



**ΤΜΗΜΑ ΜΗΧ. Η/Υ & ΠΛΗΡΟΦΟΡΙΚΗΣ
ΠΑΝΕΠΙΣΤΗΜΙΟ ΙΩΑΝΝΙΝΩΝ**

**DEPARTMENT OF COMPUTER SCIENCE & ENGINEERING
UNIVERSITY OF IOANNINA**



**Διπλωματική εργασία
«Υλοποίηση εφαρμογής ανέπαφης
ταυτοποίησης»**

Επιβλέπων Καθηγητής: Νικόλαος Μαμουλής

Σπουδαστής: Βασίλης Καρακίτσος

Ιωάννινα, Μάρτιος, 2022

Ευχαριστίες

Ευχαριστώ ιδιαίτερα την οικογένεια μου και την Δήμητρα για την στήριξή τους, και τον επιβλέπων καθηγητή Νικόλαο Μαμουλή, που μου έδωσε την ευκαιρία να διεκπεραιώσω αυτή την εργασία.

3/2022

Βασίλης Καρακίτσος

Περιγραφή

Η ταυτοποίηση ενός φυσικού προσώπου γίνεται συνήθως με την επίδειξη μίας κάρτας (ταυτότητα, πάσο, κλπ.). Στην εργασία αυτή αναπτύχθηκε εφαρμογή κινητού τηλεφώνου για λειτουργικά συστήματα Android και iOS, μέσω της οποίας ένας χρήστης μπορεί να εισάγει τα δεδομένα ταυτότητας του (ονοματεπώνυμο, αριθμού μητρώου, φωτογραφία) και κατόπιν θα μπορεί ο χρήστης να επιδείξει στον ελεγκτή μέσω του κινητού του την ταυτότητα του. Ο ελεγκτής θα μπορεί να πιστοποιήσει την εγκυρότητα σαρώνοντας από το κινητό του έναν μοναδικό QR κωδικό για το χρήστη, ο οποίος παράγεται εκείνη τη στιγμή εγγραφής του χρήστη στο σύστημα. Επιθυμητά αποτελέσματα είναι η διευκόλυνση τόσο των ελεγκτών όσο και των χρηστών αλλά και η υγειονομική ασφάλεια που παρέχει η ανέπαφη ταυτοποίηση. Για να παράγουμε, αλλά και να διατηρήσουμε τα δεδομένα της εφαρμογής, χρησιμοποιήσαμε την πλατφόρμα *“Google Firebase”*. Για την υλοποίηση σε λειτουργικό σύστημα Android, χρησιμοποιήσαμε το *“Android Studio”*, με γλώσσες προγραμματισμού *“Kotlin”* και *“HTML”*. Για την υλοποίηση σε λειτουργικό σύστημα χρησιμοποιήσαμε το *“XCode”* με γλώσσα προγραμματισμού *“Swift”*.

Λέξεις κλειδιά: Android, iOS, Firebase, Kotlin, Swift

Abstract

The identification of a person is usually done by showing a card (identification card, school pass, etc.). In this thesis, a mobile phone application for Android and iOS operating systems was developed, through which a user can enter his identification data (full name, school ID number, profile image) and then the user will be able to show his data to the auditor via his mobile phone. The auditor will be able to certify the validity of the data by scanning from his mobile phone a unique QR code for the user, which is generated at the time of registration of the user in the system. Desirable results are the convenience of both auditors and users, as well as the health and safety provided by contactless identification. In order to create and preserve user data, we used "*Google Firebase*". We implemented our application in Android using "*Android Studio*" with the languages "*Kotlin*" and "*HTML*". For the iOS implementation we used "*XCode*" with the language "*Swift*".

Keywords: Android, iOS, Firebase, Kotlin, Swift

Περιεχόμενα

Κεφάλαιο 1. Εισαγωγή	8
1.1 Αντικείμενο της διπλωματικής	8
1.2 Περιεχόμενο εφαρμογής	9
1.3 Σύγκριση	10
Κεφάλαιο 2. Ανάλυση και Σχεδίαση	11
2.1 Αρχιτεκτονική	11
2.2 Use Cases	12
2.3 Διεπαφή χρήστη-εφαρμογής	18
2.3.1 <i>Android</i>	18
2.3.2 <i>iOS</i>	18
2.3.3 <i>Το γραφικό περιβάλλον χρήστη της εφαρμογής μας</i>	19
Κεφάλαιο 3. Υλοποίηση	26
3.1 Google Firebase	26
3.1.1 <i>Τι είναι το Google Firebase</i>	26
3.1.2 <i>Firebase Authentication</i>	28
3.1.3 <i>Firestore Database</i>	29
3.1.4 <i>Cloud Storage</i>	31
3.2 Υλοποίηση Android	33
3.2.1 <i>Γενική περιγραφή</i>	33
3.2.2 <i>Περιγραφή κλάσεων</i>	34
3.3 Υλοποίηση iOS	36
3.3.1 <i>Γενική περιγραφή</i>	36
3.3.2 <i>Περιγραφή αρχείων</i>	37
3.4 Testing	40
3.4.1 <i>Firebase Test Lab</i>	40
3.4.2 <i>Espresso</i>	44
Κεφάλαιο 4. Επίλογος	45
4.1 Ενημέρωση χρηστών	45
4.2 Ενημέρωση βάσης δεδομένων	45

4.3	Σύνοψη και συμπεράσματα.....	46
	Βιβλιογραφία.....	48

Κεφάλαιο 1. Εισαγωγή

Σε αυτό το κεφάλαιο θα αναλύσουμε το αντικείμενο της διπλωματικής, θα περιγράψουμε την εφαρμογή και θα αναζητήσουμε παρόμοιες ήδη υπάρχουσες εφαρμογές.

1.1 Αντικείμενο της διπλωματικής

Μια εφαρμογή έχει σχεδιαστεί για να λειτουργεί σε μια φορητή συσκευή όπως ένα έξυπνο τηλέφωνο, μια ταμπλέτα ή ένα έξυπνο ρολόι. Αρχικά οι εφαρμογές προορίζονταν ως συμπληρώματα της παραγωγικότητας, καθώς οι πρώτες εφαρμογές παρείχαν υπηρεσίες ηλεκτρονικού ταχυδρομείου, ημερολογίων και επικοινωνιακών βάσεων δεδομένων. Η ζήτηση του κοινού για εφαρμογές επέκτεινε το πεδίο εφαρμογής τους και σε άλλους τομείς, όπως: παιχνίδια, αυτοματισμοί εργοστασίων, εφαρμογές γεωεντοπισμού και υπηρεσίες βασισμένες στην τοποθεσία, αγοράς εισιτηρίων.

Ιδιαίτερα στις ημέρες μας, τις ημέρες της πανδημίας COVID-19, γίνεται λόγος για μία ασφαλή ζωή, με όσο το δυνατόν λιγότερες δημόσιες επαφές. Στη σχολή μας, στις εξεταστικές περιόδους συνηθίζεται η ταυτοποίηση των φοιτητών να γίνεται μέσω ενός χαρτιού το οποίο μοιράζεται σε όλους τους υποψήφιους, και το οποίο πρέπει ο καθένας να υπογράψει, ώστε να δηλωθεί η παρουσία του. Επίσης στα μαθήματα οι φοιτητές καλούνται να υποδείξουν την ακαδημαϊκή τους ταυτότητα, για τον σκοπό της ταυτοποίησης τους. Έτσι δημιουργήθηκε η απαίτηση μίας ανέπαφης ταυτοποίησης από το ακαδημαϊκό προσωπικό. Αυτή την απαίτηση προσπαθήσαμε να καλύψουμε με την εφαρμογή μας.

Η εφαρμογή μας σκοπό έχει να εξαλείψει την ανάγκη επίδειξης της ακαδημαϊκής ταυτότητας και συμπλήρωση εγγράφων που διακινούνται μεταξύ εκατοντάδων φοιτητών. Η εφαρμογή δίνει την δυνατότητα στον φοιτητή, να κάνει την είσοδο του με τα προσωπικά του στοιχεία στην εφαρμογή. Μετά την επιτυχημένη είσοδο του μπορεί να δει τα στοιχεία του, καθώς και τον μοναδικό QR κωδικό που δημιουργήθηκε με βάση τον λογαριασμό του, μέσω του οποίου θα γίνεται και η ταυτοποίηση του. Ο QR κωδικός, μπορεί να σαρωθεί, μόνο από την εφαρμογή μας. Με την επιτυχημένη σάρωση του, εμφανίζονται τα στοιχεία τα οποία αντιπροσωπεύει.

1.2 Περιεχόμενο εφαρμογής

Όπως αναλύσαμε παραπάνω, ο σκοπός της εφαρμογής είναι η διευκόλυνση στην ανέπαφη ταυτοποίηση των φοιτητών. Παρακάτω δείχνουμε τις λειτουργίες που έχουμε υλοποιήσει στην εφαρμογή μας:

- *“Register”*: Ο φοιτητής μπορεί να εγγραφτεί στο σύστημα, εισάγοντας την διεύθυνση *email* του, τον κωδικό του, το ονοματεπώνυμό του, τον αριθμό μητρώου του και επιλέγοντας μία φωτογραφία προφίλ.
- *“Log in”*: Ο φοιτητής μπορεί να εισέλθει στο σύστημα, εισάγοντας την διεύθυνση *email* του και τον κωδικό του.
- *“Reset Password”*: Ο φοιτητής μπορεί να επαναφέρει τον κωδικό του.
- *“Profile”*: Ο φοιτητής μπορεί να δει τα προσωπικά του στοιχεία, όπως αποθηκεύτηκαν στο σύστημα, καθώς και τον μοναδικό *QR* κωδικό που δημιουργήθηκε βάσει του λογαριασμού του.
- *“Log out”*: Ο φοιτητής μπορεί να αποσυνδεθεί από το σύστημα.
- *“Scan”*: Ο χρήστης μπορεί να σαρώσει οποιονδήποτε συμβατό *QR* κωδικό. Δηλαδή μόνο αυτούς που δημιουργήθηκαν από την εφαρμογή.

1.3 Σύγκριση

Η εφαρμογή καλύπτει το κενό της ανάγκης για ανέπαφη ταυτοποίηση. Με μία αναζήτηση στο διαδίκτυο, δεν βρήκαμε κάποια εφαρμογή που να πληροί τις προϋποθέσεις μας. Οι εφαρμογές ανέπαφης ταυτοποίησης που υπάρχουν είναι αυτές για ανέπαφες πληρωμές από τράπεζες, και η εφαρμογή του υπουργείου για το πιστοποιητικό εμβολιασμό/νόσησης, κατά την πανδημία *COVID-19*. Καθώς η εφαρμογή προορίζεται προς χρήση εντός της σχολής ή και του πανεπιστημίου Ιωαννίνων, έχουμε το μεγάλο πλεονέκτημα της επεκτασιμότητας. Η υλοποίηση της εφαρμογής είναι αρκετά απλή και βοηθάει στην μελλοντική επέκταση της, ακόμη και από ανθρώπους που δεν έχουν προγραμματιστικές γνώσεις, λόγω της βάσης δεδομένων που χρησιμοποιούμε. Το *Google Firebase*, προσφέρει γραφικό περιβάλλον, για την εύκολη μετάλλαξη της εφαρμογής.

Κεφάλαιο 2. Ανάλυση και Σχεδίαση

2.1 Αρχιτεκτονική

Η αρχιτεκτονική που ακολουθήσαμε είναι δύο επιπέδων, back-end και front-end. Για το back-end μέρος, χρησιμοποιήσαμε τις υπηρεσίες του “*Google Firebase*” και βιβλιοθήκες για την δημιουργία και ανάκτηση πληροφορίας από τον κωδικό QR. Για το front-end μέρος, δημιουργήσαμε κλάσεις και μεθόδους, που εξυπηρετούν τον σκοπό δημιουργίας ενός φιλικού περιβάλλοντος χρήστη. Η επικοινωνία μεταξύ των δύο επιπέδων είναι αμφίδρομη, καθώς το front-end στέλνει αιτήματα προς το back-end για αποθήκευση και ανάκτηση πληροφοριών, και το back-end ενημερώνει για τα αποτελέσματα που έχει.

2.2 Use Cases

1. Use Case: RegisterUser

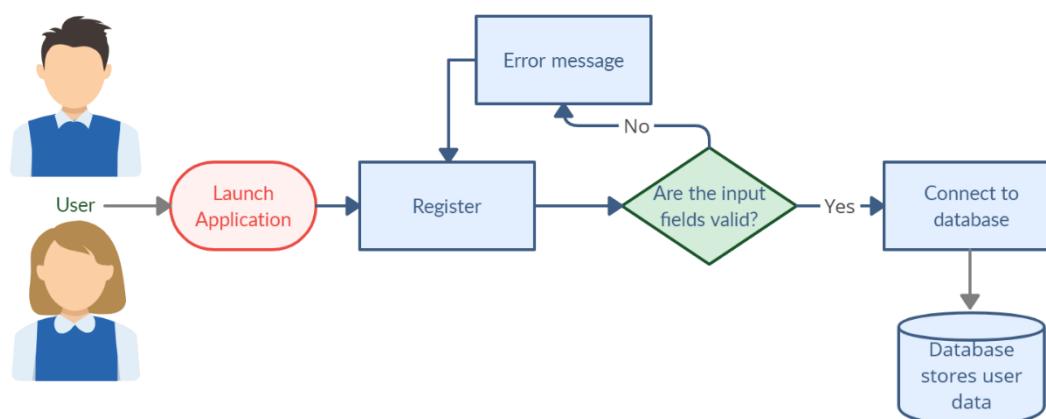
Περιγραφή: Ο χρήστης θέτοντας τα στοιχεία του στα πεδία, θέλει να δημιουργήσει λογαριασμό.

Στόχος: Η επιτυχής εγγραφή του χρήστη στο σύστημα.

Ηθοποιοί: Χρήστης

Ροή γεγονότων:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί “Log in”.
3. Ο χρήστης πατάει το κουμπί “Register”.
4. Ο χρήστης συμπληρώνει το πεδίο φωτογραφίας, εφόσον επιλέξει φωτογραφία, πρέπει να εμφανιστεί το αποτέλεσμα στην οθόνη του.
5. Ο χρήστης συμπληρώνει το πεδίο “Email”, εφόσον έχει εισάγει διεύθυνση με λανθασμένη διαμόρφωση, θα εμφανιστεί μήνυμα λάθους κατά την εγγραφή του.
6. Ο χρήστης συμπληρώνει το πεδίο “Password”, εφόσον έχει εισάγει κωδικό μήκους μικρότερου από 6 χαρακτήρες, θα εμφανιστεί μήνυμα λάθους κατά την εγγραφή του.
7. Ο χρήστης συμπληρώνει το πεδίο “Full name”.
8. Ο χρήστης συμπληρώνει το πεδίο “School ID”.
9. Ο χρήστης πατάει το κουμπί “Register”.
10. Εφόσον όλα τα πεδία είναι σωστά συμπληρωμένα, εμφανίζεται μήνυμα επιτυχίας.



Εικόνα 1: Διάγραμμα ροής δημιουργίας λογαριασμού

2. Use Case: LoginUser

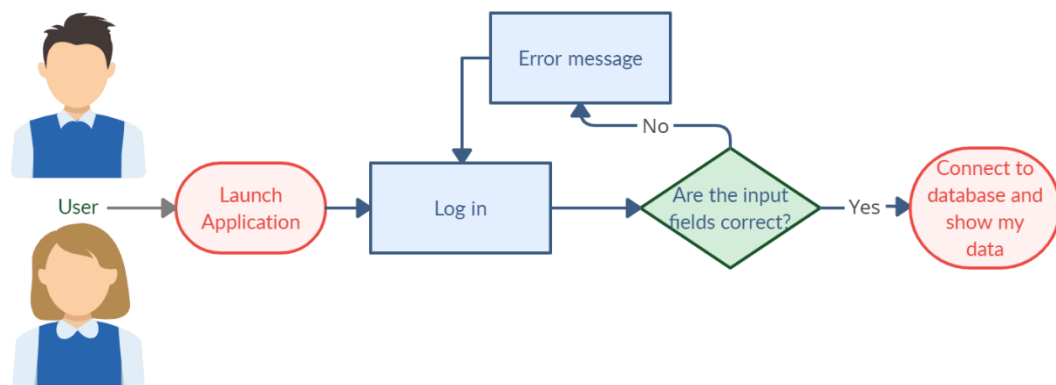
Περιγραφή: Ο χρήστης θέτοντας τα στοιχεία του στα πεδία, θέλει να συνδεθεί στο σύστημα.

Στόχος: Η επιτυχής σύνδεση και εμφάνιση των στοιχείων του χρήστη στο σύστημα.

Ηθοποιοί: Χρήστης

Ροή γεγονότων:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί “Log in”.
3. Ο χρήστης συμπληρώνει το πεδίο “Email”, εφόσον έχει εισάγει διεύθυνση με λανθασμένη διαμόρφωση, θα εμφανιστεί μήνυμα λάθους κατά την είσοδο του.
4. Ο χρήστης συμπληρώνει το πεδίο “Password”, εφόσον έχει εισάγει κωδικό μήκους μικρότερου από 6 χαρακτήρες, θα εμφανιστεί μήνυμα λάθους κατά την είσοδο του.
5. Ο χρήστης πατάει το κουμπί “Log in”.
6. Εφόσον όλα τα πεδία είναι σωστά συμπληρωμένα, εμφανίζεται μήνυμα επιτυχίας και εμφανίζεται η οθόνη προφίλ με τα στοιχεία του χρήστη.



Εικόνα 2: Διάγραμμα ροής σύνδεσης χρήστη

3. Use Case: ResetPassword

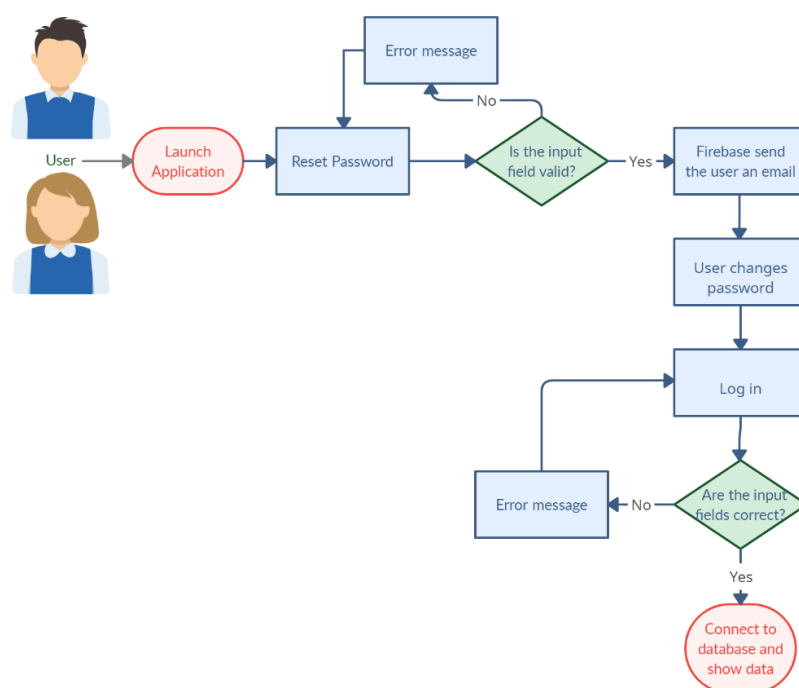
Περιγραφή: Ο χρήστης θέτοντας τον λογαριασμό email με τον οποίο έχει εγγραφεί στο σύστημα, θέλει να επαναφέρει τον κωδικό του.

Στόχος: Η επιτυχής επαναφορά κωδικού του χρήστη.

Ηθοποιοί: Χρήστης

Ροή γεγονότων:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί “Log in”.
3. Ο χρήστης πατάει το κουμπί “Forgot Password?”.
4. Ο χρήστης συμπληρώνει το πεδίο “Email”, εφόσον έχει εισάγει διεύθυνση με λανθασμένη διαμόρφωση, θα εμφανιστεί μήνυμα λάθους κατά την είσοδο του.
5. Ένα μήνυμα πρέπει να έχει αποσταλεί στην διεύθυνση email του χρήστη. Ο χρήστης μπορεί να το διαβάσει και να πατήσει τον σύνδεσμο επαναφοράς κωδικού. Εκεί ο χρήστης μπορεί να επιλέξει τον νέο του κωδικό. Εφόσον έχει σωστή διαμόρφωση, εμφανίζεται μήνυμα επιτυχίας.
6. Ο χρήστης συμπληρώνει τα πεδία εισόδου.
7. Ο χρήστης πατάει το κουμπί “Log in”.
8. Εφόσον όλα τα πεδία είναι σωστά συμπληρωμένα, εμφανίζεται μήνυμα επιτυχίας και εμφανίζεται η οθόνη προφίλ με τα στοιχεία του χρήστη.



Εικόνα 3: Διάγραμμα ροής επαναφοράς κωδικού

4. Use Case: ShowMyProfile

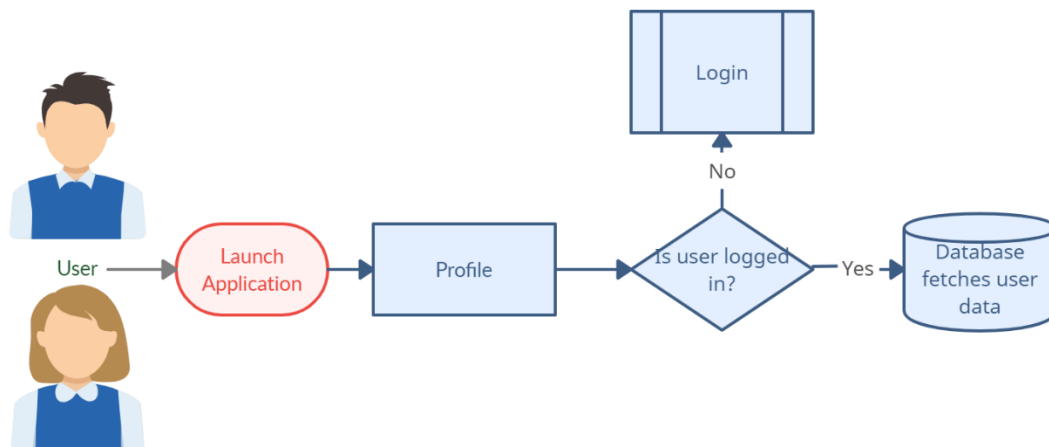
Περιγραφή: Ο χρήστης ανοίγοντας την εφαρμογή θέλει να δει τα στοιχεία του.

Στόχος: Η σωστή εμφάνιση των στοιχείων του χρήστη.

Ηθοποιοί: Χρήστης

Ροή γεγονότων:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί “Profile”.
3. Αν ο χρήστης προηγουμένως είχε συνδεθεί στο σύστημα, τα στοιχεία του εμφανίζονται επιτυχώς.
4. Αν ο χρήστης προηγουμένως δεν είχε συνδεθεί στο σύστημα, τότε μεταφέρεται στην οθόνη “Log in” και ακολουθείται το διάγραμμα ροής “LoginUser”.



Εικόνα 4: Διάγραμμα ροής προβολής προφίλ χρήστη

5. Use Case: LogoutUser

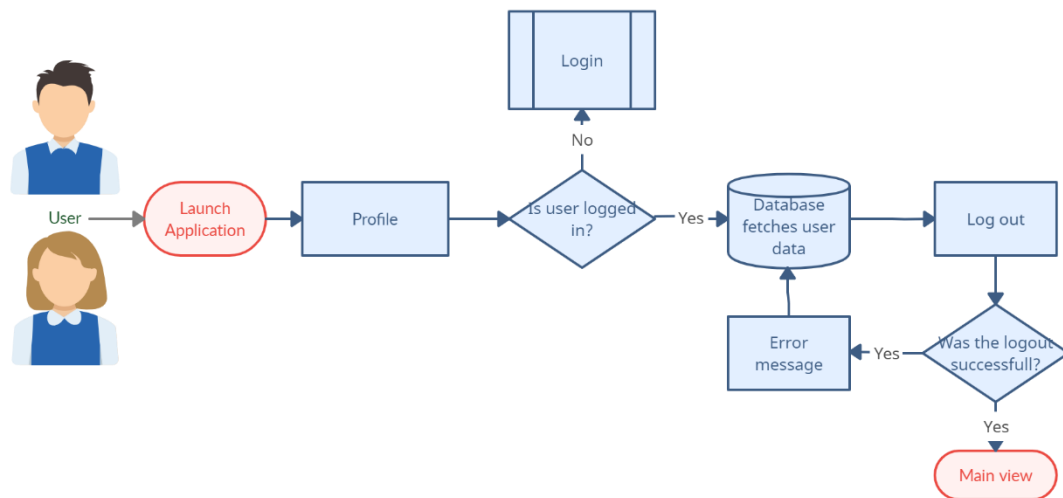
Περιγραφή: Ο χρήστης θέλει να αποσυνδεθεί από το σύστημα.

Στόχος: Η επιτυχής αποσύνδεση του χρήστη.

Ηθοποιοί: Χρήστης

Ροή γεγονότων:

1. Ο χρήστης ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί "Profile".
3. Αν ο χρήστης προηγουμένως είχε συνδεθεί στο σύστημα, τα στοιχεία του εμφανίζονται επιτυχώς.
4. Ο χρήστης πατάει το κουμπί "Log out".
5. Αν είναι επιτυχημένη η αποσύνδεση του, μεταφέρεται στην αρχική οθόνη της εφαρμογής.
6. Αν είναι αποτυχημένη η αποσύνδεση του, εμφανίζεται μήνυμα λάθους.



Εικόνα 5: Διάγραμμα ροής αποσύνδεσης χρήστη

6. Use Case: ScanQR

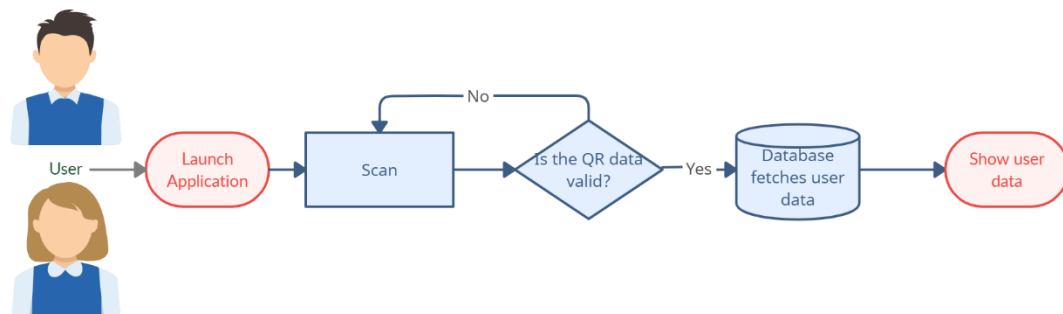
Περιγραφή: Ο ελεγκτής θέλει να σκανάρει έναν κωδικό QR.

Στόχος: Η σωστή εμφάνιση των στοιχείων του χρήστη.

Ηθοποιοί: Ελεγκτής

Ροή γεγονότων:

1. Ο ελεγκτής ανοίγει την εφαρμογή.
2. Ο χρήστης πατάει το κουμπί “Scan”.
3. Εμφανίζεται η οθόνη στην οποία, το άνω μέρος περιέχει τον σαρωτή QR κωδικών και το κάτω μέρος τα στοιχεία ενός χρήστη, τα οποία στην αρχή ορίζονται ως default.
4. Αν ο ελεγκτής σαρώσει QR κωδικό ο οποίος περιέχει την πληροφορία του μοναδικού κλειδιού του χρήστη στο σύστημα, τότε εμφανίζονται με την σειρά το ονοματεπώνυμο του, ο αριθμός μητρώου του και η φωτογραφία του.



Εικόνα 6: Διάγραμμα ροής σάρωσης QR κωδικού

2.3 Διεπαφή χρήστη-εφαρμογής

Σε αυτή την ενότητα θα περιγράψουμε την δημιουργία του περιβάλλοντος του χρήστη και τα εργαλεία που χρησιμοποιήσαμε για την επίτευξη του.

2.3.1 Android

Το περιβάλλον χρήστη για κινητά τηλέφωνα με λειτουργικό σύστημα “Android”, δημιουργήθηκε στο “Android Studio” με τη βοήθεια της “HTML”. Το προγραμματιστικό περιβάλλον του “Android Studio” μας δίνει την δυνατότητα μέσω προεπισκοπήσεων, να βλέπουμε απευθείας στον «καμβά» τις αλλαγές που κάνουμε στο εκάστοτε αρχείο “.xml”. Για κάθε οθόνη της εφαρμογής μας, χρησιμοποιούμε και ένα ξεχωριστό, ανεξάρτητο αρχείο “HTML”, το οποίο ονομάζεται “Activity”. Σε κάθε “Activity”, ορίζεται ένα “Layout”, πάνω στο οποίο μπορούμε να βάλουμε τα αντικείμενα που μας ενδιαφέρουν. Κάθε στοιχείο που υπάρχει σε ένα “Activity”, έχει τα δικά του χαρακτηριστικά. Για παράδειγμα ένα στοιχείο “Button”, έχει το πλάτος, το ύψος, το κείμενο που θα έχει μέσα του, το θέμα που ακολουθεί κλπ.

2.3.2 iOS

Το περιβάλλον χρήστη για κινητά τηλέφωνα με λειτουργικό σύστημα “iOS”, δημιουργήθηκε στο “XCode” με τη βοήθεια της “swiftUI”. Το προγραμματιστικό περιβάλλον του “XCode” μας δίνει την δυνατότητα μέσω προεπισκοπήσεων, να βλέπουμε απευθείας στον «καμβά» τις αλλαγές που κάνουμε στο εκάστοτε αρχείο “.swift”. Για κάθε οθόνη της εφαρμογής μας, χρησιμοποιούμε και ένα ξεχωριστό, ανεξάρτητο αρχείο “swiftUI”, το οποίο ονομάζεται “View”. Σε κάθε “View”, ορίζεται ένα “struct View”, πάνω στο οποίο μπορούμε να βάλουμε τα αντικείμενα που μας ενδιαφέρουν. Κάθε στοιχείο που υπάρχει σε ένα “View”, έχει τα δικά του χαρακτηριστικά. Για παράδειγμα ένα στοιχείο “Text”, έχει το μέγεθος του κειμένου, την γραμματοσειρά, το θέμα κλπ. Στην υποενότητα 1.3.3 θα δείξουμε το περιβάλλον χρήστη.

2.3.3 Το γραφικό περιβάλλον χρήστη της εφαρμογής μας

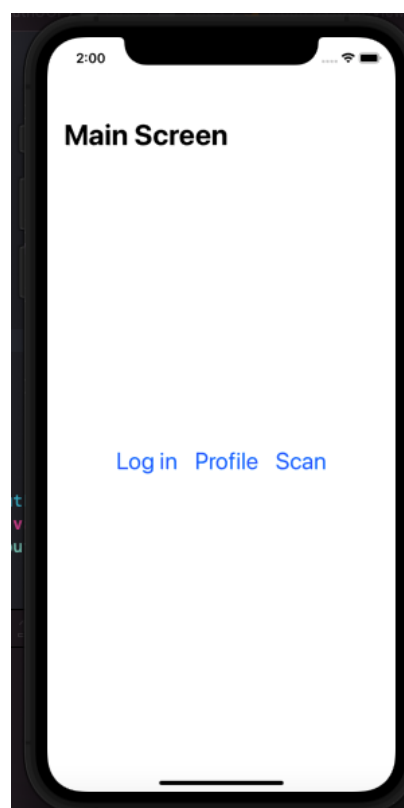
Στην εφαρμογή μας, προσπαθήσαμε να κάνουμε το περιβάλλον χρήστη όσο το δυνατόν πιο παρόμοιο ανάμεσα στα δύο λειτουργικά συστήματα. Καθώς βασιζόμαστε στα *native* γραφικά συστήματα του εκάστοτε λειτουργικού, και όχι σε μία τρίτη βιβλιοθήκη από ανεξάρτητο γραφικό περιβάλλον, αυτό κρίθηκε ικανό.

Στις εικόνες 7/8, φαίνεται τι βλέπει ο χρήστης μόλις ανοίξει την εφαρμογή. Στο κέντρο της οθόνης υπάρχουν τρία κουμπιά. Κάθε ένα εξυπηρετεί την λειτουργία που αναγράφει. Το κουμπί “*Log in*”, μας μεταφέρει στην οθόνη για είσοδο στο σύστημα. Το κουμπί “*Profile*”, μας μεταφέρει στην οθόνη του προφίλ μας, εφόσον έχουμε συνδεθεί προηγουμένως, αλλιώς στην οθόνη εισόδου. Τέλος το κουμπί “*Scan*”, μας μεταφέρει στην οθόνη σάρωσης ενός QR κωδικού.



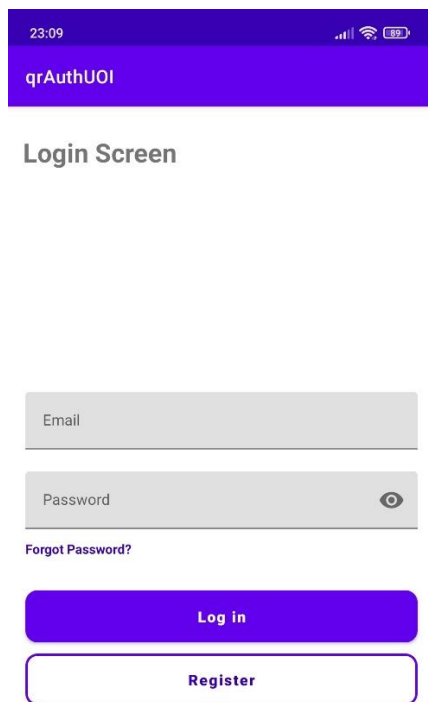
Log in Profile Scan

Εικόνα 7: Η βασική οθόνη σε Android

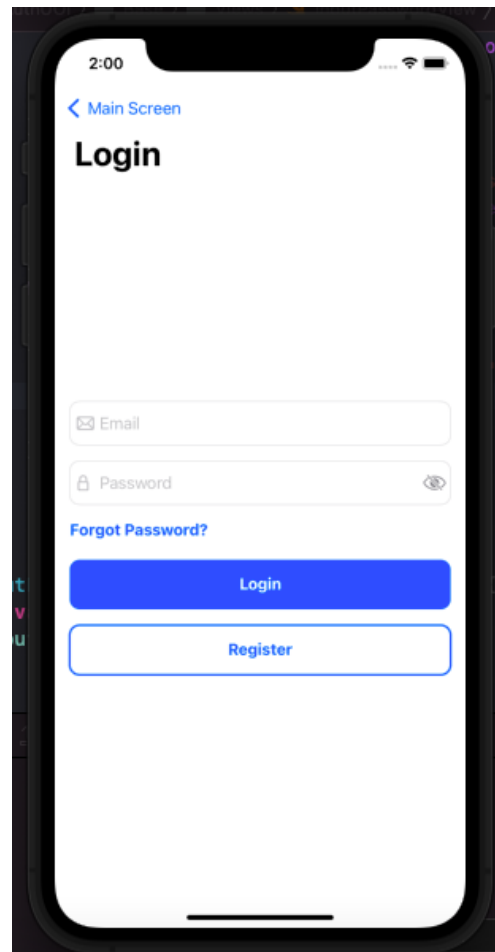


Εικόνα 8: Η βασική οθόνη σε iOS

Στις εικόνες 9/10, βρισκόμαστε στην οθόνη σύνδεσης του χρήστη. Εδώ ένας χρήστη μπορεί να εισάγει τα στοιχεία του, αν θέλει να συνδεθεί στο σύστημα. Αν θέλει να κάνει εγγραφή, πατάει το κουμπί “Register” για να μεταφερθεί στην οθόνη εγγραφής. Αν επιθυμεί να επαναφέρει τον κωδικό του, πατάει το κουμπί “Forgot Password?”, ώστε να μεταφερθεί στην οθόνη επαναφοράς κωδικού. Αν ο χρήστης εισάγει λανθασμένα στοιχεία, θα εμφανιστεί μήνυμα λάθους.

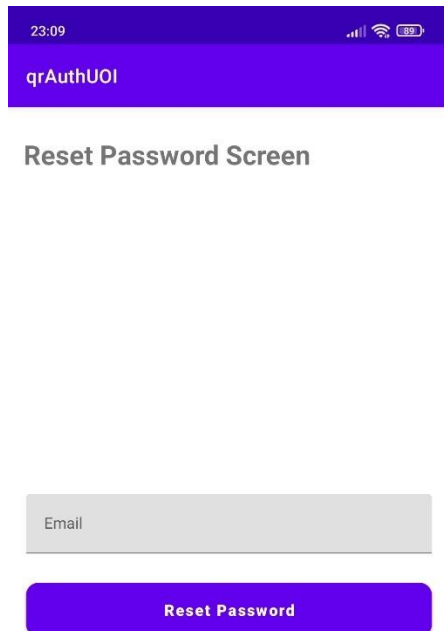


Εικόνα 9: Οθόνη σύνδεσης σε Android

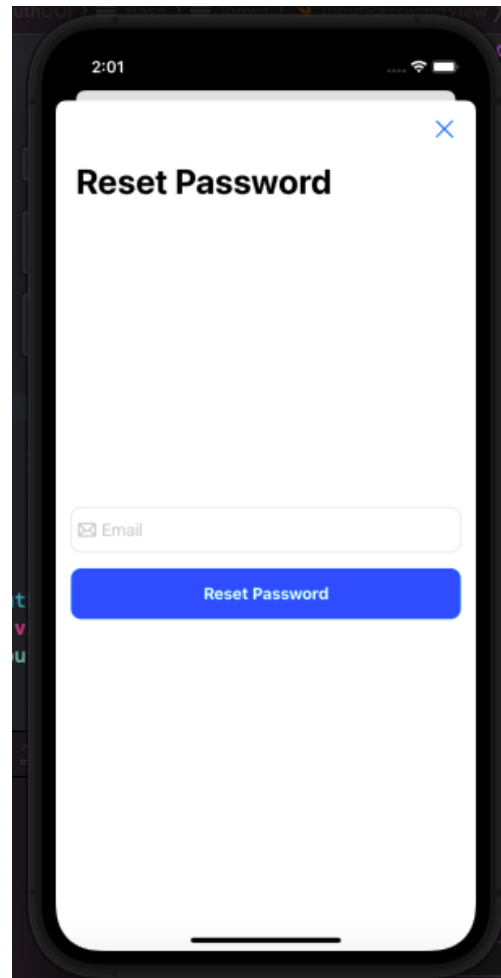


Εικόνα 10: Οθόνη σύνδεσης σε iOS

Στις εικόνες 11/12, βρισκόμαστε στην οθόνη επαναφοράς κωδικού. Εδώ ο χρήστης μπορεί να εισάγει την διεύθυνση *email* του, ώστε να του αποσταλεί ένα μήνυμα, αίτησης επαναφοράς του κωδικού του λογαριασμού του. Αν ο χρήστης εισάγει λανθασμένη διεύθυνση email, θα εμφανιστεί μήνυμα λάθους.

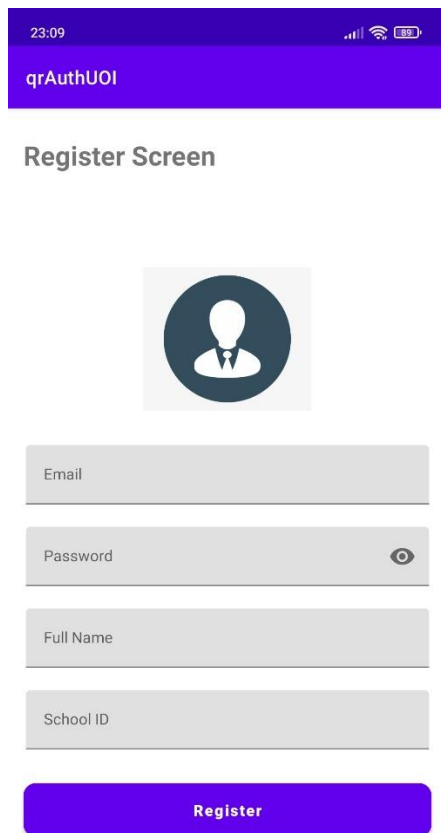


Εικόνα 11: Οθόνη επαναφοράς κωδικού σε Android

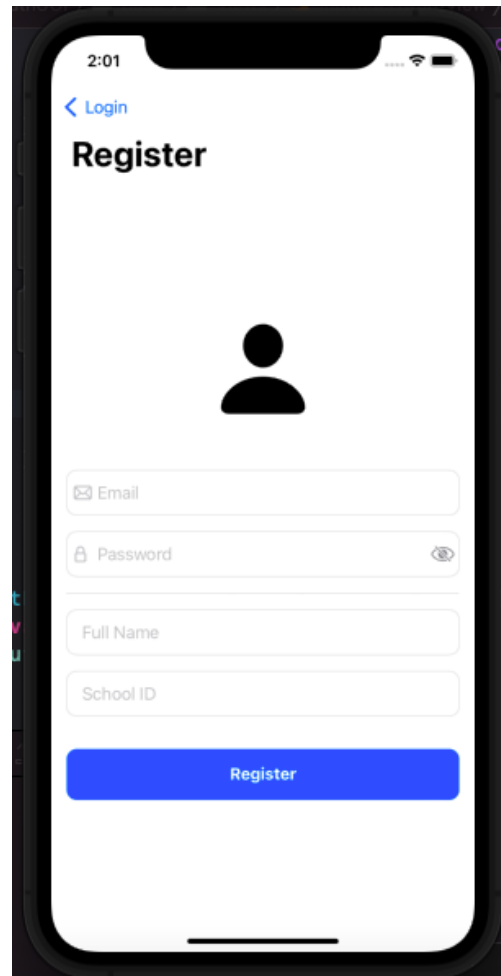


Εικόνα 12: Οθόνη επαναφοράς κωδικού σε iOS

Στις εικόνες 13/14, βρισκόμαστε στην οθόνη εγγραφής του χρήστη στο σύστημα. Εδώ ο χρήστης πατώντας στο εικονίδιο με τον άνθρωπο, μεταφέρεται στην εφαρμογή συλλογής του κινητού του, ώστε να επιλέξει την εικόνα προφίλ του. Εφόσον επιλέξει μία φωτογραφία, αυτή θα εμφανιστεί αντί για τον άνθρωπο. Στην συνέχεια μπορεί να συμπληρώσει τα πεδία, κάθε ένα κατάλληλα, και πατώντας το κουμπί “Register”, μπορεί να δοκιμάσει την εγγραφή του στο σύστημα. Αν έχει συμπληρώσει κατάλληλα τα πεδία, τότε η εγγραφή θα είναι επιτυχής και θα μεταφερθεί στην οθόνη εισόδου. Αλλιώς θα εμφανιστεί μήνυμα λάθους.

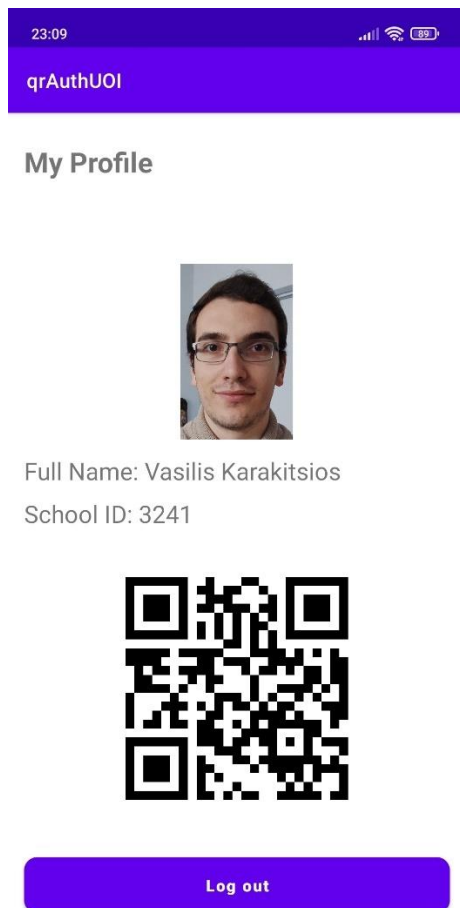


Εικόνα 13: Οθόνη εγγραφής χρήστη σε Android

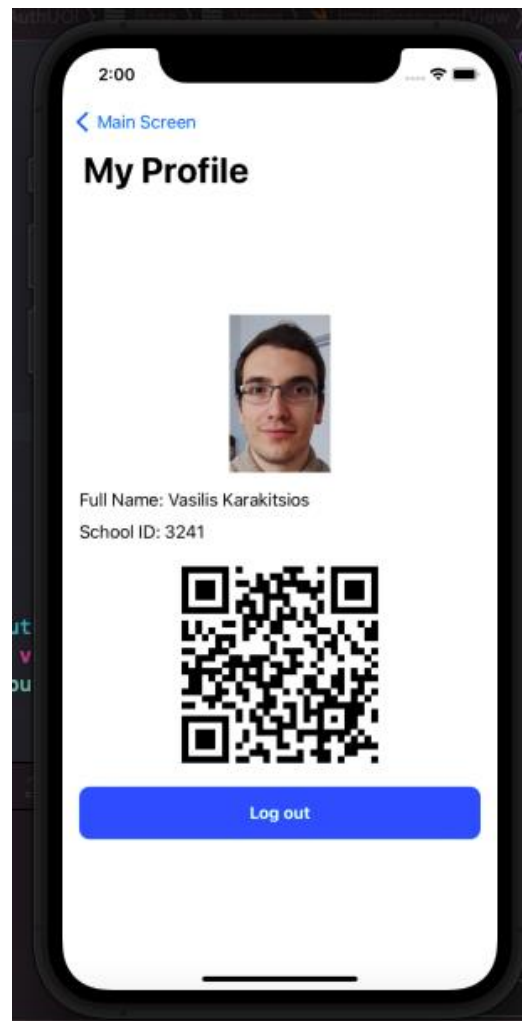


Εικόνα 14: Οθόνη εγγραφής χρήστη σε iOS

Στις εικόνες 15/16, βρισκόμαστε στην οθόνη προφίλ του χρήστη. Εδώ ένας χρήστης μετά την είσοδο του μπορεί να δει τα στοιχεία του, τα οποία ανακτώνται από την βάση δεδομένων. Ο κωδικός QR, δημιουργείται κάθε φορά που μπαίνουμε σε αυτή την οθόνη. Αυτό συμβαίνει, καθώς δε θέλουμε να αποθηκευτεί κάπου η πληροφορία που κωδικοποιεί.

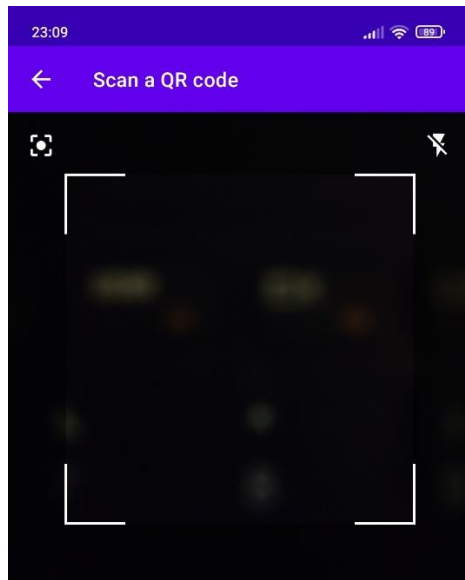


Εικόνα 15: Οθόνη προφίλ σε Android



Εικόνα 16: Οθόνη προφίλ σε iOS

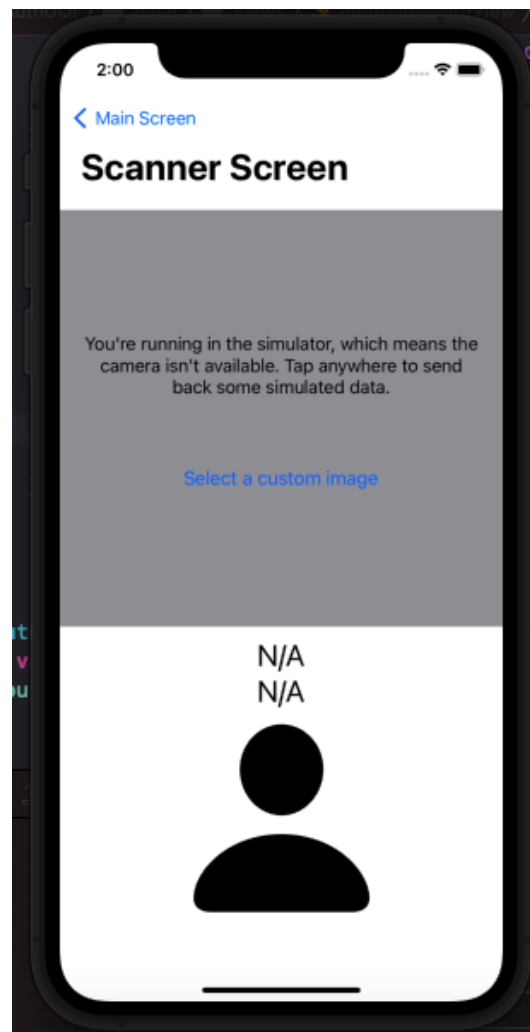
Στις εικόνες 17/18, βρισκόμαστε στην οθόνη σάρωσης QR κωδικού. Αν δεν σαρωθεί σωστός QR κωδικός, δηλαδή κωδικός που δεν αντιστοιχεί σε εγγραφή στην βάση δεδομένων, τότε τα αποτελέσματα είναι τα *default* και δεν ανανεώνονται. Αντίθετα αν βρεθεί εγγραφή στην βάση δεδομένων, που αντιστοιχεί στον κωδικό που σαρώθηκε, τότε τα αποτελέσματα ανανεώνονται κατάλληλα, όπως φαίνεται στις εικόνες 19/20.



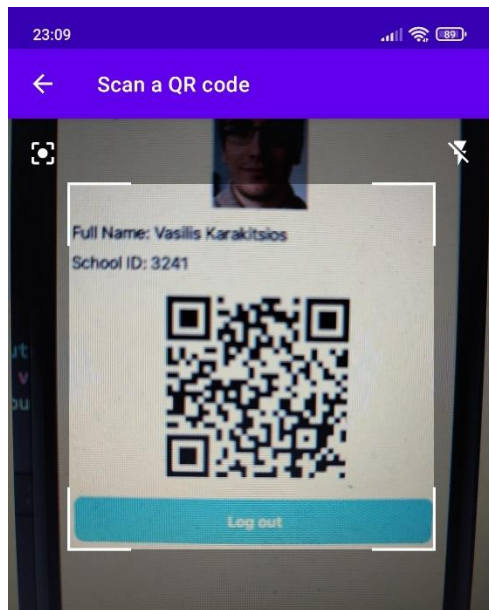
Full Name: N/A
School ID: N/A



Εικόνα 17: Οθόνη σάρωσης QR σε Android



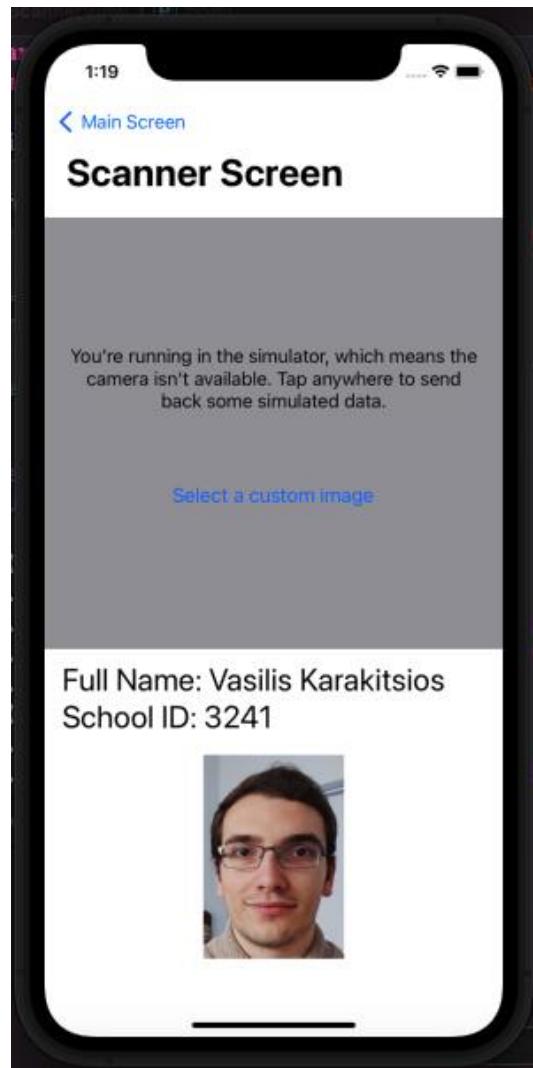
Εικόνα 18: Οθόνη σάρωσης QR σε iOS



Full Name: Vasilis Karakitsios
School ID: 3241



Εικόνα 19: Επιτυχής σάρωση κωδικού QR σε Android



Εικόνα 20: Επιτυχής σάρωση κωδικού QR σε iOS

Κεφάλαιο 3. Υλοποίηση

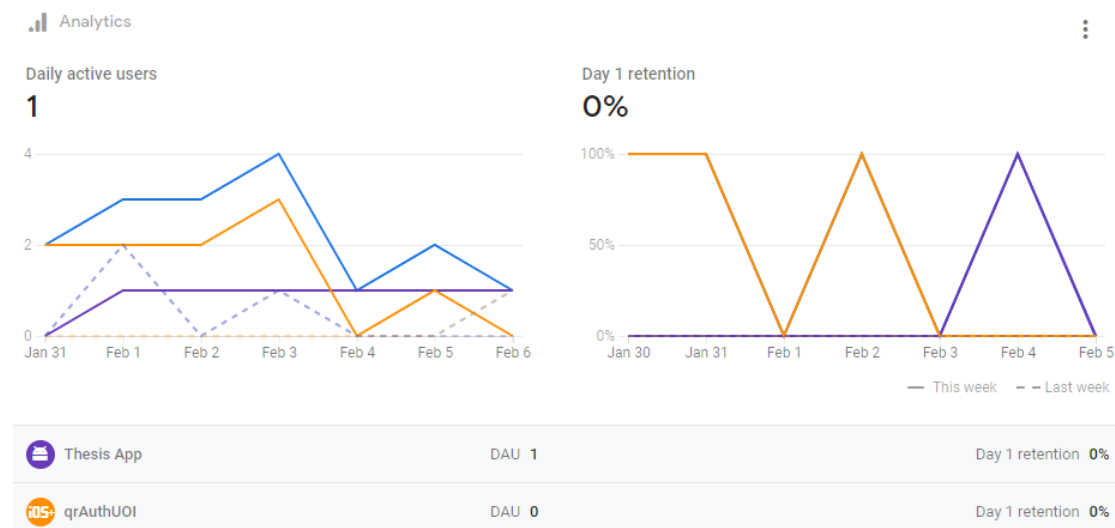
3.1 Google Firebase

Σε αυτή την ενότητα θα αναλύσουμε το “*Google Firebase*” και θα εξηγήσουμε τα εργαλεία του που χρησιμοποιούμε.

3.1.1 Τι είναι το Google Firebase

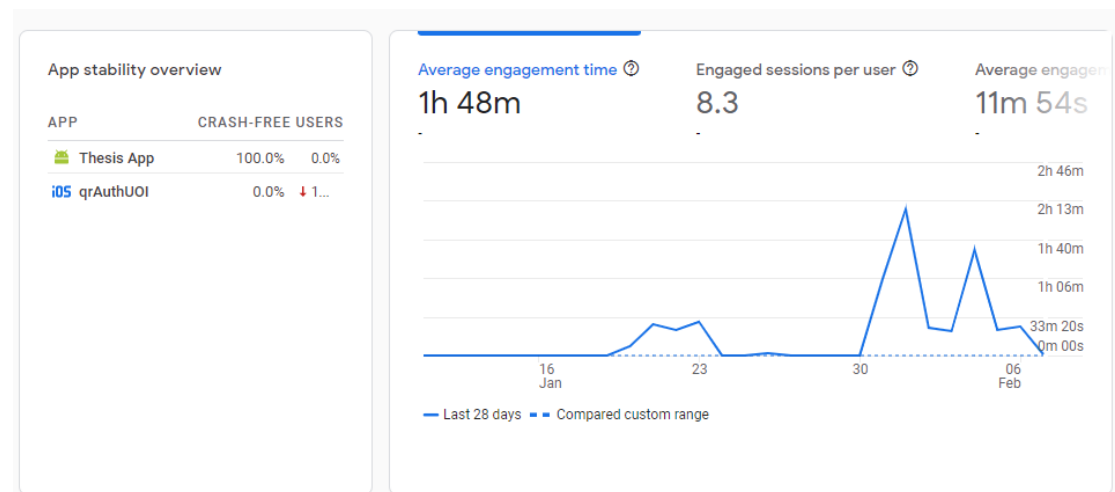
Το Firebase που ιδρύθηκε το 2011 σαν chat API και αποκτήθηκε από την Google το 2014, είναι μία πλατφόρμα που προσφέρει ένα ενεργό “Back-end as a Service” (BaaS) μοντέλο για να δημιουργήσουμε δυναμικές διαδικτυακές και κινητές εφαρμογές. Η ιδέα του Google Firebase είναι απλή. Όταν δημιουργούμε μία εφαρμογή από την πλευρά του client, εισάγοντας το Firebase η εφαρμογή μετατρέπεται σε serverless. Έτσι αφαιρεί την ανάγκη διαχείρισης και συντήρησης του server από εμάς, καθώς το κάνει από μόνο του, και μας απελευθερώνει να επικεντρωθούμε κυρίως στο front-end κομμάτι. Επομένως, προσθέτοντας το Firebase στην εφαρμογή μας, σημαίνει ότι βάζουμε ένα έτοιμο back-end και μπορούμε να το τροποποιήσουμε ακριβώς στις ανάγκες μας, χωρίς να χρειαστεί να δημιουργήσουμε ένα από την αρχή μόνοι μας. Από την άποψη της ασφάλειας, υπάρχουν ρητοί κανόνες που μπορούμε να εισάγουμε, οι οποίοι προστατεύουν το back-end από ανεπιθύμητες εγγραφές ή προβολές.

Όπως φαίνεται στην παρακάτω φωτογραφία, με το “Firebase” μπορούμε να δούμε σε απευθείας χρόνο, τον αριθμό των χρηστών που χρησιμοποιούν την εφαρμογή μας και έχουν εγγραφεί στο σύστημα. Επίσης υπάρχει η δυνατότητα να δούμε πόσοι χρήστες χρησιμοποιούν την εφαρμογή μας σε “Android” συσκευές και πόσοι σε “iOS”.



Εικόνα 21: Μετρικές του Firebase

Επίσης μπορούμε να δούμε τον χρόνο χρήσης της εφαρμογής κατά μέσο όρο και την σταθερότητα της εφαρμογής ανά πλατφόρμα, όπως φαίνεται στην φωτογραφία 22.



Εικόνα 22: Στατιστικά χρόνου και σταθερότητας

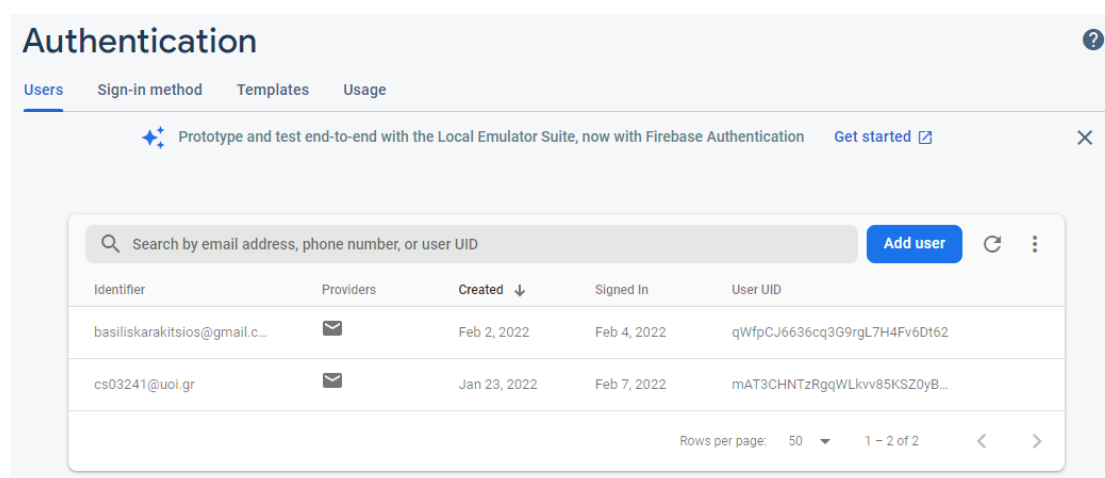
Οι υπηρεσίες του Firebase που χρησιμοποιούμε στην εφαρμογή μας είναι οι εξής:

- Firebase Authentication
- Firestore Database
- Cloud Storage

3.1.2 Firebase Authentication

Η εφαρμογή μας, όπως και οι περισσότερες χρειάζονται τα στοιχεία του χρήστη. Γνωρίζοντας τα στοιχεία του χρήστη, η εφαρμογή μπορεί με ασφάλεια να τα αποθηκεύσει στο cloud και να προσφέρει τις υπηρεσίες της σε όλες τις συσκευές που μπορεί να χρησιμοποιήσει ο χρήστης. Το Firebase Authentication μας προσφέρει back-end υπηρεσίες, έτοιμες βιβλιοθήκες που μπορούμε να χρησιμοποιήσουμε στην εφαρμογή μας. Επίσης ενσωματώνει όλες τις υπηρεσίες του Firebase και ακολουθεί το πρότυπο OAuth 2.0 για την ταυτοποίηση.

Στην εφαρμογή μας ένας χρήστης μπορεί να δημιουργήσει λογαριασμό με email διεύθυνση και κωδικό ασφαλείας. Εφόσον η δημιουργία του χρήστη κριθεί επιτυχημένη, δημιουργείται μία καινούρια εγγραφή στο Firebase Authentication, με ένα μοναδικό κλειδί χρήστη μήκους 28 χαρακτήρων που απαρτίζονται από τα γράμματα του αγγλικού αλφαβήτου και τους αριθμούς του δεκαδικού συστήματος. Έτσι κάθε χαρακτήρας έχει 62 διαφορετικούς τρόπους αναπαράστασης. Τέλος, ένας χρήστης μπορεί να επιλέξει να επαναφέρει τον κωδικό του, αν το επιθυμεί.



The screenshot shows the 'Authentication' section of the Firebase console, specifically the 'Users' tab. It features a search bar at the top with the text 'Search by email address, phone number, or user UID'. Below the search bar is a table with two rows of user data. The table has five columns: 'Identifier', 'Providers', 'Created', 'Signed In', and 'User UID'. The first row shows a user with the email 'basiliskarakitsios@gmail.c...' created on Feb 2, 2022, and signed in on Feb 4, 2022. The second row shows a user with the email 'cs03241@uoi.gr' created on Jan 23, 2022, and signed in on Feb 7, 2022. At the bottom right of the table, there is a 'Rows per page' dropdown set to 50 and a pagination indicator '1 - 2 of 2'.

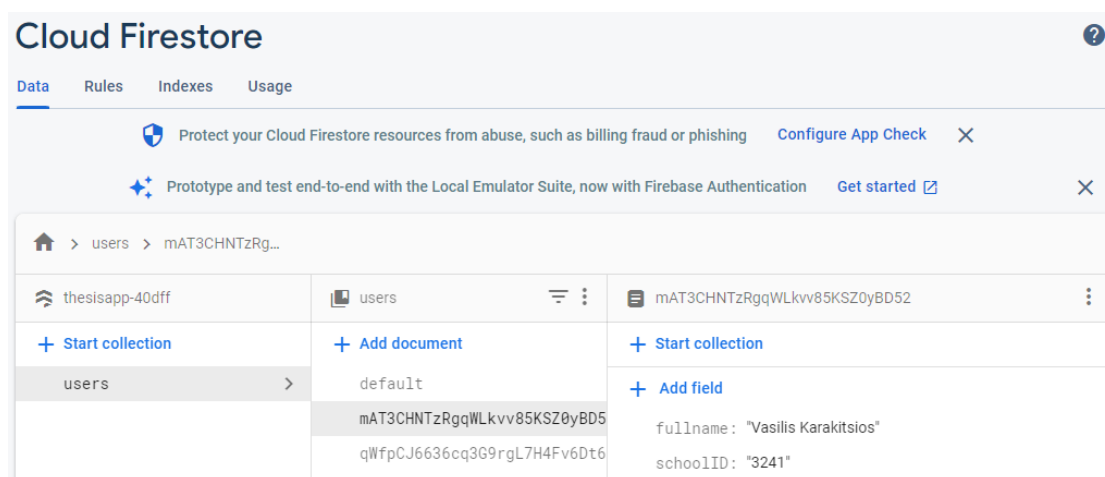
Identifier	Providers	Created ↓	Signed In	User UID
basiliskarakitsios@gmail.c...	📧	Feb 2, 2022	Feb 4, 2022	qWfpCJ6636cq3G9rgL7H4Fv6Dt62
cs03241@uoi.gr	📧	Jan 23, 2022	Feb 7, 2022	mAT3CHNTzRgqWLkv85KSZ0yB...

Εικόνα 23: Πίνακας Authentication με τους εγγεγραμμένους χρήστες

3.1.3 Firestore Database

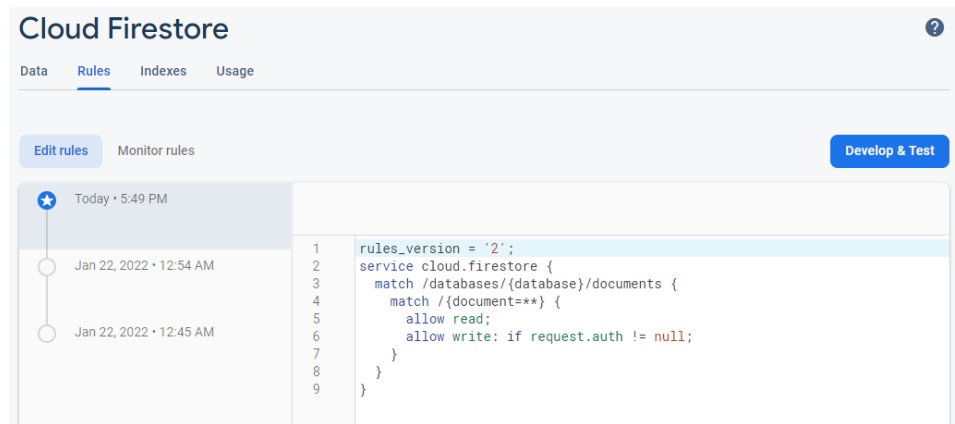
Το Firestore Database είναι μία scalable NoSQL cloud βάση δεδομένων που μας επιτρέπει εύκολα και γρήγορα να αποθηκεύουμε, συγχρονίζουμε και διατρέχουμε τα δεδομένα μας. Μπορούμε να χτίσουμε τα δεδομένα μας εύκολα χρησιμοποιώντας συλλογές και έγγραφα. Με αυτό τον τρόπο τα queries που διατρέχουμε για αποθήκευση, ανάγνωση και εγγραφή, είναι πολύ εύκολα. Ο συγχρονισμός των δεδομένων γίνεται αυτόματα.

Στην εφαρμογή μας, κατά την δημιουργία λογαριασμού του χρήστη, εισάγεται το κατάλληλο έγγραφο υπό την συλλογή “users”, με βάσει το μοναδικό κλειδί που δημιουργήθηκε και με πεδία το ονοματεπώνυμο και τον αριθμό μητρώου του χρήστη.



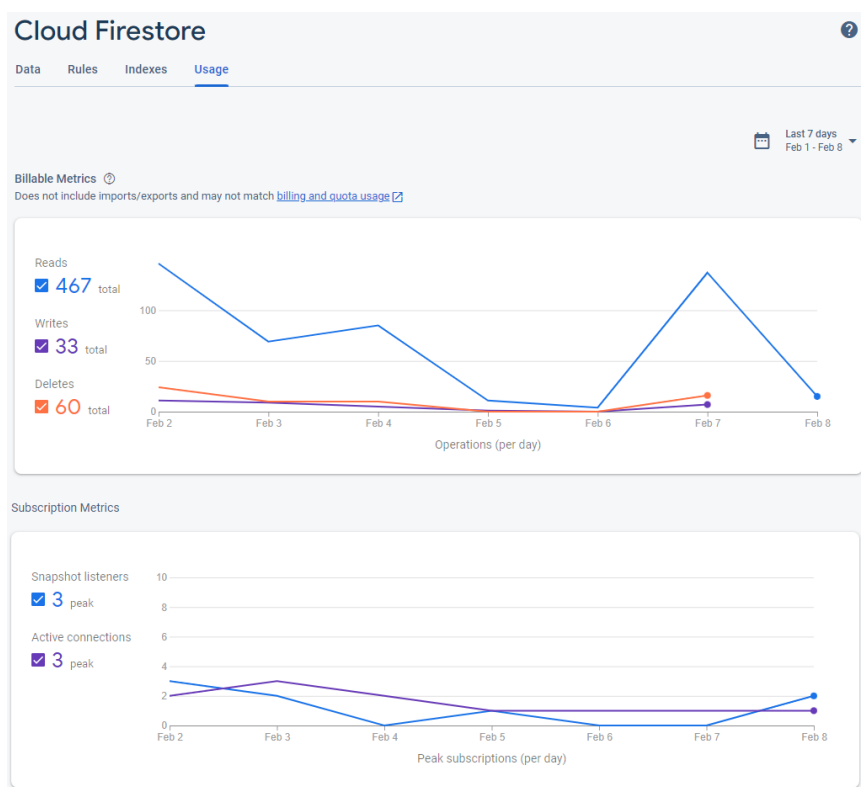
Εικόνα 24: Πίνακας Firestore με τους εγγεγραμμένους χρήστες

Για την ασφάλεια της βάσης, χρησιμοποιούμε τους κανόνες, όπως φαίνονται στην επόμενη φωτογραφία. Μπορεί οποιοσδήποτε να κάνει ανάγνωση, όμως για να κάνει εγγραφή, απαιτείται ο έγκυρος λογαριασμός του στο σύστημα.



Εικόνα 25: Κανόνες ασφάλειας στην βάση δεδομένων

Το “Firestore” μας δίνει την δυνατότητα να δούμε στατιστικές μετρικές του, όπως τον συνολικό αριθμό εγγραφών, προβολών και διαγραφών, που έγιναν στην βάση δεδομένων, σε οποιοδήποτε χρονικό διάστημα επιθυμούμε, όπως φαίνεται στην φωτογραφία 26.

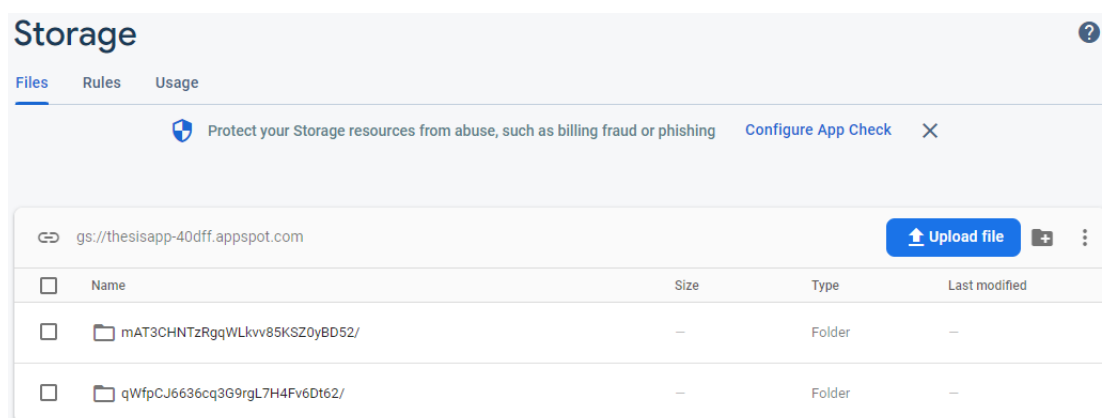


Εικόνα 26: Μετρικές Firestore

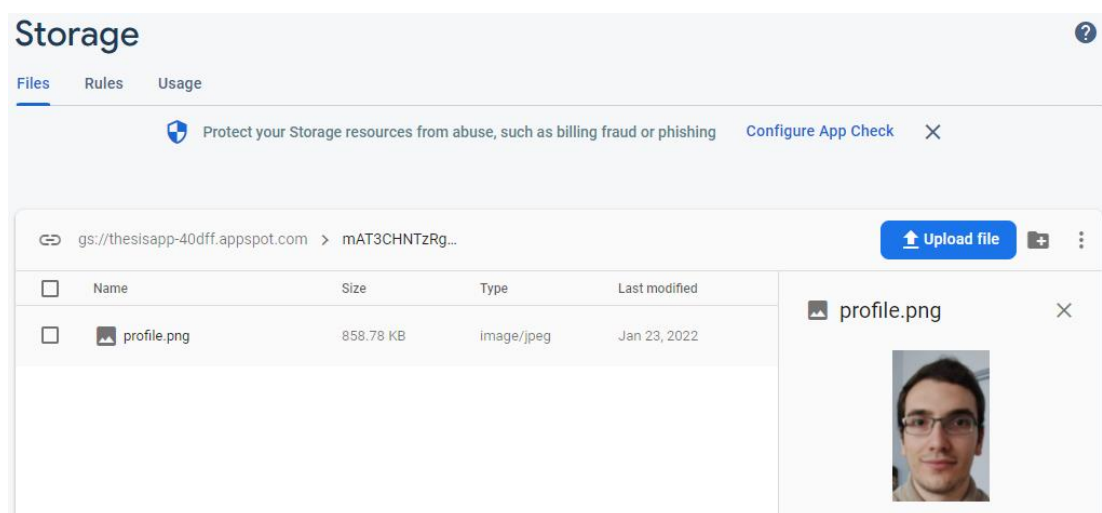
3.1.4 Cloud Storage

Το Cloud Storage δημιουργήθηκε για να εξυπηρετήσει εφαρμογές στις οποίες η πρόσβαση σε αρχεία των χρηστών είναι απαραίτητη. Τα SDK του εξασφαλίζουν ασφάλεια κατά το ανέβασμα-κατέβασμα αρχείων, εφόσον υπάρξει διακοπή σύνδεσης στο internet, σταματάει η διεργασία. Μπορεί να χρησιμοποιηθεί για αποθήκευση φωτογραφιών, αρχεία ήχων, ή οποιοδήποτε άλλο format αρχείων.

Στην εφαρμογή μας, χρησιμοποιούμε το Cloud Storage για το ανέβασμα και την προβολή των φωτογραφιών των χρηστών. Μία φωτογραφία ανεβαίνει στο Cloud Storage στον φάκελο με όνομα το μοναδικό κλειδί του χρήστη, κατά την επιτυχή εγγραφή του χρήστη στο σύστημα. Αυτή η φωτογραφία προβάλλεται στο προφίλ του χρήστη και όταν σκανάρει τον κωδικό QR του, ένας ελεγκτής.

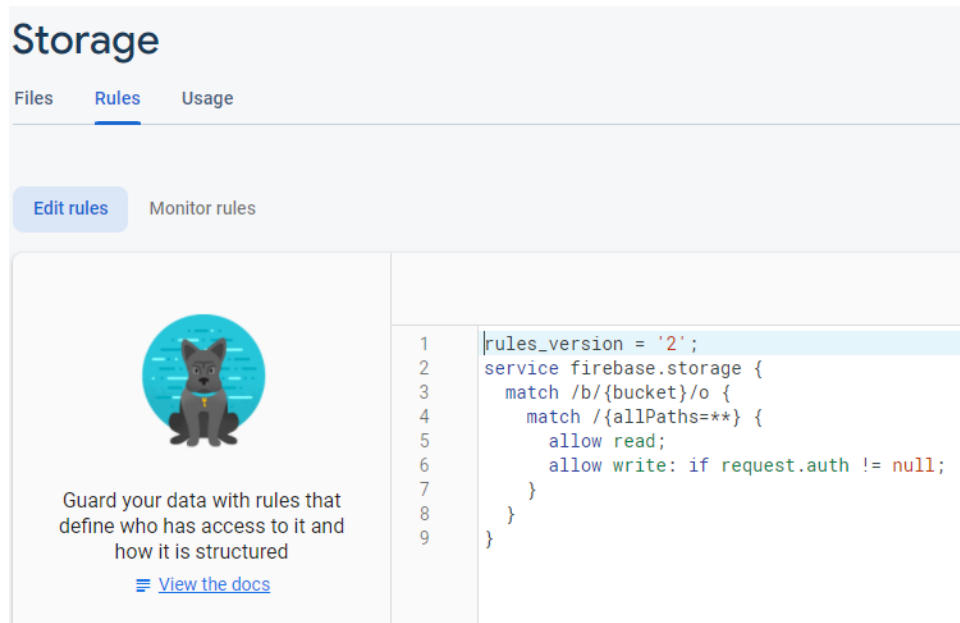


Εικόνα 27: Πίνακας Cloud Storage με τους εγγεγραμμένους χρήστες



Εικόνα 28: Υποκατάλογος χρήστη με την φωτογραφία του

Οι κανόνες ασφαλείας που έχουμε ορίσει στο “Cloud Storage” είναι οι ίδιοι με τους κανόνες στο “Firestore”. Δηλαδή ένας χρήστης μπορεί να κάνει εγγραφή στο σύστημα μόνο αν έχει εγκεκριμένο λογαριασμό.



Εικόνα 29: Κανόνες ασφαλείας στο Cloud Storage

Το “Cloud Storage” μας δίνει την δυνατότητα να μπορούμε να βλέπουμε τις κινήσεις στον χώρο σε ημερήσια βάση. Μπορούμε να δούμε το συνολικό *bandwidth*, τις συνολικές αιτήσεις, το πλήθος των αρχείων που βρίσκονται στο σύστημα, καθώς και το συνολικό μέγεθός τους.



Εικόνα 30: Μετρικές στο Cloud Storage

3.2 Υλοποίηση Android

3.2.1 Γενική περιγραφή

Για την υλοποίηση της εφαρμογής για κινητά τηλέφωνα με λειτουργικό σύστημα Android, δουλέψαμε με το *“Android Studio”*. Χρησιμοποιήσαμε τη γλώσσα προγραμματισμού *“Kotlin”* για την υλοποίηση κλάσεων, μεθόδων κλπ. Και την γλώσσα *“HTML”* για την υλοποίηση του περιβάλλοντος χρήστη. Η ελάχιστη απαιτούμενη έκδοση Android για την σωστή λειτουργία της εφαρμογής είναι η *“4.5 Lollipop”*. Το UI σχεδιάστηκε με βάση το πρότυπο *“Material”* της Google, ώστε να είναι απλό και φιλικό προς τον χρήστη. Για την υλοποίηση των απαιτήσεων όσον αφορά την δημιουργία και σάρωση του κωδικού QR, χρησιμοποιήσαμε την βιβλιοθήκη ανοιχτού κώδικα *“ZXing”* (Zebra crossing) και *“Code Scanner”* που είναι βασισμένη στην πρώτη. Τις υπόλοιπες λειτουργίες τις δημιουργήσαμε είτε με την βοήθεια του *“Google Firebase”* είτε από την αρχή. Έχουμε δημιουργήσει τον κώδικα τμηματικά σε κλάσεις. Κάθε κλάση είναι υπεύθυνη για το View της. Δηλαδή κάθε οθόνη που εμφανίζεται, είναι υλοποιημένη από μία ξεχωριστή κλάση. Κάθε κλάση αντιστοιχίζεται σε ένα αρχείο HTML, το οποίο και αξιοποιούν οι μέθοδοί της για την σωστή διεκπεραίωση των λειτουργιών.

3.2.2 Περιγραφή κλάσεων

Συνολικά στην υλοποίηση μας έχουμε έξι κλάσεις:

1. **MainActivity**: Όπου είναι υπεύθυνη για τις λειτουργίες της βασικής οθόνης της εφαρμογής. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην εμφάνιση της οθόνης, στην οποία ορίζονται οι λειτουργίες των κουμπιών, σύνδεσης, προβολής προφίλ και σάρωσης κωδικού QR. Επίσης ορίζουμε ότι πρέπει να πατηθεί δύο φορές το κουμπί εξόδους προς τα πίσω, ώστε να κλείσει η εφαρμογή.
2. **LoginActivity**: Όπου είναι υπεύθυνη για τις λειτουργίες της οθόνης σύνδεσης του χρήστη στο σύστημα. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην εμφάνιση της οθόνης. Σε αυτή την μέθοδο, ορίζουμε και ελέγχουμε την είσοδο των πεδίων εισόδου. Επίσης ορίζουμε τις λειτουργίες των κουμπιών σύνδεσης και μετάβασης προς την εγγραφή χρήστη. Πατώντας το κουμπί σύνδεσης, ελέγχονται τα πεδία και εμφανίζεται κατάλληλο μήνυμα λάθους, ή μεταφερόμαστε στην οθόνη προφίλ, εφόσον όλα κύλησαν ομαλά. Πατώντας το κουμπί επαναφοράς κωδικού, μεταφερόμαστε στην οθόνη επαναφοράς κωδικού.
3. **ResetPasswordActivity**: Όπου είναι υπεύθυνη για τις λειτουργίες της οθόνης επαναφοράς κωδικού του χρήστη. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην εμφάνιση της οθόνης. Εδώ ορίζεται η αποστολή μηνύματος προς την διεύθυνση που τοποθετείται στο πεδίο εισόδου. Στη συνέχεια εμφανίζεται κατάλληλο μήνυμα επιτυχίας/αποτυχίας στην οθόνη.
4. **RegisterActivity**: Όπου είναι υπεύθυνη για τις λειτουργίες της οθόνης εγγραφής του χρήστη στο σύστημα. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην εμφάνιση της οθόνης. Αρχικά ορίζονται το κουμπί εγγραφής και τα πεδία εισόδου στοιχείων. Δεν μπορούμε να προχωρήσουμε σε εγγραφή, αν είναι κενά. Επιπλέον ανιχνεύουν λάθη, και εμφανίζονται στην οθόνη κατάλληλα μηνύματα σφάλματος προς ενημέρωση του χρήστη. Για το ανέβασμα φωτογραφίας στο “Cloud Storage” δημιουργήσαμε δύο μεθόδους, την “*onActivityResult*”, που ελέγχει αν εμφανίστηκε ο επιλογέας φωτογραφίας και έχουν εξουσιοδοτηθεί τα δικαιώματα προς την εφαρμογή και την “*uploadImageToFirebase*” που ελέγχει το ανέβασμα της φωτογραφίας, και εμφανίζει μήνυμα λάθους, στην περίπτωση σφάλματος.
5. **MyProfileActivity**: Όπου είναι υπεύθυνη για τις λειτουργίες της οθόνης προφίλ του χρήστη. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην

εμφάνιση της οθόνης. Αρχικά παίρνουμε ένα στιγμιότυπο της βάσης και του *"Cloud Storage"*. Εφόσον υπάρξει πρόβλημα, μεταφερόμαστε την βασική οθόνη, αλλιώς δημιουργείται το γραφικό περιβάλλον της οθόνης με την μέθοδο *"createUI"*. Ορίζονται τα πεδία εμφάνισης των πληροφοριών του χρήστη, όπως η φωτογραφία του, το ονοματεπώνυμο του και ο αριθμός μητρώου του. Έχοντας πρόσβαση στο στιγμιότυπο της βάσης και του χώρου αποθήκευσης μπορούμε από το μοναδικό κλειδί του χρήστη να προσπελάσουμε την πληροφορία που ζητάμε και να την εμφανίσουμε στην εφαρμογή μας. Ο κωδικός QR δημιουργείται με την βοήθεια του *"ZXing"* με βάσει το κλειδί του χρήστη και εμφανίζεται στην οθόνη μας. Τέλος ορίζεται η λειτουργία του κουμπιού αποσύνδεσης του χρήστη από το σύστημα. Έχουμε και δύο μεθόδους, την *"onResume"* όπου ελέγχει όταν ξαναανοίξουμε την εφαρμογή και είμαστε αποσυνδεδεμένοι από το σύστημα να μας μεταφέρει στην βασική οθόνη, και την *"onBackPressed"*, όπου εφόσον πατήσουμε κουμπί προς τα πίσω, να μας μεταφέρει στην βασική οθόνη.

6. **ScanActivity:** Όπου είναι υπεύθυνη για τις λειτουργίες της οθόνης ενός κωδικού QR. Συγκεκριμένα υπάρχει μία μέθοδος *onCreate*, που καλείται στην εμφάνιση της οθόνης. Σε αυτή καλείται η συνάρτηση *"setUpPermissions"*, η οποία ελέγχει τα δικαιώματα που απαιτούνται για πρόσβαση στην κάμερα της κινητής συσκευής και καλεί τις βοηθητικές συναρτήσεις *"makeRequest"* και *"onRequestPermissionsResult"*, και η *"codeScanner"*. Σε αυτή την μέθοδο, έχουμε ένα *callback* το οποίο τρέχει συνεχόμενα, ώστε να έχουμε πάντα την τωρινή πληροφορία της κάμερας στην οθόνη μας. Έχουμε ορίσει η κάμερα που θα χρησιμοποιείται, να είναι η πίσω. Στην συνέχεια, όπως και στο *"MyProfileActivity"*, έτσι κι εδώ παίρνουμε ένα στιγμιότυπο της βάσης και του χώρου αποθήκευσης. Από την κάμερα αποκωδικοποιούμε την πληροφορία, και εφόσον αυτή είναι ένας υπάρχων κωδικός χρήστη στο σύστημα, τότε παίρνουμε το ονοματεπώνυμο, τον αριθμό μητρώου και την φωτογραφία του και τα εμφανίζουμε στην οθόνη μας. Επειδή η κάμερα τρέχει συνεχόμενα, μπορούμε να σαρώσουμε αμέσως έναν άλλο κωδικό για να εμφανιστούν τα νέα στοιχεία μπροστά μας. Η μέθοδος *"onPause"*, απελευθερώνει τους πόρους του συστήματος που κρατήσαμε, όταν βγούμε από αυτή την οθόνη. Η μέθοδος *"onResume"*, συνεχίζει την σάρωση στην περίπτωση που είχαμε κλειστή αυτή την οθόνη και την ξαναανοίξαμε.

3.3 Υλοποίηση iOS

3.3.1 Γενική περιγραφή

Για την υλοποίηση της εφαρμογής για κινητά τηλέφωνα με λειτουργικό σύστημα “iOS”, δουλέψαμε με το “XCode” με το framework “Cocoa Pods”. Χρησιμοποιήσαμε τη γλώσσα προγραμματισμού “Swift” και την γλώσσα “SwiftUI” για την υλοποίηση του περιβάλλοντος χρήστη. Η εφαρμογή σχεδιάστηκε με βάση το “iOS 15.0”, αλλά έχει δοκιμαστεί και σε παλαιότερες εκδόσεις. Για τον σχεδιασμό του UI, βασιστήκαμε στις *native* μεθόδους του λειτουργικού συστήματος. Για την υλοποίηση των απαιτήσεων όσον αφορά την δημιουργία και σάρωση του κωδικού QR, χρησιμοποιήσαμε την βιβλιοθήκη ανοιχτού κώδικα “CodeScanner”. Τις υπόλοιπες λειτουργίες τις δημιουργήσαμε είτε με την βοήθεια του “Google Firebase” είτε από την αρχή. Έχουμε δημιουργήσει τον κώδικα τμηματικά σε κλάσεις και “structs”. Κάθε “struct” είναι υπεύθυνη για το View της. Δηλαδή κάθε οθόνη που εμφανίζεται, είναι υλοποιημένη από ένα ξεχωριστό “struct”. Σε σχέση με την υλοποίηση για το “Android”, τα πράγματα είναι διαφορετικά ως προς την δομή των αρχείων, όπως θα δείξουμε και στην επόμενη υποενότητα.

3.3.2 Περιγραφή αρχείων

Συνολικά στην υλοποίηση έχουμε 6 βασικά *Views*, όμως η δομή τους είναι αρκετά διαφορετική από το “*Android*”.

1. *./Base/Modifier/CloseModifier.swift*: ορίζουμε την λειτουργία ενός κουμπιού που κλείνει μία «συρόμενη» οθόνη. Θα το χρειαστούμε στην οθόνη εγγραφής του χρήστη.
2. *./Base/View/ButtonView.swift*: ορίζουμε τις παραμέτρους ενός κουμπιού. Το *View* που ορίζεται μπορεί να κληθεί από οποιαδήποτε συνάρτηση με κατάλληλα ορίσματα. Με αυτό τον τρόπο εξασφαλίζουμε ότι κάθε κουμπί στην εφαρμογή μας θα ακολουθεί το ίδιο θέμα.
3. *./Base/View/InputPasswordView.swift*: ορίζουμε την εμφάνιση του πεδίου εισόδου κωδικού.
4. *./Base/View/InputTextView.swift*: ορίζουμε την εμφάνιση οποιουδήποτε πεδίου εισόδου χαρακτήρων.
5. *./Base/View/QRImageView.swift*: ορίζουμε την εμφάνιση και την λειτουργία για την δημιουργία του *QR* κωδικού του χρήστη.
6. *./ViewModel/AppViewModel.swift*: ορίζουμε την κλάση για το “*View Model*” της εφαρμογής μας. Σε αυτή την κλάση δημιουργούμε τις συναρτήσεις για την σύνδεση, αποσύνδεση και εγγραφή του χρήστη, και επαναφορά του κωδικού του. Όπως και στο “*Android*”, έτσι κι εδώ ακολουθούμε το ίδιο μοτίβο κλήσεων των συναρτήσεων του “*Firebase*”. Επίσης ορίζονται και μεταβλητές για έλεγχο αν ο χρήστης είναι συνδεδεμένος, προς ενημέρωση προς όλα τα “*Views*”. Αυτή η κλάση λειτουργεί ως “*Objervable Object*”, δηλαδή ως ένα αντικείμενο με το οποίο μπορούν να αλληλοεπιδράσουν όλα τα αρχεία που το δημιουργήσουν/καλέσουν.
7. *./FirebaseConnector.swift*: ορίζουμε την κλάση η οποία αρχικοποιεί τα στοιχεία του χρήστη στην βάση και στον χώρο αποθήκευσης. Δηλαδή δημιουργεί το έγγραφο στη συλλογή “*users*” στο “*Firestore*”, με το ονοματεπώνυμο και αριθμό μητρώου του χρήστη, και ανεβάζει την φωτογραφία στο “*Cloud Storage*”. Αυτή η κλάση λειτουργεί ως “*Objervable Object*”, δηλαδή ως ένα αντικείμενο με το οποίο μπορούν να αλληλοεπιδράσουν όλα τα αρχεία που το δημιουργήσουν/καλέσουν. Στην περίπτωση μας την χρειαζόμαστε κατά την προβολή του προφίλ του χρήστη, ώστε να εξαχθούν τα στοιχεία του σωστά από το σύστημα.

8. **./ImagePicker.swift:** ορίζουμε την προβολή της συλλογής φωτογραφιών, για την επιλογή εικόνας προφίλ του χρήστη.
9. **./MainView.swift:** ορίζουμε την προβολή της βασικής οθόνης της εφαρμογής μας, με τρία κουμπιά στην μέση για μετάβαση στην είσοδο του χρήστη, στο προφίλ του (εφόσον ήταν ήδη συνδεδεμένος, αλλιώς στην είσοδο), και στην σάρωση κωδικού *QR*.
10. **./LoginView.swift:** ορίζουμε την προβολή της οθόνης εισόδου του χρήστη. Εδώ όπως και στην προβολή εγγραφής του χρήστη, αξιοποιούμε τα πεδία εισόδου χαρακτήρων, και κουμπιά που δημιουργήσαμε νωρίτερα. Εφόσον τα πεδία δεν είναι κενά και υπάρχει σφάλμα κατά την είσοδο του χρήστη, θα εμφανιστεί κατάλληλο μήνυμα στην οθόνη. Επίσης υπάρχει η επιλογή για μετάβαση στην οθόνη επαναφοράς κωδικού.
11. **./ForgotPasswordView.swift:** ορίζουμε την προβολή για την επαναφορά κωδικού, στην οποία υπάρχει ένα πεδίο εισόδου για την διεύθυνση *email* του χρήστη και κουμπί για αποστολή μηνύματος επιβεβαίωσης.
12. **./RegisterView.swift:** ορίζουμε την προβολή εγγραφής του χρήστη στο σύστημα. Έχουμε επιλογή για φωτογραφία που γίνεται σύμφωνα με το *"ImagePicker"* που ορίσαμε νωρίτερα, και τα πεδία εισόδου διεύθυνσης *email*, κωδικού, ονοματεπώνυμου και αριθμού μητρώου. Αν τα πεδία είναι κενά, ο χρήστης δεν μπορεί να εγγραφτεί. Αν υπάρξει σφάλμα, εμφανίζεται κατάλληλο μήνυμα, ειδικά ο χρήστης μεταφέρεται στην οθόνη σύνδεσης.
13. **./ProfileView.swift:** ορίζουμε την προβολή του προφίλ του χρήστη. Γίνεται ανάκτηση των στοιχείων σύμφωνα με το αντικείμενου τύπου *"FirebaseConnector"*. Με την σωστή ανάγνωση εμφανίζεται η φωτογραφία του χρήστη, το ονοματεπώνυμο και ο αριθμός μητρώου του. Παράλληλα δημιουργείται και ο κωδικός *QR* από την κλάση *"QRImageView"* και την συνάρτηση *"generateQRCode"*. Ο χρήστης έχει την επιλογή να αποσυνδεθεί από το σύστημα, πατώντας το κουμπί *"Log out"*. Αναλόγως το αποτέλεσμα της ενέργειας, εμφανίζεται μήνυμα σφάλματος, ή μεταφερόμαστε στην βασική οθόνη της εφαρμογής.
14. **./ScannerView.swift:** ορίζουμε την προβολή σάρωσης *QR* κωδικού και εμφάνισης των στοιχείων του χρήστη. Κάνουμε χρήση της μεθόδου *"CodeScannerView"*, στην οποία διατρέχουμε το αποτέλεσμα της, το ανακτούμε και αναζητούμε εγγραφή στο σύστημα με αυτό. Το αποτέλεσμα όπως και στην περίπτωση του *"Android"* είναι το μοναδικό κλειδί χρήστη το οποίο δημιουργείται κατά την εγγραφή του. Παρομοίως υπάρχει η δυνατότητα συνεχών λήψεων φωτογραφιών προς

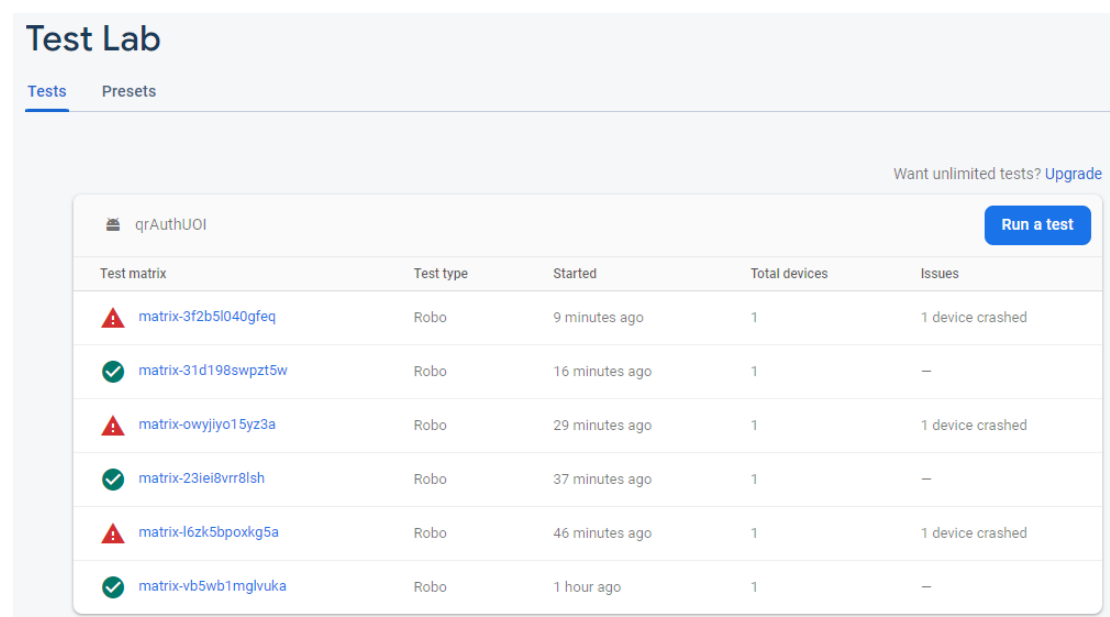
σάρωση, ώστε να διευκολύνεται η διαδικασία. Τα στοιχεία μεταβάλλονται καταλλήλως, όπως ανακτώνται από την βάση και σύμφωνα με τις συναρτήσεις *“getFullnameOrID”* και *“getImage”*.

3.4 Testing







3.4.1 Firebase Test Lab

Το “Firebase” μας δίνει την δυνατότητα για αυτοματοποιημένες ροές ελέγχων με το εργαλείο “Test Lab”. Μέσω του “Test Lab” μπορούμε να ορίσουμε την συσκευή στην οποία θέλουμε να γίνει η προσομοίωση, καθώς και ακριβώς τα βήματα που θέλουμε να ακολουθήσει η ροή. Έτσι έχουμε μπορούμε να βρούμε τυχόν σφάλματα στην εφαρμογή μας, που σε χειροκίνητο έλεγχο δε θα μπορούσαμε.

Στην φωτογραφία 31, βλέπουμε το ιστορικό των ελέγχων, ώστε να μπορούμε να βρούμε την τελευταία λειτουργική έκδοση της εφαρμογής μας.



The screenshot shows the Firebase Test Lab interface. At the top, there's a header with "Test Lab" and tabs for "Tests" and "Presets". Below the header, there's a table listing test runs. The table has columns for "Test matrix", "Test type", "Started", "Total devices", and "Issues". Each row represents a test run, with a status icon (red triangle for failure, green checkmark for success) next to the test matrix ID. A "Run a test" button is visible in the top right corner of the table area.

Test matrix	Test type	Started	Total devices	Issues
 matrix-3f2b5l040gfeq	Robo	9 minutes ago	1	1 device crashed
 matrix-31d198swpzt5w	Robo	16 minutes ago	1	—
 matrix-owyjiyo15yz3a	Robo	29 minutes ago	1	1 device crashed
 matrix-23iei8vrr8lsh	Robo	37 minutes ago	1	—
 matrix-l6zk5bpoxkg5a	Robo	46 minutes ago	1	1 device crashed
 matrix-vb5wb1mglvuka	Robo	1 hour ago	1	—

Εικόνα 31: Ιστορικό ελέγχων

Μπορούμε να κλικάρουμε ένα τεστ για να δούμε λεπτομέρειες.

The screenshot shows the Test Lab interface for a test named 'matrix-3f2b5l040gfeq'. At the top, it says 'Test Lab > qrAuthUI'. Below this, there's a summary section with a red warning icon and the text 'Robo test, 15 minutes ago'. To the right, a table shows the test results: Failed (1), Flaky (0), Passed (0), Skipped (0), and Inconclusive (0). Below the table is a link 'View screenshot clusters'. At the bottom, there's a section for device details: Device (Pixel 3, API Level 28), Locale (English (United States)), and Orientation (Portrait).

Failed	Flaky	Passed	Skipped	Inconclusive
1	0	0	0	0

View screenshot clusters →

Device	Locale	Orientation
Pixel 3, API Level 28	English (United States)	Portrait

Εικόνα 32: Αποτέλεσμα συγκεκριμένου ελέγχου

Ανάμεσα στα αποτελέσματα, υπάρχει μία λεπτομερής καρτέλα που μπορούμε να δούμε ακριβώς το σφάλμα, και τις ενέργειες με τις οποίες προέκυψε, μεταξύ άλλων και μέσω φωτογραφιών και βίντεο. Επίσης εκτυπώνεται το διάγραμμα ενεργειών για μεγαλύτερη βοήθεια ως προς τον προγραμματιστή.

The screenshot shows the 'Test Issues' section of the Test Lab interface. At the top, there's a red banner with a warning icon and the text 'App crashed during testing'. Below this, there's a summary section with a red warning icon, the text 'Failed', and details: 2/9/22, 5:54 PM, 41 sec, Portrait, English (United States). To the right is a 'Test results' button. Below the summary, there's a tabbed interface with 'Test Issues' selected. The 'Test Issues' section shows a 'Fatal exception' with the following details: 'kotlin.UninitializedPropertyAccessException: lateinit property globalURI has not been initialized'. Below this, it says 'FATAL EXCEPTION: main' and 'Process: com.example.appv2qrscanner, PID: 18962'. The stack trace shows the exception was thrown at 'com.example.appv2qrscanner.RegisterActivity.onCreate\$lambda-8\$lambda-7(RegisterActivity.kt:75)'.

Fatal exception

kotlin.UninitializedPropertyAccessException: lateinit property globalURI has not been initialized

FATAL EXCEPTION: main

Process: com.example.appv2qrscanner, PID: 18962

kotlin.UninitializedPropertyAccessException: lateinit property globalURI has not been initialized

at com.example.appv2qrscanner.RegisterActivity.onCreate\$lambda-8\$lambda-7(RegisterActivity.kt:75)

at com.example.appv2qrscanner.RegisterActivity.\$r8\$lambda\$-3zq-sYGeRGx6G50VnUsdLw0BBU(Unknown Source:0)

at com.example.appv2qrscanner.RegisterActivity\$\$ExternalSyntheticLambda3.onComplete(Unknown Source:6)

Εικόνα 33: Σφάλμα ελέγχου

The screenshot shows the 'Robo' tab of the 'Test Issues' section. At the top, there's a red banner with a warning icon and the text 'App crashed during testing'. Below this, there's a summary section with a red warning icon, the text 'Failed', and details: 2/9/22, 5:54 PM, 41 sec, Portrait, English (United States). To the right is a 'Test results' button. Below the summary, there's a tabbed interface with 'Robo' selected. The 'Robo' section shows 'Crawl duration' (36 s) and 'Crawl stats'. The 'Crawl stats' section shows 'Actions' (18), 'Activities' (3), and 'Screens' (13).

App crashed during testing

Failed 2/9/22, 5:54 PM 41 sec Portrait English (United States) Test results

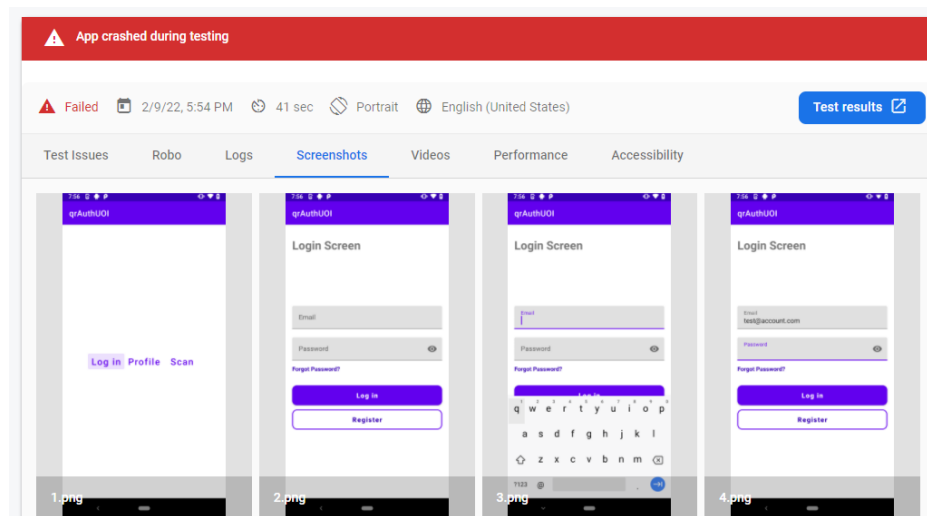
Test Issues **Robo** Logs Screenshots Videos Performance Accessibility

Crawl duration 36 s

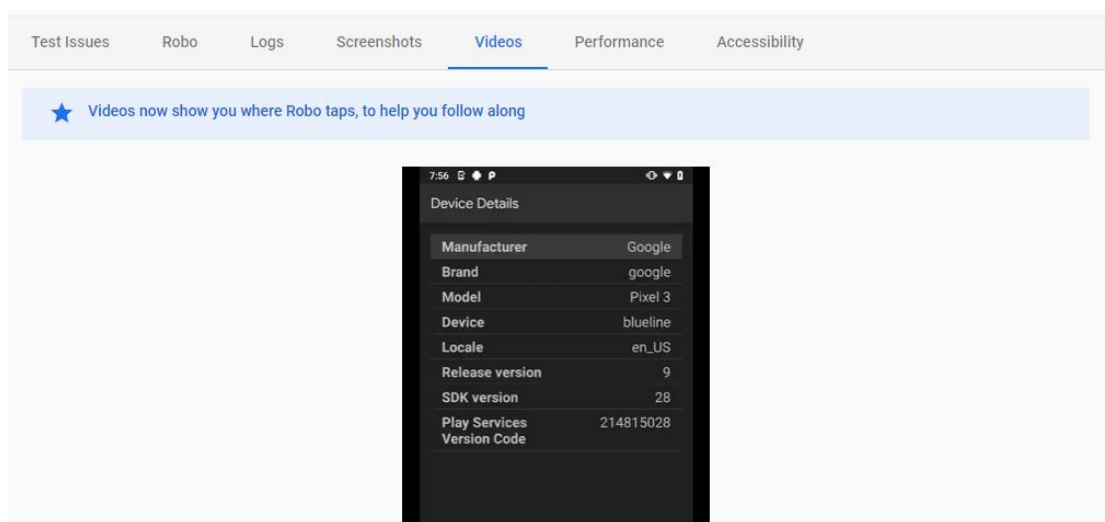
Crawl stats

Actions	Activities	Screens
18	3	13

Εικόνα 34: Λεπτομέρειες ελέγχου

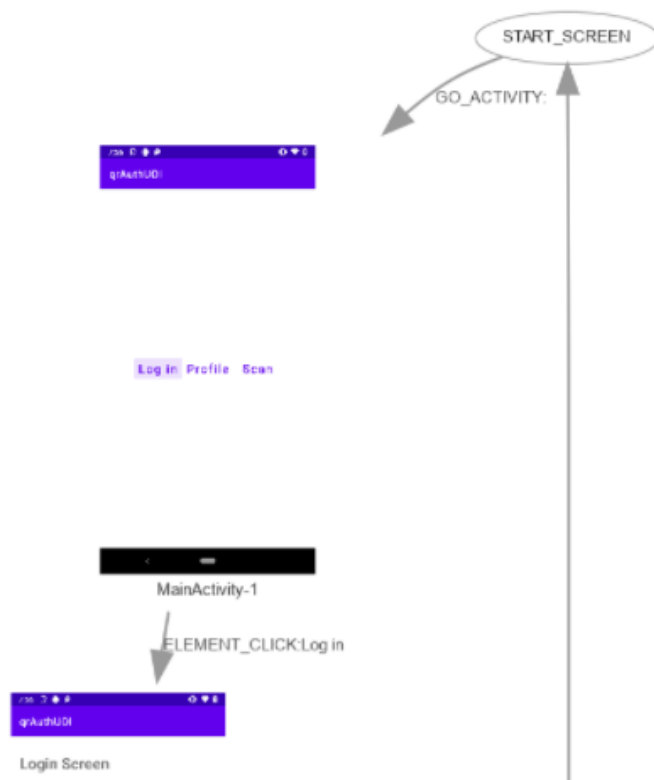


Εικόνα 35: Φωτογραφίες από τις ενέργειες του ελέγχου



Εικόνα 36: Βίντεο με όλες τις ενέργειες που εκτελέστηκαν

Crawl graph



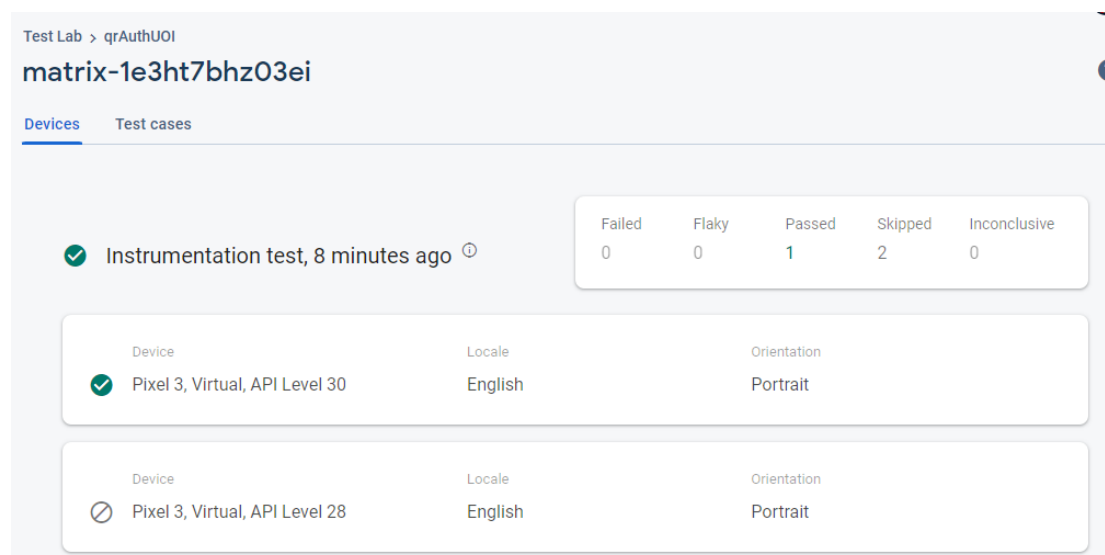
Εικόνα 37: Μέρος από την ροή ενεργειών του ελέγχου

3.4.2 Espresso

Το “Espresso” είναι ένα *API* που δημιουργήθηκε από την “Google” και ως σκοπό έχει την δημιουργία *scripts* για αυτοματοποιημένο έλεγχο των εφαρμογών. Στην δική μας εφαρμογή το χρησιμοποιούμε για αυτό το σκοπό. Έχουμε δημιουργήσει *scripts* για:

- Τον έλεγχο εγγραφής του χρήστη στο σύστημα
- Τον έλεγχο εισόδου του χρήστη στο σύστημα
- Την σωστή εμφάνιση των στοιχείων του χρήστη
- Την αποσύνδεση του χρήστη από το σύστημα

Το “Espresso” σε συνεργασία με το “Firebase Test Lab” μας δίνει την δυνατότητα να τρέχουμε τους ελέγχους μας από την πλατφόρμα του “Firebase”, οπότε υπάρχει γρηγορότερος έλεγχος μεταξύ των πλατφόρμων. Μέσω από το “Android Studio” μπορούμε να δημιουργήσουμε ένα *configuration* με τις εικονικές συσκευές που θέλουμε να τρέξουν τους ελέγχους μας, καθώς και την έκδοση “Android” που θα έχουν, ώστε να καλύψουμε όσο γίνεται μεγαλύτερο κοινό. Στην παρακάτω φωτογραφία μπορούμε να δούμε το αποτέλεσμα του ελέγχου που τρέξαμε, για όλες τις συσκευές ξεχωριστά. Επίσης όπως αναφέρθηκε στην προηγούμενη υποενότητα, μπορούμε να δούμε λεπτομερώς την εκτέλεση των ελέγχων μας.



Εικόνα 38: Αποτέλεσμα test script Espresso στο Test Lab

Κεφάλαιο 4. Επίλογος

4.1 Ενημέρωση χρηστών

Η ενημέρωση ενός χρήστη δυνατή, καθώς μπορούμε να μεταβάλλουμε τα στοιχεία του μέσω της βάσης. Μέσω του *“Cloud Firestore”* μπορούμε να αλλάξουμε το ονοματεπώνυμο και τον αριθμό μητρώου ενός χρήστη, ενώ μέσω του *“Cloud Storage”* μπορούμε να αλλάξουμε την φωτογραφία προφίλ του. Η αναζήτηση του χρήστη και στις δύο περιπτώσεις χρειάζεται να γίνει μέσω του μοναδικού κλειδιού του, το οποίο μπορούμε να βρούμε από το *“Firebase Authentication”*, εφόσον γνωρίζουμε την διεύθυνση *email* του.

Για την διαγραφή ενός χρήστη απαιτείται η διαγραφή του και από τις 3 υπηρεσίες του *“Firebase”* που χρησιμοποιούμε, ώστε να απελευθερωθούν οι πόροι συστήματος, καθώς και το κλειδί χρήστη.

Παρομοίως αν θέλουμε να προσθέσουμε έναν νέο χρήστη, απαιτείται η προσθήκη μέσω των υπηρεσιών του *“Firebase”*, για να φροντίσουμε ότι όλα τα στοιχεία του είναι αλληλοσυνδεδεμένα.

4.2 Ενημέρωση βάσης δεδομένων

Στην βάση δεδομένων *“Cloud Firestore”* μπορούμε να προσθέσουμε κατά βούληση νέες συλλογές ή έγγραφα που μπορεί να χρειαστούν στο μέλλον. Επίσης μας δίνεται η δυνατότητα να προσθέσουμε πεδία στα ήδη υπάρχοντα έγγραφα, δίχως να επηρεάζονται τα στοιχεία της βάσης. Οι νέες λειτουργικότητες χρειάζεται να υλοποιηθούν στα αρχεία κώδικα μας, ώστε να δούμε τις αλλαγές.

4.3 Σύνοψη και συμπεράσματα

Συνοπτικά, δημιουργήσαμε την εφαρμογή με μοναδικό σκοπό την ανέπαφη ταυτοποίηση των φοιτητών. Αυτό κρίθηκε δυνατόν από την δημιουργία ενός χρήστη, την εμφάνιση και σάρωση των στοιχείων του, όλα εκ των οποίων λαμβάνουν χώρο εξ ολοκλήρου στην εφαρμογή μας.

Οι δυσκολίες που αντιμετωπίσαμε ήταν η ανάγκη ταυτόχρονης υλοποίησης της εφαρμογής για τα δύο μεγαλύτερα λειτουργικά συστήματα κινητών συσκευών και η υλοποίηση μίας ασφαλούς βάσης δεδομένων. Θελήσαμε να δημιουργήσουμε μία ίδια εφαρμογή (front-end και back-end) και για τα δύο λειτουργικά συστήματα. Αυτό κρίθηκε αδύνατον καθώς τα εργαλεία που προσφέρονται για ένα ενωμένο front-end, δεν μας καλύπτουν πλήρως σε άλλους τομείς. Για παράδειγμα, μέσω του *Flutter*, θα μπορούσαμε με ένα *codebase* να δημιουργήσουμε εφαρμογή και για το *Android* και για το *iOS*. Όμως οι ιδιαιτερότητες τους ως λειτουργικά θα μπορούσαν να εμφανίσουν προβλήματα που δε θα μπορούσαν να λυθούν ανεξαρτήτως πλατφόρμας. Έτσι αναγκαστήκαμε να δημιουργήσουμε δύο εφαρμογές που το *front-end* κομμάτι τους μοιάζει αρκετά και το *back-end* θα χρησιμοποιεί την ίδια βάση δεδομένων. Αυτό το καταφέραμε με την βοήθεια του *Firebase* το οποίο είναι προσβάσιμο από όλες τις κινητές συσκευές ανεξαρτήτου λειτουργικού συστήματος.

Η υλοποίηση της εφαρμογής για λειτουργικό σύστημα *Android* ήταν σχετικά εύκολη, καθώς το *Android Studio* λειτουργεί σε όλα τα λειτουργικά συστήματα υπολογιστών, η *Kotlin* μοιάζει αρκετά με την *Java* που μαθαίνουμε στη σχολή και η δημιουργία του *front-end* είναι μία διαδικασία στην οποία χρησιμοποιούμε *HTML*, όπου και πάλι μαθαίνουμε στη σχολή.

Η υλοποίηση της εφαρμογής για λειτουργικό σύστημα *iOS*, από την άλλη ήταν μία αρκετά επίπονη διαδικασία. Τα εργαλεία που απαιτούνται για την δημιουργία εφαρμογής *iOS*, είναι διαθέσιμα αποκλειστικά σε συσκευές με λειτουργικό σύστημα *macOS*. Επειδή δεν είχα πρόσβαση σε τέτοια συσκευή, αρχικά προσπάθησα να χρησιμοποιήσω μία εικονική μηχανή, όμως η χρήση οποιασδήποτε εικονικής μηχανής για λειτουργικό σύστημα *macOS*, είναι ελάχιστη έως και καθόλου. Αυτό συμβαίνει, διότι δεν υπάρχουν *Video Drivers* για ομαλή χρήση της εικονικής μηχανής. Έτσι προσπάθησα να φτιάξω ένα σύστημα *Hackintosh*, δηλαδή μέσω συμβατικού υπολογιστή με λειτουργικό σύστημα

Windows, με κατάλληλες τροποποιήσεις να καταφέρουμε να έχουμε ένα *macOS* μηχανήμα. Αυτό πήρε αρκετό χρόνο για να γίνει, όμως τελικά το καταφέραμε. Από εκεί και πέρα η δημιουργία της εφαρμογής απαιτήσε γνώσεις του *editor XCode* καθώς και της προγραμματιστικής γλώσσας *Swift*.

Στο μέλλον η εφαρμογή θα μπορούσε να ενοποιηθεί μέσω της πλατφόρμας *Kotlin Multiplatform*, ώστε οι δηλώσεις των κλάσεων/μεθόδων να είναι κοινές. Εν τέλει δημιουργήσαμε μία εφαρμογή που ελπίζουμε να ανταπεξέλθει για τον σκοπό τον οποίο δημιουργήθηκε, δηλαδή την ανέπαφη ταυτοποίηση φοιτητών, και όχι μόνο.

Βιβλιογραφία

- Flowcharts Documentation:

<https://creately.com/diagram-type/flowchart/>

- Firebase Documentation:

<https://firebase.google.com/docs?authuser=0>

- Espresso Documentation:

<https://developer.android.com/training/testing/espresso>

- ZXing Documentation:

<https://github.com/zxing/zxing>,

<https://code.luasoftware.com/tutorials/android/android-generate-qr-code-with-zxing/>

- Code Scanner (Android) Documentation:

<https://github.com/yuriy-budiyev/code-scanner>

- Code Scanner (iOS) Documentation:

<https://github.com/twostraws/CodeScanner>