ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΣΥΣΤΗΜΑΤΟΣ ΠΑΡΟΥΣΙΟΛΟΓΙΟΥ/ΠΡΟΣΒΑΣΗΣ ΜΕ ΧΡΗΣΗ NFC

Μάριος Κονιδάρης

Α.Μ. 711161074

cs161074@uniwa.gr

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ & ΥΠΟΛΟΓΙΣΤΩΝ

ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

Υπεύθυνος καθηγητής: Δρ. Ιωάννης Βογιατζής, Καθηγητής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και ανάπτυξη συστήματος παρουσιολογίου/πρόσβασης με χρήση NFC

Μάριος Κονιδάρης
Α.Μ. 711161074

Εισηγητής:

Ιωάννης Βογιατζής, Καθηγητής

Εξεταστική Επιτροπή:

Ιωάννης Βογιατζής, Καθηγητής
Χρήστος Τρούσσας, Επίκουρος Καθηγητής
Ακριβή Κρούσκα, ΕΔΙΠ

IOANNIS VOYIATZIS Digitally signed by IOANNIS VOYIATZIS Date: 2024.11.11 13:07:03 +02'00'

Christos Troussas Digitally signed by Christos Troussas Date: 2024.11.04 15:06:00 +02'00'

Akrivi Krouska Digitally signed by Akrivi Krouska Date: 2024.11.04 16:40:12 +02'00'

Ημερομηνία εξέτασης:

16/07/24

ΔΗΛΩΣΗ ΣΥΓΓΡΑΦΕΑ ΔΙΠΛΩΜΑΤΙΚΗΣ ΕΡΓΑΣΙΑΣ

«Βεβαιώνω ότι είμαι συγγραφέας αυτής της Διπλωματικής εργασίας και κάθε βοήθεια την οποία είχα για την προετοιμασία της, είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, οι όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών ή λέξεων, είτε ακριβώς είτε παραφρασμένες, αναφέρονται στο σύνολό τους, με πλήρη αναφορά στους συγγραφείς, τον εκδοτικό οίκο ή το περιοδικό, συμπεριλαμβανομένων και των πηγών που ενδεχομένως χρησιμοποιήθηκαν από το διαδίκτυο. Επίσης, βεβαιώνω ότι αυτή η εργασία έχει συγγραφεί από μένα αποκλειστικά και αποτελεί προϊόν πνευματικής ιδιοκτησίας τόσο δικής μου, όσο και του Ιδρύματος.
Παράβαση της ανωτέρω ακαδημαϊκής μου ευθύνης αποτελεί ουσιώδη λόγο για την ανάκληση του πτυχίου μου».

Ο/Η Δηλών/ούσα
Μάριος Κονιδάρης

ΕΥΧΑΡΙΣΤΙΕΣ

Το αντικείμενο ήταν πολύ ενδιαφέρον και με έκανε να ασχοληθώ με τεχνολογίες που μου άνοιξαν νέους ορίζοντες και με βοήθησαν στην επαγγελματική μου καριέρα. Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου που με βοήθησε, καθώς και τους κοντινούς μου ανθρώπους που με στήριξαν.

Περιεχόμενα

Εισαγωγή

Στις σύγχρονες κοινωνίες και επιχειρήσεις, η αποτελεσματική διαχείριση των διαδικασιών είναι καθοριστική για την αύξηση της παραγωγικότητας και τη βελτίωση της εμπειρίας των χρηστών. Παρά την πρόοδο στις τεχνολογίες πληροφορικής, πολλές διαδικασίες εξακολουθούν να βασίζονται σε παραδοσιακές μεθόδους που δεν εκμεταλλεύονται τις δυνατότητες των σύγχρονων τεχνολογιών, όπως το NFC (Near Field Communication) και οι κωδικοί QR (Quick Response). Αυτό έχει ως αποτέλεσμα οι διαδικασίες να έχουν χαμηλή αποτελεσματικότητα, περιορισμένη ασφάλεια, περιορισμένη διαλειτουργικότητα και αδυναμία παρακολούθησης και ελέγχου. Η υιοθέτηση τεχνολογιών όπως το NFC και οι κωδικοί QR στις καθημερινές διαδικασίες όχι μόνο βελτιώνει την αποτελεσματικότητα και την ασφάλεια αλλά και ενισχύει την συνολική εμπειρία των χρηστών. Η ανάπτυξη και η εφαρμογή μίας τέτοιας λύσης εξετάζεται και υλοποιείται στην παρούσα διπλωματική εργασία που σκοπεύει να λύσει το πρόβλημα της λήψης παρουσιών στα εργαστηριακά μαθήματα, καθώς και στις εξεταστικές περιόδους. Η εφαρμογή στοχεύει στην αυτοματοποίηση της διαδικασίας, στην αύξηση της ασφάλειας και διαφάνειας αλλά και στην διαλειτουργικότητα. Παράλληλα, παρουσιάζεται μια μελέτη για τις δύο τεχνολογίες (NFC, QR), περιγράφονται τα πλεονεκτήματα και τα μειονεκτήματα των δύο και εξηγείται γιατί εν τέλει προτιμάται η μια έναντι της άλλης. Στο τέλος, γίνεται παρουσίαση του συστήματος που αναπτύχθηκε, παρατίθενται επίσης, οι τεχνολογίες που χρησιμοποιήθηκαν (Flutter, Firebase), δίνεται το σχήμα της βάσης δεδομένων και μια ξενάγηση στις διάφορες οθόνες της εφαρμογής.

1. Περιγραφή προβλήματος

Οι καθηγητές πανεπιστημίων αντιμετωπίζουν πολυάριθμα προβλήματα κατά τη διαδικασία λήψης παρουσιών χρησιμοποιώντας τις παραδοσιακές μεθόδους. Τα βασικά βήματα της διαδικασίας περιλαμβάνουν τη διανομή φύλλων παρουσίας ή την ανάγνωση ονομάτων από λίστες, κατά την οποία οι φοιτητές δηλώνουν την παρουσία τους. Αυτή η προσέγγιση είναι επιρρεπής σε πολλά προβλήματα και αδυναμίες. Πρώτον, η διαδικασία μπορεί να είναι **χρονοβόρα** σε μαθήματα με μεγάλο αριθμό φοιτητών, αυτό σημαίνει ότι μειώνεται ο διαθέσιμος χρόνος για μάθημα. Δεύτερον, η διαδικασία με την χρήση χειρόγραφων λιστών ή υπολογιστικών φύλλων, είναι επιρρεπής σε **ανθρώπινα λάθη** αφού οι καθηγητές ενδέχεται να παραλείψουν ονόματα ή να καταγράψουν λανθασμένα στοιχεία, κάτι το οποίο οδηγεί σε ανακρίβειες. Επιπλέον, υπάρχει **κίνδυνος απώλειας δεδομένων**, τα χαρτιά ή τα ψηφιακά αρχεία μπορούν εύκολα να χαθούν ή να καταστραφούν, θέτοντας σε κίνδυνο την αξιοπιστία της διαδικασίας λήψης παρουσιών. Η **πλαστοπροσωπία** είναι πάντα ένας κίνδυνος που καραδοκεί αφού ένας φοιτητής μπορεί να υποδυθεί κάποιον άλλο και αυτό υπονομεύει την διαδικασία τελικά. Τέλος, η χρήση χαρτιών και υπολογιστικών φύλλων οδηγεί σε **δυσκολία διαχείρισης δεδομένων** αφού τα στοιχεία δεν είναι οργανωμένα σε ένα σημείο, αυτό κάνει την διαδικασία ανάλυσης ή διαχείριση αυτών πιο δύσκολη.

Η χρήση τεχνολογιών όπως το NFC (Near Field Communication) και οι κωδικοί QR (Quick Response) μπορεί να βελτιώσει σημαντικά τη διαδικασία λήψης παρουσιών, προσφέροντας λύσεις στα παραπάνω προβλήματα. Η διαδικασία λήψης παρουσιών μπορεί να αυτοματοποιηθεί πλήρως με τη χρήση εφαρμογών κινητού τηλεφώνου. Αρχικά, οι φοιτητές μπορούν να σαρώσουν έναν κωδικό QR ή να χρησιμοποιήσουν τη λειτουργία NFC στο κινητό τους για να δηλώσουν την παρουσία τους άμεσα. Αυτό μειώνει δραστικά τον χρόνο που απαιτείται για την καταγραφή παρουσιών. Έπειτα, η χρήση ψηφιακών τεχνολογιών εξαλείφει τα ανθρώπινα λάθη και αυξάνει την ακρίβεια της καταγραφής. Τα δεδομένα καταγράφονται και αποθηκεύονται αυτόματα, μειώνοντας τον κίνδυνο λαθών. Επιπλέον, οι ψηφιακές καταγραφές παρουσίας αποθηκεύονται με ασφάλεια σε κεντρικές βάσεις δεδομένων, εξαλείφοντας τον κίνδυνο απώλειας ή καταστροφής των δεδομένων. Η χρήση προσωπικών συσκευών και μοναδικών κωδικών QR ή NFC ετικέτων καθιστά την πλαστοπροσωπία εξαιρετικά δύσκολη, διασφαλίζοντας ότι μόνο οι παρόντες φοιτητές δηλώνουν την παρουσία

τους. Η χρήση προσωπικών συσκευών και μοναδικών κωδικών QR ή NFC ετικέτων καθιστά την πλαστοπροσωπία εξαιρετικά δύσκολη, διασφαλίζοντας ότι μόνο οι παρόντες φοιτητές δηλώνουν την παρουσία τους. Τέλος, Οι καθηγητές μπορούν να έχουν άμεση πρόσβαση σε αναφορές και στατιστικά στοιχεία παρουσιών, διευκολύνοντας τη διαχείριση και την ανάλυση των δεδομένων. Αυτό βελτιώνει την παρακολούθηση της συμμετοχής και διευκολύνει την εξαγωγή συμπερασμάτων.

Η υιοθέτηση τεχνολογιών NFC και QR code για τη λήψη παρουσιών προσφέρει μια σύγχρονη, αποτελεσματική και αξιόπιστη λύση στα προβλήματα που αντιμετωπίζουν οι καθηγητές με την παραδοσιακή διαδικασία. Με τη χρήση αυτών των τεχνολογιών, η εκπαιδευτική διαδικασία εκσυγχρονίζεται, βελτιώνοντας τη συνολική εμπειρία τόσο των καθηγητών όσο και των φοιτητών. Στην συνέχεια, παρουσιάζουμε τις πιθανές λύσεις για την αυτοματοποίηση της διαδικασίας.

2. Πιθανές Λύσεις

2.1 Μια θεωρητική λύση

Μια θεωρητική λύση στο πρόβλημα, είναι η ανάπτυξη μιας εφαρμογής κινητού τηλεφώνου, η οποία θα χρησιμοποιείται από τους φοιτητές και τους καθηγητές. Η εφαρμογή θα έχει μια βάση δεδομένων στην οποία θα αποθηκεύει τα δεδομένα για τους χρήστες (φοιτητές και καθηγητές), καθώς και τα μαθήματα, τα εργαστήρια (αίθουσες) και τις παρουσίες. Ο καθηγητής, με την χρήση μιας τεχνολογίας ανέπαφης συναλλαγής, θα μπορεί να ταυτοποιεί τον φοιτητή, αφού ο φοιτητής επιδείξει το δικό του κινητό τηλέφωνο. Η ταυτοποίηση (επιτυχής ή ανεπιτυχής), θα αποφασίζεται από τα δεδομένα που υπάρχουν στην βάση και το αποτέλεσμα της ταυτοποίησης θα αποθηκεύεται στην συνέχεια, εκεί.

2.2 Πιθανή λύση με NFC

Στην λύση αυτή, οι δύο συσκευές πρέπει να διαθέτουν την τεχνολογία NFC. Αυτό αυτόματα κάνει αυτή την υλοποίηση λιγότερο προσιτή στους χρήστες, καθώς δεν διαθέτουν όλοι κινητά με NFC. Επίσης, η υλοποίηση είναι δυσκολότερη αφού για να επιτευχθεί, θα πρέπει να χρησιμοποιηθεί μια τεχνική που ονομάζεται Host Card Emulation (HCE). Αυτή είναι μια τεχνική, η οποία χρησιμοποιείται κυρίως για τραπεζικές συναλλαγές (Google Pay, Apple Pay), όταν επιλέγουμε να πληρώσουμε ανέπαφα σε ένα τερματικό ή σε ένα «κινητό» του/της καταστηματάρχη. Ουσιαστικά, η μία συσκευή μιμείται μια ανέπαφη κάρτα. Έτσι η άλλη μπορεί να στείλει δεδομένα. Θεωρητικά είναι αυτό που χρειαζόμαστε, αφού μια συσκευή στέλνει στην άλλη δεδομένα έτσι και στην εφαρμογή θα πρέπει η συσκευή του φοιτητή να στείλει το ακαδημαϊκό αναγνωριστικό του στην συσκευή του καθηγητή.

Για να γίνει αυτό, θα πρέπει να επικοινωνήσουμε με το κομμάτι του NFC που επεξεργάζεται τα δεδομένα που στέλνονται ή λαμβάνονται από την άλλη συσκευή. Θα πρέπει να γράψουμε κώδικα ξεχωριστό για κάθε λειτουργικό σύστημα ώστε να πάρουμε αυτή την πληροφορία και στο τέλος να προωθήσουμε τα αποτελέσματα στο Flutter το οποίο χρησιμοποιεί τα Platform channels για να διαβάσει τις πληροφορίες

που στέλνονται από τον εγγενή κώδικα. Παρακάτω, δίνεται μια πιο αναλυτική περιγραφή της πιθανής υλοποίησης για Android.

Σαν πρώτο βήμα θα πρέπει να δημιουργήσουμε μια υπηρεσία για τις συναλλαγές NFC. Για να το κάνουμε αυτό στο Android, πρέπει να μεταβούμε αρχικά στην περιοχή που βρίσκεται το AndroidManifest.xml αρχείο. Σε αυτό το αρχείο δηλώνουμε υπηρεσίες, άδειες για τις υπηρεσίες που πιθανόν θα χρησιμοποιήσει η εφαρμογή μας κ.α. Έτσι, δημιουργούμε ένα αρχείο που το ονομάζουμε nfcservice.xml. Σε αυτό το αρχείο ορίζουμε τις διάφορες επιλογές για την υπηρεσία μας. Αφού δημιουργηθεί, μεταβαίνουμε στο AndroidManifest.xml και το δηλώνουμε σαν μια νέα υπηρεσία. Τώρα σειρά έχει να γράψουμε τον εγγενή κώδικα που θα χρησιμοποιηθεί σαν μεσάζοντας από την συσκευή στο Flutter.

Αυτή η τεχνική έχει αρκετά αρνητικά. Αρχικά, καθώς η Apple δεν υποστηρίζει το HCE για εφαρμογές εκτός του Apple Pay, για λόγους ασφάλειας σύμφωνα με (PANDEY 2021), αυτό αυτόματα αποκλείει τους χρήστες που έχουν κινητά iPhone. Έπειτα, όπως αναφέραμε, θα πρέπει να γραφτεί κώδικας συγκεκριμένος για το κάθε λειτουργικό σύστημα [Android (Kotlin ή Java), iOS (Swift ή Objective-C)]. Αυτό πάει αντίθετα στην αρχική μας ιδέα, καθώς το Flutter θέλει να μας απαλλάξει από αυτή την δουλειά. Ένας κώδικας για πολλές συσκευές. Επιπλέον, ο κώδικας μας θα είναι πιο δύσκολος στο να διατηρηθεί και είναι εύκολο να γίνουν λάθη. Κάτι τέτοιο είναι απαγορευτικό όταν έχουμε να κάνουμε με την διαδικασία των παρουσιών στο πανεπιστήμιο.

Ας δούμε κάποιες πληροφορίες για την τεχνολογία NFC.

2.2.1 NFC - Με λίγα λόγια

Το NFC, το οποίο είναι ένα υποσύνολο του RFID, είναι μια αμφίδρομη τεχνολογία επικοινωνίας κοντινής απόστασης μεταξύ δύο συσκευών μέσα από ένα ηλεκτρομαγνητικό πεδίο που δημιουργείται μεταξύ τους. Η απόσταση λειτουργίας είναι περίπου 5-10 εκ. και μπορούν να μεταφερθούν 424 kb/s δεδομένων το πολύ. (Steffen, και συν. 2010). Οι συσκευές μπορεί να είναι μια κάρτα (τραπέζης ή άλλη), ένα κινητό τηλέφωνο ή μια συσκευή RFID.

2.2.1.1 Τί είναι το RFID

Το RFID, παρόμοια με το NFC, είναι μια ανέπαφη τεχνολογία που χρησιμοποιεί ηλεκτρομαγνητικά πεδία για να μεταφέρει δεδομένα. Χρησιμοποιείται κυρίως για τον

σκοπό της ταυτοποίησης. (C. -H., K. -W. and K. -W. 2018). Αυτές οι συσκευές που χρησιμοποιούν το RFID, αποτελούνται από τρία μέρη. Την κεραία, τον πομποδέκτη και τον αναμεταδότη. Ο τρόπος που λειτουργεί είναι ο εξής. Η κεραία μεταδίδει ένα σήμα σε μορφή ραδιοσυχνότητας το οποίο ενεργοποιεί τον αναμεταδότη. Αυτός περιέχει έναν ελεγκτή, ο οποίος λαμβάνει τα δεδομένα και εκτελεί την εκάστοτε εντολή. Αυτή, μπορεί να είναι από την πιο απλή, όπως το άνοιγμα μιας πόρτας, μέχρι την πιο περίπλοκη, διασύνδεση με μία βάση δεδομένων για μια συναλλαγή. (N. S. S., και συν. 2016).

2.2.2 Τρόποι επικοινωνίας NFC

Οι τρόποι επικοινωνίας του NFC είναι τρείς. (al ofeishat 2012)

- Read / Write mode
- Tag emulation mode
- Peer-to-peer mode

Στο **πρώτο**, το κινητό, στέλνει ή διαβάζει δεδομένα από / σε ένα tag. Αυτός ο τρόπος είναι ιδιαίτερα χρήσιμος όταν έχουμε προγραμματίσει το tag μας να κάνει μια δουλειά (άνοιγμα φώτων, κλείδωμα πόρτας κ.α.) και χρησιμοποιούμε το κινητό για ξεκινήσουμε αυτή την δουλειά. Στο **δεύτερο**, το κινητό μιμείται το tag, όπως μαρτυρά το όνομα. Ίσως η πιο σημαντική και διαδεδομένη χρήση του NFC, καθώς μπορούμε να πληρώνουμε ανέπαφα με το κινητό ή την χρεωστική / πιστωτική μας κάρτα όταν ψωνίζουμε, ή να εισέλθουμε σε ένα χώρο με την χρήση κάρτας, ή αντίστοιχα, στην δική μας περίπτωση, που θέλουμε να ελέγχουμε τους φοιτητές και να τους καταγράφουμε παρουσίες. Τέλος, στο **τρίτο**, τα κινητά μπορούν να συνδεθούν και να μεταφέρουν δεδομένα μεταξύ τους. Για παράδειγμα, στο παρελθόν τα κινητά μπορούσαν να μεταφέρουν φωτογραφίες μεταξύ τους.

2.2.3 Τρόποι λειτουργίας NFC

Το NFC δουλεύει με δύο τρόπους. Ενεργητικά και παθητικά. (al ofeishat 2012). Ας υποθέσουμε ότι έχουμε δύο συσκευές με NFC που θέλουν να επικοινωνήσουν. Εάν χρησιμοποιούν τον **ενεργητικό** τρόπο, τότε και οι δύο συσκευές θα πρέπει να παραγάγουν το ηλεκτρομαγνητικό πεδίο που απαιτείται για να υπάρξει σύζευξη. Η συσκευή που ξεκινάει την συναλλαγή θεωρείται ο μυητής ενώ η δεύτερη συσκευή θεωρείται ο στόχος. Ο μυητής λοιπόν, αποφασίζει τον τρόπο (ενεργητικός ή παθητικός) και την ταχύτητα μεταφοράς των δεδομένων. Στην συνέχεια, ο στόχος

μαθαίνει την ταχύτητα και τον τρόπο, οπότε πραγματοποιείται η συναλλαγή. Αυτή θα τερματιστεί, εάν πραγματοποιηθεί η συναλλαγή ή μια από τις δύο συσκευές βγει από το ηλεκτρομαγνητικό πεδίο. Ο ενεργητικός τρόπος βέβαια απαιτεί κατανάλωση μπαταρίας, οπότε, δεν είναι βέλτιστο για μια κινητή συσκευή και άρα σε εκείνη την περίπτωση γίνεται χρήση του παθητικού τρόπου. Στον **παθητικό** τρόπο, οι συσκευές που δεν είναι κινητές συσκευές, δημιουργούν το ηλεκτρομαγνητικό πεδίο. Έτσι επιτυγχάνεται εξοικονόμηση μπαταρίας. Στην συνέχεια πραγματοποιείται η συναλλαγή. Για τον τερματισμό, ισχύει ό,τι ισχύει και για τον ενεργητικό τρόπο.

2.2.4 Αρχιτεκτονική συσκευής NFC

Ένα κινητό τηλέφωνο με NFC, αποτελείται από δύο ολοκληρωμένα κυκλώματα. Το SE (Secure Element) και την διεπαφή NFC. Η διεπαφή NFC, αποτελείται και αυτή από την κεραία του NFC και ένα ακόμα ολοκληρωμένο κύκλωμα, τον ελεγκτή NFC (N. S. S., et al. 2016). Στον παρακάτω πίνακα δίνονται οι λειτουργίες του κάθε ολοκληρωμένου κυκλώματος.

Ολοκληρωμένο Κύκλωμα	**Λειτουργία**
Ελεγκτής κεντρικού υπολογιστή	Είναι υπεύθυνος για τον ορισμό του τρόπου που θα πραγματοποιηθεί η συναλλαγή στον ελεγκτή NFC, την επεξεργασία των δεδομένων που λαμβάνονται ή στέλνονται μέσω της διεπαφής του ελεγκτή του κεντρικού υπολογιστή και τέλος να δημιουργήσει μια σύνδεση ανάμεσα στον ελεγκτή NFC και το SE.
Ελεγκτής NFC	Είναι υπεύθυνος για την μεταφορά των δεδομένων από τον αναγνώστη στο SE.
SE	Είναι υπεύθυνο για την ασφαλή αποθήκευση των δεδομένων όταν αυτά είναι ευαίσθητα αλλά και την παροχή

	ενός περιβάλλοντος στο οποίο να μπορεί να γίνει η συναλλαγή. Τέλος, πρέπει να παρέχει προστασία ενάντια σε επιθέσεις.

Πίνακας 1. Λειτουργίες ολοκληρωμένων κυκλωμάτων NFC.

Οπότε, μια συναλλαγή μέσω δύο συσκευών NFC γίνεται ως εξής. Αρχικά θα πρέπει η συσκευή η οποία δεν είναι κινητό τηλέφωνο να πάρει τον ρόλο του μυητή και να επιλέξει τον τρόπο (παθητικός), την ταχύτητα (αυτές οι ρυθμίσεις πραγματοποιούνται από τον ελεγκτή NFC) και να δημιουργήσει το ηλεκτρομαγνητικό πεδίο. Επιπλέον, θα πρέπει να δημιουργήσει την σύνδεση μεταξύ του ελεγκτή του NFC και του SE. Για αυτά είναι υπεύθυνος ο ελεγκτής κεντρικού υπολογιστή. Αφού ο μυητής έχει κάνει όλες τις απαραίτητες ρυθμίσεις που αναφέραμε, θα πρέπει να λάβει ή να διαβάσει τα δεδομένα που λαμβάνει από τον ελεγκτή κεντρικού υπολογιστή κατά την συναλλαγή. Η κινητή συσκευή θα πληροφορηθεί για την ταχύτητα και τον τρόπο και έτσι θα αρχίσει η συναλλαγή. Κατά την συναλλαγή, ο ελεγκτής NFC μεταφέρει τα δεδομένα στο SE όπου και θα αποθηκευτούν και θα επεξεργαστούν, καθώς το περιβάλλον είναι ασφαλές.

2.3 Πιθανή υλοποίηση με QR code

Αυτή η υλοποίηση είναι αρκετά ευκολότερη και πιο προσιτή σε σχέση με την υλοποίηση NFC. Το μόνο που χρειάζεται είναι η συσκευή να διαθέτει κάμερα. Ο τρόπος είναι ο εξής. Κάθε μαθητής έχει ένα QR code το οποίο παράγεται με βάση το ακαδημαϊκό αναγνωριστικό του. Ο καθηγητής όταν χρειαστεί με το πάτημα ενός κουμπιού, ανοίγει την κάμερα και να διαβάσει το QR code. Η εφαρμογή αποκωδικοποιεί και εξετάζει εάν το αναγνωριστικό ανήκει στο εργαστήριο ή στο μάθημα. Η όλη διαδικασία επιτυγχάνεται με δύο πακέτα του Flutter. Πακέτα τα οποία έχουν δημιουργηθεί από ανθρώπους της κοινότητας. Το πρώτο είναι για να δημιουργείται κάθε φορά το QR code του φοιτητή και το δεύτερο για να αναγνώσει το QR code και να πάρει την τιμή.

Ο κώδικας είναι πολύ πιο απλός, εύκολος στην ανάγνωση και στην διατήρηση. Λάθη είναι πολύ πιο δύσκολο να γίνουν. Τέλος, όλος ο κώδικας είναι σε Dart και άρα η βάση του κώδικα δεν αλλάζει. Τα θετικά είναι επίσης, ότι σε αντίθεση με το NFC, όλες οι συσκευές διαθέτουν κάμερα και η εφαρμογή μπορεί να αναπτυχθεί σε Android και iOS.

Πριν προχωρήσουμε, ας δούμε επίσης το QR λίγο πιο αναλυτικά.

2.3.1 QR code - Με λίγα λόγια

Το QR code, είναι μια εικόνα που απεικονίζει έναν δισδιάστατο πίνακα με μαύρα τετράγωνα σε άσπρο φόντο. Εκεί αποθηκεύεται η πληροφορία την οποία μετά τα κινητά μπορούν να διαβάσουν. Χρησιμοποιεί τέσσερις διαφορετικούς τρόπους (numeric, alphanumeric, byte/binary, Kanji/Kana) για να μπορέσει να αποθηκεύσει τα δεδομένα του πιο αποτελεσματικά. (Bhardwaj, Garg and Shekhar 2022).

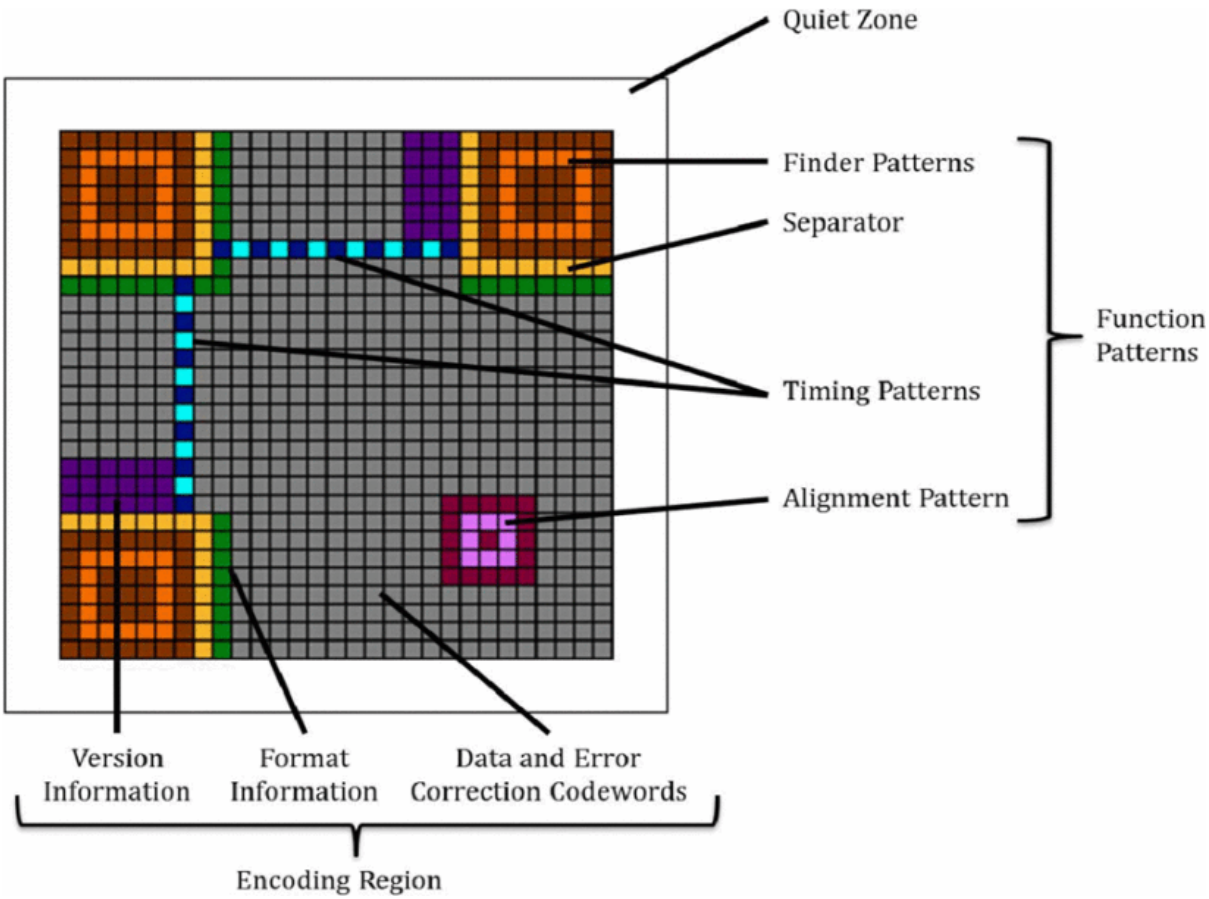
https://upload.wikimedia.org/wikipedia/commons/thumb/d/d0/QR_code_for_mobile_English_Wikipedia.svg/1200px-QR_code_for_mobile_English_Wikipedia.svg.png

2.3.2 Εκδόσεις

Οι εκδόσεις των QR code κυμαίνονται από την έκδοση 1-40. Κάθε έκδοση περιέχει διαφορετικό αριθμό τετραγώνων (άσπρων, μαύρων). Για παράδειγμα, στην πρώτη έκδοση έχουμε 21x21 τετραγωνάκια, ενώ, στην έκδοση 40 έχουμε 177x177. Συνεπώς είναι εμφανές, ότι στην έκδοση 40 το QR code μπορεί να περιέχει παραπάνω δεδομένα από την πρώτη έκδοση. (Tiwari 2016)

2.3.3 Δομή του QR code

Σε μία πιο ειδική ανάλυση, το QR code είναι ένας δυσδιάστατος πίνακας με μαύρα τετραγωνάκια τα οποία σχηματίζουν κάποια μοτίβα, τα οποία βοηθάνε στις διάφορες λειτουργίες που θα δούμε παρακάτω. Επίσης περιέχεται και μια περιοχή που χρησιμοποιείται αποκλειστικά για την αποκωδικοποίηση. Τέλος, πρέπει να υπάρχει μια ζώνη «ησυχίας» στο εξωτερικό σημείο. (Tiwari 2016)



2.3.3.1 Βασικά Μοτίβα

Στα βασικά μοτίβα συγκαταλέγονται, τα μοτίβα εύρεσης, τα μοτίβα διαχωρισμού, τα μοτίβα χρονισμού και μοτίβα ευθυγράμμισης. Στον παρακάτω πίνακα αναλύονται οι ρόλοι των κάθε μοτίβων που παρατέθηκαν.

Μοτίβο	Ρόλος
Εύρεσης	Τα μοτίβα εύρεσης εμφανίζονται στις τρείς γωνίες του πίνακα. Στις δύο πάνω και την αριστερή κάτω. Κάθε μοτίβο αποτελείται από μαύρα τετράγωνα που σχηματίζουν έναν πίνακα 7x7, εσωτερικά από ένα πίνακα με άσπρα τετράγωνα

	5x5 και τέλος, έναν πίνακα με μαύρα τετράγωνα 3x3. Αυτή η συνεχής αλλαγή από μαύρο σε άσπρο και αντίστροφα γίνεται επίτηδες ώστε να μπορεί εύκολα να αναγνωριστεί από τους QR αναγνώστες και να μπορέσει να προσανατολιστεί σωστά για την αποκωδικοποίηση.
Διαχωρισμού	Τα μοτίβα διαχωρισμού όπως μαρτυρά και η λέξη διαχωρίζουν τα μοτίβα εύρεσης και την περιοχή κωδικοποίησης. Είναι μια γραμμή με άσπρα τετράγωνα.
Χρονισμού	Τα μοτίβα χρονισμού εμφανίζονται σε δύο σημεία και με δύο τρόπους. Η οριζόντια γραμμή εμφανίζεται στην έκτη γραμμή του QR code, ανάμεσα στα μοτίβα διαχωρισμού. Η κατακόρυφη γραμμή εμφανίζεται στην έκτη στήλη επίσης μεταξύ των διαχωριστών. Ο ρόλος τους είναι να διευκολύνουν την εύρεση της πυκνότητας των συμβόλων, τις συντεταγμένες των τετραγώνων που απαρτίζουν τον QR code και την περιοχή που αναφέρεται η έκδοση.
Ευθυγράμμισης	Τα μοτίβα ευθυγράμμισης αποτελούνται από έναν πίνακα 5x5 με μαύρα τετραγωνάκια. Στο εσωτερικό, περιέχουν 3x3 με άσπρα τετραγωνάκια και ένα μαύρο τετραγωνάκι στην μέση. Τα QR με έκδοση πάνω από δύο, πρέπει να έχουν μοτίβα ευθυγράμμισης, όσα και η έκδοση των συμβόλων.

Πίνακας 2. Ρόλοι των μοτίβων στο QR code.

2.3.3.2 Λοιπά μοτίβα

Στα λοιπά μοτίβα του QR code, περιλαμβάνονται η περιοχή κωδικοποίησης και η περιοχή «ησυχίας». Όπως πάνω, ακολουθεί επίσης ένας πίνακας ο οποίος εξηγεί το ρόλο των λοιπών μοτίβων.

Μοτίβο	Ρόλος
Περιοχή κωδικοποίησης	Περιέχει πληροφορίες για την μορφοποίηση, για την έκδοση και περιέχει κώδικες για τα δεδομένα και τις διορθώσεις των λαθών.
Περιοχή «ησυχίας»	Είναι μια περιοχή 4x4 γύρων από το QR code χωρίς δεδομένα και χρησιμοποιείται για να είναι σίγουρο ότι δεν θα μπερδευτούν τα δεδομένα του QR με τα γύρω κείμενα.

Πίνακας 3. Λοιπά μοτίβα στο QR code.

2.3.4 Κωδικοποίηση & αποκωδικοποίηση ενός QR code

2.3.4.1 Κωδικοποίηση

Για την κωδικοποίηση ενός κειμένου σε QR code απαιτούνται τα εξής βήματα. Πρώτα, θα πρέπει να γίνει μια **ανάλυση** του κειμένου και να αποφασιστεί με ποιον τρόπο θα γίνει η κωδικοποίηση του κειμένου. Όπως είδαμε και παραπάνω, υπάρχουν τέσσερις διαφορετικοί τρόποι για να γίνει η κωδικοποίηση ενός κειμένου σε QR code.

Ο κάθε τρόπος κωδικοποιεί το κείμενο σαν αλφαριθμητικά από bit. Όλοι οι τρόποι προσπαθούν να χρησιμοποιήσουν τα λιγότερα bit. Αφού γίνει η ανάλυση, το κείμενο κωδικοποιείται. Εν συνεχεία, στην αρχή του QR code και πριν τα δεδομένα, **γράφεται ο τρόπος με τον οποίον έγινε η κωδικοποίηση**. Αυτός είναι ένα αλφαριθμητικό τεσσάρων bit. Επιπλέον, **γράφεται και ο αριθμός των χαρακτήρων** που κωδικοποιήθηκαν. Ένα επίσης πολύ σημαντικό κομμάτι της κωδικοποίησης, είναι οι **κώδικες διόρθωσης σφαλμάτων**. Αυτοί γράφονται σε διάφορα σημεία του QR code. Είναι πολύ σημαντικοί, καθώς, οι QR αναγνώστες διαβάζουν τα δεδομένα και αυτούς και μπορούν να είναι βέβαιοι ότι η ανάγνωση ήταν επιτυχής. Ένα από τελευταία βήματα, είναι, η αλλαγή των δεδομένων του QR σε ένα **πρότυπο** γνωστό από τους αναγνώστες καθώς τα μοτίβα που σχηματίζονται όπως είδαμε και παραπάνω δυσκολεύουν την ανάγνωση. Τέλος, προστίθενται **εικονοστοιχεία** στα κενά που έμειναν κενά από τα προηγούμενα βήματα και ίσως πληροφορίες για την έκδοση του QR code που χρησιμοποιείται. (Tiwari 2016)

2.3.4.2 Αποκωδικοποίηση

Ένας αναγνώστης για να αποκωδικοποιήσει ένα QR code, θα πρέπει να ακολουθήσει τη ανάποδη διαδικασία από αυτή της κωδικοποίησης. Αρχικά, διαβάζονται τα άσπρα και μαύρα τετραγωνάκια ως ένας πίνακας από 1 & 0. Στην συνέχεια, πρέπει να αποκωδικοποιηθούν οι πληροφορίες για την μορφή που θα φανούν χρήσιμες παρακάτω. Έπειτα, βρίσκεται το πρότυπο με το οποίο έχει κωδικοποιηθεί, κάνοντας XOR μεταξύ των πληροφοριών της μορφής με τα bit της περιοχής κωδικοποίησης. Από αυτά, έχουν ανακτηθεί τα δεδομένα και οι κώδικες διόρθωσης σφαλμάτων. Γίνεται αν χρειαστεί διόρθωση σφαλμάτων και προκύπτει το τελικό κείμενο. (Tiwari 2016)

2.4 Καλύτερη λύση

Αφού παρουσιάσαμε και τις δύο λύσεις, μπορούμε να δούμε ότι η καλύτερη λύση προκύπτει ότι είναι η λύση του QR code.

Το QR code έχει αρκετά πλεονεκτήματα σαν λύση στο πρόβλημα. Αρχικά, το μόνο που χρειάζεται για να λειτουργήσει είναι η κάμερα του κινητού τηλεφώνου. Όλα τα κινητά τηλέφωνα έχουν κάμερα στις μέρες μας. Από την άλλη η τεχνολογία NFC δεν

υπάρχει σε όλα τα κινητά. Επιπλέον, όπως είδαμε παραπάνω, η Apple δεν επιτρέπει στους προγραμματιστές να χρησιμοποιήσουν το NFC της συσκευής εκτός εάν πρόκειται για τράπεζες. Αυτοί οι δύο λόγοι καθιστούν την επιλογή του NFC απαγορευτική για μια τέτοια εφαρμογή, καθώς ούτε όλοι οι άνθρωποι διαθέτουν NFC, αλλά ούτε και Android. Τέλος, πολλές εφαρμογές που έχουν φτιαχτεί τον τελευταίο καιρό, για παράδειγμα από το κράτος, κάνουν χρήση της τεχνολογίας QR code, για τις ταυτοποιήσεις. Παράδειγμα αποτελεί η εφαρμογή Gov.gr Wallet.

3. Περιγραφή Καλύτερης Λύσης

Έχοντας διαλέξει την τεχνολογία QR code, θα γίνει περιγραφή της εφαρμογής που θα κάνει χρήση αυτής της τεχνολογίας. Η εφαρμογή θα δημιουργηθεί με την βοήθεια του Flutter. Παρατίθενται πληροφορίες για το Flutter παρακάτω.

3.1 Flutter

Το Flutter σύμφωνα με (Google LLC 2023) είναι ένα εργαλείο ανοιχτού κώδικα της Google που αναπτύχθηκε για να κάνει τη δημιουργία εφαρμογών πολλαπλών πλατφόρμων πιο εύκολη. Μια βάση κώδικα είναι αρκετή για να τρέξει η εφαρμογή σε έναν φυλλομετρητή, σε λειτουργικά συστήματα σαν παραδοσιακή εφαρμογή ή σαν εφαρμογή κινητού τηλεφώνου. Υποστηρίζονται όλα τα γνωστά λειτουργικά συστήματα (με την ανάπτυξη για το Linux να είναι σε πρώιμο στάδιο ακόμα) καθώς και οι φυλλομετρητές.

Είναι μια νέα τεχνολογία η οποία γίνεται καλύτερη μέρα με την μέρα, έχει μεγάλη υποστήριξη από την Google καθώς και από την κοινότητα. Είναι πιο γρήγορη χάρη στην Dart από την "αντίπαλη" React Native - ένα εργαλείο αντίστοιχο με το Flutter που γράφεται με Javascript – όσο αναφορά εφαρμογές κινητών τηλεφώνων σύμφωνα με (Lyman 2022).

Για το υποκείμενο λειτουργικό σύστημα, οι εφαρμογές Flutter συσκευάζονται με τον ίδιο τρόπο όπως κάθε άλλη εγγενής εφαρμογή. Ένας ειδικός για την πλατφόρμα ενσωματωτής παρέχει ένα σημείο εισόδου· συντονίζεται με το υποκείμενο λειτουργικό σύστημα για πρόσβαση σε υπηρεσίες όπως επιφάνειες απόδοσης, προσβασιμότητα και είσοδος· και διαχειρίζεται το βρόχο γεγονότων μηνυμάτων. Ο ενσωματωτής είναι γραμμένος σε μία γλώσσα κατάλληλη για την πλατφόρμα: αυτή τη στιγμή Java και C++ για Android, Objective-C/Objective-C++ για iOS και macOS, και C++ για Windows και Linux. Χρησιμοποιώντας τον ενσωματωτή, ο κώδικας Flutter μπορεί να ενσωματωθεί σε μια υπάρχουσα εφαρμογή ως μονάδα, ή ο κώδικας μπορεί να αποτελεί το σύνολο του περιεχομένου της εφαρμογής. Το Flutter περιλαμβάνει αριθμό ενσωματωτών για τις κοινές στοχευμένες πλατφόρμες, αλλά υπάρχουν και άλλοι ενσωματωτές.

Στον πυρήνα του Flutter βρίσκεται ο κινητήρας Flutter, ο οποίος είναι γραμμένος κυρίως σε C++ και υποστηρίζει τα πρωτόγονα απαραίτητα για να υποστηρίξουν όλες τις εφαρμογές Flutter. Ο κινητήρας είναι υπεύθυνος για την απόδοση σκηνών που έχουν συνθεθεί όταν χρειάζεται να ζωγραφιστεί ένα νέο πλαίσιο. Παρέχει την χαμηλού επιπέδου υλοποίηση του πυρήνα API του Flutter, περιλαμβάνοντας τα γραφικά (μέσω Impeller στο iOS και προσεχώς στο Android και macOS, και Skia σε άλλες πλατφόρμες) την διαρρύθμιση κειμένου, το αρχείο και δίκτυο I/O, υποστήριξη προσβασιμότητας, αρχιτεκτονική πρόσθετων και ένα Dart runtime και συντακτικό toolchain.

Ο κινητήρας εκτίθεται στο Flutter framework μέσω του dart:ui, το οποίο περιβάλλει τον υποκείμενο κώδικα C++ σε κλάσεις Dart. Αυτή η βιβλιοθήκη εκθέτει τα πρωτόγονα χαμηλού επιπέδου, όπως κλάσεις για την κίνηση εισόδου, γραφικών και υποσυστημάτων απόδοσης κειμένου.

Συνήθως, οι προγραμματιστές αλληλεπιδρούν με το Flutter μέσω του Flutter framework, το οποίο παρέχει ένα σύγχρονο, ανταποκρινόμενο framework γραμμένο στη γλώσσα Dart. Περιλαμβάνει ένα πλούσιο σετ πλατφορμών, διάταξης και βασικών βιβλιοθηκών, συντεθειμένων από μια σειρά στρωμάτων. Εργαζόμενοι από τα κάτω προς τα πάνω, έχουμε:

Βασικές κλάσεις και υπηρεσίες κτιρίων μπλοκ όπως κινήσεις, ζωγραφική και gestures που προσφέρουν συνήθως χρησιμοποιημένες αφαίρεσεις πάνω από το υποκείμενο ίδρυμα.
Το στρώμα απόδοσης παρέχει μια αφαίρεση για την αντιμετώπιση της διάταξης. Με αυτό το στρώμα, μπορείτε να δημιουργήσετε ένα δέντρο από αντικείμενα απόδοσης. Μπορείτε να χειρίζεστε αυτά τα αντικείμενα δυναμικά, με το δέντρο να ενημερώνεται αυτόματα τη διάταξη για να αντανακλούν τις αλλαγές σας.
Το στρώμα widgets είναι μια αφαίρεση σύνθεσης. Κάθε αντικείμενο απόδοσης στο στρώμα απόδοσης έχει μια αντίστοιχη κλάση στο στρώμα widgets. Επιπλέον, το στρώμα widgets σας επιτρέπει να ορίσετε συνδυασμούς κλάσεων που μπορείτε να χρησιμοποιήσετε ξανά. Αυτό είναι το στρώμα στο οποίο εισάγεται το μοντέλο ανταποκρινόμενου προγραμματισμού.

Οι βιβλιοθήκες Material και Cupertino προσφέρουν πλήρεις συνόλου ελέγχων που χρησιμοποιούν τις πρωτόγονα σύνθεσης του στρώματος widget για να υλοποιήσουν τις γλώσσες σχεδίασης Material ή iOS.

Το Flutter framework είναι σχετικά μικρό· πολλές υψηλότερες δυνατότητες που μπορεί να χρησιμοποιήσουν οι προγραμματιστές υλοποιούνται ως πακέτα, συμπεριλαμβανομένων των πρόσθετων πλατφόρμας όπως κάμερα και webview, καθώς και υπηρεσίες που είναι ανεξάρτητες από την πλατφόρμα όπως χαρακτήρες, http, και animations που βασίζονται στις βασικές βιβλιοθήκες Dart και Flutter. Μερικά από αυτά τα πακέτα προέρχονται από το ευρύτερο οικοσύστημα, καλύπτοντας υπηρεσίες όπως πληρωμές εντός εφαρμογής, ταυτοποίηση Apple και animations. (Google LLC 2023)

3.2 Firebase

Η εφαρμογή μας, για να λειτουργήσει, βασίζεται στο Firebase. Μπορούμε να πούμε ότι είναι ο βασικός πυλώνας της, καθώς, το χρησιμοποιούμε για να αποθηκεύσουμε όλα τα δεδομένα μας και να τακτοποιήσει τους χρήστες. Στην δική μας περίπτωση, όλα τα μαθήματα, τα εργαστήρια, οι φοιτητές και οι καθηγητές είναι αποθηκευμένοι εκεί.

Το Firebase είναι μια δέσμη από υπηρεσίες που αναπτύχθηκε από την Google και τρέχει στο νέφος. Συνήθως χρησιμοποιείται σαν βάση δεδομένων, σαν πάροχος ειδοποιήσεων στις κινητές συσκευές και πολλές ακόμα υπηρεσίες σύμφωνα με (Wikipedia 2014).

Το Firebase σύμφωνα με (Fanatic 2024) χρησιμοποιείται για μια ποικιλία σκοπών στην ανάπτυξη εφαρμογών, περιλαμβανομένων:

Βάσεις Δεδομένων σε Πραγματικό Χρόνο: Αποθήκευση και συγχρονισμός δεδομένων μεταξύ χρηστών και εφαρμογής σε πραγματικό χρόνο.

Αυθεντικοποίηση: Απλοποίηση εγγραφής και σύνδεσης χρηστών με διάφορες μεθόδους αυθεντικοποίησης.

Λειτουργίες του νέφους: Δημιουργία προσαρμοσμένου κώδικα backend χωρίς διαχείριση διακομιστών.

Φιλοξενία: Γρηγορότερη παράδοση περιεχομένου ιστού με ένα ασφαλές, παγκόσμιο CDN.

Αναλυτικά Στοιχεία: Κατανόηση για τη χρήση της εφαρμογής και την αλληλεπίδραση των χρηστών.

Μηνύματα μέσω νέφους: Αποστολή στοχευμένων μηνυμάτων και ειδοποιήσεων στους χρήστες.

Επίσης, το Firebase προτιμάται από τους προγραμματιστές για πολλούς λόγους: Αποδοτικότητα: Παρέχει πολλές λειτουργίες "έτοιμες προς χρήση", μειώνοντας τον χρόνο ανάπτυξης.

Κλιμακούμενοτητα: Οι υπηρεσίες του Firebase κλιμακώνονται αυτόματα, διαχειριζόμενες το backend της εφαρμογής καθώς αυτή μεγαλώνει.

Ένταξη: Ενσωματώνεται ομαλά με άλλες υπηρεσίες της Google και εργαλεία τρίτων. Πραγματικός Χρόνος: Οι βάσεις δεδομένων Realtime και Firestore του Firebase παρέχουν ζωντανή συγχρονιζόμενη ενημέρωση δεδομένων σε όλους τους πελάτες.

Πολυπλατφορμικότητα: Υπ (Fanatic 2024)οστηρίζει Android, iOS, ιστό και ακόμα ανάπτυξη παιχνιδιών.

Στην εφαρμογή μας χρησιμοποιείται η υπηρεσία Cloud Firestore, μια δωρεάν -μέχρι κάποιο όριο- υπηρεσία για αποθήκευση δεδομένων (βάση δεδομένων), το Firebase Authentication, για να μπορούμε να ταυτοποιούμε τους χρήστες της εφαρμογής, καθώς και το Firebase Storage, για την αποθήκευση των φωτογραφιών των χρηστών. Αυτές οι δύο υπηρεσίες δίνονται χωρίς να κάνουμε κάτι, με το που δημιουργήσουμε μια εφαρμογή στο Firebase και την συνδέσουμε με το πραγματική εφαρμογή που προγραμματίζουμε.

Το Cloud Firestore είναι μια no-SQL βάση δεδομένων. Αυτό σημαίνει ότι αντί για πίνακες, έχουμε συλλογές με έγγραφα, τα οποία έχουν τα δεδομένα μας. Αυτή η τεχνική κάνει πιο εύκολη και γρήγορη την διαδικασία της ερώτησης στην βάση δεδομένων, τη διαδικασία αποθήκευσης και ανάκτησης των δεδομένων. Τα δεδομένα

που αποθηκεύονται μπορεί να είναι απλά αλφαριθμητικά και νούμερα, μέχρι συλλογές σύμφωνα με (Google LLC χ.χ.).

Το γεγονός ότι τρέχει στο νέφος κάνει πιο εύκολη την υποστήριξη της εφαρμογής καθώς δεν χρειάζεται να διατεθούν διακομιστές. Το μειονέκτημα, είναι ότι οι υπηρεσίες κοστίζουν. Το Firebase δίνει δύο πλάνα τιμολόγησης. Το πλάνο “Spark”, το οποίο είναι χωρίς χρέωση (μέχρι κάποιο όριο) και το “Blaze” το οποίο κοστίζει ανάλογα με την χρήση. Στην εφαρμογή χρησιμοποιήθηκε το πρώτο. Παρακάτω δίνεται ένας πίνακας με την τιμολόγηση για την υπηρεσία Cloud Firestore. Αμέσως μετά, ένας πίνακας για την τιμολόγηση στο Cloud Storage.

	Τιμολόγηση	
	Spark	Blaze
Ανάγνωση εγγράφου	50 χιλιάδες / μέρα	0,037€ ανά 100.000 έγγραφα
Εγγραφή εγγράφων	20 χιλιάδες / μέρα	0,11€ ανά 100.000 έγγραφα
Διαγραφή εγγράφων	20 χιλιάδες / μέρα	0,012€ ανά 100.000 έγγραφα
Αποθήκευση δεδομένων	1 GiB	0,11€ ανά 100.000 έγγραφα

Πίνακας 4. Τιμολόγηση Cloud Firestore (τιμές Μαΐου 2023).

	Τιμολόγηση	
	Spark	Blaze
Αποθήκευση σε GB	5 GB	0,024€ / GB
Κατέβασμα σε GB	1 GB / ημέρα	0,11€ / GB
Αριθμός μεταφορτώσεων	20 χιλιάδες / μέρα	0,047€ / 10 χιλιάδες
Αριθμός κατεβασμάτων	50 χιλιάδες / μέρα	0,0037€ / 10 χιλιάδες
Δυνατότητα πολλαπλών bucket ανά πρότζεκτ	Όχι	Ναι

Πίνακας 5. Τιμολόγηση Cloud Storage (τιμές Μαΐου 2023).

3.2.1 Σχήμα βάσης δεδομένων

Εφόσον επιλέχθηκε μια noSQL βάση δεδομένων, δεν υπάρχει ένα αυστηρό και δομημένο σχήμα της βάσης, ούτε και σχέσεις μεταξύ των πινάκων. Παρόλα αυτά, υπάρχει ένα άτυπο σχήμα και αυτό αποτελείται από τις συλλογές που μπορεί εύκολα κάποιος να σκεφτεί για την εφαρμογή που αναπτύσσεται. Προφανώς, θα χρειαστεί μια συλλογή για τους χρήστες, καθηγητές και φοιτητές. Εκεί μέσα, θα φυλάσσονται τα στοιχεία του κάθε χρήστη, όνομα, επίθετο, e-mail και όλες οι υπόλοιπες πληροφορίες που τους χαρακτηρίζουν.

Έπειτα, θα χρειαστούμε μια συλλογή για τα εργαστήρια. Αυτή η συλλογή είναι η πιο περίπλοκη του σχήματος, καθώς, κάθε εργαστήριο περιέχει τουλάχιστον ένα μάθημα και για κάθε μάθημα, πρέπει να ξέρει ποιοι χρήστες (φοιτητές) μπορούν να βρίσκονται εκεί. Οπότε κάθε εργαστήριο έχει εκτός από το όνομά του που το χαρακτηρίζει, μια

λίστα με μαθήματα, τα οποία περιέχουν μέσα, τα στοιχεία του μαθήματος (μέρα και ώρα) και μια λίστα με τα αναγνωριστικά των χρηστών (φοιτητών) που έχουν πρόσβαση σε αυτό.

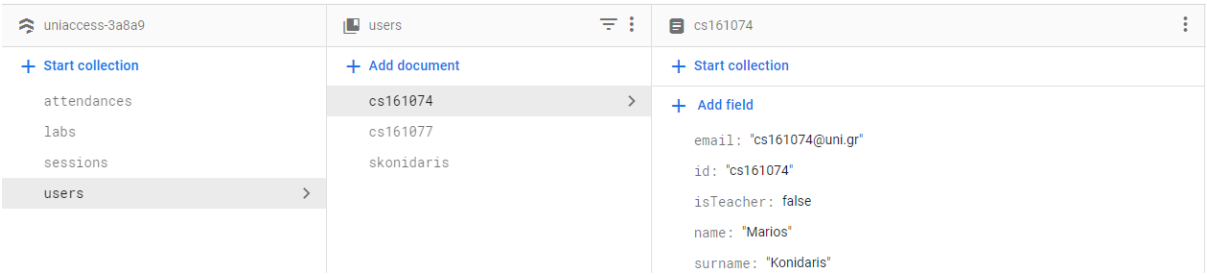
Επιπλέον, χρειάζεται μια συλλογή για τις παρουσίες του κάθε μαθήματος. Ουσιαστικά, αυτή η συλλογή, θα μπορούσε να ονομαστεί και μαθήματα, γιατί κάθε έγγραφο της συλλογής, είναι το κάθε μάθημα που υπάρχει στην εφαρμογή αλλά έμφαση δόθηκε στις παρουσίες. Έτσι, κάθε μάθημα έχει μια λίστα με τους φοιτητές και για κάθε φοιτητή, ο αριθμός των παρουσιών του. Αυτή η λίστα ανανεώνεται κάθε φορά που ο καθηγητής αποθηκεύει και μια νέα συνεδρία, αφού αλλάζουν οι παρουσίες των φοιτητών.

Τέλος, σειρά έχει η συλλογή των συνεδριών. Εκεί αποθηκεύονται οι πληροφορίες για κάθε συνεδρία που έλαβε χώρα. Οι πληροφορίες του μαθήματος, το εργαστήριο, η ημερομηνία αποθήκευσης της συνεδρίας και φυσικά οι φοιτητές που ήταν εκεί.

Παρακάτω, παρατίθεται μια σχηματική αναπαράσταση όλων όσων ειπώθηκαν παραπάνω.

ΟΝΟΜΑ ΣΥΛΛΟΓΗΣ: users		
Πεδίο	**Τύπος Δεδομένων**	**Περιγραφή**
Email	Συμβολοσειρά	Το email του χρήστη
Id	Συμβολοσειρά	Το αναγνωριστικό του χρήστη
isTeacher	Λογική μεταβλητή	Εάν ο χρήστης είναι ή δεν είναι καθηγητής
Name	Συμβολοσειρά	Το όνομα του χρήστη
Surname	Συμβολοσειρά	Το επώνυμο του χρήστη

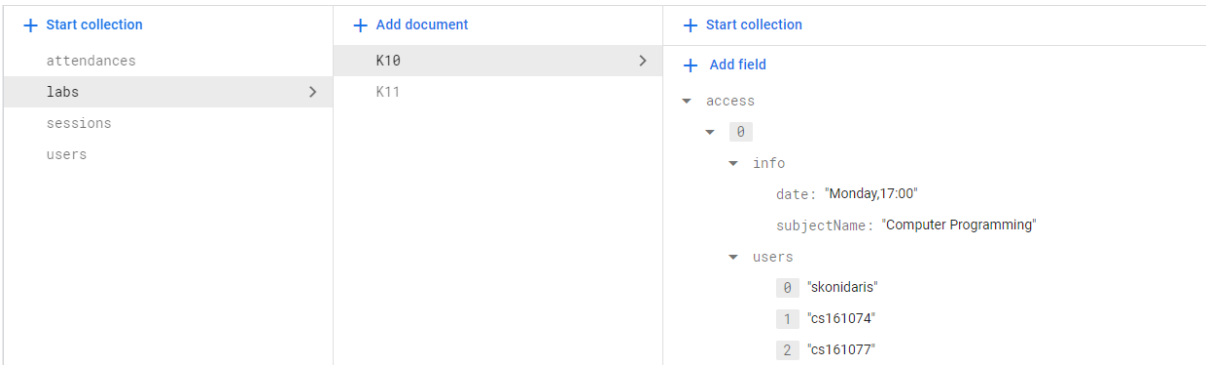
Πίνακας 6. Συλλογή χρηστών



Εικόνα 3. Συλλογή χρηστών

<table>
<tr><th colspan="3">ΟΝΟΜΑ ΣΥΛΛΟΓΗΣ: labs</th></tr>
<tr><th>Πεδίο</th><th>Τύπος Δεδομένων</th><th>Περιγραφή</th></tr>
<tr><td>Access</td><td colspan="2">Λίστα αντικειμένων (που αποτελούνται από ένα αντικείμενα και μία λίστα)</td><td rowspan="6">Δείχνει την ημερομηνία και το μάθημα στο οποίο έχει πρόσβαση ο χρήστης</td></tr>
<tr><td rowspan="5"></td><td colspan="2">info</td></tr>
<tr><td>Πεδίο</td><td>Τύπος Δεδομένων</td></tr>
<tr><td>Date</td><td>Συμβολοσειρά</td></tr>
<tr><td>subjectName</td><td>Συμβολοσειρά</td></tr>
<tr><td colspan="2">users</td></tr>
<tr><td>Πεδίο</td><td>Τύπος Δεδομένων</td></tr>
<tr><td>Users</td><td>Λίστα συμβολοσειρών</td></tr>
<tr><td>Name</td><td colspan="2">Συμβολοσειρά</td><td>Το όνομα του εργαστηρίου</td></tr>
</table>

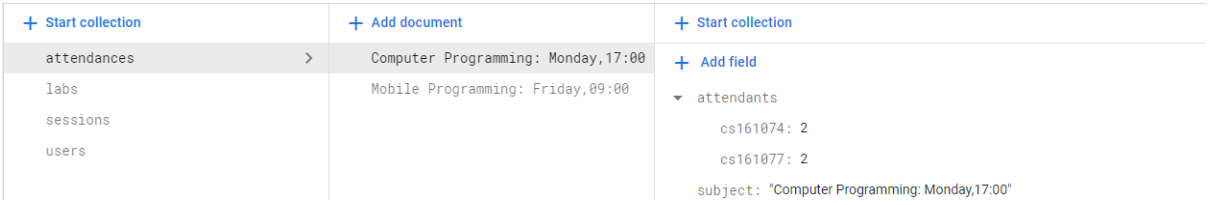
Πίνακας 7. Συλλογή εργαστηρίων



Εικόνα 4. Συλλογή εργαστηρίων

ΟΝΟΜΑ ΣΥΛΛΟΓΗΣ: attendances		
Πεδίο	**Τύπος Δεδομένων**	**Περιγραφή**
Attendants	Map συμβολοσειρών και αριθμών	Τα μοναδικά αναγνωριστικά του κάθε φοιτητή μαζί με τον αριθμό παρουσιών του στο συγκεκριμένο μάθημα
Subject	Συμβολοσειρά	Το μάθημα

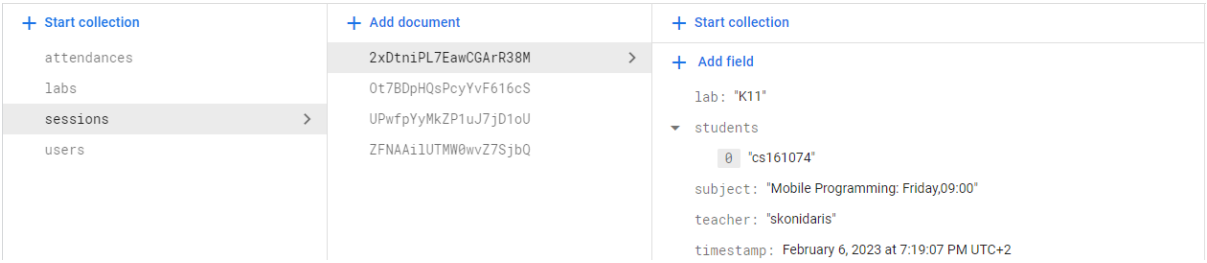
Πίνακας 8. Συλλογή Παρουσιών



Εικόνα 5. Συλλογή παρουσιών

ΟΝΟΜΑ ΣΥΛΛΟΓΗΣ: sessions		
Πεδίο	**Τύπος Δεδομένων**	**Περιγραφή**
Lab	Συμβολοσειρά	Το εργαστήριο
Students	Λίστα συμβολοσειρών	Τα αναγνωριστικά των φοιτητών που παρακολουθούν
Subject	Συμβολοσειρά	Το μάθημα
Teacher	Συμβολοσειρά	Το αναγνωριστικό του υπεύθυνου καθηγητή
Timestamp	Ημερομηνία	Η ημερομηνία που καταγράφηκε η συγκεκριμένη πληροφορία

Πίνακας 9. Συλλογή συνεδριών



Εικόνα 6. Συλλογή συνεδριών

3.3 Οθόνη σύνδεσης

Η εφαρμογή θα πρέπει αρχικά, να έχει μια οθόνη στην οποία ο χρήστης δίνει τα διαπιστευτήριά του για να συνδεθεί στην εφαρμογή. Κάθε χρήστης (φοιτητής και καθηγητής) έχει παραλάβει το όνομα χρήστη και κωδικό από την γραμματεία του τμήματος. Όπως κατά την διαδικασία εγγραφής στο τμήμα. Εάν δεν είναι σωστά τα διαπιστευτήριά του, τότε δεν μπορεί να συνδεθεί. Εφόσον ο χρήστης συνδεθεί, τότε θα μπορεί από μια μπάρα να επιλέξει ποια οθόνη θα θέλει να βλέπει. Οι οθόνες δεν

είναι ίδιες για τους φοιτητές και τους καθηγητές. Οι καθηγητές όπως είναι λογικό, έχουν παραπάνω δυνατότητες στην εφαρμογή.

3.4 Αρχική οθόνη

Παρόλα αυτά, και οι δύο χρήστες μπορούν να δουν την αρχική οθόνη, στην οποία εμφανίζονται τα στοιχεία τους, όπως το όνομά τους, τον μοναδικό κωδικό που τους έχει ανατεθεί από το τμήμα, το email τους και τον ρόλο (φοιτητής ή καθηγητής). Επίσης, κάθε χρήστης μπορεί να δει την εικόνα του και αυτό είναι ιδιαίτερα χρήσιμο για την διαδικασία ταυτοποίησης, όπως θα δούμε παρακάτω. Τέλος, οι φοιτητές έχουν την δυνατότητα με το πάτημα ενός κουμπιού (το οποίο βρίσκεται και αυτό στην αρχική οθόνη) να εμφανίσουν τον μοναδικό QR κωδικό τους.

3.5 Οθόνη εμφάνισης εργαστηρίων και μαθημάτων

Όλοι οι χρήστες μπορούν να δουν μια λίστα των εργαστηρίων των οποίων οι φοιτητές παρακολουθούν και οι καθηγητές διδάσκουν, καθώς και τα μαθήματα με τις ημέρες και ώρες που οι μεν παρακολουθούν και οι δε διδάσκουν.

3.6 Οθόνη ταυτοποίησης

Αυτή η οθόνη είναι ορατή μόνο στους καθηγητές. Ο χρήστης, επιλέγει το εργαστήριο που επιθυμεί, αυτόματα εμφανίζονται τα μαθήματα στις ώρες και μέρες που ο ίδιος έχει πρόσβαση. Επιλέγοντας ένα από αυτά, μπορεί να ξεκινήσει μια διαδικασία ταυτοποίησης.

Μόλις ξεκινήσει την διαδικασία ταυτοποίησης, αμέσως εμφανίζεται μια λίστα με τα ονόματα των φοιτητών που παρακολουθούν αυτό το μάθημα και άρα θεωρητικά θα «πρέπει» να είναι παρόντες. Όταν προσέλθει ένας φοιτητής, ο καθηγητής με το πάτημα ενός κουμπιού ανοίγει την κάμερα η οποία περιμένει να αναγνώσει ένα QR code. Από την άλλη, ο φοιτητής πατάει το κουμπί για να εμφανίσει το QR code του (που αναφέραμε παραπάνω). Η εφαρμογή εξάγει την πληροφορία, δηλαδή τον μοναδικό κωδικό του φοιτητή και ελέγχει στην βάση εάν αυτός/η έχει πρόσβαση. Εάν δεν έχει, μια κατάλληλη ένδειξη εμφανίζεται και τότε ο καθηγητής γνωρίζει ότι ο χρήστης δεν μπορεί να παρακολουθήσει το μάθημα. Εάν όμως έχει πρόσβαση, τότε ο καθηγητής μπορεί να αφήσει τον φοιτητή να παρακολουθήσει και ως δεύτερο μέτρο

ασφαλείας, μπορεί να ελέγξει και την φωτογραφία του φοιτητή που εμφανίζεται στην αρχική οθόνη του.

Αφού όλοι οι φοιτητές έχουν περάσει από την διαδικασία ταυτοποίησης, ο καθηγητής μπορεί να αποθηκεύσει την διαδικασία στην βάση δεδομένων. Αφού το κάνει αυτό, η διαδικασία αποθηκεύεται και οι παρουσίες του εκάστοτε φοιτητή για το συγκεκριμένο μάθημα ενημερώνονται.

3.7 Οθόνη προβολής προηγούμενων ταυτοποιήσεων

Σε αυτή την οθόνη έχει πρόσβαση και πάλι μόνο ο καθηγητής. Όταν μεταφερθεί, όλες οι ταυτοποιήσεις που έχει πραγματοποιήσει φορτώνονται και εμφανίζονται. Από εδώ, ο καθηγητής μπορεί να δει ανά πάσα στιγμή ποιοι φοιτητές ήταν παρών στο εκάστοτε μάθημα και τι παρουσίες είχαν εκείνη την χρονική στιγμή καθώς και την ημέρα και ώρα καταχώρησης της ταυτοποίησης. Επιπλέον, μπορεί να φιλτράρει τα αποτελέσματα για συγκεκριμένα εργαστήρια ή/και μαθήματα.

3.8 Τα πακέτα από το pub.dev

3.8.1 Σημείωση

Όλη η λειτουργικότητα του Firebase παρέχεται μέσα από πακέτα του pub.dev. Αφού όμως τα αναλύσαμε παραπάνω, δεν θα αναφερθούν σε αυτή την ενότητα. Παρακάτω παρατίθενται τα υπόλοιπα πακέτα που χρησιμοποιήθηκαν ώστε να εξασφαλιστεί η σωστή λειτουργία της εφαρμογής.

3.8.2 Πακέτα

3.8.2.1 provider

Είναι ένα πακέτο που κάνει την διαχείριση της κατάστασης της εφαρμογής πολύ πιο εύκολη. Πολλές φορές θα χρειαστεί να ανανεώσουμε αυτόματα την διεπαφή του χρήστη ανάλογα με την τιμή μιας μεταβλητής. Το provider κάνει αυτή την δουλειά πολύ πιο εύκολη. Το Flutter παρέχει ένα widget για αυτή την δουλειά (InheritedWidget), αλλά δεν είναι η καλύτερη πρακτική. Αντ' αυτού, το πιο σωστό είναι να χρησιμοποιηθεί ένα

πακέτο σαν το provider. Πρέπει να σημειωθεί ότι υπάρχουν πολλά άλλα πακέτα για την διαχείριση της κατάστασης μιας εφαρμογής. Το καθένα έχει τα πλεονεκτήματα και τα μειονεκτήματά του. Το provider είναι το πιο διαδεδομένο και το πιο προσιτό, για μικρές εφαρμογές και νέους προγραμματιστές με μικρή εμπειρία στο Flutter.

3.8.2.2 mobile_scanner

Είναι ένα πακέτο για την ανάγνωση QR codes.

3.8.2.3 qr_flutter

Είναι ένα πακέτο για την δημιουργία QR codes.

3.8.2.4 flutter_launcher_icons

Είναι ένα πακέτο για την δημιουργία του εικονιδίου της εφαρμογής. Δεν χρησιμοποιείται κατά την λειτουργία της εφαρμογής, αλλά το προσθέτουμε στο πρότζεκτ και το εκτελούμε μέσω ενός τερματικού για την δημιουργία του εικονιδίου.

3.8.2.5 showcaseview

Είναι ένα πακέτο που χρησιμοποιείται για την εμφάνιση υποδείξεων για την χρήστη στην διεπαφή. Σε πολλά σημεία θέλουμε να κάνουμε πιο κατανοητή τον ρόλο κάποιων κουμπιών για παράδειγμα.

3.8.2.6 lint

Είναι ένα πακέτο που ο προγραμματιστής μπορεί να θέσει κανόνες και σωστές πρακτικές στον τρόπο γραφής της εφαρμογής. Αυτό εξασφαλίζει ότι ο κάθε προγραμματιστής θα ακολουθεί τον ίδιο τρόπο γραφής του κώδικα.

4.6 Δομή πρότζεκτ

Παραπάνω εξηγήθηκαν οι λόγοι για την επιλογή του Flutter ως το εργαλείο για την ανάπτυξη της εφαρμογής, αλλά και το Firebase ως ο "πυλώνας" της.

Το επόμενο βήμα στην ανάπτυξη της εφαρμογής είναι να σχεδιαστεί το σχήμα της βάσης δεδομένων. Αν και οι noSQL βάσεις δεδομένων δεν έχουν κάποιο αυστηρό σχήμα, μπορεί να γίνει μια παρουσίαση των συλλογών που χρειάστηκαν. Για τις

ανάγκες της εφαρμογής θα πρέπει να κρατάμε πληροφορίες για τους χρήστες, για τα εργαστήρια, τις παρουσίες σε κάθε εργαστήριο και τις συνεδρίες που είχε ο κάθε καθηγητής για να μπορεί να έχει εικόνα. Έτσι λοιπόν, έχουμε τέσσερις συλλογές οι οποίες φαίνονται πιο αναλυτικά παρακάτω.

Αφού έχουμε φτιάξει την βάση δεδομένων, σειρά έχει η ανάπτυξη της εφαρμογής. Για να ξεκινήσουμε την ανάπτυξη της εφαρμογής στο Flutter, πρέπει να πρώτα να φτιάξουμε ένα πρότζεκτ. Τα παρακάτω βήματα χρησιμοποιούν το τερματικό, παρόλα αυτά, είναι δυνατή η χρήση του Visual Studio Code αλλά και του Android Studio / IntelliJ για την πραγματοποίηση τους. Βήμα πρώτο είναι να ανοίξουμε ένα τερματικό, και αφού πάμε στο σημείο του δίσκου που θέλουμε να το αποθηκεύσουμε γράφουμε την εντολή `flutter create <app>`. Είναι επίσης δυνατή η επιλογή της γλώσσας για το Android και το iOS, καθώς σε κάποιες περιπτώσεις ίσως χρειαστεί να γράψουμε κάτι σε ένα αρχείο ειδικό για το κάθε λειτουργικό. Οπότε η επιλογή της γλώσσας κάνει πιο εύκολη αυτή την δουλειά.

Σειρά έχει, αφού έχουμε δημιουργήσει το πρότζεκτ, να δούμε την δομή της εφαρμογής. Θα δούμε ένα τελικό στάδιο της δομής αλλά θα εξηγήσουμε τι υπάρχει και τι όχι από τα αρχικά στάδια.

.github/ISSUE_TEMPLATE

Ένας φάκελος που δημιουργείται αυτόματα από το Github εφόσον ενεργοποιήσουμε την αναφορά προβλημάτων από την κοινότητα. Δεν υπάρχει από την δημιουργία του πρότζεκτ.

.vscode

Ένας φάκελος που δημιουργεί αυτόματα το Visual Studio Code και περιέχει ένα json αρχείο με ρυθμίσεις έτσι ώστε όταν ένας άλλος εκτός από εμάς κλωνοποιήσει το πρότζεκτ και το ανοίξει στο δικό του Visual Studio Code να έχει τις ίδιες ρυθμίσεις με εμάς, κάτι που κάνει την εμπειρία καλύτερη. Δεν δημιουργείται από την αρχή.

Android

Ένας φάκελος με διάφορα αρχεία που χρησιμοποιούνται για την μεριά του Android. Αυτό τον φάκελο θα πρέπει να τον τροποποιήσουμε για να προσθέσουμε άδειες που

θέλουμε να έχει η εφαρμογή μας καθώς και εφόσον θέλουμε να προσθέσουμε τις υπηρεσίες του Firebase. Δημιουργείται από την αρχή.

Assets

Ένας φάκελος που δημιουργήσαμε και περιέχει την γραμματοσειρά αλλά και τις εικόνες που χρησιμοποιεί η εφαρμογή μας. Δεν δημιουργείται από την αρχή.

Ios

Ο αντίστοιχος φάκελος για iOS. Δημιουργείται από την αρχή.

Lib

Ο πιο σημαντικός φάκελος του πρότζεκτ, καθώς εκεί κατοικεί ολόκληρος ο κώδικας της εφαρμογής μας. Δημιουργείται από την αρχή.

Test

Ο φάκελος που κατοικούν οι δοκιμές της εφαρμογής εάν αυτές υπάρχουν. Δημιουργείται από την αρχή.

Web

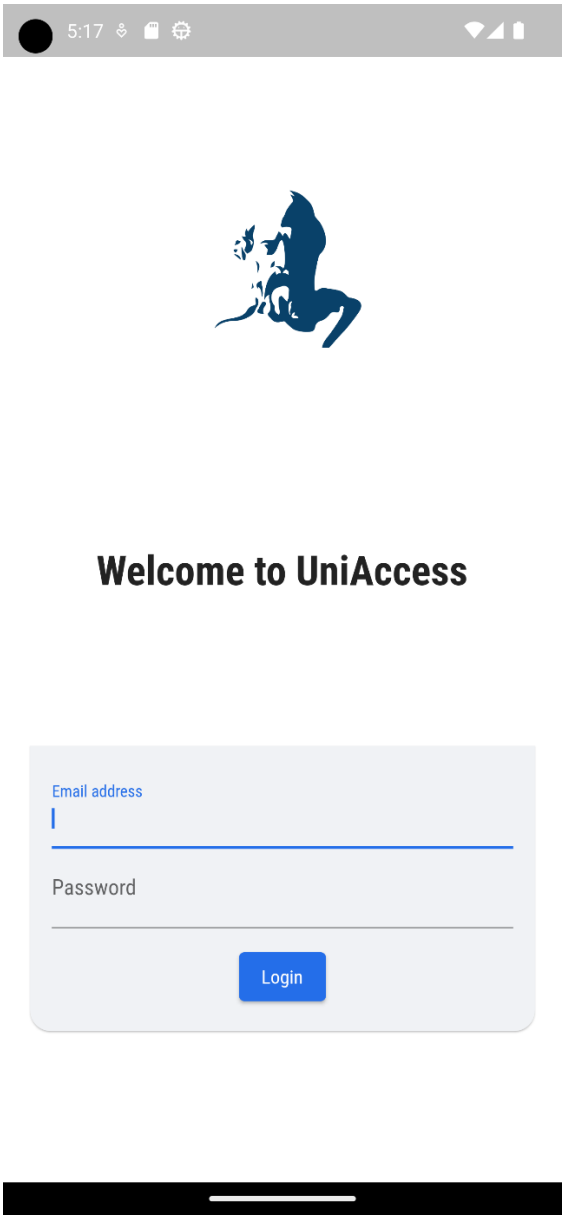
Ένας φάκελος που περιέχει αρχεία χρήσιμα για την εκκίνηση της εφαρμογής σε φυλλομετρητή. Δημιουργείται από την αρχή.

Windows

Ένας φάκελος που περιέχει αρχεία χρήσιμα για την εκκίνηση της εφαρμογής σαν εφαρμογή Windows. Αυτός ο φάκελος υπάρχει γιατί το λειτουργικό σύστημα στο οποίο αναπτύσσουμε την εφαρμογή είναι τα Windows. Εάν αναπτύσσαμε την εφαρμογή σε διανομή Linux θα λεγόταν «linux». Δημιουργείται από την αρχή.

4. Εγχειρίδιο Εφαρμογής

Σε αυτή την ενότητα, θα παραθέσουμε τις λειτουργίες της εφαρμογής δείχνοντας τις αντίστοιχες εικόνες για το πώς αυτές παρουσιάζονται αυτές, μέσα στην εφαρμογή. Θα ξεκινήσουμε με την είσοδο του χρήστη στην εφαρμογή, αφού αυτή είναι η πρώτη οθόνη που βλέπον οι χρήστες, όταν επισκέπτονται την εφαρμογή την πρώτη φορά.



Εικόνα 7. Είσοδος στην εφαρμογή

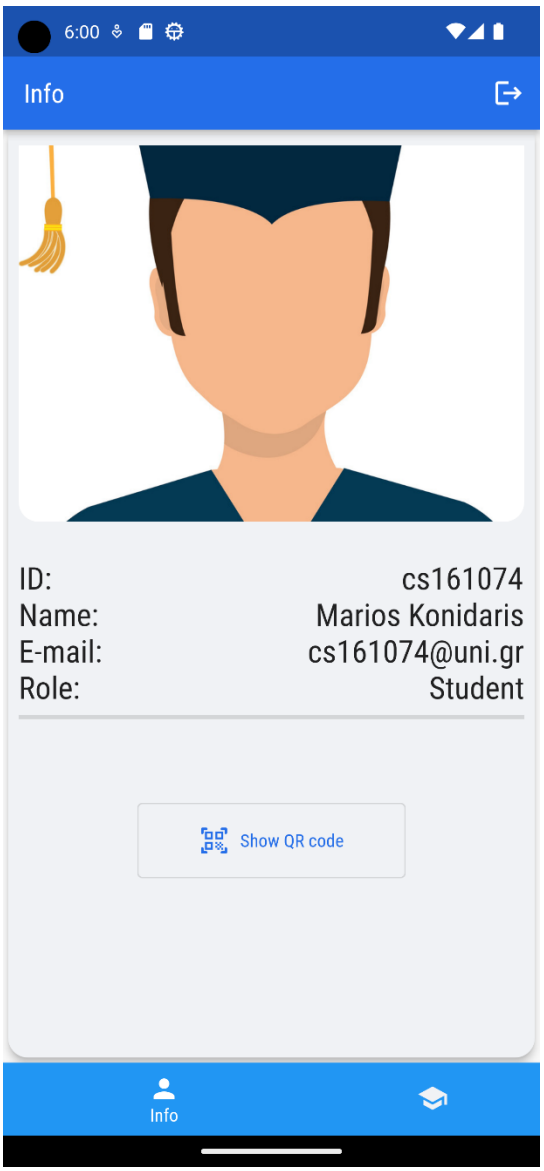
Αυτή είναι η οθόνη που αντικρίζει ο χρήστης είτε την πρώτη φορά που χρησιμοποιεί την εφαρμογή (όπως αναφέραμε), είτε εφόσον έχει επιλέξει να αποσυνδεθεί μέσα στην εφαρμογή, όπως θα δούμε παρακάτω.

Ο χρήστης καλείται να συμπληρώσει το e-mail και κωδικό που έχει παραλάβει από την γραμματεία, καθώς όλοι οι λογαριασμοί φτιάχνονται από την σχολή. Αφού συμπληρώσει αυτά με επιτυχία, μπορεί να προχωρήσει στην εφαρμογή.

Ανάλογα με τα στοιχεία που θα βάλει, το e-mail του, θα μεταφερθεί και στην ανάλογη κεντρική σελίδα. Οι φοιτητές έχουν διαφορετική διεπαφή από τους μαθητές. Θα συνεχίσουμε την υπόδειξη με τους φοιτητές και αφού τελειώσει, θα προχωρήσουμε στους καθηγητές.

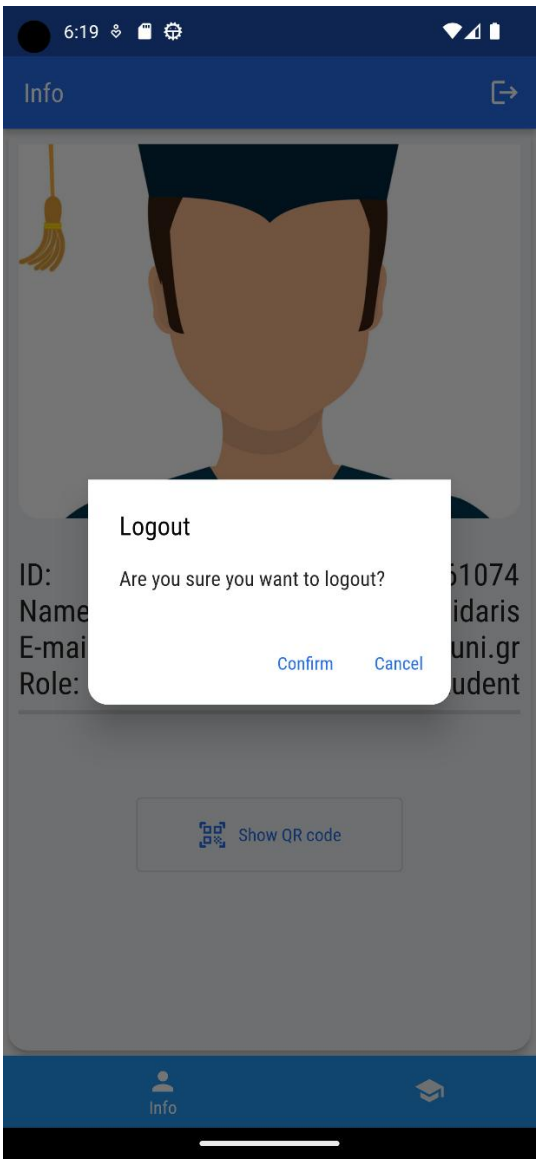
4.1 Λειτουργίες φοιτητή

Σειρά έχει η κεντρική οθόνη της εφαρμογής, όπως την βλέπουν οι μαθητές.



Εικόνα 8. Κεντρική οθόνη φοιτητή

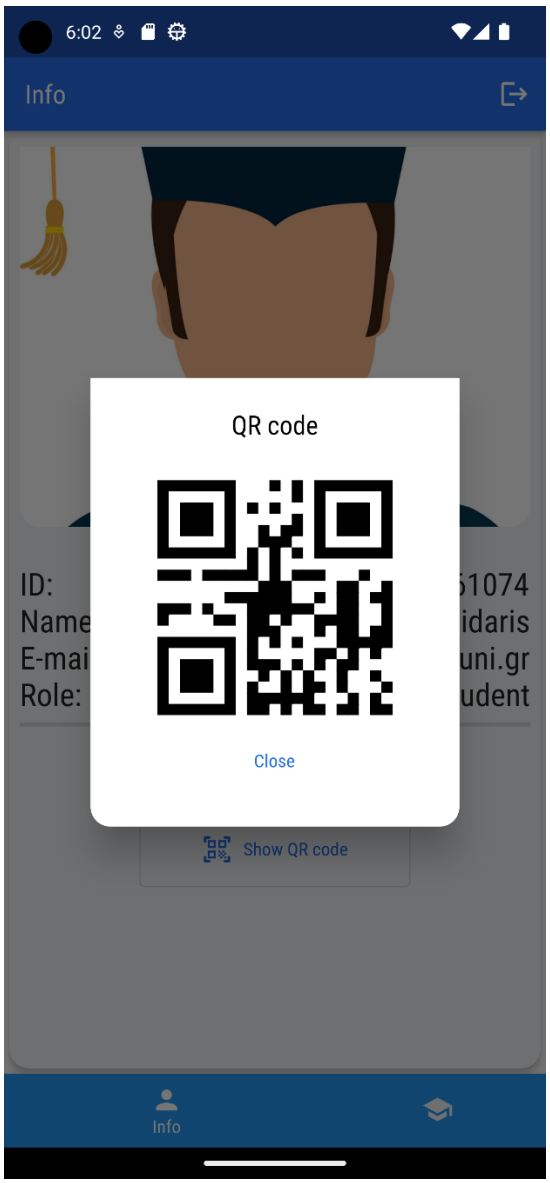
Η κεντρική οθόνη του φοιτητή χωρίζεται σε δύο μέρη. Στο πάνω μέρος, που εμφανίζονται οι πληροφορίες του, μαζί με την φωτογραφία του, καθώς και στο κάτω μέρος όπου υπάρχει ένα κουμπί για την εμφάνιση του μοναδικού QR code του φοιτητή. Τέλος, η οθόνη περιέχει μια μπάρα στο πάνω μέρος, η οποία ονομάζεται AppBar στο Flutter και μια μπάρα, με διάφορα κουμπιά στο κάτω μέρος, που οδηγούν σε άλλες οθόνες, που ονομάζεται BottomNavigationBar στο Flutter. Η πρώτη, τοποθετείται πάντα στο πάνω μέρος της οθόνης και συνήθως περιέχει έναν τίτλο που χαρακτηρίζει την οθόνη όπως για παράδειγμα εδώ, καθώς και κάποια κουμπιά με λειτουργίες στο δεξί μέρος. Εδώ είναι η αποσύνδεση από το σύστημα. Όταν ο χρήστης πατήσει αυτό το κουμπί, ένα αναδυόμενο παράθυρο εμφανίζεται για την επιβεβαίωση της αποσύνδεσης.



Εικόνα 9. Επιβεβαίωση αποσύνδεσης χρήστη

Η δεύτερη, είναι η μπάρα με την οποία ο χρήστης περιηγείται στις διάφορες οθόνες της εφαρμογής. Κάθε κουμπί περιγράφεται από ένα κατάλληλο εικονίδιο για να κάνει την περιήγηση πιο εύκολη. Στον φοιτητή οι επιλογές είναι δύο. Είτε η πρώτη οθόνη με τις πληροφορίες (αυτή που παρουσιάζεται τώρα), είτε η οθόνη που περιέχει όλα τα εργαστήρια με τα μαθήματα στα οποία έχει πρόσβαση.

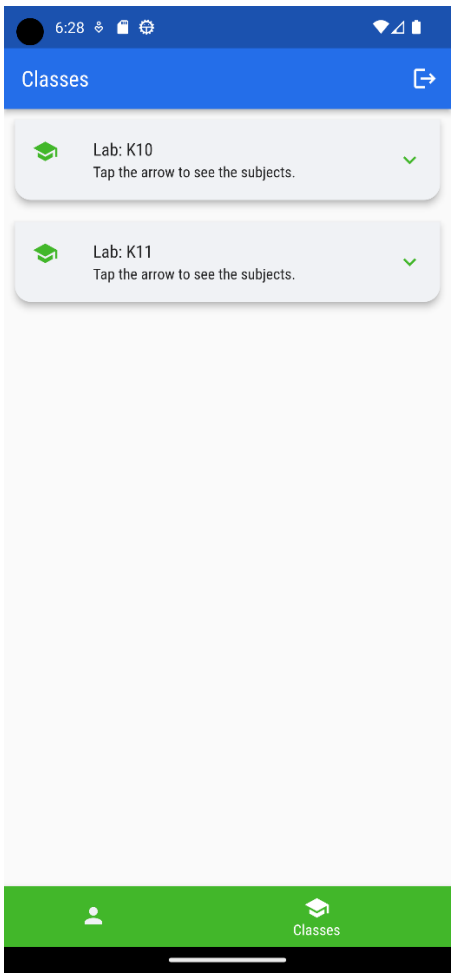
Σειρά έχει το κουμπί για την εμφάνιση του QR code του φοιτητή, που βρίσκεται στο κάτω μισό της αρχικής οθόνης.



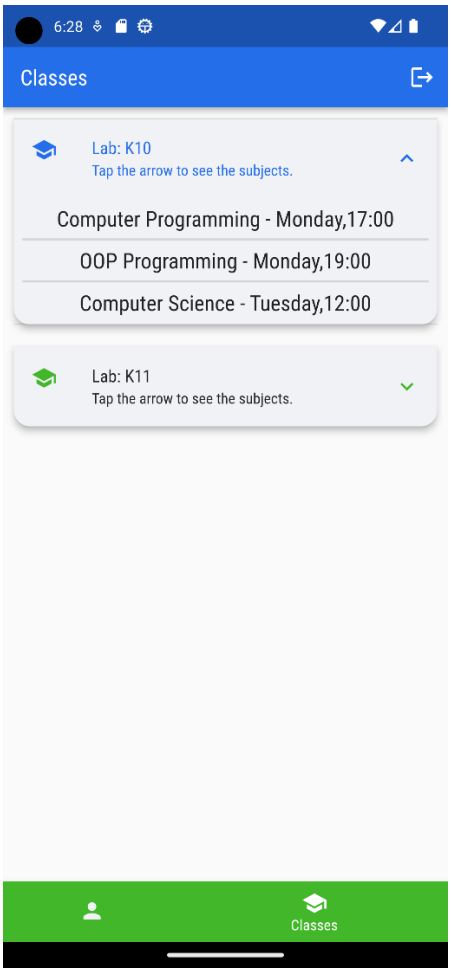
Εικόνα 10. Εμφάνιση QR code φοιτητή

Όταν ο φοιτητής πατήσει το κουμπί «Show QR code», τότε ένα αναδυόμενο παράθυρο εμφανίζεται με το QR code του αναγνωριστικού του. Αυτή η ενέργεια θα έχει την χρήση της κατά την διάρκεια της ταυτοποίησης. Ο καθηγητής θα αναγνώσει το QR code και το σύστημα θα επιβεβαιώσει εάν ο φοιτητής έχει την άδεια να βρίσκεται ή όχι στο εργαστήριο ή στην εξέταση.

Η δεύτερη επιλογή του φοιτητή είναι να πατήσει το δεύτερο κουμπί στην κάτω μπάρα περιήγησης. Όταν το κάνει αυτό, βλέπει όλα τα εργαστήρια με τα μαθήματα στα οποία έχει πρόσβαση.



Εικόνα 11. Εργαστήρια φοιτητή



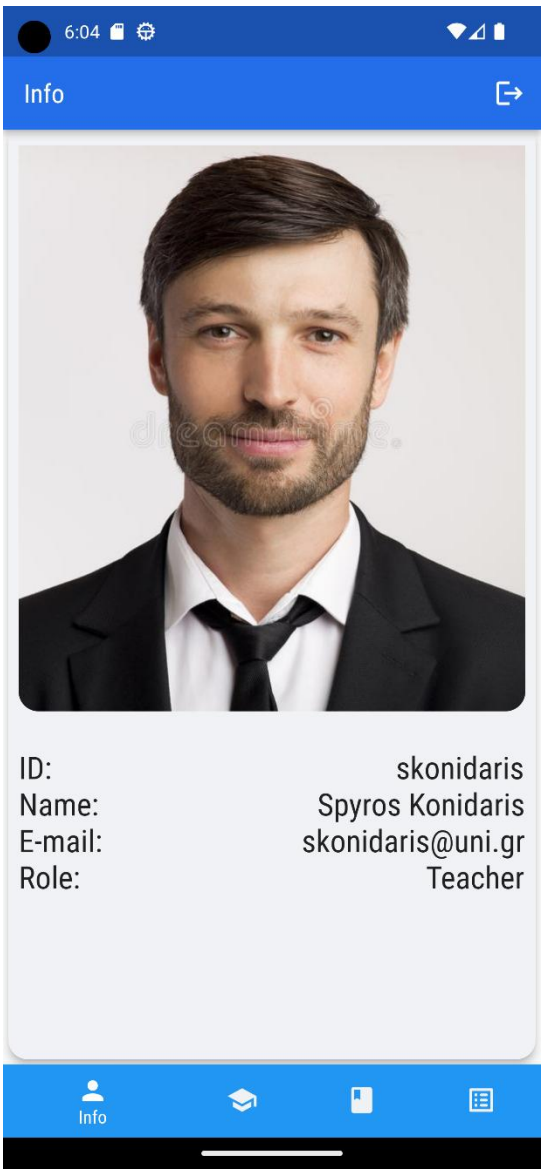
Εικόνα 12. Ανάπτυξη καρτέλας

Στις δύο εικόνες, εμφανίζονται οι κάρτες οι οποίες περιέχουν τα εργαστήρια με τα αναγνωριστικά τους (K10, K11 κ.ο.κ) καθώς και τα μαθήματα στα οποία έχει πρόσβαση - μαζί με την ημέρα και την ώρα που διεξάγονται - ο φοιτητής, εφόσον αναπτυχθεί η καρτέλα. Αυτή η οθόνη είναι χρήσιμη για τον φοιτητή καθώς μπορεί να ξέρει ανά πάσα στιγμή τα εργαστήρια του.

Σειρά έχει η διεπαφή που βλέπει ο εκάστοτε καθηγητής στην εφαρμογή.

4.2 Λειτουργίες Καθηγητή

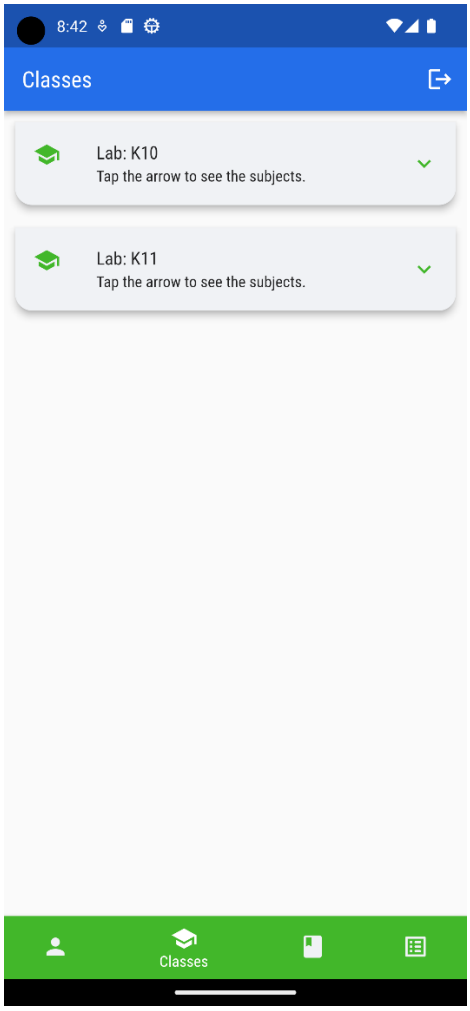
Όταν ο καθηγητής συνδεθεί στην εφαρμογή, μεταφέρεται στην αρχική οθόνη, η οποία είναι παρόμοια με αυτή του φοιτητή. Με την διαφορά όμως, ότι δεν υπάρχει κουμπί για την εμφάνιση του QR code. Κατά τα άλλα η υπόλοιπη διεπαφή είναι ίδια. Υπάρχει η άνω μπάρα, με τον τίτλο και το κουμπί της αποσύνδεσης καθώς και η κάτω μπάρα, η οποία όμως έχει παραπάνω κουμπιά διότι ο καθηγητής έχει περισσότερες δυνατότητες μέσα στην εφαρμογή.



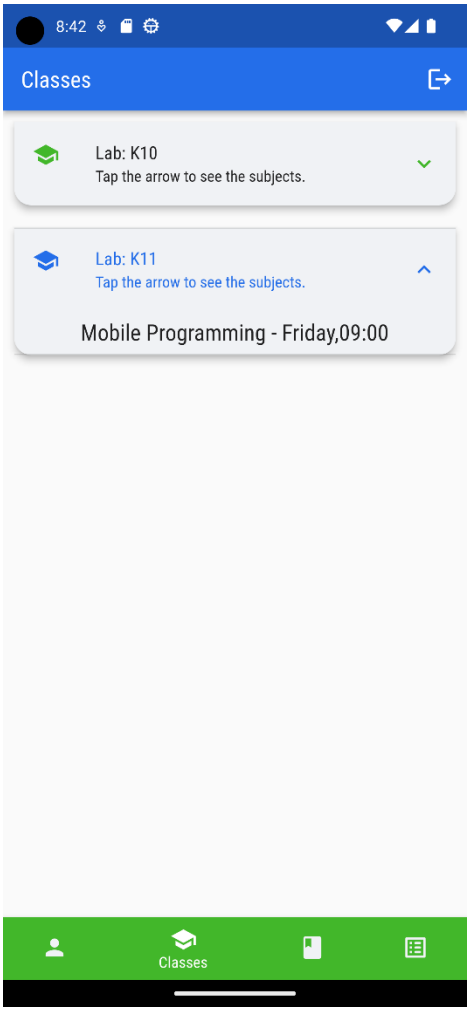
Εικόνα 13. Αρχική οθόνη καθηγητή

Από εδώ, ο καθηγητής έχει τις εξής δυνατότητες. Μπορεί, να δει –όπως και ο φοιτητής- τα εργαστήρια, μαζί με τα μαθήματα στα οποία διδάσκει, να ξεκινήσει μια συνεδρία ταυτοποίησης και να δει τις συνεδρίες τις οποίες έχει πραγματοποιήσει.

Η δεύτερη οθόνη είναι ακριβώς ίδια με του φοιτητή.

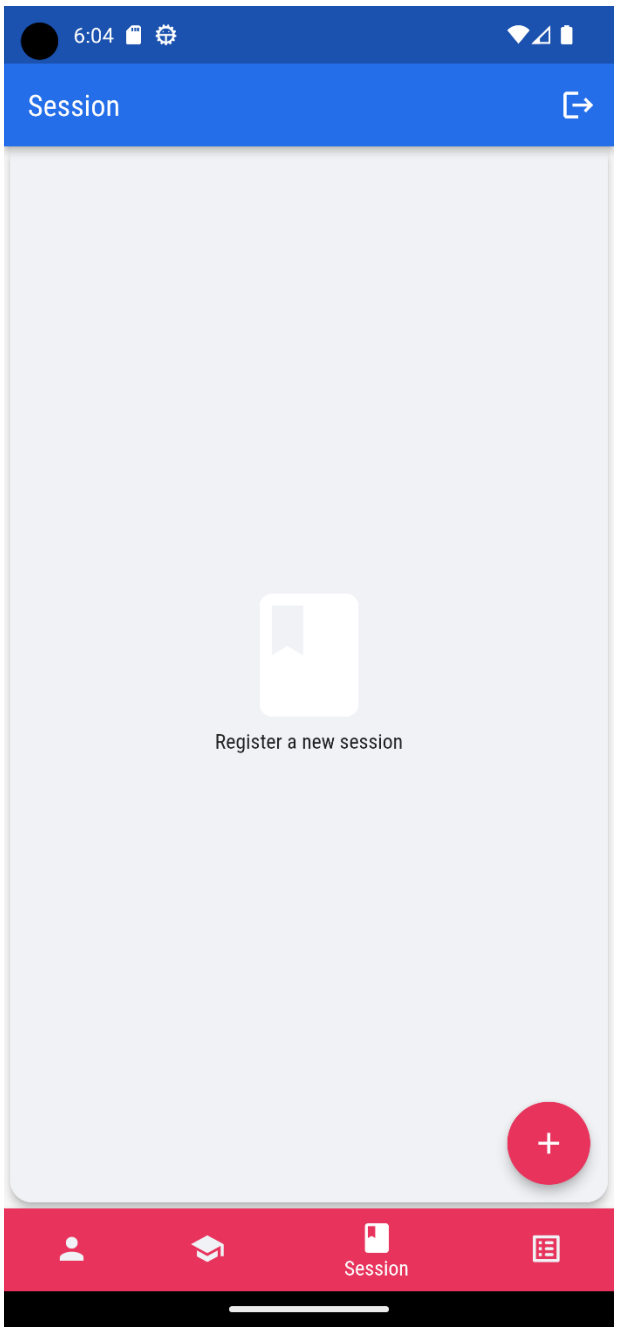


Εικόνα 14. Εργαστήρια καθηγητή



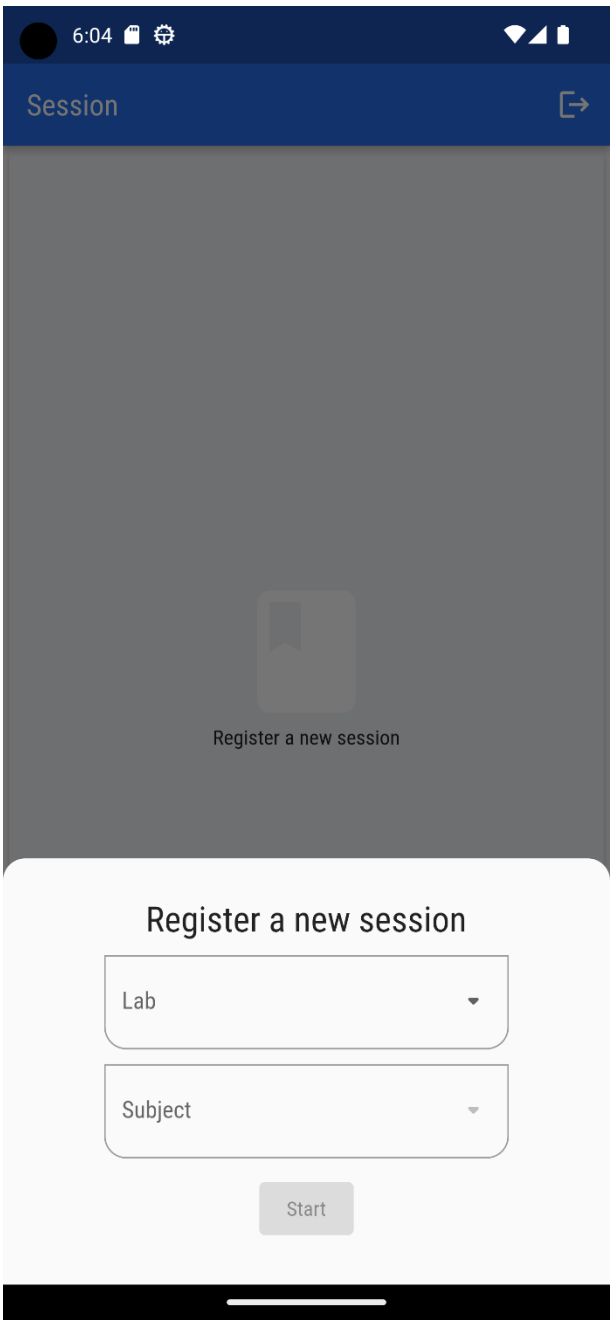
Εικόνα 15. Ανάπτυξη καρτέλας

Με την τρίτη οθόνη, ο καθηγητής μπορεί να ξεκινήσει μια συνεδρία ταυτοποίησης. Η διαδικασία έχει ως εξής. Ο καθηγητής αφού πατήσει στο τρίτο κουμπί, στην κάτω μπάρα περιήγησης, πρέπει να πατήσει το κουμπί με το «+» κάτω δεξιά.



Εικόνα 16. Οθόνη καινούριας συνεδρίας

Αφού ο καθηγητής πατήσει το κουμπί κάτω δεξιά στην οθόνη, ένα μενού επιλογών εμφανίζεται από το κάτω μέρος της οθόνης για να επιλέξει το εργαστήριο και το μάθημα για το οποίο θέλει να ξεκινήσει μια συνεδρία ταυτοποίησης. Τα εργαστήρια που εμφανίζονται, είναι μόνον αυτά στα οποία έχει πρόσβαση. Όταν επιλέξει το εργαστήριο, στην συνέχεια, φορτώνονται και τα κατάλληλα μαθήματα που διεξάγονται στο συγκεκριμένο εργαστήριο και στα οποία δάσκαλος, είναι ο ίδιος/ίδια. Παρακάτω, φαίνεται το μενού.

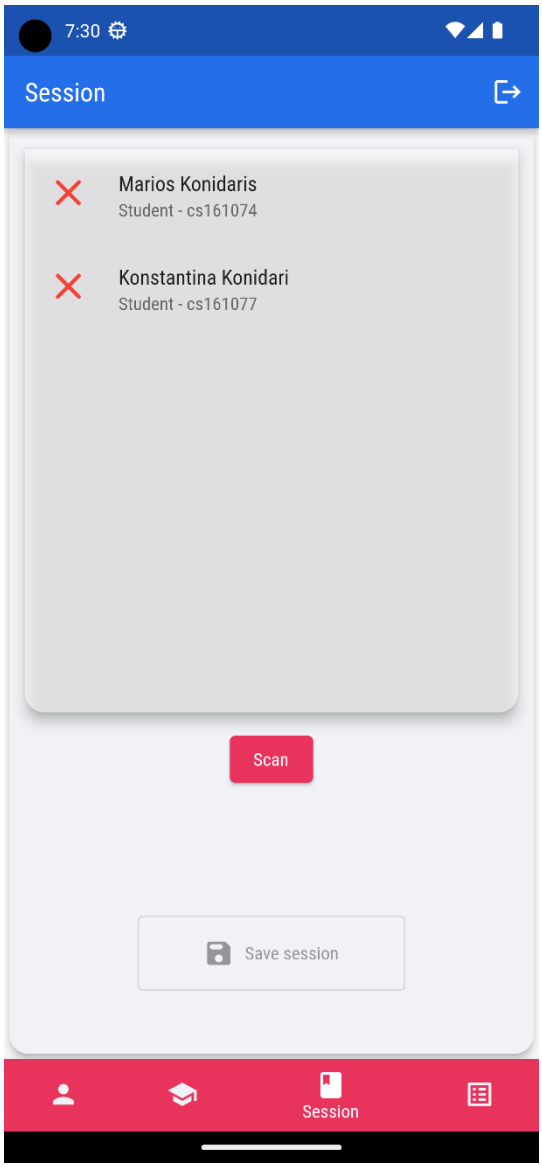


Εικόνα 17. Μενού επιλογών συνεδρίας

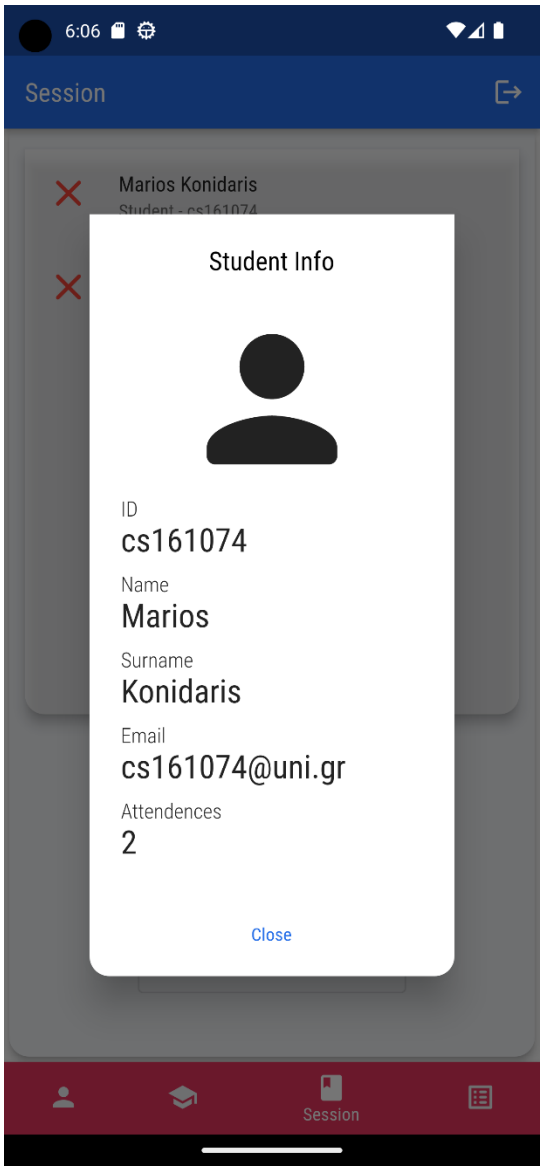
Μόνο, αφού επιλέξει και τις δύο επιλογές, θα ενεργοποιηθεί το κουμπί για την έναρξη της ταυτοποίησης.

Αφού ξεκινήσει η έναρξη, η εικόνα γεμίζει με έναν πίνακα στον οποίο περιέχονται όλοι οι φοιτητές, οι οποίοι παρακολουθούν το εργαστήριο, και εμφανίζονται με ένα εικονίδιο «X», το οποίο σημαίνει ότι δεν έχουν ταυτοποιηθεί ακόμα. Σε αυτή την κατάσταση, ο καθηγητής μπορεί να πατήσει πάνω σε κάποιον φοιτητή και να δει τις πληροφορίες του. Για να αναγνώσει το QR code ενός φοιτητή, υπάρχει ένα κουμπί «Scan», ώστε να ανοίξει η κάμερα και να αναγνώσει η εφαρμογή του καθηγητή, το QR code του

φοιτητή. Τέλος, στο κάτω μέρος, υπάρχει άλλο ένα κουμπί για να μπορέσει η συνεδρία να σωθεί στην βάση δεδομένων του συστήματος.



Εικόνα 18. Οθόνη ταυτοποίησης

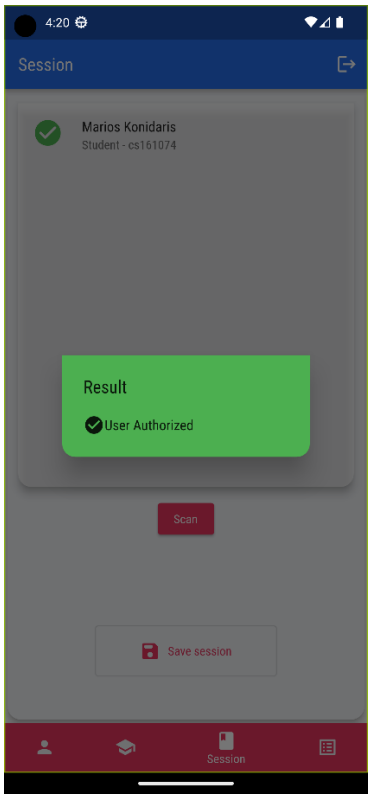


Εικόνα 19. Πληροφορίες φοιτητή στην ίδια οθόνη

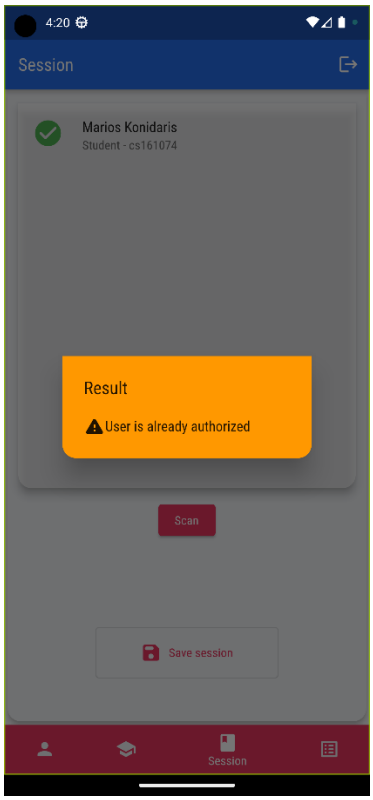
Χρήσιμο είναι να σημειωθεί, ότι ο καθηγητής δεν έχει πρόσβαση στις εικόνες των φοιτητών, αλλά μπορεί να τους δει μόνο κατά την επίδειξη της εφαρμογής από τους ίδιους και ξέρει ανά πάσα στιγμή τις μέχρι τώρα παρουσίες του φοιτητή στο συγκεκριμένο μάθημα.

Όταν πατήσει στο κουμπί «Scan», η κάμερα ανοίγει και η εφαρμογή είναι έτοιμη να αναγνώσει ένα QR code. Όταν η εφαρμογή, εντοπίσει ένα QR code, η τιμή (το αναγνωριστικό) του εξάγεται και συγκρίνεται με την λίστα των φοιτητών που

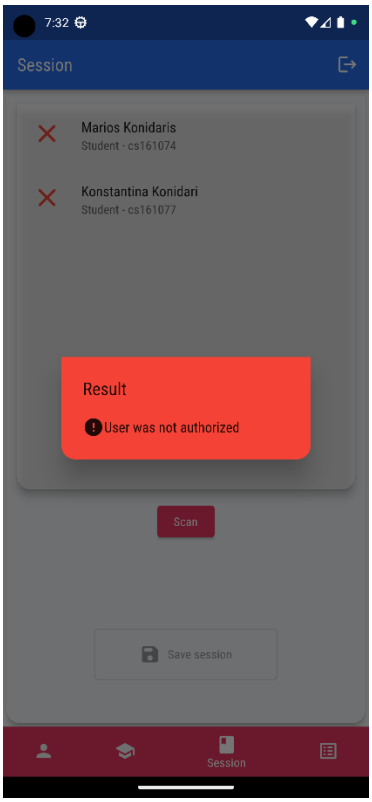
αναμένεται να προσέλθουν στο μάθημα (η λίστα που βλέπει ο καθηγητής στην οθόνη). Εφόσον υπάρχει στην λίστα και δεν έχει ταυτοποιηθεί, τότε ένα μήνυμα επιτυχίας εμφανίζεται, ο φοιτητής έχει ταυτοποιηθεί και συν μία παρουσία έχει προστεθεί. Επιπλέον, αντί για ένα «Χ», ένα τικ εμφανίζεται δίπλα από το όνομα του. Αν δεν υπάρχει το αναγνωριστικό του χρήστη στην λίστα, τότε ένα μήνυμα σφάλματος εμφανίζεται στην οθόνη, και ο χρήστης παραμένει ως έχει. Τέλος, εάν ο χρήστης για κάποιο λόγο, προσπαθήσει να ταυτοποιηθεί ξανά, τότε η εφαρμογή ενημερώνει τον καθηγητή, ότι ο χρήστης έχει ήδη ταυτοποιηθεί, συνεπώς καμία ενέργεια δεν λαμβάνει χώρα. Παρακάτω, παρατίθενται οι σχετικές εικόνες.



Εικόνα 20. Φοιτητής ταυτοποιήθηκε



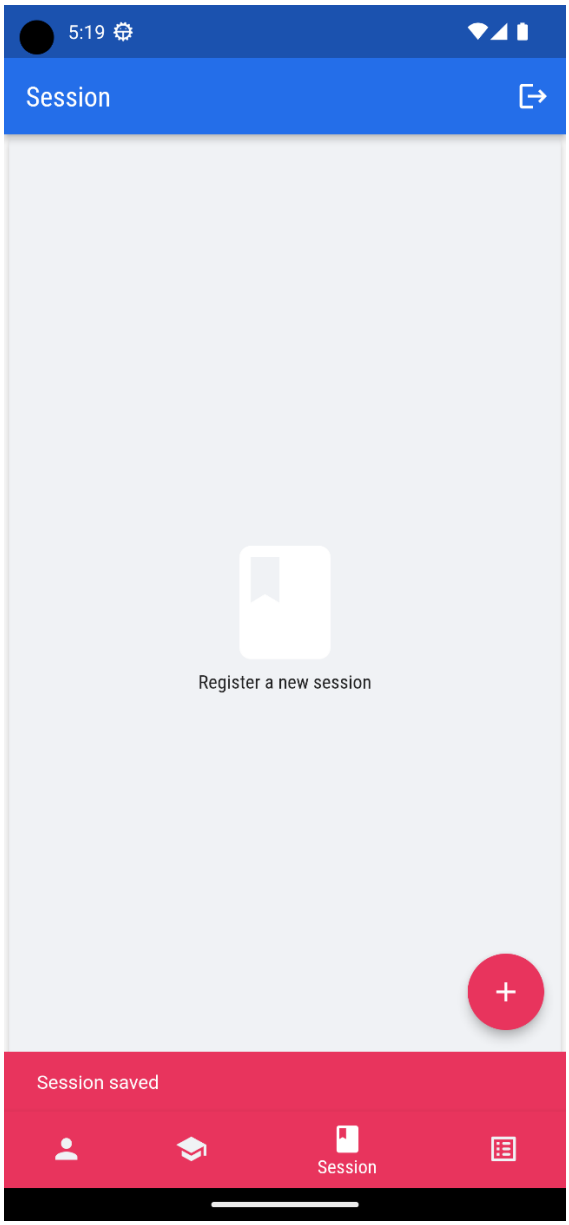
Εικόνα 21. Φοιτητής έχει ήδη ταυτοποιηθεί



Εικόνα 22. Φοιτητής δεν μπορεί να ταυτοποιηθεί

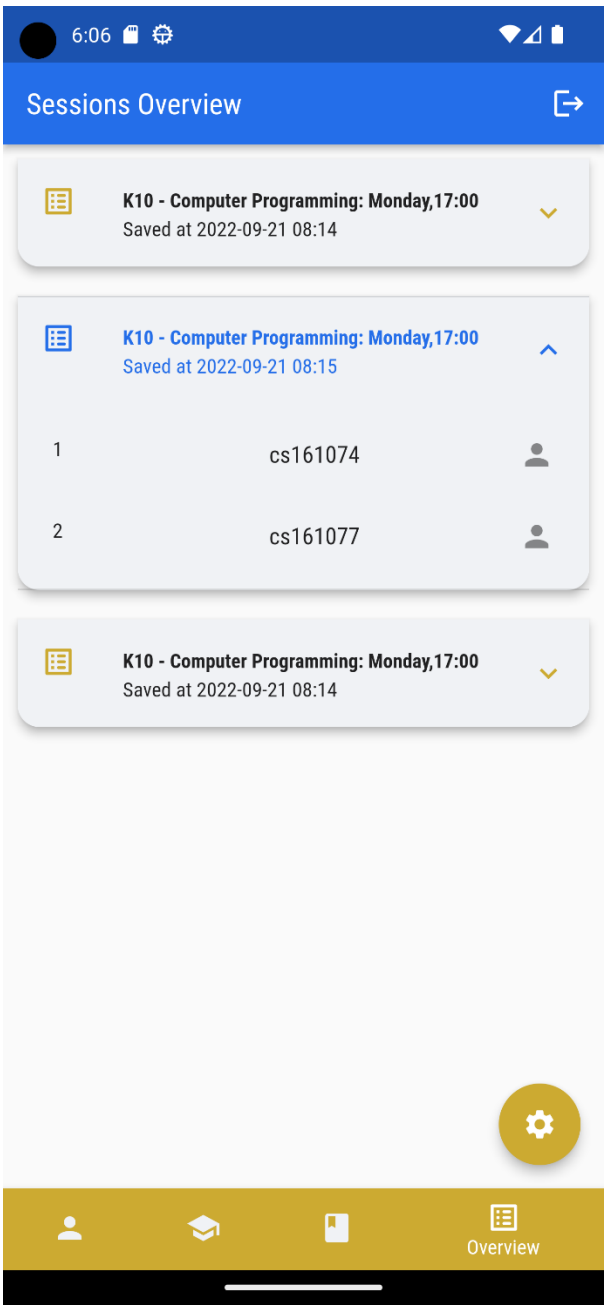
Παρατηρούμε ότι το κουμπί για την αποθήκευση της συνεδρίας δεν ενεργοποιείται μέχρι να έχει ταυτοποιηθεί τουλάχιστον ένας φοιτητής. Προφανώς αυτό συμβαίνει για να μην σπαταλάται τσάμπα χώρος στον διακομιστή από αχρείαστα έγγραφα. Όταν ο καθηγητής μπορεί και θέλει να τερματίσει την συνεδρία, το μόνο που έχει να κάνει, είναι να πατήσει το κουμπί «Save session». Εκείνη την ώρα, η εφαρμογή μαζεύει όλους τους φοιτητές που έχουν ταυτοποιηθεί με επιτυχία και μαζί με τις πληροφορίες

για το εργαστήριο τις αποθηκεύει στον διακομιστή. Εφόσον, αυτή η ενέργεια ολοκληρωθεί με επιτυχία, τότε ένα μήνυμα εμφανίζεται στο κάτω μέρος της οθόνης.



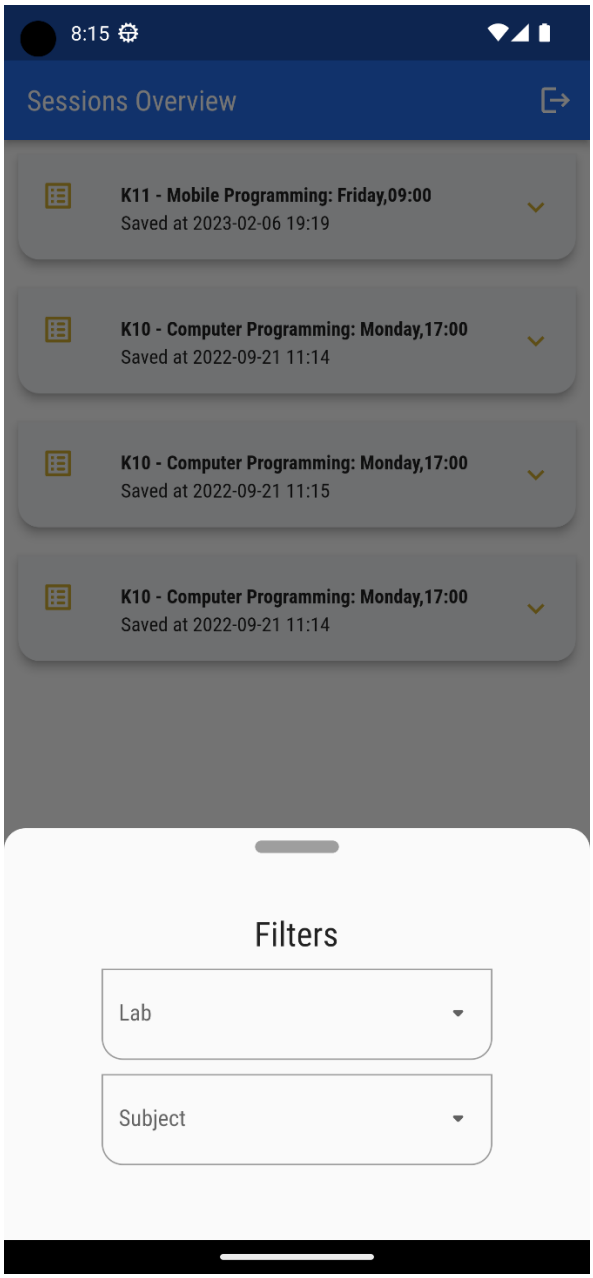
Εικόνα 23. Επιτυχής αποθήκευση συνεδρίας

Τέλος, σειρά έχει η οθόνη των συνεδριών. Ο καθηγητής μπορεί να δει όλες τις συνεδρίες τις οποίες έχει πραγματοποιήσει. Επιπλέον, για κάθε συνεδρία μπορεί να δει την ημερομηνία κατά την οποία την πραγματοποίησε, το εργαστήριο αλλά και το μάθημα στο οποίο έγινε και φυσικά όλους τους μαθητές που ήταν παρών. Παρακάτω, παρατίθενται η εικόνα.

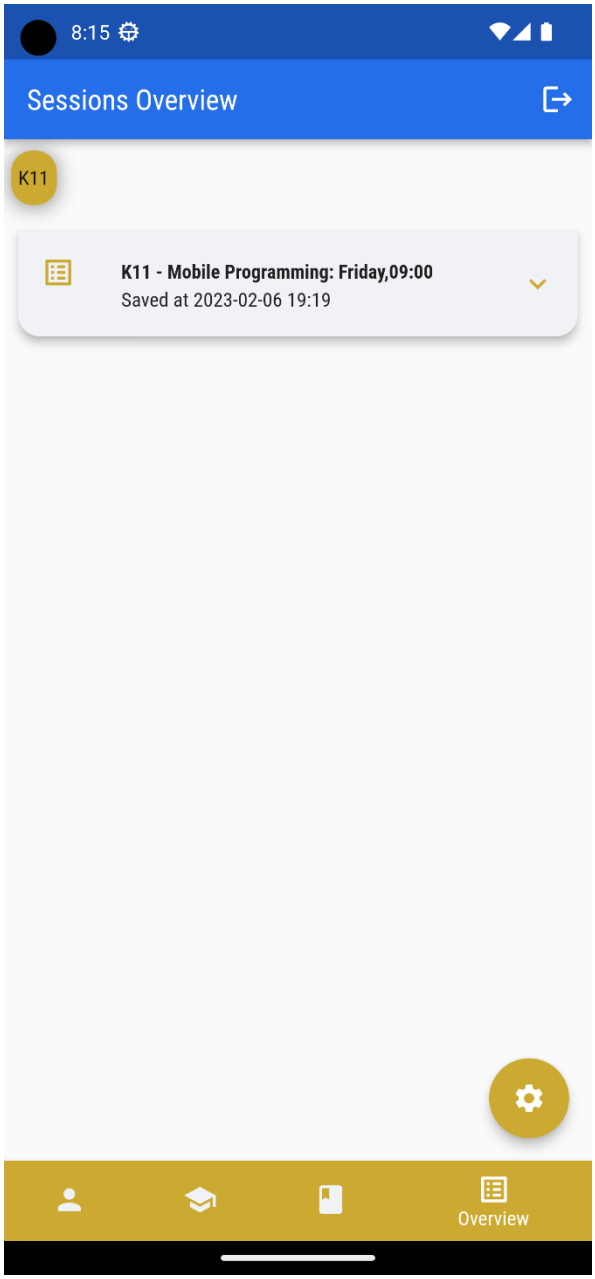


Εικόνα 24. Ανασκόπηση συνεδριών

Επίσης, με το κουμπί που υπάρχει κάτω δεξιά, ο καθηγητής μπορεί να φιλτράρει τις συνεδρίες που βλέπει στην οθόνη του. Όταν πατηθεί το κουμπί, ένα μενού εμφανίζεται στο κάτω μέρος της οθόνης (όπως στην οθόνη νέας συνεδρίας) στο οποίο μπορεί να εφαρμόσει μια σειρά από εργαστήρια και μαθήματα στα οποία διδάσκει. Όταν εφαρμοστεί ένα φίλτρο, κάθε ένα πηγαίνει στο πάνω μέρος στην οθόνη, σε μια οριζόντια λίστα. Ο καθηγητής μπορεί επίσης να πατήσει πάνω σε ένα από αυτά και να το αφαιρέσει εάν θέλει.



Εικόνα 25. Επιλογές φίλτρων



Εικόνα 26. Εφαρμογή φίλτρου

4.3 Προετοιμασία για χρήση σε πραγματικό περιβάλλον

Σε περίπτωση που η εφαρμογή που δημιουργήσαμε πρέπει να χρησιμοποιηθεί σε πραγματικές συνθήκες, τότε θα πρέπει να γίνει μια προετοιμασία, τόσο από το πανεπιστήμιο όσο και από την πλευρά μας. Αρχικά θα πρέπει να φτιαχτεί μια υπηρεσία η οποία θα αυτοματοποιεί την εισαγωγή των φοιτητών από την κεντρική βάση δεδομένων του τμήματος στην βάση δεδομένων της εφαρμογής στο Firebase. Δυστυχώς όμως, το τμήμα δεν παρέχει API ανοιχτό προς τρίτους, ώστε να είναι εφικτή αυτή η ενέργεια την δεδομένη στιγμή. Μέχρι τώρα οι φοιτητές εισάχθηκαν χειροκίνητα για να γίνει μια μικρή παρουσίαση των δυνατοτήτων της. Έπειτα, θα μπορούσε να χρησιμοποιηθεί η πύλη του τμήματος (χρησιμοποιείται κάθε φορά που θέλουμε να κάνουμε χρήση μια υπηρεσίας του τμήματος) για να γίνεται η σύνδεση στην εφαρμογή ώστε να αντικαταστήσει την μέχρι τώρα οθόνη σύνδεσης και έτσι να είναι ακόμα πιο ασφαλής η διαδικασία αυτή. Επιπλέον, θα μπορούσε να αντικατασταθεί το πακέτο provider για χάρη του πακέτου Bloc, το οποίο θεωρείται πιο προσιτό σε εφαρμογές που θέλουν να μεγαλώσουν σε όγκο. Τέλος, μικρές αλλαγές στον κώδικα ώστε να βελτιωθεί η απόδοση της εφαρμογής.

5. Επίλογος

Στο πλαίσιο αυτής της διπλωματικής εργασίας, εξετάστηκαν οι τεχνολογίες NFC και QR code, οι οποίες έχουν επηρεάσει σημαντικά την καθημερινότητά μας. Μέσα από την ανάλυση των αρχών λειτουργίας τους, των πλεονεκτημάτων και των εφαρμογών τους, κατανοήσαμε πώς μπορούν να απλοποιήσουν διαδικασίες, να εξοικονομήσουν χρόνο και να βελτιώσουν την αποτελεσματικότητα σε διάφορους τομείς.

Η εφαρμογή που αναπτύχθηκε στο Flutter, και η οποία βασίζεται στη χρήση QR code για την αυτοματοποίηση της διαδικασίας λήψης παρουσιών σε εργαστήρια και εξετάσεις μαθημάτων, αποτελεί πρακτικό παράδειγμα της δύναμης αυτών των τεχνολογιών. Με την εφαρμογή αυτή, έγινε εμφανές ότι οι τεχνολογίες αυτές μπορούν να προσφέρουν λύσεις που όχι μόνο απλοποιούν, αλλά και εκσυγχρονίζουν παραδοσιακές διαδικασίες, διευκολύνοντας τη ζωή μαθητών και καθηγητών.

Καταλήγοντας, αυτή η εργασία αναδεικνύει τη σημασία των τεχνολογιών αυτών, όχι μόνο ως απλών εργαλείων, αλλά και ως καινοτόμων λύσεων που μπορούν να οδηγήσουν σε έναν πιο ψηφιακό και αποδοτικό κόσμο. Η αξιοποίησή τους αναμένεται να γίνει ακόμα πιο εκτεταμένη στο μέλλον, προσφέροντας νέες δυνατότητες και εφαρμογές σε διάφορους τομείς της ζωής μας.

Βιβλιογραφία

al ofeishat, Hussein. 2012. «Near Field Communication (NFC).» *International Journal of Computer Science and Network Security*, Φεβρουάριος: 93-99.

Bhardwaj, Cheshtaa, Hitendra Garg, και Shashi Shekhar. 2022. «An Approach for Securing QR code using Cryptography and Visual Cryptography.» *International Conference on Computational Intelligence and Sustainable Engineering Solutions (CISES)*, 284-288.

C. -H., Li, Lao K. -W., και Tam K. -W. 2018. «A Flooding Warning System based on RFID Tag Array for Energy Facility.» *IEEE International Conference on RFID Technology & Application (RFID-TA)*, 1-4.

Google LCC. χ.χ. *Flutter.* Πρόσβαση Μάϊος 13, 2023. https://docs.flutter.dev/resources/faq.

Google LLC. χ.χ. *Firebase.* Πρόσβαση Μάϊος 13, 2023. https://firebase.google.com/docs/firestore.

Google LLC. 2023. *Flutter - Build apps for any screen.* 10 Μάιος. Πρόσβαση Μάιος 13, 2023. https://flutter.dev/.

Google LLC. χ.χ. *Flutter architectural overview.* Πρόσβαση Μάϊος 13, 2023. https://docs.flutter.dev/resources/architectural-overview.

Lyman, Isaac. 2022. *Stack Overflow.* 22 Οκτώβριος. Πρόσβαση Μάϊος 13, 2023. https://stackoverflow.blog/2022/10/31/comparing-frameworks-for-cross-platform-apps-flutter-vs-react-native/.

Macan, David. χ.χ. *Zero Molecule.* Πρόσβαση Μάϊος 14, 2023. https://www.zeromolecule.com/blog/host-card-emulation-hce-with-android-and-flutter/.

N. S. S., Shoba, Aruna K. S. P., Bhagyashree M. D. P. , και Sarita K. S. J. 2016. «NFC and NFC payments: A review.» *2016 International Conference on ICT in Business Industry & Government (ICTBIG)*, 1-7.

PANDEY, RAJESH. 2021. *iJunkie.* 26 Ιούλιος. Πρόσβαση Μάϊος 13, 2023. https://ijunkie.com/apple-nfc-android-less-secure-iphones/.

Steffen, R, J Preißinger, T Schöllermann, A Müller, και I Schnabel. 2010. «Near Field Communication (NFC) in an Automotive Environment.» *2010 Second International Workshop on Near Field Communication*, 15-20.

Tiwari, S. 2016. «An Introduction to QR Code Technology.» *2016 International Conference on Information Technology (ICIT)*, 39-44.

Wikipedia. 2014. *Wikipedia.* 11 Ιούνιος. Πρόσβαση Μάϊος 13, 2023. https://en.wikipedia.org/wiki/Firebase.

Παράρτημα

main.dart

```
// ignore_for_file: depend_on_referenced_packages, cast_nullable_to_non_nullable,
use_build_context_synchronously


import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_auth/firebase_auth.dart';
import 'package:firebase_core/firebase_core.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_uni_access/models/uni_user.dart';
import 'package:flutter_uni_access/providers/classes_provider.dart';
import 'package:flutter_uni_access/providers/session_overview_provider.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:flutter_uni_access/screens/auth_screen.dart';
import 'package:flutter_uni_access/screens/tabs_screen.dart';
import 'package:provider/provider.dart';

// Entrypoint of the app
void main() async {
  WidgetsFlutterBinding.ensureInitialized();

  // Sets the orientation of the app to portrait only
  SystemChrome.setPreferredOrientations(
    [DeviceOrientation.portraitUp, DeviceOrientation.portraitDown],
  );

  // Inits firebase
  await Firebase.initializeApp();

  // Run the app with MyApp as the first widget
  runApp(const MyApp());
}

// Father of all the widgets in the tree
```

```
class MyApp extends StatelessWidget {
  const MyApp({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    // Since we have multiple providers in the app
    // we use MultiProvider and init all of them
    // in the first component of the tree as to
    // pass the down
    return MultiProvider(
      providers: [
        ChangeNotifierProvider(
          create: (context) => SessionProvider(
            List.empty(growable: true),
            List.empty(growable: true),
            List.empty(growable: true),
          ),
        ),
        ChangeNotifierProvider(create: (context) => UserProvider()),
        ChangeNotifierProvider(
          create: (context) => SessionOverviewProvider(
            List.empty(growable: true),
            List.empty(growable: true),
            List.empty(growable: true),
            List.empty(growable: true),
            List.empty(growable: true),
          ),
        ),
        ChangeNotifierProvider(
          create: (context) => ClassesProvider(
            List.empty(growable: true),
          ),
        ),
      ],

      // Create the app
      child: MaterialApp(
        debugShowCheckedModeBanner: false,
```

```
title: 'UniAccess',
theme: ThemeData(
  colorScheme: ColorScheme.fromSwatch().copyWith(
    primary: const Color.fromRGBO(36, 110, 233, 1.0),
    secondary: const Color.fromRGBO(240, 242, 245, 1.0),
    surface: Colors.white,
  ),
  dialogBackgroundColor: Colors.white,
  fontFamily: 'RobotoCondensed',
),
home: StreamBuilder(
  // We subscibe to this stream in order
  // to keep track of the user's login status
  // every time it changes it will fire this
  // builder function
  stream: FirebaseAuth.instance.authStateChanges(),
  builder: (ctx, userSnapshot) {
    // If there are not data
    // redirect to the auth screen
    if (!userSnapshot.hasData) {
      return const AuthScreen();
    }

    // current user if any
    late final user = FirebaseAuth.instance.currentUser;

    // users collections back at firebase
    late final CollectionReference users =
        FirebaseFirestore.instance.collection('users');

    // id of the current user
    late String id;

    // If no user is present
    // redirect to auth screen
    if (user == null) {
      return const AuthScreen();
    }
```

```
// since id is included from the email
// and we have easy access to the user email
// from FirebaseAuth, take it from there
id = user.email!.split('@').elementAt(0);

// Find the user with the above id
// Set the image for the given user
// When all of this is done, procceed
return FutureBuilder<List<dynamic>>(
  future: Future.wait(
    [
      users.doc(id).get(),
      Provider.of<UserProvider>(ctx, listen: false)
          .setImageData(id),
    ],
  ),
  builder: (_, AsyncSnapshot<List<dynamic>> snapshot) {
    if (snapshot.hasError) {
      return Center(
        child: Text(snapshot.error.toString()),
      );
    }

    if (snapshot.hasData &&
        (snapshot.data == null || snapshot.data!.isEmpty)) {
      return const Center(
        child: Text('Document does not exist'),
      );
    }

    if (snapshot.connectionState == ConnectionState.done) {
      // Get the user from the snapshot data
      final uniUser = UniUser.fromFirestore(
        snapshot.data!.elementAt(0) as DocumentSnapshot<Object?>,
      );

      // Set their image
```

```
                uniUser.image = Provider.of<UserProvider>(ctx, listen: false)
                    .userImageData;

                // Store the user in the UserProvider
                // since we will need it down the line
                Provider.of<UserProvider>(ctx, listen: false).user = uniUser;

                // Proccedd to the TabsScreen
                // i.e the core app
                return TabsScreen();
              }

              return const Center(
                child: CircularProgressIndicator(),
              );
            },
          );
        },
      ),
    ),
  );
  }
}
```

models/attendances.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';

class Attendances {
  final String? subject;
  final Map<String, dynamic>? attendants;

  Attendances({required this.subject, required this.attendants});

  factory Attendances.fromFirestore(
    DocumentSnapshot<Object?> snapshot,
  ) {
    return Attendances(
      subject: snapshot.get('subject') as String,
      attendants: snapshot.get('attendants') as Map<String, dynamic>,
```

```
    );
  }
}
```

models/labs.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';

class Labs {
  final String? name;
  final List<dynamic>? access;

  Labs({required this.name, required this.access});

  factory Labs.fromFirestore(DocumentSnapshot<Object?> snapshot) {
    return Labs(
      name: snapshot.get('name') as String,
      access: snapshot.get('access') as List<dynamic>,
    );
  }
}
```

models/session.dart

```
// ignore_for_file: avoid_dynamic_calls

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:intl/intl.dart';

// Defines a session
class Session {
  final String? lab;
  final List<dynamic>? studentsIds;
  final String? subject;
  final String? teacher;
  final String? timestamp;

  Session({
    required this.lab,
```

```
    required this.studentsIds,
    required this.subject,
    required this.teacher,
    required this.timestamp,
  });

  factory Session.fromFirestore(
    QueryDocumentSnapshot<Object?> snapshot,
  ) {
    return Session(
      lab: snapshot.get('lab') as String,
      studentsIds: snapshot.get('students') as List<dynamic>,
      subject: snapshot.get('subject') as String,
      teacher: snapshot.get('teacher') as String,
      timestamp: DateFormat('yyy-MM-dd kk:mm').format(
        DateTime.fromMillisecondsSinceEpoch(
          snapshot.get('timestamp').millisecondsSinceEpoch as int,
          isUtc: true,
        ).toLocal(),
      ),
    );
  }
}
```

models/student_classes_card.dart

```
import 'package:flutter/foundation.dart';

// Defines the structure of the card to appear in the classes screen
class StudentClassesCard {
  final String? lab;
  final List<Map<String, dynamic>>? subjects;

  StudentClassesCard({@required this.lab, @required this.subjects});
}
```

models/uni_user.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
```

```
import 'package:flutter/foundation.dart';

// Defines a User (student or teacher)
class UniUser {
  final String? id;
  final String? name;
  final String? surname;
  final String? email;
  final bool? isTeacher;
  late Uint8List? image;
  bool isAuthorized = false;
  int attendaces = 0;

  UniUser({
    required this.id,
    required this.name,
    required this.surname,
    required this.email,
    required this.isTeacher,
  });

  factory UniUser.fromFirestore(DocumentSnapshot<Object?> snapshot) {
    return UniUser(
      id: snapshot.get('id') as String,
      name: snapshot.get('name') as String,
      surname: snapshot.get('surname') as String,
      email: snapshot.get('email') as String,
      isTeacher: snapshot.get('isTeacher') as bool,
    );
  }
}
```

providers/classes_provider.dart

```
// Defines a class provider
// ignore_for_file: avoid_dynamic_calls

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/student_classes_card.dart';

class ClassesProvider with ChangeNotifier {
  final List<StudentClassesCard>? studentClasses;

  ClassesProvider(this.studentClasses);

  void prepareClassesData(
    AsyncSnapshot<QuerySnapshot<Object?>> snapshot,
    String userId,
  ) {
    for (final element in snapshot.data!.docs) {
      final subjectsOfLab = List<Map<String, dynamic>>.empty(growable: true);
      for (final access in element['access'] as List) {
        final accessData = access as Map<String, dynamic>?;
        final userIdsList = accessData!['users'] as List<dynamic>;
        if (userIdsList.contains(userId)) {
          subjectsOfLab.add(
            {
              access!['info']['subjectName'] as String:
                  access['info']['date'] as String,
            },
          );
        }
      }
      if (subjectsOfLab.isNotEmpty) {
        final studentClassCard = StudentClassesCard(
          lab: element['name'] as String,
          subjects: subjectsOfLab,
        );
        studentClasses?.add(studentClassCard);
```

```
      }
    }
  }
}
```

providers/session_overview_provider.dart

```
import 'package:flutter/cupertino.dart';
import 'package:flutter_uni_access/models/session.dart';

// Defines a session overview provider
class SessionOverviewProvider with ChangeNotifier {
  List<Session>? allSessions;
  List<Session>? sessions;
  List<String>? labs;
  List<String>? subjects;
  List<String>? filters;

  SessionOverviewProvider(
    this.allSessions,
    this.labs,
    this.subjects,
    this.sessions,
    this.filters,
  );

  void populateLists() {
    for (final session in sessions!) {
      final lab = session.lab;
      if (!labs!.contains(lab)) {
        labs?.add(lab!);
      }

      final subject = session.subject;
      if (!subjects!.contains(subject)) {
        subjects?.add(subject!);
      }
    }
  }
```

```
  void filter() {
    if (filters!.isEmpty) {
      sessions = allSessions;
      notifyListeners();
      return;
    }

    if (sessions!.isEmpty) {
      sessions = allSessions;
    }

    var filteredSessions = List<Session>.empty();
    filteredSessions = sessions!
        .where(
          (element) =>
              element.subject == filters?.last || element.lab == filters?.last,
        )
        .toList();

    sessions = filteredSessions;
    notifyListeners();
  }
}
```

providers/session_provider.dart

```
// ignore_for_file: avoid_dynamic_calls, use_build_context_synchronously

import 'dart:developer';

import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/attendances.dart';
import 'package:flutter_uni_access/models/labs.dart';
import 'package:flutter_uni_access/models/uni_user.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:provider/provider.dart';

// Defines a session provider
```

```
class SessionProvider with ChangeNotifier {
  final List<UniUser>? sessionUsers;

  final List<String>? _labs;
  List<String>? _subjects;

  String? _selectedLab;
  String? _selectedSubject;

  // 1 = authorized,
  // 2 = not authorized,
  // 3 = already authorized
  int authResult = 0;

  bool _startedScanning = false;
  bool abortSessionFromTabChange = false;
  bool canSaveSession = false;
  int? currentUserAttendanceInSessionOverview;

  SessionProvider(
    this.sessionUsers,
    this._labs,
    this._subjects,
  );

  List<String> get labs {
    return [..._labs!];
  }

  List<String> get subjects => [...?_subjects];

  set selectedLab(String? lab) {
    _selectedLab = lab;
    notifyListeners();
  }

  set selectedSubject(String? subject) {
    _selectedSubject = subject;
```

```
  notifyListeners();
}

set startedScanning(bool sScanning) {
  _startedScanning = sScanning;
  notifyListeners();
}

String? get selectedLab => _selectedLab;

String? get selectedSubject => _selectedSubject;

bool get startedScanning => _startedScanning;

Future<void> populateFormData(int sw, String id, BuildContext context) async {
  final CollectionReference labs =
      FirebaseFirestore.instance.collection('labs');

  switch (sw) {
    // load labs
    case 1:
      await labs.get().then(
        (value) {
          for (final element in value.docs) {
            final lab = Labs.fromFirestore(element);
            if (lab.access == null) {
              return;
            }

            for (final access in lab.access!) {
              final accessData = access as Map<String, dynamic>?;

              if (!accessData!['users'].toString().contains(id)) {
                continue;
              }

              final labName = lab.name!;
              if (!_labs!.contains(labName)) {
```

```
                _labs?.add(labName);
              }
            }
          }
        },
        onError: (e) {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: const Text('Could not fetch labs'),
              backgroundColor: Theme.of(context).colorScheme.error,
            ),
          );
        },
      );

      break;
    // load subjects
    case 2:
      final subjects = List<String>.empty(growable: true);
      await labs.doc(_selectedLab).get().then(
        (value) {
          final lab = Labs.fromFirestore(value);
          if (lab.access == null) {
            return;
          }
          for (final access in lab.access!) {
            final accessData = access as Map<String, dynamic>?;
            if (!accessData!['users'].toString().contains(id)) {
              continue;
            }
            subjects.add(
              '${accessData['info']['subjectName']}:
${accessData['info']['date']}',
            );
          }
        },
        onError: (e) {
          ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
```

```
                content: const Text('Could not fetch subjects'),
                backgroundColor: Theme.of(context).colorScheme.error,
              ),
            );
          },
        );

        _subjects = subjects;
        notifyListeners();

        break;
    }
  }

  Future<void> findAllPermittedStudents() async {
    final CollectionReference labs =
        FirebaseFirestore.instance.collection('labs');
    final CollectionReference users =
        FirebaseFirestore.instance.collection('users');

    await labs.doc(_selectedLab).get().then((value) async {
      final lab = Labs.fromFirestore(value);

      for (final access in lab.access!) {
        final accessData = access as Map<String, dynamic>?;
        if ('${accessData?['info']['subjectName']}: ${accessData?['info']['date']}'
==
            _selectedSubject) {
          final students = (accessData!['users'] as List)
              .map((student) => student as String)
              .toList();
          for (final student in students) {
            if (!student.startsWith('cs')) {
              continue;
            }

            await users.doc(student).get().then((value) {
              final uniUser = UniUser.fromFirestore(value);
              uniUser.isAuthorized = false;
```

```
            sessionUsers?.add(uniUser);
          });
        }
      }
    }
  });

  notifyListeners();
}

void authorizeUser(String? id) {
  log('${sessionUsers?.map((e) => e.id)}');
  if (id == null) {
    authResult = 2;
    notifyListeners();

    return;
  }

  if (sessionUsers!.every((user) => user.id != id)) {
    authResult = 2;
    notifyListeners();

    return;
  }

  if (sessionUsers!.firstWhere((user) => user.id == id).isAuthorized) {
    authResult = 3;
    notifyListeners();

    return;
  }

  authResult = 1;
  sessionUsers!.firstWhere((user) => user.id == id).isAuthorized = true;
  if (!canSaveSession) {
    canSaveSession = true;
  }
```

```
  notifyListeners();
}

Future<void> addMissingAttendanceDocument(
  BuildContext context,
) async {
  for (final user in sessionUsers!) {
    user.isAuthorized ? user.attendaces = 1 : user.attendaces = 0;
  }

  final attendance = <String, dynamic>{
    'subject': _selectedSubject,
    'attendants': {
      for (final user in sessionUsers!) user.id: user.attendaces,
    },
  };

  await FirebaseFirestore.instance
      .collection('attendances')
      .doc(_selectedSubject)
      .set(attendance)
      .onError(
        (error, stackTrace) => ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: const Text('Could not initiate attendance'),
            backgroundColor: Theme.of(context).colorScheme.error,
          ),
        ),
      );
}

Future<void> updateUserAttendance(BuildContext context) async {
  await FirebaseFirestore.instance
      .collection('attendances')
      .doc(_selectedSubject)
      .get()
      .then(
    (value) async {
```

```
      if (!value.exists) {
        await addMissingAttendanceDocument(context);
        return;
      }

      final attendance = Attendances.fromFirestore(value);

      attendance.attendants?.forEach((key, value) {
        for (final user in sessionUsers!) {
          if (user.id == key && user.isAuthorized) {
            user.attendaces = (value as int) + 1;
          } else if (user.id == key) {
            user.attendaces = value as int;
          }
        }
      });
    },
    onError: (e) => ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: const Text('Could not fetch attendance data'),
        backgroundColor: Theme.of(context).colorScheme.error,
      ),
    ),
  );
}

Future<void> findStudentAttendances(String id, BuildContext context) async {
  final CollectionReference attendances =
      FirebaseFirestore.instance.collection('attendances');

  await attendances.doc(_selectedSubject).get().then(
    (value) {
      if (!value.exists) {
        currentUserAttendanceInSessionOverview = null;
        return;
      }
      final attendances = Attendances.fromFirestore(value);
      attendances.attendants?.forEach((key, value) {
```

```
        if (key == id) {
          currentUserAttendanceInSessionOverview = value as int;
        }
      });
    },
    onError: (_) => ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: const Text('Could not fetch student attendance data'),
        backgroundColor: Theme.of(context).colorScheme.error,
      ),
    ),
  );
}

Future<void> updateAttendanceCollection(BuildContext context) async {
  final currentAttendances = FirebaseFirestore.instance
      .collection('attendances')
      .doc(_selectedSubject);

  currentAttendances.update(
    {
      'attendants': {
        for (final user in sessionUsers!) user.id: user.attendaces,
      },
    },
  );
}

Future<void> saveSession(BuildContext context) async {
  await updateUserAttendance(context);

  await updateAttendanceCollection(context);

  final students = List<String>.empty(growable: true);
  for (final user in sessionUsers!) {
    if (user.isAuthorized) students.add(user.id!);
  }
```

```
  final session = <String, dynamic>{
    'lab': _selectedLab,
    'subject': _selectedSubject,
    'students': students,
    'timestamp': DateTime.now(),
    'teacher': Provider.of<UserProvider>(context, listen: false).user?.id,
  };

  await FirebaseFirestore.instance
      .collection('sessions')
      .doc()
      .set(session)
      .onError((error, stackTrace) {
    ScaffoldMessenger.of(context).showSnackBar(
      SnackBar(
        content: const Text('Could not save session'),
        backgroundColor: Theme.of(context).colorScheme.error,
      ),
    );
  });

  ScaffoldMessenger.of(context).showSnackBar(
    const SnackBar(
      content: Text('Session saved'),
      backgroundColor: Color.fromRGBO(
        232,
        52,
        93,
        1.0,
      ),
    ),
  );

  stopSession();
}

void stopSession() {
  _selectedLab = null;
```

```
    _selectedSubject = null;
    sessionUsers?.clear();
    startedScanning = false;
  }
}
```

providers/user_provider.dart

```
import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:firebase_storage/firebase_storage.dart';
import 'package:flutter/foundation.dart';
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/uni_user.dart';

// Defines a user provider
class UserProvider with ChangeNotifier {
  UniUser? user;
  Uint8List? userImageData;

  Future<UniUser> getUserFromId(String id, BuildContext context) async {
    final CollectionReference users =
        FirebaseFirestore.instance.collection('users');
    late final UniUser uniUser;

    await users.doc(id).get().then(
          (value) => uniUser = UniUser.fromFirestore(value),
          onError: (e) => ScaffoldMessenger.of(context).showSnackBar(
            SnackBar(
              content: const Text('Could not fetch user'),
              backgroundColor: Theme.of(context).colorScheme.error,
            ),
          ),
        );

    return uniUser;
  }

  Future<void> setImageData(String id) async {
    try {
      const size = 800 * 600;
```

```
      userImageData =
          await FirebaseStorage.instance.ref().child('$id.jpg').getData(size);
    } on FirebaseException catch (_) {}
  }
}
```

screens/auth_screen.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter/services.dart';
import 'package:flutter_uni_access/widgets/auth_form_widget.dart';

class AuthScreen extends StatefulWidget {
  const AuthScreen({Key? key}) : super(key: key);

  @override
  State<AuthScreen> createState() => _AuthScreenState();
}

class _AuthScreenState extends State<AuthScreen> {
  late final _auth = FirebaseAuth.instance;
  var _isLoading = false;

  Future<void> _submitAuthForm(
    String email,
    String password,
    BuildContext ctx,
  ) async {
    try {
      setState(() {
        _isLoading = true;
      });
```

```
    await _auth.signInWithEmailAndPassword(email: email, password: password);
  } on PlatformException catch (err) {
    var message = 'An error occurred, please check your credentials';

    if (err.message != null) {
      message = err.message!;
    }

    ScaffoldMessenger.of(ctx).showSnackBar(
      SnackBar(
        content: Text(message),
        backgroundColor: Theme.of(ctx).colorScheme.error,
      ),
    );

    setState(() {
      _isLoading = false;
    });
  } on FirebaseAuthException catch (err) {
    var message = 'Wrong credentials';

    if (err.message != null) {
      message = err.message!;
    }

    ScaffoldMessenger.of(ctx).showSnackBar(
      SnackBar(
        content: Text(message),
        backgroundColor: Theme.of(ctx).colorScheme.error,
      ),
```

```
    );


    setState(() {
      _isLoading = false;
    });
  } catch (err) {
    setState(() {
      _isLoading = false;
    });
  }
}


@override
Widget build(BuildContext context) {
  return GestureDetector(
    onTap: () => FocusManager.instance.primaryFocus?.unfocus(),
    child: Scaffold(
      backgroundColor: Theme.of(context).colorScheme.surface,
      body: Column(
        mainAxisAlignment: MainAxisAlignment.spaceEvenly,
        children: [
          // Show logo when keyboard is closed
          if (MediaQuery.of(context).viewInsets.bottom == 0)
            Image(
              image: const AssetImage('assets/images/pada-logo.png'),
              height: MediaQuery.of(context).size.height / 4,
            ),
          const Text(
            'Welcome to UniAccess',
            style: TextStyle(fontWeight: FontWeight.bold, fontSize: 30),
          ),
```

```
            AuthForm(_submitAuthForm, _isLoading),
          ],
        ),
      ),
    );
  }
}
```

screens/new_session_screen.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/widgets/attendence_widget.dart';
import 'package:provider/provider.dart';

class NewSessionScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Whether the user can save the session
    final canSave = Provider.of<SessionProvider>(context).canSaveSession;

    return Card(
      color: Theme.of(context).colorScheme.secondary,
      elevation: 5,
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      child: Column(
        children: [
          if (!Provider.of<SessionProvider>(context).startedScanning)
```

```
  SizedBox(
    height: MediaQuery.of(context).size.height / 3,
  ),
if (Provider.of<SessionProvider>(context).startedScanning)
  Column(
    children: [
      const AttendanceWidget(),
      SizedBox(
        height: MediaQuery.of(context).size.height / 9,
      ),
      Tooltip(
        message: 'Save session',
        child: OutlinedButton.icon(
          style: OutlinedButton.styleFrom(
            foregroundColor: canSave
                ? const Color.fromRGBO(232, 52, 93, 1.0)
                : Theme.of(context).colorScheme.surface,
            fixedSize: Size(
              MediaQuery.of(context).size.width / 2,
              MediaQuery.of(context).size.height / 15,
            ),
          ),
          onPressed: canSave
              ? () => Provider.of<SessionProvider>(
                    context,
                    listen: false,
                  ).saveSession(context)
              : null,
          icon: const Icon(Icons.save_rounded),
          label: const Text('Save session'),
        ),
```

```
            ),
          ],
        )
      else
        const Column(
          children: [
            Center(
              child: Column(
                children: [
                  Icon(
                    Icons.class_rounded,
                    size: 100.0,
                    color: Colors.white,
                  ),
                  Text('Register a new session'),
                ],
              ),
            ),
          ],
        ),
      ],
    ),
  );
 }
}
```

screens/session_overview_screen.dart

```
// ignore_for_file: avoid_dynamic_calls


import 'package:cloud_firestore/cloud_firestore.dart';

import 'package:flutter/material.dart';

import 'package:flutter_uni_access/models/session.dart';

import 'package:flutter_uni_access/providers/session_overview_provider.dart';

import 'package:flutter_uni_access/providers/user_provider.dart';

import 'package:flutter_uni_access/widgets/session_overview_widget.dart';

import 'package:provider/provider.dart';


class SessionOverviewScreen extends StatelessWidget {

  SessionOverviewScreen({Key? key}) : super(key: key);


  // Get a reference to the sessions collection as a stream

  late final Stream<QuerySnapshot> _sessionOverviewStream =

      FirebaseFirestore.instance.collection('sessions').snapshots();


  @override

  Widget build(BuildContext context) {

    return StreamBuilder(

      stream: _sessionOverviewStream,

      builder: (BuildContext context, AsyncSnapshot<QuerySnapshot> snapshot) {

        if (snapshot.hasError) {

          return Center(child: Text(snapshot.error.toString()));

        }


        if (snapshot.connectionState == ConnectionState.waiting) {

          return const Center(child: CircularProgressIndicator());

        }
```

```
if (snapshot.data?.docs == null) {
  return const Center(child: Text('Data is null.'));
}

final sessions = List<Session>.empty(growable: true);

for (final element in snapshot.data!.docs) {
  final session = Session.fromFirestore(element);
  // If the session's teacher is not the current user, ingore it
  if (session.teacher?.compareTo(
        Provider.of<UserProvider>(context, listen: false).user!.id!,
      ) !=
      0) {
    continue;
  }

  final studentsIds = List<String>.empty(growable: true);
  for (final id in session.studentsIds!) {
    studentsIds.add(id as String);
  }

  sessions.add(session);
}

Provider.of<SessionOverviewProvider>(context, listen: false).sessions =
    sessions;
Provider.of<SessionOverviewProvider>(context, listen: false)
    .allSessions = sessions;
Provider.of<SessionOverviewProvider>(context, listen: false)
    .populateLists();
```

```
        // TODO: Maybe more efficient?
        return const SessionOverviewsWidget();
      },
    );
  }
}
```

screens/tabs_screen.dart

```
import 'package:firebase_auth/firebase_auth.dart';
import 'package:flutter/material.dart';
import 'package:flutter/rendering.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:flutter_uni_access/screens/new_session_screen.dart';
import 'package:flutter_uni_access/screens/session_overview_screen.dart';
import 'package:flutter_uni_access/screens/user_classes_screen.dart';
import 'package:flutter_uni_access/screens/user_info_screen.dart';
import 'package:flutter_uni_access/widgets/dialog_widget.dart';
import
'package:flutter_uni_access/widgets/filters_session_overview_dialog_widget.dart';
import 'package:flutter_uni_access/widgets/new_session_dialog_widget.dart';
import 'package:provider/provider.dart';

class TabsScreen extends StatefulWidget {
  @override
  State<TabsScreen> createState() => _TabsScreenState();
}

class _TabsScreenState extends State<TabsScreen> {
  // all the possible pages / screens of the app
  late final List<Map<String, Object>> _pages;
```

```
// current page shown
int _selectedIndex = 0;
// whether to show or not the floating action button
bool _showFloatingActionButton = true;

// When the widget (TabsScreen) is added to the tree
// it will fire this method.
@override
void initState() {
  super.initState();

  // take the current user in our UserProvider
  final user = Provider.of<UserProvider>(context, listen: false).user!;

  // Load seperate pages for each type of user
  if (!user.isTeacher!) {
    _pages = [
      {'page': UsertInfoScreen(), 'title': 'Info'},
      {'page': UserClassesScreen(), 'title': 'Classes'},
    ];
  } else {
    _pages = [
      {'page': UsertInfoScreen(), 'title': 'Info'},
      {'page': UserClassesScreen(), 'title': 'Classes'},
      {'page': NewSessionScreen(), 'title': 'Session'},
      {'page': SessionOverviewScreen(), 'title': 'Sessions Overview'},
    ];
  }
}

// This method is being called
```

```
// when the user changes screen
// taps at tab button.
// it uses setState which
// rebuilds the whole screen
// and as such loads the next screen.
// It takes the the index of the screen
// in which the user wants to go
void _changeTab(int index) {
  setState(() {
    _selectedIndex = index;
  });
}

// This method checks whether
// the user is in a Session state when trying
// to change screen.
// It takes the index of the screen in which the user
// is currently in and whether has confirmed that
// he/she will
Future<void> _checkIfUserExitsFromSession(
  int index,
  bool abortFromTabChange,
) async {
  // Take the current user
  final user = Provider.of<UserProvider>(context, listen: false).user!;

  // If the user is not a teacher
  // then we don't care
  // and change page
  if (!user.isTeacher!) {
    _changeTab(index);
```

```
    return;
  }

  // We are on sessions screen and
  //authorization process has started
  if (_selectedIndex == 2 &&
      index != _selectedIndex &&
      Provider.of<SessionProvider>(context, listen: false).startedScanning) {
    await _showExitDialog(context);
  } else {
    _changeTab(index);
  }

  if (abortFromTabChange) {
    _changeTab(index);
  }
}

// Helper function for the showDialog widget.
// It closes popup if the user presses cancel
void _cancelChangeTabFromNewSession() {
  Provider.of<SessionProvider>(context, listen: false)
      .abortSessionFromTabChange = false;
  Navigator.of(context).pop();
}

// Helper function for the showDialog widget.
// It aborts the session that the user initiated
void _confirmChangeTabFromNewSession() {
  Provider.of<SessionProvider>(context, listen: false)
      .abortSessionFromTabChange = true;
```

```
  Provider.of<SessionProvider>(context, listen: false).stopSession();
  Navigator.of(context).pop();
}

// It shows a popup that warns the user that he/she's trying
// to cancel the authorization process
Future<void> _showExitDialog(BuildContext context) async {
  return showDialog(
    barrierDismissible: false,
    context: context,
    builder: (context) => DialogWidget(
      titleText: 'Authorization cancel',
      content: const Text('Do you want to cancel the authorization process?'),
      confirmFunction: _confirmChangeTabFromNewSession,
      cancelFunction: _cancelChangeTabFromNewSession,
    ),
  );
}

// Signs out the user from the app.
// If a session is underway, it stops it.
Future<void> _logout() async {
  if (Provider.of<SessionProvider>(context, listen: false).startedScanning) {
    Provider.of<SessionProvider>(context, listen: false).stopSession();
  }

  Navigator.of(context).pop();
  await FirebaseAuth.instance.signOut();
}

// If the user pressess the logout button
```

```
// a popup is shown warning him
Future<void> _showLogoutDialog(BuildContext context) async {
  return showDialog(
    barrierDismissible: false,
    context: context,
    builder: (context) => DialogWidget(
      titleText: 'Logout',
      content: const Text('Are you sure you want to logout?'),
      confirmFunction: _logout,
    ),
  );
}

// When a user wants to start a new session
// A bottom popup is being displayed giving him
// all the options to choose
// This popup can't be dismissed!
Future<void> _showNewSessionDialog() async {
  return showModalBottomSheet<void>(
    context: context,
    constraints: BoxConstraints(minWidth: MediaQuery.of(context).size.width),
    enableDrag: false,
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        topLeft: Radius.circular(15.0),
        topRight: Radius.circular(15.0),
      ),
    ),
    builder: (context) => const NewSessionDialogWidget(),
  );
}
```

```
// When the user wants to filter his/her sessions
// a bottom popup is being shown with all the options
Future<void> _showFiltersDialog() async {
  return showModalBottomSheet<void>(
    context: context,
    showDragHandle: true,
    constraints: BoxConstraints(minWidth: MediaQuery.of(context).size.width),
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        topLeft: Radius.circular(15.0),
        topRight: Radius.circular(15.0),
      ),
    ),
    builder: (context) => const FiltersSessionOverviewDialogWidget(),
  );
}

@override
Widget build(BuildContext context) {
  // Duration of the animation of the floating action button
  const duration = Duration(milliseconds: 300);

  return Scaffold(
    backgroundColor: Theme.of(context).colorScheme.surface,
    appBar: AppBar(
      title: Text(_pages[_selectedIndex]['title']! as String),
      actions: [
        IconButton(
          onPressed: () => _showLogoutDialog(context),
          icon: const Tooltip(message: 'Logout', child: Icon(Icons.logout)),
```

```
    ),
  ],
),
// NotificationListener widget is used to determine
// whether the user reached the end of the scrollable
// page and hide the floating action button (session overview)
body: NotificationListener<UserScrollNotification>(
  onNotification: (notification) {
    final scrollDirection = notification.direction;
    setState(() {
      if (scrollDirection == ScrollDirection.reverse) {
        _showFloatingActionButton = false;
      } else if (scrollDirection == ScrollDirection.forward ||
          scrollDirection == ScrollDirection.idle) {
        _showFloatingActionButton = true;
      }
    });

    return true;
  },
  child: _pages[_selectedIndex]['page']! as Widget,
),
floatingActionButton: (_selectedIndex == 2 &&
            !Provider.of<SessionProvider>(context).startedScanning) ||
        _selectedIndex == 3
    ? AnimatedSlide(
        duration: duration,
        offset:
            _showFloatingActionButton ? Offset.zero : const Offset(0, 2),
        child: AnimatedOpacity(
          duration: duration,
```

```
        opacity: _showFloatingActionButton ? 1 : 0,
        child: FloatingActionButton(
          onPressed: _selectedIndex == 2
              ? () async {
                  await Provider.of<SessionProvider>(
                    context,
                    listen: false,
                  ).populateFormData(
                    1,
                    Provider.of<UserProvider>(context, listen: false)
                        .user!
                        .id!,
                    context,
                  );
                  _showNewSessionDialog();
                }
              : () async => _showFiltersDialog(),
          tooltip:
              _selectedIndex == 2 ? 'Add a new session' : 'Add filters',
          backgroundColor: _selectedIndex == 2
              ? const Color.fromRGBO(232, 52, 93, 1.0)
              : const Color.fromARGB(255, 204, 170, 49),
          child: _selectedIndex == 2
              ? const Icon(Icons.add)
              : const Icon(Icons.settings),
        ),
      ),
    )
  : null,
bottomNavigationBar: BottomNavigationBar(
  unselectedItemColor: Theme.of(context).colorScheme.secondary,
```

```
selectedItemColor: Theme.of(context).colorScheme.surface,
type: BottomNavigationBarType.shifting,
items:
    Provider.of<UserProvider>(context, listen: false).user!.isTeacher!
        ? [
            BottomNavigationBarItem(
              icon: const Icon(Icons.person),
              label: 'Info',
              tooltip: 'User info',
              backgroundColor: Theme.of(context).primaryColor,
            ),
            const BottomNavigationBarItem(
              icon: Icon(Icons.school),
              label: 'Classes',
              tooltip: "User's classes",
              backgroundColor: Color.fromRGBO(66, 183, 42, 1.0),
            ),
            const BottomNavigationBarItem(
              icon: Icon(Icons.class_),
              label: 'Session',
              tooltip: 'Start a session',
              backgroundColor: Color.fromRGBO(232, 52, 93, 1.0),
            ),
            const BottomNavigationBarItem(
              icon: Icon(Icons.list_alt_rounded),
              label: 'Overview',
              tooltip: 'Show sessions',
              backgroundColor: Color.fromARGB(255, 204, 170, 49),
            ),
          ]
        : [
```

```
                    BottomNavigationBarItem(
                      icon: const Icon(Icons.person),
                      label: 'Info',
                      tooltip: 'User info',
                      backgroundColor: Theme.of(context).primaryColor,
                    ),
                    const BottomNavigationBarItem(
                      icon: Icon(Icons.school),
                      label: 'Classes',
                      tooltip: "User's classes",
                      backgroundColor: Color.fromRGBO(66, 183, 42, 1.0),
                    ),
                  ],
        currentIndex: _selectedIndex,
        onTap: (index) => _checkIfUserExitsFromSession(
          index,
          Provider.of<SessionProvider>(context, listen: false)
              .abortSessionFromTabChange,
        ),
      ),
    );
  }
}
```

screens/user_classes_screen.dart

```
// ignore_for_file: avoid_dynamic_calls


import 'package:cloud_firestore/cloud_firestore.dart';
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/classes_provider.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:flutter_uni_access/widgets/display_user_classes_widget.dart';
```

```
import 'package:provider/provider.dart';


class UserClassesScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    // Reference to the labs collection
    late final CollectionReference labs =
        FirebaseFirestore.instance.collection('labs');
    // Current user ID
    late final userId =
        Provider.of<UserProvider>(context, listen: false).user!.id!;
    // Number of classes the user is attending
    late final studentClassesLength =
        Provider.of<ClassesProvider>(context, listen: false)
            .studentClasses
            ?.length;
    // The classes the user attends
    var userClasses =
        Provider.of<ClassesProvider>(context, listen: false).studentClasses;


    // If we don't have the student classes in cache
    // then ask firebase.
    // Else serve them
    return studentClassesLength == 0
        ? FutureBuilder(
            future: labs.get(),
            builder: (
              BuildContext context,
              AsyncSnapshot<QuerySnapshot<Object?>> snapshot,
            ) {
              if (snapshot.hasError) {
```

```
            return Center(child: Text(snapshot.error.toString()));
          }

          if (snapshot.connectionState == ConnectionState.waiting) {
            return const Center(child: CircularProgressIndicator());
          }

          if (snapshot.connectionState == ConnectionState.done) {
            Provider.of<ClassesProvider>(context, listen: false)
                .prepareClassesData(snapshot, userId);
            userClasses =
                Provider.of<ClassesProvider>(context, listen: false)
                    .studentClasses;

            return ListView.builder(
              itemCount: userClasses?.length,
              itemBuilder: (BuildContext context, int index) =>
                  DisplayUserClasses(userClasses![index]),
            );
          }
          return const CircularProgressIndicator();
        },
      )
    : ListView.builder(
        itemCount: userClasses?.length,
        itemBuilder: (BuildContext context, int index) =>
            DisplayUserClasses(userClasses![index]),
      );
  }
}
```

screens/user_info_screen.dart

```
import 'package:flutter/material.dart';


import 'package:flutter_uni_access/widgets/display_user_info_widget.dart';


class UsertInfoScreen extends StatelessWidget {
  @override
  Widget build(BuildContext context) {
    return DisplayUserInfo();
  }
}
```

utils/capitalize_first_lettter.dart

```
extension CapitalizeFirstLetter on String {
  String capitalize() {
    return "${this[0].toUpperCase()}${substring(1).toLowerCase()}";
  }
}
```

widgets/alert_result_widget.dart

```
import 'package:flutter/material.dart';


class AlertResultWidget extends StatefulWidget {
  const AlertResultWidget({Key? key, required int mode, required String msg})
      : _mode = mode,
        _msg = msg,
        super(key: key);


  final int _mode;
  final String _msg;


  @override
```

```
  State<AlertResultWidget> createState() => _AlertResultWidgetState();
}

class _AlertResultWidgetState extends State<AlertResultWidget>
    with SingleTickerProviderStateMixin {
  late final AnimationController animationController;
  late final Animation<double> scaleAnimation;

  @override
  void initState() {
    super.initState();

    animationController = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 300),
    );
    scaleAnimation = CurvedAnimation(
      parent: animationController,
      curve: Curves.easeInOut,
    );

    animationController.forward();
  }

  @override
  void dispose() {
    animationController.dispose();

    super.dispose();
  }
```

```
@override
Widget build(BuildContext context) {
  IconData? icon;
  Color? color;

  switch (widget._mode) {
    case 1:
      icon = Icons.check_circle;
      color = Colors.green;
      break;
    case 2:
      icon = Icons.error_rounded;
      color = Colors.red;
      break;
    case 3:
      icon = Icons.warning_rounded;
      color = Colors.orange;
      break;
  }

  return ScaleTransition(
    scale: scaleAnimation,
    child: AlertDialog(
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      title: const Text('Result'),
      content: Row(children: [Icon(icon), Text(widget._msg)]),
```

```
        backgroundColor: color,
      ),
    );
  }
}
```

widgets/alert_student_profile_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/uni_user.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/utils/capitalize_first_letter.dart';
import 'package:provider/provider.dart';

class AlertStudentProfileWidget extends StatefulWidget {
  const AlertStudentProfileWidget({Key? key, required UniUser user})
      : _user = user,
        super(key: key);

  final UniUser _user;

  @override
  State<AlertStudentProfileWidget> createState() =>
      _AlertStudentProfileWidgetState();
}

class _AlertStudentProfileWidgetState extends State<AlertStudentProfileWidget>
    with SingleTickerProviderStateMixin {
  late final AnimationController animationController;
  late final Animation<double> scaleAnimation;

  @override
  void initState() {
```

```
  super.initState();


  animationController = AnimationController(
    vsync: this,
    duration: const Duration(milliseconds: 300),
  );
  scaleAnimation = CurvedAnimation(
    parent: animationController,
    curve: Curves.easeInOut,
  );


  animationController.forward();
}


@override
void dispose() {
  animationController.dispose();


  super.dispose();
}


@override
Widget build(BuildContext context) {
  return ScaleTransition(
    scale: scaleAnimation,
    child: AlertDialog(
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
```

```
),
title: const Center(
  child: Text(
    'Student Info',
    style: TextStyle(color: Colors.black),
  ),
),
content: Column(
  mainAxisSize: MainAxisSize.min,
  crossAxisAlignment: CrossAxisAlignment.start,
  children: [
    const Center(child: Icon(Icons.person_rounded, size: 150.0)),
    const Text('ID', style: TextStyle(fontWeight: FontWeight.w100)),
    Text(
      widget._user.id!,
      style: const TextStyle(fontSize: 25),
    ),
    const SizedBox(
      height: 10,
    ),
    const Text('Name', style: TextStyle(fontWeight: FontWeight.w100)),
    Text(
      widget._user.name.toString().capitalize(),
      style: const TextStyle(fontSize: 25),
    ),
    const SizedBox(
      height: 10,
    ),
    const Text(
      'Surname',
      style: TextStyle(fontWeight: FontWeight.w100),
```

```
    ),
    Text(
      widget._user.surname.toString().capitalize(),
      style: const TextStyle(fontSize: 25),
    ),
    const SizedBox(
      height: 10,
    ),
    const Text('Email', style: TextStyle(fontWeight: FontWeight.w100)),
    Text(
      widget._user.email!,
      style: const TextStyle(fontSize: 25),
    ),
    const SizedBox(
      height: 10,
    ),
    const Text(
      'Attendences',
      style: TextStyle(fontWeight: FontWeight.w100),
    ),
    Text(
      Provider.of<SessionProvider>(context, listen: false)
          .currentUserAttendanceInSessionOverview
          .toString(),
      style: const TextStyle(fontSize: 25),
    ),
  ],
),
actions: [
  Tooltip(
    message: 'Close',
```

```
            child: TextButton(
              onPressed: () => Navigator.of(context).pop(),
              child: const Text('Close'),
            ),
          ),
        ],
        actionsAlignment: MainAxisAlignment.center,
      ),
    );
  }
}
```

widgets/attendance_widget.dart

```
// ignore_for_file: avoid_print

import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/uni_user.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/utils/capitalize_first_letter.dart';
import 'package:flutter_uni_access/widgets/alert_student_profile_widget.dart';
import 'package:flutter_uni_access/widgets/scan_widget.dart';
import 'package:provider/provider.dart';

class AttendanceWidget extends StatefulWidget {
  const AttendanceWidget({Key? key}) : super(key: key);

  @override
  State<AttendanceWidget> createState() => _AttendanceWidgetState();
}

class _AttendanceWidgetState extends State<AttendanceWidget> {
  @override
```

```
void initState() {
  super.initState();

  WidgetsBinding.instance.addPostFrameCallback((_) async {
    await Provider.of<SessionProvider>(context, listen: false)
        .findAllPermittedStudents();

    if (!mounted) return;
  });
}

Future<void> _showStudentProfileAlertDialog(
  BuildContext context,
  UniUser user,
) async {
  await Provider.of<SessionProvider>(context, listen: false)
      .findStudentAttendances(user.id!, context);

  if (!context.mounted) return;

  return showDialog(
    barrierDismissible: false,
    context: context,
    builder: (context) => AlertStudentProfileWidget(user: user),
  );
}

@override
Widget build(BuildContext context) {
  return Column(
    children: [
```

```
Padding(
  padding: const EdgeInsets.all(8.0),
  child: Card(
    color: const Color.fromRGBO(255, 255, 255, 0.6),
    elevation: 8,
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        bottomLeft: Radius.circular(15.0),
        bottomRight: Radius.circular(15.0),
      ),
    ),
    child: SizedBox(
      height: MediaQuery.of(context).size.height / 2,
      child: SingleChildScrollView(
        child: Column(
          children: Provider.of<SessionProvider>(context)
              .sessionUsers!
              .map(
                (user) => ListTile(
                  leading: user.isAuthorized
                      ? const Icon(
                          Icons.check_circle_rounded,
                          color: Colors.green,
                          size: 35,
                        )
                      : const Icon(
                          Icons.close_rounded,
                          color: Colors.red,
                          size: 35,
                        ),
                  title: Text(
```

```
                            '${user.name.toString().capitalize()}
${user.surname.toString().capitalize()}',
                          ),
                          subtitle: user.isTeacher!
                              ? Text('Teacher - ${user.id}')
                              : Text('Student - ${user.id}'),
                          onTap: () async => _showStudentProfileAlertDialog(
                            context,
                            user,
                          ),
                        ),
                      )
                      .toList(),
                ),
              ),
            ),
          ),
        ),
        Center(
          child: Tooltip(
            message: 'Scan a QR code',
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: const Color.fromRGBO(232, 52, 93, 1.0),
              ),
              onPressed: () => Navigator.of(context).push(
                MaterialPageRoute(builder: (context) => const ScanWidget()),
              ),
              child: const Text(
                'Scan',
                style: TextStyle(color: Colors.white),
              ),
```

```
            ),
          ),
        ),
      ],
    );
  }
}
```

widgets/auth_form_widget.dart

```
import 'package:flutter/material.dart';


class AuthForm extends StatefulWidget {
  final void Function(String username, String password, BuildContext ctx)
      submitFn;
  final bool isLoading;


  // ignore: avoid_positional_boolean_parameters
  const AuthForm(this.submitFn, this.isLoading);


  @override
  State<AuthForm> createState() => _AuthFormState();
}


class _AuthFormState extends State<AuthForm> {
  final _formKey = GlobalKey<FormState>();
  late String _userEmail;
  late String _userPassword;


  void _trySubmit() {
    final isValid = _formKey.currentState?.validate();
    FocusScope.of(context).unfocus();
```

```
  if (isValid!) {
    _formKey.currentState?.save();
    widget.submitFn(_userEmail.trim(), _userPassword.trim(), context);
  }
}

@override
Widget build(BuildContext context) {
  return Center(
    child: Card(
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      color: Theme.of(context).colorScheme.secondary,
      margin: const EdgeInsets.all(20),
      child: SingleChildScrollView(
        child: Padding(
          padding: const EdgeInsets.all(16),
          child: Form(
            key: _formKey,
            child: Column(
              mainAxisSize: MainAxisSize.min,
              children: [
                TextFormField(
                  key: const ValueKey('email'),
                  validator: (value) {
                    if (value!.isEmpty ||
                        !value.contains('@') ||
```

```
            !value.contains('uni')) {
          return 'Please enter a valid email address.';
        }
        return null;
      },
      keyboardType: TextInputType.emailAddress,
      decoration:
          const InputDecoration(labelText: 'Email address'),
      textInputAction: TextInputAction.next,
      autofocus: true,
      onSaved: (value) => _userEmail = value!,
    ),
    TextFormField(
      key: const ValueKey('password'),
      validator: (value) {
        if (value!.isEmpty) {
          return 'Password must not be empty';
        }
        return null;
      },
      decoration: const InputDecoration(labelText: 'Password'),
      obscureText: true,
      textInputAction: TextInputAction.done,
      onSaved: (value) => _userPassword = value!,
    ),
    const SizedBox(
      height: 12,
    ),
    if (widget.isLoading) const CircularProgressIndicator(),
    if (!widget.isLoading)
      ElevatedButton(
```

```
                    onPressed: _trySubmit,
                    child: const Text('Login'),
                  ),
                ],
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

widgets/custom_bottom_outline_button.dart

```
import 'package:flutter/material.dart';


class CustomBottomOutlineButton extends StatelessWidget {
  const CustomBottomOutlineButton({
    Key? key,
    required String hintMessage,
    required double width,
    required double height,
    required VoidCallback onPressedMethod,
    required IconData icon,
    required String label,
  })  : _hintMessage = hintMessage,
        _width = width,
        _height = height,
        _onPressedMethod = onPressedMethod,
        _icon = icon,
        _label = label,
        super(key: key);
```

```
  final String _hintMessage;
  final double _width;
  final double _height;
  final VoidCallback? _onPressedMethod;
  final IconData _icon;
  final String _label;

  @override
  Widget build(BuildContext context) {
    return Tooltip(
      message: _hintMessage,
      child: OutlinedButton.icon(
        style: OutlinedButton.styleFrom(fixedSize: Size(_width, _height)),
        onPressed: _onPressedMethod,
        icon: Icon(_icon),
        label: Text(_label),
      ),
    );
  }
}
```

widgets/dialog_widget.dart

```
import 'package:flutter/material.dart';

class DialogWidget extends StatefulWidget {
  const DialogWidget({
    Key? key,
    required String titleText,
    required Widget content,
    required VoidCallback confirmFunction,
    VoidCallback? cancelFunction,
```

```
  })  : _titleText = titleText,
        _content = content,
        _confirmFunction = confirmFunction,
        _cancelFunction = cancelFunction,
        super(key: key);

  final String _titleText;
  final Widget _content;
  final VoidCallback? _confirmFunction;
  final VoidCallback? _cancelFunction;

  @override
  State<DialogWidget> createState() => _DialogWidgetState();
}

class _DialogWidgetState extends State<DialogWidget>
    with SingleTickerProviderStateMixin {
  late final AnimationController animationController;
  late final Animation<double> scaleAnimation;

  @override
  void initState() {
    super.initState();

    animationController = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 300),
    );
    scaleAnimation = CurvedAnimation(
      parent: animationController,
      curve: Curves.easeInOut,
```

```
  );

  animationController.forward();
}

@override
void dispose() {
  animationController.dispose();

  super.dispose();
}

@override
Widget build(BuildContext context) {
  return ScaleTransition(
    scale: scaleAnimation,
    child: AlertDialog(
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      title: Text(
        widget._titleText,
        style: const TextStyle(color: Colors.black),
      ),
      content: widget._content,
      actions: [
        Tooltip(
          message: 'Confirm',
```

```
            child: TextButton(
              onPressed: widget._confirmFunction,
              child: const Text('Confirm'),
            ),
          ),
          Tooltip(
            message: 'Cancel',
            child: TextButton(
              onPressed:
                  widget._cancelFunction ?? () => Navigator.of(context).pop(),
              child: const Text('Cancel'),
            ),
          ),
        ],
      ),
    );
  }
}
```

widgets/display_user_classes_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/student_classes_card.dart';

class DisplayUserClasses extends StatelessWidget {
  final StudentClassesCard card;

  const DisplayUserClasses(this.card);

  @override
  Widget build(BuildContext context) {
    return Card(
      elevation: 5,
```

```
      margin: const EdgeInsets.all(10),
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      color: Theme.of(context).colorScheme.secondary,
      child: ExpansionTile(
        leading: const Icon(Icons.school),
        title: Text('Lab: ${card.lab!}'),
        subtitle: const Text('Tap the arrow to see the subjects.'),
        collapsedIconColor: const Color.fromRGBO(66, 183, 42, 1.0),
        childrenPadding: const EdgeInsets.all(8.0),
        children: [
          ListView.builder(
            shrinkWrap: true,
            physics: const NeverScrollableScrollPhysics(),
            itemCount: card.subjects?.length,
            itemBuilder: (BuildContext context, int index) {
              return Column(
                children: [
                  Center(
                    child: Text(
                      '${card.subjects![index].keys.first}                    -
${card.subjects![index].values.first}',
                      style: const TextStyle(fontSize: 20),
                    ),
                  ),
                  if (index != ((card.subjects?.length)! - 1))
                    const Divider(
                      thickness: 2.0,
```

```
                    )
                  else
                    const SizedBox.shrink(),
                ],
              );
            },
          ),
        ],
      ),
    );
  }
}
```

widgets/display_user_info_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:flutter_uni_access/utils/capitalize_first_letter.dart';
import 'package:flutter_uni_access/widgets/custom_bottom_outline_button.dart';
import 'package:flutter_uni_access/widgets/qr_image_widget.dart';
import 'package:provider/provider.dart';


class DisplayUserInfo extends StatelessWidget {
  // This method displays a popup
  // showing the qr code of the user
  Future<void> _showUserQrCode(
    BuildContext context,
    String? id,
  ) async {
    return showDialog(
      barrierDismissible: false,
      context: context,
      builder: (context) => AlertQrImageWidget(id: id!),
```

```
  );
}

@override
Widget build(BuildContext context) {
  // Get current user
  final uniUser = Provider.of<UserProvider>(context, listen: false).user!;
  // With what number we will divide
  // the height of the screen
  int denominator;

  // If the user is a teacher
  // then the image can be bigger
  // since we don't need space for
  // the qr code button
  if (uniUser.isTeacher!) {
    denominator = 2;
  } else {
    denominator = 3;
  }

  return Card(
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        bottomLeft: Radius.circular(15.0),
        bottomRight: Radius.circular(15.0),
      ),
    ),
    color: Theme.of(context).colorScheme.secondary,
    elevation: 5,
    child: Padding(
```

```
padding: const EdgeInsets.all(8.0),
child: Column(
  children: [
    Container(
      height: MediaQuery.of(context).size.height / denominator,
      decoration: BoxDecoration(
        borderRadius: const BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
        image: DecorationImage(
          image: Image.memory(uniUser.image!).image,
          fit: BoxFit.cover,
        ),
      ),
    ),
    const SizedBox(height: 30),
    Row(
      mainAxisAlignment: MainAxisAlignment.spaceBetween,
      children: [
        Text(
          'ID: ',
          style: Theme.of(context).textTheme.headlineSmall,
        ),
        Text(
          uniUser.id!,
          style: Theme.of(context).textTheme.headlineSmall,
        ),
      ],
    ),
    Row(
```

```
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(
                'Name: ',
                style: Theme.of(context).textTheme.headlineSmall,
              ),
              Text(
                '${uniUser.name.toString().capitalize()}
${uniUser.surname.toString().capitalize()}',
                style: Theme.of(context).textTheme.headlineSmall,
              ),
            ],
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(
                'E-mail: ',
                style: Theme.of(context).textTheme.headlineSmall,
              ),
              Text(
                '${uniUser.email}',
                style: Theme.of(context).textTheme.headlineSmall,
              ),
            ],
          ),
          Row(
            mainAxisAlignment: MainAxisAlignment.spaceBetween,
            children: [
              Text(
                'Role: ',
                style: Theme.of(context).textTheme.headlineSmall,
```

```
            ),
            Text(
              !uniUser.isTeacher! ? 'Student' : 'Teacher',
              style: Theme.of(context).textTheme.headlineSmall,
            ),
          ],
        ),
        if (!uniUser.isTeacher!)
          Column(
            children: [
              const Divider(
                thickness: 3.0,
              ),
              SizedBox(
                height: MediaQuery.of(context).size.height / 15,
              ),
              CustomBottomOutlineButton(
                hintMessage: 'Show QR code',
                width: MediaQuery.of(context).size.width / 2,
                height: MediaQuery.of(context).size.height / 15,
                onPressedMethod: () async =>
                    _showUserQrCode(context, uniUser.id),
                icon: Icons.qr_code_scanner_rounded,
                label: 'Show QR code',
              ),
            ],
          ),
      ],
    ),
  ),
);
```

```
  }
}
```

widgets/filter_session_overview_form_option_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_overview_provider.dart';
import 'package:provider/provider.dart';

class FilterSessionOverviewFormOptionWidget extends StatefulWidget {
  const FilterSessionOverviewFormOptionWidget({
    Key? key,
    required List<String> options,
    required String titleOption,
  })  : _options = options,
        _titleOption = titleOption,
        super(key: key);

  final List<String> _options;
  final String _titleOption;

  @override
  State<FilterSessionOverviewFormOptionWidget> createState() =>
      _FilterSessionOverviewFormOptionWidgetState();
}

class _FilterSessionOverviewFormOptionWidgetState
    extends State<FilterSessionOverviewFormOptionWidget> {
  String? _selectedItem;

  @override
  Widget build(BuildContext context) {
```

```
return SizedBox(
  width: MediaQuery.of(context).size.width / 1.5,
  child: InputDecorator(
    decoration: const InputDecoration(
      border: OutlineInputBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15),
          bottomRight: Radius.circular(15),
        ),
      ),
    ),
    child: DropdownButtonFormField<String>(
      decoration: const InputDecoration.collapsed(hintText: ''),
      isExpanded: true,
      borderRadius: const BorderRadius.only(
        bottomLeft: Radius.circular(15.0),
        bottomRight: Radius.circular(15.0),
      ),
      dropdownColor: Theme.of(context).colorScheme.surface,
      hint: Text(widget._titleOption),
      value: _selectedItem ??= null,
      icon: const Tooltip(
        message: 'Expand options',
        child: Icon(Icons.arrow_drop_down_rounded),
      ),
      onChanged: (String? newValue) {
        final filters =
            Provider.of<SessionOverviewProvider>(context, listen: false)
                .filters;
        if (!filters!.contains(newValue)) {
          filters.add(newValue!);
```

```
            }
            Provider.of<SessionOverviewProvider>(context, listen: false)
                .filter();
            setState(() {
              _selectedItem = newValue;
            });
          },
          items: widget._options.map<DropdownMenuItem<String>>((value) {
            return DropdownMenuItem<String>(
              value: value,
              child: Text(value),
            );
          }).toList(),
        ),
      ),
    );
  }
}
```

widgets/filter_session_overview_dialog_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_overview_provider.dart';
import
'package:flutter_uni_access/widgets/filter_session_overview_form_option_widget.dart
';
import 'package:provider/provider.dart';


class FiltersSessionOverviewDialogWidget extends StatelessWidget {
  const FiltersSessionOverviewDialogWidget({Key? key}) : super(key: key);


  @override
  Widget build(BuildContext context) {
    return Wrap(
```

```
      direction: Axis.vertical,
      runAlignment: WrapAlignment.center,
      crossAxisAlignment: WrapCrossAlignment.center,
      children: [
        Text(
          'Filters',
          style: Theme.of(context).textTheme.headlineSmall,
        ),
        const SizedBox(height: 10),
        FilterSessionOverviewFormOptionWidget(
          key: const Key('lab'),
          options: Provider.of<SessionOverviewProvider>(context, listen: false)
              .labs!,
          titleOption: 'Lab',
        ),
        const SizedBox(
          height: 10,
        ),
        FilterSessionOverviewFormOptionWidget(
          key: const Key('subject'),
          options: Provider.of<SessionOverviewProvider>(context, listen: false)
              .subjects!,
          titleOption: 'Subject',
        ),
      ],
    );
  }
}
```

widgets/new_session_dialog_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/widgets/new_session_form_option_widget.dart';
import 'package:provider/provider.dart';

class NewSessionDialogWidget extends StatelessWidget {
  const NewSessionDialogWidget({Key? key}) : super(key: key);

  void _startSession(BuildContext context) {
    Provider.of<SessionProvider>(context, listen: false).startedScanning = true;
    Provider.of<SessionProvider>(context, listen: false).canSaveSession = false;
    Navigator.of(context).pop();
  }

  @override
  Widget build(BuildContext context) {
    return Padding(
      padding: const EdgeInsets.all(16),
      child: Wrap(
        direction: Axis.vertical,
        runAlignment: WrapAlignment.center,
        crossAxisAlignment: WrapCrossAlignment.center,
        children: [
          Text(
            'Register a new session',
            style: Theme.of(context).textTheme.headlineSmall,
          ),
          NewSessionFormOptionWidget(
            key: const Key('lab'),
```

```
            option: Provider.of<SessionProvider>(context, listen: false).labs,
            titleOption: 'Lab',
          ),
          NewSessionFormOptionWidget(
            key: const Key('subject'),
            option: Provider.of<SessionProvider>(context).subjects,
            titleOption: 'Subject',
          ),
          Tooltip(
            message: 'Start the authorization process',
            child: ElevatedButton(
              style: ElevatedButton.styleFrom(
                backgroundColor: const Color.fromRGBO(232, 52, 93, 1.0),
              ),
              onPressed: Provider.of<SessionProvider>(context).selectedLab !=
                          null &&
                      Provider.of<SessionProvider>(context).selectedSubject !=
                          null
                  ? () => _startSession(context)
                  : null,
              child: const Text(
                'Start',
                style: TextStyle(color: Colors.white),
              ),
            ),
          ),
        ],
      ),
    );
  }
}
```

widgets/new_session_form_option_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:provider/provider.dart';

class NewSessionFormOptionWidget extends StatefulWidget {
  const NewSessionFormOptionWidget({
    Key? key,
    required List<String> option,
    required String titleOption,
  })  : _option = option,
        _titleOption = titleOption,
        super(key: key);

  final List<String> _option;
  final String _titleOption;

  @override
  State<NewSessionFormOptionWidget> createState() => _FormOptionState();
}

class _FormOptionState extends State<NewSessionFormOptionWidget> {
  String? _selectedItem;

  void _updateSessionSelectedItem(String? newValue, BuildContext context) {
    if (newValue!.contains(':')) {
      Provider.of<SessionProvider>(context, listen: false).selectedSubject =
          newValue;
    } else {
```

```
    Provider.of<SessionProvider>(context, listen: false).selectedLab =
        newValue;
  }
}

@override
Widget build(BuildContext context) {
  return SizedBox(
    width: MediaQuery.of(context).size.width / 1.5,
    child: InputDecorator(
      decoration: const InputDecoration(
        border: OutlineInputBorder(
          borderRadius: BorderRadius.only(
            bottomLeft: Radius.circular(15),
            bottomRight: Radius.circular(15),
          ),
        ),
      ),
      child: DropdownButtonFormField<String>(
        decoration: const InputDecoration.collapsed(hintText: ''),
        isExpanded: true,
        borderRadius: const BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
        dropdownColor: Theme.of(context).colorScheme.background,
        hint: Text(widget._titleOption),
        value: _selectedItem ??= null,
        icon: const Tooltip(
          message: 'Expand options',
          child: Icon(Icons.arrow_drop_down_rounded),
```

```
    ),
    onChanged: widget.key.toString().compareTo("[<'lab'>]") == 0 ||
            (widget.key.toString().compareTo("[<'subject'>]") == 0 &&
                Provider.of<SessionProvider>(context, listen: false)
                        .selectedLab !=
                    null)
        ? (String? newValue) async {
            _updateSessionSelectedItem(newValue, context);
            if (widget.key.toString().compareTo("[<'lab'>]") == 0) {
              await Provider.of<SessionProvider>(context, listen: false)
                  .populateFormData(
                2,
                Provider.of<UserProvider>(context, listen: false)
                    .user!
                    .id!,
                context,
              );
            }
            setState(() {
              _selectedItem = newValue;
            });
          }
        : null,
    items: widget._option.map<DropdownMenuItem<String>>((value) {
      return DropdownMenuItem<String>(
        value: value,
        child: Text(value),
      );
    }).toList(),
  ),
),
```

```
    );
  }
}
```

widgets/qr_image_widget.dart

```
import 'package:flutter/material.dart';
import 'package:qr_flutter/qr_flutter.dart';

class AlertQrImageWidget extends StatefulWidget {
  const AlertQrImageWidget({
    Key? key,
    required this.id,
  }) : super(key: key);

  final String id;

  @override
  State<AlertQrImageWidget> createState() => _AlertQrImageWidgetState();
}

class _AlertQrImageWidgetState extends State<AlertQrImageWidget>
    with SingleTickerProviderStateMixin {
  late final AnimationController animationController;
  late final Animation<double> scaleAnimation;

  @override
  void initState() {
    super.initState();

    animationController = AnimationController(
      vsync: this,
      duration: const Duration(milliseconds: 300),
```

```
  );
  scaleAnimation = CurvedAnimation(
    parent: animationController,
    curve: Curves.easeInOut,
  );

  animationController.forward();
}

@override
void dispose() {
  animationController.dispose();

  super.dispose();
}

@override
Widget build(BuildContext context) {
  return ScaleTransition(
    scale: scaleAnimation,
    child: AlertDialog(
      shape: const RoundedRectangleBorder(
        borderRadius: BorderRadius.only(
          bottomLeft: Radius.circular(15.0),
          bottomRight: Radius.circular(15.0),
        ),
      ),
      title: const Center(
        child: Text(
          'QR code',
          style: TextStyle(color: Colors.black),
```

```
          ),
        ),
        content: SizedBox(
          width: 200,
          height: 250,
          child: Column(
            children: [
              Center(
                child: QrImageView(
                  data: widget.id,
                  size: 200,
                  backgroundColor: Colors.white,
                ),
              ),
              Tooltip(
                message: 'Close popup',
                child: TextButton(
                  onPressed: () => Navigator.of(context).pop(),
                  child: const Text('Close'),
                ),
              ),
            ],
          ),
        ),
      ),
    );
  }
}
```

widgets/scan_widget.dart

```
import 'package:flutter/material.dart';

import 'package:flutter/services.dart';

import 'package:flutter_uni_access/providers/session_provider.dart';

import 'package:flutter_uni_access/widgets/alert_result_widget.dart';

import 'package:mobile_scanner/mobile_scanner.dart';

import 'package:provider/provider.dart';


class ScanWidget extends StatelessWidget {

  const ScanWidget({Key? key}) : super(key: key);


  Future<void> _showAuthorizeAlertDialog(

    BuildContext context,

    int mode,

    String msg,

  ) async {

    return showDialog(

      barrierDismissible: false,

      context: context,

      builder: (BuildContext context) {

        Future.delayed(

          const Duration(seconds: 2),

          () => Navigator.of(context).pop(true),

        );

        return AlertResultWidget(mode: mode, msg: msg);

      },

    );

  }


  @override
```

```
Widget build(BuildContext context) {
  return MobileScanner(
    onDetect: (barcode) async {
      if (barcode.raw == null) {
        ScaffoldMessenger.of(context).showSnackBar(
          SnackBar(
            content: const Text('Barcode has no value'),
            backgroundColor: Theme.of(context).colorScheme.error,
          ),
        );

        return;
      }

      final result = barcode.barcodes;
      HapticFeedback.vibrate();
      Provider.of<SessionProvider>(context, listen: false)
          .authorizeUser(result.first.rawValue);

      final rightsResult =
          Provider.of<SessionProvider>(context, listen: false).authResult;

      var msg = '';
      switch (rightsResult) {
        case 1:
          msg = 'User Authorized';
          break;
        case 2:
          msg = 'User was not authorized';
          break;
        case 3:
```

```
            msg = 'User is already authorized';
            break;
        }

        Navigator.of(context).pop();

        await _showAuthorizeAlertDialog(context, rightsResult, msg);
      },
    );
  }
}
```

widgets/session_overview_card_widget.dart

```
// ignore_for_file: use_build_context_synchronously

import 'package:flutter/material.dart';
import 'package:flutter_uni_access/models/session.dart';
import 'package:flutter_uni_access/models/uni_user.dart';
import 'package:flutter_uni_access/providers/session_provider.dart';
import 'package:flutter_uni_access/providers/user_provider.dart';
import 'package:flutter_uni_access/widgets/alert_student_profile_widget.dart';
import 'package:provider/provider.dart';

class SessionOverviewCardWidget extends StatelessWidget {
  const SessionOverviewCardWidget({Key? key, required Session session})
      : _session = session,
        super(key: key);

  final Session _session;

  Future<void> _showStudentProfileAlertDialog(
    BuildContext context,
```

```
  UniUser user,
) async {
  Provider.of<SessionProvider>(context, listen: false).selectedSubject =
      _session.subject;
  await Provider.of<SessionProvider>(context, listen: false)
      .findStudentAttendances(user.id!, context);
  return showDialog(
    barrierDismissible: false,
    context: context,
    builder: (context) => AlertStudentProfileWidget(user: user),
  );
}

@override
Widget build(BuildContext context) {
  return Card(
    elevation: 5,
    margin: const EdgeInsets.all(10),
    shape: const RoundedRectangleBorder(
      borderRadius: BorderRadius.only(
        bottomLeft: Radius.circular(15.0),
        bottomRight: Radius.circular(15.0),
      ),
    ),
    color: Theme.of(context).colorScheme.secondary,
    child: ExpansionTile(
      leading: const Icon(Icons.list_alt_rounded),
      title: Text(
        '${_session.lab} - ${_session.subject!}',
        style: const TextStyle(fontSize: 13.5, fontWeight: FontWeight.bold),
      ),
```

```
        subtitle: Text('Saved at ${_session.timestamp}'),
        collapsedIconColor: const Color.fromARGB(255, 204, 170, 49),
        childrenPadding: const EdgeInsets.all(8.0),
        children: _session.studentsIds!
            .asMap()
            .entries
            .map(
              (studentId) => ListTile(
                leading: Text('${studentId.key + 1}'),
                title: Center(child: Text(studentId.value as String)),
                trailing: const Icon(Icons.person),
                onTap: () async {
                  final user =
                      await Provider.of<UserProvider>(context, listen: false)
                          .getUserFromId(studentId.value as String, context);

                  return _showStudentProfileAlertDialog(context, user);
                },
              ),
            )
            .toList(),
      ),
    );
  }
}
```

widgets/session_overview_list_filter_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_overview_provider.dart';
import 'package:provider/provider.dart';

class SessionOverviewFilterListWidget extends StatefulWidget {
```

```
  const SessionOverviewFilterListWidget({Key? key}) : super(key: key);


  @override
  State<SessionOverviewFilterListWidget> createState() =>
      _SessionOverviewFilterListWidgetState();
}


class _SessionOverviewFilterListWidgetState
    extends State<SessionOverviewFilterListWidget> {
  @override
  Widget build(BuildContext A) {
    final filters = Provider.of<SessionOverviewProvider>(context).filters!;


    return SizedBox(
      height: MediaQuery.of(context).size.height / 16,
      width: MediaQuery.of(context).size.width,
      child: ListView.builder(
        scrollDirection: Axis.horizontal,
        itemCount: filters.length,
        itemBuilder: (_, index) => Padding(
          padding: const EdgeInsets.only(top: 8.0, left: 5.0, bottom: 8.0),
          child: InkWell(
            onTap: () {
              setState(() {
                filters.removeAt(index);
              });
              Provider.of<SessionOverviewProvider>(
                context,
                listen: false,
              ).filter();
            },
```

```
            child: DecoratedBox(
              decoration: BoxDecoration(
                boxShadow: const [
                  BoxShadow(
                    color: Colors.grey,
                    blurRadius: 10.0,
                    spreadRadius: 1.0,
                    offset: Offset(0, 1),
                  ),
                ],
                color: const Color.fromARGB(255, 204, 170, 49),
                borderRadius: BorderRadius.circular(15.0),
              ),
              child: Padding(
                padding: const EdgeInsets.all(5.0),
                child: Center(child: Text(filters[index])),
              ),
            ),
          ),
        ),
      ),
    );
  }
}
```

widgets/session_overview_widget.dart

```
import 'package:flutter/material.dart';
import 'package:flutter_uni_access/providers/session_overview_provider.dart';
import 'package:flutter_uni_access/widgets/session_overview_card_widget.dart';
import
'package:flutter_uni_access/widgets/session_overview_filter_list_widget.dart';
import 'package:provider/provider.dart';
```

```
class SessionOverviewsWidget extends StatelessWidget {
  const SessionOverviewsWidget({Key? key}) : super(key: key);

  @override
  Widget build(BuildContext context) {
    final filteredSessions =
        Provider.of<SessionOverviewProvider>(context).sessions!;
    final filters =
        Provider.of<SessionOverviewProvider>(context, listen: false).filters!;

    return SingleChildScrollView(
      child: Column(
        children: [
          if (filters.isNotEmpty)
            const SessionOverviewFilterListWidget()
          else
            Container(),
          if (filteredSessions.isNotEmpty)
            ListView.builder(
              shrinkWrap: true,
              physics: const NeverScrollableScrollPhysics(),
              itemCount: filteredSessions.length,
              itemBuilder: (_, int index) => SessionOverviewCardWidget(
                session: filteredSessions[index],
              ),
            )
          else
            SizedBox(
              height: MediaQuery.of(context).size.height / 1.5,
              child: const Center(
```

```
                child: Text(
                  'Could not find any sessions',
                  style: TextStyle(fontSize: 20.0),
                ),
              ),
            ),
          ],
        ),
      );
  }
}
```