θέματα πάνω και γύρω από τα framework. Για παράδειγμα μπορεί κάποιος να βρει απαντήσεις για την αντιμετώπιση προβλημάτων πάνω σε κώδικα η στη λειτουργία του εργαλείου για τη σωστή διαχείριση του κώδικα.

Από την άλλη πλευρά ένα framework περιορίζει το προγραμματιστή να τροποποιήσει σε μεγάλο βαθμό το κώδικα καθώς περιέχει μέσα κομμάτια του για τη σωστή του λειτουργία. Με αυτό το τρόπο

κάποιος που χρησιμοποιεί framework μαθαίνει να το χειρίζεται και να αναπτύσσει εφαρμογές, όμως δε μαθαίνει τη γλώσσα προγραμματισμού στην οποία αναπτύσσει την εφαρμογή. Γενικά τα framework έχουν υλοποιηθεί για το καλύτερο σχεδιασμό του λογισμικού αλλά δεν έχουν καμία σχέση με τη γλώσσα προγραμματισμού. Όπως αναφέραμε και προηγουμένως κάθε framework έχει περιορισμούς, έτσι κάθε εφαρμογή που χτίζεται μέσα σε ένα framework πρέπει να πληροί κάποιες βασικές προϋποθέσεις. Αυτό έχει σαν αποτέλεσμα να περιορίζονται οι δυνατότητες που έχει κάποιος για τη δημιουργία μιας νέας εφαρμογής καθώς θα πρέπει να συμβιβαστεί με τους περιορισμούς του κάθε framework.

ΚΕΦΑΛΑΙΟ 4 ΑΝΑΛΥΣΗ ΕΦΑΡΜΟΓΗΣ

ΠΕΡΙΓΡΑΦΗ ΤΟΥ ΚΩΔΙΚΑ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

ANDROID MANIFEST

Για να μπορέσουμε να χρησιμοποιήσουμε το Wi-Fi και να το χειριστούμε σωστά στην εφαρμογή, θα πρέπει να δώσουμε κάποιες άδεις έτσι ώστε η εφαρμογή να έχει πρόσβαση στο κινητό και να μπορεί:

- να αλλάξει τη κατάσταση του Wi-Fi
- να έχει τη δυνατότητα της αναζήτησης άλλων συσκευών
- να πραγματοποιήσει σύνδεση με μια άλλη συσκευή
- να προβάλει τις συσκευές που εντόπισε σε μια μορφή λίστας

```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.myapplication">

    <uses-permission android:name="android.permission.ACCESS_COARSE_LOCATION" />
    <uses-permission android:name="android.permission.ACCESS_FINE_LOCATION" />
    <uses-permission android:name="android.permission.READ_PHONE_STATE" />
    <uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
    <uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
    <uses-permission android:name="android.permission.READ_PRIVILEGED_PHONE_STATE"
        tools:ignore="ProtectedPermissions" />
    <uses-permission android:name="android.permission.READ_PRIVILEGED_PHONE_STATE"
        tools:ignore="ProtectedPermissions" />

    <application
        android:allowBackup="true"
        android:icon="@mipmap/ic_launcher"
        android:label="Wifi P2P"
        android:roundIcon="@mipmap/ic_launcher_round"
        android:supportsRtl="true"
        android:theme="@style/AppTheme">
        <activity android:name=".SingInActivity"></activity>
        <activity android:name=".StudentDetailsActivity" />
        <activity android:name=".NewStudentActivity" />
        <activity android:name=".MainActivity2" />
        <activity
            android:name=".MainActivity"
            tools:ignore="Instantiatable">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />

                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>

    </application>

</manifest>
```

Εικόνα 2

Όπως φαίνεται και στην εικόνα 2 πιο πάνω προσθέσαμε κάποια permissions για να μπορέσουμε να έχουμε όλες αυτές τις δυνατότητες στην εφαρμογή. Οι άδειες που έχουν να κάνουν με το Wi-Fi είναι οι εξής :

```xml
<uses-permission android:name="android.permission.ACCESS_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_WIFI_STATE" />
<uses-permission android:name="android.permission.CHANGE_NETWORK_STATE" />
<uses-permission android:name="android.permission.INTERNET" />
<uses-permission android:name="android.permission.ACCESS_NETWORK_STATE" />
```

Με την ένταξη αυτών των αδειών στο κώδικα πλέον η εφαρμογή μπορεί να διαχειριστεί τους πόρους του Wi-Fi και να τους αξιοποιήσει για την αλλαγή κατάστασης του Wi-Fi (ON/OFF),για την αναζήτηση συσκευών που έχουν ανοιχτό το Wi-Fi, τη προβολή αυτών σε λίστα καθώς και τη σύνδεση με κάποια συσκευή από τη λίστα.

Επιπλέον έχουμε άδειες για το location(GPS) που είναι αναγκαία για τη σωστή λειτουργία της εφαρμογής. Όπως έχουμε αναφέρει και νωρίτερα η εφαρμογή χρησιμοποιεί το Wi-Fi και πιο συγκεκριμένα το Wi-Fi direct. Αυτή η λειτουργία βγάζει τα αποτελέσματα της αναζήτησης σε μια λίστα έτσι ώστε να μπορεί να τα χειριστεί ο χρήστης και να κάνεις συνδέσεις με άλλες συσκευές. Μέχρι την έκδοση του λειτουργικού android 7 το Wi-Fi direct για να εμφανίσει αυτή τη λίστα χρειάζεται μόνο τις άδειες για το Wi-Fi. Από την επόμενη έκδοσή του όμως, το android για να μπορέσει να βρει τη συσκευή και να την εμφανίσει στη λίστα θα πρέπει να έχει ενεργοποιημένη τη λειτουργία της τοποθεσίας (GPS). Συνεπώς με τις άδειες:

```
<uses-permission      android:name="android.permission.ACCESS_COARSE_LOCATION"      />
<uses-permission       android:name="android.permission.ACCESS_FINE_LOCATION"       />
<uses-permission android:name="android.permission.READ_PHONE_STATE" />
```

μπορούμε και στις πιο σύγχρονες εκδόσεις του android να έχουμε τη δυνατότητα της ομαλής εκτέλεσης της εφαρμογής, χωρίς τεχνικές αστοχίες.

Προηγουμένως είχαμε αναφέρει τα προβλήματα που θα κληθούμε να λύσουμε στην εφαρμογή και ένα από αυτά ήταν η μοναδικότητα του χρήστη και πως μπορούμε να την εξασφαλίσουμε. Σαν λύση είχε αναφερθεί ο αριθμός IMEI κάθε συσκευής να συνδέεται με το χρήστη. Για να μπορέσουμε να έχουμε πρόσβαση και να αντιγράψουμε τη διεύθυνση του χρήστη θα πρέπει να δώσουμε την απαραίτητη άδεια στην εφαρμογή.

```
<uses-permission        android:name="android.permission.READ_PRIVILEGED_PHONE_STATE"
    tools:ignore="ProtectedPermissions" />
```

Στη συνέχεια του κώδικα αυτής της σελίδας έχουμε την αρχική δήλωση της εφαρμογής, γενικές μεταβλητές για τη παρουσίασή της όπως το όνομα της η αρχική σελίδα της, το background.

STUDENT

Σε αυτό το κομμάτι του κώδικα γίνεται η σύνταξη μιας νέας κλάσης που θα είναι υπεύθυνη για τη δημιουργία ενός μαθητή και παράλληλα θα δηλώνει τα στοιχεία που θα έχει ο κάθε μαθητής. Δημιουργεί όλες τις απαραίτητες μεταβλητές που θα χρειαστούν ώστε να μπορεί η εφαρμογή να αποθηκεύει για κάθε χρήστη τις πληροφορίες για αυτόν, όπως για παράδειγμα το όνομά του, το email του κλπ.

```
    public String getFullname() {
        return fullname;
    }
    public void setFullname(String fullname) {
        this.fullname = fullname;
    }
    public String getPresences() {
        return presences;
    }
    public void setPresences(String presences) {
        this.presences = presences;
    }
    public String getAbsences() {
        return absences;
    }
    public void setAbsences(String absences) {
        this.absences = absences;
    }
    public String getEmail() {
        return email;
    }
    public void setEmail(String email) {this.email = email;}
}
```

Εικόνα 3

Για το λόγο αυτό υπάρχει ο contractor που είναι υπεύθυνος για τη δημιουργία του νέου μαθητή και την ανάθεση των πληροφοριών στις μεταβλητές που θα αποθηκεύουν τα στοιχεία του. Στη συνέχεια ακολουθούν τα get και set, που είναι μικρά functions για τη διαχείριση και τη πιο εύκολη ροή της πληροφορίας. Μας δίνουν τη δυνατότητα να μπορούμε να καλέσουμε για κάθε μαθητή τα στοιχεία του ανάλογα με το τι χρειαζόμαστε κάθε φορά, αλλά και τη δυνατότητα να αλλάξουμε και να ενημερώσουμε τα στοιχεία του σε οποιοδήποτε κομμάτι του κώδικα.

```
public Student(String am,String MAC,String date, String fullname, String email, String
presences,                  String                  absences)                  {
    this.am                                 =                                 am;
    this.MAC=MAC;
    this.date=date;
    this.fullname                           =                           fullname;
    this.email                              =                              email;
    this.presences                          =                          presences;
    this.absences                           =                           absences;
}
```

STUDENT DETAILS ACTIVITY

Σε αυτή τη σελίδα του κώδικα μπορούμε να διαχειριστούμε τα στοιχεία ενός χρήστη για να τα ενημερώσουμε είτε να διαγράψουμε το χρήστη. Αρχικά δηλώνουμε όλα τα απαραίτητα κουμπιά και

πλαίσια κειμένου καθώς αυτά θα περιέχουν τα στοιχεία κάθε χρήστη. Όλα αυτά θα εμφανιστούν στη πλευρά του user interface (UI),δηλαδή στη πλευρά που θα βλέπει ο χρήστης τη συγκεκριμένη σελίδα του κώδικα. Όπως φαίνεται από τις εικόνες πιο πάνω πρέπει να χρησιμοποιήσουμε και να διαχειριστούμε το front-end και back-end της εφαρμογής. Το back-end κομμάτι ασχολείται με τη μορφή που θα έχει ο κώδικας της εφαρμογής, σε αντίθεση με το front-end που έχει να κάνει με τη τελική εικόνα που θα βλέπει ο χρήστης από την εφαρμογή.

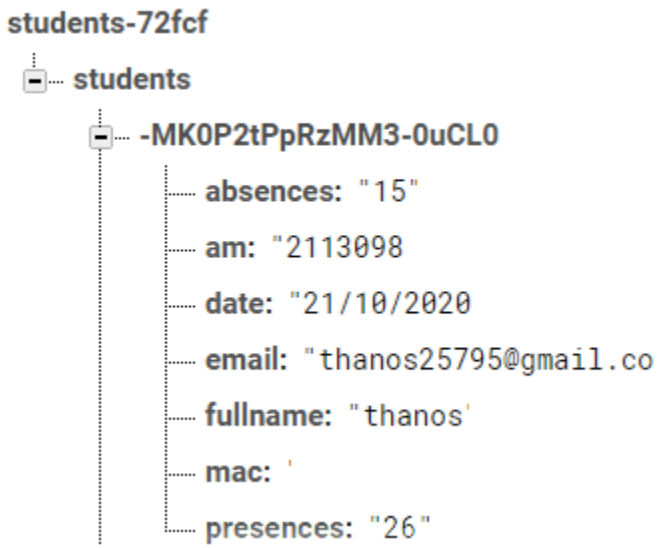
```java
package com.example.myapplication;

import ...

public class StudentDetailsActivity extends AppCompatActivity {

    private EditText mFullName_txt;
    private EditText mAm_txt;
    private EditText mEmail_txt;
    private EditText mPresences_txt;
    private EditText mAbsences_txt;
    private Button mUpdate_btn;
    private Button mDelete_btn;
    private Button mBack_btn;
    private String keys;
    private String fullname;
    private String am;
    private String email;
    private String presences;
    private String absences;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        keys=getIntent().getStringExtra( name: "key");
        fullname=getIntent().getStringExtra( name: "fullname");
        am=getIntent().getStringExtra( name: "am");
        email=getIntent().getStringExtra( name: "email");
        presences=getIntent().getStringExtra( name: "presences");
        absences=getIntent().getStringExtra( name: "absences");


        setContentView(R.layout.activity_student_details);
        mFullName_txt=(EditText) findViewById(R.id.fullname_editText);
        mFullName_txt.setText(fullname);
        mAm_txt=(EditText) findViewById(R.id.am_editTxt);
        mAm_txt.setText(am);
        mEmail_txt=(EditText) findViewById(R.id.email_edittxt);
        mEmail_txt.setText(email);
        mPresences_txt=(EditText) findViewById(R.id.presences_editTxt);
        mPresences_txt.setText(presences);
        mAbsences_txt=(EditText) findViewById(R.id.absences_editTxt);
        mAbsences_txt.setText(absences);
        mUpdate_btn=(Button) findViewById(R.id.update_btn);
```

Εικόνα 4

Αφού έχουμε δηλώσει τα στοιχεία που θα χρειαστούν στη συνέχεια πρέπει να περάσουμε τις πληροφορίες που μας δίνονται για κάθε μαθητή στις αντίστοιχες μεταβλητές έτσι ώστε η ενημέρωσή του να γίνεται σωστά. Τα στοιχεία που θα εμφανιστούν για κάθε μαθητή θα είναι από τη βάση δεδομένων που έχουμε δημιουργήσει. Στο κομμάτι που ακολουθεί λοιπόν ζητάμε από τη βάση σε μια συγκεκριμένη θέση που έχει με βάση το κλειδί, που είναι ξεχωριστό για κάθε μαθητή, να εμφανίσει τις πληροφορίες.

```
keys=getIntent().getStringExtra("key");
fullname=getIntent().getStringExtra("fullname");
am=getIntent().getStringExtra("am");
email=getIntent().getStringExtra("email");
presences=getIntent().getStringExtra("presences");
absences=getIntent().getStringExtra("absences");
```

Η βάση δεδομένων έχει την ακόλουθη μορφή :



Άρα με την αναζήτηση που θα κάνουμε στο κώδικα όταν βρούμε το μαθητή μέσω του κλειδιού θα μπούμε στα στοιχεία του και όταν υπάρξει αντιστοιχία στη λέξη κλειδί. Σαν αποτέλεσμα θα πάρουμε τη κατάλληλη πληροφορία. Για παράδειγμα στη πρώτη γραμμή ζητάμε το full name του μαθητή. Η λέξη κλειδί για αυτή την αναζήτηση είναι το fullname. Η κατηγορία αυτή υπάρχει μέσα σε κάθε φοιτητή επομένως οτιδήποτε υπάρχει μέσα στη γραμμή του fullname θα το επιστρέψει στη μεταβλητή του κώδικα που έχουμε ονομάσει fullname. Με τον ίδιο τρόπο μπορούμε να πάρουμε από τη βάση δεδομένων όποιο στοιχείο χρειαστούμε για να κάνουμε την ανάλογη λειτουργία. Στη συνέχεια υπάρχουν οι δυο συναρτήσεις για την υλοποίηση της ενημέρωσης και της διαγράφης ενός μαθητή από τη βάση δεδομένων. Αυτές οι συναρτήσεις υπάρχουν στην εικόνα 5.

```
mDelete_btn=(Button) findViewById(R.id.delete_btn);
mBack_btn=(Button) findViewById(R.id.back_btn);

mUpdate_btn.setOnClickListener((v) -> {
        Student student=new Student();
        student.setFullname(mFullName_txt.getText().toString());
        student.setAm(mAm_txt.getText().toString());
        student.setEmail(mEmail_txt.getText().toString());
        student.setPresences(mPresences_txt.getText().toString());
        student.setAbsences(mAbsences_txt.getText().toString());

        new FirebaseDatabaseHelper().updateStudent(keys, student, new FirebaseDatabaseHelper.DataStatus() {
            @Override
            public void DataIsLoaded(List<Student> students, List<String> keys) {

            }

            @Override
            public void DataIsInserted() {

            }

            @Override
            public void DataIsUpdated() {
                Toast.makeText(getApplicationContext(), text: "The student has been updated successfully",Toast.LENGTH_SHORT).show();
            }

            @Override
            public void DataIsDeleted(List<Student> students, List<String> keys) {

            }

            @Override
            public void DataIsDeleted() {

            }
        });
});
mDelete_btn.setOnClickListener((v) -> {
        new FirebaseDatabaseHelper().deleteStudent(keys, new FirebaseDatabaseHelper.DataStatus() {
            @Override
            public void DataIsLoaded(List<Student> students, List<String> keys) {
```

Εικόνα 5

Και στις δυο συναρτήσεις για να μπορέσουμε να περάσουμε τις αλλαγές στη βάση δεδομένων θα πρέπει να δηλώσουμε ένα αντικείμενο ιδίου τύπου με τη βάση. Στη πρώτη από τις συναρτήσεις περνάμε τις νέες τιμές στις μεταβλητές που αντιπροσωπεύουν το μαθητή και στη συνέχεια πρέπει να τις κάνουμε εισαγωγή μέσα στη βάση. Αυτό μπορεί να γίνει με τον εξής τρόπο:

```
new FirebaseDatabaseHelper().updateStudent(keys, student, new FirebaseDatabaseHelper.DataStatus()
```

δημιουργούμε ένα νέο αντικείμενο FirebaseDatabaseHelper() που στην ουσία είναι η κλάση που υποστηρίζει όλες τις λειτουργίες της βάσης δεδομένων και θα εξηγηθεί περεταίρω στη συνέχεια του κεφαλαίου. Το updateStudent είναι μια συνάρτηση της FirebaseDatabaseHelper() που όταν βάζουμε τα κατάλληλα στοιχεία σαν ορίσματα κάνει ενημέρωση της βάσης δεδομένων στην θέση που υποδεικνύει το κλειδί (η μεταβλητή keys). Αφού ολοκληρωθεί η ενημέρωση του μαθητή εμφανίζει ένα αντιπροσωπευτικό μήνυμα στην οθόνη του χρήστη.

```
                }

                @Override
                public void DataIsInserted() {

                }

                @Override
                public void DataIsUpdated() {

                }

                @Override
                public void DataIsDeleted(List<Student> students, List<String> keys) {

                }

                @Override
                public void DataIsDeleted() {
                    Toast.makeText(getApplicationContext(), text: "The student has been deleted successfully",Toast.LENGTH_SHORT).show();
                    finish();return;
                }
            });
            mBack_btn.setOnClickListener((v) -> { finish();return; });
        });
    }


}
```

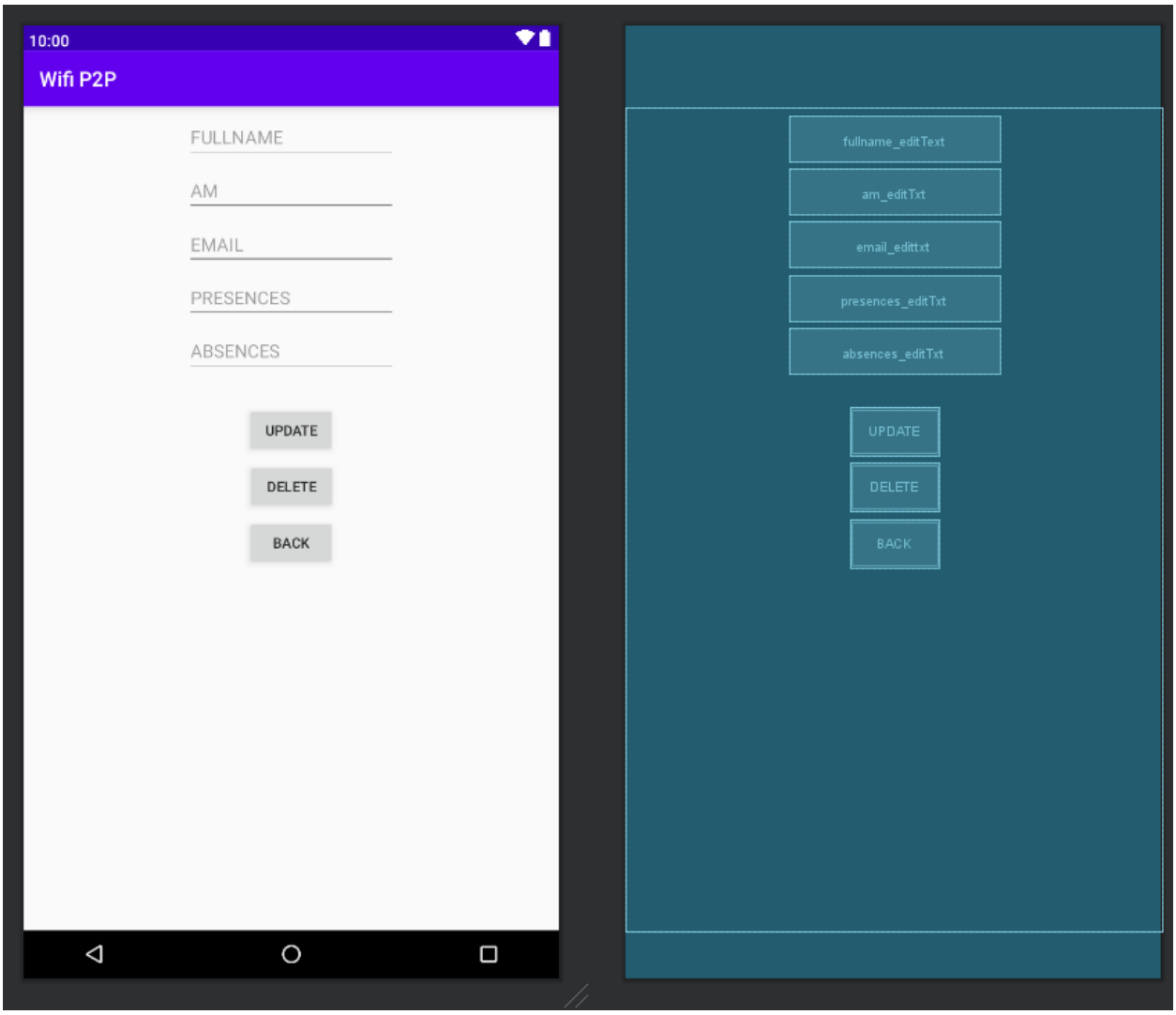
Εικόνα 6

Αντίστοιχα και στη συνάρτηση της διαγράφης, που περιγράφεται στις εικόνες 5 και 6 δημιουργείτε ένα νέο αντικείμενο της κλάσης FirebaseDatabaseHelper(). Η κλάση αυτή έχει συνάρτηση για διαγραφή ενός χρήστη από τη βάση δεδομένων, συνεπώς περνώντας τα κατάλληλα ορίσματα στη συνάρτηση

```
new FirebaseDatabaseHelper().deleteStudent(keys, new FirebaseDatabaseHelper.DataStatus()
```

μπορούμε να διαγράψουμε ένα μαθητή από τη βάση και στη συνέχεια θα εμφανιστεί ξανά ένα μήνυμα αντίστοιχο με τη λειτουργία που εκτελέστηκε και θα αναφέρει ότι ο μαθητής έχει διαγραφεί επιτυχώς.

Τέλος στην εικόνα 7 υπάρχει το user interface που θα βλέπει ο χρήστης στην εφαρμογή από τη σελίδα του κώδικα.

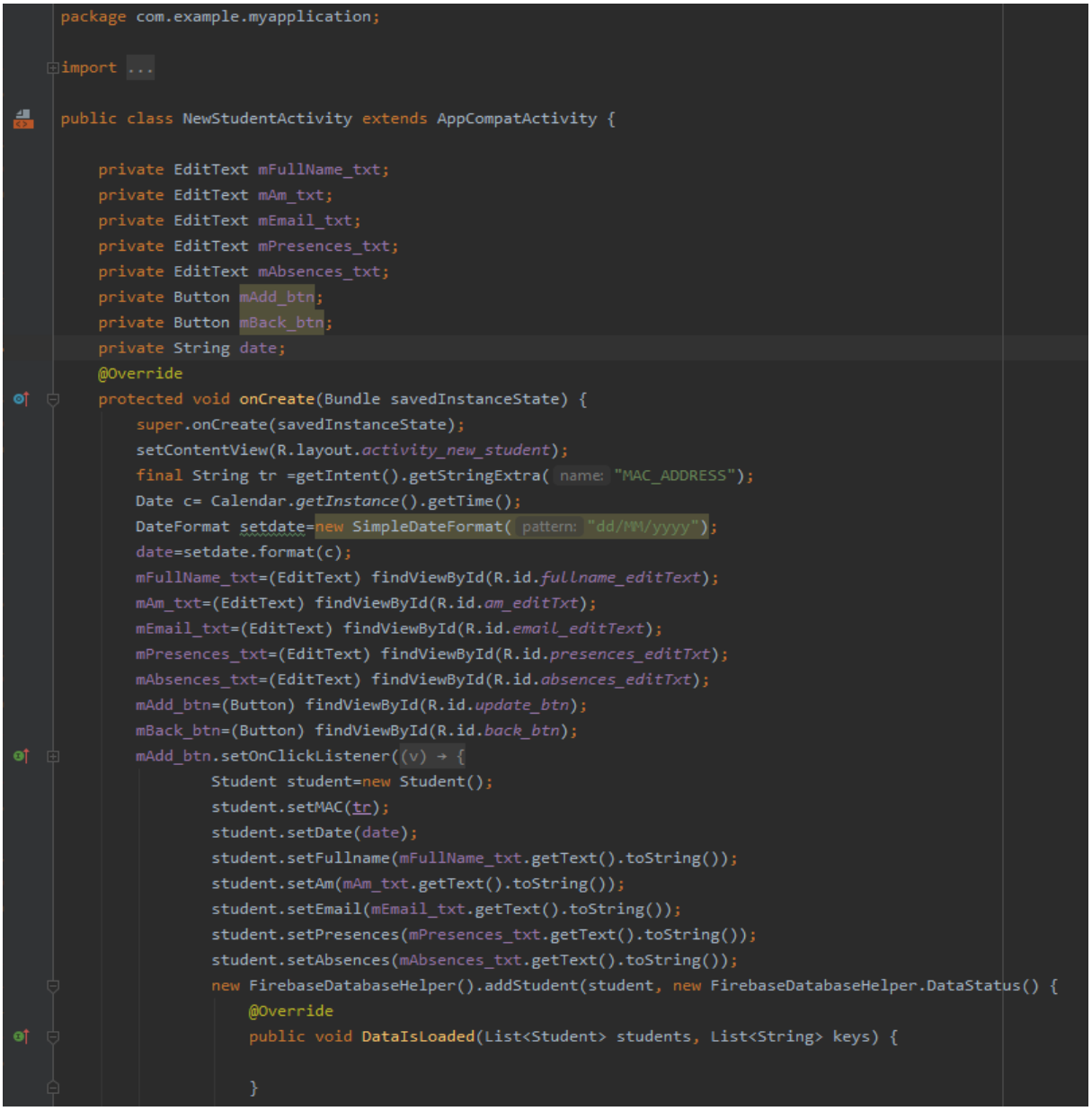


Εικόνα 7

NEW STUDENT ACTIVITY

Αυτή η κλάση έχει να κάνει με τη δημιουργία και την εισαγωγή ενός νέου μαθητή στην υπάρχουσα βάση δεδομένων. Σε αυτή τη κλάση έχουν πρόσβαση και οι δυο πλευρές, καθηγητής και μαθητής. Άρα και οι δυο πλευρές έχουν τη δυνατότητα να προσθέσουν ένα νέο μαθητή στη βάση. Όπως και στις προηγούμενες σελίδες του κώδικα έτσι και σε αυτή θα πρέπει πρώτα να δηλώσουμε και να δημιουργήσουμε το κατάλληλο user interface(UI) για να μπορούν οι χρήστες να προσθέσουν ένα μαθητή. Τα πεδία που θα πρέπει να έχει αυτό το UI θα πρέπει να ταυτίζονται με τα ήδη υπάρχοντα στη βάση δεδομένων για να γίνει σωστή αντιστοίχιση και να μπορέσουν να αποθηκευτούν. Ο χρήστης θα συμπληρώσει τα στοιχεία του στα πλαίσια κειμένου και στη συνέχεια έχουμε μια συνάρτηση που με το πάτημα του κουμπιού add από το UI ενεργοποιείται. Το πρώτο που γίνεται όταν μπαίνουμε στη συνάρτηση είναι η δημιουργία ενός αντικειμένου της κλάσης Student, που έχουμε αναφέρει πιο πάνω,

έτσι ώστε ο νέος μαθητής να έχει όλες τις πληροφορίες. Οι συναρτήσεις που υπάρχουν στη κλάση Student θα μας βοηθήσουν να περάσουμε τα στοιχεία του μαθητή από το UI στη βάση δεδομένων.



```
package com.example.myapplication;

import ...

public class NewStudentActivity extends AppCompatActivity {

    private EditText mFullName_txt;
    private EditText mAm_txt;
    private EditText mEmail_txt;
    private EditText mPresences_txt;
    private EditText mAbsences_txt;
    private Button mAdd_btn;
    private Button mBack_btn;
    private String date;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_new_student);
        final String tr =getIntent().getStringExtra( name: "MAC_ADDRESS");
        Date c= Calendar.getInstance().getTime();
        DateFormat setdate=new SimpleDateFormat( pattern: "dd/MM/yyyy");
        date=setdate.format(c);
        mFullName_txt=(EditText) findViewById(R.id.fullname_editText);
        mAm_txt=(EditText) findViewById(R.id.am_editTxt);
        mEmail_txt=(EditText) findViewById(R.id.email_editText);
        mPresences_txt=(EditText) findViewById(R.id.presences_editTxt);
        mAbsences_txt=(EditText) findViewById(R.id.absences_editTxt);
        mAdd_btn=(Button) findViewById(R.id.update_btn);
        mBack_btn=(Button) findViewById(R.id.back_btn);
        mAdd_btn.setOnClickListener((v) -> {
                Student student=new Student();
                student.setMAC(tr);
                student.setDate(date);
                student.setFullname(mFullName_txt.getText().toString());
                student.setAm(mAm_txt.getText().toString());
                student.setEmail(mEmail_txt.getText().toString());
                student.setPresences(mPresences_txt.getText().toString());
                student.setAbsences(mAbsences_txt.getText().toString());
                new FirebaseDatabaseHelper().addStudent(student, new FirebaseDatabaseHelper.DataStatus() {
                    @Override
                    public void DataIsLoaded(List<Student> students, List<String> keys) {

                    }
```

Εικόνα 8

Εδώ πρέπει να αναφέρουμε ότι πέρα από τα στοιχεία που θα περάσει ένας μαθητής στη βάση αποθηκεύουμε και κάποια επιπλέον για την εφαρμογή των απουσιών (ημερομηνία σύνδεσης) του και τη μοναδικότητα του χρήστη (mac address). Αυτές οι πληροφορίες δημιουργούνται σε μια άλλη κλάση που θα αναπτύξουμε στη συνέχεια (Main Activity) από την οποία τις διοχετεύουμε μέσω της λειτουργίας του Intent. Είναι σαν ένα νήμα να ενώνει αυτές τις δυο σελίδες του κώδικα για να μπορέσουμε να περάσουμε τη πληροφορία από τη μια στην άλλη. Το επόμενο βήμα μετά τη δημιουργία του μαθητή είναι η προσθήκη του μέσα στη βάση δεδομένων. Για αυτό δημιουργούμε ένα αντικείμενο της κλάσης

της βάσης δεδομένων (FirebaseDatabaseHelper) η οποία διαθέτει τη συνάρτηση προσθήκης ενός μαθητή:

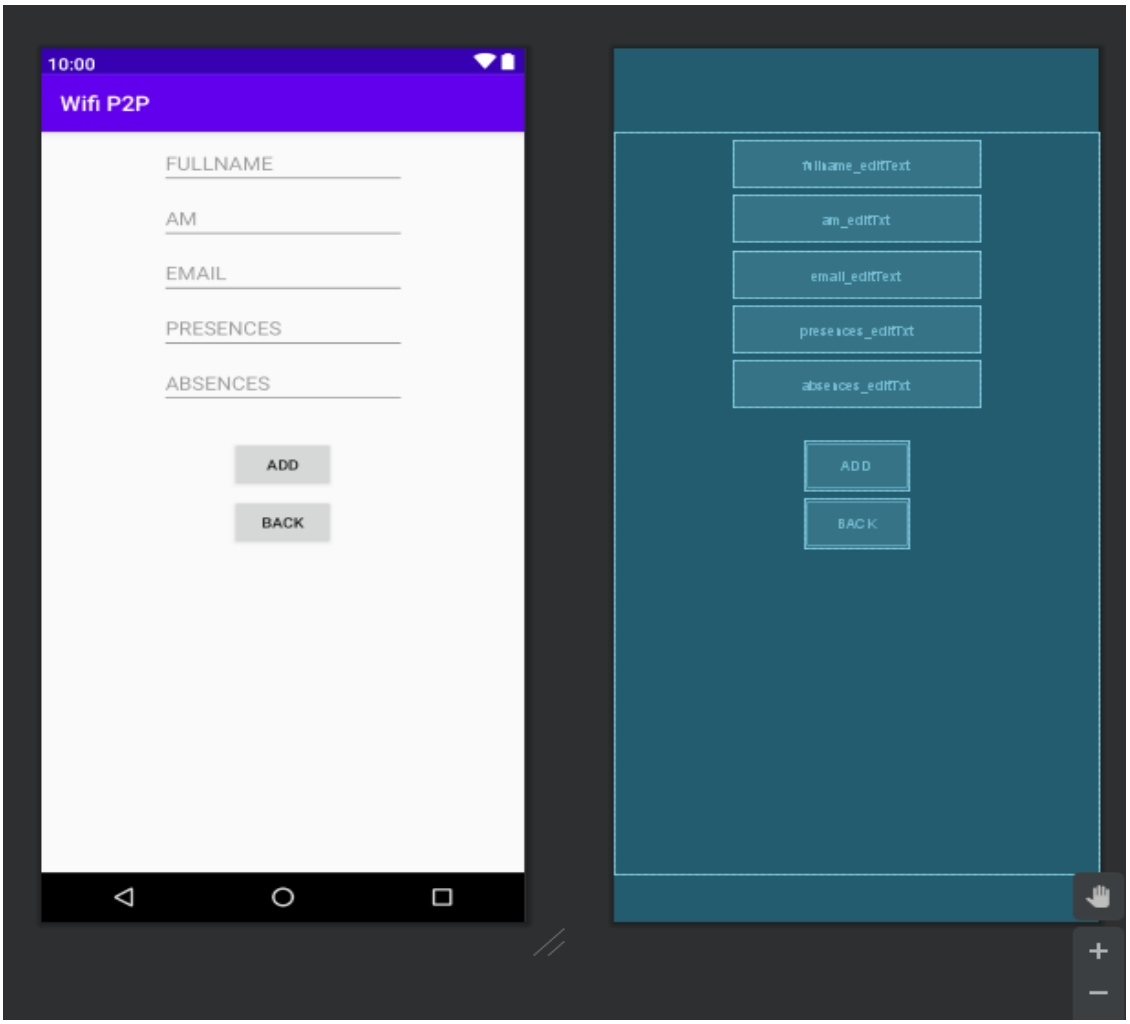
```
newFirebaseDatabaseHelper().addStudent(student,newFirebaseDatabaseHelper.DataStatus()
```

που έχει σαν ορίσματα το μαθητή που δημιουργήσαμε, και την έναρξη της βάσης δεδομένων για την εισαγωγή του. Ανάλογα με τα άτομα που διαθέτη η βάση θα δώσει και το κατάλληλο κλειδί για τη θέση του μαθητή έτσι ώστε την επόμενη φορά που θα χρειαστεί να εμφανιστεί να μπορέσουμε να τον βρούμε μέσω του κλειδιού για παράδειγμα αν υπάρχουν τρεις μαθητές στη βάση ο νέος μαθητής θα εισαχθεί στη θέση τέσσερα και το κλειδί του θα είναι αυτό. Μετά την εισαγωγή του μαθητή στη βάση δεδομένων θα εμφανιστεί κατάλληλο μήνυμα για την επιτυχία της εκτέλεσης και θα κλείσει η συνάρτηση. Για να επιστρέψουμε στο αρχικό UI έχουμε προσθέσει ένα ακόμα κουμπί back που ενεργοποιεί μια συνάρτηση η οποία τερματίζει το UI που βρισκόμαστε για τη προσθήκη του χρήστη και μας γυρίζει στο προηγούμενο.

```
                @Override
                public void DataIsInserted() {
                    Toast.makeText(getApplicationContext(), text: "The student has been inserted successfully",Toast.LENGTH_SHORT).show();
                }

                @Override
                public void DataIsUpdated() {

                }

                @Override
                public void DataIsDeleted(List<Student> students, List<String> keys) {

                }

                @Override
                public void DataIsDeleted() {

                }
            });
        });
        mBack_btn.setOnClickListener((v) -> {
            finish();
            return;
        });
    }
}
```


Εικόνα 9

Η εικόνα 10 αντιπροσωπεύει το user interface που θα βλέπει ο χρήστης από αυτή τη σελίδα του κώδικα όταν θα χρησιμοποιεί την εφαρμογή.

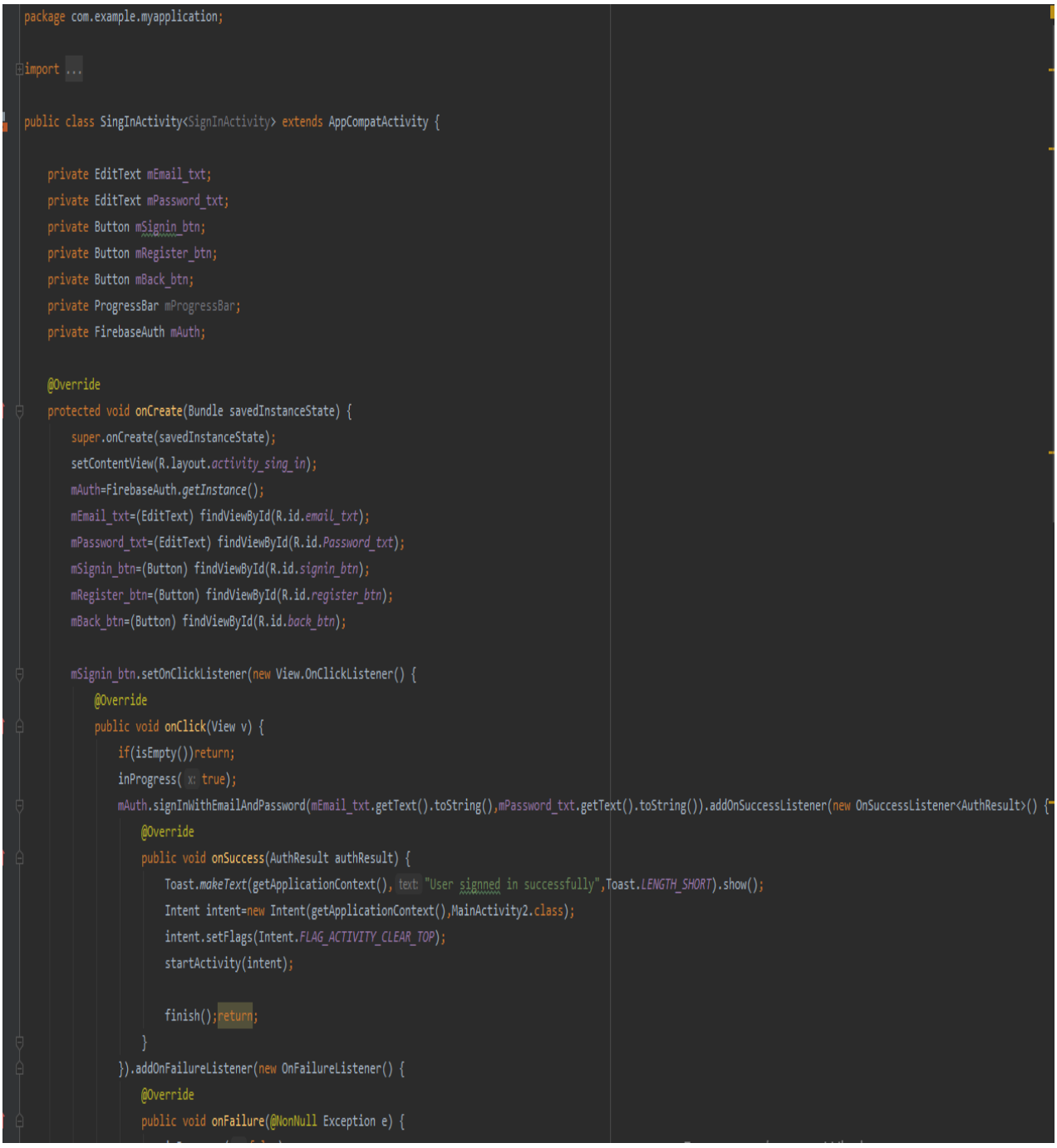


Εικόνα 10

SING IN ACTIVITY

Σε αυτή τη σελίδα του κώδικα έχει υλοποιηθεί ο τρόπος σύνδεσης του καθηγητή μέσα στη βάση δεδομένων για να μπορεί να διαχειριστεί τα στοιχεία της βάσης δεδομένων, για τη διόρθωση τυχών λαθών, τη προσθήκη ή τη διαγραφή μαθητών κλπ. Όπως και στις προηγούμενες περιπτώσεις του κώδικα

στην αρχή έχουμε τις δηλώσεις των button και textEditor αντικειμένων μου θα χρειαστούν στο UI. Στη συνέχεια ακολουθεί η συνάρτηση του button sing in.



```
package com.example.myapplication;

import ...

public class SingInActivity<SignInActivity> extends AppCompatActivity {

    private EditText mEmail_txt;
    private EditText mPassword_txt;
    private Button mSignin_btn;
    private Button mRegister_btn;
    private Button mBack_btn;
    private ProgressBar mProgressBar;
    private FirebaseAuth mAuth;

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_sing_in);
        mAuth=FirebaseAuth.getInstance();
        mEmail_txt=(EditText) findViewById(R.id.email_txt);
        mPassword_txt=(EditText) findViewById(R.id.Password_txt);
        mSignin_btn=(Button) findViewById(R.id.signin_btn);
        mRegister_btn=(Button) findViewById(R.id.register_btn);
        mBack_btn=(Button) findViewById(R.id.back_btn);

        mSignin_btn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(isEmpty())return;
                inProgress( x: true);
                mAuth.signInWithEmailAndPassword(mEmail_txt.getText().toString(),mPassword_txt.getText().toString()).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                    @Override
                    public void onSuccess(AuthResult authResult) {
                        Toast.makeText(getApplicationContext(), text: "User signned in successfully",Toast.LENGTH_SHORT).show();
                        Intent intent=new Intent(getApplicationContext(),MainActivity2.class);
                        intent.setFlags(Intent.FLAG_ACTIVITY_CLEAR_TOP);
                        startActivity(intent);

                        finish();return;
                    }
                }).addOnFailureListener(new OnFailureListener() {
                    @Override
                    public void onFailure(@NonNull Exception e) {
```

Εικόνα 11

Μόλις πατήσει το κουμπί ο χρήστης για την είσοδο στο λογαριασμό του θα πρέπει να δούμε αρχικά αν έχει εισάγει τα στοιχεία του η όχι. Αυτό γίνεται μέσω της συνάρτησης isEmpty(), που υπάρχει στην εικόνα 13, η οποία ελέγχει αν τα πλαίσια κειμένου έχουν συμπληρωθεί, ή ο χρήστης πάτησε το κουμπί της συνδέσεις πριν εισάγει τα στοιχεία του. Αν δεν έχει βάλει τα στοιχεία του εμφανίζεται σχετικό μήνυμα στην οθόνη που αναφέρει ότι είναι αναγκαία η συμπλήρωση τους. Αντιθέτως αν έχει βάλει τα στοιχεία του και πατήσει για σύνδεση τότε γίνεται έλεγχος αν υπάρχει ο κωδικός και το email που έχει χρησιμοποιήσει. Σε περίπτωση επιτυχίας εμφανίζει κατάλληλο μήνυμα στην οθόνη και οδηγεί το χρήστη στην οθόνη της βάσης δεδομένων με τη δυνατότητα πλέον διαχείρισης. Σε περίπτωση αποτυχίας της αντιστοίχισης των στοιχείων του εμφανίζεται μήνυμα αποτυχίας, δηλαδή είτε έχει εισάγει λάθος στοιχεία είτε δεν υπάρχει. Στην περίπτωση που ο χρήστης δεν υπάρχει, θα πρέπει να κάνει πρώτα εγγραφή και στη συνέχεια είσοδο.

```
                    inProgress( x: false);
                        Toast.makeText(getApplicationContext(), text: "Signned in failed",Toast.LENGTH_SHORT).show();
                }
            });
        }
    });
    mRegister_btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            if(isEmpty())return;
            inProgress( x: true);
            mAuth.createUserWithEmailAndPassword(mEmail_txt.getText().toString(),mPassword_txt.getText().toString()).addOnSuccessListener(new OnSuccessListener<AuthResult>() {
                @Override
                public void onSuccess(AuthResult authResult) {
                    Toast.makeText(getApplicationContext(), text: "User registered successfully",Toast.LENGTH_SHORT).show();
                    inProgress( x: false);
                }
            }).addOnFailureListener(new OnFailureListener() {
                @Override
                public void onFailure(@NonNull Exception e) {
                    inProgress( x: false);
                    Toast.makeText(getApplicationContext(), text: "Registration failed",Toast.LENGTH_SHORT).show();
                }
            });
        }
    });
    mBack_btn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View v) {
            finish();return;
        }
    });
}
private void inProgress(boolean x){
    if(x){
        mBack_btn.setEnabled(false);
        mSignin_btn.setEnabled(false);
        mRegister_btn.setEnabled(false);
    }else{
        mBack_btn.setEnabled(true);
        mSignin_btn.setEnabled(true);
        mRegister_btn.setEnabled(true);
```

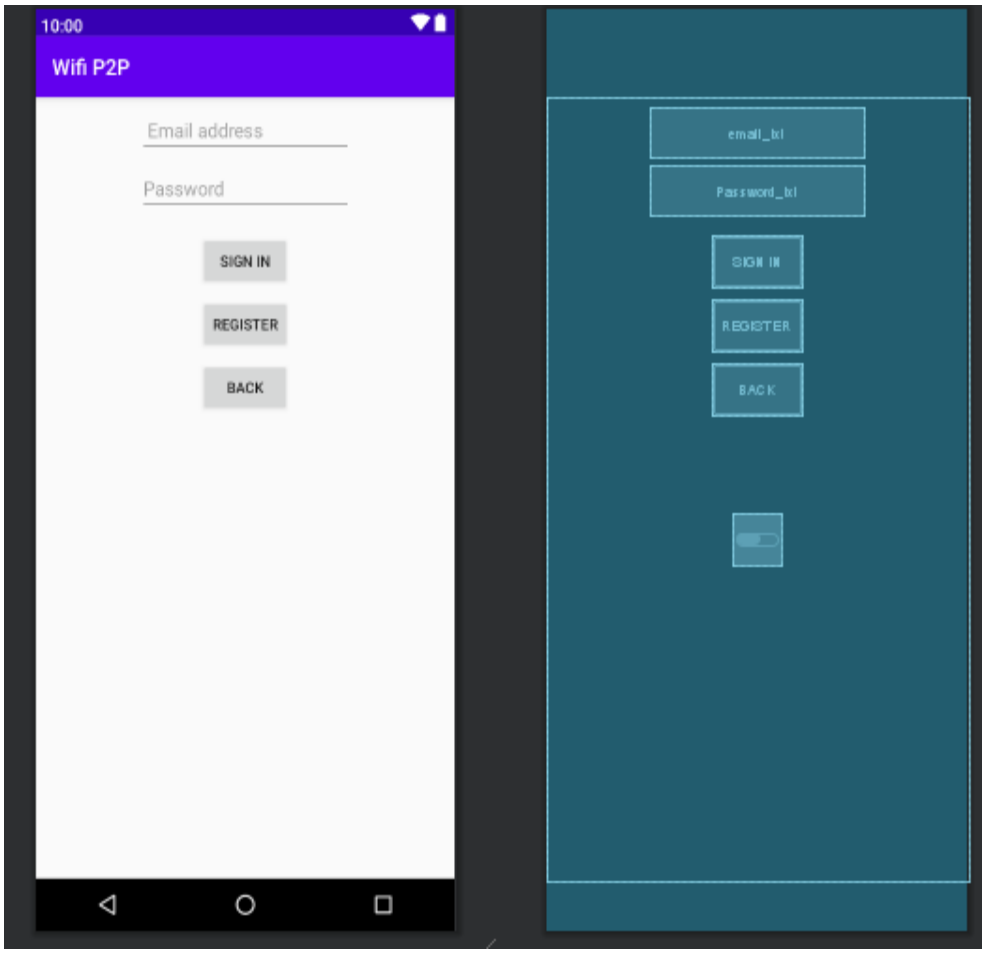
Εικόνα 12

Αφού εισάγει τα στοιχεία του λοιπόν θα πρέπει να προχωρήσει στο κομμάτι του register πατώντας το κουμπί, έτσι ώστε να περαστούν τα στοιχεία του και να μπορεί να συνδέεται με το email και το κωδικό του. Στη συνάρτηση που αντιστοιχεί στο κουμπί του register περνάμε τα στοιχεία από τα δυο πλαίσια κειμένου και μόλις κάνει εγγραφή τα στοιχεία περνούν σε ένα ξεχωριστό σημείο της βάσης δεδομένων, διότι ο καθηγητής δε θα πρέπει να εμφανίζεται στην ίδια σελίδα της βάσης με τους μαθητές. Αν γίνει επιτυχώς η εγγραφή στην οθόνη εμφανίζεται μήνυμα επιτυχίας. Σε αντίθετη περίπτωση εμφανίζεται μήνυμα αποτυχίας και θα πρέπει να επαναληφθεί η διαδικασία. Αυτό το τμήμα του κώδικα υπάρχει στην εικόνα 12.

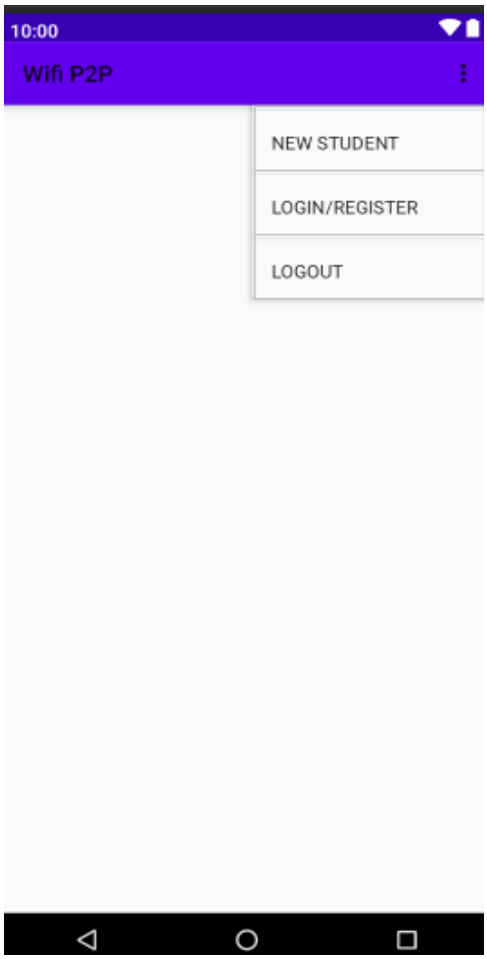
```
        }
    }
    private boolean isEmpty(){
        if(TextUtils.isEmpty(mEmail_txt.getText().toString())){
            mEmail_txt.setError("REQUIRED");
            return true;
        }
        if(TextUtils.isEmpty(mPassword_txt.getText().toString())){
            mPassword_txt.setError("REQUIRED");
            return true;
        }
        return false;
    }
}
```

Εικόνα 13

Από τη πλευρά του χρήστη τώρα θα πρέπει να υπάρχουν τα κατάλληλα UI για να γίνεται αυτή η διαδικασία πιο γρήγορη και εύκολη. Για τη σωστή υποστήριξη όλων των προηγουμένων θα χρειαστούν δυο τέτοια UI. Το ένα θα είναι υπεύθυνο για την εγγραφή η τη σύνδεση του χρήστη και το άλλο θα είναι υπεύθυνο για την είσοδο και την έξοδο από το λογαριασμό του καθώς και τη προσθήκη μαθητή αν αυτό είναι επιθυμητό. Τα user interfaces που έχουν χρησιμοποιηθεί είναι τα ακόλουθα. Όπως φαίνεται και πιο κάτω στην εικόνα 14 το αριστερό UI είναι αυτό που θα μας επιτρέπει να κάνουμε σύνδεση και εγγραφή ενός χρήστη στη βάση ενώ αυτό στα δεξιά στην εικόνα 15 θα είναι αυτό που θα μας παρέχει αυτή τη δυνατότητα. Μόλις ο καθηγητής ανοίξει τη βάση δε θα έχει δικαιώματα διαχείρισης. Θα πρέπει να κάνει είσοδο στο λογαριασμό του χρησιμοποιώντας και τα δυο UI πρώτα αυτό στα δεξιά που θα εμφανιστεί πατώντας τις τρεις τελείες στο πάνω δεξιά μέρος της οθόνης της εφαρμογής και στη συνέχεια θα μεταφερθεί στο αριστερό UI. Αν ο χρήστης θελήσει να προσθέσει ένα χρήστη η να αποσυνδεθεί θα χρησιμοποιήσει ξανά το δεύτερο UI. Με την επιλογή της προσθήκης θα μεταφερθεί στο UI για την εγγραφή νέου μαθητή που αναπτύξαμε προηγουμένως.



Εικόνα 15



Εικόνα 14

WIFI DIRECT BROADCAST RECEIVER

Σε αυτή τη σελίδα της εφαρμογής έχει αναπτυχθεί ένας μηχανισμός-βοηθός για τη δημιουργία και τη σύνδεση μεταξύ δυο συσκευών, που βοηθά στη σωστή λειτουργία της κυρίας σελίδας του κώδικα. Στην ουσία το κομμάτι του κώδικα αυτού είναι υπεύθυνο να μπορεί να δημιουργήσει ένα αντικείμενο το οποίο θα παίρνει τις τιμές μέσω των ορισμάτων που δίνουμε στο constructor και ανάλογα με αυτές όταν υπάρχει μια αλλαγή στη κατάσταση του Wi-Fi από ενεργό σε ανενεργό ή υπάρχει αλλαγή στο κανάλι που συνδέει τις δύο συσκευές να εκτυπώνει τα κατάλληλα μηνύματα, να κάνει έναρξη της αναζήτησης συσκευών, ή να κάνει το έτοιμα σύνδεσης της συσκευής με μια άλλη. Στο πρώτο if που υπάρχει ο έλεγχος για την αλλαγή της κατάστασης του Wi-Fi p2p, όταν στην εφαρμογή πατήσουμε το κουμπί για το Wi-Fi αυτόματα θα μπει μέσα στο πρώτο if και αν το Wi-Fi ήταν κλειστό θα εμφανίσει μήνυμα ενεργοποίησης, αλλιώς αν ήταν ήδη ανοιχτό και το κλείσαμε θα εμφανίσει μήνυμα απενεργοποίησης του Wi-Fi.

Αν πατήσουμε το κουμπί discover από την εφαρμογή αυτή θα αρχίσει να κάνει αναζήτηση για άλλες συσκευές (peers) με τους οποίους μπορεί να κάνει σύνδεση. Άρα θα είμαστε στο δεύτερο έλεγχο και ο broadcast receiver θα αρχίσει να δημιουργεί μια λίστα με τα ονόματα των συσκευών που βρίσκει

διαθέσιμα. Αν υπάρξει αλλαγή στη κατάσταση της σύνδεσης μεταξύ των δύο συσκευών τότε θα είμαστε στο τρίτο έλεγχο που θα μας ενημερώνει σχετικά με το αν έγινε με επιτυχία η σύνδεση μεταξύ των δυο συσκευών, καθώς και το όνομα της συσκευής με την οποία έχουμε συνδεθεί, ενώ αν γίνει αποσύνδεση η αποτυχία θα εμφανιστεί στην οθόνη κατάλληλο μήνυμα.

```java
package com.example.myapplication;
import ...
public class WifiDirectBroadcastReceiver extends BroadcastReceiver {
    private WifiP2pManager mManager;
    private WifiP2pManager.Channel mChannel;
    private MainActivity mActivity;

    public WifiDirectBroadcastReceiver(WifiP2pManager mManager,WifiP2pManager.Channel mChannel,MainActivity mActivity){
        this.mManager= mManager;
        this.mChannel= mChannel;
        this.mActivity= mActivity; }
    @SuppressLint("MissingPermission")
    @Override
    public void onReceive(Context context, Intent intent) {
        String action= intent.getAction();

        if(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION.equals(action)){
            int state=intent.getIntExtra(WifiP2pManager.EXTRA_WIFI_STATE, defaultValue: -1);

            if(state==WifiP2pManager.WIFI_P2P_STATE_ENABLED){
                Toast.makeText(context, text: "Wifi is ON",Toast.LENGTH_SHORT).show();
            }else{
                Toast.makeText(context, text: "Wifi is OFF",Toast.LENGTH_SHORT).show();
            }
        }else if(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION.equals(action)){
            if(mManager!=null){
                mManager.requestPeers(mChannel,mActivity.peerListListener);
            }
        }else if(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION.equals(action)){
            if(mManager==null){
                return;
            }
            NetworkInfo networkInfo=intent.getParcelableExtra(WifiP2pManager.EXTRA_NETWORK_INFO);
            if(networkInfo.isConnected()){
                mManager.requestConnectionInfo(mChannel,mActivity.connectionInfoListener);
            }else{
                mActivity.connectionStatus.setText("Device Disconnected");
            }
        }else if(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION.equals(action)){

        }
    }
}
```

Εικόνα 16

RECYCLER VIEW CONFIG

Στη κλάση reclyclerView γίνεται η ενημέρωση και η προβολή νέων στοιχείων της βάση δεδομένων στην εφαρμογή από τη πλευρά του καθηγητή καθώς μόνο αυτός έχει πρόσβαση σε αυτή και μπορεί να δει τα στοιχεία και τις πληροφορίες της βάσης. Για να γίνει η προβολή των μαθητών σωστά έχουμε τη συνάρτηση setConfig και με τα ορίσματα που περνάμε σε αυτή δημιουργείται μια λίστα που περιέχει όλους τους μαθητές καθώς και τα κλειδιά των μαθητών.

```java
package com.example.myapplication;

import android.content.Context;
import android.content.Intent;
import android.view.LayoutInflater;
import android.view.View;
import android.view.ViewGroup;
import android.widget.TextView;

import androidx.annotation.NonNull;
import androidx.recyclerview.widget.LinearLayoutManager;
import androidx.recyclerview.widget.RecyclerView;

import com.google.firebase.auth.FirebaseAuth;
import com.google.firebase.auth.FirebaseUser;

import java.util.List;

public class RecyclerView_Config {
    FirebaseAuth mAuth;
    private static FirebaseUser user;
    private Context mContext;
    private StudentsAdapter mStudentsAdapter;
    public void setConfig(RecyclerView recyclerView,Context context,List<Student> students,List<String> keys)
        mAuth=FirebaseAuth.getInstance();
        user =mAuth.getCurrentUser();
        mContext=context;
        mStudentsAdapter=new StudentsAdapter(students,keys);
        recyclerView.setLayoutManager(new LinearLayoutManager(context));
        recyclerView.setAdapter(mStudentsAdapter);

    }

    class StudentItemView extends RecyclerView.ViewHolder {
        private TextView mFullname;
        private TextView mAm;
        public TextView mMAC;
        private TextView mEmail;
        private TextView mDate;
        private TextView mAbsence;
        private TextView mPresence;
        private String keys;
        public StudentItemView(ViewGroup parent){
```

Εικόνα 17

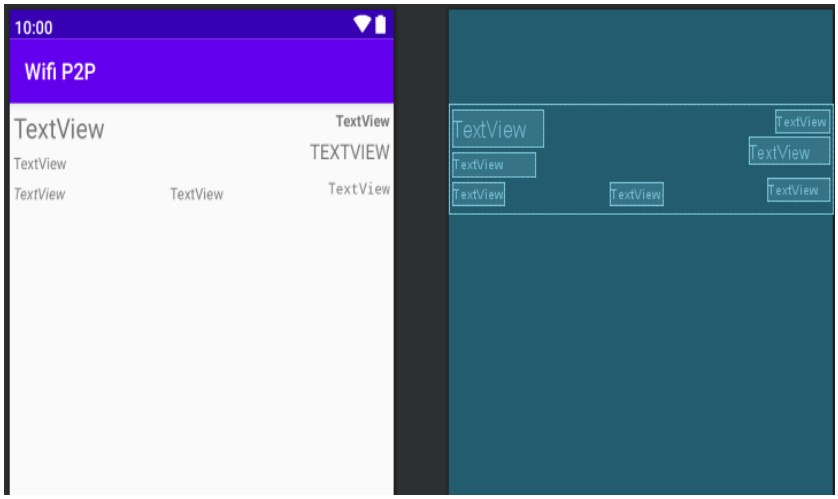
```
        super(LayoutInflater.from(mContext).
        inflate(R.layout.student_list_item,parent, attachToRoot: false));
        mDate=(TextView) itemView.findViewById(R.id.date_textView);
        mFullname=(TextView) itemView.findViewById(R.id.Student_txtView);
        mEmail=(TextView) itemView.findViewById(R.id.email_txt);
        mAm=(TextView) itemView.findViewById(R.id.am_txtView);
        mMAC=(TextView) itemView.findViewById(R.id.mac_textView);
        mPresence=(TextView) itemView.findViewById(R.id.presences_txtView);
        mAbsence=(TextView) itemView.findViewById(R.id.absences_txtView);

        itemView.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View v) {
                if(user!=null) {
                    Intent intent = new Intent(mContext, StudentDetailsActivity.class);
                    intent.putExtra( name: "key", keys);
                    intent.putExtra( name: "fullname", mFullname.getText().toString());
                    intent.putExtra( name: "am", mAm.getText().toString());
                    intent.putExtra( name: "mac", mMAC.getText().toString());
                    intent.putExtra( name: "date", mDate.getText().toString());
                    intent.putExtra( name: "email", mEmail.getText().toString());
                    intent.putExtra( name: "presences", mPresence.getText().toString());
                    intent.putExtra( name: "absences", mAbsence.getText().toString());

                    mContext.startActivity(intent);
                }else{
                    mContext.startActivity(new Intent(mContext,SingInActivity.class));
                }
            }
        });
    }
    public void bind(Student student,String key){
        mFullname.setText(student.getFullname());
        mAm.setText(student.getAm());
        mMAC.setText(student.getMAC());
        mDate.setText(student.getDate());
        mEmail.setText(student.getEmail());
        mPresence.setText(student.getPresences());
        mAbsence.setText(student.getAbsences());
        this.keys=key;
    }

}
```

Εικόνα 18

```
class StudentsAdapter extends RecyclerView.Adapter<StudentItemView>{
    private List<Student> mStudentList;
    private List<String> mKeys;

    public StudentsAdapter(List<Student> mStudentList, List<String> mKeys) {
        this.mStudentList = mStudentList;
        this.mKeys = mKeys;
    }


    @NonNull
    @Override
    public StudentItemView onCreateViewHolder(@NonNull ViewGroup parent, int viewType) {
        return new StudentItemView(parent);
    }

    @Override
    public void onBindViewHolder(@NonNull StudentItemView holder, int position) {
        holder.bind(mStudentList.get(position),mKeys.get(position));
    }
    @Override
    public int getItemCount() { return mStudentList.size(); }
}
public static void longout() { user=null; }

}
```

Εικόνα 19

Επιπλέον θα χρειαστεί να δημιουργήσουμε ένα UI για να μπορέσουμε για κάθε μαθητή να εμφανίζουμε τις απαραίτητες πληροφορίες που θέλουμε, έτσι ώστε στη συνέχεια να δημιουργηθεί μια λίστα με όλους τους μαθητές και τα στοιχεία του καθενός. Όπως φαίνεται και στην εικόνα 20 στο κάθε μαθητή θα αντιστοιχεί ένα block. Μέσα σε αυτό θα υπάρχουν όλα τα στοιχεία που έχουν περαστεί στη βάση δεδομένων για αυτό το μαθητή. Για να μπορέσουμε να τραβήξουμε τα στοιχεία από τη βάση, μέσα στη συνάρτηση που έχουμε από τη πλευρά του καθηγητή, μόλις πατήσει το κουμπί για προβολή της βάσης δεδομένων, μέσω της intent και με τη κλάση StudentDetailsActivity, έχουμε πρόσβαση στη βάση δεδομένων και με βάση τις λέξεις κλειδιά που έχουμε περάσει θα μπορέσουμε να ανακτήσουμε τα στοιχεία του κάθε μαθητή. Για παράδειγμα για ένα μαθητή με τη λέξη κλειδί



Εικόνα 20

intent.putExtra("fullname", mFullname.getText().toString()); μπορούμε να πάρουμε από τη βάση το όνομά του και να το περάσουμε σαν string σε ένα πλαίσιο κειμένου, εικόνα 19. Στην ουσία η συνάρτηση intent είναι ένας δίαυλος επικοινωνίας μεταξύ δυο αντικειμένων, είτε αυτά είναι δυο διαφορετικές κλάσεις είτε αυτά είναι μια βάση δεδομένων και μια σελίδα κώδικα. Δημιουργεί μια σύνδεση για να μπορέσει να μεταφέρει πληροφορίες από το ένα σημείο στο άλλο. Σε συνδυασμό λοιπόν με τις λέξεις κλειδιά της βάσης δεδομένων τραβάμε τα στοιχεία για κάθε μαθητή και τα τοποθετούμε σε πλαίσια κειμένου δημιουργώντας έτσι ένα block σαν μια μικρή καρτέλα για κάθε μαθητή. Με το mFullname.getText().toString() εισάγουμε τη πληροφορία από τη βάση ενώ με το mFullname.setText(student.getFullname()); αναθέτουμε τη πληροφορία σε ένα Textview όπως φαίνεται και στην εικόνα πιο πάνω. Όλα τα block που έχουν δημιουργηθεί θα εμφανίζονται σε μορφή λίστας μέσω της κλάσης StudentsAdapter που είναι τύπου RecyclerView.Adapter<StudentItemView>.

FIREBASE DATABASE

Η κλάση αυτή περιέχει το κώδικα που αντιπροσωπεύει τη βάση δεδομένων στην εφαρμογή. Έχει όλες τις συναρτήσεις και τις βασικές λειτουργίες που χρειάζονται για να μπορέσει η εφαρμογή να δημιουργεί τη βάση δεδομένων καθώς και να διαχειρίζεται τα στοιχεία της. Το interface datastatus περιέχει τις συναρτήσεις που θα χρειαστούμε για να κάνουμε απαραίτητες ενέργειες όπως εισαγωγή μαθητή (data insert),ενημέρωση, διαγραφή και φόρτωση της λίστας. Η συνάρτηση firebase database helper υπάρχει για να κάνει τον έλεγχο ύπαρξης της λίστας, δηλαδή στο χώρο που μας δίνει η firebase πηγαίνει και δημιουργεί σύνδεση με τη βάση του ονόματος που της έχουμε δώσει σαν λέξη κλειδί αν υπάρχει (mDatabase.getReference("students")). Στη συνέχεια του κώδικα ακολουθούν οι τέσσερις βασικές συναρτήσεις που θα χρειαστούμε για τη διαχείριση αυτής της βάσης μέσω της εφαρμογής. Η πρώτη από αυτές είναι η read students που υλοποιεί τη φόρτωση των μαθητών στην εφαρμογή. Όσο η βάση δεδομένων δεν έχει φτάσει στο τέλος της μπαίνει σε μια εντολή επανάληψης και για κάθε μαθητή που βρίσκει εισάγει με μια λίστα με κλειδιά τον αριθμό του μαθητή, και σε μια λίστα τύπου student το μαθητή μαζί με τις πληροφορίες του. Σαν αποτέλεσμα από αυτή τη συνάρτηση έχουμε δυο λίστες μια με τα κλειδιά των μαθητών (τον αριθμό τους δηλαδή, πχ ο πρώτος μαθητής που θα μπει θα έχει κλειδί 1)και μια λίστα με τους μαθητές. Αυτές τις δυο λίστες τις περνάμε σαν ορίσματα στη συνάρτηση data is loaded του datastatus, που περνάει τις δυο λίστες μέσα στην εφαρμογή.

```java
package com.example.myapplication;

import androidx.annotation.NonNull;

import com.google.android.gms.tasks.OnSuccessListener;
import com.google.firebase.database.DataSnapshot;
import com.google.firebase.database.DatabaseError;
import com.google.firebase.database.DatabaseReference;
import com.google.firebase.database.FirebaseDatabase;
import com.google.firebase.database.ValueEventListener;

import java.util.ArrayList;
import java.util.List;

public class FirebaseDatabaseHelper {
    private FirebaseDatabase mDatabase;
    private DatabaseReference mReferenceStudents;
    private List<Student> students=new ArrayList<>();
    public interface DataStatus{
        void DataIsLoaded(List<Student> students,List<String> keys);
        void DataIsInserted();
        void DataIsUpdated();
        void DataIsDeleted(List<Student> students, List<String> keys);

        void DataIsDeleted();
    }
    public FirebaseDatabaseHelper() {
        mDatabase= FirebaseDatabase.getInstance();
        mReferenceStudents=mDatabase.getReference( path: "students");
    }

    public void readStudents(final DataStatus dataStatus){
        mReferenceStudents.addValueEventListener(new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot dataSnapshot) {
                students.clear();
                List<String> keys=new ArrayList<>();
                for(DataSnapshot keyNode : dataSnapshot.getChildren()){
                    keys.add(keyNode.getKey());
                    Student student=keyNode.getValue(Student.class);
                    students.add(student);
                }
                dataStatus.DataIsLoaded(students,keys);
```

Εικόνα 21

Στις ακόλουθες συναρτήσεις στην εικόνα 22,εφόσον έχουμε δημιουργήσει μια λίστα με κλειδιά για τις θέσεις των μαθητών τα πράγματα είναι πιο εύκολα. Για παράδειγμα στη συνάρτηση add student που θέλουμε να εισάγουμε ένα μαθητή χρειάζεται να αυξήσουμε τα κλειδιά κατά ένα και έχοντας το μαθητή σαν όρισμα στη συνάρτηση το τοποθετούμε σε αυτή τη θέση. Το όρισμα student είναι τύπου της κλάσης Student που έχουμε αναλύσει πιο πάνω άρα έχουμε το κατάλληλο UI για να περάσουμε τις πληροφορίες του μαθητή και το μόνο που μένει είναι η τοποθέτηση του στη βάση στη θέση του κλειδιού που μόλις βάλαμε στο τέλος της προηγουμένης λίστας. Στη συνέχεια γίνεται ενημέρωση της βάσης δεδομένων με ορίσματα το κλειδί της θέσης ένα αντικείμενο τύπου κλάσης Student και η χρήση αυτής της συνάρτησης είναι για την ενημέρωσή των στοιχείων ενός μαθητή με κλειδί που του δώσαμε. Το ίδιο γίνεται και στη συνάρτηση delete που περνώντας μόνο το κλειδί σαν όρισμα μπορούμε να διαγράψουμε ένα μαθητή που υπάρχει στη θέση της βάσης με το κλειδί αυτό.

```
            }


            @Override
            public void onCancelled(@NonNull DatabaseError databaseError) {

            }
        });
    }
    public void addStudent(Student student,final DataStatus dataStatus){
        String key=mReferenceStudents.push().getKey();
        mReferenceStudents.child(key).setValue(student).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                dataStatus.DataIsInserted();
            }
        });
    }

    public void updateStudent(String key,Student student,final DataStatus datastatus){
        mReferenceStudents.child(key).setValue(student).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                datastatus.DataIsUpdated();
            }
        });
    }

    public void deleteStudent(String key,final DataStatus datastatus){
        mReferenceStudents.child(key).setValue(null).addOnSuccessListener(new OnSuccessListener<Void>() {
            @Override
            public void onSuccess(Void aVoid) {
                datastatus.DataIsDeleted();
            }
        });
    }
}
```

Εικόνα 22

MAIN ACTIVITY 2

```
public class MainActivity2 extends AppCompatActivity {
    private RecyclerView mRecyclerView;
    private FirebaseAuth mAuth;
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main2);
        mAuth=FirebaseAuth.getInstance();
        mRecyclerView=(RecyclerView) findViewById(R.id.recyclerview_students);
        new FirebaseDatabaseHelper().readStudents(new FirebaseDatabaseHelper.DataStatus() {
            @Override
            public void DataIsLoaded(List<Student> students, List<String> keys) {
                new RecyclerView_Config().setConfig(mRecyclerView, context: MainActivity2.this,students,keys)
            }

            @Override
            public void DataIsInserted() {

            }

            @Override
            public void DataIsUpdated() {

            }

            @Override
            public void DataIsDeleted(List<Student> students, List<String> keys) {

            }

            @Override
            public void DataIsDeleted() {

            }
        });
    }

    @Override
    public boolean onCreateOptionsMenu(Menu menu) {
        FirebaseUser user =mAuth.getCurrentUser();
        getMenuInflater().inflate(R.menu.studentactivitymenu,menu);
        if(user!=null){
            menu.getItem( index: 0).setVisible(true);//new student
```

Εικόνα 23

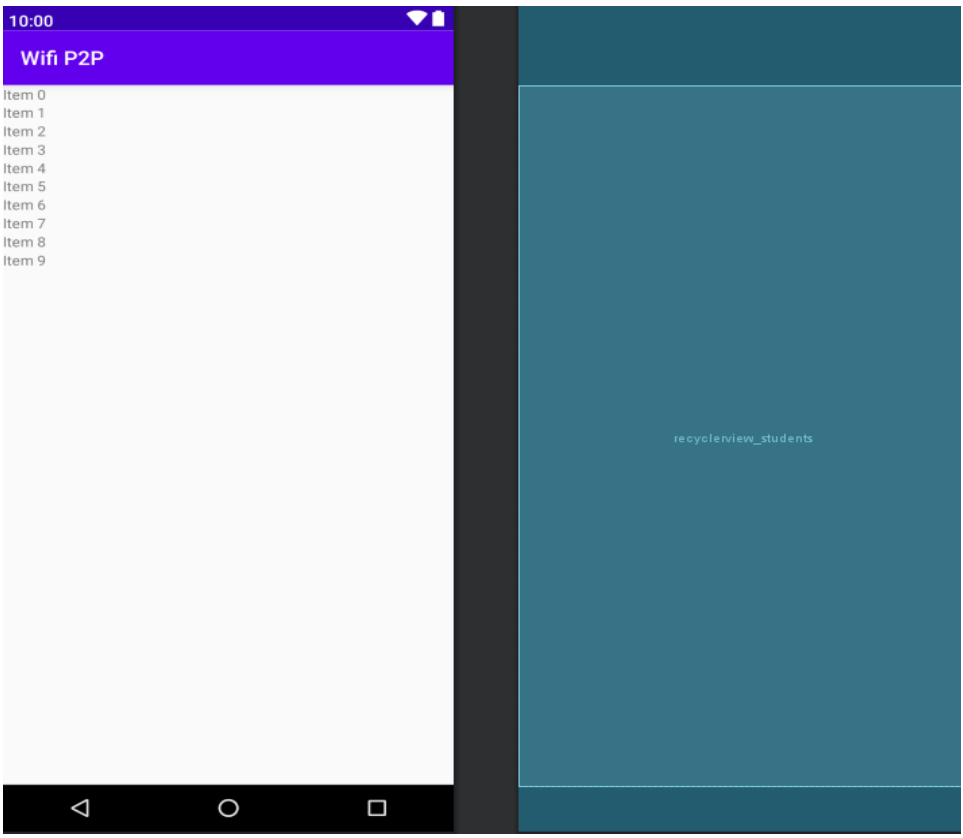
Η σελίδα MainActivity2 έχει να κάνει με το κώδικα που αντιπροσωπεύει τη βάση δεδομένων. Περιέχει ένα user interface το όποιο απευθύνεται σε χρήστες από τη πλευρά του καθηγητή καθώς μόνο αυτοί έχουν τη δυνατότητα πρόσβασης και προβολής της βάσης δεδομένων. Όπως αναφέραμε και προηγουμένως η βάση δεδομένων που θα εμφανίζεται στην εφαρμογή θα έχει τη μορφή λίστας από καρτέλες των μαθητών με τις πληροφορίες που είναι απαραίτητες για τη περιγραφή τους. Αρχικά στο

κώδικα δημιουργούμε ένα αντικείμενο της κλάσης της βάσης δεδομένων και καλούμε τη συνάρτηση readStudents όπως φαίνεται στην εικόνα 23.

```
            menu.getItem( index: 1).setVisible(false);//sign/register
            menu.getItem( index: 2).setVisible(true);//logout
        }else{
            menu.getItem( index: 0).setVisible(false);//new student
            menu.getItem( index: 1).setVisible(true);//sign/register
            menu.getItem( index: 2).setVisible(false);//logout
        }
        return super.onCreateOptionsMenu(menu);
    }

    @Override
    public boolean onPrepareOptionsMenu(Menu menu) {
        FirebaseUser user =mAuth.getCurrentUser();
        if(user!=null){
            menu.getItem( index: 0).setVisible(true);//new student
            menu.getItem( index: 1).setVisible(false);//sign/register
            menu.getItem( index: 2).setVisible(true);//logout
        }else{
            menu.getItem( index: 0).setVisible(false);//new student
            menu.getItem( index: 1).setVisible(true);//sign/register
            menu.getItem( index: 2).setVisible(false);//logout
        }
        return super.onPrepareOptionsMenu(menu);
    }

    @Override
    public boolean onOptionsItemSelected(@NonNull MenuItem item) {
        switch (item.getItemId()){
            case R.id.new_student:
                startActivity(new Intent( packageContext: this,NewStudentActivity.class));
                return true;
            case R.id.sign_in:
                startActivity(new Intent( packageContext: this,SingInActivity.class));
                return true;
            case R.id.sign_out:
                mAuth.signOut();
                invalidateOptionsMenu();
                RecyclerView_Config.Longout();
                return true;
        }
        return super.onOptionsItemSelected(item);
    }
}
```

Εικόνα 24

Με αυτό το τρόπο θα έχουμε πρόσβαση στη βάση για να ανακτήσουμε όλες τις πληροφορίες. Για να τις εμφανίσουμε σαν μια λίστα, στη συνάρτηση που έχουμε δημιουργήσει μέσα στη κλάση της βάσης δεδομένων (DataIsLoaded) δηλώνουμε ένα νέο αντικείμενο της κλάσης Recyclerview που είναι υπεύθυνη για τη προβολή της λίστας, με βάση τα στοιχεία που θα της περάσουμε. Όπως βλέπουμε και στο blueprint του user interface στην εικόνα 25 δεξιά, έχουμε ένα αντικείμενο τύπου recycler view για τη προβολή των στοιχείων σε μορφή λίστας. Με αυτό το τρόπο :
RecyclerView_Config().setConfig(
mRecyclerView,MainActivity2.this,
students,keys);
δημιουργούμε ένα αντικείμενο που του λέμε ότι στη κλάση MainActivity2 θα προβάλεις τα στοιχεία της λίστας students που το μέγεθος της είναι ίσο με τον αριθμό των κλειδιών της λίστας keys, και θα εμφανιστεί στη κλάση με τη μορφή αντικειμένου RecyclerView.



Εικόνα 25

Στη συνέχεια αυτής της σελίδα του κώδικα έχουμε τρεις συναρτήσεις που υπάρχουν στις εικόνες 23 και 24, οι οποίες ενεργοποιούν η απενεργοποιούν τα κουμπιά που υπάρχουν σαν δυνατότητες πάνω δεξιά στο user interface. Πιο συγκεκριμένα πρόκειται για την προσθήκη ενός μαθητή, τη σύνδεση ή την εγγραφή ενός χρήστη για να έχει δικαιώματα διαχείρισης της βάσης δεδομένων και την έξοδο από ένα λογαριασμό.

Η πρώτη συνάρτηση έχει να κάνει με είσοδο στη βάση δεδομένων που θα δημιουργηθεί το Options menu οι επιλογές δηλαδή στο πάνω δεξιά μέρος του user interface. Αν ο χρήστης έχει κάνει σύνδεση από τη προηγουμένη φορά χωρίς να κάνει έξοδο από το λογαριασμό του τότε την επόμενη φορά που θα ανοίξει την εφαρμογή θα είναι ακόμα συνδεδεμένος στον ίδιο λογαριασμό, και τα κουμπιά που θα είναι ενεργά θα είναι η προσθήκη μαθητή και η έξοδος. Αντίθετα αν είναι η πρώτη σύνδεση η μόνη επιλογή που θα έχει πάνω δεξιά θα είναι η εγγραφή ή είσοδος σε λογαριασμό. Παρόμοια λειτουργία έχει και η δεύτερη συνάρτηση που ανάλογα με την ύπαρξη συνδεδεμένου χρήστη ενεργοποιεί τις κατάλληλες επιλογές. Η τρίτη συνάρτηση έχει να κάνει με το τι πρέπει να γίνει μόλις επιλέξουμε ένα κουμπί από το μενού. Δηλαδή αν επιλέξουμε την σύνδεση σε λογαριασμό /εγγραφή θα πρέπει να μεταφερθούμε σε ένα user interface που θα μπορούμε να βάλουμε τα στοιχεία μας για να προχωρήσουμε με την εφαρμογή. Περιέχει μια εντολή διακλάδωσης η οποία ανάλογα με το κουμπί που θα πατήσουμε θα μας μεταφέρει στο κατάλληλο user interface.

MAIN ACTIVITY

Η main Activity είναι η κύρια σελίδα της εφαρμογής. Είναι το UI που έχει τις περισσότερες λειτουργίες του κώδικα και έχουνε πρόσβαση και οι δυο πλευρές χρηστών. Η πρώτη συνάρτηση που υπάρχει onCreate έχει να κάνει με τη δημιουργία και την εκκίνηση της εφαρμογής αφού δηλωθούν οι απαραίτητες βιβλιοθήκες και μεταβλητές. Μέσα σε αυτή υπάρχουν δυο μεγαλύτερες συναρτήσεις μια για τη δήλωση άλλων ενεργειών ή συναρτήσεων και η άλλη για την εκτέλεση κώδικα όπως είναι το πάτημα ενός κουμπιού.

```
public class MainActivity<gradleEnterprise> extends AppCompatActivity {

    Button btnOnOff, btnDiscover,btnDatabase,btnPresence,btnAbsences;
    ListView listView;
    TextView connectionStatus;
    EditText emailAdd;
    WifiManager wifiManager;
    WifiP2pManager mManager;
    WifiP2pManager.Channel mChannel;
    String y;
    BroadcastReceiver mReceiver;
    IntentFilter mIntentFilter;
    DatabaseReference mReferenceStudents;
    List<WifiP2pDevice> peers = new ArrayList<WifiP2pDevice>();
    List<String> temp=new ArrayList<>();

    String strMac="";
    String[] deviceNameArray;
    WifiP2pDevice[] deviceArray;
    int x=0;
    int num=0;
    Student student1;
    String macad;

    public MainActivity() {
    }
    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_main);

        initialWork();
        exqListener();
    }
    private void exqListener() {
        btnOnOff.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                if (wifiManager.isWifiEnabled()) {
                    wifiManager.setWifiEnabled(false);
                    btnOnOff.setText("ON");
                } else {
                    wifiManager.setWifiEnabled(true);
                    btnOnOff.setText("OFF");
```

Εικόνα 26

Μέσα στην exqListener η πρώτη συνάρτηση που υπάρχει αφορά την υλοποίηση της λειτουργίας του κουμπιού Wi-Fi που μέσω της μεταβλητής wifimanager μπορούμε να αλλάξουμε τη κατάσταση του Wi-Fi στο κινητό.

Στη συνέχεια υπάρχει ο κώδικας για το κουμπί discover που όταν πατηθεί μέσω της βιβλιοθήκης Wi-Fi p2p manager και της συνάρτησης που διαθέτει discover peers αρχίζει η αναζήτηση άλλων συσκευών για σύνδεση μέσα από ένα κανάλι που έχουμε περάσει εμείς σαν όρισμα στη συνάρτηση. Αν η διαδικασία ξεκινήσει εμφανίζει μήνυμα επιτυχίας, αλλιώς εμφανίζει αποτυχίας.

Στη συνάρτηση listView.setOnItemClickListener υπάρχει ο κώδικας για τη σύνδεση σε μια συσκευή όταν την επιλέξουμε από τη λίστα που θα εμφανιστεί με τις διαθέσιμες συσκευές μετά την αναζήτηση της προηγουμένης συνάρτησης. Μέσω της συναρτησης mManager.connect της βιβλιοθήκης Wi-Fi p2p manager και με όρισμα το κανάλι και ένα νέο αντικείμενο WifiP2pManager.ActionListener(), που είναι μια συνάρτηση για να ελέγχουμε αν έχει επιλέξει κάποια συσκευή ο χρήστης από τη λίστα, μπορούμε να ελέγχουμε αν έχει γίνει σύνδεση η όχι και εμφανίζεται κατάλληλο μήνυμα με το όνομα της συσκευής που συνδεθήκαμε.

```
        }
    }
});

btnDiscover.setOnClickListener(new View.OnClickListener() {
    @SuppressLint("MissingPermission")
    @Override
    public void onClick(View view) {

        mManager.discoverPeers(mChannel, new WifiP2pManager.ActionListener() {
            @Override
            public void onSuccess() { connectionStatus.setText("Discover Started"); }

            @Override
            public void onFailure(int i) {
                connectionStatus.setText("Discover Starting Fail");
            }
        });
    }
});
listView.setOnItemClickListener(new AdapterView.OnItemClickListener() {

    @SuppressLint("MissingPermission")
    @Override
    public void onItemClick(AdapterView<?> parent, View view, int i, long l) {
        final WifiP2pDevice device = deviceArray[i];
        WifiP2pConfig config = new WifiP2pConfig();
        config.deviceAddress = device.deviceAddress;
        mManager.connect(mChannel, config, new WifiP2pManager.ActionListener() {
            @Override
            public void onSuccess() {

                Toast.makeText(getApplicationContext(), text: "Connected to" + device.deviceName, Toast.LENGTH_SHORT).show();



            }

            @Override
            public void onFailure(int reason) {

                Toast.makeText(getApplicationContext(), text: "Not Connected", Toast.LENGTH_SHORT).show();
            }
        });
    }
```

Εικόνα 27

Η επόμενη συνάρτηση αφορά το κουμπί database που απλά έχει ως λειτουργία μόλις πατηθεί το κουμπί να μας μεταφέρει στη βάση δεδομένων. Αυτό το κουμπί είναι ενεργό μόνο σε χρήστες που χαρακτηρίζονται ως host δηλαδή έναν καθηγητή. Οι μαθητές ανάλογα με την έκδοση του android έχουν τη δυνατότητα απλά να το δουν αλλά να μην έχουν τη δυνατότητα να το πατήσουν ή δε θα μπορούν να το δουν καθόλου στην εφαρμογή.

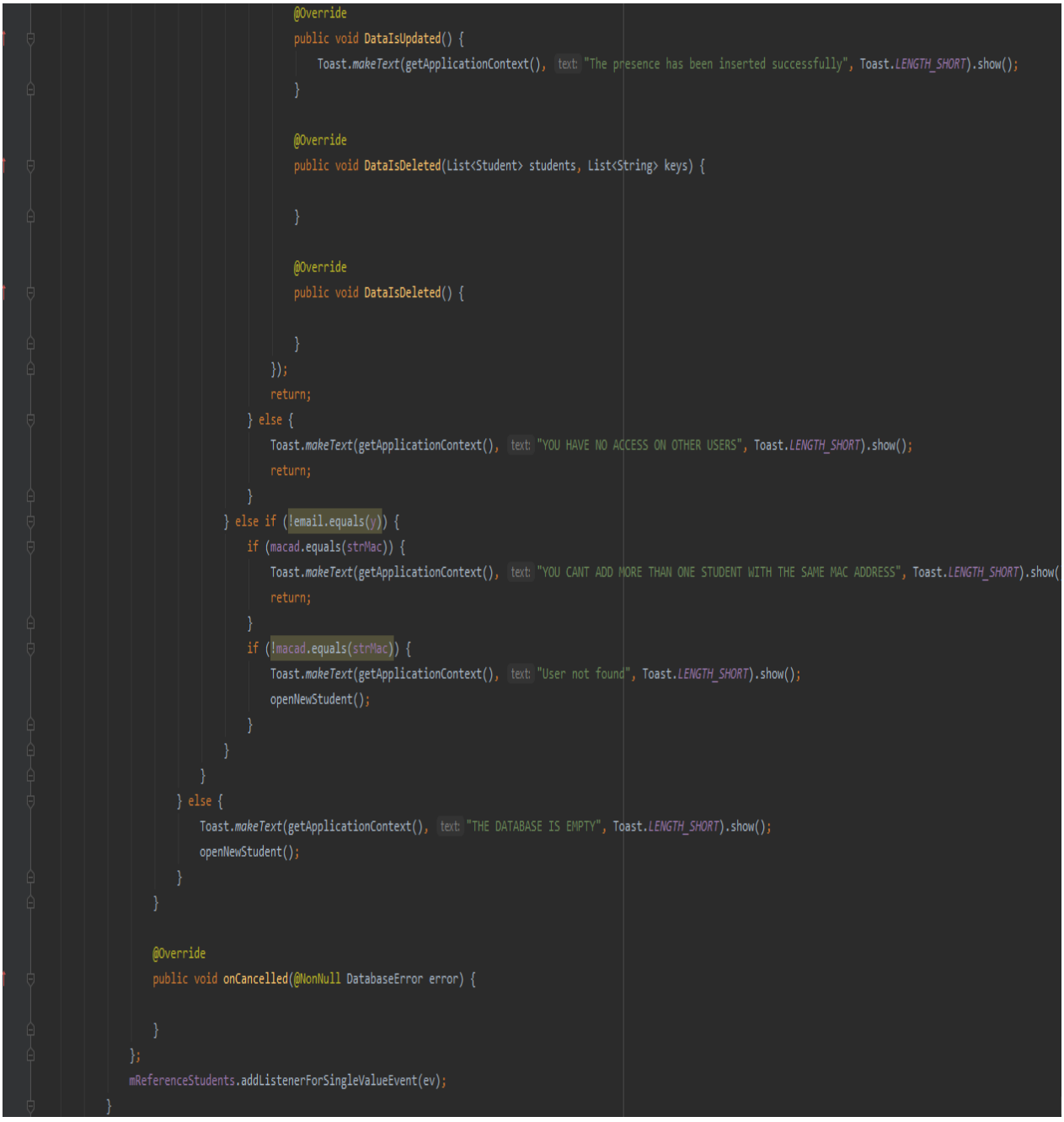
```
});

btnDatabase.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        openActivity2();
    }
});
btnPresence.setOnClickListener(new View.OnClickListener() {
    @Override
    public void onClick(View v) {
        y = emailAdd.getText().toString();
        if (y.equals("")) {
            Toast.makeText(getApplicationContext(), text: "YOU HAVE TO ENTER THE EMAIL FIRST ", Toast.LENGTH_SHORT).show();
            return;
        }

        mReferenceStudents = FirebaseDatabase.getInstance().getReference( path: "students");
        ValueEventListener ev = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                ArrayList<String> emails = new ArrayList<>();
                if (snapshot.exists()) {
                    for (DataSnapshot ds : snapshot.getChildren()) {
                        String email = ds.child("email").getValue().toString();
                        emails.add(email);
                        macad = ds.child("mac").getValue().toString();
                        if (email.equals(y)) {
                            if (macad.equals(strMac)) {
                                Toast.makeText(getApplicationContext(), text: "Nice", Toast.LENGTH_SHORT).show();
                                int y = Integer.parseInt(ds.child("presences").getValue().toString());
                                y++;
                                String s = String.valueOf(y);
                                Date c= Calendar.getInstance().getTime();
                                DateFormat setdate=new SimpleDateFormat( pattern: "dd/MM/yyyy");
                                String date=setdate.format(c);
                                temp.add(email);
                                Toast.makeText(getApplicationContext(), text: "User found"+ temp.get(x), Toast.LENGTH_SHORT).show();
                                x++;
                                Toast.makeText(getApplicationContext(), text: "User found", Toast.LENGTH_SHORT).show();
                                student1 = ds.getValue(Student.class);
                                student1.setMAC(strMac);
                                student1.setDate(date);
                                student1.setPresences(s);
```

Εικόνα 28

Στη συνέχεια του κώδικα έχουμε το κουμπί για την εισαγωγή της παρουσίας του μαθητή στη βάση δεδομένων. Για να μπορέσουμε να εισάγουμε παρουσία σε ένα μαθητή θα πρέπει πρώτα να βρεθεί ο μαθητής μέσω κάποιων συνθηκών. Κάθε μαθητής έχει εισάγει το email του στη βάση. Έτσι με την αναζήτηση ενός email μπορούμε να βρούμε σε ποιο μαθητή θέλουμε να βάλουμε τη παρουσία. Στο κώδικα της συνάρτησης για το κουμπί έχουμε πρώτα έναν έλεγχο για το πλαίσιο κειμένου που αν είναι κενό, δηλαδή ο μαθητής δεν έχει βάλει το email του για να γίνει η αναζήτηση, εμφανίζει κατάλληλο μήνυμα για τη συμπλήρωσή του. Αφού γίνει αυτό δημιουργούμε ένα αντικείμενο της κλάσης firebase database για να έχουμε πρόσβαση στα στοιχεία από τους μαθητές και να μπορούμε να κάνουμε τον έλεγχο για την εισαγωγή της παρουσίας. Μέσα στην εντολή επανάληψης έχουμε μια μεταβλητή τύπου string που παίρνει κάθε φορά το email του κάθε μαθητή. Η επανάληψη να σταματήσει όταν βρεθεί ο μαθητής με το email που έχει περαστεί στο πλαίσιο κειμένου. Για να μην υπάρχει πιθανότητα να βρεθούν δυο μαθητές με το ίδιο email έχουμε βάλει και έλεγχο της mac address, έτσι ακόμα και το email να είναι ίδιο ο αριθμός της συσκευής θα είναι διαφορετικός. Αν ο μαθητής βρεθεί, αν περάσουμε και από τους δυο ελέγχους, τότε εμφανίζουμε ένα μήνυμα ότι έχει βρεθεί ο μαθητής δημιουργούμε ένα αντικείμενο της βιβλιοθήκης date για να μπορέσουμε να κρατήσουμε την ημερομηνία της σύνδεσης, δηλαδή την ημερομηνία της παρουσίας, βάζουμε τις σωστές τιμές στις μεταβλητές του μαθητή, αυξάνουμε τις παρουσίες του και περνάμε την ημερομηνία στη βάση, και στη συνέχεια κάνουμε ενημέρωσή τη βάση δεδομένων για να μπορέσουμε να περάσουμε τα νέα στοιχεία. Αν η mac address είναι άδεια τότε ο μαθητής είχε προστεθεί από το καθηγητή άρα πρέπει να υπάρξει πρώτα σύνδεση για εισαγωγή της mac address στη βάση και στη συνέχεια να περαστεί η απουσία. Άρα θα έχουμε πρώτα ενημέρωση της βάσης για τη mac address του μαθητή και αφού γίνει επιστροφή θα πρέπει να επαναληφθεί η διαδικασία για την εισαγωγή της παρουσίας. Αν δεν υπάρχει το email αλλά υπάρχει η mac address εμφανίζεται μήνυμα που λέει ότι δε γίνεται να υπάρχει άλλος χρήστης με την ίδια mac address, ενώ αν δεν υπάρχει ούτε το email ούτε η mac address τότε εμφανίζεται μήνυμα ότι δεν υπάρχει ο χρήστης. Όλα αυτά συμβαίνουν όταν η βάση δεδομένων δεν είναι κενή, δηλαδή όταν υπάρχει το αντικείμενο snapshot της Data snapshot, η απεικόνιση της βάσης δεδομένων. Αν δεν υπάρχει το snapshot τότε η βάση είναι κενή και εμφανίζεται μήνυμα. Τα κομμάτια από το κώδικα αυτής της συνάρτησης υπάρχουν στις εικόνες 28, 29 και 30

```
                    new FirebaseDatabaseHelper().updateStudent(ds.getKey(), student1, new FirebaseDatabaseHelper.DataStatus() {
                        @Override
                        public void DataIsLoaded(List<Student> students, List<String> keys) {

                        }

                        @Override
                        public void DataIsInserted() {

                        }

                        @Override
                        public void DataIsUpdated() {
                            Toast.makeText(getApplicationContext(), text: "The presence has been inserted successfully", Toast.LENGTH_SHORT).show();
                        }

                        @Override
                        public void DataIsDeleted(List<Student> students, List<String> keys) {

                        }

                        @Override
                        public void DataIsDeleted() {

                        }
                    });
                    return;
                } else if (macad.equals("")) {
                    Toast.makeText(getApplicationContext(), text: "FIRST TIME CONNECTION", Toast.LENGTH_SHORT).show();
                    openStudentDetails();
                    student1 = ds.getValue(Student.class);
                    student1.setMAC(strMac);
                    Toast.makeText(getApplicationContext(), text: "User found" + y, Toast.LENGTH_SHORT).show();
                    new FirebaseDatabaseHelper().updateStudent(ds.getKey(), student1, new FirebaseDatabaseHelper.DataStatus() {
                        @Override
                        public void DataIsLoaded(List<Student> students, List<String> keys) {

                        }

                        @Override
                        public void DataIsInserted() {

                        }
```

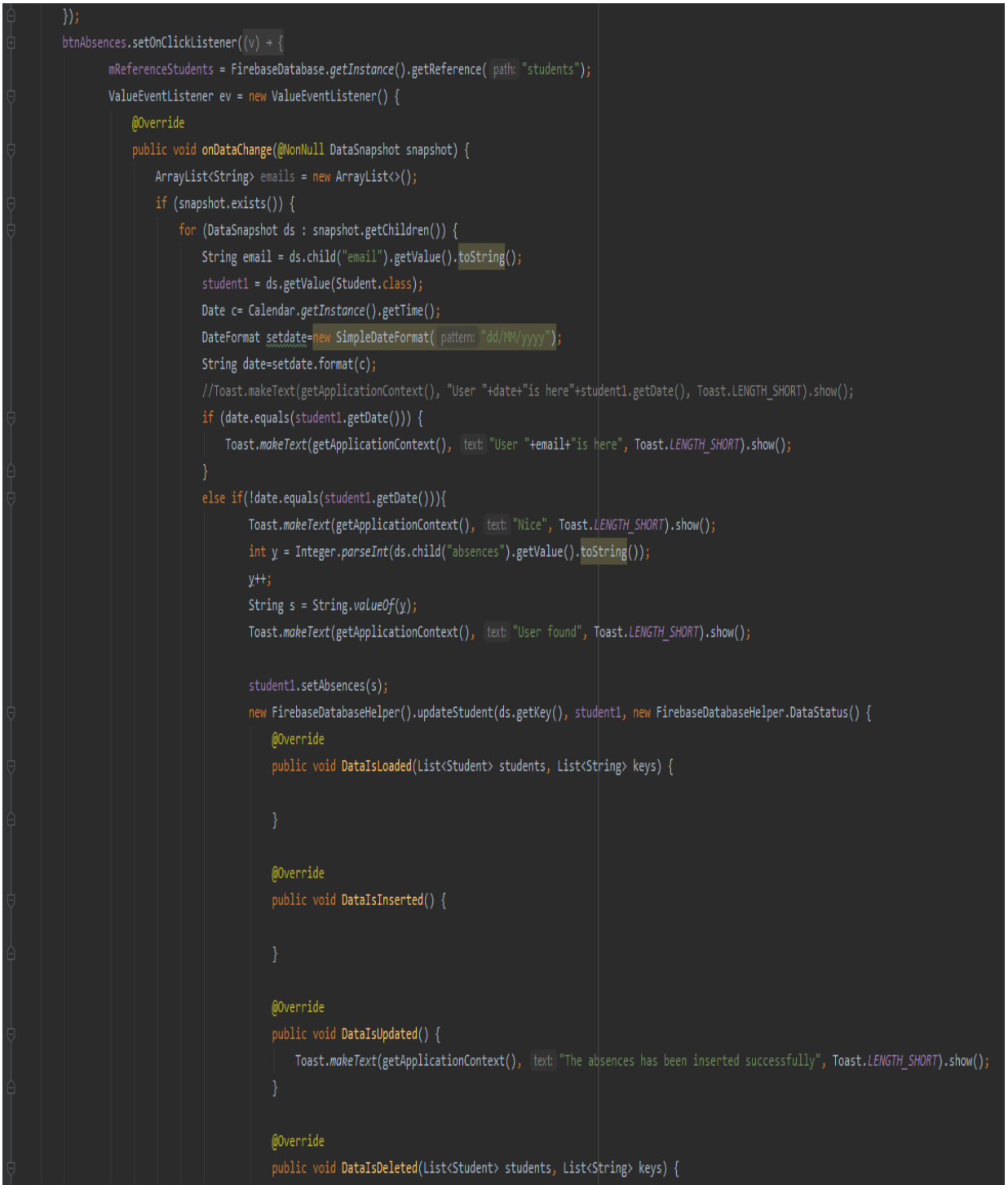
Εικόνα 29


@Override
public void DataIsUpdated() {
Toast.makeText(getApplicationContext(), text: "The presence has been inserted successfully", Toast.LENGTH_SHORT).show();
}
@Override
public void DataIsDeleted(List<Student> students, List<String> keys) {
}
@Override
public void DataIsDeleted() {
}
});
return;
} else {
Toast.makeText(getApplicationContext(), text: "YOU HAVE NO ACCESS ON OTHER USERS", Toast.LENGTH_SHORT).show();
return;
}
} else if (!email.equals(y)) {
if (macad.equals(strMac)) {
Toast.makeText(getApplicationContext(), text: "YOU CANT ADD MORE THAN ONE STUDENT WITH THE SAME MAC ADDRESS", Toast.LENGTH_SHORT).show(
return;
}
if (!macad.equals(strMac)) {
Toast.makeText(getApplicationContext(), text: "User not found", Toast.LENGTH_SHORT).show();
openNewStudent();
}
}
}
} else {
Toast.makeText(getApplicationContext(), text: "THE DATABASE IS EMPTY", Toast.LENGTH_SHORT).show();
openNewStudent();
}
}
@Override
public void onCancelled(@NonNull DatabaseError error) {
}
};
mReferenceStudents.addListenerForSingleValueEvent(ev);
}

Εικόνα 30

Η επόμενη συνάρτηση αφορά τη λειτουργία του κουμπιού για τις απουσίες των μαθητών, στο όποιο έχει πρόσβαση ο καθηγητής. Όπως και πριν με τις παρουσίες έτσι και τώρα πρέπει να υπάρχει ένα αντικείμενο της βάσης δεδομένων έτσι ώστε να έχουμε πρόσβαση σε αυτή για να κάνουμε τους απαραίτητους ελέγχους για τη προσθήκη της απουσίας σε όλους τους μαθητές που δε συνδέθηκαν. Αν η βάση δεδομένων είναι κενή τότε δεν υπάρχει το αντικείμενο της Data snapshot και εμφανίζεται απλά ένα μήνυμα. Αν το αντικείμενο υπάρχει, άρα η βάση δεδομένων περιέχει μαθητές, τότε μπαίνουμε μέσα σε μια εντολή επανάληψης η οποία θα τερματίσει όταν ελεγχθούν όλα τα στοιχεία της βάσης. Μέσα στην επανάληψη δημιουργείτε ένα αντικείμενο τύπου date για να πάρουμε την ημερομηνία, και από τη βάση δεδομένων ορίζουμε μέσω της δεύτερης data snapshot ένα μαθητή με τις δυνατότητες και τις συναρτήσεις που διαθέτει κάθε μαθητής της βάσης δεδομένων. Έτσι μπορούμε να πάρουμε την ημερομηνία της τελευταίας σύνδεσης κάθε μαθητή. Όταν η ημερομηνία του μαθητή είναι ίδια με αυτή του αντικειμένου date που δημιουργήσαμε, τότε έχει παραβρεθεί στο μάθημα και εμφανίζεται ένα μήνυμα ότι ο μαθητής με το email του μαθητή ήταν στο μάθημα. Όταν η ημερομηνία του μαθητή διαφέρει από την ημερομηνία που έχει δημιουργηθεί τότε εμφανίζεται μήνυμα ότι βρέθηκε μαθητής τραβάμε από τη βάση τη τιμή των απουσιών του, την αυξάνουμε και τη ξαναπερνάμε στη μεταβλητή του μαθητή και ενημερώνουμε τη βάση δεδομένων μέσω της update από τη FirebaseDatabaseHelper(). Στο τέλος της επανάληψης θα έχουν περαστεί απουσίες σε όσους μαθητές δεν έχουν την ίδια ημερομηνία με αυτή του καθηγητή. Για να είναι σωστή αυτή η διαδικασία, το κουμπί των απουσιών θα πρέπει να πατηθεί από το καθηγητή μετρά από ένα χρονικό διάστημα στη μέρα του μαθήματος, είτε αυτό είναι 10λεπτα αφού ξεκινήσει το μάθημα είτε αυτό είναι στο τέλος του μαθήματος. Αν ο καθηγητής πατήσει το κουμπί την επόμενη μέρα, για παράδειγμα, θα εισαχθούν απουσίες σε όλους τους μαθητές που είναι μέσα στη βάση.

Στη συνέχεια υπάρχουν τρεις συναρτήσεις, η openActivity2, η openNewStudent και η openStudentDetais. Όλες αυτές ανοίγουν μια άλλη σελίδα του κώδικα και παράλληλα εμφανίζουν στο χρήστη ένα άλλο UI για την εφαρμογή, με τη μόνη διαφορά ότι η δεύτερη συνάρτηση μέσω της intent περνάει μια τιμή από τη κλάση main activity στη κλάση open new student. Αυτή η τιμή είναι η mac address του κινητού για να μπορέσει να περαστεί μέσα στη βάση δεδομένων. Αυτά περιέχονται στις εικόνες 31 και 32.



```
});
btnAbsences.setOnClickListener((v) -> {
        mReferenceStudents = FirebaseDatabase.getInstance().getReference( path: "students");
        ValueEventListener ev = new ValueEventListener() {
            @Override
            public void onDataChange(@NonNull DataSnapshot snapshot) {
                ArrayList<String> emails = new ArrayList<>();
                if (snapshot.exists()) {
                    for (DataSnapshot ds : snapshot.getChildren()) {
                        String email = ds.child("email").getValue().toString();
                        student1 = ds.getValue(Student.class);
                        Date c= Calendar.getInstance().getTime();
                        DateFormat setdate=new SimpleDateFormat( pattern: "dd/MM/yyyy");
                        String date=setdate.format(c);
                        //Toast.makeText(getApplicationContext(), "User "+date+"is here"+student1.getDate(), Toast.LENGTH_SHORT).show();
                        if (date.equals(student1.getDate())) {
                            Toast.makeText(getApplicationContext(),  text: "User "+email+"is here", Toast.LENGTH_SHORT).show();
                        }
                        else if(!date.equals(student1.getDate())){
                                Toast.makeText(getApplicationContext(),  text: "Nice", Toast.LENGTH_SHORT).show();
                                int y = Integer.parseInt(ds.child("absences").getValue().toString());
                                y++;
                                String s = String.valueOf(y);
                                Toast.makeText(getApplicationContext(),  text: "User found", Toast.LENGTH_SHORT).show();

                                student1.setAbsences(s);
                                new FirebaseDatabaseHelper().updateStudent(ds.getKey(), student1, new FirebaseDatabaseHelper.DataStatus() {
                                    @Override
                                    public void DataIsLoaded(List<Student> students, List<String> keys) {

                                    }

                                    @Override
                                    public void DataIsInserted() {

                                    }

                                    @Override
                                    public void DataIsUpdated() {
                                        Toast.makeText(getApplicationContext(),  text: "The absences has been inserted successfully", Toast.LENGTH_SHORT).show();
                                    }

                                    @Override
                                    public void DataIsDeleted(List<Student> students, List<String> keys) {
```

Εικόνα 31

```
                                    }

                                    @Override
                                    public void DataIsDeleted() {

                                    }
                                });
                                num++;
                            }
                        }


                    } else {
                        Toast.makeText(getApplicationContext(), text: "THE DATABASE IS EMPTY", Toast.LENGTH_SHORT).show();
                        openNewStudent();
                    }
                }

                @Override
                public void onCancelled(@NonNull DatabaseError error) {

                }
            };
            mReferenceStudents.addListenerForSingleValueEvent(ev);
        });
    }
    private void openActivity2(){
        Intent intent =new Intent( packageContext: this,MainActivity2.class);
        startActivity(intent);
    }
    private void openNewStudent(){
        Intent intent =new Intent( packageContext: this,NewStudentActivity.class);
        intent.putExtra( name: "MAC_ADDRESS",strMac);
        Toast.makeText(getApplicationContext(), text: "YOU"+intent.getStringExtra( name: "MAC_ADDRESS"),Toast.LENGTH_SHORT).show();
        startActivity(intent);
    }
    private void openStudentDetails(){
        Intent intent =new Intent( packageContext: this,StudentDetailsActivity.class);
        startActivity(intent);
    }
    private void initialWork() {
        btnPresence=(Button)findViewById(R.id.Presence);
        btnPresence.setEnabled(false);
```

Εικόνα 32

```
        btnAbsences=(Button)findViewById(R.id.Absence);
        btnAbsences.setEnabled(false);
        btnOnOff=(Button) findViewById(R.id.onOff);
        btnDatabase=(Button) findViewById(R.id.Database);
        btnDatabase.setEnabled(false);
        emailAdd=(EditText)findViewById(R.id.TextEmailAddress);
        emailAdd.setEnabled(true);
        btnDiscover=(Button) findViewById(R.id.discover);
        listView=(ListView) findViewById(R.id.peerListView);
        connectionStatus=(TextView) findViewById(R.id.connectionStatus);
        wifiManager= (WifiManager) getApplicationContext().getSystemService(Context.WIFI_SERVICE);
        try{
            List<NetworkInterface> nIList= Collections.list(NetworkInterface.getNetworkInterfaces());

            for(NetworkInterface nI:nIList){
                if(nI.getName().equalsIgnoreCase( anotherString: "wlan0")){
                    for(int i=0;i<nI.getHardwareAddress().length;i++){
                        String strMacByte=Integer.toHexString( i: nI.getHardwareAddress()[i]&0xFF);
                        if(strMacByte.length()==1){
                            strMacByte="0"+strMacByte;
                        }
                        if(i<nI.getHardwareAddress().length-1){
                            strMac=strMac+strMacByte.toUpperCase()+":";
                        }else{
                            strMac=strMac+strMacByte.toUpperCase();
                        }
                    }
                    break;
                }
            }
            Toast.makeText(getApplicationContext(), text: "Mac address="+strMac,Toast.LENGTH_SHORT).show();
        } catch (SocketException e) {
            e.printStackTrace();
        }


        mManager= (WifiP2pManager) getSystemService(Context.WIFI_P2P_SERVICE);
        mChannel=mManager.initialize( srcContext: this,getMainLooper(), listener: null);

        mReceiver=new WifiDirectBroadcastReceiver(mManager,mChannel, mActivity: this);
        mIntentFilter=new IntentFilter();
        mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_STATE_CHANGED_ACTION);
        mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_PEERS_CHANGED_ACTION);
        mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_CONNECTION_CHANGED_ACTION);
```

Εικόνα 33

Ακολουθεί η συνάρτηση initialWork, εικόνες 32 και 33. Μέσα σε αυτή τη κλάση γίνονται οι αρχικές δηλώσεις όλων των κουμπιών και αντικειμένων που χρειάζεται για να μπορέσει να λειτουργήσει η εφαρμογή και το UI της σελίδας του κώδικα. Στην αρχή δηλώνουμε τα κουμπιά και παράλληλα πια από αυτά θέλουμε να είναι ενεργά και ποια όχι. Στη συνέχεια υπάρχει η διαδικασία για την ανάκτηση της mac address του κινητού και την αποθήκευση της σε μια μεταβλητή. Γίνονται οι δηλώσεις των αντικειμένων από τις βιβλιοθήκες όπως Wi-Fi p2p manager, γίνεται η δημιουργία του καναλιού για την επικοινωνία των δυο συσκευών και περνάμε τις άδειες για τις αλλαγές σε Wi-Fi.

```
    mIntentFilter.addAction(WifiP2pManager.WIFI_P2P_THIS_DEVICE_CHANGED_ACTION);
}

WifiP2pManager.PeerListListener peerListListener=(peerList) → {
        if(!peerList.getDeviceList().equals(peers)){
            peers.clear();
            peers.addAll(peerList.getDeviceList());


            deviceNameArray=new String[peerList.getDeviceList().size()];
            deviceArray=new WifiP2pDevice[peerList.getDeviceList().size()];
            int index=0;

            for(WifiP2pDevice device : peerList.getDeviceList()){
                deviceNameArray[index]=device.deviceName;
                deviceArray[index]=device;
                index++;
            }

            ArrayAdapter<String> adapter=new ArrayAdapter<~>(getApplicationContext(),android.R.layout.simple_list_item_activated_1,deviceNameArray);
            listView.setAdapter(adapter);
        }
        if(peers.size()==0){

            Toast.makeText(getApplicationContext(), text: "No device found",Toast.LENGTH_SHORT).show();
            Toast.makeText(getApplicationContext(), text: "You have to enable the GPS and repress the discover",Toast.LENGTH_SHORT).show();
            return;
        }
};

WifiP2pManager.ConnectionInfoListener connectionInfoListener=new WifiP2pManager.ConnectionInfoListener() {
    @Override
    public void onConnectionInfoAvailable(WifiP2pInfo wifiP2pInfo) {
        final InetAddress groupOwnerAddress=wifiP2pInfo.groupOwnerAddress;

        if(wifiP2pInfo.groupFormed && wifiP2pInfo.isGroupOwner){
            connectionStatus.setText("Host");
            btnDatabase.setEnabled(true);
            btnAbsences.setEnabled(true);
        }else if(wifiP2pInfo.groupFormed){
            connectionStatus.setText("Client");
            btnPresence.setEnabled(true);
        }
    }
```

Εικόνα 34

Μετά υπάρχει η δημιουργία του WifiP2pManager.PeerListListener peer List Listener = new WifiP2pManager.PeerListListener() εικόνα 34. Είναι μια λίστα από τις συσκευές που θα βρεθούν μέσω της αναζήτησης Wi-Fi p2p. Όταν μια συσκευή έχει ενεργοποιήσει το Wi-Fi της φαίνεται σαν ενεργή στη λίστα και έτσι εισάγεται το όνομα της στο peerlistlistener. Αν η συσκευή του κινητού έχει λειτουργικό android μεγαλύτερης έκδοσης από την 8 και πάνω τότε θα πρέπει για να γίνει σωστά η προβολή της λίστας των συσκευών και ο εντοπισμός των συσκευών να έχουμε ανοιχτά τα δεδομένα τοποθεσίας (το GPS).

Στο τελευταίο κομμάτι του κώδικα υπάρχει το connection info listener που είναι υπεύθυνο για τον ορισμό του χρήστη host και client. Ανάλογα με τη συσκευή που έχει δημιουργήσει τη σύνδεση, όπου είναι πάντα η ίδια, ονομάζει τις συσκευές σαν host ή client και κατάλληλα ενεργοποιεί τα κουμπιά που θα πρέπει να έχει η κάθε πλευρά. Αν θέλαμε να περιορίσουμε ποιος θα είναι ο host θα μπορούσαμε εδώ να προσθέσουμε έλεγχο με τη mac address μιας συσκευής και να λέγαμε ότι η συσκευή με την mac address 1 πχ θα είναι ο καθηγητής και με την 2 ο μαθητής ή ότι με την 1 θα είναι ο καθηγητής και οποία άλλη mac address βρεθεί να γίνει σαν μαθητής. Αυτό μπορεί να υλοποιηθεί σε ενημέρωση η αναβάθμιση της ήδη υπάρχουσας εφαρμογής.

```
    @Override
    protected void onResume() {
        super.onResume();
        registerReceiver(mReceiver,mIntentFilter);
    }

    @Override
    protected void onPause() {
        super.onPause();
        unregisterReceiver(mReceiver);
    }
}
```

Εικόνα 35

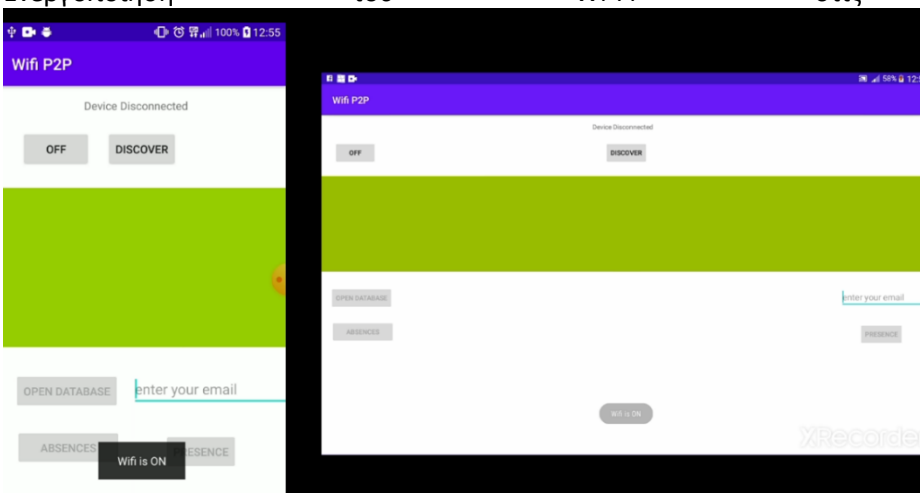
Ακολουθούν δυο συναρτήσεις onResume και onPause εικόνα 35. Αυτές οι δυο είναι υπεύθυνες για τη δημιουργία του καναλιού Wi-Fi broadcast receiver και στην ουσία στη πρώτη συνάρτηση κάνουμε register για ενεργοποίηση του καναλιού επικοινωνίας για να μπορέσουμε να έχουμε σύνδεση μεταξύ των συσκευών, και η δεύτερη κάνει unregister για να σταματήσει τη λειτουργία του καναλιού. Χωρίς αυτές τις δυο συναρτήσεις δε θα χρησιμοποιείτε η κλάση wifibroadcast receiver και δε θα μπορέσουμε να εντοπίσουμε τις συσκευές μέσω του κουμπιού discover.

ΠΕΡΙΓΡΑΦΗ ΤΗΣ ΛΕΙΤΟΥΡΓΙΑΣ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

ΧΡΗΣΗ ΤΗΣ ΕΦΑΡΜΟΓΗΣ

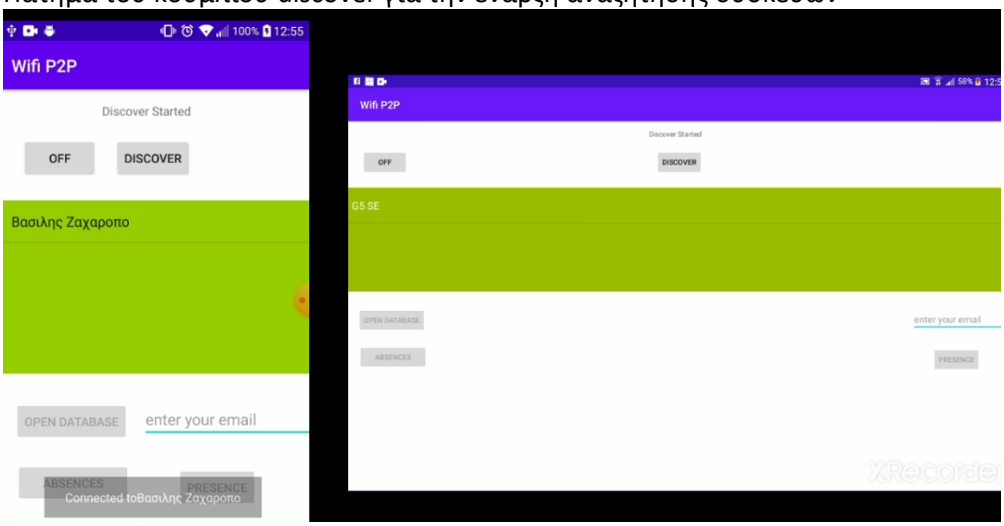
Τα βήματα για τη λειτουργία της εφαρμογής είναι τα εξής:

- Ενεργοποίηση του Wi-Fi στις συσκευές



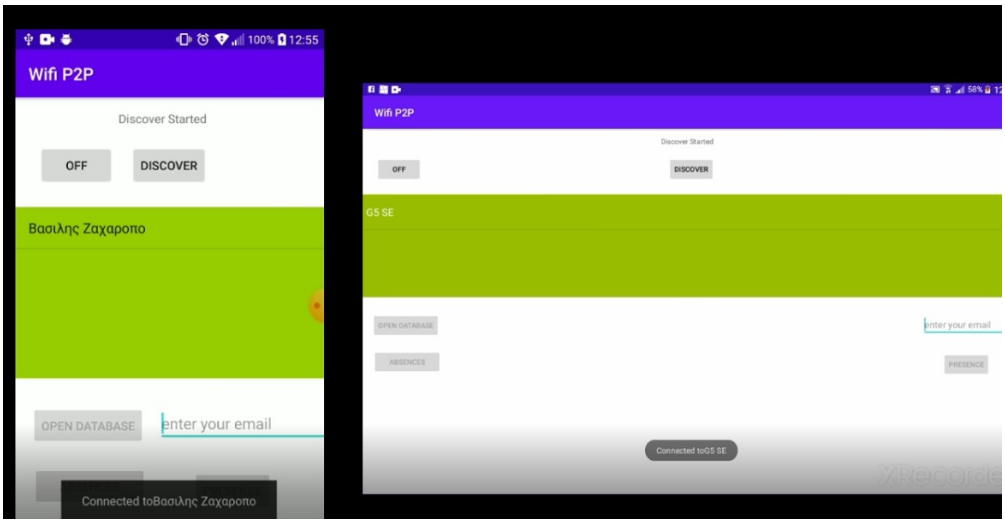
Εικόνα 36

- Πάτημα του κουμπιού discover για την έναρξη αναζήτησης συσκευών



Εικόνα 37

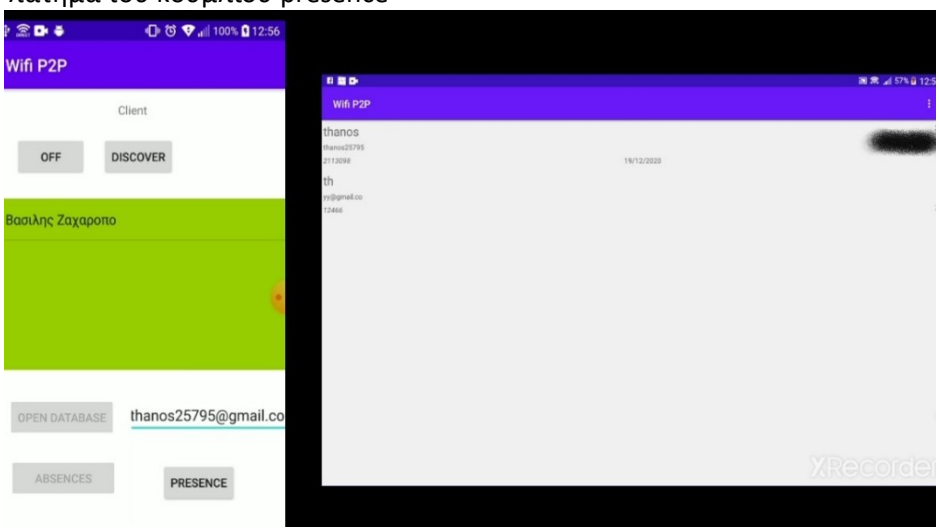
- Επιλογή μιας συσκευής από τη λίστα για σύνδεση με αυτή



Εικόνα 38

Απο τη πλευρά του client :

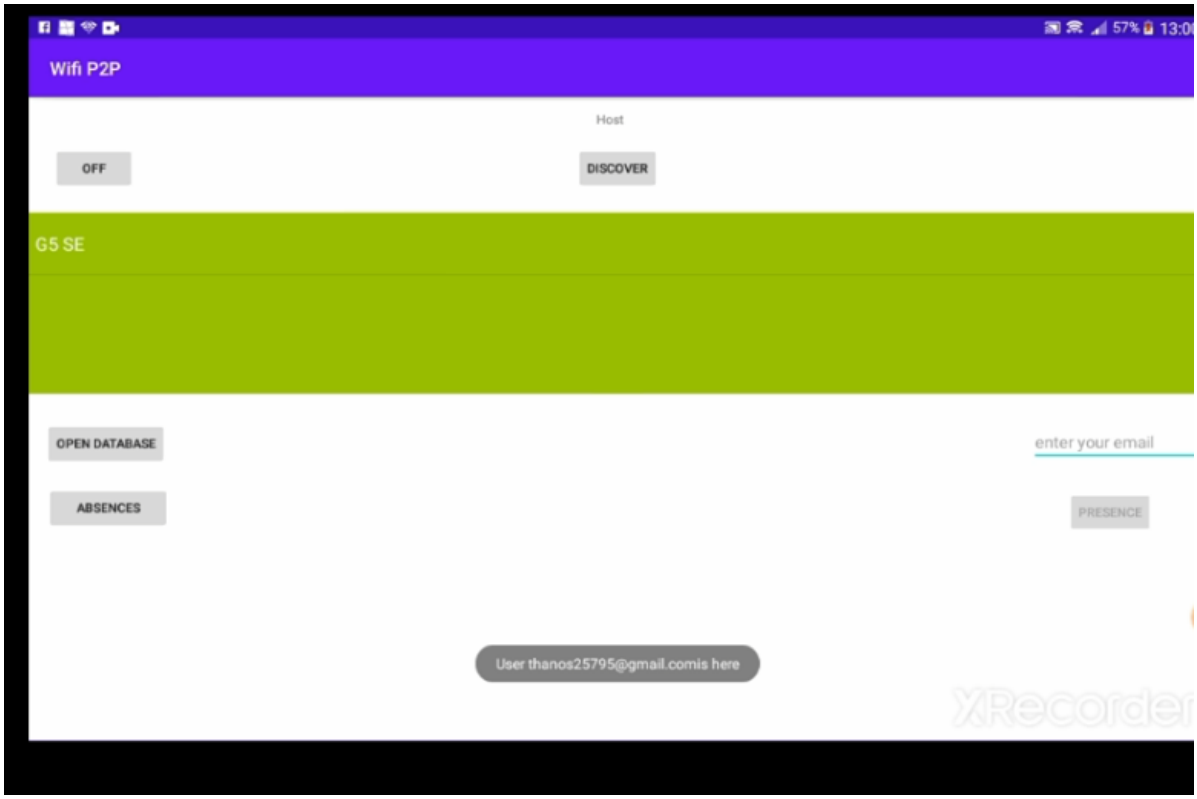
- Εισαγωγή του email του χρήστη και πάτημα του κουμπιού presence



Εικόνα 39

Απο τη πλευρά του host :

- Δυνατότητα πρόσβασης στη βάση δεδομένων και διαχείριση αυτής μετρά τη σύνδεση
- Πάτημα του κουμπιού absences για την εισαγωγή απουσιών



Εικόνα 40

ΚΕΦΑΛΑΙΟ 5 Συμπεράσματα και μελλοντική εξέλιξη

Σαν στόχος της πτυχιακής είχε τεθεί η δημιουργία μιας απλής και εύκολης εφαρμογής στη χρήση της, που θα μπορεί:

- να συνδέσει δυο συσκευές μέσω Wi-Fi για την ανταλλαγή πληροφοριών
- θα δημιουργεί μια βάση δεδομένων για την αποθήκευση των στοιχείων και των μαθητών
- θα έχει ξεχωριστούς τομείς διαχείρισης του καθηγητή από το μαθητή
- θα διαχειρίζεται τις απουσίες και τις παρουσίες των μαθητών με εύκολο τρόπο
- να τηρεί τη μοναδικότητα του κάθε χρήστη

Το πρώτο πράγμα λοιπόν που θα πρέπει να αναφέρουμε σα συμπέρασμα είναι το ποσοστό επιτυχίας του στόχου που είχε τεθεί. Η εφαρμογή συνδέει δυο συσκευές μέσω της λειτουργίας του Wi-Fi direct και

αν δεν γίνει πρώτα αυτή η σύνδεση μεταξύ των συσκευών κανείς από τους δυο, είτε μαθητής είτε καθηγητής, δε θα έχει πρόσβαση στο ηλεκτρονικό παρουσιολόγιο. Άρα καταφέραμε να έχουμε εξάρτηση αναμεσά τους και επίτευξη του πρώτου μέρους της πτυχιακής που ήταν ένα από τα πιο σημαντικά κομμάτια.

Το δεύτερο κομμάτι που θα έπρεπε να διαχειριστούμε είναι οι διαφορετικές λειτουργίες και χρήσεις που θα είχε ένας μαθητής και ένας καθηγητής από αυτή την εφαρμογή. Ο μαθητής χρησιμοποιεί την εφαρμογή σαν ένα απλό παρουσιολόγιο οπότε εισέρχεται στην εφαρμογή για να περάσει τη παρουσία του στο μάθημα. Από την άλλη μεριά ο καθηγητής έχει πρόσβαση στη βάση δεδομένων για να μπορεί να ενημερώνεται για τις απουσίες που έχει ο κάθε μαθητής καθώς και τη διορθώσει λαθών. Επιπλέον είναι αυτός που θα εισάγει τις απουσίες στην εφαρμογή με το πάτημα ενός κουμπιού. Με τη χρήση διαφορετικών UI έχουμε καταφέρει να πετύχουμε και αυτό το κομμάτι της πτυχιακής καθώς υπάρχουν διαφορές από χρήστη σε χρήστη, και ο καθένας από αυτούς έχει διαφορετικό τρόπο χρήσης της εφαρμογής.

Για την αποθήκευση όλων των μαθητών και των στοιχείων τους επιλέχθηκε η χρήση μιας online βάσης δεδομένων σε πραγματικό χρόνο η οποία ενημερώνεται αυτόματα μετά από κάθε αλλαγή η προσθήκη με το πάτημα του κατάλληλου κουμπιού σε κάθε UI που τη χρησιμοποιεί. Όσο για τη μοναδικότητα του κάθε χρήστη, η σύνδεση που δημιουργήσαμε ανάμεσα σε μαθητή και στη mac address της συσκευής του εξασφαλίζει τη μοναδικότητα του.

Σαν αποτέλεσμα ο στόχος της πτυχιακής έχει καλυφθεί στο μέγιστο καθώς έχουμε εισάγει όλα τα στοιχεία που θέλαμε να υπάρχουν μέσα σε αυτή.

ΜΕΛΛΟΝΤΙΚΗ ΕΞΕΛΙΞΗ

Κάποιες προτάσεις για ενημέρωση της εφαρμογής αυτής και την εξέλιξη της θα μπορούσε να ήταν οι εξής :

- αντί να υπάρχει ένα κανάλι σύνδεσης τύπου peer to peer, δηλαδή να συνδέεται κάθε φορά μόνο ένας χρήστης με το καθηγητή για την εισαγωγή της παρουσίας του, να δημιουργηθεί ένα κανάλι τύπου peers to peer. Έτσι θα είχαν τη δυνατότητα πολλοί χρήστες να συνδεθούν ταυτόχρονα με το καθηγητή και την ίδια στιγμή να εισάγουν τη παρουσία τους στη βάση και θα έπαιρνε λιγότερο χρόνο η διαδικασία.
- μια ακόμα πρόταση που θα μπορούσε να γίνει είναι η εφαρμογή να ειδοποιεί τους μαθητές για το πλήθος των απουσιών τους. Με αυτό το τρόπο θα είχαν τη δυνατότητα να είναι ενήμεροι σε περίπτωση που ένα μάθημα είχε όριο απουσιών.
- επιπλέον θα μπορούσε να υπάρξει μια αναβάθμιση στην εμφάνιση της εφαρμογής όπως για παράδειγμα εισαγωγή ενός λογότυπου και διαχείριση των χρωμάτων της
- μια ακόμα μελλοντική εξέλιξη που θα έκανε πιο προσιτή και ακόμα πιο εύκολη τη χρήση της εφαρμογής θα ήταν η δημιουργία ενός user guide ή εμφάνιση περισσότερων μηνυμάτων για την απλούστευση της σε ένα μέσο χρήστη.

ΠΡΟΒΛΗΜΑ ΠΟΥ ΑΝΤΙΜΕΤΩΠΙΣΤΗΚΕ

Κατά τη δημιουργία της εφαρμογής, πέρα από τις αποφάσεις που έπρεπε να ληφθούν για τη μέθοδο υλοποίησής της, αντιμετωπίστηκαν και τεχνικές δυσκολίες όπως για παράδειγμα η αδυναμία του emulator να μπορέσει να τρέξει την εφαρμογή. Αυτό συνέβαινε επειδή τα εικονικά κινητά που έδινε το πρόγραμμα σαν συσκευές για δοκιμή των εφαρμογών δεν είχαν υποστήριξη του Wi-Fi direct οπότε η εφαρμογή έπρεπε να δοκιμαστεί σε πραγματική συσκευή κινητού με άδειες προγραμματιστή. Οπότε

κάθε φορά που προχωρούσαμε στο κώδικα και θέλαμε να τρέξουμε την εφαρμογή για τις αλλαγές, έπρεπε να γίνει σύνδεση δυο συσκευών με το android studio με άδειες προγραμματιστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

https://en.wikipedia.org/wiki/Software_framework

https://el.wikipedia.org/wiki/Android

https://en.wikipedia.org/wiki/Android_Studio

https://firebase.google.com/

https://hackr.io/blog/what-is-frameworks

https://www.android.com/

https://developer.android.com/

https://developer.android.com/studio