ΠΑΝΕΠΙΣΤΗΜΙΟ ΔΥΤΙΚΗΣ ΑΤΤΙΚΗΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΥΠΟΛΟΓΙΣΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Ανάπτυξη Android Client εφαρμογής μέσω Web Scraping

Ρένιτσα Ντάνιελ
Α.Μ. 711171111

Εισηγητής: Ιωάννης Βογιατζής

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

Σχεδιασμός και Ανάπτυξη Android Client εφαρμογής μέσω Web Scraping

Ρένιτσα Ντάνιελ
711171111

Εισηγητής :

Ιωάννης Βογιατζής

Εξεταστές :

Βογιατζής Ιωάννης (καθηγητής)
Σγουροπούλου Κλειώ (καθηγήτρια)
Μελετίου Γεώργιος (ΕΔΙΠ)

Ioannis Voyiatzis
Digitally signed by Ioannis Voyiatzis
Date: 2023.01.17 10:59:20 +02'00'

Kleio Sgouropoulou
Digitally signed by Kleio Sgouropoulou
Date: 2023.01.13 15:35:20 +02'00'

Georgios Meletiou
Digitally signed by Georgios Meletiou
Date: 2022.11.10 09:30:26 +02'00'

Ημερομηνία εξέτασης 10/10/2022

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.

Ο/Η Δηλών/ούσα

Ευχαριστίες

Ολοκληρώνοντας τη διπλωματική μου εργασία θα ήθελα να ευχαριστήσω τον καθηγητή μου κο Βογιατζή για την εμπιστοσύνη που μου έδειξε και τη δυνατότητα που μου έδωσε να εκπονήσω αυτή τη διπλωματική εργασία.

Επίσης θα ήθελα να ευχαριστήσω ιδιαίτερα τον κο Μελετίου για το χρόνο που αφιέρωσε και την καθοδήγηση που παρείχε καθ’ όλη τη διάρκεια της συγγραφής αυτής της εργασίας.

Τέλος, θα ήθελα να ευχαριστήσω θερμά την οικογένειά μου και τους φίλους μου για την πολύτιμη εμπιστοσύνη και τη στήριξη τους καθ’ όλη τη διάρκεια των σπουδών μου.

Περίληψη

Στην ιστοσελίδα του eClass του τμήματος, μπορεί κανείς αφού συνδεθεί να βρει μεταξύ άλλων πληροφορίες για τις ανακοινώσεις και τις εργασίες για τα μαθήματα που τον αφορούν, αυτές οι δύο επιλογές είναι μάλλον οι πιο συχνά επισκεπτόμενες από τους χρήστες. Η παρούσα εργασία πραγματεύεται την ανάπτυξη εφαρμογής για την όσο άμεση ενημέρωση των χρηστών για καινούργια ανακοίνωση ή εργασία σε κάποιο μάθημα. Αυτό έγινε εφικτό χρησιμοποιώντας τη σχέση Client-Server όπου ο Server κάθε ένα λεπτό ελέγχει για τυχών καινούργια ανακοίνωση/ανακοινώσεις και εργασία/εργασίες και εάν υπάρξει τέτοια, ενημερώνει τους χρήστες που τους αφορούν δηλαδή είναι γραμμένη στο συγκεκριμένο μάθημα. Ο Client θα λαμβάνει τις ειδοποιήσεις αυτές του Server στο Android κινητό που θα έχει ήδη συνδεθεί στην εν λόγω εφαρμογή.

Λέξεις Κλειδιά

eClass, ανακοινώσεις, εργασίες, Client, Server, ειδοποίηση, Android

Abstract

On the eClass website, once logged in, the user can find information on announcements and works for the courses that concern him/her, these two options are probably the most frequently visited by users. In this research work an application is developed to inform users as directly as possible about a new announcement or works in a course. This was made possible by using Client-Server relationship where the Server checks every minute for any new announcement/announcements and work/works and if there is any, informs interested users who are enrolled in this course. The Client will receive these notifications from the Server on the Android mobile already connected to that application.

Keywords

eClass, announcements, works, Client, Server, notifications, Android

Table of Contents

Table of Figures

Εισαγωγή

Καθώς η συνεχής ενημέρωση των ανθρώπων γίνεται όλο και πιο απαιτητική στη σήμερον ημέρα, ο χρήστης πρέπει-χρειάζεται να ενημερώνεται όσο γίνεται πιο άμεσα. Έρευνα έδειξε [1] πως από το 2016 έως και το 2022 υπήρξε δραματική αύξηση των χρηστών που χρησιμοποιούν κινητές συσκευές smartphones. Το 2016 μόλις 3.7 δισεκατομμύρια χρήστες είχαν στην κατοχή τους και χρησιμοποιούσαν κινητή συσκευή, τώρα, 6 χρόνια αργότερα, ο αριθμός αυτός έφτασε στα 6.6 δισεκατομμύρια, υπολογισμοί λένε πως μέχρι το 2027 το 85% του πληθυσμού θα χρησιμοποιεί smartphone. Έχοντας αυτούς τους γιγαντιαίους αριθμούς μπορεί να βγει το συμπέρασμα πως οι χρήστες θα διευκολύνονταν περισσότερα εάν ενημερωνόντουσαν άμεσα για τυχόν ανακοινώσεις στο χώρο εργασίας τους, στο χώρο εκπαίδευσης τους κ.α. κατευθείαν από την κινητή συσκευή τους. Ήδη από τη νεανική ηλικία ενός παιδιού υπάρχει ανάγκη πληροφόρησης, για παράδειγμα μια αλλαγή στο αυριανό πρόγραμμα μαθημάτων ή για την προσωρινή διακοπή λειτουργίας του σχολείου λόγο καιρικών συνθηκών ή εκλογών. Αυτό ιδίως συμβαίνει στη φοιτητική περίοδο των μαθητών. Ο φοιτητής έχει συνήθως ένα αρκετά απαιτητικό πρόγραμμα. Από το πρώτο κιόλας έτος των σπουδών του έχει πλήθος εργασιών σε διάφορα μαθήματα, πράγμα που σημαίνει ότι χρειάζεται μια σχετική οργάνωση και εγρήγορση καθώς τις περισσότερες φορές οι εργασίες αυτές είναι με συγκεκριμένη διορία. Πέρα από τις εργασίες οι οποίες είναι μια από τις σημαντικότερες ασχολίες του φοιτητή υπάρχουν και οι σημειώσεις που αναρτώνται σε ένα μάθημα και αφορούν το διάβασμα του, όπως επίσης οι ανακοινώσεις, η εκδηλώσεις που μπορούν να λάβουν μέρος σε ένα Πανεπιστημιακό χώρο. Επιπλέον αρκετοί φοιτητές επιλέγουν είτε αναγκάζονται να επιλέξουν την παράλληλη εργασία τους σε έναν χώρο. Γνωρίζοντας αυτά τα λίγα στοιχεία, κανείς μπορεί να καταλάβει πως ένας φοιτητής έχει ένα αρκετά πολυάσχολο πρόγραμμα, δίνοντας του συγκεκριμένο ελεύθερο χρόνο και μια αν όχι συνεχή, αρκετά συχνή ανάγκη ενημέρωσης του τρέχων προγράμματος του. Είναι αρκετά σημαντική η έγκαιρη ενημέρωση του, για παράδειγμα στην ανάρτηση μια καινούργιας εργασίας, διότι ο φοιτητής όπως ειπώθηκε, λόγο της πλήρης απασχόλησης του ίσως να καθυστερήσει να το ελέγξει ο ίδιος και αυτό μπορεί να έχει ως επίπτωση τον αρκετά περιορισμένο πια χρόνο για τη λύση της ή και καθόλου γιατί μπορεί να έχει λήξει η διορία της. Αυτό είναι πιο συχνό φαινόμενο στις εγγραφές σε εργαστηριακά τμήματα καθώς κάποιοι αν όχι αρκετοί φοιτητές δεν προλαβαίνουν να γραφτούν γιατί είναι περιορισμένες οι θέσεις και ενημερώνονται για αυτές καθυστερημένα.

Με την αφορμή αυτή, σκοπός της διπλωματικής αυτής εργασίας είναι η κατασκευή και η μελέτη ενός συστήματος ενημέρωσης του χρήστη για τις δύο από τις βασικότερες λειτουργίες ενός μαθήματος, τις ανακοινώσεις του και τις εργασίες του. Κύριοι στόχοι της εφαρμογής αυτής είναι η όσο γίνεται πιο άμεση ενημέρωση των χρηστών μέσω ειδοποιήσεων στην κινητή του συσκευή και η φιλική αλληλεπίδραση με το χρήστη.

Ακολουθείται η εξής δομή. Αρχικά αναφέρεται μια από τις πιο δυνατές πλατφόρμες που ασχολούνται με τη διαχείριση των μαθημάτων ενός πανεπιστημίου και αυτή που θα χρησιμοποιηθεί στο υπόβαθρο της εφαρμογής της διπλωματικής αυτής, το eClass. Έπειτα αναφέρονται το μοντέλο σχεδιασμού και οι περιορισμοί που ακολούθησαν. Γίνεται μια μικρή εισαγωγή-ιστορική αναδρομή στις γλώσσες προγραμματισμού που επιλέχθηκαν και στη συνέχεια ακολουθεί μια αρκετά αναλυτική αναφορά στις τεχνολογίες που λάβανε μέρος. Τέλος συμπληρώθηκαν ένας μικρός οδηγός για την ορθή ρύθμιση και χρήση της εφαρμογής καθώς και τα παραρτήματα με τον τελικό κώδικα.

Κεφάλαιο 1

Πλατφόρμα Open eClass

1.1 Λίγα λόγια

Η πλατφόρμα Open eClass [2], [3] (παλαιότερα γνωστή ως GUnet eClass) είναι ένα σύστημα διαχείρισης μάθησης (LMS). Έχει σχεδιαστεί και αναπτυχθεί από την ομάδα GUnet eLearning ως λογισμικό ανοικτού κώδικα (υπό την άδεια GNU GPL). Η διανομή της πλατφόρμας ξεκίνησε τον Μάρτιο του 2003 και, από τότε, πολλοί προγραμματιστές και χρήστες έχουν συνεισφέρει μεγάλο όγκο εργασίας και ιδεών. Είναι ένα πλήρες σύστημα διαχείρισης μαθημάτων, που χρησιμοποιείται για την αποθήκευση και παρουσίαση εκπαιδευτικού υλικού. Είναι η λύση που προσφέρει το Ελληνικό Ακαδημαϊκό Δίκτυο (GUnet) για την υποστήριξη ασύγχρονων υπηρεσιών ηλεκτρονικής μάθησης. Στόχος του είναι η ενσωμάτωση και η εποικοδομητική χρήση του διαδικτύου και των διαδικτυακών τεχνολογιών στη διδακτική και μαθησιακή διαδικασία. Η εισαγωγή της ηλεκτρονικής μάθησης στην παραδοσιακή διδακτική διαδικασία παρέχει νέες δυνατότητες και επιτρέπει νέα μέσα αλληλεπίδρασης μεταξύ μαθητών και καθηγητών. Ταυτόχρονα, υποστηρίζει την ηλεκτρονική διαχείριση, αποθήκευση και παρουσίαση του διδακτικού υλικού, υπερβαίνοντας τους περιορισμούς του χώρου και του χρόνου και δημιουργώντας τις απαραίτητες προϋποθέσεις για ένα δυναμικό περιβάλλον διδασκαλίας. Η πλατφόρμα Open eClass έχει σχεδιαστεί για να παρέχει έναν εναλλακτικό τρόπο διδασκαλίας και μάθησης. Οι εκπαιδευτικοί είναι σε θέση να οργανώνουν γρήγορα πρακτικά online μαθήματα, αξιοποιώντας το υπάρχον εκπαιδευτικό υλικό: κείμενα, έγγραφα, παρουσιάσεις, εικόνες, βίντεο, ασκήσεις και ούτω καθεξής. Οι ίδιοι οι μαθητές μπορούν να έχουν πρόσβαση στο παρεχόμενο υλικό μέσω ενός εναλλακτικού καναλιού.

Εικόνα 1.1 : Open eClass logo [4]

Κεφάλαιο 2

Προσέγγιση και περιορισμοί

2.1 Αρχική ιδέα

Παρόλο που το eClass είναι μια πολύ καλή πλατφόρμα και εξυπηρετεί καθηγητές και μαθητές σε όλη την Ελλάδα, έχει κάποιες μικρές ελλείψεις, που όταν υλοποιηθούν θα την ολοκληρώσουν ακόμη περισσότερο. Μία από αυτές και αυτή που ασχολείται η παρόν διπλωματική εργασία, είναι η “άμεση” ενημέρωση των χρηστών με ειδοποίηση στις κινητές συσκευές των ενδιαφερόμενων. Πιο συγκεκριμένα το σύστημα αυτό ενημέρωσης υλοποιείται με επικοινωνία μεταξύ Server και Client. Η ιδέα ήταν να υπάρχει ένας Server που θα ελέγχει κάθε ένα λεπτό για καινούργια ανακοίνωση ή και καινούργια εργασία και θα ενημερώνει τον(-ους) ενδιαφερόμενο(-ους) Client(s). Η Server εφαρμογή θα “τρέχει” σε υπολογιστή και η Client σε Android συσκευή.

2.2 Ρόλοι της Server και της Client εφαρμογής

Η Client εφαρμογή στην ουσία επιτρέπει την είσοδο του χρήστη με τα πραγματικά ιδρυματικά στοιχεία του, μετά την επιτυχή είσοδο στέλνει στον Server (με τη βοήθεια της Web Scraping τεχνολογίας) το ονοματεπώνυμο του, τα ονόματα των μαθημάτων και τα links των μαθημάτων που είναι γραμμένος ο εν λόγω φοιτητής. Ο Server από τη μεριά του δέχεται αυτά τα στοιχεία και ενημερώνει ανάλογα τη βάση δεδομένων του (SQLAlchemy). Από εκεί και έπειτα, ο Server κάθε ένα λεπτό θα έπρεπε να ελέγχει (πάλι μέσω Web Scraping) σε όλα τα μαθήματα του πανεπιστημίου για καινούργια ανακοίνωση ή εργασία και εάν αυτές υπάρχουν ενημερώνει τη βάση δεδομένων του και στέλνει μια σχετική ειδοποίηση (με την Push Notification τεχνολογία της Google) στους χρήστες που ενδιαφέρονται (είναι γραμμένοι στο μάθημα αυτό). Ο Server και ο Client επικοινωνούν μέσω Web Services μεταξύ τους και πρέπει να είναι συνδεδεμένοι στο eClass του ίδιου ακριβώς πανεπιστημίου.

2.3 Η σύνδεση του Server με το eClass

Στον αρχικό σχεδιασμό αλλά και στην πορεία ανάπτυξης της εφαρμογής εμφανίστηκαν κάποια θέματα που έπρεπε να επιλυθούν με βάση τους περιορισμούς και τις ανάγκες. Επειδή και ο Server στην πραγματικότητα θα χρησιμοποιεί και αυτός έναν έγκυρο λογαριασμό στο eClass, το ορθό θα ήταν να μπορούσε να συνδεθεί σαν διαχειριστής συστήματος ή έστω σαν χρήστης που είναι γραμμένος σε όλα τα μαθήματα στο eClass του τμήματος ώστε έτσι το πλήθος των μαθημάτων που θα “κοιτάει” για καινούργιες ανακοινώσεις και εργασίες να καλύπτει όλα τα μαθήματα που θα είναι γραμμένοι οι χρήστες του. Αλλά αυτό δε θα μπορούσε να είναι εφικτό. Αυτό που συμβαίνει είναι ότι συνδέεται με λογαριασμό ενός φοιτητή που σίγουρα δεν είναι γραμμένος σε όλα τα μαθήματα του πανεπιστημίου, με επίπτωση στην όχι ορθή ενημέρωση των χρηστών που θα λειτουργήσουν την Client εφαρμογή. Για παράδειγμα εάν ο χρήστης που συνδεθεί στην Client εφαρμογή είναι γραμμένος στο μάθημα “Αντικειμενοστραφής προγραμματισμός” ενώ ο Server δεν είναι, τότε δε θα λάβει σχετική ενημέρωση. Αυτό συμβαίνει διότι τα περισσότερα μαθήματα στο eClass απαιτούν πρώτα εγγραφή σε αυτά και μετά μπορεί κάποιος να δει τις ανακοινώσεις και τις λοιπές πληροφορίες γι’ αυτό.

2.4 Χρήση “semaphore”

Εφόσον ο Server θα διαχειρίζεται ένα πλήθος πληροφοριών που χρειάζεται για τους χρήστες του και κάθε ένα χρονικό διάστημα (1 λεπτό) πρέπει να ελέγχει για καινούργιες ανακοινώσεις και εργασίες, έπρεπε να βρεθεί ένας τρόπος για την αποθήκευση των δεδομένων και την ευέλικτη πρόσβαση αυτών. Επιλέχθηκε γι’ αυτόν τον σκοπό η χρήση βάση δεδομένων SQL και πιο συγκεκριμένα η “SQLAlchemy” που είναι ένα εργαλείο διαχείρισης βάσης δεδομένων μέσω της γλώσσας προγραμματισμού Python. Έχοντας όμως παραπάνω από ένα σημείο μέσα στον κώδικα που καλείτε η ίδια βάση δεδομένων, πράγμα που δεν επιτρέπεται, διότι πρέπει πρώτα να τελειώσει η εκτέλεση μιας SQL εντολής και ύστερα να εκτελεστεί η επόμενη, έπρεπε να βρεθεί κάποιος τρόπος για την αποφυγή ταυτόχρονης πρόσβασης στην ίδια βάση. Επιλέχτηκε η χρήση ελέγχου πρόσβασης με μία global μεταβλητή η οποία έχει αναλάβει το ρόλο του σεμαφόρου (“semaphore”).

2.5 Αποστολή notification

Αφού ο Server διαχειριστεί και επιλέξει για ποίες ανακοινώσεις ή και εργασίες πρέπει να ενημερώσει του αντίστοιχους χρήστες του, έπρεπε να βρεθεί ένας τρόπος για την αποστολή των κατάλληλων δεδομένων στις συσκευές των Clients και ύστερα τη δημιουργία τις αντίστοιχης ειδοποίησης. Ένας περιορισμός που υπήρχε σε αυτήν την υλοποίηση ήταν ο τρόπος που θα έλεγχε η συσκευή του Client για καινούργια ενημέρωση από τον Server και αν αυτή υπήρχε, διότι εάν δεν αλλάξει κάποια κατάσταση όπως μια καινούργια ανακοίνωση σε κάποιο μάθημα, τότε ο Server δε θα στείλει τίποτα στον Client. Έπρεπε να επιλεχθεί ένας τρόπος που θα ταίριαζε περισσότερο στις πλέον καινούργιες τεχνολογίες που έχουν οι Android συσκευές και επιπρόσθετα να μην βαραίνει τη χρήση της μπαταρίας. Για τους λόγους αυτούς επιλέχθηκε η “Push Notification” τεχνολογία της “Google Firebase Cloud”. Η τεχνολογία αυτή επιτρέπει τη μεταξύ επικοινωνία του Server με τις Client συσκευές. Ο Server στην ουσία στέλνει τα δεδομένα που χρειάζεται με την κατάλληλη μορφή μέσω ενός “POST request” στην “Google Firebase Cloud”, ο Server της Google αναλαμβάνει από και έπειτα να εμφανιστεί η ειδοποίηση στην οθόνη του χρήστη.

2.6 Διαχείριση πολλαπλών αιτημάτων

Έτσι όπως είχε υλοποιηθεί ο Server αρχικά φαινόταν ότι δούλευε κανονικά, εξυπηρετώντας τους χρήστες που κάνανε τα ερωτήματα (POST, GET Requests) αλλά εάν δύο ή παραπάνω χρήστες στέλνανε έστω και υποθετικά μιλώντας ακριβώς την ίδια στιγμή ένα ερώτημα (π.χ. κάνανε login την ίδια στιγμή από τις Android συσκευές τους) τότε ένα από τα δύο ερωτήματα θα αγνοούταν. Λόγο αυτού το προβλήματος επιλέχθηκε η εκτέλεση του Server με έναν πιο “πλούσιο” τρόπο, δηλαδή να “τρέχει” μέσω του “Gunicorn” ο οποίος παρέχει μεταξύ άλλων και την επιλογή για πολλαπλές “παράλληλες” εκτελέσεις του κώδικα χρησιμοποιώντας threads. Επιλέχθηκε παρόλα αυτά η εκτέλεση μόνο ενός thread για να γίνει αποφυγή της πολλαπλής πρόσβασης της βάσης δεδομένων από διαφορετικά threads, καθώς ο τρόπος του σεμαφόρου έτσι όπως έχει υλοποιηθεί δεν μπορεί να λύσει το πρόβλημα όταν υπάρχουν threads. Και άρα τώρα οι χρήστες που κάνουν πολλαπλά ερωτήματα την ίδια ακριβώς στιγμή μπαίνουν σε μια σειρά προτεραιότητας που επιλέγει ο “Gunicorn” Server και διαδοχικά γίνεται η εξυπηρέτηση τους. Αυτό έχει σαν επίπτωση μια μικρή καθυστέρηση για τον λόγο του ότι ο τελευταίος ας πούμε χρήστης πρέπει πρώτα να περιμένει την εξυπηρέτηση των προηγούμενων χρηστών και ύστερα να εξυπηρετηθεί και αυτός.

Κεφάλαιο 3

Γλώσσες και τεχνολογίες που χρησιμοποιήθηκαν

3.1 Γλώσσες προγραμματισμού και σχεδιασμού

Έγινε επιλογή της γλώσσας Python στη μεριά του Server διότι είναι πιο ευέλικτη στο χειρισμό των τεχνολογιών που χρειάσθηκαν σε αυτήν την εργασία. Επίσης προτιμήθηκε και λόγο της έντονης χρήσης της τα τελευταία χρόνια, καθώς συνέχεια αναβαθμίζεται και μπορεί να προσφέρει σημαντικές γνώσεις και εμπειρίες στο συγγραφέα (προγραμματιστή) της. Για τη μεριά του Client επιλέχθηκε η Java (Android Studio) και σε μικρό μέρος χρησιμοποιήθηκε η xml (για την εμφάνιση της εφαρμογής). Το Android Studio έχει παραπάνω από μια επιλογές για τη γλώσσα συγγραφής κώδικα, επιλέχθηκε η γλώσσα Java καθαρά για προσωπικούς λόγους καθώς υπάρχει παραπάνω εμπειρία και άνεση σε αυτήν.

3.2 Τεχνολογίες

Για να γίνει εφικτό το έργο αυτό χρησιμοποιήθηκαν οι παρακάτω τεχνολογίες. Αρχικά χρειάστηκε να γίνει web scrapping στη κατάλληλη ιστοσελίδα του eClass από την Client εφαρμογή, αφού γίνει επιτυχής είσοδος από τον χρήστη. Χρησιμοποιήθηκαν τα “Document” και “Element” Interfaces. Ύστερα αφού εξαχθούν τα κατάλληλα δεδομένα, πρέπει να αποσταλούν στον Server. Επιλέχθηκε η Java βιβλιοθήκη “Volley” γι’ αυτό τον σκοπό, καθώς εκτός από τα άλλα πλεονεκτήματα είναι εύκολη, κατανοητή και γρήγορη. Επίσης χρειάστηκε να γίνει web scrapping και από τη μεριά του Server, για αυτό χρησιμοποιήθηκαν οι βιβλιοθήκες “requests” και “BeatifulSoup” της Python. Με τη βιβλιοθήκη “requests” γίνονται τα ερωτήματα “GET” και “POST” προς τις ιστοσελίδες και δημιουργούνται τα αντίστοιχα “session” που είναι αναγκαία κυρίως για τα “cookies”. Η βιβλιοθήκη “BeatifulSoup” χρειάσθηκε για την μορφοποίηση ενός “session” και την κατάλληλη διαχείριση και εξαγωγή των αναγκαίων δεδομένων από την ιστοσελίδα. Η αποθήκευση των δεδομένων γίνεται σε βάση δεδομένων SQL και η διαχείριση της γίνεται μέσω της “SQLAlchemy” τεχνολογίας. Επίσης, χρησιμοποιούνται δύο τεχνολογίες για την επικοινωνία Server με Client, η πρώτη είναι η “Flask”, η οποία είναι ένα web framework και επιτρέπει με τη βοήθεια ενός API της “flask_restful” βιβλιοθήκης να γίνονται ερωτήματα από τον Client προς τον Server. Από την άλλη πλευρά ο Server για να στείλει την κατάλληλη ειδοποίηση προς τον Client χρησιμοποιεί την “Push Notification” τεχνολογία της “Google Firebase Cloud”. Για την επιθυμητή λειτουργία του Server χρησιμοποιείται ο “Gunicorn” ο οποίος είναι ένας WSGI (Web Server Gateway Interface) HTTP Server της Pyhton και είναι αναγκαίος για τη διαχείριση με ευέλικτο τρόπο πολλών χρηστών (“ταυτόχρονα” πολλαπλά ερωτήματα προς τον Server). Τέλος χρησιμοποιείται η “BackgroundScheduler” βιβλιοθήκη η οποία επιτρέπει την απαραίτητη εκτέλεση κάποιων συναρτήσεων του κώδικα ανά ένα χρονικό διάστημα (για να γίνονται οι απαραίτητες ενημερώσεις στη βάση δεδομένων).

3.3 Android Studio

Το Android Studio [5] είναι το επίσημο ολοκληρωμένο περιβάλλον ανάπτυξης (IDE) για το λειτουργικό σύστημα Android της Google, το οποίο βασίζεται στο λογισμικό IntelliJ IDEA της JetBrains και έχει σχεδιαστεί ειδικά για την ανάπτυξη Android. Είναι διαθέσιμο για λήψη σε λειτουργικά συστήματα που βασίζονται σε Windows, macOS και Linux. Αποτελεί αντικαταστάτη του Eclipse Android Development Tools (E-ADT) ως το κύριο IDE για την ανάπτυξη εγγενών εφαρμογών Android. Το Android Studio ανακοινώθηκε στις 16 Μαΐου 2013, στο συνέδριο Google I/O. Βρισκόταν σε στάδιο προεπισκόπησης πρώιμης πρόσβασης ξεκινώντας από την έκδοση 0.1 τον Μάιο του 2013, ενώ στη συνέχεια εισήλθε σε στάδιο beta ξεκινώντας από την έκδοση 0.8 η οποία κυκλοφόρησε τον Ιούνιο του 2014. Το πρώτο σταθερό build κυκλοφόρησε τον Δεκέμβριο του 2014, ξεκινώντας από την έκδοση 1.0. Στις 7 Μαΐου 2019, η Kotlin αντικατέστησε τη Java ως προτιμώμενη γλώσσα της Google για την ανάπτυξη εφαρμογών Android. Η Java εξακολουθεί να υποστηρίζεται, όπως και η C++. Το Android Studio υποστηρίζει όλες τις ίδιες γλώσσες προγραμματισμού του IntelliJ (και του CLion), π.χ. Java, C++ και άλλες με επεκτάσεις (extensions), όπως, επίσης το Android Studio 3.0 ή νεότερη έκδοση υποστηρίζει την Kotlin και "όλα τα χαρακτηριστικά της γλώσσας Java 7 και ένα υποσύνολο των χαρακτηριστικών της γλώσσας Java 8 που διαφέρουν ανάλογα με την έκδοση της πλατφόρμας". Τα εξωτερικά έργα υποστηρίζουν ορισμένα χαρακτηριστικά της Java 9. Ενώ η IntelliJ δηλώνει ότι το Android Studio υποστηρίζει όλες τις εκδόσεις Java που έχουν κυκλοφορήσει και τη Java 12, δεν είναι σαφές σε ποιο επίπεδο το Android Studio υποστηρίζει εκδόσεις Java μέχρι τη Java 12 (η τεκμηρίωση αναφέρει μερική υποστήριξη της Java 8). Τουλάχιστον ορισμένα νέα χαρακτηριστικά της γλώσσας μέχρι και την Java 12 είναι χρησιμοποιήσιμα στο Android.

Εικόνα 3.3 : Android Studio logo [6]

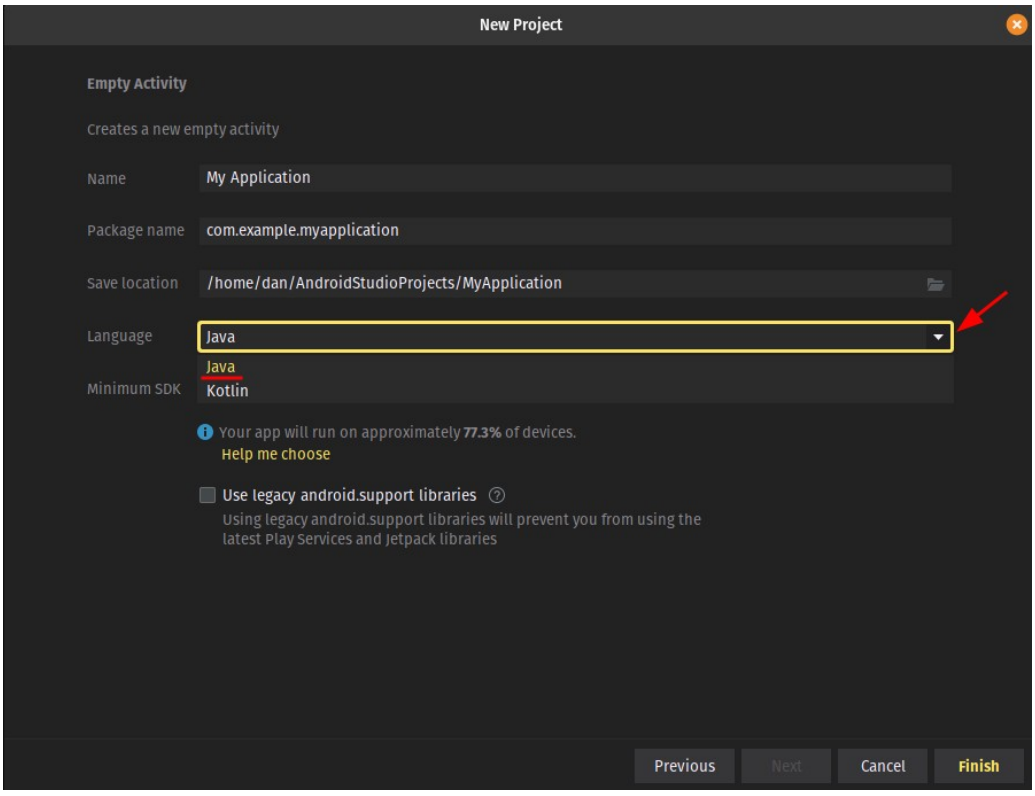
3.4 Python

Η Python είναι μια υψηλού επιπέδου γλώσσα προγραμματισμού γενικού σκοπού. Η φιλοσοφία σχεδιασμού της δίνει έμφαση στην αναγνωσιμότητα του κώδικα με τη χρήση σημαντικών εσοχών. Η Python είναι δυναμικά τυποποιημένη και συλλέχτης-σκουπιδιών (garbage-collected). Υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, συμπεριλαμβανομένων του δομημένου (ιδιαίτερα διαδικαστικού), του αντικειμενοστραφούς και του λειτουργικού προγραμματισμού. Συχνά περιγράφεται ως μια γλώσσα που περιλαμβάνει "μπαταρίες" λόγω της περιεκτικής τυποποιημένης βιβλιοθήκης της. Ο Guido van Rossum άρχισε να εργάζεται πάνω στην Python στα τέλη της δεκαετίας του 1980 ως διάδοχος της γλώσσας προγραμματισμού ABC και την κυκλοφόρησε για πρώτη φορά το 1991 ως Python 0.9.0. Η Python 2.0 κυκλοφόρησε το 2000 και εισήγαγε νέα χαρακτηριστικά όπως η κατανόηση λιστών, η συλλογή σκουπιδιών με ανίχνευση κύκλου, η μέτρηση αναφορών και η υποστήριξη Unicode. Η Python 3.0, που κυκλοφόρησε το 2008, ήταν μια σημαντική αναθεώρηση που δεν είναι πλήρως συμβατή με τις προηγούμενες εκδόσεις. Η Python 2 διακόπηκε με την έκδοση 2.7.18 το 2020. Η Python κατατάσσεται σταθερά ως μία από τις πιο δημοφιλείς γλώσσες προγραμματισμού. Προορίζεται να είναι μια εύκολα αναγνώσιμη γλώσσα. Η μορφοποίησή της είναι οπτικά ξεκάθαρη και συχνά χρησιμοποιεί αγγλικές λέξεις-κλειδιά, όπου άλλες γλώσσες χρησιμοποιούν σημεία στίξης. Σε αντίθεση με πολλές άλλες γλώσσες, δεν χρησιμοποιεί “κουρδιστές” αγκύλες για να οριοθετεί τα μπλοκ, και οι άνω και κάτω τελεία μετά τις εντολές επιτρέπονται αλλά χρησιμοποιούνται σπάνια. Έχει λιγότερες συντακτικές εξαιρέσεις και ειδικές περιπτώσεις από τη C ή τη Pascal [7].

Εικόνα 3.4 : Ο σχεδιαστής της Python, Guido van Rossum [8]

3.5 Java

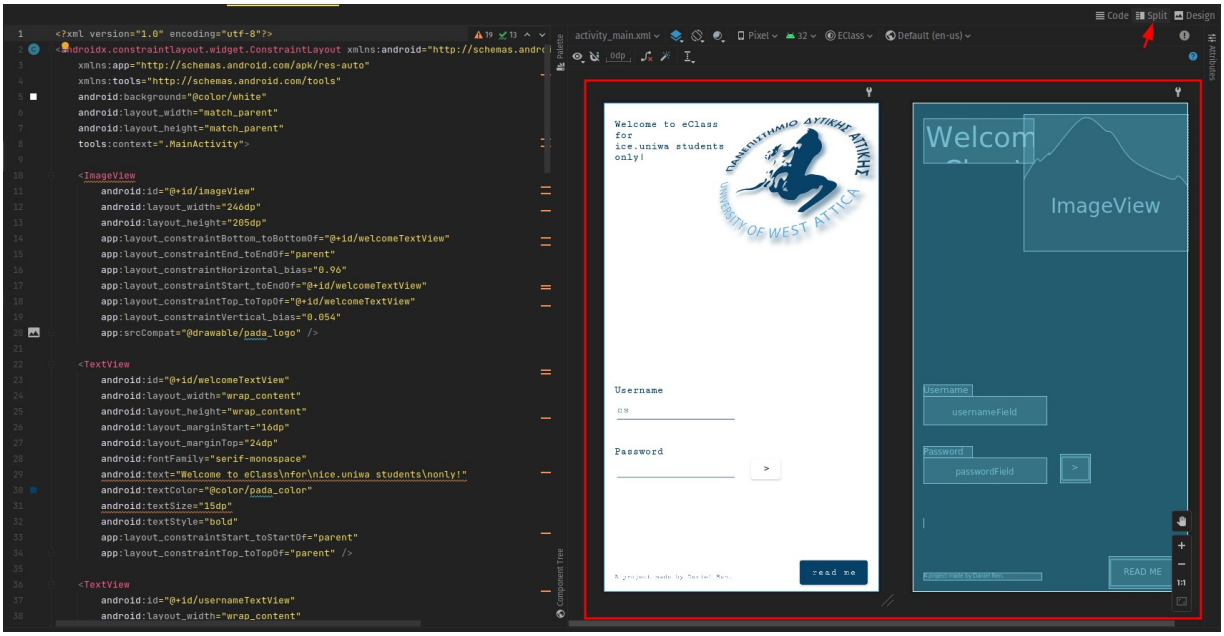
Η Java [9] είναι μια γλώσσα προγραμματισμού και μια υπολογιστική πλατφόρμα που κυκλοφόρησε για πρώτη φορά από τη Sun Microsystems το 1995. Από τις ταπεινές της απαρχές εξελίχθηκε και τροφοδοτεί ένα μεγάλο μέρος του σημερινού ψηφιακού κόσμου, παρέχοντας την αξιόπιστη πλατφόρμα πάνω στην οποία βασίζονται πολλές υπηρεσίες και εφαρμογές. Νέα, καινοτόμα προϊόντα και ψηφιακές υπηρεσίες που έχουν σχεδιαστεί για το μέλλον συνεχίζουν να βασίζονται επίσης στη Java. Ενώ οι περισσότερες σύγχρονες εφαρμογές Java συνδυάζουν το χρόνο εκτέλεσης της Java και την εφαρμογή μαζί, εξακολουθούν να υπάρχουν πολλές εφαρμογές και ακόμη και ορισμένοι ιστότοποι που δεν θα λειτουργήσουν αν δεν είναι εγκαταστημένη η Java. Οι τελευταίες επιδιορθώσεις Java περιέχουν σημαντικές βελτιώσεις για τη βελτίωση της απόδοσης, της σταθερότητας και της ασφάλειας των εφαρμογών Java που εκτελούνται στον υπολογιστή σας. Η εγκατάσταση αυτών των ενημερώσεων θα διασφαλίσει ότι οι εφαρμογές Java σας θα συνεχίσουν να εκτελούνται με την πιο ενημερωμένη έκδοση. Στην περίπτωση της διπλωματικής αυτής έρευνας δε χρειάστηκε να εγκατασταθεί χειροκίνητα η Java ή κάποιο άλλο βοηθητικό πρόγραμμα της. Με την εγκατάσταση του Android Studio (που χρησιμοποιήθηκε) συμπεριλαμβάνεται εκτός των άλλων και η γλώσσα προγραμματισμού Java.



Εικόνα 3.5 : Επιλογή της γλώσσας προγραμματισμού Java σε ένα καινούργιο project

3.6 XML

Αν και η χρήση της ήταν ελάχιστη διότι το Android Studio παρέχει τη δυνατότητα στον προγραμματιστή της διαχείρισης της γλώσσας αυτής με εύκολο και εικονικό τρόπο, αξίζει να αναφερθούν λίγα λόγια. Η Extensible Markup Language (XML) [10] είναι μια γλώσσα σήμανσης και μια μορφή αρχείου για την αποθήκευση, τη μετάδοση και την ανακατασκευή αυθαίρετων δεδομένων. Ορίζει ένα σύνολο κανόνων για την κωδικοποίηση εγγράφων σε μορφή που είναι τόσο αναγνώσιμη από τον άνθρωπο όσο και από μηχανήματα. Η προδιαγραφή XML 1.0 της Κοινοπραξίας του Παγκόσμιου Ιστού του 1998 και διάφορες άλλες σχετικές προδιαγραφές (όλες τους ελεύθερα ανοικτά πρότυπα) ορίζουν την XML. Οι σχεδιαστικοί στόχοι της XML δίνουν έμφαση στην απλότητα, τη γενικότητα και τη χρηστικότητα στο Διαδίκτυο. Είναι μια μορφή δεδομένων κειμένου με ισχυρή υποστήριξη μέσω Unicode για διάφορες ανθρώπινες γλώσσες. Αν και ο σχεδιασμός της XML επικεντρώνεται στα έγγραφα, η γλώσσα χρησιμοποιείται ευρέως για την αναπαράσταση αυθαίρετων δομών δεδομένων, όπως αυτές που χρησιμοποιούνται στις υπηρεσίες ιστού. Υπάρχουν διάφορα συστήματα σχημάτων που βοηθούν στον ορισμό γλωσσών που βασίζονται στην XML, ενώ οι προγραμματιστές έχουν αναπτύξει πολλές διεπαφές προγραμματισμού εφαρμογών (API) για να βοηθήσουν την επεξεργασία δεδομένων XML.



Εικόνα 3.6 : Το εικονικό περιβάλλον διαχείρισης XML κώδικα στο Android Studio

Κεφάλαιο 4

Βάση δεδομένων

Ο Server για την αποθήκευση των δεδομένων χρησιμοποιεί βάση δεδομένων και την ενημερώνει αναλόγως. Ενημέρωση γίνεται όταν η Client εφαρμογή στείλει τα στοιχεία του χρήστη στην Server εφαρμογή και όταν βρεθεί καινούργια ανακοίνωση ή και εργασία στον περιοδικό έλεγχο που κάνει ο Server κάθε ένα λεπτό. Η βάση δεδομένων διαχειρίζεται με τη βοήθεια της SQLAlchemy βιβλιοθήκης της Python.

4.1 Εισαγωγή στην SQLAlchemy

Η SQLAlchemy είναι μια δημοφιλής εργαλειοθήκη SQL. Είναι γραμμένη σε Python και παρέχει πλήρη ισχύ και ευελιξία της SQL σε έναν προγραμματιστή εφαρμογών [11]. Είναι ένα λογισμικό ανοικτού κώδικα που κυκλοφορεί με άδεια MIT. Η SQLAlchemy είναι διάσημη για τον αντικειμενο-σχεσιακό χαρτογράφο της (ORM), χρησιμοποιώντας τον οι κλάσεις μπορούν να αντιστοιχιστούν στη βάση δεδομένων, επιτρέποντας έτσι στο μοντέλο αντικειμένων και στο σχήμα της βάσης δεδομένων να αναπτυχθεί με καθαρά αποσυνδεδεμένο τρόπο από την αρχή. Ο πυρήνας του SQLAlchemy περιλαμβάνει μηχανή απόδοσης SQL, ενσωμάτωση DBAPI, συναλλαγή και υπηρεσίες περιγραφής σχήματος. Ο πυρήνας SQLAlchemy χρησιμοποιεί SQL Expression Language. Η SQL Expression Language παρουσιάζει ένα σύστημα αναπαράστασης σχεσιακών βάσεων δεδομένων, δομών και εκφράσεων χρησιμοποιώντας Python constructs.

4.2 Τι είναι το ORM

Το ORM (Object Relational Mapping) [11] είναι μια τεχνική προγραμματισμού για τη μετατροπή δεδομένων μεταξύ ασυμβίβαστων τύπων συστημάτων, σε αντικειμενοστραφείς γλώσσες προγραμματισμού. Συνήθως, ο τύπος που χρησιμοποιείται σε μια αντικειμενοστραφή γλώσσα όπως η Python περιέχει μη κλιμακωτούς τύπους. Αυτοί δεν μπορούν να εκφραστούν ως πρωτόγονοι τύποι, όπως οι ακέραιοι και οι συμβολοσειρές. Ως εκ τούτου, ο προγραμματιστής πρέπει να μετατρέψει τα αντικείμενα σε κλιμακωτά δεδομένα για να αλληλεπιδράσει με τη βάση δεδομένων. Ωστόσο, οι τύποι δεδομένων στα περισσότερα προϊόντα βάσεων δεδομένων, όπως η Oracle, η MySQL κ.λπ. είναι πρωτογενείς. Σε ένα σύστημα ORM, κάθε κλάση αντιστοιχίζεται σε έναν πίνακα στην βάση δεδομένων. Δίνοντας έτσι στον προγραμματιστή την ευκολία να μη γράφει “κουραστικό” κώδικα, αλλά να επικεντρωθεί στη λογική του συστήματος.

4.3 SQLAlchemy και DBAPI

Η SQLAlchemy έχει σχεδιαστεί για να λειτουργεί με μια υλοποίηση DBAPI που έχει κατασκευαστεί για μια συγκεκριμένη βάση δεδομένων. Χρησιμοποιεί σύστημα διαλέκτων (dialect system) για την επικοινωνία με διάφορους τύπους DBAPI υλοποιήσεις και βάσεις δεδομένων. Όλες οι διάλεκτοι απαιτούν να υπάρχει ένας κατάλληλος οδηγός DBAPI εγκατεστημένος [11].

Περιλαμβάνονται οι ακόλουθοι διάλεκτοι :

- Firebird
- Microsoft SQL Server
- MySQL
- Oracle
- PostgreSQL
- SQLite
- Sybase

4.4 Εγκατάσταση SQLAlchemy στην Python

Οποιαδήποτε έκδοση της Python υψηλότερη από την 2.7 είναι απαραίτητη για την εγκατάσταση της SQLAlchemy [11]. Ο ευκολότερος τρόπος για την εγκατάσταση είναι με τη χρήση του Python Package Manager, pip. Αυτό το βοηθητικό πρόγραμμα περιλαμβάνεται στο πρότυπο διανομής της Python (δλδ. είναι ήδη εγκαταστημένο με την python). Χρησιμοποιώντας την εντολή “pip install SQLAlchemy” γίνεται η εγκατάσταση της τελευταίας έκδοσης της SQLAlchemy στο σύστημα. Σε περίπτωση που κάποιος χρησιμοποιεί anaconda distribution of Python, η SQLAlchemy μπορεί να εγκατασταθεί “τρέχοντας” την εντολή “conda install -c anaconda SQLAlchemy” στο τερματικό conda.

Εικόνα 4.4 : Έλεγχος ότι η SQLAlchemy είναι επιτυχώς εγκαταστημένη και η έκδοση της

4.5 Textual SQL

Μεταξύ των διαφορετικών format που υποστίζονται, η SQLAlchemy επιτρέπει τη χρήση απλών συμβολοσειρών [12], για τις περιπτώσεις που η SQL εντολή είναι ήδη γνωστή και δεν υπάρχει έντονη ανάγκη να υποστηρίζει δυναμικά χαρακτηριστικά. Ο κατασκευαστής text() χρησιμοποιείται για τη σύνθεση μιας δήλωσης κειμένου που περνάει στη βάση δεδομένων ως επί το πλείστον αμετάβλητη.

Τα πλεονεκτήματα που παρέχει η text() σε σχέση με μια απλή συμβολοσειρά είναι :

- backend-ουδέτερη υποστήριξη για παραμέτρους δέσμευσης
- επιλογές εκτέλεσης ανά εντολή
- συμπεριφορά δακτυλογράφησης στη στήλη αποτελεσμάτων

Η συνάρτηση text() απαιτεί δεσμευμένες παραμέτρους σε μορφή ονομαστικής άνω και κάτω τελείας. Αυτές είναι ανεξάρτητες από το backend της βάσης δεδομένων.

Η μέθοδος execute() επιτρέπει να γίνει η αντιστοίχηση των τιμών που ο συγγραφέας ορίζει περνώντας τες ως πρόσθετα ορίσματα.

```
from sqlalchemy.sql import text

s=text("select students.name, students.lastname from students where
students.name between :x and :y")

conn.execute(s, x='A', y='L').fetchall()
```

Εικόνα 4.5 : Παράδειγμα χρήσης Textual SQL format

4.6 Πίνακες της βάσης δεδομένων

Επιλέχθηκαν οι εξής πίνακες με βάση τις απαιτήσεις του σχεδιασμού :

“studentInfoTable”, περιέχει 3 κελιά :

- “id” : Το username εισόδου του χρήστη, το οποίο είναι και το primary key.
- “fullName” : Ονοματεπώνυμο χρήστη χωρισμένα με τον χαρακτήρα space (‘ ‘) ανάμεσα τους.
- “deviceToken” : Θα περιέχει το id της συσκευής του χρήστη που χρειάζεται ο Firebase Cloud Messaging Server της Google για να ξέρει σε ποια συσκευή θα στείλει το notification που μελλοντικά θα του ετοιμάσει ο Server.

“lessonsLinksTabel”, περιέχει 3 κελιά :

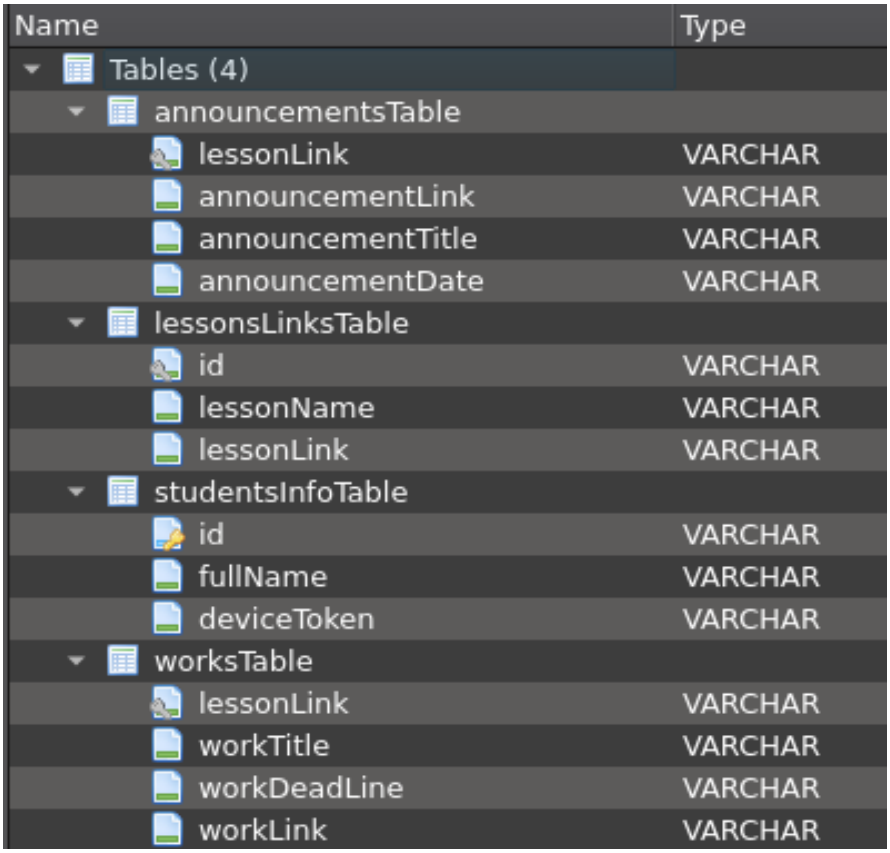
- “id” : Το οποίο είναι foreign key, και αναφέρετε στο primary key του πίνακα “studentInfoTable”.
- “lessonName” : Ο τίτλος του μαθήματος θα περιέχεται σε αυτό το κελί.
- “lessonLink” : Θα αντιστοιχεί με το link του μαθήματος (το οποίο θα είναι και μοναδικό για κάθε μάθημα, ενώ ο τίτλος μπορεί να είναι ίδιος σε κάποια μαθήματα)

“announcementsTable”, περιέχει 4 κελιά :

- “lessonLink” : Το οποίο είναι foreign key, και αναφέρετε στο κελί “lessonLink” του πίνακα “lessonsLinksTabel”.
- “announcementLink” : Θα αντιστοιχεί στο link της μιας ανακοίνωσης ενός μαθήματος (το οποίο θα είναι και μοναδικό)
- “announcementTitle" : Θα αντιστοιχεί με το link του μαθήματος (το οποίο θα είναι και μοναδικό για κάθε μάθημα, ενώ ο τίτλος μπορεί να είναι ίδιος σε κάποια μαθήματα)
- “announcementDate” : Θα περιέχει την ημερομηνία που δημιουργήθηκε η ανακοίνωση.

“worksTable”, περιέχει 4 κελιά :

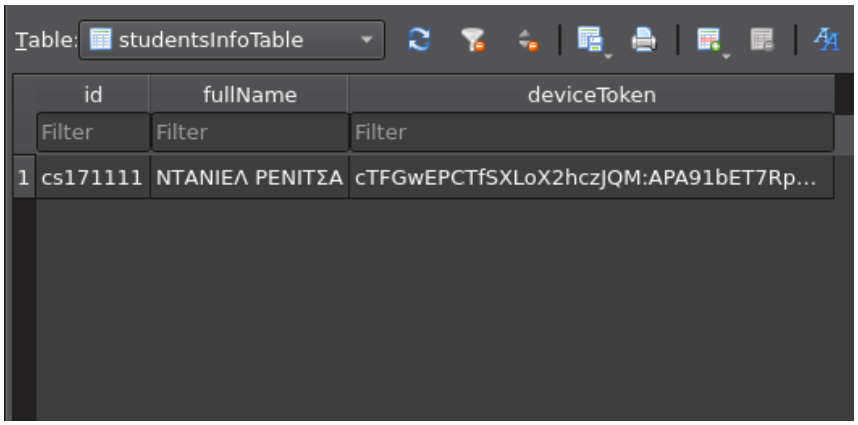
- “lessonLink” : Το οποίο είναι foreign key, και αναφέρετε στο κελί “lessonLink” του πίνακα “lessonsLinksTabel”.
- “workTitle” : Τίτλος της εργασίας.
- “workDeadLine” : Η ακριβή ημερομηνία λήξης της εργασίας (με όλα τα στοιχεία από μέρα μέχρι λεπτά και δευτερόλεπτα).
- “workLink” : Το link της εργασίας (το οποίο θα είναι και μοναδικό).



Εικόνα 4.6 : Πίνακες της βάσης δεδομένων με τα αντίστοιχα κελιά

4.7 Υποδειγματική είσοδος χρήστη και αντίστοιχη ενημέρωση των πινάκων

Οι παρακάτω εικόνες μας δείχνουν το περιεχόμενο των πινάκων μετά την επιτυχή είσοδο ενός χρήστη.



Εικόνα 4.7.1 : Ο πίνακας studentInfoTable

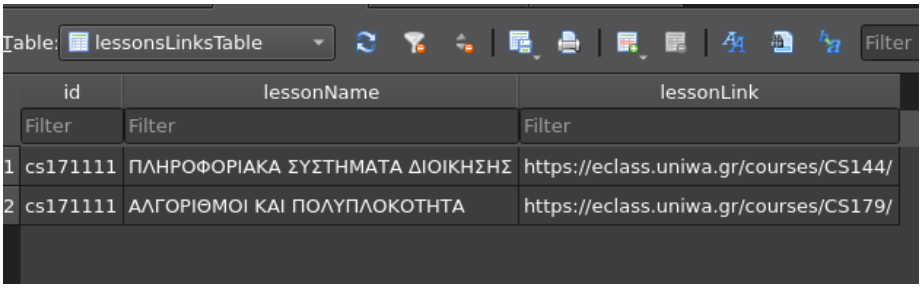


Table: lessonsLinksTable

	id	lessonName	lessonLink
	Filter	Filter	Filter
1	cs171111	ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ΔΙΟΙΚΗΣΗΣ	https://eclass.uniwa.gr/courses/CS144/
2	cs171111	ΑΛΓΟΡΙΘΜΟΙ ΚΑΙ ΠΟΛΥΠΛΟΚΟΤΗΤΑ	https://eclass.uniwa.gr/courses/CS179/

Εικόνα 4.7.2 : Ο πίνακας lessonsLinksTable

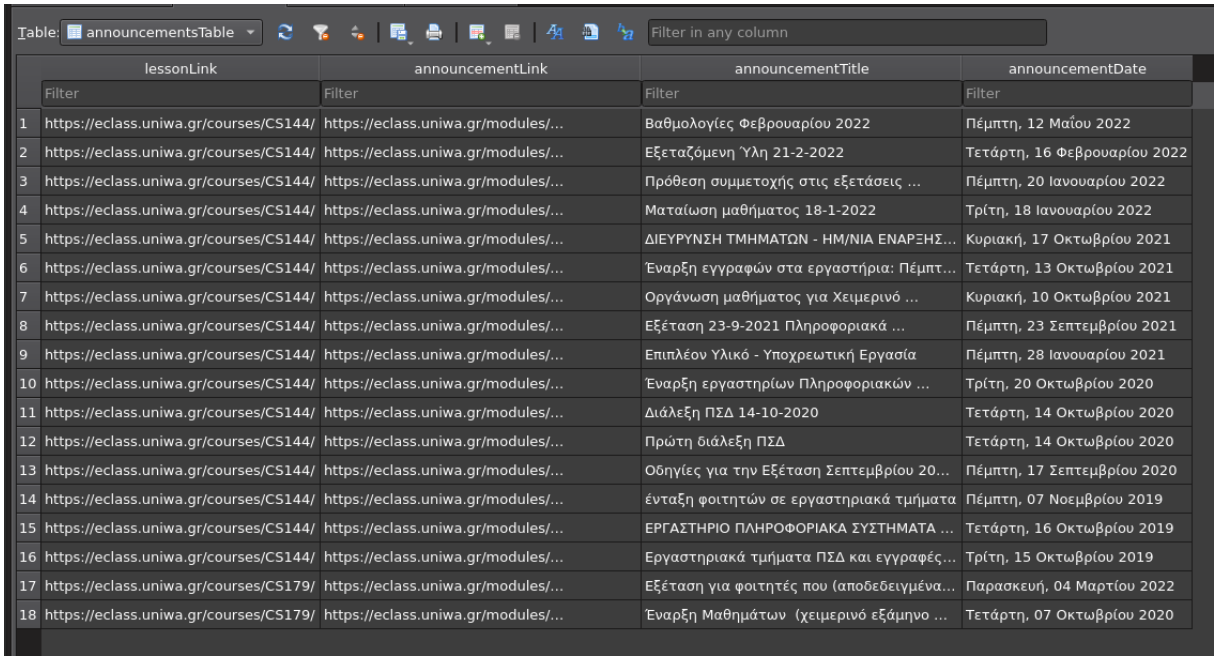


Table: announcementsTable

	lessonLink	announcementLink	announcementTitle	announcementDate
	Filter	Filter	Filter	Filter
1	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Βαθμολογίες Φεβρουαρίου 2022	Πέμπτη, 12 Μαΐου 2022
2	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Εξεταζόμενη Ύλη 21-2-2022	Τετάρτη, 16 Φεβρουαρίου 2022
3	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Πρόθεση συμμετοχής στις εξετάσεις ...	Πέμπτη, 20 Ιανουαρίου 2022
4	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Ματαίωση μαθήματος 18-1-2022	Τρίτη, 18 Ιανουαρίου 2022
5	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	ΔΙΕΥΡΥΝΣΗ ΤΜΗΜΑΤΩΝ - ΗΜ/ΝΙΑ ΕΝΑΡΞΗΣ...	Κυριακή, 17 Οκτωβρίου 2021
6	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Έναρξη εγγραφών στα εργαστήρια: Πέμπτ...	Τετάρτη, 13 Οκτωβρίου 2021
7	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Οργάνωση μαθήματος για Χειμερινό ...	Κυριακή, 10 Οκτωβρίου 2021
8	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Εξέταση 23-9-2021 Πληροφοριακά ...	Πέμπτη, 23 Σεπτεμβρίου 2021
9	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Επιπλέον Υλικό - Υποχρεωτική Εργασία	Πέμπτη, 28 Ιανουαρίου 2021
10	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Έναρξη εργαστηρίων Πληροφοριακών ...	Τρίτη, 20 Οκτωβρίου 2020
11	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Διάλεξη ΠΣΔ 14-10-2020	Τετάρτη, 14 Οκτωβρίου 2020
12	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Πρώτη διάλεξη ΠΣΔ	Τετάρτη, 14 Οκτωβρίου 2020
13	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Οδηγίες για την Εξέταση Σεπτεμβρίου 20...	Πέμπτη, 17 Σεπτεμβρίου 2020
14	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	ένταξη φοιτητών σε εργαστηριακά τμήματα	Πέμπτη, 07 Νοεμβρίου 2019
15	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	ΕΡΓΑΣΤΗΡΙΟ ΠΛΗΡΟΦΟΡΙΑΚΑ ΣΥΣΤΗΜΑΤΑ ...	Τετάρτη, 16 Οκτωβρίου 2019
16	https://eclass.uniwa.gr/courses/CS144/	https://eclass.uniwa.gr/modules/...	Εργαστηριακά τμήματα ΠΣΔ και εγγραφές...	Τρίτη, 15 Οκτωβρίου 2019
17	https://eclass.uniwa.gr/courses/CS179/	https://eclass.uniwa.gr/modules/...	Εξέταση για φοιτητές που (αποδεδειγμένα...	Παρασκευή, 04 Μαρτίου 2022
18	https://eclass.uniwa.gr/courses/CS179/	https://eclass.uniwa.gr/modules/...	Έναρξη Μαθημάτων (χειμερινό εξάμηνο ...	Τετάρτη, 07 Οκτωβρίου 2020

Εικόνα 4.7.3 : Ο πίνακας announcementsTable

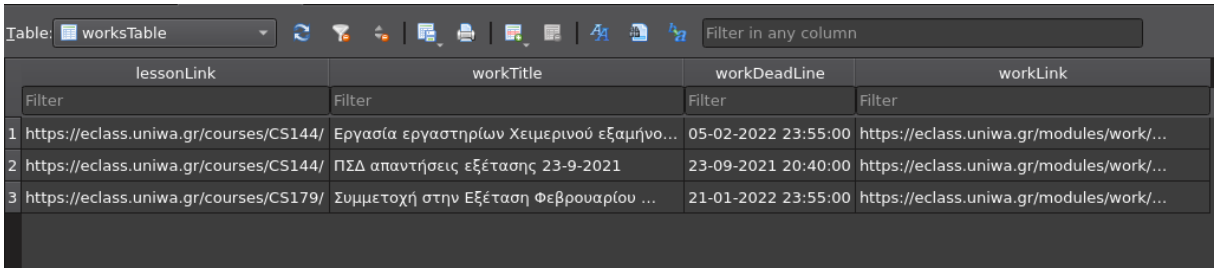


Table: worksTable

	lessonLink	workTitle	workDeadLine	workLink
	Filter	Filter	Filter	Filter
1	https://eclass.uniwa.gr/courses/CS144/	Εργασία εργαστηρίων Χειμερινού εξαμήνο...	05-02-2022 23:55:00	https://eclass.uniwa.gr/modules/work/...
2	https://eclass.uniwa.gr/courses/CS144/	ΠΣΔ απαντήσεις εξέτασης 23-9-2021	23-09-2021 20:40:00	https://eclass.uniwa.gr/modules/work/...
3	https://eclass.uniwa.gr/courses/CS179/	Συμμετοχή στην Εξέταση Φεβρουαρίου ...	21-01-2022 23:55:00	https://eclass.uniwa.gr/modules/work/...

Εικόνα 4.7.4 : Ο πίνακας worksTable

Κεφάλαιο 5

Firebase Cloud Messaging

Όταν ο Server βρει καινούργια ανακοίνωση ή εργασία ετοιμάζει να στείλει μια ειδοποίηση στην ενδιαφερόμενη Client εφαρμογή. Αυτό έγινε εφικτό με την Google Firebase Cloud Messaging τεχνολογία.

5.1 Εισαγωγή στο FCM

Το Google Firebase Cloud Messaging (FCM) παρέχει μια αξιόπιστη και αποδοτική σύνδεση μεταξύ διακομιστή και συσκευής, η οποία επιτρέπει να παράδοση και να τη λήψη μηνυμάτων και ειδοποιήσεων σε iOS, Android και στο διαδίκτυο χωρίς κόστος, ένα μήνυμα μπορεί να μεταφέρει ωφέλιμο φορτίο έως και 4000 bytes σε μια εφαρμογή-πελάτη. Επιτρέπεται η αποστολή μηνυμάτων μέσω του Firebase Admin SDK ή των πρωτοκόλλων του διακομιστή FCM [13].

5.2 Πως δουλεύει το FCM

Μια εφαρμογή FCM περιλαμβάνει δύο κύρια στοιχεία για την αποστολή και τη λήψη [13] :

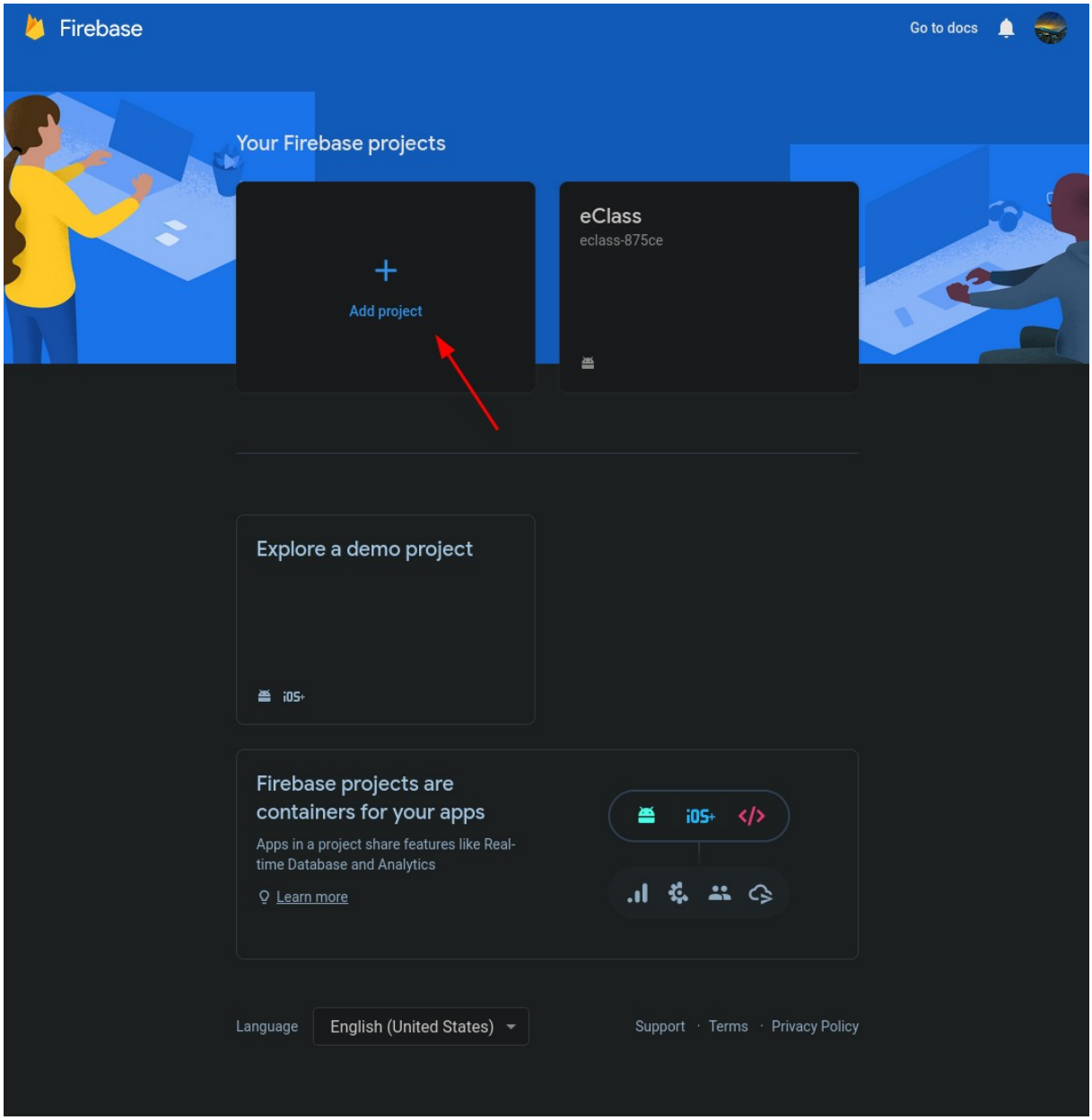
1. Ένα αξιόπιστο περιβάλλον, όπως το Cloud Functions for Firebase ή ένας διακομιστής εφαρμογών του οποίου επιτρέπεται η δημιουργία, αποστολή και λήψη μηνυμάτων.
2. Μια εφαρμογή Apple, Android ή του διαδικτύου (JavaScript) που λαμβάνει μηνύματα μέσω της αντίστοιχης υπηρεσίας μεταφοράς της συγκεκριμένης πλατφόρμας.

Εικόνα 5.2 : Firebase Cloud messaging logo [13]

5.3 Ενδεικτικά βήματα για τη δημιουργία ενός project στο FCM

Βήμα 1ο.

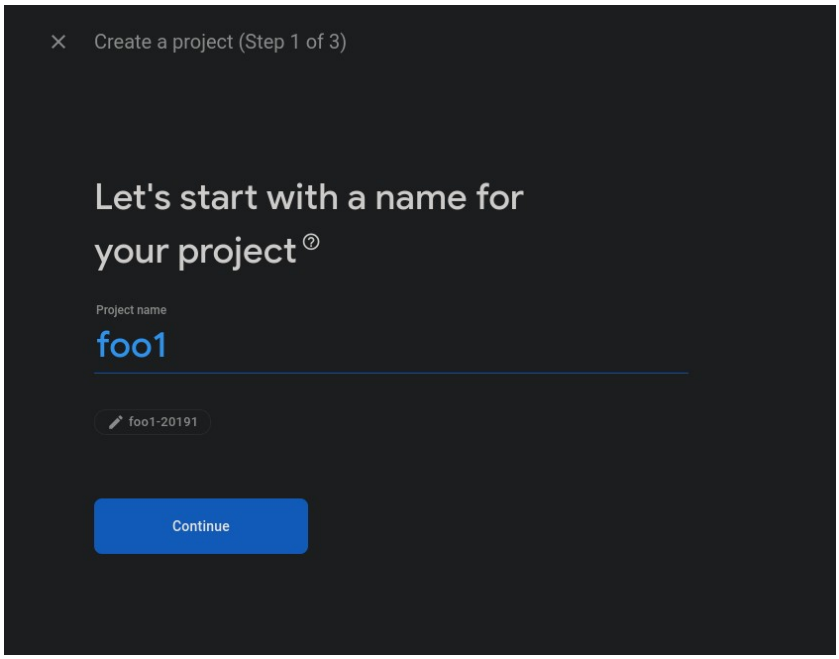
Αφού γίνει η σύνδεση με τα gmail στοιχεία του χρήστη στο σύνδεσμο “https://console.Firebase.Google.com/?pli=1”, τότε επιλέγεται η προσθήκη-δημιουργία νέου project.



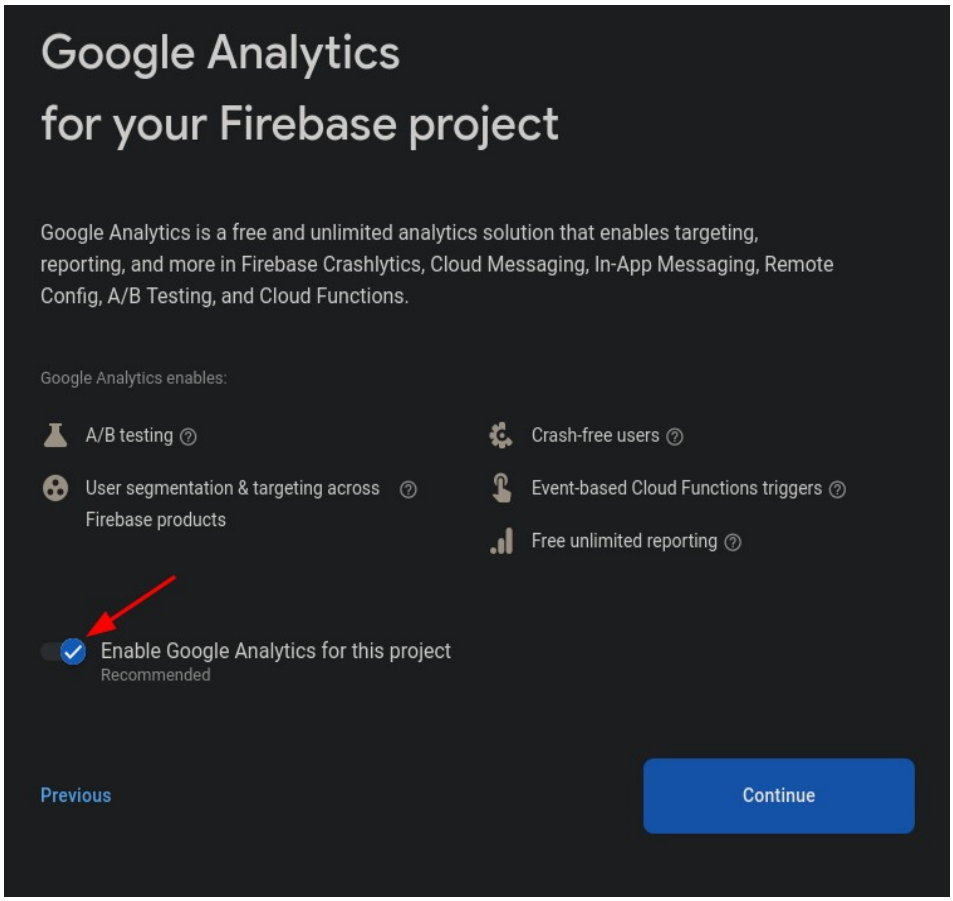
Εικόνα 5.3.1 : Δημιουργία project

Βήμα 2ο + 3ο.

Δίνεται το όνομα του project και βεβαιώνεται ότι το “Google Analytics” είναι ενεργοποιημένο.



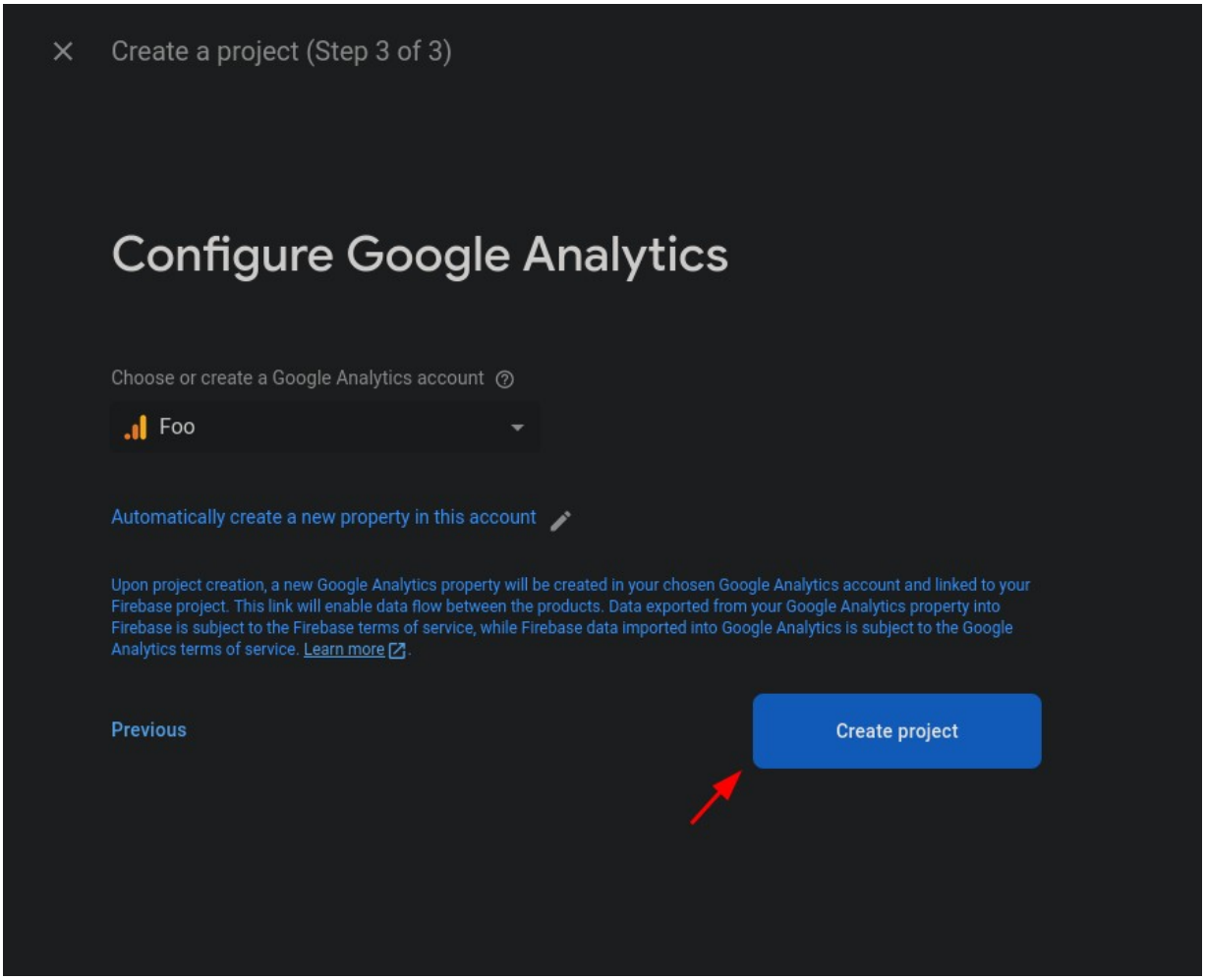
Εικόνα 5.3.2 : Ονομασία project



Εικόνα 5.3.3 : Ενεργοποιημένο το Google Analytics

Βήμα 4ο.

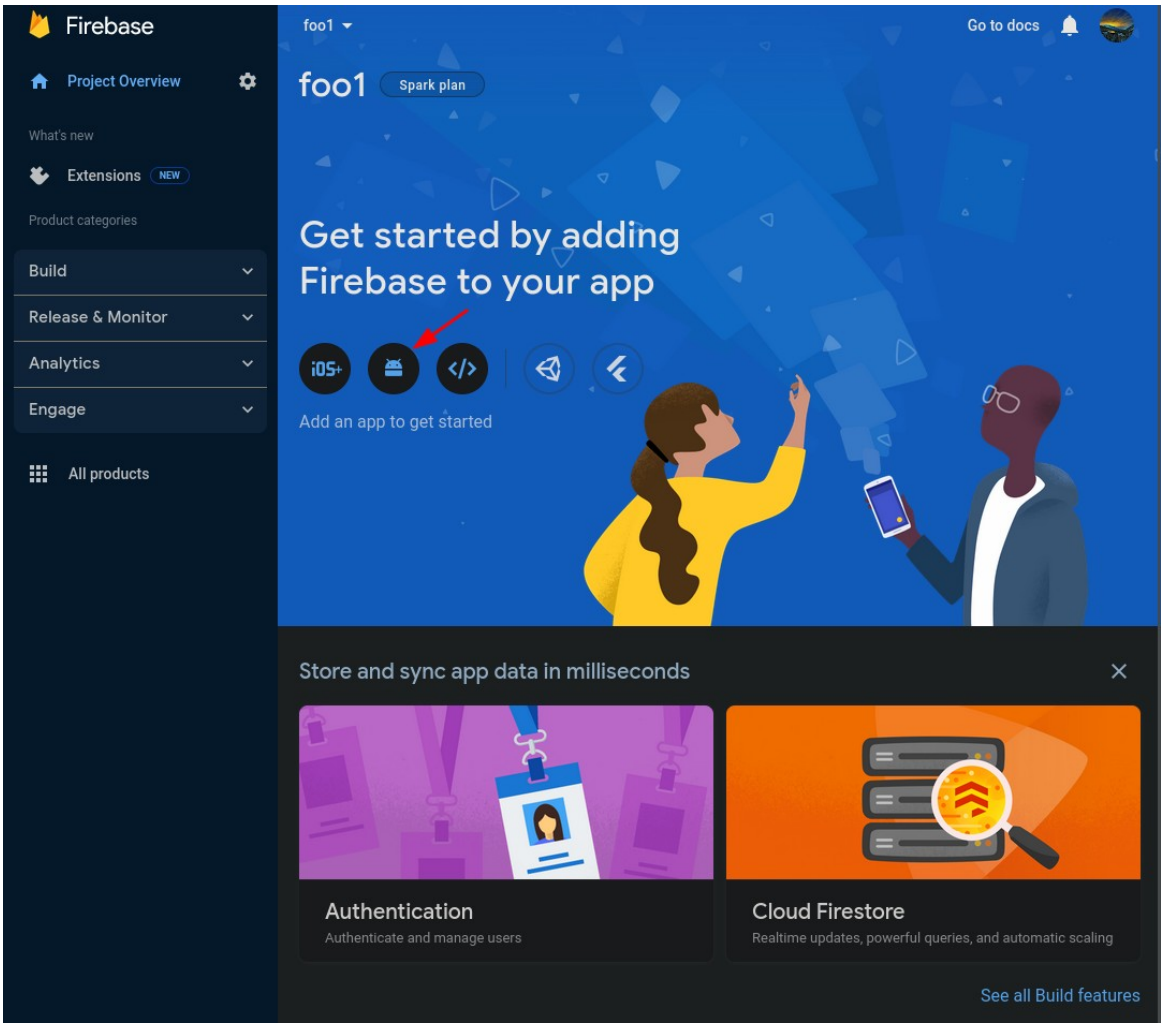
Επιλέγεται ένα προφίλ για τα Google Analytics, εάν δεν υπάρχει υπάρχει η δυνατότητα δημιουργίας νέου προφίλ. Είναι σημαντικό να είναι ενεργοποιημένο το Google Analytics αλλιώς δε θα δουλέψει σωστά η τεχνολογία.



Εικόνα 5.3.4 : Επιλογή ή δημιουργία Google Analytics λογαριασμού και δημιουργία project

Βήμα 5ο.

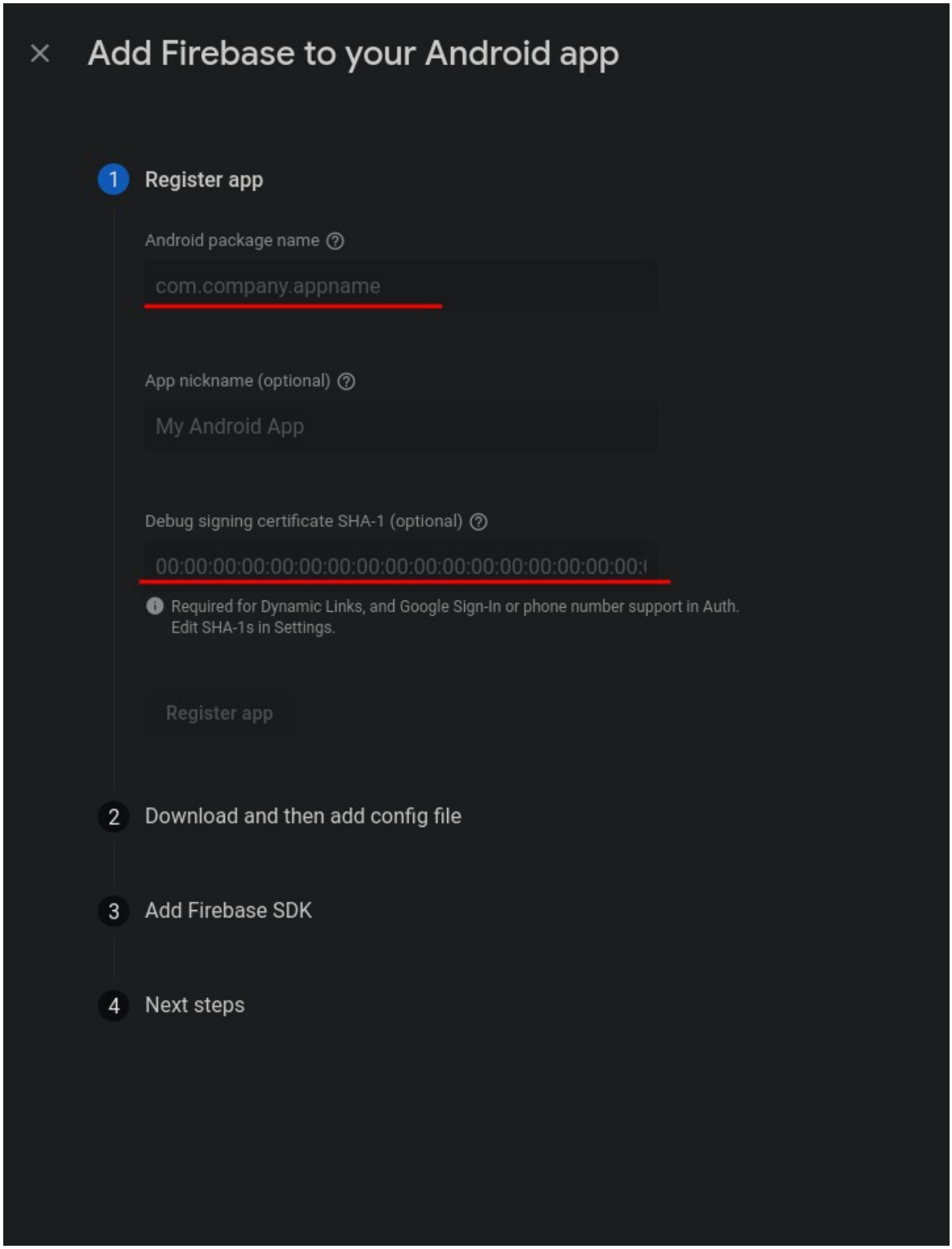
Αφού δημιουργηθεί επιτυχώς το project ο χρήστης μπορεί πια να πάει σε αυτό και να διαλέξει την προσθήκη καινούργιας εφαρμογής Android.



Εικόνα 5.3.5 : Προσθήκη Android εφαρμογής στο project

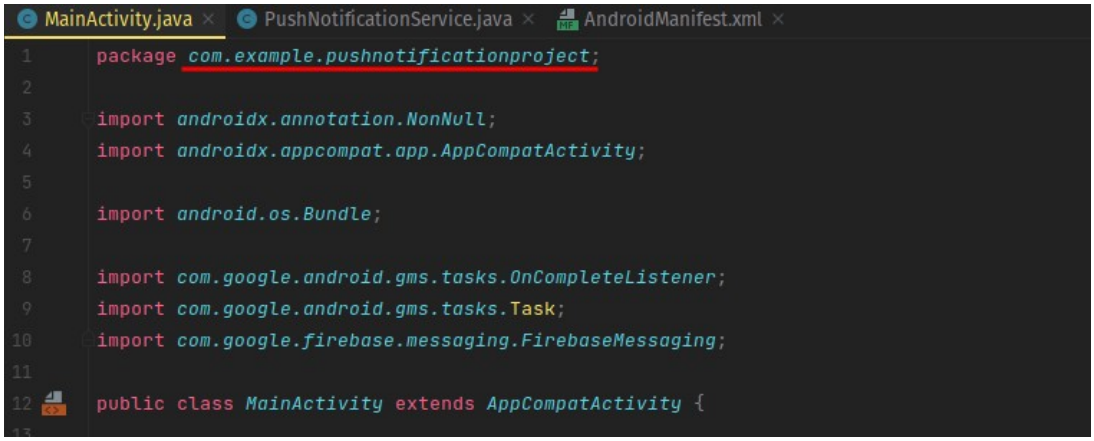
Βήμα 6ο + 7ο.

Τα δύο υποχρεωτικά πεδία που πρέπει να συμπληρώσει ο χρήστης είναι το όνομα πακέτου της Android εφαρμογής και την ψηφιακή υπογραφή (SHA-1).

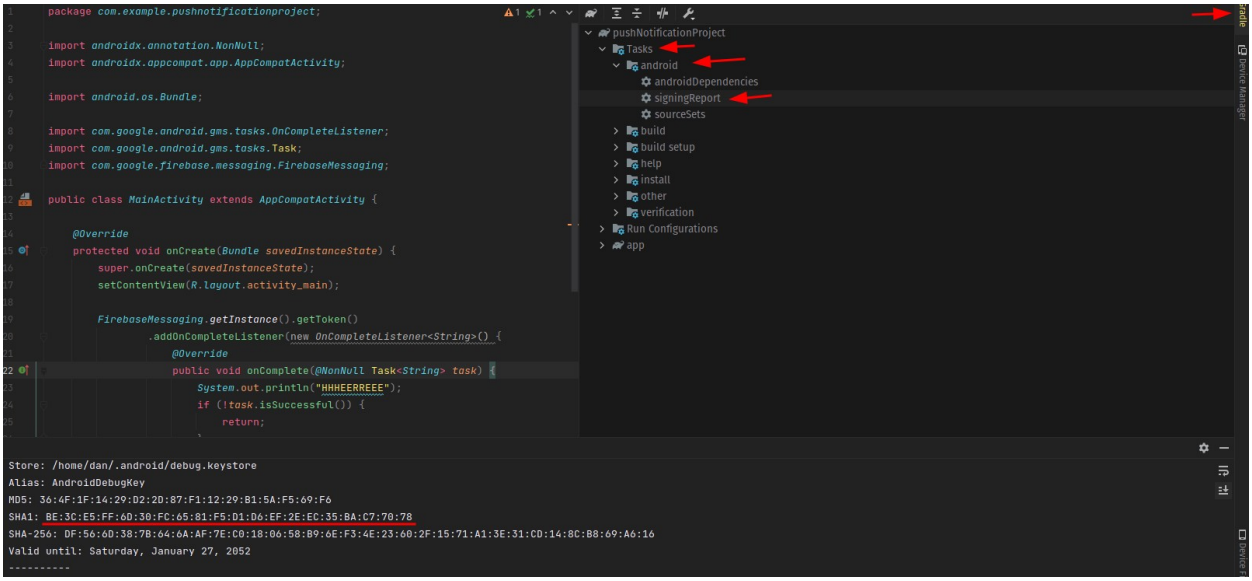


Εικόνα 5.3.6 : Τα υποχρεωτικά πεδία

Οι παρακάτω εικόνες αναπαριστούν τις θέσεις όπου βρίσκονται τα απαιτούμενα αυτά στοιχεία.



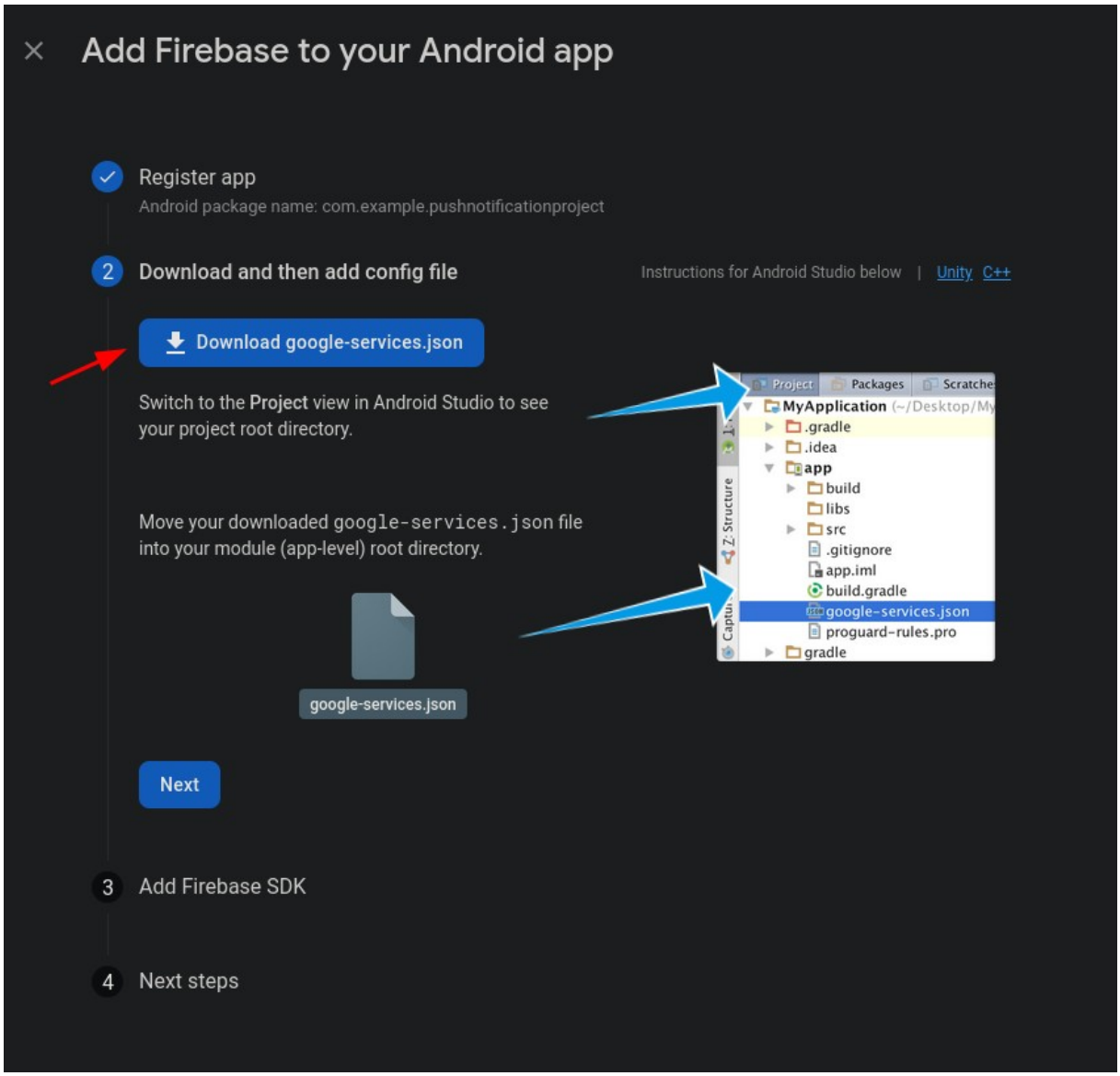
Εικόνα 5.3.7 : Εύρεση ονόματος της εφαρμογής



Εικόνα 5.3.8 : Εύρεση της ψηφιακής υπογραφής (SHA-1) της εφαρμογής

Βήμα 8ο.

Ύστερα θα δημιουργηθεί ένα “.json” αρχείο το οποίο είναι πλέον διαμορφωμένο και έτοιμο να περαστεί στην Android εφαρμογή στο “app/” φάκελο.



Εικόνα 5.3.9 : Κατέβασμα και αντιγραφή του “.json” αρχείο

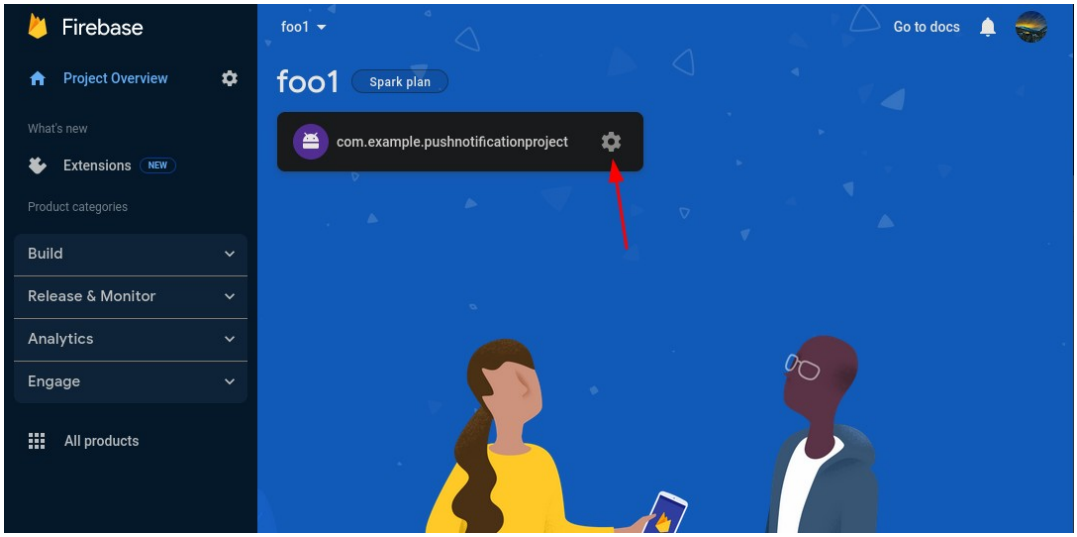
Βήμα 9ο.

Έπειτα θα πρέπει ο χρήστης να προσθέσει τις ακόλουθες γραμμές κώδικα στα αρχεία που υποδεικνύει η παρακάτω εικόνα.

Εικόνα 5.3.10 : Ο απαιτούμενος κώδικας που πρέπει να αντιγραφεί.

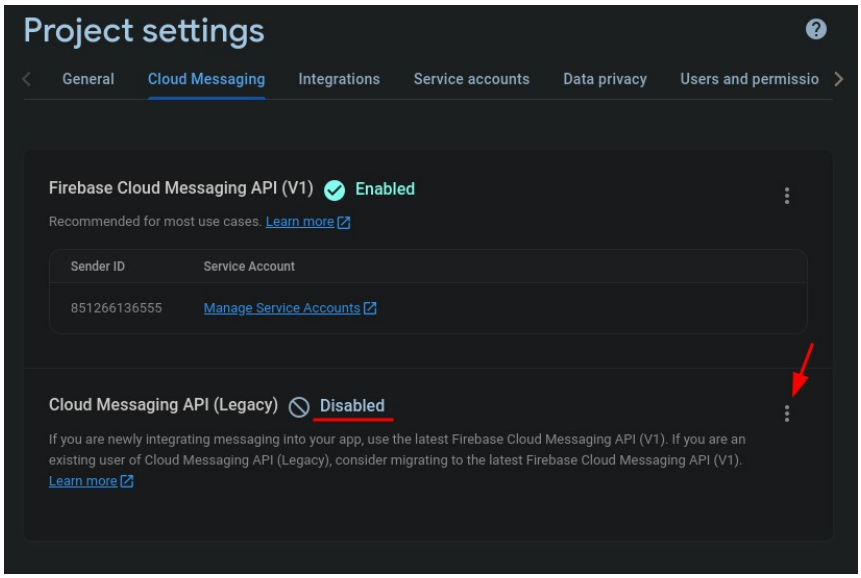
Βήμα 10ο + 11ο.

Τώρα μένει μόνο να ενεργοποιηθεί η δυνατότητα για πρόσβαση της Firebase Cloud εφαρμογής μέσω του API που παρέχει η Google. Για να γίνει αυτό πρέπει αρχικά να εισέλθει ο χρήστης στις ρυθμίσεις της εφαρμογής από το γρανάζι που υπάρχει δίπλα στον τίτλο της.



Εικόνα 5.3.11 : Μετάβαση στις ρυθμίσεις της καινούργιας εφαρμογής του project

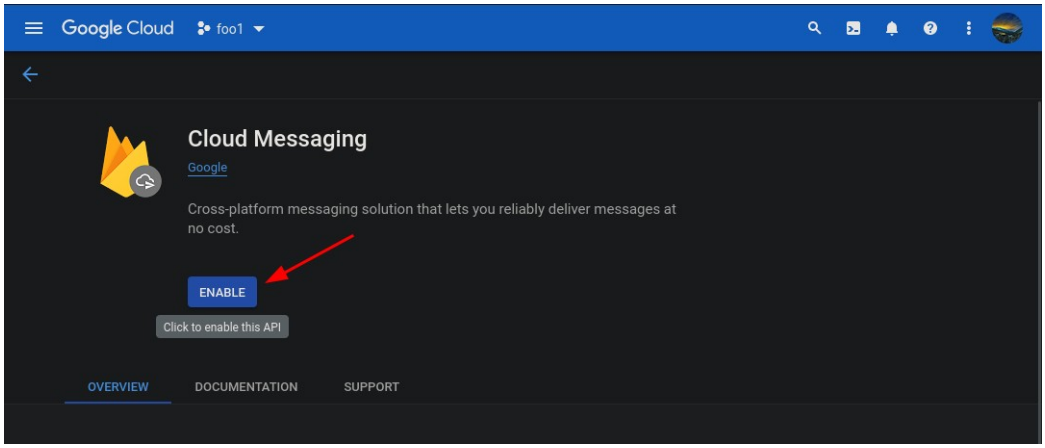
Ύστερα πρέπει να επιλέξει τη δεύτερη καρτέλα με το όνομα “Cloud Messaging” και από εκεί πια διαλέγει τις τρις τελείες που υπάρχουν δίπλα στο “Cloud Messaging API (Legacy)”. Και τέλος θα πατήσει την επιλογή που λέει “Manage API in Google Clound Console”.



Εικόνα 5.3.12 : Μετάβαση στις ρυθμίσεις του Cloud Messaging API

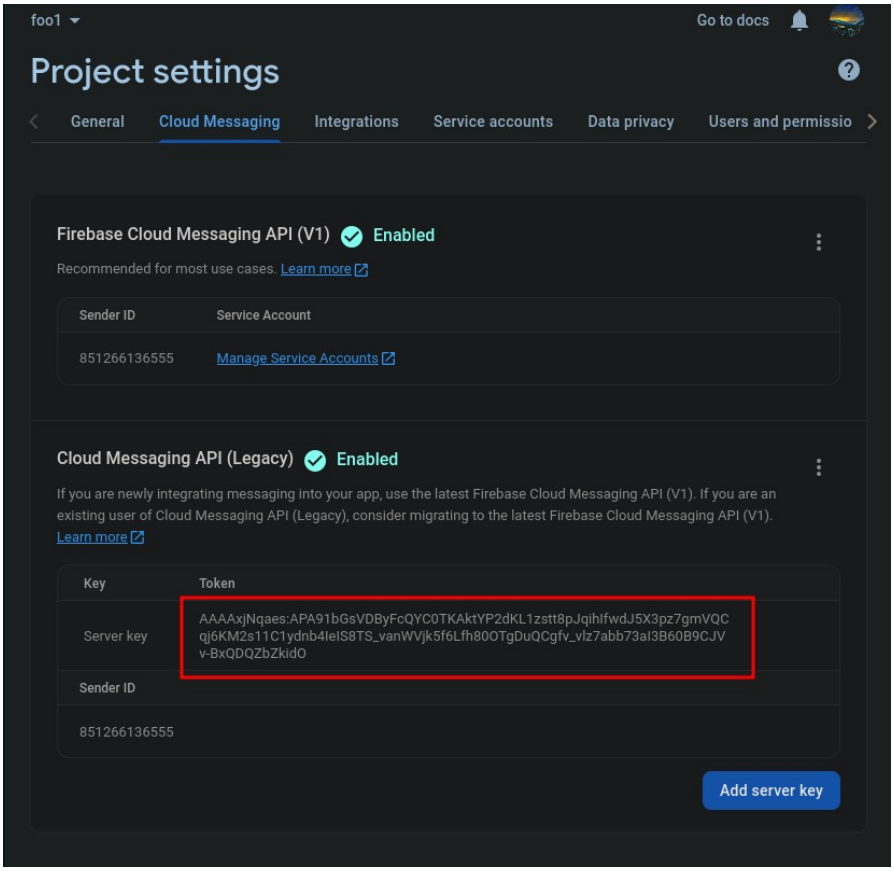
Βήμα 12ο + 13ο.

Αυτό θα ανακατευθύνει το χρήστη σε μια νέα σελίδα όπου αυτός πρέπει να πατήσει το κουμπί “Enable”.



Εικόνα 5.3.13 : Ενεργοποίηση του Cloud Messaging

Τέλος αν ξανά γυρίσει στη σελίδα των ρυθμίσεων θα διαπιστώσει πως από Disabled που είναι η προκαθορισμένη κατάσταση του Cloud Messaging API, είναι πια Enabled. Τώρα μπορεί πλέον ο χρήστης να αντιγράψει το Server key (ή Server token) ώστε να το περάσει στην εφαρμογή που θέλει να προωθεί τα notification που θα πρέπει να φτάσουν στους Clients.



Εικόνα 5.3.14 : Εύρεση του Server key

5.4 FCM & Android

Για να μπορέσει να υπάρξει επικοινωνία μεταξύ του Server και του Android Client, πρέπει αρχικά προστεθούν κάποια δικαιώματα και να γίνουν μερικές αλλαγές σε κάποια αρχεία της Client εφαρμογής.

Αρχικά πρέπει προστεθούν οι παρακάτω γραμμές κώδικα στο app's manifest αρχείο [14]:

```
<service
    Android:name=".java.MyFirebaseMessagingService"
    Android:exported="false">
    <intent-filter>
        <action Android:name="com.Google.Firebase.MESSAGING_EVENT" />
    </intent-filter>
</service>
```

Αυτό είναι απαραίτητο ώστε να γίνεται οποιοσδήποτε χειρισμός μηνυμάτων πέρα από την απλή λήψη ειδοποιήσεων σε εφαρμογές στο παρασκήνιο. Ύστερα πρέπει με κάποιον τρόπο να εξαχθεί το "ID" ή αλλιώς "Device Token" της Android συσκευής ώστε να γνωρίζει ο Server σε ποίον Client να στέλνει την ειδοποίηση ή το μήνυμα.

Ο παρακάτω κώδικας κάνει ακριβώς αυτή τη δουλεία [14] :

```
FirebaseMessaging.getInstance().getToken().addOnCompleteListener(new OnCompleteListener<String>() {

    @Override
    public void onComplete(@NonNull Task<String> task) {
        if (!task.isSuccessful()) {
            Log.w(TAG, "Fetching FCM registration token failed",                    task.getException());
        return;
        }
    // Get new FCM registration token
    String token = task.getResult();
    // Log and toast
    String msg = getString(R.string.msg_token_fmt, token);
    Log.d(TAG, msg);
    Toast.makeText(MainActivity.this, msg, Toast.LENGTH_SHORT).show();
    }
});
```

Και αυτό φυσικά με τον προϋπόθεση ότι η Android συσκευή του Client έχει εγκαταστημένο και "τρέχει" το Google Play Services, σε διαφορετική περίπτωση η εντολή "task.getResult()" θα επιστρέψει null. Οι εφαρμογές που βασίζονται στο SDK υπηρεσιών Play θα πρέπει πάντα να ελέγχουν τη συσκευή για ένα συμβατό APK υπηρεσιών Google Play πριν αποκτήσουν πρόσβαση στις λειτουργίες των υπηρεσιών Google Play. Συνιστάται να γίνεται αυτό σε δύο σημεία, στη μέθοδο onCreate() της κύριας δραστηριότητας και στη μέθοδο onResume(). Ο έλεγχος στη μέθοδο onCreate() διασφαλίζει ότι η εφαρμογή δεν μπορεί να χρησιμοποιηθεί χωρίς επιτυχή έλεγχο. Ο έλεγχος στην onResume() διασφαλίζει ότι αν ο χρήστης επιστρέψει στην εφαρμογή που εκτελείται μέσω κάποιου άλλου μέσου, όπως μέσω του κουμπιού back, ο έλεγχος εξακολουθεί να εκτελείται. Εάν η συσκευή δε διαθέτει συμβατή έκδοση των υπηρεσιών Google, η εφαρμογή μπορεί να καλέσει το GoogleApiAvailability.makeGooglePlayServicesAvailable() για να επιτρέψει στους χρήστες να κατεβάσουν τις υπηρεσίες Google Play από το Play Store [14].

Κεφάλαιο 6

Flask & Gunicorn

Όπως ήδη αναφέρθηκε για την επικοινωνία μεταξύ Server και Client χρησιμοποιήθηκαν Web Services. Για να επικοινωνήσει ο Server με τον Client μεσολάβησε το Flask Rest API της Python και για την επικοινωνία του Client με τον Server η Volley βιβλιοθήκη της Java που αναφέρεται στο επόμενο κεφάλαιο.

6.1 REST API

Υπάρχει ένας καταπληκτικός όγκος δεδομένων που είναι διαθέσιμος στον Παγκόσμιο Ιστό. Πολλές διαδικτυακές υπηρεσίες, όπως το YouTube και το GitHub, καθιστούν τα δεδομένα τους προσβάσιμα σε εφαρμογές τρίτων μέσω μιας διεπαφής προγραμματισμού εφαρμογών (API). Ένας από τους πιο δημοφιλείς τρόπους δημιουργίας API είναι το στυλ αρχιτεκτονικής REST. Η Python παρέχει μερικά εξαιρετικά εργαλεία όχι μόνο για να λήψη δεδομένων από REST APIs αλλά και τη δημιουργία Python REST APIs. Το REST σημαίνει αναπαραστατική μεταφορά κατάστασης (representational state transfer) και είναι ένα στυλ αρχιτεκτονικής λογισμικού που ορίζει ένα πρότυπο για την επικοινωνία μεταξύ πελάτη και διακομιστή μέσω δικτύου. Το REST παρέχει ένα σύνολο περιορισμών για την αρχιτεκτονική λογισμικού για την προώθηση της απόδοσης, της επεκτασιμότητας, της απλότητας και της αξιοπιστίας του συστήματος [15].

6.2 Flask

Στη διπλωματική αυτή εργασία χρησιμοποιήθηκε το Flask REST API για την ανταλλαγή μηνυμάτων (POST method) μεταξύ της Android συσκευής του Client και του Server. Το Flask [15] είναι ένα microframework της Python που χρησιμοποιείται για τη δημιουργία εφαρμογών ιστού και REST APIs. Το Flask παρέχει μια σταθερή ραχοκοκαλιά για τις εφαρμογές, ενώ αφήνει πολλές σχεδιαστικές επιλογές στον προγραμματιστή. Η κύρια δουλειά του Flask είναι να χειρίζεται τα αιτήματα HTTP και να τα δρομολογεί στην κατάλληλη λειτουργία της εφαρμογής. Επίσης είναι ένα από τα δημοφιλή εργαλεία της Python που χρησιμοποιούνται από προγραμματιστές για την ανάπτυξη ιστοσελίδων.

6.3 WSGI Servers

Το WSGI αναφέρεται στη Διασύνδεση Πύλης Διακομιστή Ιστού (Web Server Gateway Interface). Παίζει ζωτικό ρόλο κατά την ανάπτυξη της εφαρμογής Django ή Flask.
Εξηγεί πώς ένας διακομιστής ιστού επικοινωνεί με διαδικτυακές εφαρμογές python και πώς οι διαδικτυακές εφαρμογές μπορούν να συνδεθούν αλυσιδωτά για την επεξεργασία ενός αιτήματος [16]. Το WSGI δίνει ευελιξία [17], οι προγραμματιστές εφαρμογών μπορούν να αντικαταστήσουν τα στοιχεία της στοίβας ιστού με άλλα. Για παράδειγμα, ένας προγραμματιστής μπορεί να αλλάξει από το Green Unicorn στο uWSGI χωρίς να τροποποιήσει την εφαρμογή ή το πλαίσιο που υλοποιεί το

WSGI. Επίσης οι διακομιστές WSGI προωθούν την κλιμάκωση. Η ταυτόχρονη εξυπηρέτηση χιλιάδων αιτήσεων για δυναμικό περιεχόμενο είναι ο τομέας των διακομιστών WSGI και όχι των πλαισίων. Οι διακομιστές WSGI αναλαμβάνουν την επεξεργασία των αιτημάτων από τον διακομιστή ιστού και αποφασίζουν πώς να επικοινωνήσουν αυτά τα αιτήματα στη διαδικασία ενός πλαισίου εφαρμογών. Ο διαχωρισμός των αρμοδιοτήτων είναι σημαντικός για την αποτελεσματική κλιμάκωση της διαδικτυακής κίνησης.

6.4 Τι είναι το Gunicorn

Το Gunicorn 'Green Unicorn' [18] είναι ένας Python WSGI HTTP Server για UNIX. Είναι ένα pre-fork worker model από το έργο Unicorn της Ruby. Ο διακομιστής Gunicorn είναι ευρέως συμβατός με διάφορα web frameworks, απλά υλοποιημένος, ελαφρύς σε πόρους διακομιστή και αρκετά γρήγορος. Υποστηρίζει εγγενώς διάφορα frameworks με τους προσαρμογείς του, καθιστώντας τον εξαιρετικά εύχρηστο drop-in αντικαταστάτη για πολλούς διακομιστές ανάπτυξης που χρησιμοποιούνται κατά τη διάρκεια της ανάπτυξης. Τεχνικά, ο τρόπος λειτουργίας του Gunicorn είναι πολύ παρόμοιος με τον επιτυχημένο διακομιστή ιστού Unicorn για εφαρμογές Ruby. Και οι δύο χρησιμοποιούν αυτό που αναφέρεται ως το μοντέλο pre-fork. Αυτό, στην ουσία, αναθέτει στην κεντρική [Gunicorn] κύρια διαδικασία να χειριστεί τη διαχείριση των Clients, τη δημιουργία υποδοχών (sockets), συνδέσεων, κ.λπ.

6.5 Gunicorn και Threading

Φυσικά ο Gunicorn παρέχει μεταξύ άλλων τη δυνατότητα για threading δηλαδή την “παράλληλη” εκτέλεση του κώδικα του Server ώστε έτσι να μπορεί να εξυπηρετεί πολλαπλούς χρήστες την ίδια στιγμή. Αλλά όπως ήδη αναφέρθηκε πιο πριν επειδή έπρεπε να αποφευχθεί ο κοινή χρήση της βάσης δεδομένων από παραπάνω από έναν Client την ίδια χρονική στιγμή, προτιμήθηκε να τρέχει ο Gunicorn με ένα μόνο thread.

6.6 Εγκατάσταση του Gunicorn

Για να εγκατασταθεί η τελευταία έκδοση του Gunicorn [19] φτάνει να τρέξει η εντολή “pip install gunicorn” στην κονσόλα της python.

Αν χρησιμοποιείται το Debian GNU/Linux σύστημα, συνιστάται να χρησιμοποιούνται τα πακέτα συστήματος(system packages) για την εγκατάσταση του Gunicorn, εκτός ίσως από την περίπτωση που πρόκειται να χρησιμοποιηθούν διαφορετικές εκδόσεις του Gunicorn με το virtualenv.

Αυτό έχει μια σειρά από πλεονεκτήματα :

- Εγκατάσταση χωρίς κόπο, ξεκινά αυτόματα πολλαπλές περιπτώσεις Gunicorn με βασισμένα στις ρυθμίσεις που καθορίζονται στο αρχείο /etc/gunicorn.d
- Λογικές προεπιλεγμένες τοποθεσίες για τα αρχεία καταγραφής (/var/log/gunicorn). Τα αρχεία καταγραφής μπορούν να περιστρέφονται και να συμπιέζονται αυτόματα χρησιμοποιώντας το logrotate
- Βελτιωμένη ασφάλεια: Μπορεί εύκολα να τρέξει κάθε περίπτωση Gunicorn με έναν αποκλειστικό χρήστη/ομάδα UNIX.

Κεφάλαιο 7

Volley βιβλιοθήκη

7.1 Εισαγωγή

Η Volley είναι μια βιβλιοθήκη HTTP που κάνει τη δικτύωση για εφαρμογές Android ευκολότερη και, κυρίως, ταχύτερη. Το Volley είναι διαθέσιμο στο GitHub [20].

Προσφέρει τα ακόλουθα οφέλη :

- Αυτόματος προγραμματισμός αιτήσεων δικτύου.
- Πολλαπλές ταυτόχρονες συνδέσεις δικτύου.
- Διαφανής προσωρινή αποθήκευση απαντήσεων σε δίσκο και μνήμη με τυπική συνοχή κρυφής μνήμης HTTP.
- Υποστήριξη προτεραιοποίησης αιτήσεων.
- API αίτησης ακύρωσης. Μπορεί να ακυρωθεί ένα μεμονωμένο αίτημα ή να οριστεί μπλοκ ή πεδία αιτημάτων προς ακύρωση.
- Ευκολία προσαρμογής, για παράδειγμα, για επανάληψη και backoff.
- Ισχυρή ταξινόμηση που διευκολύνει τη σωστή συμπλήρωση του UI με δεδομένα που λαμβάνονται ασύγχρονα από το δίκτυο.
- Εργαλεία εντοπισμού σφαλμάτων και ανίχνευσης.

Η Volley [20] ενσωματώνεται εύκολα με οποιοδήποτε πρωτόκολλο και έρχεται από το κουτί με υποστήριξη για ακατέργαστες συμβολοσειρές, εικόνες και JSON. Παρέχοντας ενσωματωμένη υποστήριξη για τα χαρακτηριστικά που χρειάζεστε, το Volley απαλλάσσει τον προγραμματιστή από τη συγγραφή κώδικα boilerplate και επιτρέπει να αφοσίωση στη λογική που είναι συγκεκριμένη για την εφαρμογή. Η Volley δεν είναι κατάλληλη για μεγάλες λειτουργίες λήψης ή ροής, καθώς κρατάει όλες τις απαντήσεις στη μνήμη κατά τη διάρκεια της ανάλυσης. Για μεγάλες λειτουργίες λήψης, υπάρχουν εναλλακτικές λύσης όπως το DownloadManager. Η βασική βιβλιοθήκη της αναπτύσσεται στο GitHub και περιέχει τον κύριο αγωγό αποστολής αιτήσεων, καθώς και ένα σύνολο κοινά εφαρμόσιμων βοηθητικών προγραμμάτων, που είναι διαθέσιμα στην "εργαλειοθήκη" Volley.

7.2 Ενσωμάτωση στο Android project

Ο ευκολότερος τρόπος για να ενσωματωθεί η Volley [20] στο έργο γίνεται με την προθήκη των ακόλουθων γραμμών στο αρχείο build.gradle της εφαρμογής :

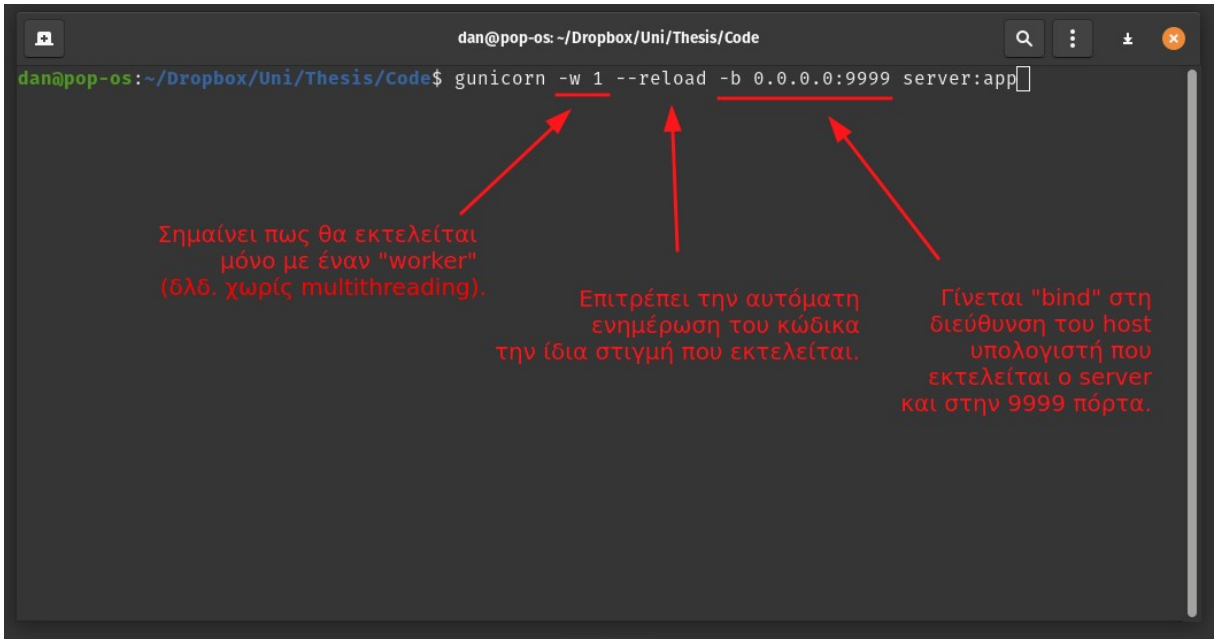
```
dependencies {
    implementation 'com.Android.volley:volley:1.2.1'
}
```

Κεφάλαιο 8

Λειτουργία της εφαρμογής

8.1 Εκτέλεση Server

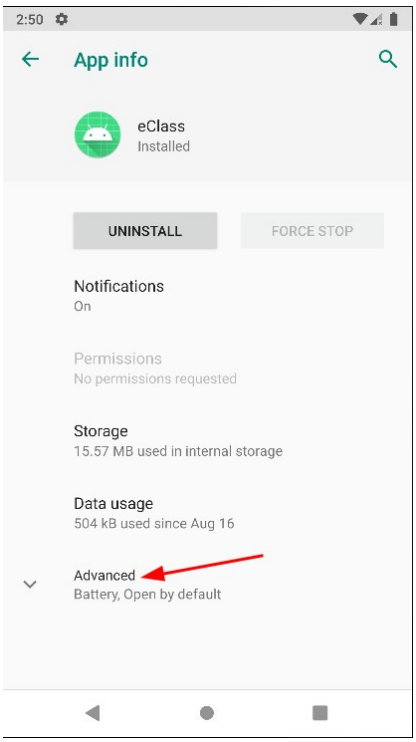
Για να επιτευχθεί η επικοινωνία μεταξύ του Server και των Clients πρέπει αρχικά να “τρέξει” ο Server μέσω το Gunicorn και με τις σωστές ρυθμίσεις όπως φαίνεται στην παρακάτω εικόνα.

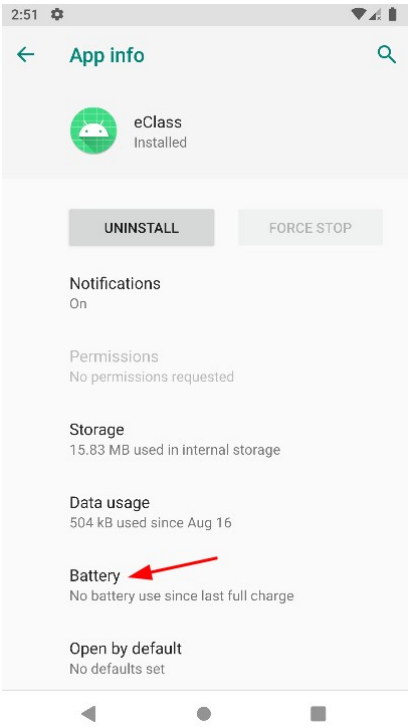


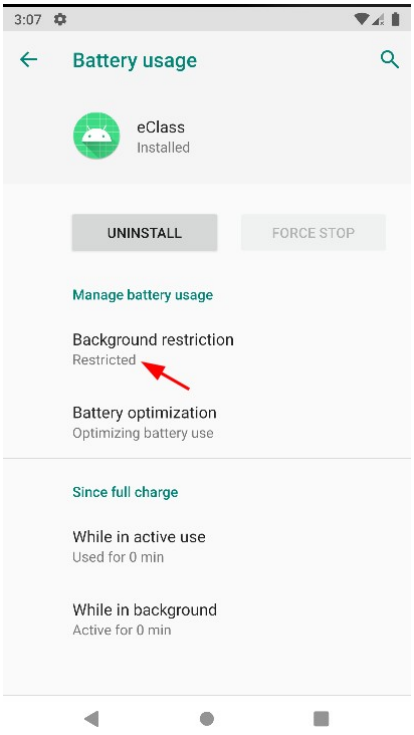
Εικόνα 8.1 : Εκτέλεση Server μέσω terminal linux συστήματος

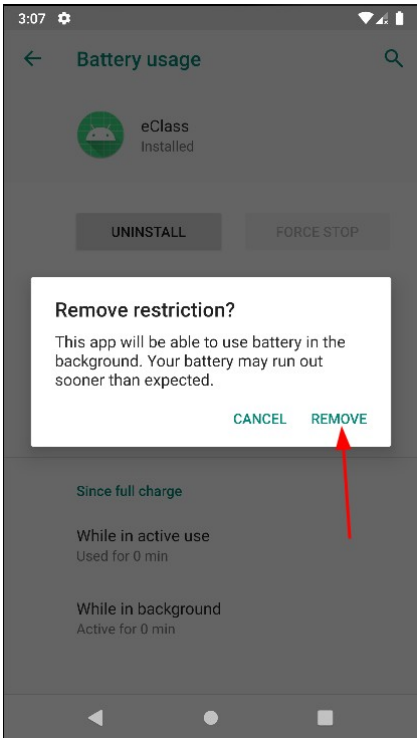
8.2 Εγκατάσταση Client εφαρμογής

Από τη μεριά του Client πρέπει εγκατασταθεί η αντίστοιχη εφαρμογή στην Android συσκευή, είτε κατευθείαν από το περιβάλλον “Android Studio” ή σαν .apk αρχείο. Προτείνεται αφού εγκατασταθεί η εφαρμογή ο χρήστης να επιτρέψει στην εφαρμογή να δουλέψει στο παρασκήνιο, αφαιρώντας την από την εξοικονόμηση μπαταρίας.





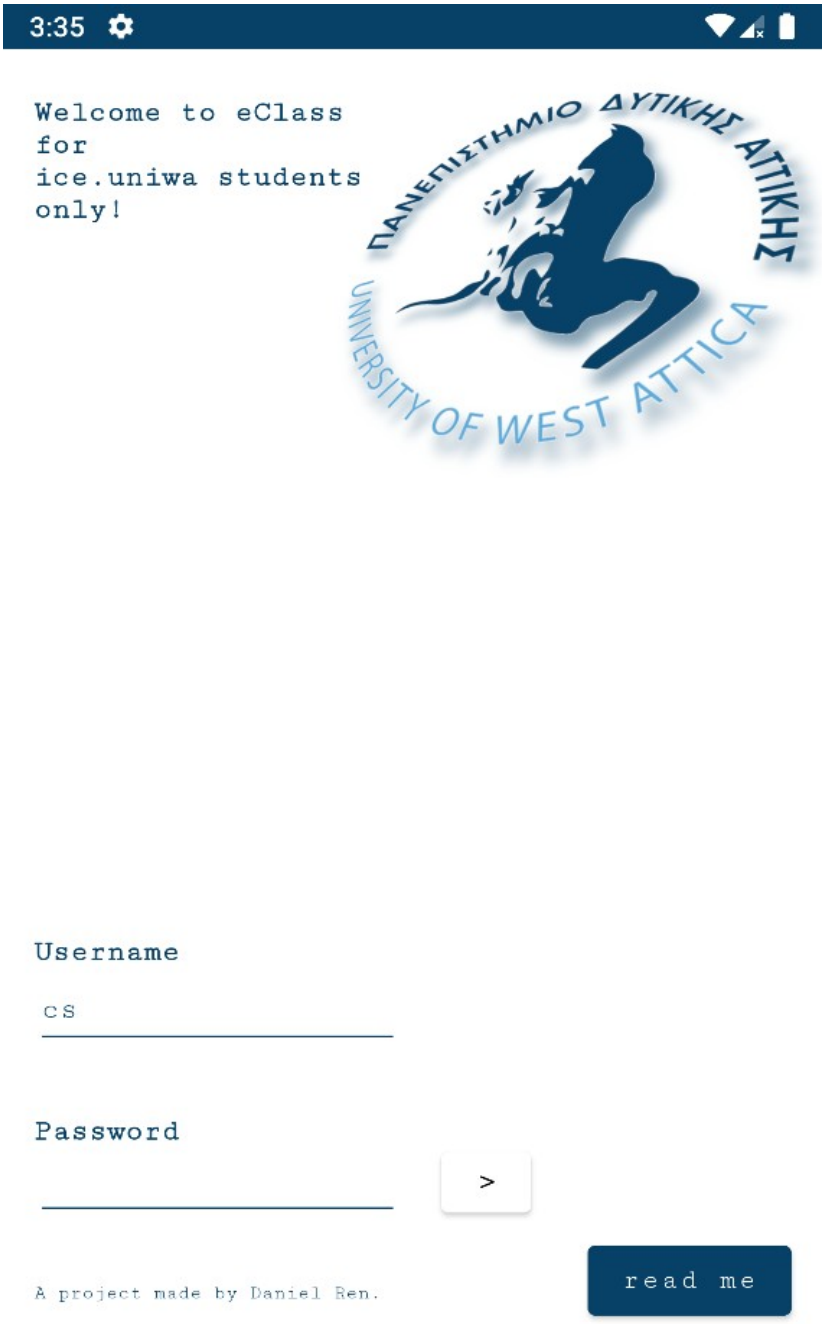




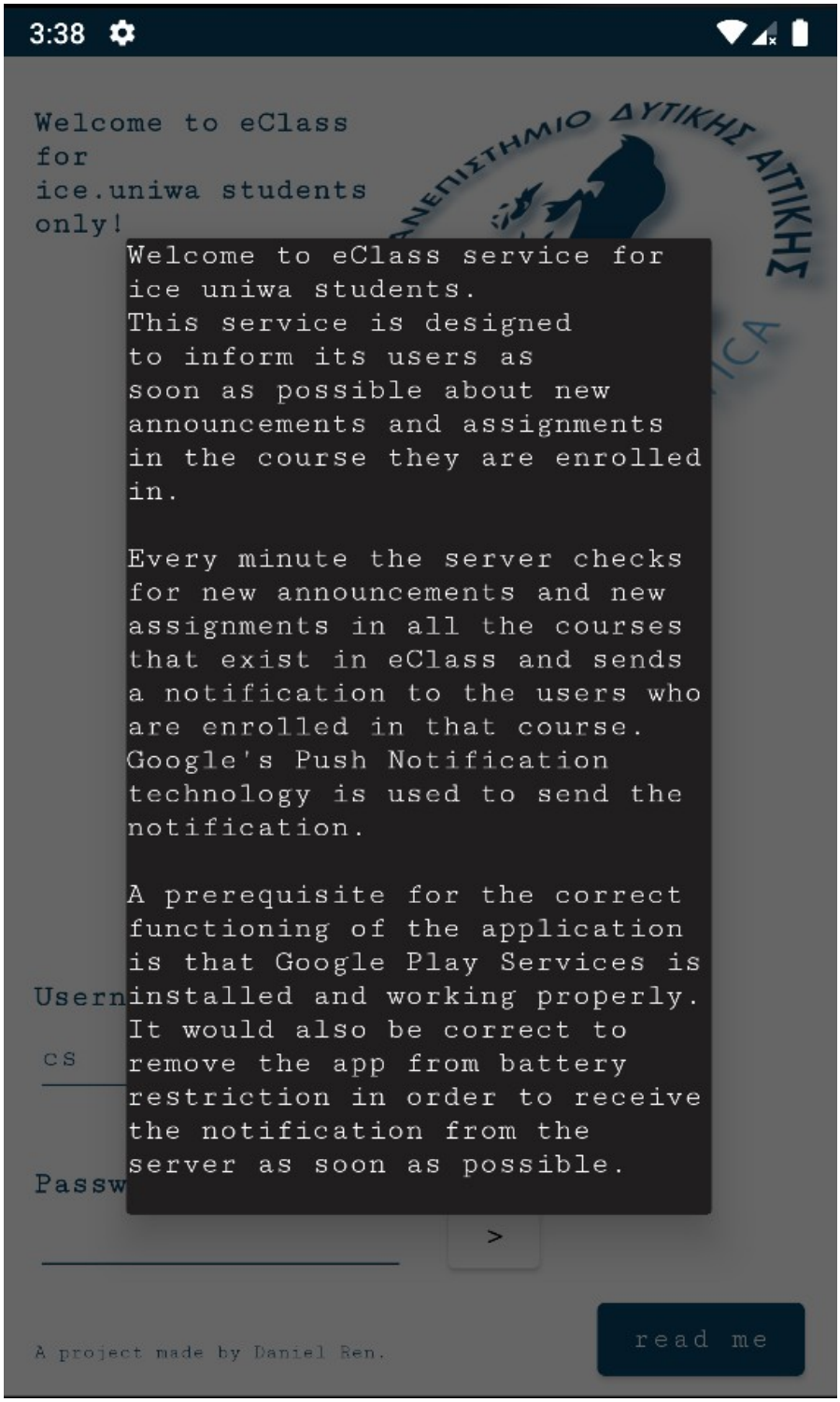
Εικόνες 8.2.1 – 8.2.4 : Ενδεικτική αφαίρεση της εφαρμογής από την εξοικονόμηση μπαταρίας

8.3 Λίγα λόγια σχετικά με το αρχικό μενού

Όπως φαίνεται και στην παρακάτω εικόνα δόθηκε έμφαση πιο πολύ στη λειτουργία της εφαρμογής απ' ότι στην εμφάνιση της, καθώς είναι σχεδιασμένη πολύ απλά χωρίς πολλές επιλογές. Περιέχει μεταξύ των άλλων δύο πεδία για την εισαγωγή των username και password, ένα κουμπί για να σταλθούν τα στοιχεία αυτά (να γίνει login) και ένα ακόμη κουμπί το οποίο όταν το πατήσει ο χρήστης θα του εμφανιστεί ένα παράθυρο με λίγα λόγια σχετικά με την εφαρμογή στα αγγλικά γραμμένα. Το μενού μετά τη σύνδεση του χρήστη αλλάζει λίγο, όπως θα φανεί σε επόμενο κεφάλαιο.



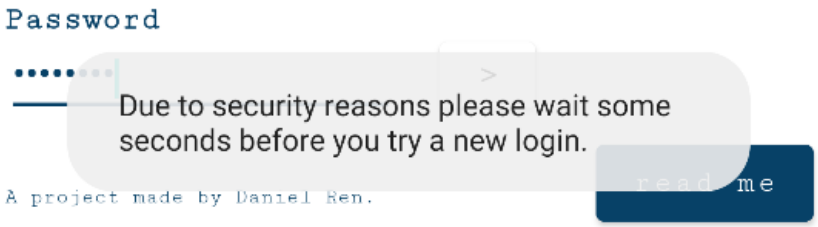
Εικόνα 8.3.1 : Αρχικό μενού της Client εφαρμογής



Εικόνα 8.3.2 : Το παράθυρο που εμφανίζεται αφού ο χρήστης πατήσει το κουμπί “read me”

8.4 Προσπάθεια εισόδου χρήστη με λάθος στοιχεία

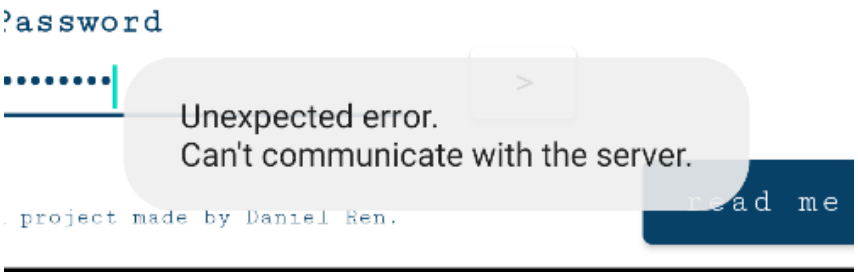
Εάν ο χρήστης πατήσει σύνδεση με λάθος στοιχεία (είτε username είτε password) θα του εμφανιστεί το λεγόμενο “Toast”, δηλαδή ένα μικρό συννεφάκι στο κάτω μέρος της οθόνης με το μήνυμα “Due to security reasons please wait some seconds before you try a new login.”. Διαλέχτηκε αυτό το μήνυμα επειδή η ιστοσελίδα του eClass κάποια περίοδο είχε όριο ανεπιτυχών προσπαθειών εισόδου, λέγοντας στου χρήστες να περιμένουν για ένα χρονικό διάστημα πριν ξανά προσπαθήσουν να συνδεθούν.



Εικόνα 8.4 Προσπάθεια εισόδου χρήστη με λάθος στοιχεία

8.5 Προσπάθεια εισόδου χρήστη χωρίς σύνδεση με τον Server

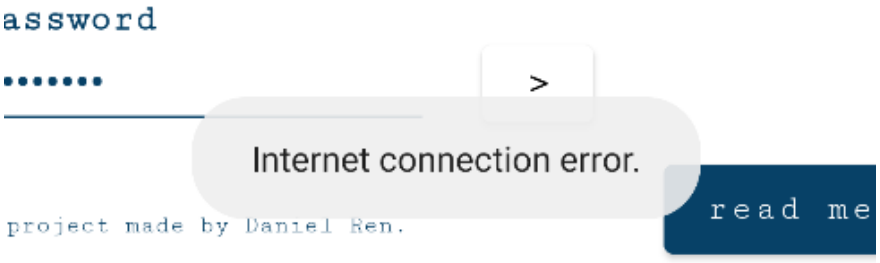
Όπως ήδη προαναφέρθηκε, έτσι όπως έχει σχεδιασθεί η εφαρμογή, για να επιτευχθεί η σύνδεση μεταξύ του Server και του Client, πρέπει και οι δύο να βρίσκονται στο ίδιο τοπικό δίκτυο. Εάν αυτό δε συμβεί (και ο χρήστης δώσει τα σωστά στοιχεία), επειδή ο Client δεν είναι στο ίδιο δίκτυο ή ο Server “πέσει” για κάποιο λόγο, τότε ένα “Toast” εμφανίζεται προς ενημέρωση του χρήστη με το μήνυμα “Unexpected error. Can't communicate with the Server.”.



Εικόνα 8.5 Προσπάθεια εισόδου ενώ δεν υπάρχει επικοινωνία με τον Server

8.6 Προσπάθεια εισόδου χρήστη χωρίς σύνδεση στο διαδίκτυο

Στην περίπτωση που ο χρήστης δεν έχει καθόλου σύνδεση στο διαδίκτυο τότε εμφανίζεται ένα “Toast” με το μήνυμα “Internet connection error.”.

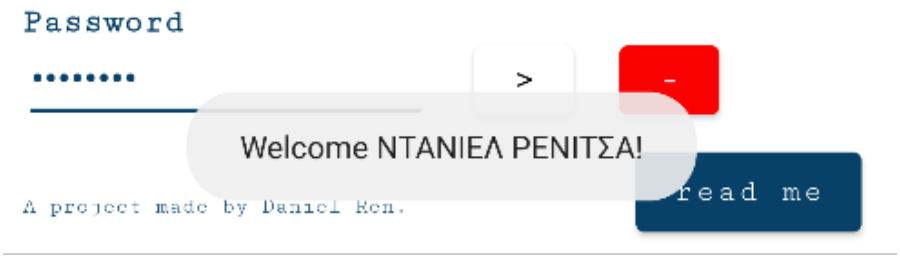


Εικόνα 8.6 Προσπάθεια εισόδου ενώ δεν υπάρχει σύνδεση στο διαδίκτυο

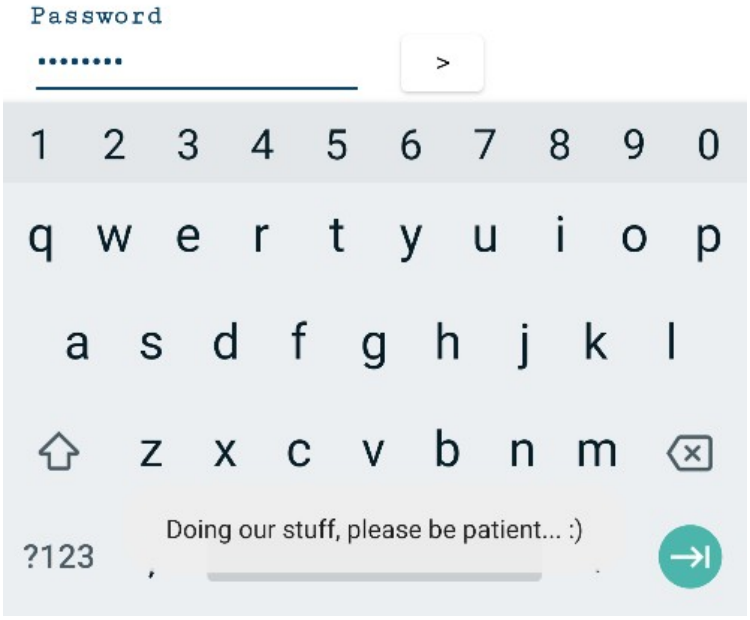
8.7 Επιτυχής είσοδος & αφαίρεση χρήστη

Όταν ο χρήστης δώσει τα σωστά στοιχεία και ο Server δουλεύει σωστά στο ίδιο δίκτυο με την Client συσκευή τότε εμφανίζεται ένα “Toast” με το μήνυμα “Welcome ” + το ονοματεπώνυμο του χρήστη που έχει στο λογαριασμό του eClass. Επειδή στην επιτυχή σύνδεση γίνεται πρώτα η ενημέρωση της βάσης δεδομένων με τα στοιχεία του χρήστη και των μαθημάτων του και μετά, εφόσον οι ενημερώσεις γίνουν με επιτυχία εμφανίζεται στο χρήστη το “Toast” που αναφέρθηκε προηγουμένους, υπάρχει μια μικρή καθυστέρηση. Σε αυτήν την περίπτωση ενημερώνεται ο χρήστης με αντίστοιχο “Toast” μήνυμα, ώστε να γνωρίζει ότι η εφαρμογή επεξεργάζεται τα στοιχεία του πριν την τελική είσοδο του. Επίσης μετά από μια επιτυχή είσοδο εμφανίζεται ένα κόκκινο κουμπί ώστε ο χρήστης να έχει τη δυνατότητα να αφαιρέσει τον εαυτό του μελλοντικά εάν το επιθυμήσει. Αυτό είναι αρκετά σημαντικό καθώς ναι μεν μπορεί να σταματήσει να δέχεται ειδοποιήσεις εάν διαγράψει απλά την εφαρμογή από το κινητό του, αλλά η βάση δεδομένων θα τον έχει ακόμη ως εγγραφή και αυτό δε θα ήταν ορθό. Για το λόγο αυτό είναι απαραίτητο ο χρήστης πριν τη διαγραφή της εφαρμογής να ενημερώσει και τον Server ότι πρόκειται να αποχωρήσει. Για να γίνει με επιτυχία η διαγραφή ενός χρήστη από τη βάση του Server θα πρέπει και πάλι να είναι συνδεδεμένη στο ίδιο δίκτυο.

Επίσης σημαντική σημείωση, από τη στιγμή που ο χρήστης συνδεθεί επιτυχώς, από εκεί και πέρα η εφαρμογή θα θυμάται τα στοιχεία του και θα τα έχει συμπληρωμένα στα αντίστοιχα πεδία, ακόμη και μετά από kill της, αυτό όμως δεν ισχύει μετά από διαγραφή και ξανά εγκατάστασης της. Εάν ο χρήστης προσπαθήσει να συνδεθεί με διαφορετικά στοιχεία από αυτά που είναι ήδη συνδεδεμένος τότε αντίστοιχο μήνυμα θα τον ενημερώσει ότι θα πρέπει πρώτα να αποσυνδεθεί από τον προηγούμενο λογαριασμό του πριν συνδεθεί σε άλλον.



Εικόνα 8.7.1 : Το μήνυμα που εμφανίζεται μετά από επιτυχή είσοδο του χρήστη



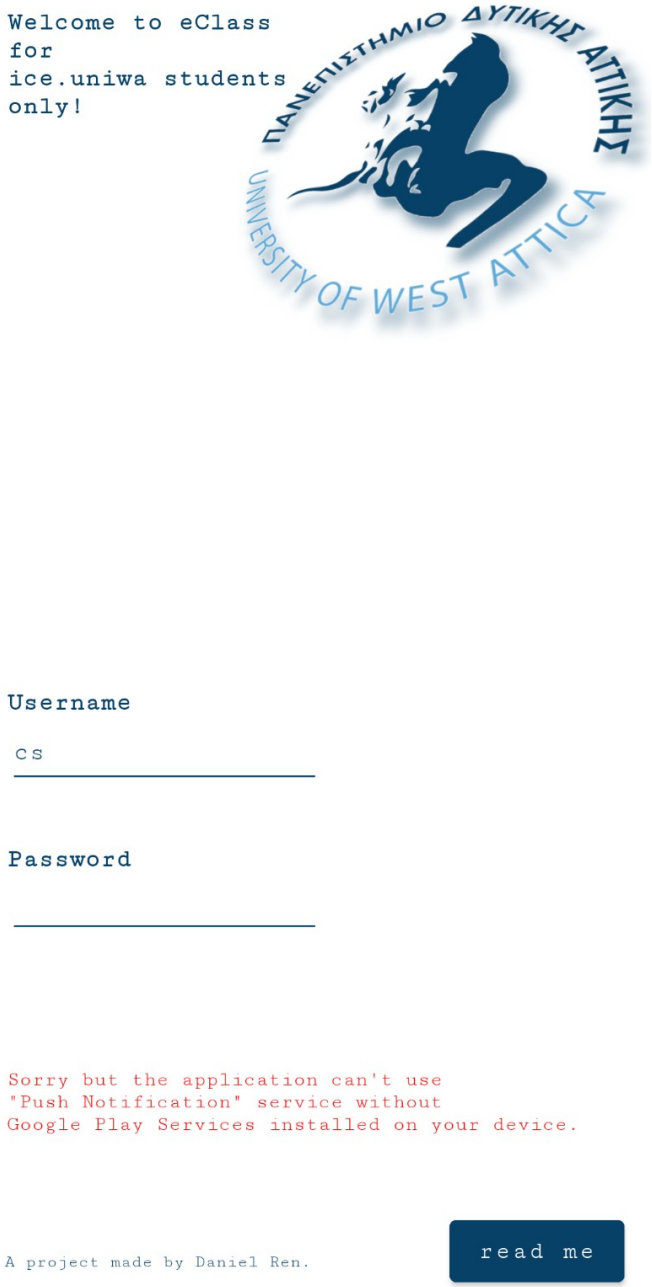
Εικόνα 8.7.2 : Το μήνυμα που εμφανίζεται την ώρα που γίνεται η επεξεργασία των δεδομένων πριν την τελική είσοδο

Εικόνα 8.7.3 : Το μήνυμα που εμφανίζεται όταν ο χρήστης προσπαθήσει να συνδεθεί με διαφορετικά στοιχεία από αυτά που είναι ήδη συνδεδεμένος

Εικόνες 8.7.4 – 8.7.6 : Διαδικασία αφαίρεσης του χρήστη

8.8 Περίπτωση συσκευής χωρίς “Google Play Services”

Στην περίπτωση που η συσκευή του χρήστη δε διαθέτη υπηρεσίες της Google εμφανίζεται με κόκκινα γράμματα ένα μήνυμα στο αρχικό μενού που ενημερώνει το χρήστη ότι η εφαρμογή δεν μπορεί να τρέξει χωρίς τη βοήθεια της Google και γίνεται invisible το login κουμπί. Η Google υπηρεσίες χρειάζονται μόνο για να μπορέσει να λειτουργήσει η τεχνολογία “Push Notification” της Google Firebase Cloud Messaging (FCM).

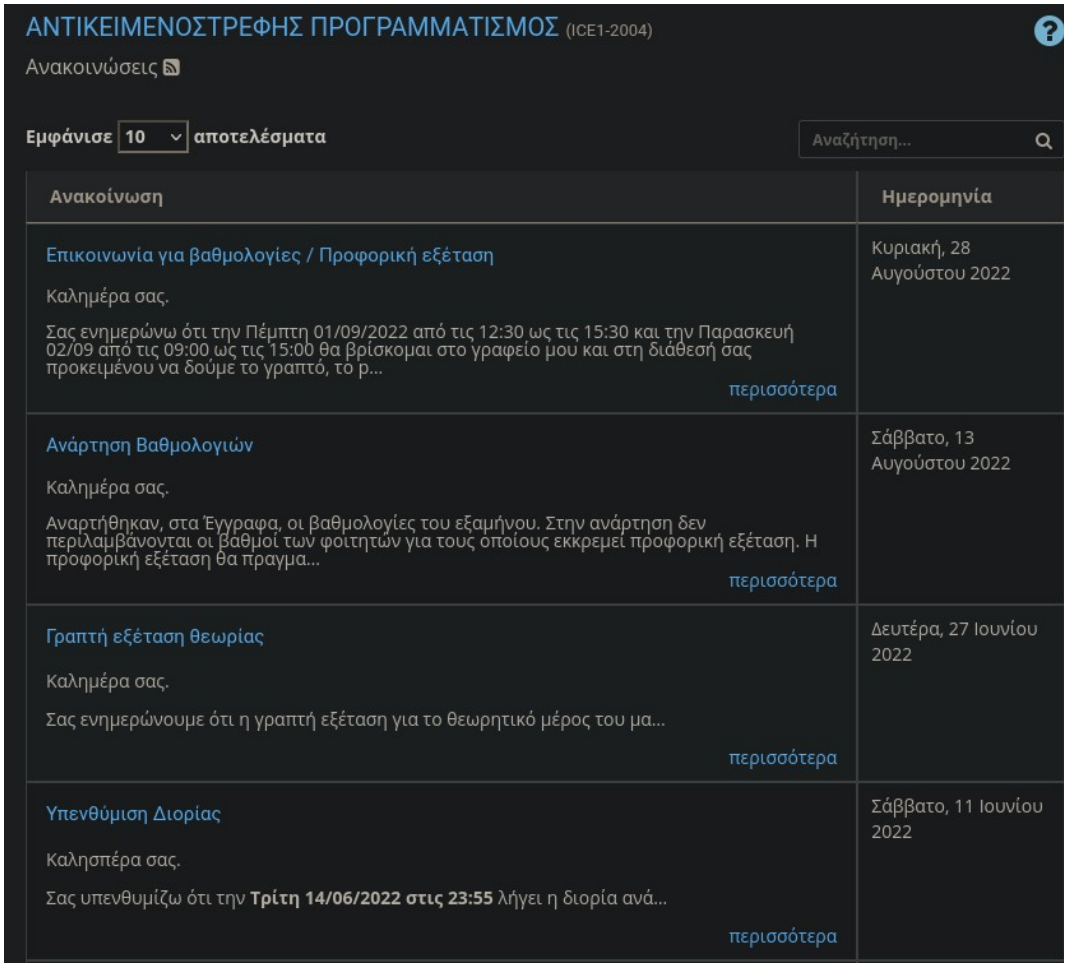


Εικόνα 8.8 : Συσκευή που δεν έχει τις βασικές υπηρεσίες της Google εγκαταστημένες

8.9 Πραγματικό παράδειγμα ενημέρωσης του χρήστη με ειδοποίηση

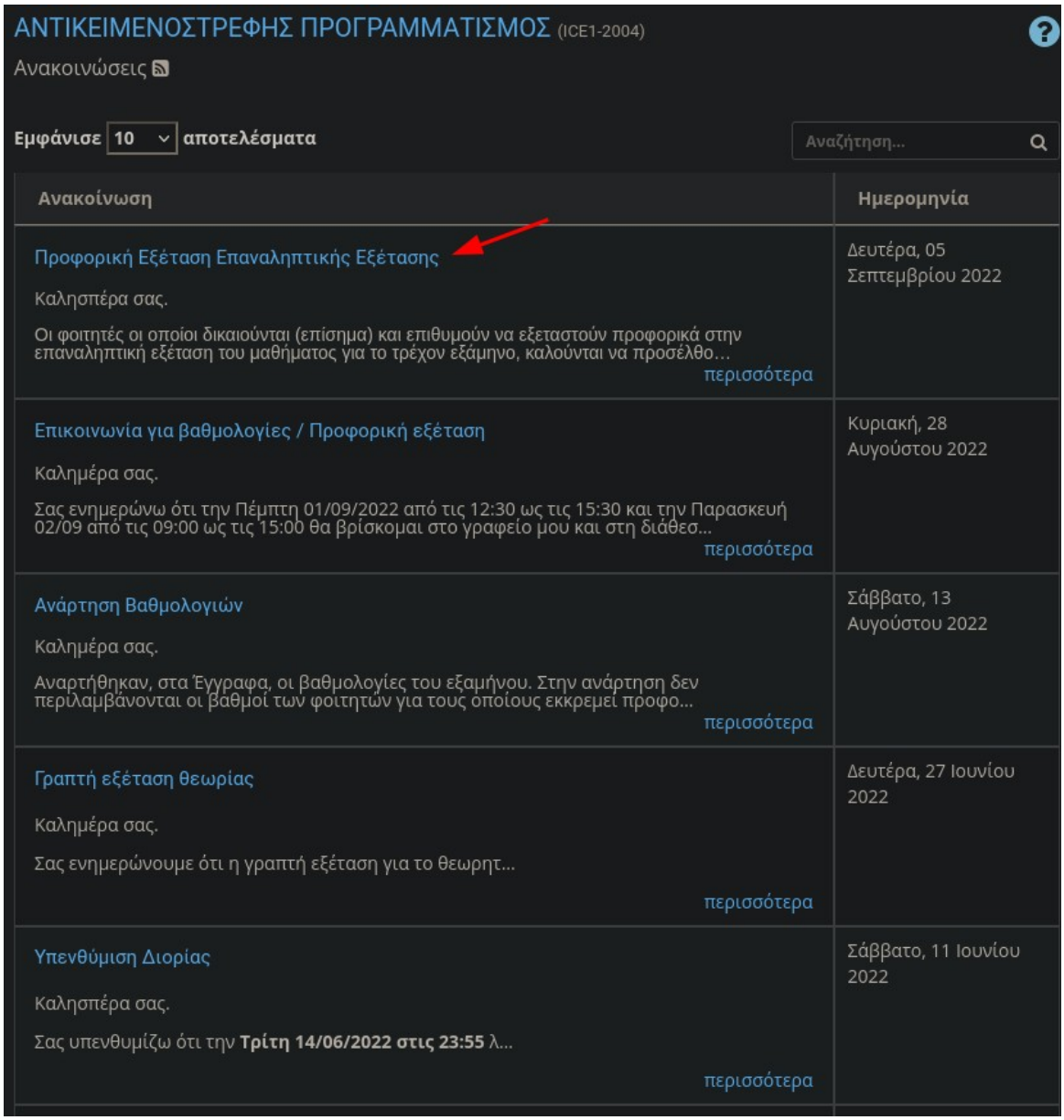
Εφόσον συνδεθεί επιτυχώς ο χρήστης στην εφαρμογή, ο Server δε θα του στείλει κάποια ειδοποίηση γιατί υποτίθεται ότι τη στιγμή που συνδέθηκε όλες οι ανακοινώσεις που έχουν αναρτηθεί στα μαθήματα του έως εκείνη τη στιγμή είναι διαβασμένες από αυτόν. Αν αντιθέτως στέλνονταν ειδοποίηση για κάθε ανακοίνωση και εργασία για όλα τα μαθήματα του χρήστη, τότε θα γινόταν μια επίθεση από ειδοποιήσεις ακριβώς μετά την επιτυχή είσοδο του, πράγμα που δεν πρέπει να συμβεί. Γι αυτό στην μόνη περίπτωση που θα εμφανιστεί ειδοποίηση στη συσκευή του χρήστη είναι όταν αναρτηθεί μια νέα ανακοίνωση ή εργασία σε κάποιο ενδιαφερόμενο μάθημα. Ο Server όπως έχει ήδη αναφερθεί εκτελεί αυτόν τον έλεγχο για κάποια νέα τέτοια ανάρτηση κάθε ένα λεπτό. Σε περίπτωση που βρει κάποια καινούργια, ελέγχει πόσοι χρήστες του είναι γραμμένοι στο εν λόγω μάθημα. Για κάθε έναν τέτοιο χρήστη βρίσκει το Device Token της συσκευής του και ετοιμάζει την φόρμα της ειδοποίησης. Στέλνει την τελική μορφή της ειδοποίηση στο Server της Google Firebase και αυτή μετά ξέροντας από ποίον ήρθε και σε ποίον πρέπει να φτάσει την προωθεί στην συσκευή του χρήστη. Τέλος ο Server ενημερώνει τη βάση δεδομένων του ώστε να μπορέσει να ξεχωρίσει την πιθανή μελλοντική ανακοίνωση από τις προηγούμενες. Το ίδιο ακριβώς συμβαίνει και όταν ανακοινώνεται καινούργια εργασία σε κάποια ενδιαφερόμενο μάθημα (δηλαδή μάθημα που είναι γραμμένος ο χρήστης).

Έστω στο ένα φοιτητής είναι γραμμένος μεταξύ των άλλως και στο μάθημα “Αντικειμενοστρεφής προγραμματισμός”. Τη στιγμή που συνδέθηκε για πρώτη φορά στην εφαρμογή υπήρχαν οι παρακάτω ανακοινώσεις σε αυτό το μάθημα.



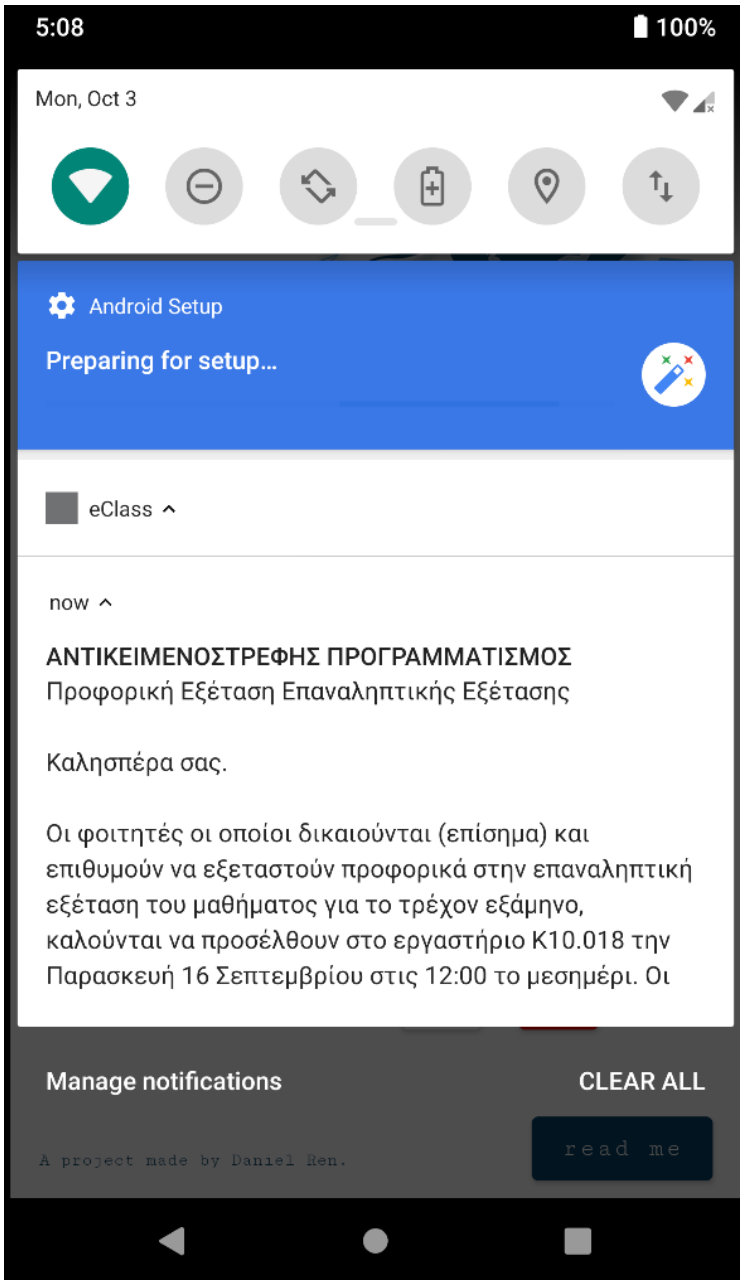
Εικόνα 8.9.1 : Παράδειγμα ανακοινώσεων στο eClass (1)

Η εφαρμογή όπως αναφέρθηκε δε θα εμφανίσει καμία από αυτές τις ανακοινώσεις καθώς ο χρήστης μόλις συνδέθηκε. Έστω ότι μετά από ένα διάστημα ανεβαίνει μία καινούργια ανακοίνωση στο ίδιο μάθημα και ο χρήστης είναι ακόμη συνδεδεμένος στη Client εφαρμογή (θεωρείται ότι δεν έχει επιλέξει έξοδο και διέγραψε την εφαρμογή και ότι έχει επιτρέψει να “τρέχει” στο backgroud βγάζοντας την από την “εξοικονόμηση μπαταρίας”).



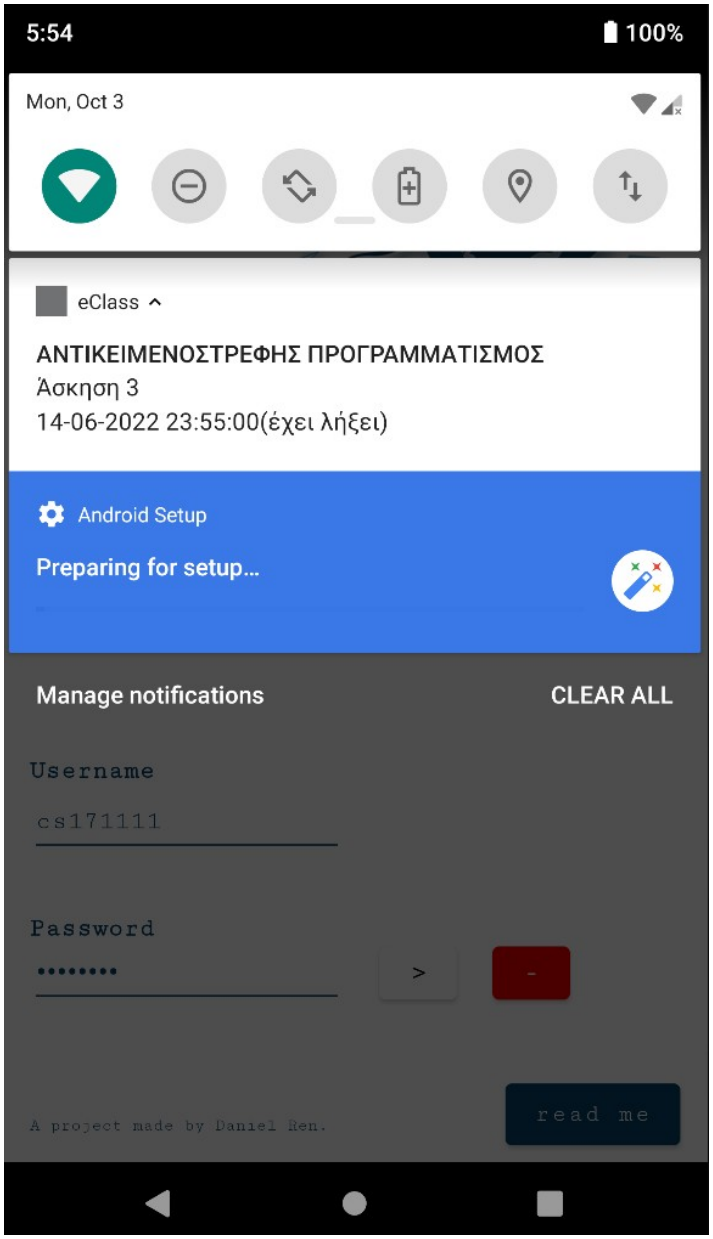
Εικόνα 8.9.2 : Παράδειγμα ανακοινώσεων στο eClass (2)

Τότε όταν ο Server εντοπίσει την καινούργια αυτή ανακοίνωση θα κάνει όλες τις λειτουργίες που αναφέρθηκαν πριν και θα εμφανιστεί η αντίστοιχη ειδοποίηση στην οθόνη του χρήστη ή των χρηστών που πρέπει να ενημερωθούν.

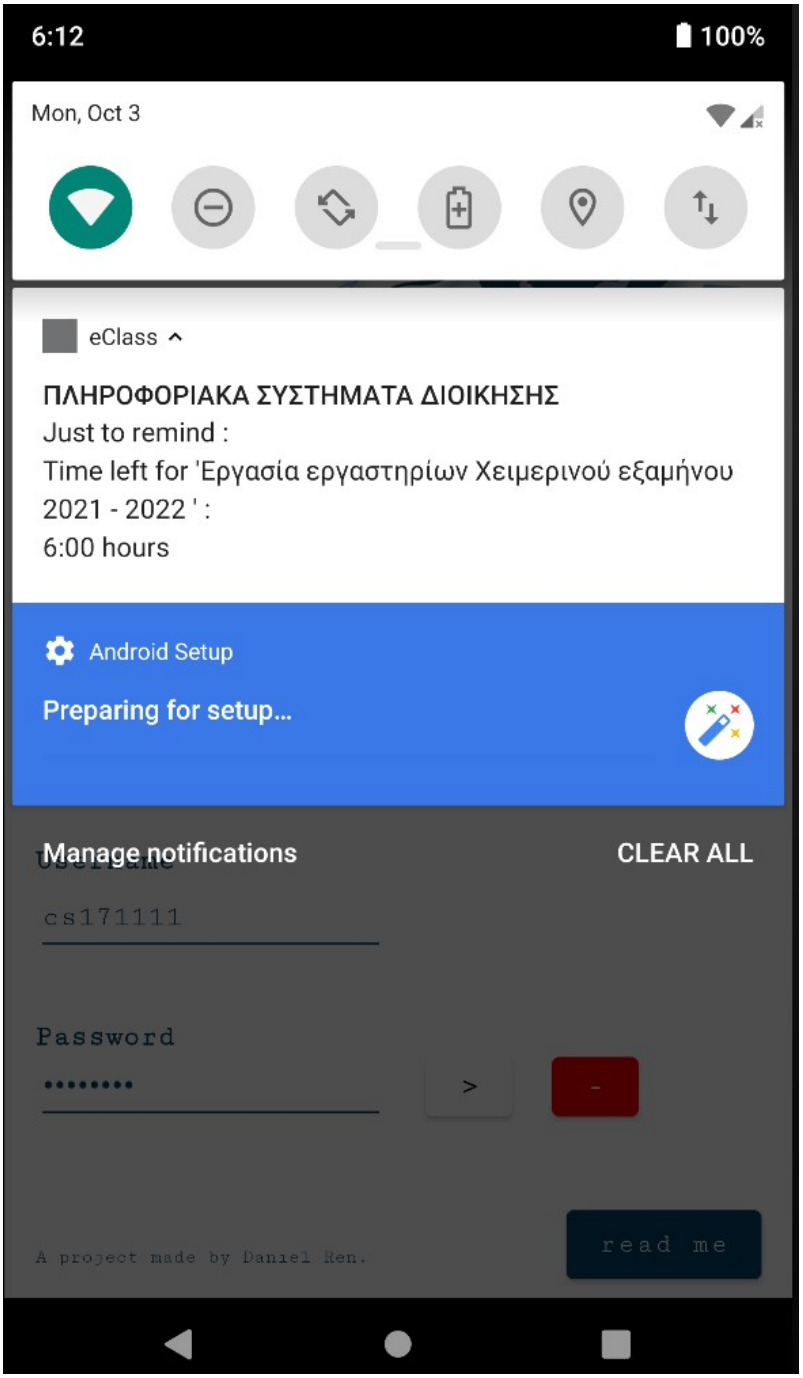


Εικόνα 8.9.3 : Ειδοποίηση για καινούργια ανακοίνωση

Το ίδιο περίπου συμβαίνει και στην περίπτωση καινούργιας εργασίας, με τη διαφορά ότι γίνεται ένας παραπάνω έλεγχος. Στους περιοδικούς ελέγχους που κάνει ο Server κάθε ένα λεπτό γίνεται ένας τελευταίος έλεγχος σχετικά με τις διορίες των εργασιών. Εάν κάποια εργασία φτάσει στο όριο των τριών ημερών τότε εμφανίζεται μια διαφορετική ειδοποίηση στους χρήστες που σχετίζονται με αυτήν, λέγοντας τους ότι πρόκειται για υπενθύμιση ότι έχουν ακόμη τρεις μέρες διορία στην εργασία αυτήν. Το ίδιο συμβαίνει και για μία μέρα πριν, έξι ώρες πριν, τρις ώρες πριν, μια ώρα πριν.

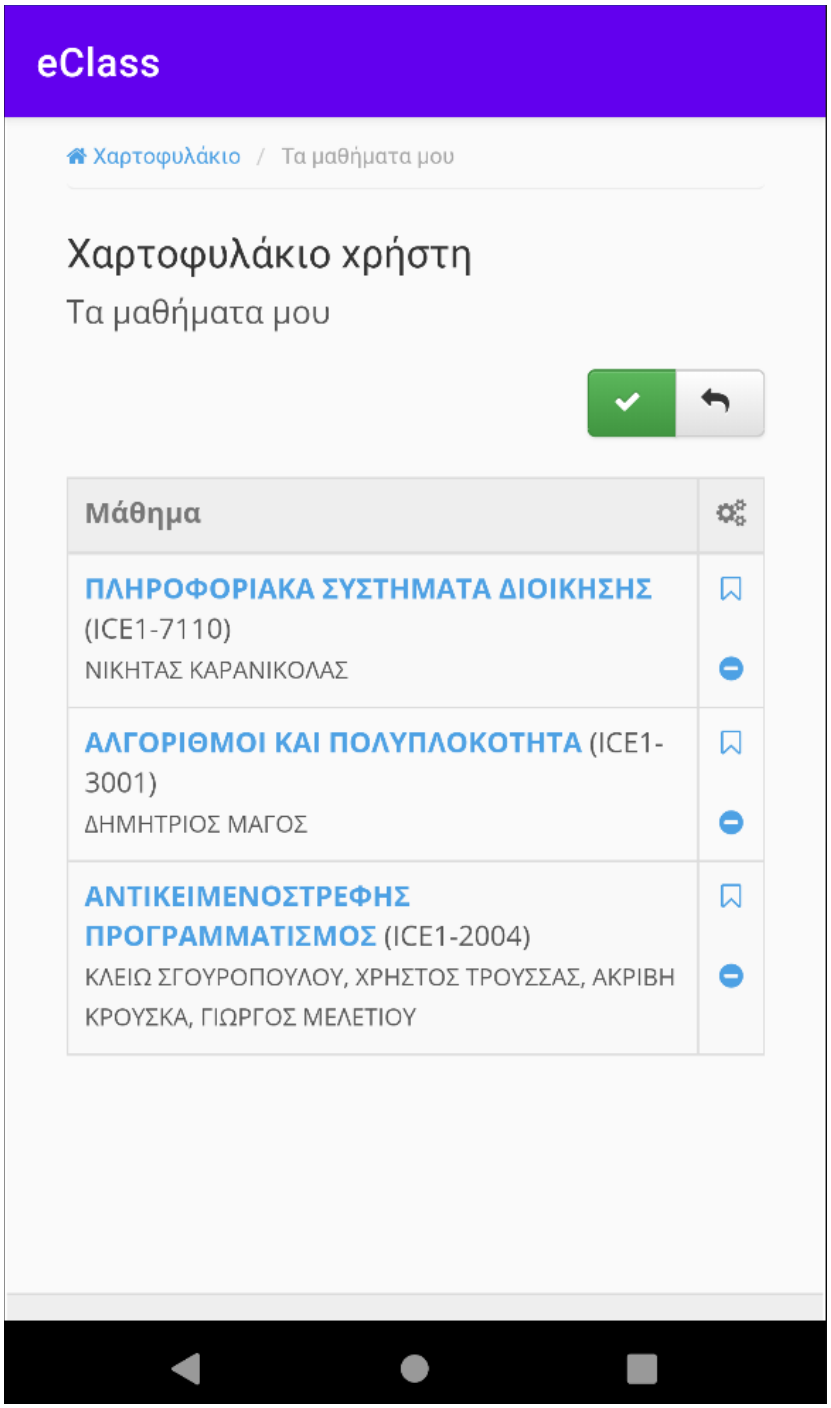


Εικόνα 8.9.4 : Ειδοποίηση για καινούργια εργασία



Εικόνα 8.9.5 : Υπενθύμιση ότι απέμειναν έξι ώρες για τη λήξη της εργασίας

Όταν ο χρήστης πατήσει πάνω σε αυτήν την ανακοίνωση (όπως και σε οποιαδήποτε άλλη ειδοποίηση αυτής της εφαρμογής) θα ανακατευθυνθεί στην αρχική σελίδα του eClass ήδη συνδεδεμένος από την εφαρμογή.



Εικόνα 8.9.6 : Μετά την ανακατεύθυνση που έγινε με το πάτημα μιας ανακοίνωσης της εφαρμογής

Παραρτήματα

Από τη μεριά του Server υπάρχει μόνο ένα αρχείο κώδικα python. Η Client εφαρμογή επειδή δημιουργήθηκε από το Android Studio το οποίο δημιουργεί αρκετά αρχεία, για λόγους πρακτικούς θα αναφερθούν μόνο τα βασικά από αυτά.

Παράρτημα 1

Server.py

Σε αυτό το αρχείο γίνεται όλες οι λειτουργίες του Server που αναφέρθηκαν σε προηγούμενα κεφάλαια.

```
import requests
from bs4 import BeautifulSoup
from flask import Flask, request
from flask_restful import Api
from SQLAlchemy import create_engine, MetaData, Table, Column, String, ForeignKey, text
import os.path
import json
from datetime import datetime
from apscheduler.schedulers.background import BackgroundScheduler


app = Flask(__name__)
api = Api(app)
newLessons = []
semaphore = True


'''
login() function connects to eClass as the "Server" account.
We supposes that the account we are using has access to all courses in the eClass.
We first send a get request so that necessary cookies will be created and after that we send the final post request with
the credentials.
'''
def login() :
    login_url = "https://eClass.uniwa.gr/"
    f = open("credentials.txt", 'r')
    username = f.readline()
    password = f.readline()
    f.close()

    payload = {
        "uname" : username,
        "pass" :        password,
        "submit" : "Είσοδος"
    }

    session = requests.Session()
    session.get(login_url)
    session.post(login_url, payload)

    return session


'''
pushNotification() function use's the Google Firebase Cloud messaging api.
We first created an account to Google Firebase, and allow the api use of the Server, a Server token is required so that
we can
```

```
send notification. We actually send a post request to Google fcm (Firebase Cloud messaging) link with specific headers and body,
the Server after that serves our request by sending the notification to the Client with the token we included in the body.
'''
def pushNotification(deviceToken, title, body) :
    ServerToken = "AAAAnDTAR58:APA91bFozBkqud99zdoorDY5cLAm3vXxWQNvqEFv7DIJKzOHv5z2H-yFV5jqDUJ6QcEXotbDAPwR5KHzoPEUsCKEqvi3bv6p0ilHLCWVz_t8rVBT6RfgYtIQn97VvZzBtG1ssyFAIfZm"

    headers = {
        "Content-Type" : "application/json",
        "Authorization" : "key = " + ServerToken
    }

    body = {
        "data": {
            "title" : title,
            "body" : body
        },
        "to":
            deviceToken
    }

    requests.post("https://fcm.Googleapis.com/fcm/send", headers = headers, data = json.dumps(body))


'''
manageStudentsDB() function manage the database.
If database doesn't exist; we created it, after we check if user who made the request exist in the database, if he doesn't exist,
we insert his info and his lessons info. If the user exist then we check for any updates in the lessons (maybe new course or deleted from older one).
If user want's to remove him self from the database then we first remove his record from studentsInfoTable and after that we remove any announcements or work which
it isn't referring to any student anymore.
'''
def manageStudentsDB(inputStudentInfo, inputLessonsNames, inputLessonsLinks) :
    global newLessons
    engine = create_engine("sqlite:///eClass.db", echo = False)
    meta = MetaData()
    print(inputStudentInfo[2])


    if (not os.path.exists("eClass.db")) :
        studentsInfoTable = Table (
            "studentsInfoTable",
            meta,
            Column("id", String, primary_key = True),
            Column("fullName", String),
            Column("deviceToken", String),
        )

        lessonsLinksTable = Table (
            "lessonsLinksTable",
            meta,
            Column("id", String, ForeignKey("studentsInfoTable.id")),
            Column("lessonName", String),
            Column("lessonLink", String),
        )

        announcementsTable = Table (
            "announcementsTable",
            meta,
            Column("lessonLink", String, ForeignKey("lessonsLinksTable.lessonLink")),
            Column("announcementLink", String),
            Column("announcementTitle", String),
            Column("announcementDate", String),
        )
```

```
        worksTable = Table (
            "worksTable",
            meta,
            Column("lessonLink", String, ForeignKey("lessonsLinksTable.lessonLink")),
            Column("workTitle", String),
            Column("workDeadLine", String),
            Column("workLink", String),
        )
        meta.create_all(engine)
    else :
        studentsInfoTable = Table (
            "studentsInfoTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        lessonsLinksTable = Table (
            "lessonsLinksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )
    conn = engine.connect()

    if (inputLessonsNames == "delete") :
        id_ = inputStudentInfo
        conn.execute(text("DELETE FROM studentsInfoTable WHERE id = :id"), id = id_)

    else :
        id_ = inputStudentInfo[0]
        studentExist = False
        for student in conn.execute(text("SELECT * FROM studentsInfoTable WHERE id = :id"), id = id_) :
            studentExist = True
        if (not studentExist) :
            conn.execute(text("INSERT INTO studentsInfoTable (id, fullName, deviceToken) VALUES
(:id, :fullName, :deviceToken)"), id = id_, fullName = inputStudentInfo[1], deviceToken = inputStudentInfo[2])

        for (inputLessonName, inputLessonLink) in zip(inputLessonsNames, inputLessonsLinks) :
            lessonLinkExist = False
            for sqlLessonLink in conn.execute(text("SELECT * FROM lessonsLinksTable WHERE id = :id AND lessonLink
= :lessonLink"), id = id_, lessonLink = inputLessonLink) :
                lessonLinkExist = True
            if (not lessonLinkExist) :
                conn.execute(text("INSERT INTO lessonsLinksTable (id, lessonName, lessonLink) VALUES (:id, :lessonName,
:lessonLink)"), id = id_, lessonName = inputLessonName, lessonLink = inputLessonLink)
                newLessons.append(inputStudentInfo[2] + inputLessonLink)

    for previousLessonLink in conn.execute(text("SELECT lessonLink FROM lessonsLinksTable WHERE id = :id"), id =
id_) :
        previousLessonLink = str(previousLessonLink)[2:-3]
        if (previousLessonLink not in inputLessonsLinks) :
            conn.execute(text("DELETE FROM lessonsLinksTable WHERE id = :id AND lessonLink = :lessonLink"), id = id_,
lessonLink = previousLessonLink)

    singleLessonsLinks = []
    for lessonLinkFromAnnouncementsTable in conn.execute(text("SELECT lessonLink FROM announcementsTable")) :
        lessonLinkFromAnnouncementsTable = str(lessonLinkFromAnnouncementsTable)[2:-3]
        if (lessonLinkFromAnnouncementsTable not in singleLessonsLinks) :
            singleLessonsLinks.append(lessonLinkFromAnnouncementsTable)
            lessonExist = False
            for lessonLink in conn.execute(text("SELECT * FROM lessonsLinksTable WHERE lessonLink = :lessonLink"),
lessonLink = lessonLinkFromAnnouncementsTable) :
                lessonExist = True
            if (not lessonExist) :
```

```
            conn.execute(text("DELETE FROM announcementsTable WHERE lessonLink = :lessonLink"), lessonLink =
lessonLinkFromAnnouncementsTable)

    singleLessonsLinks = []
    for lessonLinkFromWorksTable in conn.execute(text("SELECT lessonLink FROM worksTable")) :
        lessonLinkFromWorksTable = str(lessonLinkFromWorksTable)[2:-3]
        if (lessonLinkFromWorksTable not in singleLessonsLinks) :
            singleLessonsLinks.append(lessonLinkFromWorksTable)
            lessonExist = False
            for lessonLink in conn.execute(text("SELECT * FROM lessonsLinksTable WHERE lessonLink = :lessonLink"),
lessonLink = lessonLinkFromWorksTable) :
                lessonExist = True
            if (not lessonExist) :
                conn.execute(text("DELETE FROM worksTable WHERE lessonLink = :lessonLink"), lessonLink =
lessonLinkFromWorksTable)


'''
exportAnnouncements() function uses Beautiful Soup technology for Web Scraping the announcements of every lesson
in the eClass.
We first scrap all the current announcements and save it to an array, after for every announcement (from newer to older)
we check if it isn't in our database;
if it isn't then we insert it to our database and send a push notification to the every student who is listed in the specific
lesson of the announcement. Otherwise if the notification is in our database
it means that we already check it show the rest of the search stops.
Note that if the students is first time connecting to our app then we prevent pushing all the notification from the past, in
other words,
the new users want get any previous notification, they will get notification only from new announcements which were
created after the time the login to our app.
'''
def exportAnnouncements(session) :
    global newLessons
    engine = create_engine("sqlite:///eClass.db", echo = False)
    meta = MetaData()

    if (os.path.exists("eClass.db")) :
        studentsInfoTable = Table (
            "studentsInfoTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        lessonsLinksTable = Table (
            "lessonsLinksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        announcementsTable = Table (
            "announcementsTable",
            meta,
            autoload = True,
            autoload_with = engine
        )
        conn = engine.connect()

        payload = {
            "iDisplayStart" : 0,
            "iDisplayLength" : -1,
        }

        headers = {
            "X-Requested-With": "XMLHttpRequest",
        }
```

```
        singleLessonsLinks = []
        for sqlLessonLink in conn.execute(text("SELECT lessonLink FROM lessonsLinksTable")) :
            sqlLessonLink = str(sqlLessonLink)[2:-3]
            if (sqlLessonLink not in singleLessonsLinks) :
                singleLessonsLinks.append(sqlLessonLink)
                currentLessonLinks = []
                currentAnnouncementsLinks = []
                currentAnnouncementsTitles = []
                currentAnnouncementsDates = []
                maybeRestOfTitle = False
                courseId = sqlLessonLink.split('/')[-2]
                response = session.get("https://eClass.uniwa.gr/modules/announcements/?course=" + courseId, headers =
headers, data = payload).text.split("\"")
                announcementDateBlock = ""
                firstAnnouncementDateBlock = True
                titleOnNextLine = False
                i = 0
                for item in response :
                    announcementDateBlock += item

                    if (maybeRestOfTitle) :
                        if ("table_td_body" not in item) :
                            currentAnnouncementsTitles[-1] = currentAnnouncementsTitles[-1] + " " + item.split('<')[0]
[1:].replace('\\', '')
                        maybeRestOfTitle = False

                    if (titleOnNextLine) :
                        currentLessonLinks.append(sqlLessonLink)
                        currentAnnouncementsTitles.append(item.split('<')[0][1:].replace('\\', ''))
                        titleOnNextLine = False
                        maybeRestOfTitle = True

                    if ("\/modules\/announcements\/index.php" in item) :
                        currentAnnouncementsLinks.append("https://eClass.uniwa.gr" + item.replace('\\', ''))
                        titleOnNextLine = True

                        if (not firstAnnouncementDateBlock) :
                            currentAnnouncementsDates.append(announcementDateBlock.split("div")[-3].split("\"")[2])
                            announcementDateBlock = ""
                        else :
                            firstAnnouncementDateBlock = not firstAnnouncementDateBlock

                currentAnnouncementsDates.append(announcementDateBlock.split("div")[-1].split("\"")[2])

                stopSearching = False
                for currentAnnouncementLink in currentAnnouncementsLinks :
                    for sqlAnnouncementLink in conn.execute(text("SELECT * FROM announcementsTable WHERE
announcementLink = :announcementLink"), announcementLink = currentAnnouncementLink) :
                        stopSearching = True
                    if (stopSearching) :
                        break
                    else :
                        conn.execute(text("INSERT INTO announcementsTable (lessonLink, announcementLink,
announcementTitle, announcementDate) VALUES
(:lessonLink, :announcementLink, :announcementTitle, :announcementDate)"), lessonLink = currentLessonLinks[i],
announcementLink = currentAnnouncementsLinks[i], announcementTitle = currentAnnouncementsTitles[i],
announcementDate = currentAnnouncementsDates[i])
                        for lessonName in conn.execute(text("SELECT lessonName FROM lessonsLinksTable WHERE
lessonLink = :lessonLink"), lessonLink = currentLessonLinks[i]) :
                            lessonName = str(lessonName)[2:-3]
                        page = session.get(currentAnnouncementsLinks[i])
                        soup = BeautifulSoup(page.text, "html.parser")
                        div = soup.find(class_ = "announcement-main")
                        announcementBody = currentAnnouncementsTitles[i] + "\n" + div.text

                        for studentId in conn.execute(text("SELECT id FROM lessonsLinksTable WHERE lessonLink
= :lessonLink"), lessonLink = currentLessonLinks[i]) :
```

```
                        studentId = str(studentId)[2:-3]
                        for deviceToken in conn.execute(text("SELECT deviceToken FROM studentsInfoTable WHERE id
= :id"), id = studentId) :
                            deviceToken = str(deviceToken)[2:-3]
                        if (deviceToken + currentLessonLinks[i] not in newLessons) :
                            pushNotification(deviceToken, lessonName, announcementBody)
                    i += 1


'''
exportWorks() function is pretty much same as exportAnnouncements()
We first scrap and save all the works from the eClass and check if we have any new work (if it isn't in our database), if
yes we update the database with it
and send a push notification to the students who are listed in this lesson.
Same goes here the new users, the won't get notification which are older by the time the first login in to our app.
'''
def exportWorks(session) :
    global newLessons
    engine = create_engine("sqlite:///eClass.db", echo = False)
    meta = MetaData()

    if (os.path.exists("eClass.db")) :
        studentsInfoTable = Table (
            "studentsInfoTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        lessonsLinksTable = Table (
            "lessonsLinksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        worksTable = Table (
            "worksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )
        conn = engine.connect()

        singleLessonsLinks = []
        for sqlLessonLink in conn.execute(text("SELECT lessonLink FROM lessonsLinksTable")) :
            sqlLessonLink = str(sqlLessonLink)[2:-3]
            if (sqlLessonLink not in singleLessonsLinks) :
                singleLessonsLinks.append(sqlLessonLink)
                currentLessonLinks = []
                currentWorksTitlesAndDeadLines = []
                currentWorksLinks = []
                maybeRestOfTitle = False
                i = 0
                courseId = sqlLessonLink.split('/')[-2]
                page = session.get("https://eClass.uniwa.gr/modules/work/?course=" + courseId)
                soup = BeautifulSoup(page.text, "html.parser")
                table = soup.find_all("table", {"id": "assignment_table"})

                for item in table :
                    for tr in item.find_all("tr") :
                        firstTwoTd = 0
                        for td in tr.find_all("td") :
                            td = td.text
                            if (len(td)) :
                                if (firstTwoTd < 2) :
                                    if (maybeRestOfTitle) :
```

```
                        currentWorksTitlesAndDeadLines[-1] = currentWorksTitlesAndDeadLines[-1] + "\n" + td
                        maybeRestOfTitle = False
                        currentLessonLinks.append(sqlLessonLink)
                    else :
                        currentWorksTitlesAndDeadLines.append(td)
                        maybeRestOfTitle = True
                    firstTwoTd += 1

            for td in item.find_all(href=True) :
                currentWorksLinks.append("https://eClass.uniwa.gr" + td["href"])

        stopSearching = False
        for currentWorkLink in currentWorksLinks :
            for sqlWorkLink in conn.execute(text("SELECT * FROM worksTable WHERE workLink = :workLink"),
workLink = currentWorkLink) :
                stopSearching = True
            if (stopSearching) :
                break
            else :
                currentWorkTitle = currentWorksTitlesAndDeadLines[i].split("\n")[0]
                currentWorkDeadLine = currentWorksTitlesAndDeadLines[i].split("\n")[1].split('(')[0]
                conn.execute(text("INSERT INTO worksTable (lessonLink, workTitle, workDeadLine, workLink) VALUES
(:lessonLink, :workTitle, :workDeadLine, :workLink)"), lessonLink = currentLessonLinks[i], workTitle = currentWorkTitle,
workDeadLine = currentWorkDeadLine, workLink = currentWorksLinks[i])
                for lessonName in conn.execute(text("SELECT lessonName FROM lessonsLinksTable WHERE
lessonLink = :lessonLink"), lessonLink = currentLessonLinks[i]) :
                    lessonName = str(lessonName)[2:-3]

                for studentId in conn.execute(text("SELECT id FROM lessonsLinksTable WHERE lessonLink
= :lessonLink"), lessonLink = currentLessonLinks[i]) :
                    studentId = str(studentId)[2:-3]
                    for deviceToken in conn.execute(text("SELECT deviceToken FROM studentsInfoTable WHERE id
= :id"), id = studentId) :
                        deviceToken = str(deviceToken)[2:-3]
                    if (deviceToken + currentLessonLinks[i] not in newLessons) :
                        pushNotification(deviceToken, lessonName, currentWorksTitlesAndDeadLines[i])
                i += 1
    newLessons = []


'''
worksReminder() function if responsible for reminding to user (with push notification) for an event on specific times.
The times are :
3 days (exactly) before the deadline
1 day before
6 hours before
3 hours before
1 hour before
'''
def worksReminder() :
    engine = create_engine("sqlite:///eClass.db", echo = False)
    meta = MetaData()

    if (os.path.exists("eClass.db")) :
        studentsInfoTable = Table (
            "studentsInfoTable",
            meta,
            autoload = True,
            autoload_with = engine
        )

        lessonsLinksTable = Table (
            "lessonsLinksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )
```

```
        worksTable = Table (
            "worksTable",
            meta,
            autoload = True,
            autoload_with = engine
        )
        conn = engine.connect()

        response = conn.execute(text("SELECT lessonLink, workTitle, workDeadLine FROM worksTable"))
        for item in response :
            lessonLink = str(item).split(',')[0].split("'\'")[1]
            workTitle = str(item).split(',')[1].split("'\'")[1]
            workDeadline = str(item).split(',')[2].split("'\'")[1]

            dateTimeNow = datetime.now()
            dayDeadLine = workDeadline.split(' ')[0].split('-')
            hourDeadLine = workDeadline.split(' ')[1].split(':')
            deadLineDateTime = datetime(day = int(dayDeadLine[0]), month = int(dayDeadLine[1]), year =
int(dayDeadLine[2]), hour = int(hourDeadLine[0]), minute = int(hourDeadLine[1]), second = int(hourDeadLine[2]))
            timeLeft = str(deadLineDateTime - dateTimeNow).split(',')
            daysLeft = timeLeft[0].split(' ')[0]
            hoursAndMinsLeft = (timeLeft[1].split(':')[0] + ':' + timeLeft[1].split(':')[1]).replace(' ', '')

            if (daysLeft == '3' or daysLeft == '1' or daysLeft == '0') :
                reminderBody = "Just to remind :\nTime left for '" + workTitle + "' :\n"
                deviceToken = ""
                for innerResponce in conn.execute(text("SELECT id, lessonName FROM lessonsLinksTable WHERE
lessonLink = :lessonLink"), lessonLink = lessonLink) :
                    studentId = str(innerResponce).split(',')[0].split("'\'")[1]
                    lessonName = str(innerResponce).split(',')[1].split("'\'")[1]
                    for deviceToken in conn.execute(text("SELECT deviceToken FROM studentsInfoTable WHERE id = :id"), id
= studentId) :
                        deviceToken = str(deviceToken)[2:-3]
                    if (daysLeft == '0') :
                        if (hoursAndMinsLeft == "6:00" or hoursAndMinsLeft == "3:00" or hoursAndMinsLeft == "1:00") :
                            reminderBody += hoursAndMinsLeft
                    else :
                        if (hoursAndMinsLeft == "6:00") :
                            reminderBody += daysLeft + " days and " + hoursAndMinsLeft + " hours."
                    pushNotification(deviceToken, lessonName, reminderBody)


'''
mainService() function purpose is to run every 1 minute, what it does is scraping for new announcements and works from
all lesson in eClass
and inform the specific users. We user a "semaphore" variable to avoid any conflict in the database access.
mainService() function is allowed to access the database only if no other function is using it (like a new login from a
user).
Same goes for the other functions, they are not allowed to use the database until mainService() is done.
'''
def mainService() :
    global semaphore
    if (semaphore) :
        try :
            semaphore = False
            session = login()
            exportAnnouncements(session)
            exportWorks(session)
            worksReminder()
            semaphore = True
        except :
            semaphore = True


'''
postUserData() is called from the Client side with HTTPS POST request.
```

```
Clients send 3 arrays :
studentInfo (contains : id, fullName, deviceToken*)
lessonsNames (contains : the names of the lessons which its signed to)
lessonsLinks (contains : the links of the lessons which its signed to)
deviceToken* is basically a long id which is use for the push notification service.
'''
@app.route("/postUserData", methods = ["POST"])
def postUserData() :
    global semaphore
    studentInfo = request.form["studentInfo"].split(", ")
    lessonsNames = request.form["lessonsNames"].split(", ")
    lessonsLinks = request.form["lessonsLinks"].split(", ")

    while (True) :
        if (semaphore) :
            try :
                semaphore = False
                manageStudentsDB(studentInfo, lessonsNames, lessonsLinks)
                session = login()
                exportAnnouncements(session)
                exportWorks(session)
                semaphore = True
                return "All ok."
            except :
                semaphore = True
                break


'''
deleteUserData() behavior is pretty much the same as postUserData().
Client call's it with HTTPS POST method but this time the arrays are :
studentInfo (contains : only the student id)
lessonsNames (contains : the phrase "delete")
lessonsLinks (contains : the phrase "delete")
so with this data we let the manageStudentsDB() function know that the user want's to delete himself from the database.
'''
@app.route("/deleteUserData", methods = ["POST"])
def deleteUserData() :
    global semaphore
    studentInfo = request.form["studentInfo"]
    lessonsNames = request.form["lessonsNames"]
    lessonsLinks = request.form["lessonsLinks"]

    while (True) :
        if (semaphore) :
            try :
                semaphore = False
                manageStudentsDB(studentInfo, lessonsNames, lessonsLinks)
                semaphore = True
                return "All ok."
            except :
                semaphore = True
                break


# Create the BackgroundScheduler so that mainService() will now be executed every 1 minute.
scheduler = BackgroundScheduler()
scheduler.add_job(func = mainService, trigger = "interval", seconds = 60)
scheduler.start()


# Our Server will run in the pc ip (0.0.0.0 means pc ip like 192.168.2.105) and in the 9999 port.
if (__name__ == "__main__") :
    app.run(host = "0.0.0.0", port = "9999")
```

Παράρτημα 2

MainActivity.java

Ο κώδικας αυτός αποτελεί την κύρια λειτουργία της Client εφαρμογής. Μερικές από τις βασικές λειτουργίες που επιτελεί είναι να ελέγχει για την ορθή σύνδεση (επικοινωνία) μεταξύ Server και Client, έλεγχος για τις απαιτούμενες Google υπηρεσίες που χρειάζονται, αποστολή δεδομένων από τον Client στο Server μέσω Volley τεχνολογίας κ.α.

```
package com.example.eClass;

import Android.app.Activity;
import Android.app.AlertDialog;
import Android.content.Context;
import Android.content.DialogInterface;
import Android.content.Intent;
import Android.content.SharedPreferences;
import Android.net.ConnectivityManager;
import Android.net.NetworkInfo;
import Android.os.AsyncTask;
import Android.os.Bundle;
import Android.view.View;
import Android.view.inputmethod.InputMethodManager;
import Android.widget.Button;
import Android.widget.EditText;
import Android.widget.TextView;
import Android.widget.Toast;

import Androidx.annotation.NonNull;
import Androidx.appcompat.app.AppCompatActivity;

import com.Android.volley.DefaultRetryPolicy;
import com.Android.volley.Request;
import com.Android.volley.RequestQueue;
import com.Android.volley.Response;
import com.Android.volley.RetryPolicy;
import com.Android.volley.toolbox.StringRequest;
import com.Android.volley.toolbox.Volley;
import com.Google.Android.gms.common.ConnectionResult;
import com.Google.Android.gms.common.GoogleApiAvailability;
import com.Google.Android.gms.tasks.OnCompleteListener;
import com.Google.Android.gms.tasks.Task;
import com.Google.Firebase.messaging.FirebaseMessaging;

import org.jsoup.Connection;
import org.jsoup.Jsoup;
import org.jsoup.nodes.Document;
import org.jsoup.nodes.Element;
import org.jsoup.select.Elements;

import java.io.IOException;
import java.nio.charset.StandardCharsets;
import java.util.HashMap;
import java.util.Map;
import java.util.concurrent.ExecutionException;

public class MainActivity extends AppCompatActivity {
    String studentInfo = "";
    String lessonsLinks = "";
    String lessonsNames = "";
    String deviceToken = "";

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

```
        getSupportActionBar().hide();
        TextView usernameField = findViewById(R.id.usernameField);
        TextView passwordField = findViewById(R.id.passwordField);
        Button loginBtn = findViewById(R.id.loginBtn);
        Button logoutBtn = findViewById(R.id.logoutBtn);
        Button readMeBtn = findViewById(R.id.readMeBtn);

        SharedPreferences sharedPreferences = getSharedPreferences("studentCredentials", MODE_PRIVATE);
        SharedPreferences.Editor editor = sharedPreferences.edit();
        String previousId = sharedPreferences.getString("id", null);

        if (previousId != null) {
            usernameField.setText(previousId);
            passwordField.setText(sharedPreferences.getString("password", null));
            logoutBtn.setVisibility(View.VISIBLE);
        }

        if (checkGooglePlayServicesAvailable()) {
            FirebaseMessaging.getInstance().getToken()
                    .addOnCompleteListener(new OnCompleteListener<String>() {
                        @Override
                        public void onComplete(@NonNull Task<String> task) {
                            if (!task.isSuccessful()) {
                                return;
                            }
                            // Get new FCM registration token
                            deviceToken = task.getResult();
                        }
                    });

            loginBtn.setOnClickListener(new View.OnClickListener() {
                @Override
                public void onClick(View view) {
                    studentInfo = "";
                    lessonsLinks = "";
                    lessonsNames = "";
                    String currentId = usernameField.getText().toString();
                    String currentPassword = passwordField.getText().toString();
                    String messageToUser = "Unexpected error.";
                    String previousId = sharedPreferences.getString("id", null);

                    try {
                        ConnectivityManager manager = (ConnectivityManager)
getApplicationContext().getSystemService(Context.CONNECTIVITY_SERVICE);
                        NetworkInfo activeNetwork = manager.getActiveNetworkInfo();
                        if (activeNetwork == null) {
                            messageToUser = "Internet connection error.";
                        }
                        else {
                            if ((!currentId.equals(previousId)) && (previousId != null)) {
                                messageToUser = "Please logout first from the previous account.";
                            }
                            else {
                                messageToUser = new Login().execute(currentId, currentPassword).get();
                                if (messageToUser.contains("be patient")) {
                                    postDataUsingVolley("http://192.168.2.105:9999/postUserData", editor, logoutBtn, usernameField,
passwordField, currentId, currentPassword);
                                }
                            }
                        }
                    } catch (ExecutionException | InterruptedException e) {
                        e.printStackTrace();
                    }
                    showToast(messageToUser);
                }
            });
```

```
        logoutBtn.setOnClickListener(new View.OnClickListener() {
            @Override
            public void onClick(View view) {
                AlertDialog.Builder builder = new AlertDialog.Builder(view.getContext());
                builder.setTitle("Confirm");
                builder.setMessage("Are you sure you want to logout?");
                builder.setCancelable(false);
                builder.setPositiveButton("Yes", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        String previousId = sharedPreferences.getString("id", null);
                        postDataUsingVolley("http://192.168.2.105:9999/deleteUserData", editor, logoutBtn, usernameField,
passwordField, previousId, "");
                        dialogInterface.dismiss();
                    }
                });
                builder.setNegativeButton("Cancel", new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialogInterface, int i) {
                        dialogInterface.dismiss();
                    }
                });
                builder.show();
            }
        });
    }
    else {
        denyAccess();
    }
    readMeBtn.setOnClickListener(new View.OnClickListener() {
        @Override
        public void onClick(View view) {
            DialogFragment dialogFragment = new DialogFragment();
            dialogFragment.show(getSupportFragmentManager(), "Read me fragment");
        }
    });
}


public class Login extends AsyncTask<String, String, String> {
    String response = "";
    @Override
    protected String doInBackground(String... params) {
        String homePageUrl = "https://eClass.uniwa.gr/";
        String userAgent = "python-requests/2.25.1";

        HashMap<String, String> getCookies = new HashMap<>();
        HashMap<String, String> postCookies = new HashMap<>();
        HashMap<String, String> postFormData = new HashMap<>();

        try {
            Connection.Response getConn =
Jsoup.connect(homePageUrl).method(Connection.Method.GET).userAgent(userAgent).execute();
            getCookies.putAll(getConn.cookies()); // save the cookies, this will be passed on to next request

            postFormData.put("uname", params[0]);
            postFormData.put("pass", params[1]);
            postFormData.put("submit", "Είσοδος");
            Connection.Response postConn = Jsoup.connect(homePageUrl)
                    .cookies(getCookies)
                    .data(postFormData)
                    .method(Connection.Method.POST)
                    .userAgent(userAgent)
                    .header("Accept-Encoding", "gzip, deflate")
                    .header("Accept", "*/*")
                    .header("Connection", "keep-alive")
                    .execute();
```

```
            postCookies.putAll(postConn.cookies()); // save the cookies, this will be passed on to next request

            Document studentInfoDoc = Jsoup.connect("https://eClass.uniwa.gr/main/profile/display_profile.php")
                    .cookies(postCookies)
                    .get();

            studentInfo += params[0] + ", ";
            studentInfo += studentInfoDoc.select("div.profile-name").text() + ", ";
            studentInfo += deviceToken;

            Document lessonsDoc = Jsoup.connect("https://eClass.uniwa.gr/main/my_courses.php")
                    .cookies(postCookies)
                    .get();

            Elements linksElements = lessonsDoc.select("div.table-responsive").select("a[href]");
            for (Element linksElement : linksElements) {
                String stringLessonLink = linksElement.attr("href");
                if (stringLessonLink.contains("courses")) {
                    lessonsLinks += stringLessonLink + ", ";
                }
            }
            if (!lessonsLinks.isEmpty()) {
                lessonsLinks = lessonsLinks.substring(0, lessonsLinks.length() - 2);
            }

            Elements nameElements = lessonsDoc.select("div.table-responsive").select("a");
            for (Element nameElement : nameElements) {
                if (!nameElement.text().equals("")) {
                    lessonsNames += nameElement.text() + ", ";
                }
            }
            if (!lessonsNames.isEmpty()) {
                lessonsNames = lessonsNames.substring(0, lessonsNames.length() - 2);
            }

            if (lessonsLinks.isEmpty()) {
                response = "Due to security reasons please wait some seconds before you try a new login.";
            }
            else {
                response = "Doing our stuff, please be patient... :)";
            }
        } catch (IOException e) {
            response = "Unexpected Error.";
        }
        return response;
    }
}


    private void postDataUsingVolley(String url, SharedPreferences.Editor editor, Button logoutBtn, TextView
usernameField, TextView passwordField, String id, String password) {
        StringRequest request = new StringRequest(Request.Method.POST, url,
                response -> {
                    if (response.equals("All ok.")) {
                        if (url.contains("postUserData")) {
                            manageCredentials(editor, id, password);
                            logoutBtn.setVisibility(View.VISIBLE);
                            hideKeyboard(getApplicationContext(), getCurrentFocus());
                            showToast("Welcome " + studentInfo.split(", ")[1] + "!");
                        } else {
                            editor.clear().apply();
                            usernameField.setText("cs");
                            passwordField.setText("");
                            logoutBtn.setVisibility(View.GONE);
                            showToast("You have successfully logged out.");
                        }
                    }
```

```
            },
            error -> showToast("Unexpected error. \nCan't communicate with the Server.")) {

        @Override
        protected Map<String, String> getParams() {
            // below line we are creating a map for
            // storing our values in key and value pair.
            if (url.contains("deleteUserData")) {
                studentInfo = id;
                lessonsLinks = "delete";
                lessonsNames = "delete";
            }
            Map<String, String> params = new HashMap<>();
            // on below line we are passing our key
            // and value pair to our parameters.
            params.put("studentInfo", studentInfo);
            params.put("lessonsLinks", lessonsLinks);
            params.put("lessonsNames", lessonsNames);

            // at last we are
            // returning our params.
            return params;
        }
    };
    // below line is to make
    // a json object request.
    request.setRetryPolicy(new DefaultRetryPolicy(600000, 0, DefaultRetryPolicy.DEFAULT_BACKOFF_MULT));
    RequestQueue queue = Volley.newRequestQueue(MainActivity.this);
    queue.add(request);
}


private void denyAccess() {
    Button loginBtn = findViewById(R.id.loginBtn);
    loginBtn.setVisibility(View.GONE);
    TextView GooglePlayServicesWarning = findViewById(R.id.GooglePlayServicesWarning);
    GooglePlayServicesWarning.setText("Sorry but the application can't use\n\"Push Notification\" service without \
nGoogle Play Services installed on your device.");
}


private void manageCredentials(SharedPreferences.Editor editor, String id, String password) {
    editor.putString("id", id);
    editor.putString("password", password);
    editor.apply();
}

private boolean checkGooglePlayServicesAvailable() {
    final int status = GoogleApiAvailability.getInstance().isGooglePlayServicesAvailable(getApplicationContext());
    if (status == ConnectionResult.SUCCESS) {
        return true;
    }
    return false;
}

public void hideKeyboard(Context context, View view) {
    InputMethodManager inputMethodManager = (InputMethodManager)
context.getSystemService(Activity.INPUT_METHOD_SERVICE);
    if (view != null) {
        inputMethodManager.hideSoftInputFromWindow(view.getWindowToken(), 0);
    }
}

private void showToast(String message) {
    Toast.makeText(getApplicationContext(), message, Toast.LENGTH_LONG).show();
}
}
```

Παράρτημα 3

PushNotificationService.java

Ο παρακάτω κώδικας έχει αναλάβει την αποδοχή και τη διαχείριση των ειδοποιήσεων που στέλνει ο Server της Google Firebase Cloud. Επίσης με το πάτημα μίας τέτοια ειδοποίησης καλεί την “GoToBrowser” κλάση που αναφέρεται στο επόμενο παράρτημα.

```
package com.example.eClass;

import Android.app.Notification;
import Android.app.NotificationChannel;
import Android.app.NotificationManager;
import Android.app.PendingIntent;
import Android.content.Intent;
import Android.net.Uri;
import Androidx.annotation.NonNull;
import Androidx.core.app.NotificationManagerCompat;
import com.Google.Firebase.messaging.FirebaseMessagingService;
import com.Google.Firebase.messaging.RemoteMessage;
import java.util.Random;

public class PushNotificationService extends FirebaseMessagingService {
    @Override
    public void onMessageReceived(@NonNull RemoteMessage remoteMessage) {
        String title = remoteMessage.getData().get("title");
        String body = remoteMessage.getData().get("body");
        int randomId = new Random().nextInt(1000);
        final String CHANNEL_ID = "channelIdHere";

        NotificationChannel notificationChannel = new NotificationChannel(
                CHANNEL_ID,
                "channelName",
                NotificationManager.IMPORTANCE_HIGH
        );

        Intent goToBrowserIntent = new Intent(PushNotificationService.this, GoToBrowser.class);
        PendingIntent pendingIntent = PendingIntent.getActivity(this, 0, goToBrowserIntent,
PendingIntent.FLAG_UPDATE_CURRENT);

        getSystemService(NotificationManager.class).createNotificationChannel(notificationChannel);
        Notification.Builder notification = new Notification.Builder(this, CHANNEL_ID)
                .setContentTitle(title)
                .setContentText(body)
                .setContentIntent(pendingIntent)
                .setStyle(new Notification.BigTextStyle().bigText(body))
                .setSmallIcon(R.drawable.ic_launcher_background)
                .setAutoCancel(true);

        NotificationManagerCompat.from(PushNotificationService.this).notify(randomId, notification.build());
        super.onMessageReceived(remoteMessage);
    }
}
```

Παράρτημα 4

GoToBrowser.java

Η “GoToBrowser” κλάση έχει ως σκοπό την ανακατεύθυνση του χρήστη στην αρχική σελίδα του eClass μέσω ενός bult-in browse χωρίς να χρειαστεί να βάλει ο χρήστης τα στοιχεία του ώστε να συνδεθεί. Αυτό επιτυγχάνεται χάρις τα αποθηκευμένα στοιχεία login στοιχεία που έδωσε αρχικά ο χρήστης όταν επιτυχώς συνδέθηκε στην εφαρμογή. Τα στοιχεία του (username & password) αποθηκεύονται locally μόνο στο κινητό με τη βοήθεια του “SharedPreferences” Interface. Ύστερα πριν ακόμη ανοίξει ο browser, γίνεται ένα “αθόρυβο” login (δλδ. στο background) ώστε να δημιουργηθεί ένα valid cookie. Τέλος γίνεται load η αρχική σελίδα του eClass όμως πια με το cookie που πριν λίγο δημιουργήθηκε και έτσι δε χρειάζεται να ζητηθούν τα login στοιχεία.

```
package com.example.eClass;

import Androidx.appcompat.app.AppCompatActivity;

import Android.content.SharedPreferences;
import Android.os.AsyncTask;
import Android.os.Bundle;
import Android.webkit.CookieManager;
import Android.webkit.WebSettings;
import Android.webkit.WebView;
import Android.webkit.WebViewClient;

import org.jsoup.Connection;
import org.jsoup.Jsoup;

import java.io.IOException;
import java.util.HashMap;
import java.util.concurrent.ExecutionException;

public class GoToBrowser extends AppCompatActivity {

    @Override
    protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.activity_go_to_browser);

        SharedPreferences sharedPreferences = getSharedPreferences("studentCredentials", MODE_PRIVATE);
        String username = sharedPreferences.getString("id", null);
        String password = sharedPreferences.getString("password", null);
        String url = "https://eClass.uniwa.gr/main/my_courses.php";
        String cookie = "";

        try {
            Login login = new Login();
            cookie = login.execute(username, password).get();
            String[] splittedCookie = cookie.split("=");
            cookie = splittedCookie[1].substring(0, splittedCookie[1].length() - 1);
            System.out.println(cookie);
        } catch (ExecutionException e) {
            e.printStackTrace();
        } catch (InterruptedException e) {
            e.printStackTrace();
        }

        WebView webView = (WebView) findViewById(R.id.webView);
        webView.setWebViewClient(new WebViewClient());  // For letting us to navigate to eClass on our webview with the
specific cookies
        WebSettings webSettings = webView.getSettings();
        webSettings.setJavaScriptEnabled(true);
        CookieManager.getInstance().setCookie(url, "PHPSESSID=" + cookie + "; path=/");
        webView.loadUrl(url);
```

```
    }

    public class Login extends AsyncTask<String, String, String> {
        String cookies = "";
        @Override
        protected String doInBackground(String... params) {
            String homePageUrl = "https://eClass.uniwa.gr/";
            String userAgent = "python-requests/2.25.1";

            HashMap<String, String> getCookies = new HashMap<>();
            HashMap<String, String> postFormData = new HashMap<>();

            try {
                Connection.Response getConn =
Jsoup.connect(homePageUrl).method(Connection.Method.GET).userAgent(userAgent).execute();
                getCookies.putAll(getConn.cookies()); // save the cookies, this will be passed on to next request

                postFormData.put("uname", params[0]);
                postFormData.put("pass", params[1]);
                postFormData.put("submit", "Είσοδος");
                Connection.Response postConn = Jsoup.connect(homePageUrl)
                        .cookies(getCookies)
                        .data(postFormData)
                        .method(Connection.Method.POST)
                        .userAgent(userAgent)
                        .header("Accept-Encoding", "gzip, deflate")
                        .header("Accept", "*/*")
                        .header("Connection", "keep-alive")
                        .execute();

                cookies = String.valueOf(postConn.cookies());
            } catch (IOException e) {
                e.printStackTrace();
            }
            return cookies;
        }
    }
}
```

Παράρτημα 5

DialogFragment.java

Η απλή αυτή κλάση έχει αναλάβει την εμφάνιση ενός παραθύρου (DialogFragment) όπως φαίνεται στην “Εικόνα 7.3.2”.

```
package com.example.eClass;

import Android.os.Bundle;
import Android.view.LayoutInflater;
import Android.view.View;
import Android.view.ViewGroup;

import Androidx.annotation.NonNull;
import Androidx.annotation.Nullable;

public class DialogFragment extends Androidx.fragment.app.DialogFragment {
    @Nullable
    @Override
    public View onCreateView(@NonNull LayoutInflater inflater, @Nullable ViewGroup container, @Nullable Bundle
savedInstanceState) {
        super.onCreateView(inflater, container, savedInstanceState);
        return inflater.inflate(R.layout.dialog_fragment, container, false);
    }
}
```

Παράρτημα 6

activity_go_to_browser.xml

Τα “.xml” αρχεία έχουν να κάνουν με την εμφάνιση αντικειμένων στην οθόνη της Android συσκευής του χρήστη. Στο παρόν αρχείο υπάρχει ένα “webview” το οποίο επιτρέπει την εμφάνιση ιστοσελίδων κατευθείαν από την εφαρμογή του χρήστη όταν αυτός πατήσει μια καινούργια ειδοποίηση και όχι ανοίγοντας άλλων εγκαταστημένο browser. Η λειτουργία αυτή γίνεται σε συνεργασία με την κλάση “GoToBrowser” που αναφέρθηκε στο παράρτημα 4.

```
WebView webView = (WebView) findViewById(R.id.webView);
webView.setWebViewClient(new WebViewClient());  // For letting us to navigate to eclass on our webview with the specific cookies
WebSettings webSettings = webView.getSettings();
webSettings.setJavaScriptEnabled(true);
CookieManager.getInstance().setCookie(url,  s1: "PHPSESSID=" + cookie + "; path=/");
webView.loadUrl(url);
```

Εικόνα Παραρτήματος 6 : Αρχικοποίηση και εκτέλεση του webview με τις κατάλληλες παραμέτρους

```
<?xml version="1.0" encoding="utf-8"?>
<Androidx.constraintlayout.widget.ConstraintLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    xmlns:app="http://schemas.Android.com/apk/res-auto"
    xmlns:tools="http://schemas.Android.com/tools"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"
    tools:context=".GoToBrowser">
    <WebView
        Android:id="@+id/webView"
        Android:layout_width="409dp"
        Android:layout_height="729dp"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />
</Androidx.constraintlayout.widget.ConstraintLayout>
```

Παράρτημα 7

activity_main.xml

Στο αρχείο βρίσκεται το κύριο μενού της Android εφαρμογής.

```
<?xml version="1.0" encoding="utf-8"?>
<Androidx.constraintlayout.widget.ConstraintLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    xmlns:app="http://schemas.Android.com/apk/res-auto"
    xmlns:tools="http://schemas.Android.com/tools"
    Android:background="@color/white"
    Android:layout_width="match_parent"
    Android:layout_height="match_parent"
    tools:context=".MainActivity">

    <ImageView
        Android:id="@+id/imageView"
        Android:layout_width="246dp"
        Android:layout_height="205dp"
        app:layout_constraintBottom_toBottomOf="@+id/welcomeTextView"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.96"
        app:layout_constraintStart_toEndOf="@+id/welcomeTextView"
        app:layout_constraintTop_toTopOf="@+id/welcomeTextView"
        app:layout_constraintVertical_bias="0.054"
        app:srcCompat="@drawable/pada_logo" />

    <TextView
        Android:id="@+id/welcomeTextView"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_marginStart="16dp"
        Android:layout_marginTop="24dp"
        Android:fontFamily="serif-monospace"
        Android:text="Welcome to eClass\nfor\nice.uniwa students\nonly!"
        Android:textColor="@color/pada_color"
        Android:textSize="15dp"
        Android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent" />

    <TextView
        Android:id="@+id/usernameTextView"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_marginStart="16dp"
        Android:layout_marginTop="332dp"
        Android:fontFamily="serif-monospace"
        Android:text="Username"
        Android:textColor="@color/pada_color"
        Android:textSize="15dp"
        Android:textStyle="bold"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toBottomOf="@+id/welcomeTextView" />

    <EditText
        Android:id="@+id/usernameField"
        Android:layout_width="184dp"
        Android:layout_height="43dp"
        Android:backgroundTint="@color/pada_color"
        Android:ems="10"
        Android:fontFamily="serif-monospace"
        Android:inputType="textPersonName"
        Android:text="cs"
        Android:textColor="@color/pada_color"
        Android:textSize="15dp"
        app:layout_constraintStart_toStartOf="@+id/usernameTextView"
```

```
    app:layout_constraintTop_toBottomOf="@+id/usernameTextView"
    tools:ignore="TouchTargetSizeCheck" />

<TextView
    Android:id="@+id/passwordTextView"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_marginTop="32dp"
    Android:fontFamily="serif-monospace"
    Android:textColor="@color/pada_color"
    Android:text="Password"
    Android:textSize="15dp"
    Android:textStyle="bold"
    app:layout_constraintStart_toStartOf="@+id/usernameField"
    app:layout_constraintTop_toBottomOf="@+id/usernameField" />

<EditText
    Android:id="@+id/passwordField"
    Android:layout_width="184dp"
    Android:layout_height="39dp"
    Android:backgroundTint="@color/pada_color"
    Android:ems="10"
    Android:inputType="textPassword"
    Android:textColor="@color/pada_color"
    app:layout_constraintStart_toStartOf="@+id/passwordTextView"
    app:layout_constraintTop_toBottomOf="@+id/passwordTextView"
    tools:ignore="SpeakableTextPresentCheck,TouchTargetSizeCheck" />

<TextView
    Android:id="@+id/creditsTextView"
    Android:layout_width="wrap_content"
    Android:layout_height="wrap_content"
    Android:layout_marginBottom="16dp"
    Android:fontFamily="serif-monospace"
    Android:textColor="@color/pada_color"
    Android:text="A project made by Daniel Ren."
    Android:textSize="10dp"
    app:layout_constraintBottom_toBottomOf="parent"
    app:layout_constraintEnd_toEndOf="parent"
    app:layout_constraintHorizontal_bias="0.068"
    app:layout_constraintStart_toStartOf="parent" />

<Button
    Android:id="@+id/loginBtn"
    Android:layout_width="45dp"
    Android:layout_height="43dp"
    Android:layout_marginStart="20dp"
    Android:backgroundTint="@color/cardview_light_background"
    Android:fontFamily="serif-monospace"
    Android:text=">"
    Android:textColor="@color/black"
    Android:textStyle="bold"
    app:layout_constraintBottom_toBottomOf="@+id/passwordField"
    app:layout_constraintStart_toEndOf="@+id/passwordField"
    app:layout_constraintTop_toTopOf="@+id/passwordField"
    app:layout_constraintVertical_bias="1.0"
    tools:ignore="TouchTargetSizeCheck" />

<Button
    Android:id="@+id/logoutBtn"
    Android:layout_width="45dp"
    Android:layout_height="43dp"
    Android:layout_marginStart="12dp"
    Android:textStyle="bold"
    Android:fontFamily="serif-monospace"
    Android:text="-"
    Android:backgroundTint="#FA0101"
```

```
        Android:visibility="gone"
        app:layout_constraintBottom_toBottomOf="@+id/loginBtn"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintHorizontal_bias="0.1"
        app:layout_constraintStart_toEndOf="@+id/loginBtn"
        app:layout_constraintTop_toTopOf="@+id/loginBtn"
        app:layout_constraintVertical_bias="0.0" />

    <TextView
        Android:id="@+id/GooglePlayServicesWarning"
        Android:layout_width="wrap_content"
        Android:layout_height="wrap_content"
        Android:layout_marginBottom="68dp"
        Android:fontFamily="serif-monospace"
        Android:text=""
        Android:textColor="@color/red"
        Android:textSize="12dp"
        app:layout_constraintBottom_toTopOf="@+id/creditsTextView"
        app:layout_constraintEnd_toEndOf="@+id/passwordField"
        app:layout_constraintHorizontal_bias="0.0"
        app:layout_constraintStart_toStartOf="@+id/passwordField"
        app:layout_constraintTop_toBottomOf="@+id/passwordField"
        app:layout_constraintVertical_bias="1.0" />

    <Button
        Android:id="@+id/readMeBtn"
        Android:layout_width="102dp"
        Android:layout_height="48dp"
        Android:layout_marginEnd="16dp"
        Android:layout_marginBottom="4dp"
        Android:backgroundTint="#074167"
        Android:fontFamily="serif-monospace"
        Android:textColor="@color/white"
        Android:text="read me"
        Android:textAllCaps="false"
        app:layout_constraintBottom_toBottomOf="parent"
        app:layout_constraintEnd_toEndOf="parent" />
</Androidx.constraintlayout.widget.ConstraintLayout>
```

Παράρτημα 8

dialog_fragment.xml

Με τον κώδικα αυτού του αρχείου και σε συνεργασία με την κλάση “DialogFragment” του παραρτήματος 5 γίνεται εφικτή η εμφάνιση του αντίστοιχου παραθύρου όταν ο χρήστης πατήσει το κουμπί “read me” που θα βρίσκεται κάτω δεξιά στην οθόνη του.

```
<?xml version="1.0" encoding="utf-8"?>
<Androidx.constraintlayout.widget.ConstraintLayout xmlns:Android="http://schemas.Android.com/apk/res/Android"
    xmlns:app="http://schemas.Android.com/apk/res-auto"
    xmlns:tools="http://schemas.Android.com/tools"
    Android:layout_width="match_parent"
    Android:background="#201E1E"
    Android:layout_height="match_parent">

    <ScrollView
        Android:layout_width="match_parent"
        Android:layout_height="wrap_content"
        app:layout_constraintEnd_toEndOf="parent"
        app:layout_constraintStart_toStartOf="parent"
        app:layout_constraintTop_toTopOf="parent">

        <RelativeLayout
            Android:layout_width="match_parent"
            Android:layout_height="wrap_content"
            Android:orientation="vertical">

            <TextView
                Android:id="@+id/textView"
                Android:layout_width="match_parent"
                Android:layout_height="match_parent"
                Android:layout_alignParentStart="true"
                Android:textSize="15dp"
                Android:layout_alignParentEnd="true"
                Android:layout_marginStart="0dp"
                Android:layout_marginEnd="0dp"
                Android:fontFamily="serif-monospace"
                Android:text="Welcome to eClass service for ice uniwa students.\nThis service is designed to inform its users as soon as possible about new announcements and assignments in the course they are enrolled in.\n\nEvery minute the Server checks for new announcements and new assignments in all the courses that exist in eClass and sends a notification to the users who are enrolled in that course.\nGoogle's Push Notification technology is used to send the notification.\n\nA prerequisite for the correct functioning of the application is that Google Play Services is installed and working properly.\nIt would also be correct to remove the app from battery restriction in order to receive the notification from the Server as soon as possible.\n"
                Android:textColor="@color/white" />
        </RelativeLayout>
    </ScrollView>
</Androidx.constraintlayout.widget.ConstraintLayout>
```

Παράρτημα 9

AndroidManifest.xml

Κάθε έργο εφαρμογής πρέπει να έχει ένα αρχείο AndroidManifest.xml (με αυτό ακριβώς το όνομα) στη ρίζα του συνόλου πηγών του έργου. Το αρχείο “manifest” περιγράφει βασικές πληροφορίες σχετικά με την εφαρμογή στα εργαλεία κατασκευής Android, στο λειτουργικό σύστημα Android και στο Google Play [21].

Μεταξύ πολλών άλλων πραγμάτων, το αρχείο “manifest” απαιτείται να δηλώνει τα εξής [21]:

- Τα στοιχεία της εφαρμογής, τα οποία περιλαμβάνουν όλες τις δραστηριότητες, τις υπηρεσίες, τους δέκτες εκπομπής και τους παρόχους περιεχομένου. Κάθε συστατικό (component) πρέπει να ορίζει βασικές ιδιότητες, όπως το όνομα της κλάσης Kotlin ή Java. Μπορεί επίσης να δηλώσει δυνατότητες, όπως ποιες διαμορφώσεις συσκευών μπορεί να χειριστεί και φίλτρα πρόθεσης που περιγράφουν τον τρόπο με τον οποίο μπορεί να ξεκινήσει το συστατικό.
- Τα δικαιώματα που χρειάζεται η εφαρμογή για να έχει πρόσβαση σε προστατευμένα τμήματα του συστήματος ή σε άλλες εφαρμογές. Επίσης, δηλώνει τυχόν δικαιώματα που πρέπει να έχουν άλλες εφαρμογές αν θέλουν να έχουν πρόσβαση σε περιεχόμενο από αυτή την εφαρμογή.
- Τα χαρακτηριστικά υλικού και λογισμικού που απαιτεί η εφαρμογή, τα οποία επηρεάζουν τις συσκευές που μπορούν να εγκαταστήσουν την εφαρμογή από το Google Play.

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:Android="http://schemas.Android.com/apk/res/Android"
   package="com.example.eClass">
   <uses-permission Android:name="Android.permission.INTERNET" />
   <uses-permission Android:name="Android.permission.ACCESS_NETWORK_STATE"/>
   <uses-permission Android:name="Android.permission.ACCESS_WIFI_STATE"/>
   <application
      Android:allowBackup="true"
      Android:icon="@mipmap/ic_launcher"
      Android:label="@string/app_name"
      Android:roundIcon="@mipmap/ic_launcher_round"
      Android:supportsRtl="true"
      Android:theme="@style/Theme.eClass"
      Android:usesCleartextTraffic="true">
      <activity
         Android:name=".GoToBrowser"
         Android:exported="false" />
      <activity
         Android:name=".MainActivity"
         Android:exported="true"
         Android:screenOrientation="portrait">
         <intent-filter>
            <action Android:name="Android.intent.action.MAIN" />

            <category Android:name="Android.intent.category.LAUNCHER" />
         </intent-filter>
      </activity>
      <service
         Android:name=".PushNotificationService"
         Android:exported="false">
         <intent-filter>
            <action Android:name="com.Google.Firebase.MESSAGING_EVENT" />
         </intent-filter>
      </service>    </application> </manifest>
```

Παράρτημα 10

build.gradle

Το λειτουργικό Android μεταγλωττίζει τους πόρους και τον πηγαίο κώδικα της εφαρμογής και τα πακετάρει σε APK ή Android App Bundles που μπορούν να δοκιμαστούν, αναπτυχθούν, υπογραφτούν και να διανεμειθούν. Το Android Studio χρησιμοποιεί το Gradle, μια προηγμένη εργαλειοθήκη δημιουργίας, επιτρέπει να ορίζονται ευέλικτες προσαρμοσμένες διαμορφώσεις δημιουργίας. Κάθε διαμόρφωση κατασκευής μπορεί να ορίζει το δικό της σύνολο κώδικα και πόρων, ενώ παράλληλα επαναχρησιμοποιεί τα μέρη που είναι κοινά για όλες τις εκδόσεις της εφαρμογής. Το πρόσθετο Android για το Gradle συνεργάζεται με την εργαλειοθήκη κατασκευής για να παρέχει διαδικασίες και διαμορφώσιμες ρυθμίσεις που αφορούν ειδικά την κατασκευή και τον έλεγχο εφαρμογών Android [22].

```
<?php
plugins {
id 'com.Android.application'
}

apply plugin: 'com.Google.gms.Google-services'
Android {
compileSdk 32

defaultConfig {
applicationId "com.example.eClass"
minSdk 28
targetSdk 32
versionCode 1
versionName "1.0"
testInstrumentationRunner "Androidx.test.runner.AndroidJUnitRunner"
}

buildTypes {
release {
minifyEnabled false
proguardFiles getDefaultProguardFile('proguard-Android-optimize.txt'), 'proguard-rules.pro'}
}
compileOptions {
sourceCompatibility JavaVersion.VERSION_1_8
targetCompatibility JavaVersion.VERSION_1_8
}
buildFeatures {
viewBinding true}
}

dependencies {
implementation 'Androidx.appcompat:appcompat:1.4.2'
implementation 'com.Google.Android.material:material:1.6.1'
implementation 'Androidx.constraintlayout:constraintlayout:2.1.4'
implementation 'com.Android.volley:volley:1.2.1'
implementation 'Androidx.lifecycle:lifecycle-livedata-ktx:2.5.0'
implementation 'Androidx.lifecycle:lifecycle-viewmodel-ktx:2.5.0'
implementation 'Androidx.navigation:navigation-fragment:2.5.0'
implementation 'Androidx.navigation:navigation-ui:2.5.0'
implementation 'org.jsoup:jsoup:1.13.1'
testImplementation 'junit:junit:4.13.2'
AndroidTestImplementation 'Androidx.test.ext:junit:1.1.3'
AndroidTestImplementation 'Androidx.test.espresso:espresso-core:3.4.0'
implementation platform('com.Google.Firebase:Firebase-bom:30.2.0')
implementation 'com.Google.Firebase:Firebase-analytics:21.1.0'
implementation 'com.Google.Firebase:Firebase-messaging:23.0.6'
}
?>
```

Παράρτημα 11

Google-services.json [13]

Το αρχείο Google-services.json τοποθετείται γενικά στον κατάλογο “app/” (στη ρίζα της ενότητας εφαρμογής του Android Studio). Από την έκδοση 2.2.0, το plugin υποστηρίζει αρχεία JSON για συγκεκριμένο τύπο κατασκευής.


```
{
  "project_info": {
    "project_number": "670899914655",
    "project_id": "eClass-875ce",
    "storage_bucket": "eClass-875ce.appspot.com"
  },
  "Client": [
    {
      "Client_info": {
        "mobilesdk_app_id": "1:670899914655:Android:ad7e265d00c2de23479faa",
        "Android_Client_info": {
          "package_name": "com.example.eClass"
        }
      },
      "oauth_Client": [
        {
          "Client_id": "670899914655-7t1n8sudpbng6cblppcsont73jeodush.apps.Googleusercontent.com",
          "Client_type": 1,
          "Android_info": {
            "package_name": "com.example.eClass",
            "certificate_hash": "be3ce5ff6d30fc6581f5d1d6ef2eec35bac77078"
          }
        },
        {
          "Client_id": "670899914655-liaqt63hpq2ucl2detl4p0v0rg9tsiru.apps.Googleusercontent.com",
          "Client_type": 3
        }
      ],
      "api_key": [
        {
          "current_key": "AIzaSyAoGWO6czzN_jb5BGjS-2hA8719axpn0ic"
        }
      ],
      "services": {
        "appinvite_service": {
          "other_platform_oauth_Client": [
            {
              "Client_id": "670899914655-liaqt63hpq2ucl2detl4p0v0rg9tsiru.apps.Googleusercontent.com",
              "Client_type": 3
            }
          ]
        }
      }
    }
  ],
  "configuration_version": "1"
}
```

Αναφορές

1. *How Many People Have Smartphones? [Apr 2022 Update]* | *Oberlo*. (n.d.). Retrieved October 24, 2022, from https://www.oberlo.com/statistics/how-many-people-have-smartphones

2. *Open eClass Documentation*. (n.d.). Retrieved October 4, 2022, from https://docs.openeclass.org/en/general

3. *Open eClass Documentation*. (n.d.). Retrieved October 4, 2022, from https://docs.openeclass.org/en/short_description

4. *Open eClass*. (n.d.). Retrieved October 4, 2022, from https://www.openeclass.org/

5. Android Studio. (2022). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Android_Studio&oldid=1114003015

6. *Download Android Studio & App Tools*. (n.d.). Android Developers. Retrieved October 4, 2022, from https://developer.android.com/studio

7. Python (programming language). (2022). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=Python_(programming_language)&oldid=1114003506

8. *Guido_van_Rossum_OSCON_2006_cropped.png (Εικόνα PNG, 1175 × 1762 εικονοστοιχεία)— Σε κλίμακα (49%)*. (n.d.). Retrieved October 4, 2022, from https://upload.wikimedia.org/wikipedia/commons/9/94/Guido_van_Rossum_OSCON_2006_cropped.png

9. *What is Java and why do I need it?* (n.d.). Retrieved October 4, 2022, from https://www.java.com/en/download/help/whatis_java.html

10. XML. (2022). In *Wikipedia*. https://en.wikipedia.org/w/index.php?title=XML&oldid=1112984754

11. *SQLAlchemy—Introduction*. (n.d.). Retrieved October 4, 2022, from https://www.tutorialspoint.com/sqlalchemy/sqlalchemy_introduction.htm

12. *SQLAlchemy Core—Using Textual SQL*. (n.d.). Retrieved October 4, 2022, from https://www.tutorialspoint.com/sqlalchemy/sqlalchemy_core_using_textual_sql.htm

13. *Firebase Cloud Messaging*. (n.d.). Firebase. Retrieved October 4, 2022, from https://firebase.google.com/docs/cloud-messaging

14. *Set up a Firebase Cloud Messaging client app on Android*. (n.d.). Retrieved October 4, 2022, from https://firebase.google.com/docs/cloud-messaging/android/client

15. *Python and REST APIs: Interacting With Web Services – Real Python*. (n.d.). Retrieved October 4, 2022, from https://realpython.com/api-integration-in-python/

16. Stud, P. (2020, September 30). What is WSGI (Web Server Gateway Interface)? *Analytics Vidhya*. https://medium.com/analytics-vidhya/what-is-wsgi-web-server-gateway-interface-ed2d290449e

17. *WSGI Servers*. (n.d.). Retrieved October 4, 2022, from https://www.fullstackpython.com/wsgi-servers.html

18. *Gunicorn—WSGI server—Gunicorn 20.1.0 documentation*. (n.d.). Retrieved October 4, 2022, from https://docs.gunicorn.org/en/stable/

19. *Installation—Gunicorn 20.1.0 documentation*. (n.d.). Retrieved October 4, 2022, from https://docs.gunicorn.org/en/stable/install.html

20. *Volley overview*. (n.d.). Volley. Retrieved October 4, 2022, from https://google.github.io/volley/

21. *App Manifest Overview*. (n.d.). Android Developers. Retrieved October 4, 2022, from https://developer.android.com/guide/topics/manifest/manifest-intro

22. *Configure your build*. (n.d.). Android Developers. Retrieved October 4, 2022, from https://developer.android.com/studio/build