ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ

ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής (iOS) διαχείρισης μαθημάτων για τη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΝΣΤΑΝΤΙΝΟΣ Δ. ΜΕΛΕΤΙΟΥ
ΝΙΚΟΛΑΟΣ Θ. ΚΟΥΚΟΥΤΗΣ

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Συμμετοχή στην επίβλεψη : Δρ. Σ. Καπελλάκη, ΕΔΙΠ

Αθήνα, Νοέμβριος 2019

ΕΘΝΙΚΟ ΜΕΤΣΟΒΙΟ ΠΟΛΥΤΕΧΝΕΙΟ
ΣΧΟΛΗ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ
ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ
ΤΟΜΕΑΣ ΣΥΣΤΗΜΑΤΩΝ ΜΕΤΑΔΟΣΗΣ ΠΛΗΡΟΦΟΡΙΑΣ ΚΑΙ
ΤΕΧΝΟΛΟΓΙΑΣ ΥΛΙΚΩΝ

Σχεδίαση και ανάπτυξη εφαρμογής (iOS) διαχείρισης μαθημάτων για τη Σχολή Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

ΚΩΝΣΤΑΝΤΙΝΟΣ Δ. ΜΕΛΕΤΙΟΥ
ΝΙΚΟΛΑΟΣ Θ. ΚΟΥΚΟΥΤΗΣ

Επιβλέπων : Ιάκωβος Βενιέρης
Καθηγητής Ε.Μ.Π.

Συμμετοχή στην επίβλεψη : Δρ. Σ. Καπελλάκη, ΕΔΙΠ

Εγκρίθηκε από την τριμελή εξεταστική επιτροπή τη 19η Νοεμβρίου 2019.

.....	.....	.....
Ιάκωβος Βενιέρης Καθηγητής Ε.Μ.Π.	Δήμητρα Θεοδώρα Κακλαμάνη Καθηγήτρια Ε.Μ.Π.	Γεώργιος Ματσόπουλος Καθηγητής Ε.Μ.Π

Αθήνα, Νοέμβριος 2019

…….

Κωνσταντίνος Δ. Μελετίου

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

…….

Νικόλαος Θ. Κουκουτής

Διπλωματούχος Ηλεκτρολόγος Μηχανικός και Μηχανικός Υπολογιστών Ε.Μ.Π.

Copyright © **Κωνσταντίνος Δ. Μελετίου, Νικόλαος Θ. Κουκουτής** 2019

Με επιφύλαξη παντός δικαιώματος. All rights reserved.

Απαγορεύεται η αντιγραφή, αποθήκευση και διανομή της παρούσας εργασίας, εξ ολοκλήρου ή τμήματος αυτής, για εμπορικό σκοπό. Επιτρέπεται η ανατύπωση, αποθήκευση και διανομή για σκοπό μη κερδοσκοπικό, εκπαιδευτικής ή ερευνητικής φύσης, υπό την προϋπόθεση να αναφέρεται η πηγή προέλευσης και να διατηρείται το παρόν μήνυμα. Ερωτήματα που αφορούν τη χρήση της εργασίας για κερδοσκοπικό σκοπό πρέπει να απευθύνονται προς τους συγγραφείς.

Οι απόψεις και τα συμπεράσματα που περιέχονται σε αυτό το έγγραφο εκφράζουν τους συγγραφείς και δεν πρέπει να ερμηνευθεί ότι αντιπροσωπεύουν τις επίσημες θέσεις του Εθνικού Μετσόβιου Πολυτεχνείου.

Περίληψη

Η τεχνολογία αποτελεί σήμερα αναπόσπαστο κομμάτι της ζωής κάθε ανθρώπου, καθώς προσφέρει υπηρεσίες και εργαλεία, που μπορούν να διευκολύνουν σημαντικά την καθημερινότητα του, αλλά και πιο εξειδικευμένες εργασίες, όπου η σύγχρονη τεχνολογία καθίσταται απαραίτητη για την περάτωσή τους.

Σκοπός της διπλωματικής εργασίας είναι η σχεδίαση και η ανάπτυξη ενός συστήματος που θα βοηθά τους φοιτητές, τους καθηγητές, αλλά και τη γραμματεία της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών να διαχειρίζονται τα μαθήματα τους. Η εφαρμογή αναπτύχθηκε με σκοπό τη χρήση της σε έξυπνες κινητές συσκευές λειτουργικού συστήματος iOS (iPhone).

Παρέχει την δυνατότητα προσθήκης, οργάνωσης και επεξεργασίας των μαθημάτων, προσαρμοσμένη στις ανάγκες διαχείρισης της σχολής, αλλά και τη δυνατότητα παρακολούθησης μαθημάτων από τους φοιτητές, ενώ στους διδάσκοντες, προσφέρεται μέσω αυτής, η ευκαιρία να βελτιώσουν την εμπειρία των ίδιων, αλλά και των φοιτητών, μέσω ανάρτησης υλικού, ασκήσεων, βαθμολογίας, αλλά και ευκολότερης αλληλεπίδρασης με το μάθημα.

Πιο συγκεκριμένα, η γραμματεία εφόσον λάβει από τους δημιουργούς και διαχειριστές του συστήματος έναν λογαριασμό με τα αντίστοιχα διαπιστευτήρια, μπορεί να επεξεργαστεί ήδη υπάρχοντα μαθήματα, ή να προσθέσει καινούργια, και να αναθέσει σε καθηγητές την διδασκαλία τους. Τέλος, έχει πρόσβαση σε λεπτομέρειες σχετικές με τους διδάσκοντες, τα μαθήματα και τους φοιτητές για τους οποίους μπορεί να υποβάλει βαθμολογία.

Οι καθηγητές, πέρα από πρόσβαση στα στοιχεία του μαθήματος και στα στοιχεία των φοιτητών που το παρακολουθούν, μπορούν μέσω της εφαρμογής myUniLife να αναρτούν σχετικά έγγραφα για το μάθημα, προσβάσιμα από τους φοιτητές, καθώς και εκφωνήσεις ασκήσεων. Τέλος, έχουν πρόσβαση στις λύσεις των ασκήσεων που εχουν υποβάλει οι φοιτητές τις οποίες και μπορούν εν τέλει να βαθμολογήσουν.

Οι φοιτητές επιλέγουν τα μαθήματα που θα ακολουθήσουν, και με τον τρόπο αυτό, αποκτούν πρόσβαση σε επιπλέον πληροφορίες και υλικό του μαθήματος. Τους προσφέρεται ένα εύχρηστο και φιλικό περιβάλλον, στο οποίο έχουν συγκεντρωμένα όλα τα μαθήματα τους, ανά ροή και εξάμηνο διδασκαλίας, και την αντίστοιχη βαθμολογία τους σε αυτό, αλλά και τις ασκήσεις που έχουν υποβάλει.

Τέλος, τόσο στους καθηγητές όσο και στους φοιτητές η εφαρμογή παρέχει ένα εβδομαδιαίο πρόγραμμα, όπου φαίνονται με χρονολογική σειρά οι διαλέξεις στις οποίες πρέπει να παραστούν, καθώς και η ακριβής τοποθεσία διεξαγωγής τους.

Λέξεις Κλειδιά: iOS, iPhone, Swift, εφαρμογή, φοιτητής, καθηγητής, διαχείριση μαθημάτων, postgreSQL, Docker, NodeJS.

Abstract

Technology today is an integral part of every person's life, as it offers services and tools that can greatly improve their daily lives, and more sophisticated tasks, where modern technology is needed to complete them.

The purpose of the thesis is to design and develop a system, that will assist students, professors and the secretariat of the Department of Electrical and Computer Engineering to manage their courses. The application was developed to be used on smartphones that are running iOS (iPhone).

It provides the ability to add organization and edit courses, designed to the needs of the school, and the ability to students to be up to date with the courses, while giving professors the opportunity to enhance their own and students' experience, by uploading material, and more easily, interact with the lesson.

Specifically, the secretariat, after receiving from the creators and system administrators an account, with corresponding credentials, can edit existing courses or add new ones, and assign professors to them. Finally, she has access to the professors' details, classes and students for whom she can add grades.

In addition, to accessing course's data and students attending the course, teachers can post relevant documentation of the course, which is accessible to students. Finally, they have access to the solutions of the exercises, submitted by the students, in which they can add grades..

Students are able to choose the courses that they want to follow and thus, gain access to additional information and material. They have all their courses in one place, their respective grades and the exercises that they have submitted, through a user-friendly environment.

Finally, for both professors and students, the application provides a weekly schedule, showing the date of the lectures, that they need to attend, and their exact location.

Λέξεις Κλειδιά: iOS, iPhone, Swift, Mobile App, student, professor, course management, postgreSQL, Docker, NodeJS.

Ευχαριστίες

Αισθανόμαστε την ανάγκη να ευχαριστήσουμε τους ανθρώπους που μας στήριξαν και μας βοήθησαν, τόσο στην εκπλήρωση της παρούσας διπλωματικής, όσο και στην ολοκλήρωση των σπουδών μας στην ανώτατη εκπαίδευση.

Αρχικά, θα θέλαμε να εκφράσουμε τις θερμές ευχαριστίες μας στον κύριο Ιάκωβο Βενιέρη, καθηγητή της σχολής Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών και επιβλέποντα καθηγητή της διπλωματικής μας εργασίας, ο οποίος μας έδωσε την ευκαιρία και ώθηση να ασχοληθούμε με ένα θέμα που επιθυμούσαμε πολύ και οι δύο. Οι γνώσεις του, και ο ζήλος που επέδειξε, για να τις μοιραστεί μαζί μας, έπαιξαν καθοριστικό ρόλο στην εκπόνηση της παρούσας διπλωματικής και στην επίτευξη του επιθυμητού αποτελέσματος.

Στη συνέχεια, ευχαριστούμε τα μέλη της επιτροπής, καθηγήτρια Δήμητρα Θεοδώρα Κακλαμάνη και τον καθηγητή Γεώργιο Ματσόπουλο για την υποστήριξη τους καθ' όλη τη διάρκεια περαίωσης της διπλωματικής.

Έπειτα, θερμές ευχαριστίες θα θέλαμε να απευθύνουμε στον κύριο Πέτρο-Φλώριο Μπάκαλο, υποψήφιο διδάκτορα και μέλος της ερευνητικής ομάδας του Εργαστηρίου Ευφυών Επικοινωνιών και Δικτύων Ευρείας Ζώνης του ΕΜΠ, και την Δρ. Καπελλάκη Σοφία, χωρίς τη βοήθεια των οποίων θα ήταν αδύνατη η ολοκλήρωση της διπλωματικής εργασίας. Η συμβολή τους στην ανάπτυξη της εφαρμογής και στη συγγραφή της εργασίας υπήρξε καθοριστική.

Τέλος, ο καθένας μας ξεχωριστά θα θέλαμε να ευχαριστήσουμε τις οικογένειες μας και τους δικούς μας ανθρώπους που μας στήριξαν.

Εγώ, ο Μελετίου Κωνσταντίνος θα ήθελα να ευχαριστήσω του γονείς και τα αδέρφια μου, που ήταν δίπλα μου σε όλη τη διάρκεια των σπουδών μου.

Εγώ, ο Κουκουτής Νικόλαος θα ήθελα να ευχαριστήσω όλη την οικογένεια μου και ιδιαίτερα τη μητέρα μου Αικατερίνη, στην οποία και αφιερώνω την παρούσα διπλωματική εργασία για τη στήριξη που μου προσφέρει όλα αυτά τα χρόνια.

Πίνακας Περιεχομένων

Κατάλογος σχημάτων

1

Εισαγωγή

1.1 Σκοπός Διπλωματικής Εργασίας

Η παρούσα διπλωματική εργασία έχει ως σκοπό να βελτιώσει τη σχέση που έχουν οι φοιτητές με την σχολή, αλλά και να διευκολύνει τη διαχείριση των μαθημάτων, από τους καθηγητές και την γραμματεία, αναλύοντας τη διαδικασία ανάπτυξης μιας πλατφόρμας για κινητές συσκευές. Μέσω της πλατφόρμας αυτής, βελτιώνεται σημαντικά η ακαδημαϊκή επικοινωνία, και η ενημέρωση των φοιτητών, σχετικά με την πρόοδο τους, καθώς έχουν άμεση πρόσβαση σε πληροφορίες σχετικές με τα μαθήματα, από το κινητό τους τηλέφωνο. Επιπλέον, η γραμματεία και οι καθηγητές, με το πάτημα ενός κουμπιού, έχουν την δυνατότητα να διαχειριστούν το ίδρυμα, καθώς και να ολοκληρώσουν βασικές καθημερινές ανάγκες, όπως είναι η ανάρτηση εργασιών και η προσθήκη βαθμολογίας σε περιόδους εξεταστικής.

1.2 Δομή Διπλωματικής Εργασίας

Στο πλαίσιο ανάπτυξης της διπλωματικής θελήσαμε να ακολουθήσουμε τα προβλεπόμενα βήματα και διαδικασίες, τόσο για την σωστή διαχείριση μιας ερευνητικής εργασίας, όσο και για την ανάπτυξη λογισμικού. Τα βήματα που ακολουθήσαμε για την ολοκλήρωσή της αντικατοπτρίζονται από τα κεφάλαια της διπλωματικής, τα οποία διαρθρώνονται με βάση τα μικρότερα βήματα στα οποία διαμοιράστηκε το συνολικό έργο. Στο δεύτερο κεφάλαιο, παρουσιάζονται οι τεχνολογίες με τις οποίες εξοικιωθήκαμε, προκειμένου να πετύχουμε το στόχο μας και στο τρίτο, περιγράφονται οι σχεδιαστικές και λειτουργικές απαιτήσεις του συστήματος, και αναλύονται τα σενάρια χρήσης και λειτουργιών. Στη συνέχεια, στο κεφάλαιο τέσσερα περιγράφεται λεπτομερώς η σχεδίαση του front end, αλλά και του server της εφαρμογής, ενώ στο πέμπτο η υλοποίηση αυτών, καθώς και χαρακτηριστικά παραδείγματα χρήσης της εφαρμογής. Τέλος, στο κεφάλαιο έξι φαίνονται τα συμπεράσματα στα οποία καταλήξαμε μέσα από όλη την προαναφερθείσα διαδικασία, καθώς και πιθανές μελλοντικές επεκτάσεις του συστήματος myUniLife.

2

Τεχνολογίες

2.1 Η γλώσσα προγραμματισμού Swift

Το 2014, η Apple ανακοίνωσε τη γλώσσα προγραμματισμού Swift, προκειμένου να επιταχύνει την ανάπτυξη εφαρμογών για τα λειτουργικά συστήματα iOS και macOS, που η ίδια προσφέρει. Πρόκειται για μια αντικειμενοστραφή γλώσσα προγραμματισμού γενικού σκοπού, η οποία υποστηρίζει και το μοντέλο του συναρτησιακού προγραμματισμού. Τον επόμενο χρόνο η εταιρεία ανακοίνωσε μια νέα, ανοιχτού κώδικα, έκδοση της γλώσσας, και από τότε ο καθένας έχει τη δυνατότητα να συνεισφέρει στην ανάπτυξη της. [1]

2.2 Το λειτουργικό σύστημα iOS

Ένα από τα πιο κυρίαρχα λειτουργικά συστήματα στον κόσμο των κινητών συσκευών είναι το iOS, το οποίο παρουσιάστηκε το 2007, μαζί με το πρώτο iPhone, και έκτοτε η εξέλιξη του είναι ραγδαία. Αποτελεί το κύριο λειτουργικό σύστημα των συσκευών της Apple (iPhone, iPad, iPod). Βασίζεται στο λειτουργικό σύστημα UNIX, και είναι όσο των δυνατών βέλτιστα ανεπτυγμένο, ώστε να διαχειρίζεται το υλικό (hardware) των συσκευών με τον πλέον αποτελεσματικό τρόπο. Κάθε χρόνο παρουσιάζεται καινούργια έκδοση του λειτουργικού και αυτή τη στιγμή βρίσκεται στην έκδοση 13.

2.3 Η Αρχιτεκτονική του iOS

Το λειτουργικό σύστημα iOS είναι βασισμένο σε μια πολυεπίπεδη αρχιτεκτονική τεσσάρων στρωμάτων με βασικές τεχνολογίες σε κάθε στρώμα.

COCOA TOUCH

Στο στρώμα αυτό περιέχονται τα πιο βασικά πλαίσια (frameworks) που χρησιμοποιούνται σχεδόν από όλες τις εφαρμογές iOS. Το κυριότερο από τα πλαίσια αυτά είναι το UIKit.

Το πλαίσιο UIKit παρέχει την απαιτούμενη υποδομή για την ανάπτυξη εφαρμογών σε iOS. Αποτελείται από την αρχιτεκτονική παράθυρου και προβολής, με την οποία γίνεται η υλοποίηση της διεπαφής με τον χρήστη, και από την υποδομή διαχείρισης γεγονότων πολλαπλής αφής (Tap, Swipe, Pinch, double-tap). Επιπλέον, αποτελείται από τον κύριο βρόχο εκτέλεσης, που απαιτείται για τη διαχείριση των αλληλεπιδράσεων μεταξύ του χρήστη, του συστήματος και της εφαρμογής. Άλλες λειτουργίες που προσφέρονται από το πλαίσιο, σχετίζονται με την υποστήριξη εγγράφων, την υποστήριξη σχεδίασης και εκτύπωσης, πληροφορίες σχετικά με την τρέχουσα συσκευή, διαχείριση κειμένου και εμφάνιση, υποστήριξη αναζήτησης, υποστήριξη προσβασιμότητας, υποστήριξη επέκτασης εφαρμογών και διαχείριση πόρων. [2]

MEDIA LAYER

Ο ρόλος του στρώματος Media είναι να παρέχει στο iOS δυνατότητες δημιουργίας, επεξεργασίας και αναπαραγωγής ήχου, βίντεο, κινούμενων εικόνων, γραφικών, καθώς και εγγράφων κειμένου. Τα κυριότερα πλαίσια είναι :

- AV Foundation
- Core Animation
- Core Audio
- Core Image
- Core Text
- OpenAL
- OpenGL
- Quartz

CORE SERVICES

Το στρώμα βασικών υπηρεσιών παρέχει μια αφαίρεση των υπηρεσιών που παρέχονται στο επίπεδο του Core OS. Είναι υπεύθυνο για τη διαχείριση των θεμελιωδών υπηρεσιών του συστήματος που χρησιμοποιούν οι εγγενείς εφαρμογές iOS. Το επίπεδο βασικών υπηρεσιών παρέχει μια σειρά από απαραίτητες λειτουργίες, όπως υποστήριξη εφαρμογών για συστήματα με επεξεργαστές πολλαπλών πυρήνων και άλλα συμμετρικά συστήματα πολλαπλών επεξεργασιών (Grand Central

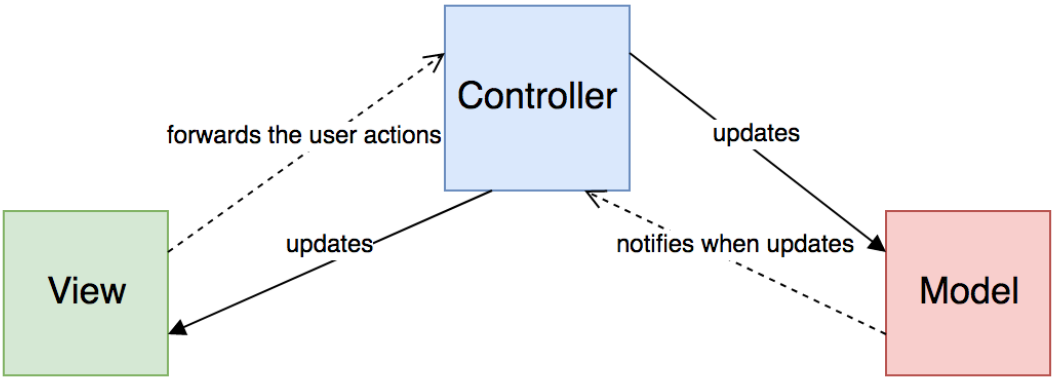
Dispatch), αντικείμενα block (επαναχρησιμοποιούμενο τμήμα κώδικα), αγορές εντός εφαρμογής (in-app purchases) και αποθήκευση νέφους (iCloud).

CORE OS

Το στρώμα Core OS είναι το κατώτερο στρώμα της ιεραρχίας και είναι υπεύθυνο για την ίδρυση του λειτουργικού συστήματος. Παρέχει τις τεχνολογίες και τα πλαίσια χαμηλού επιπέδου που σχετίζονται με το υλικό και τα δίκτυα. Οι υπηρεσίες αυτές βασίζονται σε εγκαταστάσεις στο επίπεδο πυρήνα και οδήγησης συσκευών. Αποτελεί το πιο σημαντικό στρώμα, καθώς είναι υπεύθυνο για τη διαχείριση της μνήμης και της ασφάλειας του συστήματος, παρέχοντας λειτουργίες κατανομής και απελευθέρωσης μνήμης, όπως και εκτέλεσης κρυπτογραφικών λειτουργιών (κρυπτογράφηση / αποκρυπτογράφηση δεδομένων). Τέλος, είναι αρμόδιο για την διατήρηση της ισχύος του συστήματος τερματίζοντας οποιεσδήποτε λειτουργίες υλικού δεν χρησιμοποιούνται τη δεδομένη χρονική στιγμή. [3]

2.4 Σχεδιαστικά Πρότυπα στο iOS

- **Πρότυπο Μοντέλο-Προβολή-Ελεγκτής (MVC)**



Εικόνα 1 Το πρότυπο σχεδίασης MVC. (Πηγή [4])

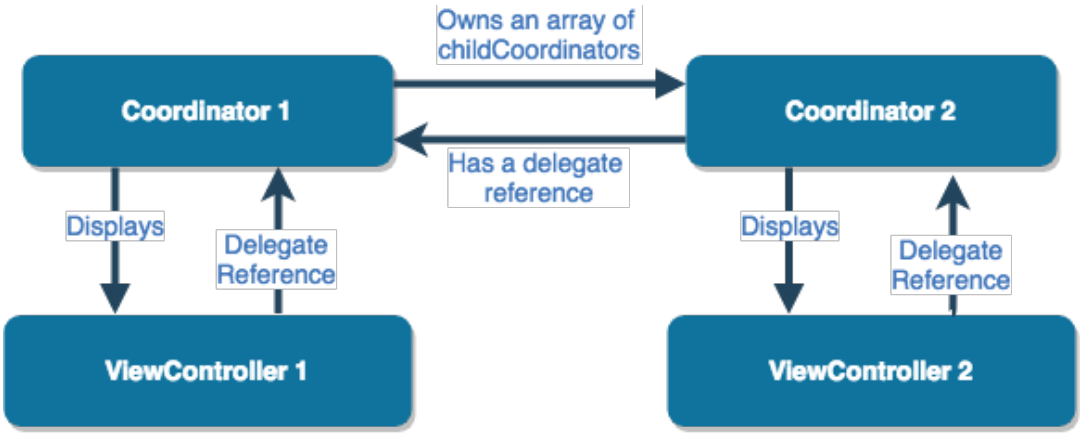
Το πρότυπο MVC αποτελεί το πιο συνηθισμένο πρότυπο σχεδίασης εφαρμογών στο λειτουργικό σύστημα iOS. Χρησιμοποιείται εκτεταμένα από την Apple, στους κώδικες που παρέχει ως δείγμα, και αποτελεί μια παραλλαγή του κλασικού προτύπου MVC που χρησιμοποιείται ευρέως στον προγραμματισμό.

Τα κύρια συστατικά του είναι η προβολή, ο ελεγκτής και το μοντέλο. Την προβολή μπορεί κανείς να την σκεφτεί σαν τη διεπαφή με το χρήστη την κάθε χρονική στιγμή. Μοντέλο είναι τα δεδομένα που προβάλλονται στην οθόνη του

κινητού, και ελεγκτής είναι μια γέφυρα μεταξύ της προβολής και του μοντέλου. Κατά την διάρκεια εκτέλεσης της εφαρμογής, ο ελεγκτής διαθέτει μια προβολή και συνδέει ένα μοντέλο δεδομένων με αυτήν. Στον χώρο του iOS, ο ελεγκτής και η προβολή συνδυάζονται σε ένα αρχείο προσφέροντας τον ελεγκτή προβολής.

Σε μια τυπική εφαρμογή που χρησιμοποιεί ως αρχιτεκτονική το MVC, ο ελεγκτής προβολής είναι αρμόδιος να χειρίζεται την πλοήγηση μεταξύ άλλων ελεγκτών προβολής. Αυτό σημαίνει ότι, ο ελεγκτής προβολής πρέπει να γνωρίζει εκ των προτέρων τους άλλους ελεγκτές με τους οποίους θα συνδεθεί. Έτσι, δημιουργείται μια στενή σύζευξη μεταξύ των ελεγκτών, με αποτέλεσμα να δυσκολεύεται η επαναχρησιμοποίηση τους και να καθιστά δύσκολη την ανάγνωση του κώδικα και τη δοκιμή της εφαρμογής. Επιπλέον, η προσέγγιση αυτή παραβιάζει την αρχή της ενιαίας ευθύνης, σύμφωνα με την οποία κάθε ενότητα, κλάση ή λειτουργία θα πρέπει να φέρει την ευθύνη για ένα μόνο μέρος της λειτουργικότητας, που παρέχεται από το λογισμικό, και ότι η ευθύνη θα πρέπει να είναι εντελώς εγκλωβισμένη από την κλάση ή τη λειτουργία. [4][19]

- Πρότυπο Συντονιστών (Coordinators pattern)



Εικόνα 2 *Το πρότυπο σχεδίασης συντονιστών. (**Πηγή [5]**)*

Μια λύση στο προηγούμενο πρόβλημα έρχεται να δώσει το πρότυπο των συντονιστών, το οποίο προτάθηκε από τον Soroush Khanlou το 2015 και από τότε έχει αποκτήσει μεγάλη απήχηση. Βασική ευθύνη ενός συντονιστή είναι να χειρίζεται τη δημιουργία της εφαρμογής και τη λογική ροή μετάβασης μεταξύ των οθονών. Αναλαμβάνει τη διαχείριση όλων των εξαρτήσεων που απαιτούνται για μια συγκεκριμένη οθόνη, δηλαδή συλλέγει τα απαραίτητα δεδομένα και τα τροφοδοτεί στον ελεγκτή προβολής. Ο ελεγκτής προβολής στη συνέχεια, όταν θέλει να εκτελέσει μια λειτουργία, όπως η παρουσίαση ενός άλλου ελεγκτή, ή να πραγματοποιήσει μια επικοινωνία μέσω δικτύου, απευθύνεται στον υπεύθυνο για αυτόν συντονιστή, ο οποίος με τη σειρά του οργανώνει και εκτελεί τις απαιτούμενες

ενέργειες. Με τον τρόπο αυτό, οι ελεγκτές προβολής αποκτούν απομόνωση, καθώς δεν γνωρίζουν τι θα ακολουθήσει μετέπειτα στην πορεία της εφαρμογής. [5]

2.5 Εργαλεία ανάπτυξης

Xcode

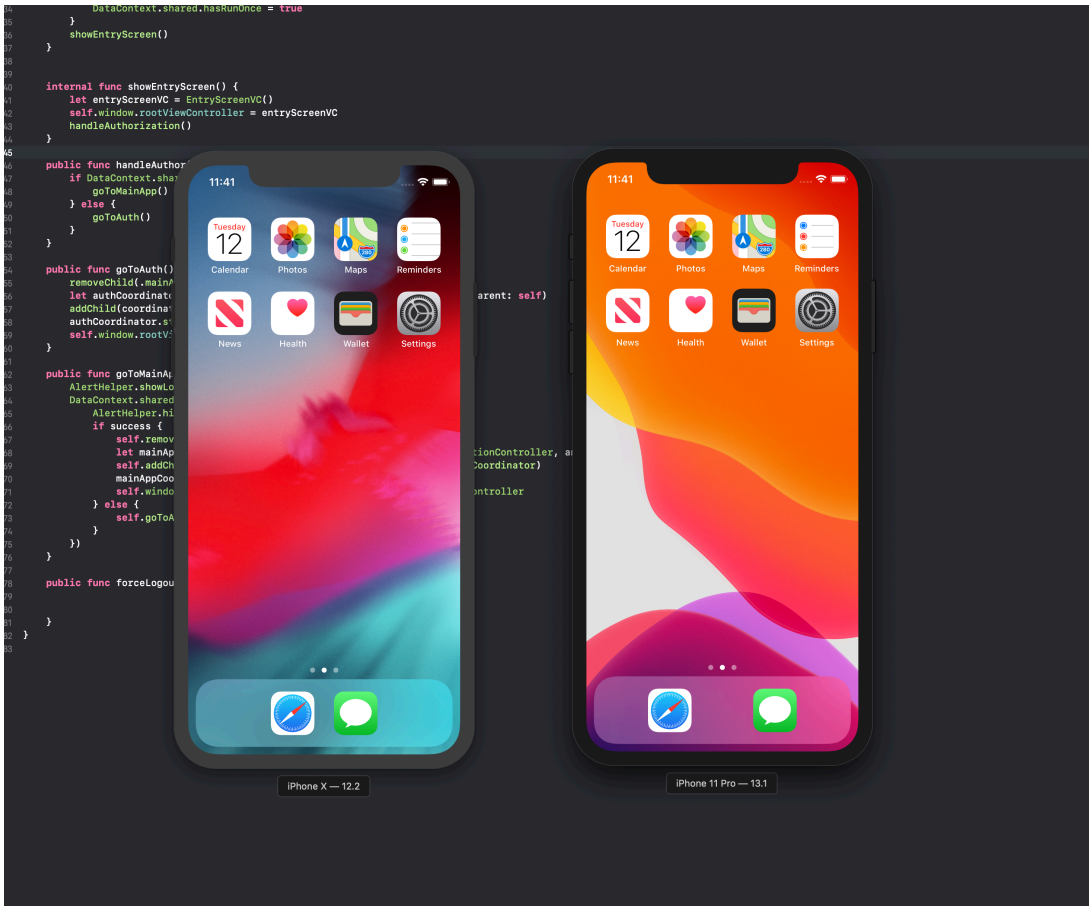
Η σύγχρονη ανάπτυξη λογισμικού απαιτεί έναν ιλιγγιώδη αριθμό διαφορετικών προγραμμάτων. Για να μπορέσει κανείς να δοκιμάσει μια εφαρμογή iOS, θα χρειαστεί επεξεργαστές, μεταγλωττιστές, συνδέσμους, ελεγκτές σύνταξης, κρυπτογραφικούς υπογράφοντες, μεταγλωττιστές πόρων, εντοπιστές σφαλμάτων, προσομοιωτές, αναλυτές επιδόσεων και πολλά άλλα. Το Xcode ενορχηστρώνει όλα αυτά τα μεμονωμένα εργαλεία σε μια διεπαφή και αποφασίζει εκείνο ποια εργαλεία πρέπει να εκτελεστούν και πότε.

Επιπρόσθετα, εκτός από το ότι περιλαμβάνει όλα τα εργαλεία που χρειάζονται, το Xcode μπορεί να φιλοξενήσει διάφορα κιτ ανάπτυξης λογισμικού (SDK). Ένα SDK είναι μια συλλογή αρχείων που παρέχει στο Xcode τις κατάλληλες διεπαφές προγραμματισμού εφαρμογών (API) που χρειάζεται, προκειμένου να δημιουργήσει μια εφαρμογή για ένα συγκεκριμένο λειτουργικό σύστημα, όπως το iOS. Οι διεπαφές αυτές επιταχύνουν την ανάπτυξη των εφαρμογών, καθώς πολλά από τα πράγματα που συνήθως απαιτούνται σε νέες εφαρμογές, έχουν ήδη υλοποιηθεί (π.χ. εμφάνιση ειδοποίησης, λήψη φωτογραφίας κ.α). [6]

Εικόνα 3 *Στιγμιότυπο εργαλείου Xcode. (**Πηγή [7]**)*

Simulators

Αποτελεί μια εφαρμογή, αποκλειστικά για το λειτουργικό σύστημα Mac OS, η οποία προσφέρεται εγκατεστημένη ως μέρος των εργαλείων του Xcode. Ο προσομοιωτής δίνει τη δυνατότητα στον προγραμματιστή να δοκιμάσει γρήγορα και εύκολα την κατάσταση της εφαρμογής, κατά τη διάρκεια της διαδικασίας ανάπτυξης, μέσα από ένα προσομοιωμένο περιβάλλον iPhone, iPad, Apple Watch ή Apple TV. Τρέχοντας την εφαρμογή στον προσομοιωτή, ο προγραμματιστής μπορεί να περιηγηθεί μέσα σε αυτή και να εντοπίσει τυχόν σφάλματα, που αφορούν τη διεπαφή με τον χρήστη, ή τις λειτουργίες της εφαρμογής.

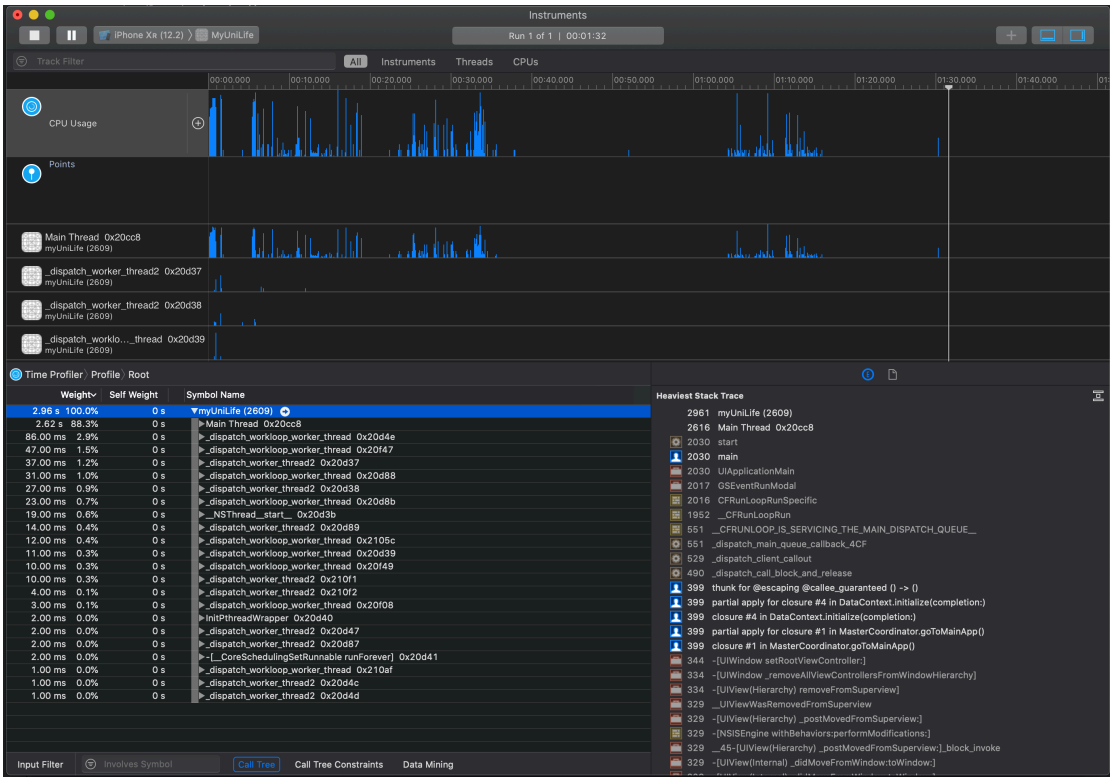


Εικόνα 4 *Στιγμιότυπο εργαλείου simulators.*

Instruments

Το εργαλείο Instruments είναι ένα ισχυρό και ευέλικτο εργαλείο για την ανάλυση των επιδόσεων του συστήματος. Είναι και αυτό μέρος του συνόλου των εργαλείων του Xcode. Η χρήση του συνεισφέρει αποτελεσματικά στην αύξηση της απόδοσης και στην καλύτερη αξιοποίηση των πόρων της κάθε εφαρμογής, καθώς

διευκολύνει την εύρεση διαρροών μνήμης και την παρακολούθηση των δικτυακών συνδέσεων.



Εικόνα 5 *Στιγμιότυπο εργαλείου instruments.*

Η πλατφόρμα postman

Πρόκειται για μια πλατφόρμα υποβοήθησης ανάπτυξη APIs. Επιτρέπει την εύκολη και γρήγορη δημιουργία και ανάπτυξη requests, καθώς και αυτοματοποιημένο testing, ώστε να εξασφαλιστεί ότι οι αλλαγές στον κώδικα δεν θα επηρεάσουν αρνητικά ήδη υπάρχοντα apis.

Προσομοιώνοντας τα endpoints και τα responses αυτών, επιτρέπει την ανάπτυξη λογισμικού χωρίς να απαιτείται να δημιουργηθεί και να εκτελείται ένας server. Επιπλέον, βοηθάει στην ανάπτυξη, στον έλεγχο και στην αποσφαλμάτωση ενός server, χωρίς την ανάγκη ενός front end client. Τέλος, διαθέτει επίσης εργαλεία για την παραγωγή machine readable documentation. [20]

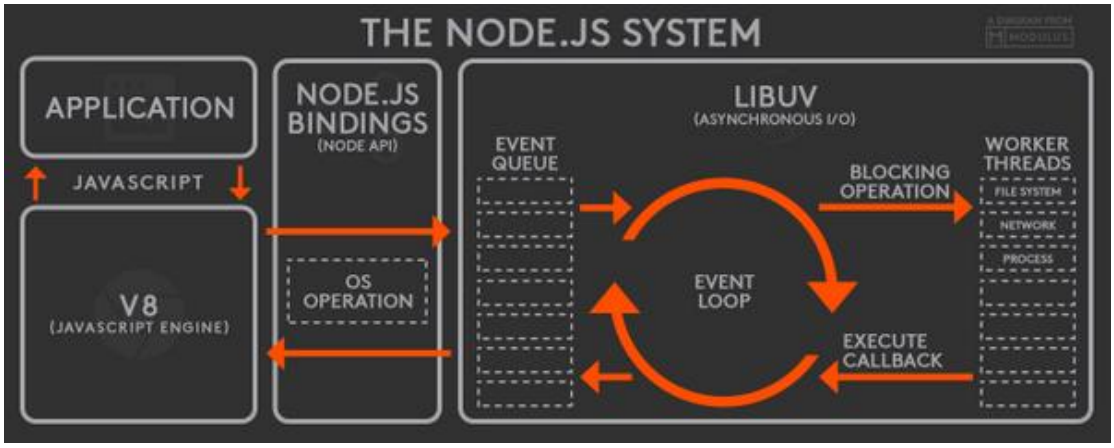
2.6 Το περιβάλλον ανάπτυξης Node.js

Για το σχεδιασμό του server side μέρους της εφαρμογής χρησιμοποιήθηκε η γλώσσα προγραμματισμού javascript στο περιβάλλον ανάπτυξης Node.js, σε συνδυασμό με express framework.

Το Node.js δημιουργήθηκε από τον Ryan Dahl το 2009. Η δημιουργία και η συντήρηση του έργου χορηγήθηκε από την εταιρία Joyent. Η ιδέα για την ανάπτυξη του *node* προήλθε από την ανάγκη του Ryan Dahl να βρει τον πιο αποδοτικό τρόπο να ενημερώνει τον χρήστη σε πραγματικό χρόνο για την κατάσταση ενός αρχείου που ανέβαζε στο διαδίκτυο. Επίσης, επηρεάστηκε από το Mongrel του Zed Shaw. Επιπροσθέτως, μετά από αποτυχημένα έργα σε C, Lua, Haskell η κυκλοφορία της μηχανής V8 (V8 JavaScript Engine) της Google τον ώθησε να ασχοληθεί με την Javascript.

Κύριο χαρακτηριστικό του node.js είναι το γεγονός ότι είναι event driven, δηλαδή όλες οι ασύγχρονες συναρτήσεις εκτελούνται όταν πραγματοποιηθεί ένα προκαθορισμένο συμβάν (μια συνθήκη). Με τον τρόπο αυτό οι συναρτήσεις αποφεύγουν τα deadlocks και χαρακτηρίζονται ως non-blocking, γεγονός που καθιστά το node.js την καλύτερη επιλογή για την ανάπτυξη scalable συστημάτων.

Πρόκειται για μια πλατφόρμα ανάπτυξης λογισμικού (κυρίως διακομιστών) χτισμένη σε περιβάλλον Javascript. Στόχος του Node είναι να παρέχει ένα εύκολο τρόπο δημιουργίας κλιμακωτών διαδικτυακών εφαρμογών. Σε αντίθεση με τα περισσότερα σύγχρονα περιβάλλοντα ανάπτυξης εφαρμογών δικτύων, μία διεργασία node δεν στηρίζεται στην πολυνηματικότητα, αλλά σε ένα μοντέλο ασύγχρονης επικοινωνίας εισόδου/εξόδου. [8]



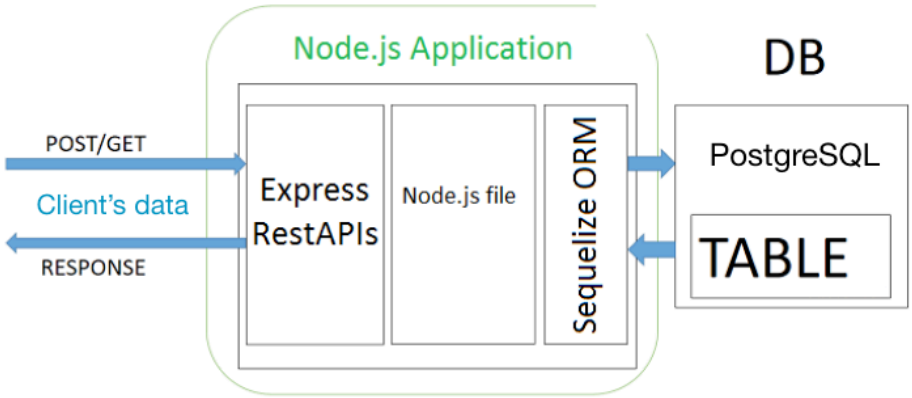
Εικόνα 6 *Το σύστημα node JS.* (***Πηγή: [9]***)

2.7 Διαχείριση των πακέτων του Node.js

Για τη διαχείριση των πακέτων του Node.js χρησιμοποιήθηκε το npm (node package manager).

Πρόκειται για έναν διαχειριστή πακέτων για τη γλώσσα javascript. Είναι ο προκαθορισμένος διαχειριστής πακέτων του nodeJS. Αποτελείται από ένα εργαλείο για τη γραμμή εντολών (το οποίο αποκαλείται επίσης npm) και μία βάση που περιέχει δωρεάν, αλλά και επί πληρωμής πακέτα. Είναι σχεδιασμένο εξ ολοκλήρου με τη χρήση της γλώσσας javascript, από τον Isaac Z. Schlueter. Δίνει τη δυνατότητα μέσω του package.json να εγκατασταθούν όλα τα απαραίτητα για την εφαρμογή πακέτα, με μια μόνο εντολή, ενώ υπάρχουν διαθέσιμα πανω απο 477.000 πακέτα. Για την καταγραφή και προσδιορισμό της ακριβής έκδοσης των πακέτων διατίθεται απο το npm το package-lock.json.

2.8 Sequelize



Εικόνα 7 *Μηχανισμός sequelize. (**Πηγή: [10]**)*

Για την αλληλεπίδραση με την βάση δεδομένων χρησιμοποιήθηκε το sequelize. Αποτελεί στη ουσία ενα orm, βασισμένο στο μοντέλο των ασύγχρονων promises που επιτρέπει την ασφαλή και ευκολότερη επικοινωνία με τη βάση δεδομένων. Προσφέρει υποστήριξη για τα μοντέλα που χρησιμοποιούνται στην NodeJs διευκολύνοντας με τον τον τρόπο αυτό τον ορισμό και χρήση των σχέσεων (associations) ανάμεσα στις δομές που αναπαριστούν οντότητες στη βάση. [11]

2.9 Η βάση δεδομένων PostgreSQL

Για την αποθήκευση των δεδομένων, που απαιτεί η ομαλή λειτουργία της εφαρμογής αξιοποιήθηκε η σχεσιακή βάση δεδομένων psql (postgreSql).

Είναι μια σχεσιακή δεδομένων ανοικτού κώδικα με πολλές δυνατότητες. Η ανάπτυξη της διαρκεί ήδη πάνω από δύο δεκαετίες και βασίζεται σε μια αποδεδειγμένα καλή αρχιτεκτονική, η οποία έχει δημιουργήσει μια ισχυρή αντίληψη των χρηστών της γύρω από την αξιοπιστία, την ακεραιότητα δεδομένων και την ορθή λειτουργία.

Η PostgreSQL τρέχει σε όλα τα βασικά λειτουργικά συστήματα, στα οποία περιλαμβάνονται το Linux, το UNIX (AIX, BSD, HP-UX, SGI, IRIX, MAC OS X, Solaris, Tru64) και τα Windows. Είναι συμβατή με ACID, και συμπεριλαμβάνει τους περισσότερους SQL92 και SQL99 τύπους δεδομένων συμπεριλαμβανομένων INTEGER, NUMERIC, BOOLEAN, CHAR, VARCHAR, DATE, INTERVAL και TIMESTAMP. Επίσης, υποστηρίζει αποθήκευση μεγάλων δυαδικών αντικειμένων (binary), όπως εικόνες, ήχοι ή βίντεο. Διαθέτει επίσης, περιβάλλοντα προγραμματισμού για τις γλώσσες προγραμματισμού C, C++, Java, Perl, Python, Ruby, Tcl, και υποστήριξη για την πλατφόρμα .NET και το πρότυπο ODBC, ενώ περιλαμβάνει και εξαιρετικό εγχειρίδιο χρήσης.

Η PostgreSQL προέρχεται από το πακέτο *POSTGRES*, το οποίο γράφτηκε στο Πανεπιστήμιο του Μπέρκλεϊ στην Καλιφόρνια των Η.Π.Α. και αποτελεί πλέον την πιο προχωρημένη βάση δεδομένων ανοιχτού κώδικα. [12]

2.10 Η σύγχρονη διαδικτυακή υπηρεσία Okeanos

Κατά τη διαδικασία ανάπτυξης του backend της εφαρμογής έγινε χρήση του okeanos. Πρόκειται για μια υπηρεσια τύπου IaaS (Infrastructure as a Service) μέσω της οποίας δίνεται η δυνατότητα στο χρήστη να δημιουργήσει εύκολα και άμεσα εικονικές μηχανές (υπολογιστές), που θα είναι ενεργοί και θα παραμένουν μόνιμα συνδεδεμένοι και διαθέσιμοι στο διαδικτυο, χωρίς να χρειάζεται μέριμνα για το πραγματικό hardware και τη συνδεσιμότητα αυτών. Μέσω ενός φυλλομετρητή του παρέχεται η δυνατότητα να διαχειριστεί, να διαγράψει, να ξανα δημιουργήσει ή να παραμετροποιήσει σύμφωνα με τις ανάγκες του το εικονικό του μηχάνημα, καθώς και να αποθηκεύσει και να μοιραστεί αρχεία. [13]

2.11 Το εργαλείο ελέγχου εκδόσεων Git

Το Git είναι ένα σύστημα ελέγχου εκδόσεων (καλείται και σύστημα ελέγχου αναθεωρήσεων ή σύστημα ελέγχου πηγαίου κώδικα) με έμφαση στην ταχύτητα, στην ακεραιότητα των δεδομένων και στην υποστήριξη για κατανεμημένες, μη γραμμικές, ροές εργασίας. Το Git σχεδιάστηκε και αναπτύχθηκε αρχικά από τον Λίνους Τόρβαλντς για τη ανάπτυξη του πυρήνα Linux το 2005 και έχει γίνει από τότε το πιο διαδεδομένο σύστημα ελέγχου εκδόσεων για ανάπτυξη λογισμικού.

Όπως τα περισσότερα άλλα κατανεμημένα συστήματα ελέγχου εκδόσεων/ αναθεωρήσεων, και αντίθετα με τα περισσότερα συστήματα πελάτη-διακομιστή, κάθε κατάλογος εργασίας του Git είναι ένα ολοκληρωμένο αποθετήριο λογισμικού με πλήρες ιστορικό και δυνατότητες πλήρους παρακολούθησης των εκδόσεων, ανεξάρτητο από την πρόσβαση δικτύου, χωρίς να απαιτείται η χρήση κεντρικού διακομιστή. Όπως ο πυρήνας Linux, το Git είναι Ελεύθερο λογισμικό, που διανέμεται κάτω από τους όρους της έκδοσης 2 της Γενικής Άδειας Δημόσιας Χρήσης GNU. [14]

2.12 UUIDs

Ένα καθολικά μοναδικό αναγνωριστικό (UUID) είναι ένας αριθμός 128-bit που χρησιμοποιείται για τον εντοπισμό πληροφοριών σε συστήματα υπολογιστών.

Όταν παράγονται, σύμφωνα με τις τυπικές μεθόδους, τα UUIDs είναι πρακτικά μοναδικά. Η μοναδικότητά τους δεν εξαρτάται από την κεντρική αρχή καταχώρισης ή τον συντονισμό μεταξύ των μερών που τα παράγουν, σε αντίθεση με τα περισσότερα άλλα συστήματα αριθμοδότησης. Θεωρητικά η πιθανότητα να αναπαραχθεί ένα UUID δεν είναι μηδέν, αλλά είναι αρκετά κοντά στο μηδέν, ώστε να θεωρείται αμελητέα.

Έτσι, ο καθένας μπορεί να δημιουργήσει ένα UUID και να το χρησιμοποιήσει για να εντοπίσει κάτι, σχεδόν βέβαιος ότι το αναγνωριστικό δεν αντικατοπτρίζει κάτι που έχει ήδη δημιουργηθεί ή πρόκειται να δημιουργηθεί για να αναπαριστά κάτι άλλο. Επομένως, οι πληροφορίες που έχουν επισημανθεί με UUID από ανεξάρτητα μέρη μπορούν αργότερα, να συνδυαστούν σε μία ενιαία βάση δεδομένων ή να μεταδοθούν στο ίδιο κανάλι, με αμελητέα πιθανότητα επανάληψης.

Η υιοθέτηση των UUIDs είναι ευρέως διαδεδομένη, με πολλές πλατφόρμες υπολογιστών να παρέχουν υποστήριξη για τη δημιουργία και την ανάλυση τους. [15]

2.13 Η υπηρεσία αποθήκευσης s3 της Amazon

Η υπηρεσία s3 της Amazon (Amazon S3) είναι μια υπηρεσία αποθήκευσης αντικειμένων (objects) που προσφέρει κορυφαία κλιμάκωση, διαθεσιμότητα δεδομένων, ασφάλεια και απόδοση. Αυτό σημαίνει ότι οι πελάτες όλων των μεγεθών και βιομηχανιών μπορούν να τη χρησιμοποιήσουν για την αποθήκευση και προστασία οποιουδήποτε όγκου δεδομένων, για μια σειρά περιπτώσεων χρήσης, όπως ιστότοποι, εφαρμογές για κινητά, backup και αποκατάσταση, αρχεία, επιχειρησιακές εφαρμογές, συσκευές IoT και μεγάλες αναλύσεις δεδομένων. Η υπηρεσία Amazon S3 παρέχει εύχρηστες λειτουργίες διαχείρισης, ώστε να μπορεί ο χρήστης να οργανώνει τα δεδομένα και να διαμορφώνει λεπτές ρυθμίσεις ελέγχου πρόσβασης για να ικανοποιεί τις συγκεκριμένες επιχειρηματικές και οργανωτικές απαιτήσεις. Η Amazon S3 έχει σχεδιαστεί για αντοχή 99,9% και αποθηκεύει δεδομένα για εκατομμύρια εφαρμογές σε επιχειρήσεις σε όλο τον κόσμο.

΄Εχει πλεονεκτήματα, όπως είναι η ευκολία κλιμάκωσης της υπηρεσίας, η ανθεκτικότητα και αξιοπιστία του συστήματος (καθώς αναλαμβάνει αυτόματα την δημιουργία αντιγράφων ασφαλείας) και παρέχει αρκετά πλάνα χρέωσης ανάλογα με τις ανάγκες του συστήματος, χωρίς όμως να θυσιάζεται η απόδοση. Ακόμη, παρέχει προηγμένη ασφάλεια μέσω κρυπτογραφίας και εργαλείων διαχείρισης, πληθώρα επιλογών για public access και access control, καθώς και σύστημα καταγραφής και παρακολούθησης των συμβάντων το οποίο μπορεί να φανεί ιδιαίτερα χρήσιμο στον διαχειριστή του συστήματος.

Για τους developers που επιθυμούν να αξιοποιήσουν την υπηρεσία, παρέχεται από την amazon (για πολλές γλώσσες προγραμματισμού συμπεριλαμβανομένης και της NodeJs) ενα sdk που διευκολύνει την χρήση της υπηρεσίας στο πλαίσιο ανάπτυξης ενός συστήματος. [16]

2.14 Η πλατφόρμα Docker

Το Docker είναι μια πλατφόρμα λογισμικού ανοιχτού κώδικα που υλοποιεί εικονικοποίηση (Virtualization) σε επίπεδο λειτουργικού συστήματος. Ουσιαστικά το Docker προσφέρει αυτοματοποιημένες διαδικασίες για την ανάπτυξη εφαρμογών σε απομονωμένες περιοχές χρήστη (User Spaces) που ονομάζονται Software Containers. Το λογισμικό χρησιμοποιεί τεχνολογίες του πυρήνα του Linux, όπως τα cgroups, και οι χώροι ονομάτων πυρήνα (kernel namespaces), για να επιτρέπει σε ανεξάρτητα software containers να εκτελούνται στο ίδιο λειτουργικό σύστημα. Έτσι, αποφεύγεται η χρήση επιπλέον υπολογιστικών πόρων που θα απαιτούσε μια εικονική μηχανή (virtual machine).

Η Docker, inc. ιδρύθηκε από τους Solomon Hykes και Sebastien Pahl κατά τη διάρκεια του Y Combinator Summer 2010 και ξεκίνησε το 2011. Ο Hykes ξεκίνησε το έργο Docker στη Γαλλία, ως εσωτερικό έργο στο πλαίσιο της dotCloud, μιας εταιρίας τύπου "πλατφόρμα ως υπηρεσία" (platform-as-a-service). Το Docker έκανε το ντεμπούτο του στο κοινό στη Santa Clara, στο PyCon το 2013. Δημοσιεύτηκε ως λογισμικό ανοικτού κώδικα τον Μάρτιο του 2013. Την εποχή εκείνη, χρησιμοποίησε το LXC ως το προεπιλεγμένο περιβάλλον εκτέλεσης. Ένα χρόνο αργότερα, με την κυκλοφορία της έκδοσης 0.9, το Docker αντικατέστησε το LXC με το δικό του component, το οποίο γράφτηκε στη γλώσσα προγραμματισμού Go.

Το Docker μπορεί να συμπεριλάβει μια εφαρμογή και τις εξαρτήσεις της σε ένα virtual container, που μπορεί να τρέξει σε οποιοδήποτε σύστημα Linux. Αυτό βοηθά στην παροχή ευελιξίας και φορητότητας, επιτρέποντας την εκτέλεση της εφαρμογής σε διάφορες τοποθεσίες, είτε on-premises, είτε σε public cloud ή σε private cloud. Το Docker χρησιμοποιεί τις λειτουργίες απομόνωσης πόρων του πυρήνα του Linux (όπως cgroups και namespaces του πυρήνα) και ένα union-capable σύστημα αρχείων (όπως το OverlayFS), για να επιτρέπει στα containers να εκτελούνται μέσα σε ένα ενιαίο στιγμιότυπο Linux, αποφεύγοντας το επιπλέον υπολογιστικό κόστος εκκίνησης και διατήρησης εικονικών μηχανών. Επειδή, τα πακέτα Docker (containers) είναι ελαφριά απο υπολογιστικής απόψεως, ένας και μόνο διακομιστής ή εικονική μηχανή μπορεί να τρέξει ταυτόχρονα πολλά containers. Μια ανάλυση το 2018 διαπίστωσε ότι μια τυπική περίπτωση χρήσης Docker περιλαμβάνει την εκτέλεση οκτώ containers ανά host, και ότι το ένα τέταρτο των οργανισμών που αναλύθηκαν τρέχουν 18 ή περισσότερους ανά host. Η υποστήριξη του πυρήνα του Linux για χώρους ονομάτων, ως επί το πλείστον, απομονώνει την πρόσβαση και αλληλεπίδραση μιας εφαρμογής με το λειτουργικό σύστημα, συμπεριλαμβανομένων των δέντρων διεργασιών, του δικτύου, των αναγνωριστικών χρήστη, και των συστημάτων αρχείων, ενώ τα cgroups του πυρήνα παρέχουν περιορισμό στη χρήση πόρων για τη μνήμη και τη CPU. Επιπρόσθετα, εκτός από τη χρήση abstracted διεπαφών εικονικοποίησης μέσω libvirt, LXC και systemd-nspawn, από την έκδοση 0.9, το Docker περιλαμβάνει το δικό του component (που ονομάζεται "libcontainer") για να χρησιμοποιεί απευθείας τις δυνατότητες εικονικοποίησης, που παρέχει ο πυρήνας του Linux. Τέλος, το Docker υλοποιεί ένα API υψηλού επιπέδου για την παροχή containers που εκτελούν διαδικασίες μεμονωμένα. [17]

2.14.1 Το εργαλείο docker-compose

Πρόκειται για ένα εργαλείο που αποσκοπεί στην αποδοτικότερη διαχείριση πολλαπλών containers σε μια εφαρμογή. Γίνεται χρήση ενός αρχείου YAML, όπου ορίζονται και παραμετροποιούνται όλες οι υπηρεσίες που θα χρησιμοποιηθούν. Έτσι, με τη χρήση μιας μόνο εντολής εκκινούν (και έπειτα ελέγχονται) όλες οι υπηρεσίες που χρειάζεται το εκάστοτε σύστημα.

Το cli του docker-compose επιτρέπει στους χρήστες να εκτελούν εντολές σε πολλά containers ταυτόχρονα. Για παράδειγμα, να χτίζουν docker images, να κάνουν το επιθυμητό scaling στα containers, να τρέχουν τα containers που έχουν σταματήσει καθώς και πολλά άλλα. Οι εντολές που σχετίζονται με τη χειραγώγηση των images ή τις διαδραστικές επιλογές χρήστη δεν σχετίζονται με το Docker Compose, διότι αφορούν ένα συγκεκριμένο container. [18]

3

Ανάλυση

3.1 Απαιτήσεις εφαρμογής διαχείρισης μαθημάτων

Στη σημερινή εποχή ο σύγχρονος φοιτητής καλείται να αντιμετωπίσει καθημερινά, μια πληθώρα υποχρεώσεων τόσο ακαδημαϊκής όσο και κοινωνικής φύσεως. Η σύγχρονη τεχνολογία με τις διευκολύνσεις που παρέχει μπορεί να απολέσει σύμμαχο του στην προσπάθεια αυτή. Ταυτόχρονα, είναι γεγονός ότι οι έξυπνες κινητές συσκευές (smartphones) έχουν γίνει αναπόσπαστο κομμάτι της ζωή του, καθώς παρέχουν χιλιάδες εφαρμογές για μετακινήσεις, ψυχαγωγία, επικοινωνία, αθλητισμό αλλά και μόρφωση. Συνεπώς, ιδιαίτερα χρήσιμη θα φαινόταν μια εφαρμογή για κινητές συσκευές που θα τον διευκόλυνε στην περάτωση των σπουδών του, παρέχοντας του έναν εύκολο και φιλικό τρόπο διαχείρισης των μαθημάτων.

3.2 Η εφαρμογή myUniLife

Η myUniLfe είναι μια εφαρμογή που βοηθάει τους φοιτητές τους καθηγητές, καθώς και τη γραμματεία μιας σχολής να διαχειρίζονται τα μαθήματά τους, Οι χρήστες της εφαρμογής χωρίζονται σε τρεις κατηγορίες, τους φοιτητές και τους καθηγητές και την γραμματεία. Ο φοιτητής έχει τη δυνατότητα να εγγραφεί στα μαθήματα του κάθε εξαμήνου, ώστε να έχει πρόσβαση σε περιεχόμενο σχετικό με τα μαθήματα αυτά και σε ανακοινώσεις. Ο καθηγητής θα μπορεί να βλέπει ποιοι φοιτητές παρακολουθούν το μάθημα, να προσθέτει σχετικό περιεχόμενο και να καταθέτει βαθμολογία. Στους διαχειριστές του κάθε τμήματος (γραμματεία) παρέχεται η δυνατότητα εποπτείας και οργάνωσης των μαθημάτων, του προγράμματος, αλλά και του ακαδημαϊκού προσωπικού.

3.2.1 Λειτουργίες της πλατφόρμας

Ο φοιτητής εφόσον εγγραφεί στην εφαρμογή χρησιμοποιώντας τα ακαδημαϊκά του διαπιστευτήρια (όνομα χρήστη και κωδικό), μπορεί να επιλέξει οποιοδήποτε μάθημα του τμήματος επιθυμεί και να εγγραφεί σε αυτό. Έτσι, του παρέχεται η δυνατότητα για:

- Προβολή υλικού σχετικό με το μάθημα
- Υποβολή εργασιών

- Ενημέρωση για την πρόοδο των εργασιών
- Έλεγχο της βαθμολογίας του μαθήματος
- Προβολή ανακοινώσεων, μέσω ανακατεύθυνσης στην ιστοσελίδα του τμήματος
- Επικοινωνία μέσω ηλεκτρονικού ταχυδρομείου με τον καθηγητή
- Προβολή εβδομαδιαίου προγράμματος μαθημάτων, με ακριβή ώρα και τοποθεσία
- Προβολή των στοιχείων του καθηγητή
- Προβολή όλων των μαθημάτων που διδάσκονται στη σχολή, ταξινομημένα τόσο ως προς εξάμηνο όσο και ως προς τις ροές

Ο καθηγητής εφόσον συνδεθεί στην εφαρμογή, με τα κατάλληλα διαπιστευτήρια που του έχουν αποσταλεί μέσω ηλεκτρονικού ταχυδρομείου, έχει τη δυνατότητα για:

- Προβολή εβδομαδιαίου προγράμματος
- Προβολή των μαθημάτων τα οποία του έχουν ανατεθεί για διδασκαλία
- Προβολή και επεξεργασία των γενικών στοιχείων του μαθήματος
- Ανάρτηση εγγράφων σχετικών με το εκάστοτε μάθημα (π.χ. σημειώσεις, υλικό μελέτης, χρήσιμα άρθρα)
- Ανάρτηση εργασιών για τους φοιτητές του μαθήματος
- Πρόσβαση στις λύσεις των εργασιών, που έχουν υποβληθεί από τους φοιτητές
- Βαθμολόγηση των εργασιών
- Προβολή όλων των εγγεγραμμένων φοιτητών του κάθε μαθήματος
- Καταχώρηση τελικής βαθμολογίας σε κάθε φοιτητή, αλλά και ενημέρωση αυτής
- Προβολή ανακοινώσεων του τμήματος

Η γραμματεία έπειτα από επικοινωνία με τους δημιουργούς της εφαρμογής, δέχεται έναν μοναδικό λογαριασμό με τον οποίο μπορεί να συνδεθεί ως διαχειριστής του τμήματος. Στη συνέχεια της παρέχονται οι ακόλουθες λειτουργίες:

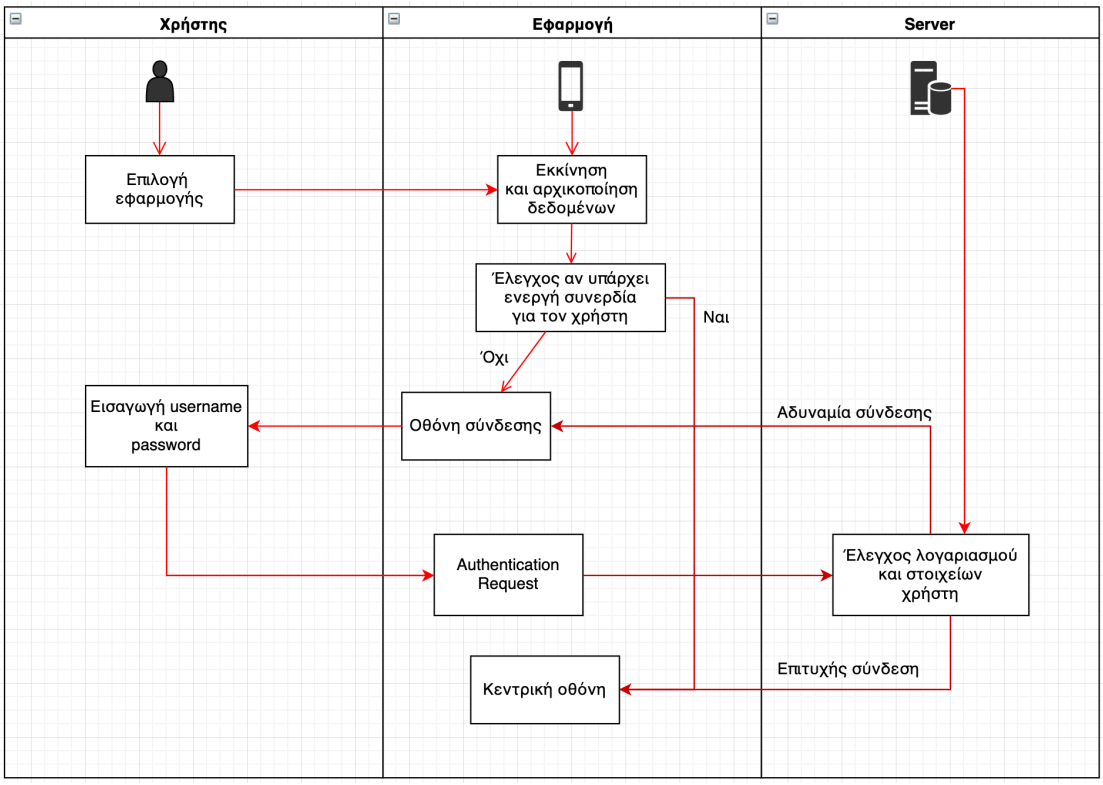
- Δημιουργία νέου μαθήματος
- Ανανέωση των στοιχείων ενός μαθήματος

- Ανάθεση της διδασκαλίας σε κάποιον καθηγητή
- Προβολή και επεξεργασία του προγράμματος των μαθημάτων
- Έλεγχος όλων των εγγεγραμμένων φοιτητών του κάθε μαθήματος
- Καταχώρηση τελικής βαθμολογίας σε κάθε φοιτητή

3.3 Σενάρια χρήσης

3.3.1 Είσοδος χρήστη

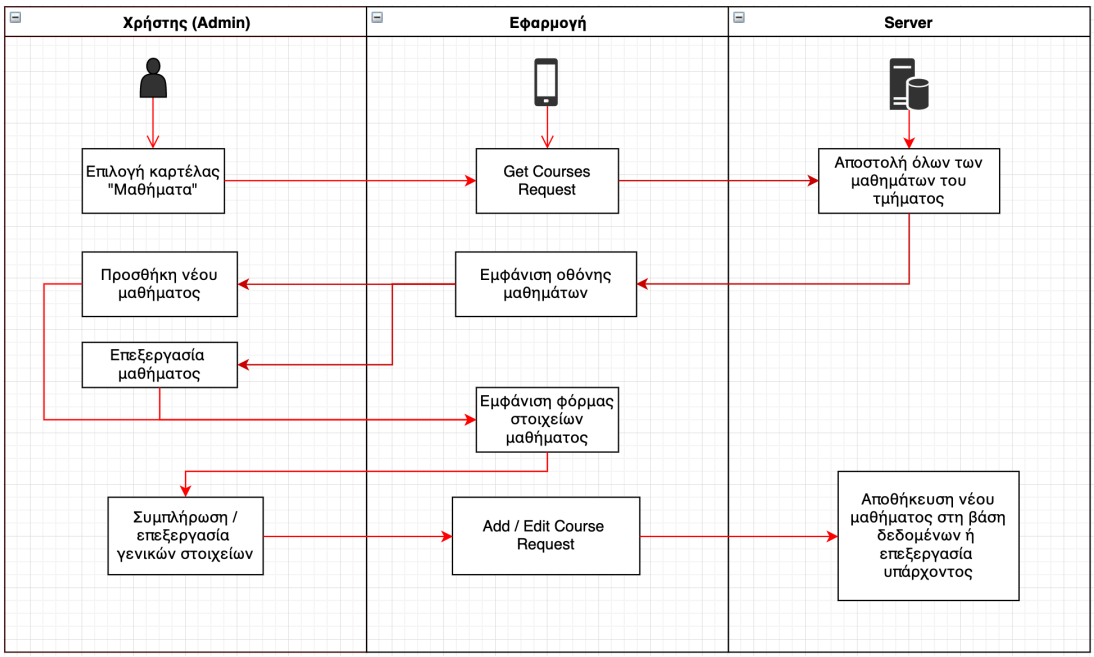
Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθείται για την είσοδο ενός χρήστη στο σύστημα.



Εικόνα 8 Σενάριο εισόδου χρήστη.

3.3.2 Δημιουργία μαθήματος

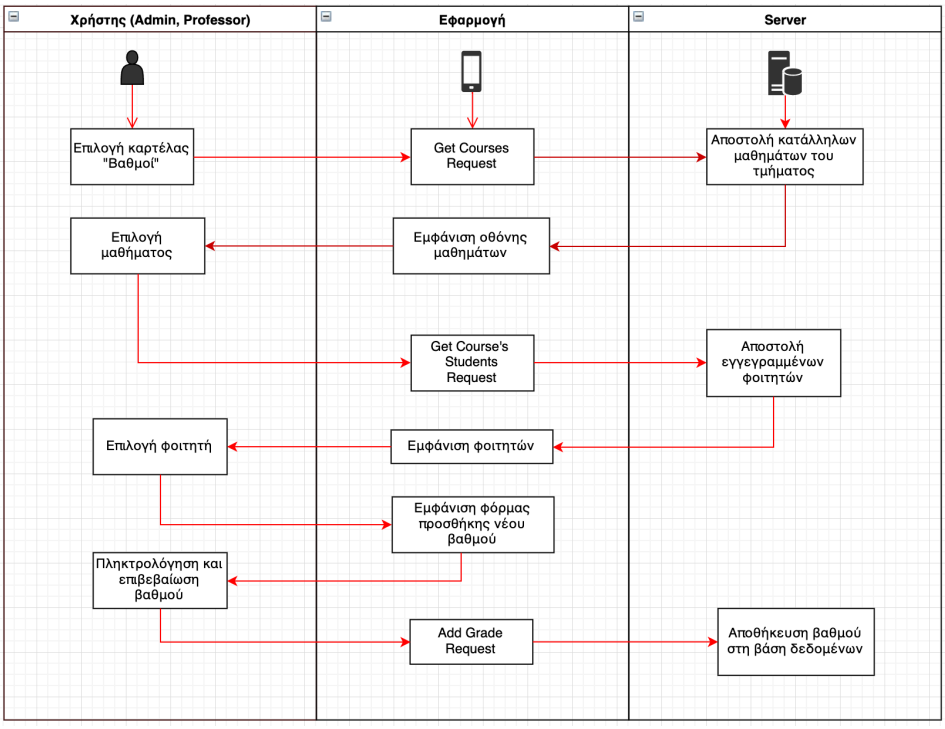
Το παρακάτω σενάριο προσομοιώνει τη διαδικασία δημιουργίας νέου μαθήματος ή επεξεργασίας ήδη υπάρχοντος, από τον καθηγητή ή την γραμματεία του τμήματος.



Εικόνα 9 *Σενάριο Δημιουργίας μαθήματος.*

3.3.3 Προσθήκη βαθμού σε μάθημα

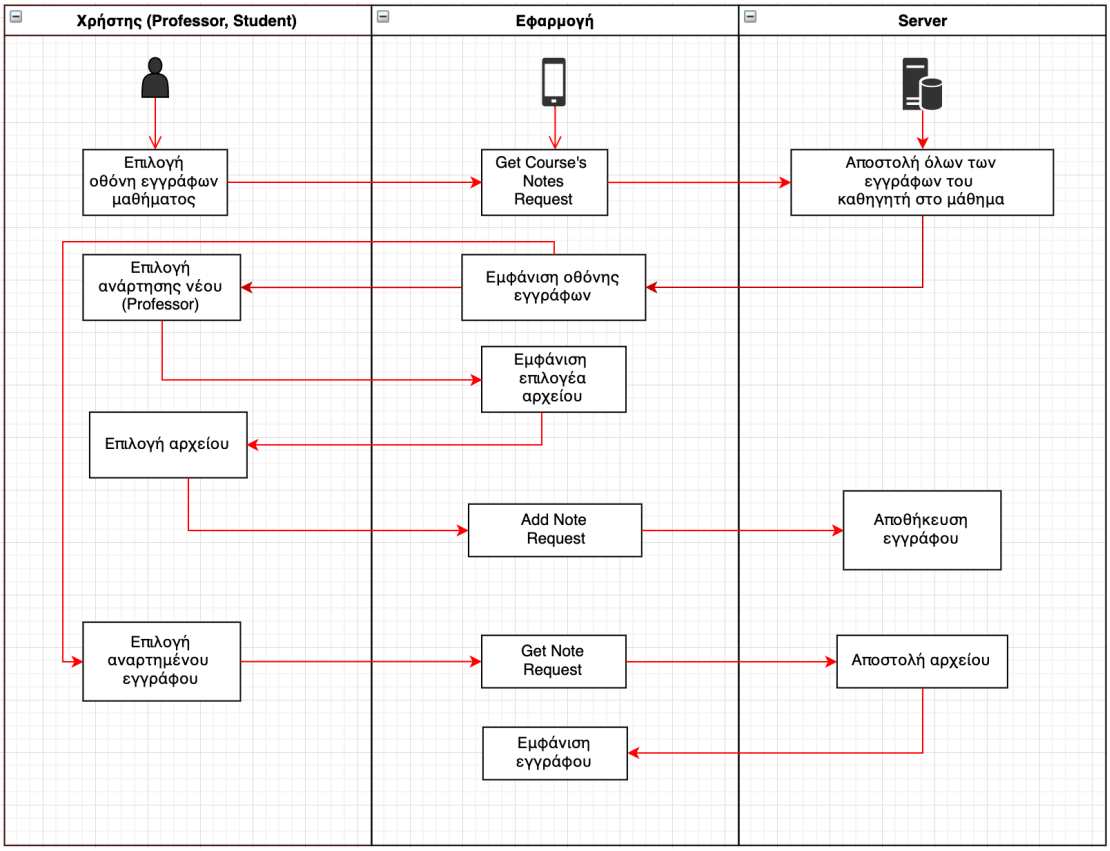
Το σενάριο αυτό δείχνει τη διαδικασία που χρειάζεται να ακολουθήσει ο καθηγητής ή η γραμματεία του τμήματος προκειμένου να προσθέσει τη βαθμολογία ενός φοιτητή σε ένα μάθημα.



Εικόνα 10 *Σενάριο προσθήκης βαθμού.*

3.3.4 Διαχείριση εγγράφων

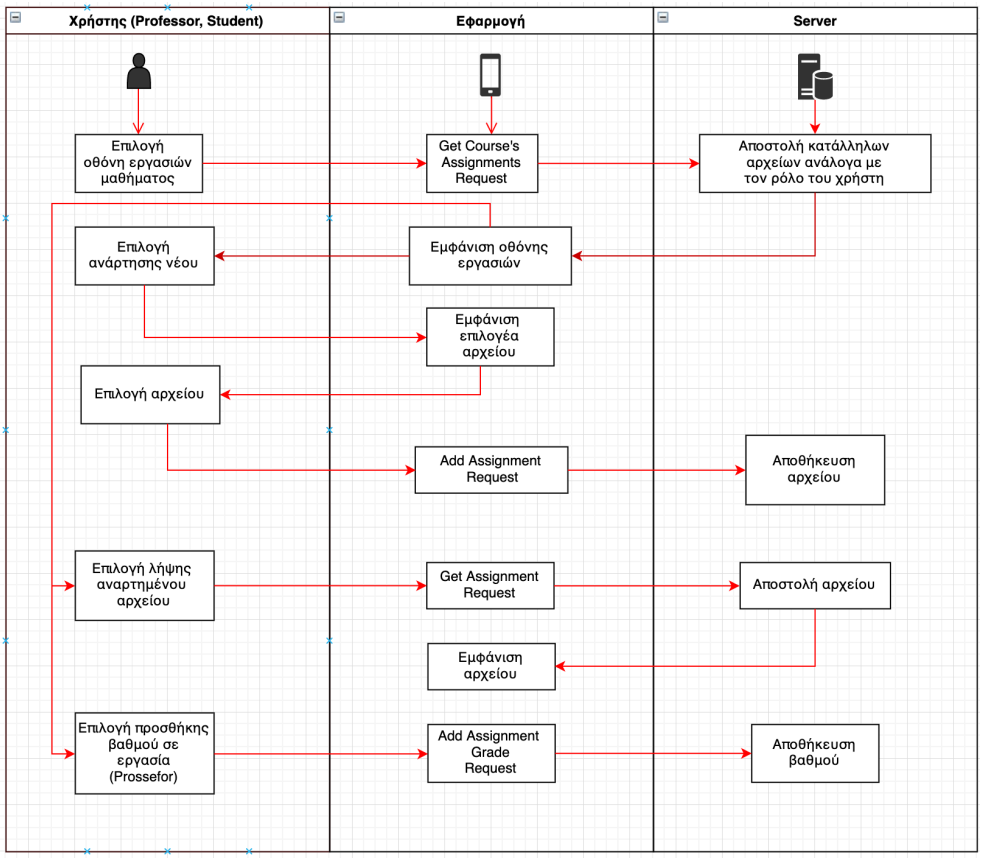
Το σενάριο αυτό αναπαριστά τη διαδικασία διαχείρισης των εγγράφων ενός μαθήματος από τον καθηγητή.



Εικόνα 11 *Σενάριο διαχείρισης εγγράφων.*

3.3.5 Διαχείριση εργασιών

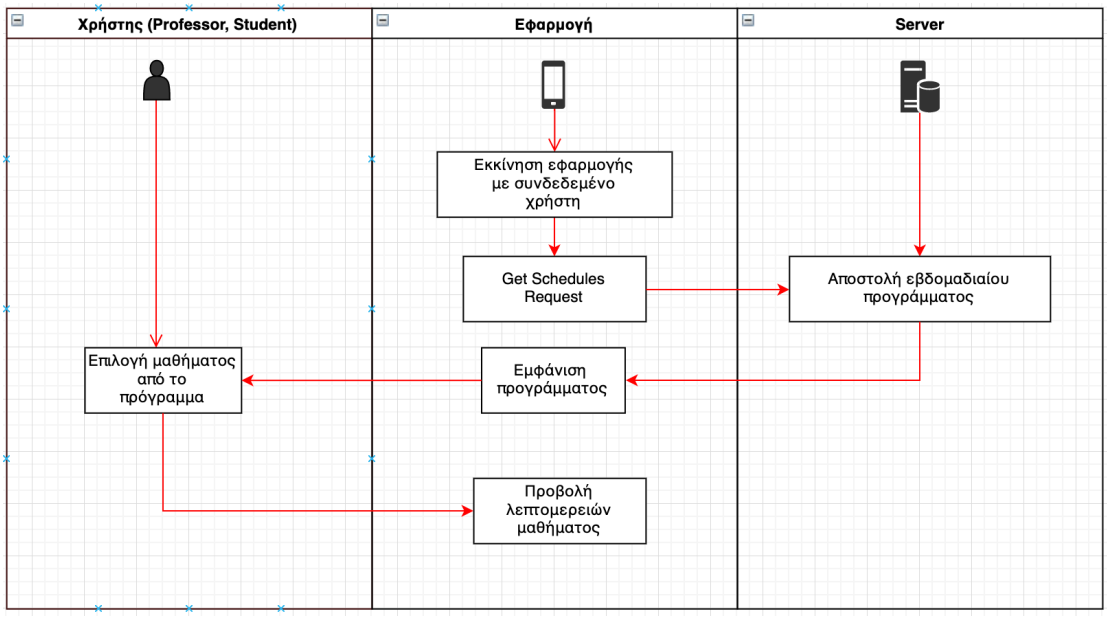
Το σενάριο αυτό αναπαριστά τη διαδικασία ανάρτησης νέων εργασιών, από τον καθηγητή του μαθήματος, την ανάρτηση λύσεων από τους φοιτητές, καθώς και τη λήψη αυτών.



Εικόνα 12 *Σενάριο διαχείρισης εργασιών.*

3.3.6 Προβολή εβδομαδιαίου προγράμματος

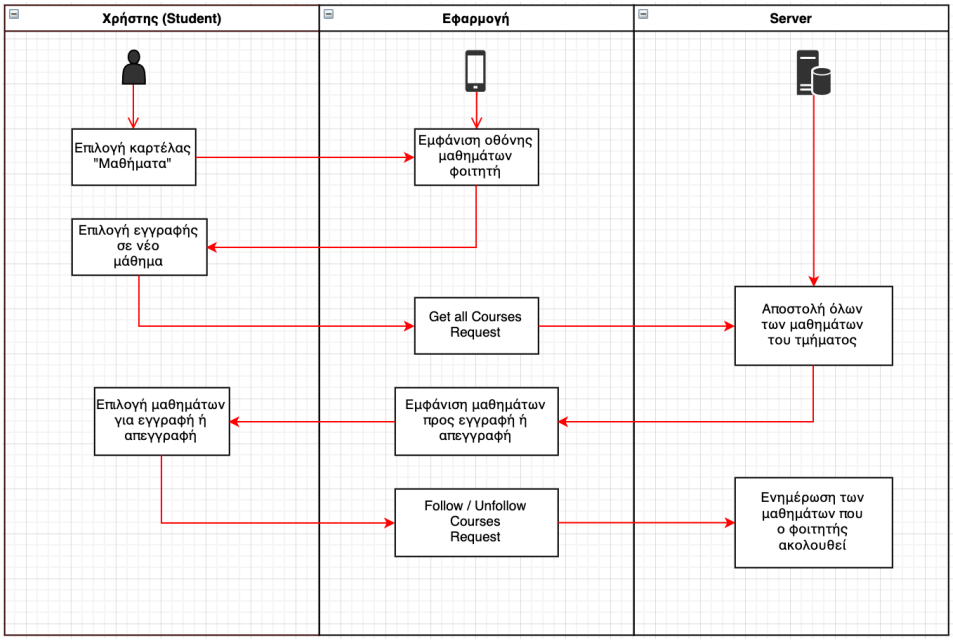
Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθεί ο χρήστης για να προβάλλει το εβδομαδιαίο πρόγραμμα των μαθημάτων του.



Εικόνα 13 *Σενάριο προβολής προγράμματος.*

3.3.7 Εγγραφή σε νέο μάθημα

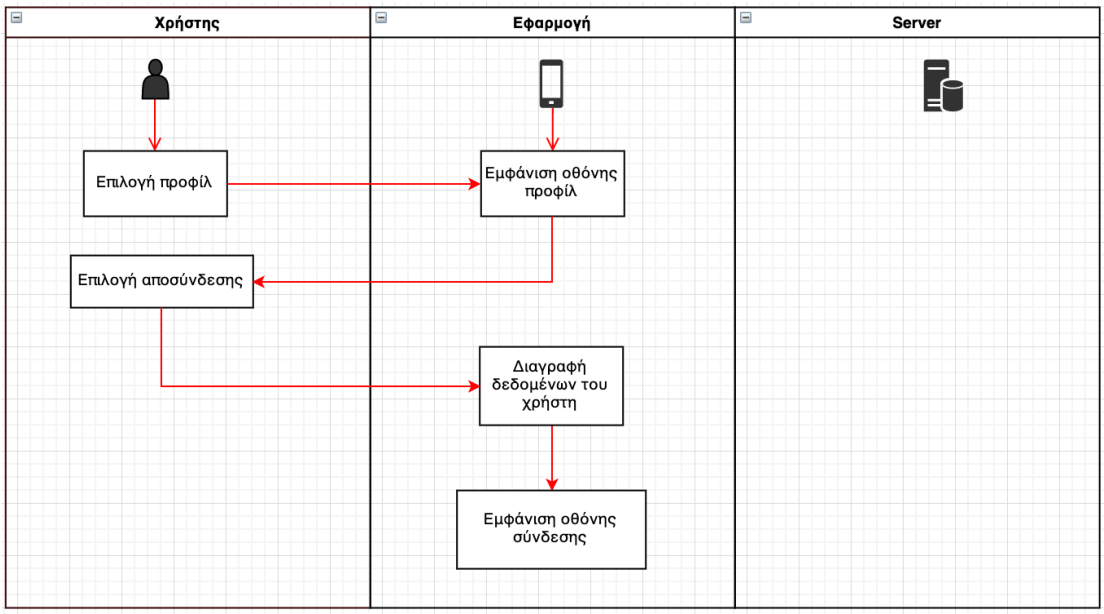
Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθεί ο χρήστης για να εγγραφεί ή να απεγγραφεί από ένα μάθημα.



Εικόνα 14 *Σενάριο εγγραφής σε νέο μάθημα.*

3.3.8 Προφίλ χρήστη

Το σενάριο αυτό αναπαριστά τη διαδικασία που ακολουθεί ο χρήστης για να δει τα προσωπικά του στοιχεία και να αποσυνδεθεί από το σύστημα, αν αυτός το επιθυμεί.



Εικόνα 15 *Σενάριο προβολής προφίλ χρήστη.*

4

Σχεδίαση

4.1 Η Εφαρμογή myUniLife

4.1.1 Οι οθόνες της εφαρμογής

Καθοριστικό ρόλο στη σχεδίαση του συστήματος, έπαιξε ο αρχικός σχεδιασμός των βασικών διεπιφανειών χρήστη, καθώς βοήθησε στην κατηγοριοποίηση και στην οργάνωση των κύριων λειτουργιών της εφαρμογής. Στη συνέχεια γίνεται μια σύντομη περιγραφή των λειτουργιών, της κάθε οθόνης, και έπειτα παρουσιάζεται η οργάνωση και η σχεδίαση της αρχιτεκτονικής της εφαρμογής.

Οθόνη σύνδεσης και εγγραφής

Αποτελεί την πρώτη οθόνη με την οποία έρχεται σε επαφή ο χρήστης της εφαρμογής. Στο σημείο αυτό, ο χρήστης έχει τη δυνατότητα να πραγματοποιήσει εγγραφή, στην περίπτωση που θέλει να δημιουργήσει λογαριασμό ή σύνδεση στην περίπτωση που έχει ήδη έναν ενεργό λογαριασμό.

	Όνομα	**Τύπος**	**Περιγραφή**
Κλάση	LoginRegisterVC	UIViewController	
Μέθοδοι	loginAction	((String, String) -> ())?	Εκτέλεση σύνδεσης με ήδη υπάρχων λογαριασμό
	registerAction	((String, String) -> ())?	Εγγραφή στο σύστημα

Οθόνη προβολής προγράμματος

Η οθόνη αυτή είναι αρμόδια να προβάλει το εβδομαδιαίο πρόγραμμα των μαθημάτων του χρήστη, είτε αυτός είναι φοιτητής είτε καθηγητής, μαζί με τα

απαραίτητα στοιχεία αυτών, όπως είναι οι ώρες διδασκαλίας, ο διδάσκων και η τοποθεσία του μαθήματος. Τα μάθημα παρουσιάζονται χωρισμένα σύμφωνα με την ημέρα διδασκαλίας, και πατώντας πάνω τους ο χρήστης μεταβαίνει στη σελίδα του αντίστοιχου μαθήματος.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	HomeVC	UIViewController	
Μέθοδοι	openCourse	((CourseModel) -> ())?	Μετάβαση στην οθόνη του μαθήματος
	registerToNewCourse Action	Optional Closure	Ο χρήστης μεταβαίνει στην οθόνη των μαθημάτων για να εγγραφεί σε νέο μάθημα, σε περίπτωση που δεν έχει εγγραφεί σε κανένα

Οθόνη προβολής όλων των μαθημάτων

Με την μετάβαση στην οθόνη αυτή ο χρήστης, ως φοιτητής, έχει τη δυνατότητα να προβάλλει όλα τα μαθήματα στα οποία είναι εγγεγραμμένος, επιλέγοντας όποιο εξάμηνο επιθυμεί. Τα μαθήματα εμφανίζονται χωρισμένα ανά εξάμηνο, αλλά και ανά ροή κατεύθυνσης. Ακόμα, ο χρήστης μπορεί να δει όλα τα μαθήματα που ακολουθεί συνολικά, αποκτώντας έτσι μια πιο ολοκληρωμένη εικόνα από την πρόοδο που έχει έως τώρα στη σχολή. Ομοίως, ο χρήστης ως καθηγητής, μπορεί να δει τα μαθήματα που διδάσκει. Από την άλλη μεριά, η γραμματεία, όντας διαχειριστής του τμήματος, είναι ικανή να δει και να επεξεργαστεί όλα τα μαθήματα της σχολής, καθώς και να δημιουργήσει νέα.

Επιπλέον, μέσα από την οθόνη αυτή, δίνεται η δυνατότητα στον φοιτητή να προβάλλει όλα τα μαθήματα του τμήματος, έτσι ώστε να πραγματοποιήσει εγγραφή σε κάποιο νέο ή απεγγραφή από κάποιο ήδη υπάρχων.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	CoursesVC	UIViewController	
Μέθοδοι	didSelectCourseAction	((CourseModel) -> ())?	Μετάβαση στην οθόνη του μαθήματος

	addNewCourseAction	Optional Closure	Μετάβαση στη φόρμα δημιουργίας νέου μαθήματος (μόνο ο διαχειριστής)
	editCourseAction	((CourseModel) -> ())?	Μετάβαση στη φόρμα δημιουργίας επεξεργασίας μαθήματος (μόνο ο διαχειριστής)
	registerNewCoursesAction	(([String],[String])->())?	Εγγραφή/ Απεγγραφή από μαθήματα (μόνο ο φοιτητής)

Κεντρική οθόνη μαθήματος

Στην οθόνη αυτή παρουσιάζονται τα γενικά στοιχεία του μαθήματος, τα οποία ενημερώνει ο καθηγητής, ή η γραμματεία. Μέσα από την οθόνη αυτή, οι φοιτητές μαθαίνουν το αντικείμενο που διδάσκεται και έτσι αποκτούν μια ιδέα σχετικά με τι έχει να κάνει το μάθημα. Αυτό έχει ως σκοπό να βοηθήσει τους φοιτητές να διαλέξουν νέα μαθήματα στην αρχή κάθε εξαμήνου, αλλά και να κρατά ενημερωμένους τους ήδη εγγεγραμμένους φοιτητές.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	CourseVC	UIViewController	
Μέθοδοι	saveCourseDescriptionAction	((String) -> ())?	Ενημέρωση των γενικών στοιχείων του μαθήματος

Γενικές οθόνες μαθήματος

Επιλέγοντας το μενού, που εμφανίζεται στο πάνω δεξιό μέρος της οθόνης που αναφέρθηκε ανωτέρω, παρέχεται η δυνατότητα στους χρήστες να μεταβούν σε μία από τις παρακάτω οθόνες που αφορούν το συγκεκριμένο μάθημα :

- Οθόνη προγράμματος, όπου προβάλλεται το εβδομαδιαίο πρόγραμμα μόνο του συγκεκριμένου μαθήματος.

• Οθόνη εγγράφων, στην οποία ο καθηγητής του μαθήματος έχει τη δυνατότητα να ανεβάσει αρχεία, όπως σημειώσεις και διδακτικό υλικό.

• Οθόνη εργασιών, μέσα από την οποία ο καθηγητής αναρτά νέες εργασίες προς παράδοση και κατεβάζει ήδη υποβεβλημένες εργασίες των φοιτητών, τις οποίες και βαθμολογεί άμεσα από το ίδιο σημείο. Οι φοιτητές από την άλλη μεριά, έχουν την δυνατότητα να κατεβάσουν και να προβάλλουν τις εκφωνήσεις των εργασιών, καθώς και να υποβάλουν λύσεις αυτών.

• Οθόνη καθηγητών, όπου οι φοιτητές αλλά και οι καθηγητές μπορούν να δουν πληροφορίες σχετικά με τους διδάσκοντες του μαθήματος, όπως όνομα, επώνυμο και διεύθυνση ηλεκτρονικού ταχυδρομείου.

• Οθόνη φοιτητών, στην οποία οι καθηγητές βλέπουν το πλήθος αλλά και πληροφορίες σχετικά με τους φοιτητές που έχουν εγγραφεί στο συγκεκριμένο μάθημα. Έχοντας τις πληροφορίες αυτές, οι διδάσκοντες μπορούν να προετοιμαστούν καλύτερα για τις εξετάσεις, εκτυπώνοντας ακριβή αριθμό γραπτών.

• Οθόνη βαθμολογίας, η οποία επιτρέπει στους διδάσκοντες να αναζητήσουν φοιτητές, σύμφωνα με τον αριθμό μητρώου ή το ονοματεπώνυμο τους, και στη συνέχεια να προσθέσουν τον βαθμό που τους αναλογεί για την τρέχουσα εξεταστική περίοδο.

οθόνη εγγράφων

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	CourseDocumentsVC	UIViewController	
Μέθοδοι	uploadFileAction	Optional Closure	Επιλογή εγγράφου προς ανέβασμα (μόνο καθηγητές)
	downloadAction	((String, String)->())?	Κατέβασμα επιλεγμένου αρχείου

οθόνη εργασιών

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	AssignmentsVC	UIViewController	

Μέθοδοι	submitAssignmentAction	Optional Closure	Επιλογή αρχείου εκφώνησης ή λύσεων προς ανέβασμα
	downloadAction	((String, String)->())?	Κατέβασμα επιλεγμένης εργασίας (είτε εκφώνησης, είτε λύσης)
	addGradeToAssignment	((String)->())?	Προσθήκη ή επεξεργασία βαθμού εργασίας (μόνο καθηγητές)

οθόνη καθηγητών/φοιτητών

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	PersonsListVC	UIViewController	
Μέθοδοι	openUserProfile	((UserModel)->())?	Προβολή στοιχείων καθηγητή ή φοιτητή

Οθόνη ανακοινώσεων σχολής

Στην οθόνη αυτή εμφανίζεται μια λίστα από τις ανακοινώσεις του τμήματος που έχουν αναρτηθεί στην ιστοσελίδα της σχολής. Ο χρήστης, αφού επιλέξει μια από αυτές μεταβαίνει με έναν, ενσωματωμένο στην εφαρμογή, φυλλομετρητή ιστού στην αντίστοιχη ιστοσελίδα της ανακοίνωσης.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	AnnouncementsVC	UIViewController	
Μέθοδοι	openAnnouncementAction	((Int)->())?	Ανακατεύθυνση στην επιλεγμένη ανακοίνωση στην ιστοσελίδα της σχολής

Οθόνη βαθμολογίας

Ανάλογα με το ρόλο που έχει ο χρήστης στην εφαρμογή, η οθόνη αυτή προσφέρει διαφορετικές λειτουργίες.

• Ο φοιτητής μπορεί να δει όλες τις βαθμολογίες που έχει λάβει έως τώρα στις εξεταστικές περιόδους, καθώς και να διακρίνει εύκολα σε ποια μαθήματα δεν έχει λάβει προβιβάσιμο βαθμό.

• Ο καθηγητής έχει τη δυνατότητα να επιλέξει, μεταξύ των μαθημάτων που διδάσκει, σε ποιο μάθημα θέλει να προσθέσει βαθμολογία, και στη συνέχεια, αφού επιλέξει τον επιθυμητό φοιτητή, να τον βαθμολογήσει.

• Η γραμματεία, ως διαχειριστής όλων των μαθημάτων, έχει το δικαίωμα να δει όλα τα διδασκόμενα μαθήματα και έπειτα να βαθμολογήσει όποιον φοιτητή επιθυμεί.

οθόνη αναζήτησης χρηστών (και βαθμολογίας)

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	PersonsSearchVC	UIViewController	
Μέθοδοι	addGradeAction	((String, Int)->())?	Προσθήκη βαθμού εξεταστικής (μόνο καθηγητές)
	addProfessorInCourseAction	((UserModel)->())?	Ανάθεση καθηγητή σε μάθημα (μόνο διαχειριστής)

Οθόνη προσωπικών στοιχείων χρήστη

Η οθόνη αυτή εμφανίζεται σε διάφορα σημεία κατά την πλοήγηση στην εφαρμογής. Αποτελεί την οθόνη μέσα από την οποία ο χρήστης βλέπει τα προσωπικά του στοιχεία, όπως ονοματεπώνυμο, διεύθυνση ηλεκτρονικού ταχυδρομείου, αριθμό μητρώου και τηλέφωνο. Επιπλέον, δίνει τη δυνατότητα αποσύνδεσης του χρήστη από την εφαρμογή. Η ίδια οθόνη, με τις απαραίτητες τροποποιήσεις, χρησιμοποιείται και όταν ένας χρήστης προβάλει τα στοιχεία ενός καθηγητή ή ενός φοιτητή.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	ProfileVC	UIViewController	

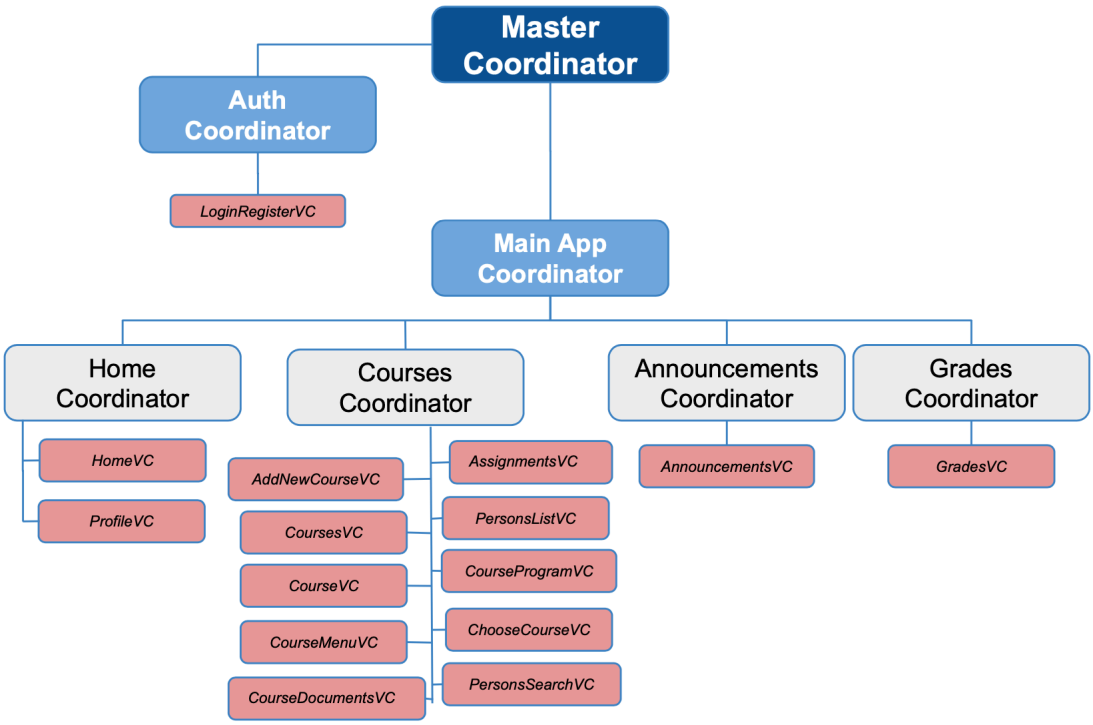
Μέθοδοι	logoutAction	((String, Int)->())?	Προσθήκη βαθμού εξεταστικής (μόνο καθηγητές)
	closeAction	Optional Closure	Κλείσιμο της παρούσας οθόνης

Οθόνη δημιουργίας και επεξεργασίας μαθημάτων

Η δημιουργία και η επεξεργασία των μαθημάτων γίνεται με τη χρήση μιας φόρμας, στην οποία η γραμματεία μπορεί να προσθέσει τα απαραίτητα στοιχεία. Αυτά είναι ο κωδικός, ο τίτλος, το εξάμηνο, η κατηγορία (προπτυχιακό, μεταπτυχιακό), η κυριότητα (υποχρεωτικό, κατ' επιλογήν υποχρεωτικό, υποχρεωτικό ολόκληρης ροής, υποχρεωτικό μισής ροής), η ροή κατεύθυνσης, η αίθουσα διδασκαλίας, η περιγραφή, η διάρκεια, και το εβδομαδιαίο πρόγραμμα του μαθήματος. Επίσης, από την ίδια φόρμα γίνεται και η ανάθεση ενός καθηγητή σε ένα μάθημα.

	Όνομα	***Τύπος***	***Περιγραφή***
Κλάση	AddNewCourseVC	UIViewController	
Μέθοδοι	addNewCourseAction	((String])->())?	Προσθήκη νέου μαθήματος
	editCourseAction	((String])->())?	Επεξεργασία μαθήματος
	addProfessorAction	Optional Closure	Ανάθεση καθηγητή σε μάθημα
	addScheduleAction	((Int,String,String,String?)->())?	Προσθήκη προγράμματος
	openCourseScedule Action	Optional Closure	Προβολή ήδη υπάρχοντος προγράμματος

4.1.2 Σχεδίαση της αρχιτεκτονικής



Εικόνα 16 *Γραφική απεικόνιση αρχιτεκτονικής.*

Η αρχιτεκτονική του συστήματος βασίστηκε στο σχεδιαστικό πρότυπο των **Συντονιστών**, διότι όλη η βασική πλοήγηση της εφαρμογής γίνεται με τη χρήση ενός **Tab Bar** τεσσάρων καρτελών. Για κάθε καρτέλα σχεδιάστηκε ένας ξεχωριστός συντονιστής, ο οποίος είναι υπεύθυνος να την ενημερώνει με τα στοιχεία που χρειάζεται, και να εκτελεί κάθε ενέργεια της.

Επιπλέον, εκτός από τους τέσσερις αυτούς συντονιστές, απαραίτητος ήταν και ο σχεδιασμός άλλων τριών, ευθύνη των οποίων είναι η διαχείριση αυτών.

- **Master Coordinator**, ο οποίος έχει ως κύριο σκοπό την αρχικοποίηση του συστήματος και του παραθύρου προβολής της εφαρμογής.
- **Auth Coordinator**, ο οποίος είναι υπεύθυνος για την πιστοποίηση του χρήστη.
- **Main App Coordinator**, ο οποίος αρχικοποιεί και διαχειρίζεται τους τέσσερις βασικούς συντονιστές της εφαρμογής.

4.2 Ο server του συστήματος

4.2.1 Η βάση δεδομένων

Επιλέχθηκε μια σχεσιακή σύγχρονη βάση δεδομένων που να προσφέρει ταχύτητα και ασφάλεια και πιο συγκεκριμένα η PostgreSQL στη έκδοση 11.4

Όλα τα tables στη βάση δεδομένων συνοδεύονται από ένα id, αλλά και από ένα uuid (Universal Unique Identifier). Η χρήση των uuid τα οποία είναι random generated και μπορούν να μεταφερθούν αυτούσια σε μια άλλη βάση ή ακομα και να τροποποιηθούν σε σύγκριση με τα ids τα οποία παράγονται αυτόματα και με αύξουσα σειρά εισαγωγής στη βάση δεδομένων προσφέρει αρκετά πλεονεκτήματα. Αποτελούν μοναδικά χαρακτηριστικά σε αντίθεση με τα id τα οποία είναι κοινα μεταξύ δύο διαφορετικών tables στη βάση, η μορφοποίηση τους είναι κοινή για κάθε είδους βάση ενώ ορισμένες μπορεί να μην υποστηρίζουν τη αυτόματη δημιουργία ενός αύξοντα ακεραίου ενώ τέλος η χρήση ενός κατανεμημένου συστήματος βάσης δεδομένων καθίσταται αδύνατη με τη χρήση ids κατι το οποίο όμως τα uuids το επιτρέπουν.

4.2.2 Αποθήκευση αρχείων

Μέσω του συστήματος myUniLife επιτρέπεται στους χρήστες η αποθήκευση και ο διαμοιρασμός αρχείων, τόσο με τη μορφή σημειώσεων, όσο και υπο την μορφή ασκήσεων (λύσεις και εκφωνήσεις) για το κάθε μάθημα. Για την ανάγκη αυτή, επιλέχθηκε προς χρήση η υπηρεσία s3 της amazon. Τα αρχεία θα αποθηκεύονται υπο τη μορφή αντικειμένων και αντιπροσωπεύονται από ένα μοναδικό χαρακτηριστικό κωδικό ταυτοποίησης.

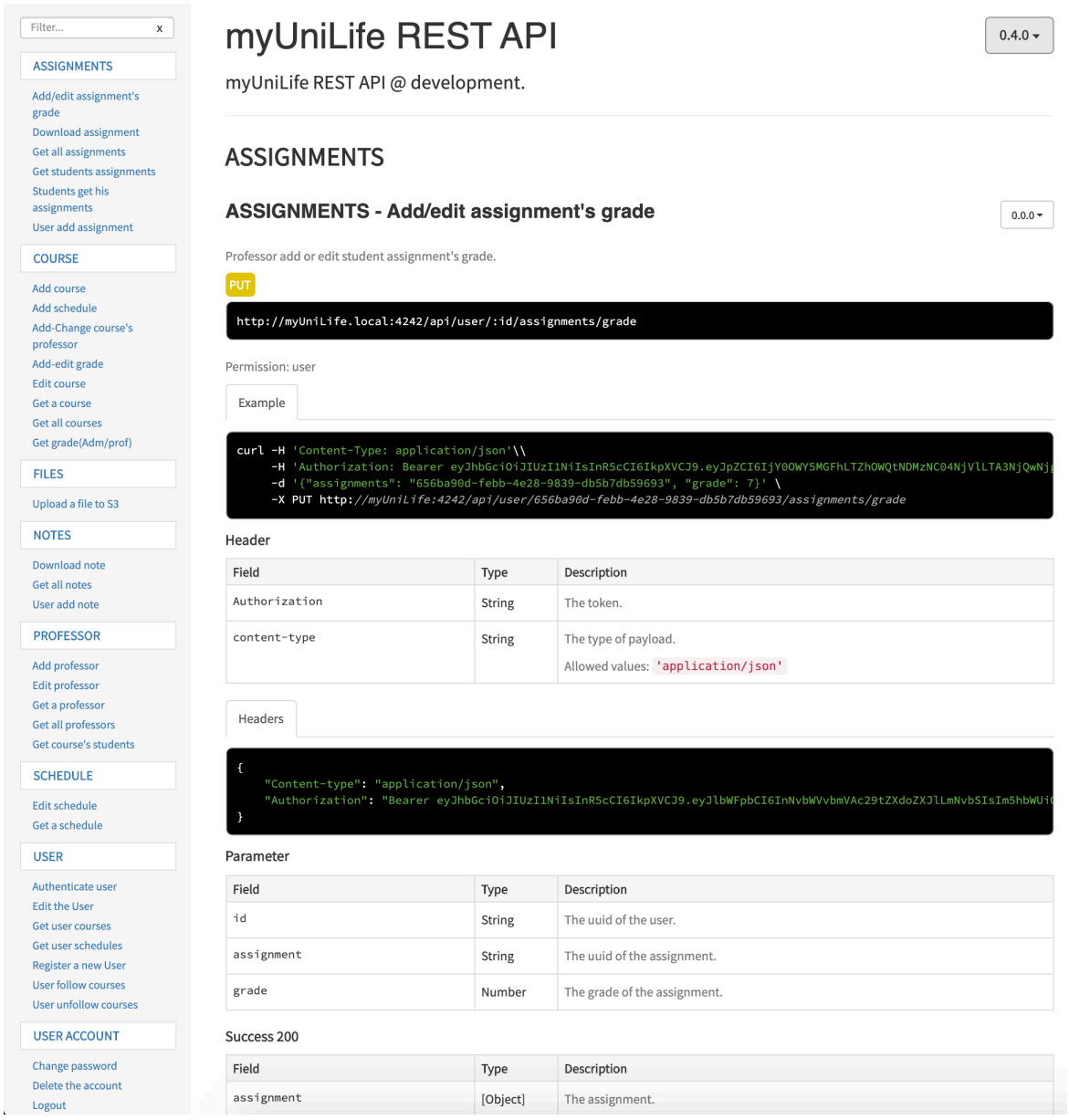
Αυτός ο κωδικός αποθηκεύεται εν τέλη στη βάση, ώστε το σύστημα να είναι σε θέση να αντιστοιχίσει ένα αρχείο με τον τίτλο και τον ιδιοκτήτη του.

4.2.3 Docker

Για τη βάση δεδομένων του συστήματος, αλλά και για το localstack που προσομοιώνει την υπηρεσία s3 της amazon χρησιμοποιήθηκε η τεχνολογια του docker. Πιο συγκεκριμένα, δημιουργείται ένα docker container για τη postgreSQL και ένα για localstack. Για τον χειρισμό αυτών αξιοποιήθηκε το εργαλείο docker-compose το οποίο επιτρέπει τον ορισμό και εκτέλεση εφαρμογών πολλαπλών containers μέσω ενος αρχείου yaml όπου και ορίστηκαν τα δύο containers για τις προαναφερθείσες υπηρεσίες.

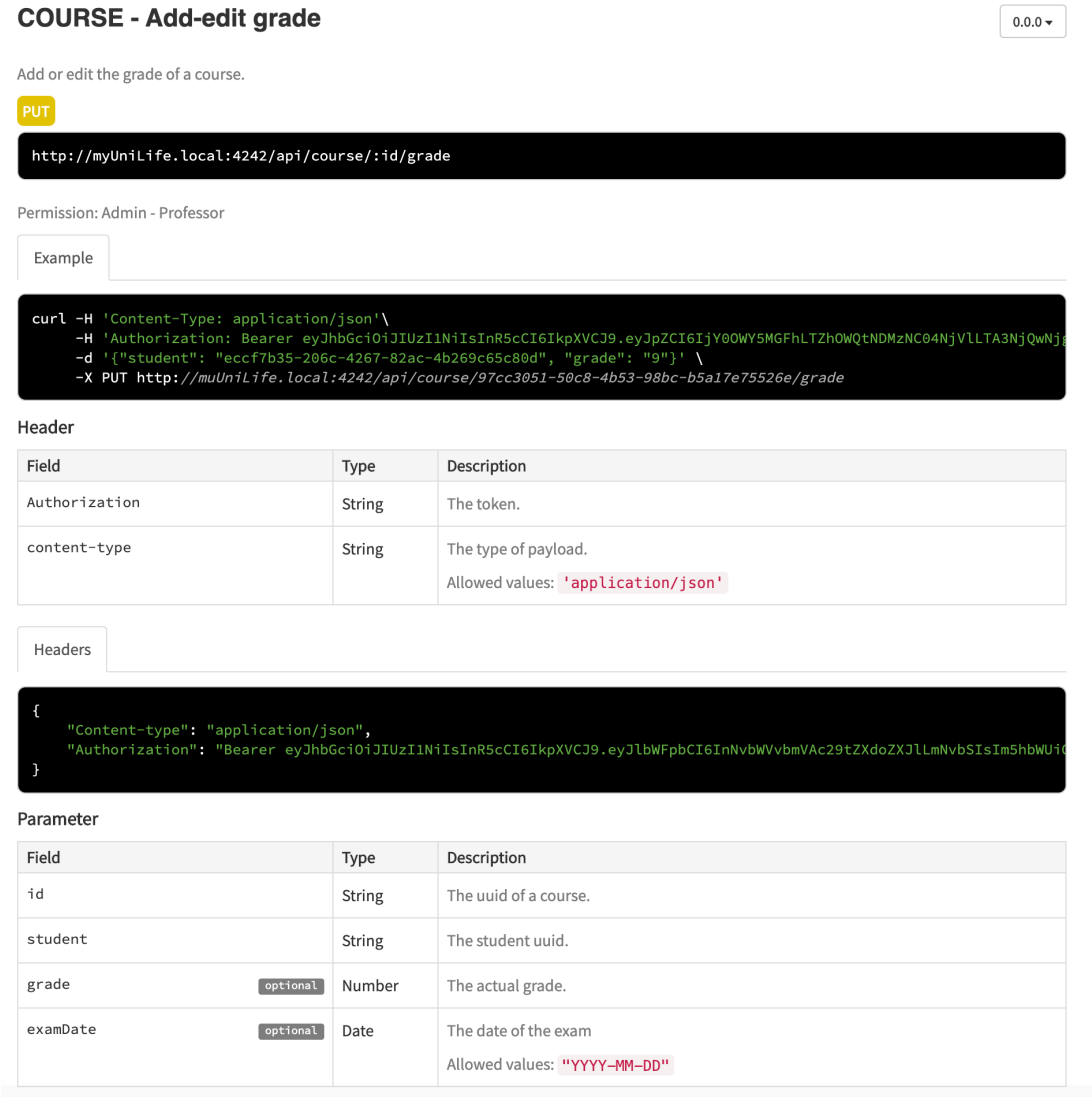
4.2.4 NodeJS - restful APIs

Η επικοινωνία μεταξύ της εφαρμογής και του server, που απαιτείται για τον συγχρονισμό των δεδομένων, πραγματοποιείται μέσω ενός **restful API**. Στη συνέχεια, παρουσιάζεται ένα χαρακτηριστικό στιγμιότυπο της λεπτομερής σχεδίασης όλων των **endpoints**. Το ακόλουθο documentation δημιουργήθηκε με τη χρήση του πακέτου **api-doc**, και προβάλλεται μέσω ενός browser.



Εικόνα 17 *Rest Api documentation.*

Παρακάτω αναλύεται η δομή και χρήση ενός endpoint, και συγκεκριμένα, εκείνου που χρησιμοποιείται για την προσθήκη και επεξεργασία του βαθμού ενός φοιτητή στο μάθημα.



Εικόνα 18 *Rest Api documentation (request).*

Αρχικά, αναγράφονται το ακριβές όνομα και η περιγραφή του endpoint, καθώς και το url επικοινωνίας, σε συνδυασμό με το είδος του request που απαιτείται (PUT). Έπειτα, ακολουθεί ένα χαρακτηριστικό παράδειγμα χρήσης του, μέσω την εντολής **curl**, και οι παράμετροι της κεφαλίδας που απαιτούνται. Τέλος, αναλύονται οι παράμετροι του σώματος του request, και συγκεκριμένα το είδος τους, οι προκαθορισμένες τιμές τους, η μορφή και μια σύντομη περιγραφή τους.

Success 200

Field	Type	Description
grade	[Object]	The garde.
uuid	String	The uuid of the grade.
grade	Number	The user's grade for the course.
examDate	Date	The exam date of the course.
pass	Boolean	If the user has passed or not the course. Allowed values: `true`, `false`
createdAt	Date	When the grade was created.
updatedAt	Date	When the grade was last updated.

Success

```
HTTP/1.1 200 OK
{
  "grade": {
    "uuid": "eccf7b35-206c-4267-82ac-4b269c65c80d",
    "grade": "7",
    "examDate": "2018-09-20",
    "pass": "true",
    "createdAt": "2018-09-20T05:03:44.570Z",
    "updatedAt": "2018-09-20T05:03:44.570Z"
  }
}
```

Error 400

Name	Description
COURSE_NOT_FOUND	The course was not found.
STUDENT_NOT_FOUND	The student was not found.
MISSING_ID	The id was missing.
INVALID_ID	The id was invalid.
MISSING_STUDENT	The student uuid was missing.
INVALID_STUDENT	The student uuid was invalid.
MISSING_GRADE	The grade was missing.
INVALID_GRADE	The grade was invalid.
MISSING_EXAM_DATE	The examDate was missing.
INVALID_EXAM_DATE	The examDate was invalid.

Εικόνα 19 *Rest Api documentation (response).*

Το documentation συνεχίζει με ένα παράδειγμα επιτυχούς απάντησης του request, και των παραμέτρων που θα αποσταλούν πίσω στον client που το πραγματοποίησε. Τέλος, αναλύονται οι πιθανές περιπτώσεις αποτυχίας του αιτήματος, όπως για παράδειγμα αποστολή λανθασμένου κωδικού μαθήματος, ή παράλειψη αποστολής του βαθμού.

5

Ανάπτυξη συστήματος

5.1 Ανάπτυξη του Front-End

5.1.1 Κύρια δομή συστήματος

Όπως αναφέρθηκε και στην σχεδίαση του συστήματος, για την ανάπτυξη της εφαρμογής έγινε χρήση του σχεδιαστικού προτύπου των συντονιστών. Η αρχικοποίηση του κύριου συντονιστή (Master Coordinator), έγινε μέσα από την κύρια κλάση στο iOS, την **AppDelegate.**

```
class AppDelegate: UIResponder, UIApplicationDelegate {

    var window: UIWindow?

    lazy var appCoordinator: MasterCoordinator = {
        return MasterCoordinator(in: self.window!)
    }()


    func application(_ application: UIApplication, didFinish-
LaunchingWithOptions launchOptions: [UIApplication.LaunchOptionsKey:
Any]?) -> Bool {

        // Keyboard Manager
        IQKeyboardManager.shared.enable = true


        window = UIWindow(frame: UIScreen.main.bounds)
        appCoordinator.start()

        return true
    }
// Συνέχιση της κλάσης AppDelegate . . .
}
```

Υλοποίηση του πρωτοκόλλου Coordinator

```
protocol Coordinator: class {
    var navigationController: UINavigationController? { get }
    var childCoordinators: [CoordinatorKeys:Coordinator]
     { get set }

    func start()
    func addChild(coordinator: Coordinator,
                    with key: CoordinatorKeys)
    func removeChild(coordinator: Coordinator)
}

extension Coordinator {
    func addChild(coordinator: Coordinator, with
        key: CoordinatorKeys) {

        childCoordinators[key] = coordinator
    }

    func removeChild(coordinator: Coordinator) {
        childCoordinators = childCoordinators.filter {
            $0.value !== coordinator
        }
    }

    func removeChild(_ key: CoordinatorKeys) {
        if let coord = childCoordinators[key] {
            removeChild(coordinator: coord)
        }
    }
}
```

Βασικές μέθοδοι του κύριου συντονιστή Master Coordinator

```
init(in window: UIWindow) {
     self.navigationController = MainAppNC()
     self.childCoordinators = [:]
     self.window = window
     window.backgroundColor = .white
     self.window.rootViewController = rootViewController
     self.window.makeKeyAndVisible()
}

// Έλεγχος αν υπάρχει ενεργή συνεδρία του χρήστη

public func handleAuthorization() {
      if DataContext.shared.userAuth != nil {
          goToMainApp()
      } else {
          goToAuth()
      }
}
```

```
    public func goToAuth() {
        removeChild(.mainAppCoordinator)
        let authCoordinator = AuthCoordinator(with: navigationController,
                                              andParent: self)
        addChild(coordinator: authCoordinator, with: .authCoordinator)
        authCoordinator.start()
        self.window.rootViewController =
                        authCoordinator.rootViewController
    }

    public func goToMainApp() {
        AlertHelper.showLoader()
        DataContext.shared.initialize(completion: { (success) in
            AlertHelper.hideLoader()
            if success {
                self.removeChild(.authCoordinator)
                let mainAppCoordinator = MainAppCoordinator(with:
                                         self.navigationController,
                                         andParent: self)
                self.addChild(coordinator: mainAppCoordinator,
                              with: .mainAppCoordinator)
                mainAppCoordinator.start()
                self.window.rootViewController =
                              mainAppCoordinator.rootViewController
            } else {
                self.goToAuth()
            }
        })
    }
```

Ενδεικτική μέθοδος μετάβασης σε νέα οθόνη

```
    internal func showGrades() {

        let gradesVC = GradesVC()
        navigate(to: gradesVC, with: .push)

        gradesVC.refreshGradesAction = {
            DataContext.shared.getMyCourses(completion: { _ in
                gradesVC.updateUI()
            })
        }

        navigationController?.viewControllers = [gradesVC]
    }
```

5.1.2 Διαχείριση δεδομένων εφαρμογής

Για να μπορεί η εφαρμογή να έχει άμεσα, και σε οποιοδήποτε σημείο, τα δεδομένα που απαιτούνται, δημιουργήθηκε η κλάση **DataContext**. Έχει βασιστεί στο πρότυπο **singleton**, γεγονός που την καθιστά εύκολα προσβάσιμη από οποιοδήποτε σημείο του κώδικα.

```
class DataContext {
    static let shared = DataContext()


    var allCourses: [CourseModel]?
    var myCourses: [CourseModel]?
    var allProfessors: [UserModel]?
    var todayCourses: [CourseModel]?

    var user: UserModel?

    var currentUserRole: UserRole {
        return user?.userRole ?? UserRole.student
    }

    func initialize(completion: ((Bool) -> Void)? = nil){
        let group = DispatchGroup()

        switch currentUserRole {
        case .student:
            group.enter()
            getSchedules(semesters: [1,2,3,4,5,6,7,8,9],
                         days: [1,2,3,4,5],
                         completion: { (_) in
                group.leave()
            })

            group.enter()
            getMyCourses { (_) in
                group.leave()
            }

            group.enter()
            getAllCourses { (_) in
                group.leave()
            }

            group.notify(queue: .main) { [weak self] in
                let success = self?.todayCourses != nil
                completion?(success)
            }
    // Συνέχεια κλάσης. . .
}
```

5.1.3 Ασφάλεια εφαρμογής

Για την ασφάλεια των ευαίσθητων πληροφοριών του χρήστη, όπως είναι το **κλειδί πιστοποίησης (token)**, χρησιμοποιήθηκε το API υπηρεσιών **keychain**, το οποίο προσφέρει έναν μηχανισμό για την αποθήκευση πληροφοριών σε μια κρυπτογραφημένη βάση δεδομένων, που ονομάζεται keychain.

```
import Foundation

extension DataContext {
    enum KeychainKey: String {
        case userAuth = "USER_AUTH"

        var key: String {
            return self.rawValue
        }
    }

    var userAuth: AuthModel? {
        get {
            guard let authData = Keychain.read(dataWithKey:
KeychainKey.userAuth.key) else { return nil }
            do {
                let userAuth = try JSONDecoder().decode(Auth-
Model.self, from: authData)
                return userAuth
            } catch {
                print(error)
                return nil
            }
        }
        set {
            guard let newValue = newValue else {
                Keychain.delete(valueWithKey: KeychainKey.user-
Auth.key)
                return
            }
            do {
                let authData = try JSONEncoder().encode(newValue)
                Keychain.store(data: authData, withKey: Keychain-
Key.userAuth.key)
            } catch {
                print(error)
            }
        }
    }

}
```

5.1.4 Διαχειριστής πακέτων Cocoapods

Ο διαχειριστής πακέτων **CocoaPods** είναι αρμόδιος για τη συλλογή και την ενσωμάτωση βιβλιοθηκών τρίτων (**third party libraries**) στο Xcode, οι οποίες στη συνέχεια, βοηθούν και επιταχύνουν την ανάπτυξη το έργου. [21]

Στην εφαρμογή **myUniLife** χρησιμοποιήθηκαν τα παρακάτω πακέτα :

- Alamofire, για τις επικοινωνίες μέσω δικτύου
- CodableAlamofire, για την διαχείριση των απαντήσεων του server
- SVProgressHUD, για τον loader της εφαρμογής
- BotMot, για τη μορφοποίηση του κειμένου
- IQKeyboardManagerSwift, για λειτουργίες πληκτρολογίου
- Eureka, για την φόρμα προσθήκης νέου μαθήματος

```
# Uncomment the next line to define a global platform for your project
platform :ios, '12.0'

target 'myUniLife' do
  # Comment the next line if you're not using Swift and don't want to
use dynamic frameworks
  use_frameworks!

  # Pods for myUniLife
  pod 'Alamofire', '4.7.3'
  pod 'CodableAlamofire', '1.1.0'
  pod 'SVProgressHUD', '2.2.5'
  pod 'BonMot', '5.3'
  pod 'IQKeyboardManagerSwift', '6.2.0'
  pod 'Eureka', '4.3.1'

end
```

5.1.5 Επικοινωνία με το Back-End

Για την επικοινωνία με τον server της εφαρμογής χρειάστηκε να υλοποιηθούν δύο βασικές κλάσεις, η κλάση **PathRouter** και η κλάση **Requests**.

Στην κλάση **PathRouter** ορίζονται όλα τα αιτήματα προς τον server, μαζί με τις παραμέτρους απαιτούνται για την εκτέλεση τους.

```
import Alamofire

enum PathRouter: URLRequestConvertible {

    static let baseURLPath = AppConfiguration.apiBaseUrl

    case register(username: String, password: String)
    case userLogin(username: String, password: String)
    case getMyCourses
    case getAllCourses
    case followCourses(courses: [String])

    // . . .
    var method: HTTPMethod {
        switch self {
        case .userLogin  return .post
        case .getMyCourses, .getAllCourses return .get
        case .followCourses: return .put
        }
    }
    var path: String {
        switch self {
        case .register:
            return "/user/register"
        case .userLogin:
            return "/user/authenticate"
        case .getMyCourses:
            return "user/\(DataContext.shared.userAuth?.uuid ??
"")/courses"
        case .getAllCourses:
            return "/course"
        case .followCourses:
            return "user/\(DataContext.shared.userAuth?.uuid ??
"")/courses/follow"
        }
    }
    var encoding: ParameterEncoding {
        switch method {
        case .put, .post, .get:
            switch self {
            case .register, .userLogin, .getMyCourses, .getAll-
Courses,
                    .followCourses:
                return JSONEncoding.default
            }
        default:
            return URLEncoding.queryString
        }
    }
    var headers: [String : String] {
        var headers = [
            "Content-Type": "application/json"
        ]
        switch self {
        case .getMyCourses, .getAllCourses,.followCourses:
           if let auth = DataContext.shared.userAuth {
                headers["Authorization"] = "Bearer \
(auth.tokenId!)"
            }
        default:break}
        return headers } // . . .
```

Στην κλάση **Requests** υλοποιούνται όλα τα αιτήματα προς τον server, έτσι ώστε να είναι διαθέσιμα, προς εκτέλεση, από την κλάση **DataContext**.

```
class Requests {

    static let sessionManager: SessionManager = {
        let sessMan = SessionManager()
        return sessMan
    }()

static func getMyCourses(completion: @escaping ([CourseModel]?)->Void) {
        sessionManager.request(PathRouter.getMyCourses)
            .validate().responseDecodableObject { (response: DataResponse<[CourseModel]>) in
                print(response)
                switch response.result {
                case .success(let myCourses):
                    completion(myCourses)
                case .failure(let error):
                    print(error.localizedDescription)
                    completion(nil)
                    break
                }
        }
    }
} . . .
```

5.2 Ανάπτυξη του Back-End

5.2.1 Η βάση δεδομένων

Οντότητες

User

Για όλους του πιθανούς users της εφαρμογής χρησιμοποιείται ένας κοινός πίνακας Users. Το είδος του χρήστη καθορίζεται από το column "role" το οποίο λαμβάνει τις εξής τιμές:

- **STUDENT** Αντιπροσωπεύει τον φοιτητή μιας σχολής
- **PROFESSOR** Αντιπροσωπεύει έναν καθηγητή
- **ADMIN** Αντιπροσωπεύει τον διαχειριστή της σχολής που συνήθως είναι η γραμματεία

Department

Για λόγους μελλοντικής ανάπτυξης και επέκτασης του συστήματος ο κάθε χρήστης ανήκει σε ένα συγκεκριμένο department το οποίο αντιπροσωπεύει τα διάφορα τμήματα που μπορεί να έχει ένα εκπαιδευτικό ίδρυμα. Κάθε department έχει έναν μόνο λογαριασμό χρήστη που θεωρείται διαχειριστής του.

Organization

Αντίστοιχα η βάση περιλαμβάνει και μια εγγραφή για κάθε τέτοιο ίδρυμα-πανεπιστήμιο (table: organization) με τα στοιχεία του.

Course

Το κάθε μάθημα για μία σχολή αναπαρίσταται από μία εγγραφή στον πίνακα course στην οποία υπάρχουν πληροφορίες για το ποιος καθηγητής το διδάσκει, σε ποιό τμήμα ανήκει, καθώς και το εξάμηνο, την περιγραφή, την ροή στην οποία μπορεί να ανήκει, το αν είναι υποχρεωτικο και τυχον εσωτερικό κωδικό του μαθήματος για τη σχολή. Τέλος με σκοπό στο μέλλον η εφαρμογή να εξυπηρετήσει και μεταπτυχιακούς φοιτητές καταγράφεται και ένα επιπλέον πεδίο graduation_level με πιθανές τιμές:

- POSTGRADUATED

- AFTERGRADUATED

User_Follow_Course

Όταν ένας φοιτητής επιλέγει να ακολουθήσει ένα μάθημα αυτό αντιπροσωπεύεται από μία εγγραφή στον πίνακα user_follow_course όπου αποθηκεύεται και ο τελικός τρέχων βαθμός του για αυτό, αν υπάρχει (ο μεγαλύτερος βαθμός που συγκέντρωσε σε όλες τις προσπάθειές του).

Grade

Οι βαθμοί ενός φοιτητή αποθηκεύονται στον πίνακα grade της βάσης όπου σώζεται και η ημερομηνία εξέτασης του μαθήματος και το αν ο φοιτητής έλαβε προβιβάσιμο βαθμό η οφείλει το μάθημα.

Schedule

Το πρόγραμμα κάθε μαθήματος με ημέρα, ώρα, και τοποθεσία αποθηκεύεται στον πίνακα schedule και μπορεί να υπάρχουν πολλαπλά entries για κάθε μάθημα.

Note

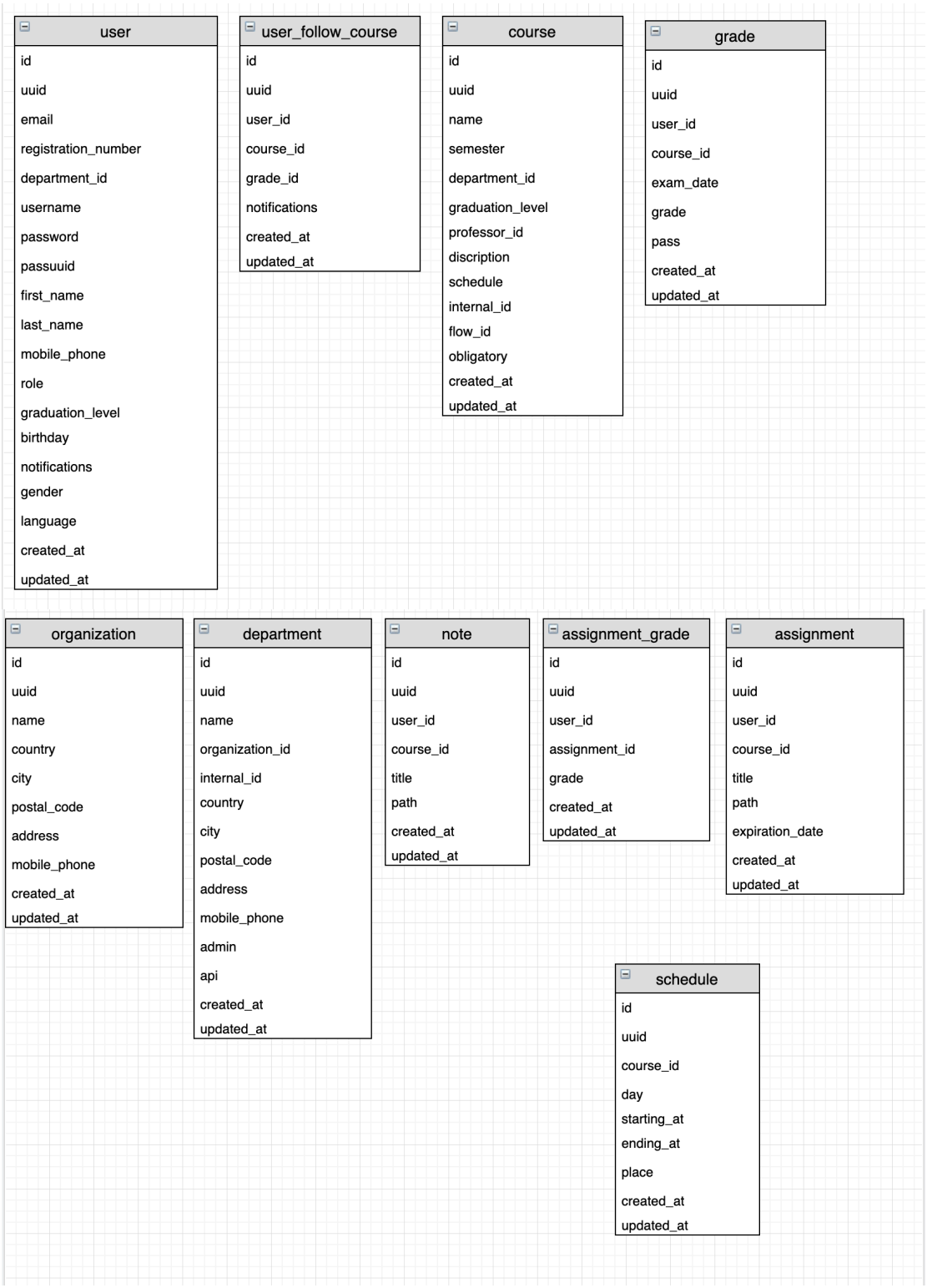
Ο καθηγητής κάθε μαθήματος έχει τη δυνατότητα να αναρτήσει υλικό σχετικό με το μάθημα στο οποίο η φοιτητές αργότερα έχουν πρόσβαση. Αυτό επιτυγχάνεται μέσω της εγγραφής “note” όπου αποθηκεύεται ο ιδιοκτήτης του εγγράφου, ο τίτλος του καθώς και η τοποθεσία στη οποία είναι αποθηκευμένο το έγγραφο. Στο σύστημα myUniLife επειδή όλα τα έγγραφα αποθηκεύονται στο s3 bucket στο αντίστοιχο πεδίο καταγράφεται το αναγνωριστικό του αρχείο (file id).

Assignment

Οι καθηγητές μπορούν επίσης για κάθε μάθημα τους να αναρτούν ασκήσεις προς επίλυση και οι φοιτητές έγγραφα με τις λύσεις τους για αυτές τις ασκήσεις. Και τα δύο αυτά είδη εγγράφων αποθηκεύονται στον στη βάση στον πίνακα assignments με παρόμοιο τρόπο με τις σημειώσεις.

Assignment_grade

Για τη βαθμολόγηση των λύσεων των φοιτητών από τους καθηγητές χρησιμοποιείται η εγγραφή assignment_grade όπου ο καθηγητής υποβάλει την τελική βαθμολογία του για κάθε φοιτητή που έχει υποβάλει λύση.



Εικόνα 20 *Η βάση δεδομένων της εφαρμογής.*

5.2.2 Αποθήκευση αρχείων στο s3 bucket

Για τις ανάγκες της διπλωματικής εργασίας και της διαδικασίας ανάπτυξης του συστήματος χρησιμοποιήθηκε ένα local stack που προσομοιώνει την ακριβής λειτουργία της υπηρεσίας s3-aws. Όλες οι διαδικασίες επικοινωνίας προσομοιώνουν επίσης την επικοινωνία με ένα αυθεντικό σύστημα s3 με μοναδική αλλαγή το τελικό endpoint των requests.
Για τα παραπάνω χρησιμοποιήθηκε το NodeJs module: aws-sdk

Συνάρτηση για ανάγνωση και αποθήκευση αρχείων

```
// Reads the file with the given FID.
const read = (fid) => {
  logger.debug(`System reads file ${ fid }.`);
  return Promise.try(() => {
    return s3.getObject({Key: fid}).promise();
  }).then((result) => {
    logger.debug(`System read file ${ fid }.`);
    // Detect the mime type of the file.
    return new Promise.Promise((resolve, reject) => {
      magic.detect(result.Body, (error, contentType) => {
        if (error) {
          reject(error);
        } else {
          resolve({contentType, data: result.Body});
        }
      });
    });
  }).then((result) => {
    // Return the content type and the actual data.
    logger.debug(`System read file ${ fid }.`);
    return result;
  }).catch((error) => {
    logger.error(`System failed to read file ${ fid }.`, error);
    throw error;
  });
};
```

5.2.3 Docker

Για τις docker υπηρεσίες που απαιτεί η εφαρμογή (με την αξιοποίηση του εργαλείου docker - compose) χρησιμοποιήθηκε το ακόλουθο YAML αρχείο.

```
version: '3.7'

# For now we have only one service for our datbase
services:
  db:
    image: postgres:11.4
    container_name: db-myunilife
    #postgres will run on port YYYY on docker container and i will
expose it to XXXX on server
    ports:
      - "YYYY:XXXX"
    environment:
      POSTGRES_USER: yourUsername
      POSTGRES_PASSWORD: yourPassHere

  localstack:
    image: localstack/localstack
    container_name: s3-myunilife
    ports:
      - "YYYY:XXXX"
      - "${PORT_WEB_UI-8090}:${PORT_WEB_UI-8090}"
    environment:
      - SERVICES=s3
      - DEBUG=1
      - PORT_WEB_UI=8090
      - DATA_DIR=/tmp/localstack/data
    volumes: #to prevent data loss on docker restart
      - './.localstack:/tmp/localstack/data'
```

Στην πρώτη γραμμή ορίζεται η έκδοση του docker-compose που θα χρησιμοποιηθεί.

Έπειτα, στο τμήμα με τα services δηλώνονται οι λεπτομέρειες για τα δύο docker containers που θα χρησιμοποιηθούν. Πιο συγκεκριμένα ορίζεται το image που θα χρησιμοποιηθεί για τη δημιουργία το τελικό όνομα του container και τα ports με οποία μπορεί κάποιος να επικοινωνήσει με το service είτε μέσω του docker container είτε απευθείας εξωτερικά.

Τέλος, στο τμήμα environment θέτονται οι απαραίτητες μεταβλητές για την ορθή λειτουργία καθε service. Για την βάση δεδομένων απαιτείται το όνομα και ο κωδικός του χρήστη της βάσης ενώ στο localstack εφόσον καθοριστεί ότι το service για το οποίο θα χρησιμοποιηθεί είναι το s3, ακολουθούν οι παράμετροι για το πραγματικό path στο οποίο θα αποθηκεύονται πραγματικά τα αντικείμενα του s3-bucket προκειμένου να αποτρέψουμε την απώλεια αυτών κατα τον τερματισμό η την επανεκκίνηση του docker container.

5.2.4 NodeJS

Χρησιμοποιήθηκε η έκδοση nodeJS 10.16.0 και η έκδοση 6.9.0 για το npm.

Παρακάτω δίνονται χαρακτηριστικά παραδείγματα υλοποίησης για διάφορα ενδεικτικα τμήματα του server του συστήματος.

Υλοποίηση του router για ένα endpoint

```
const courses = require('./courses');
router.use('/:id/courses', courses);
```

και μέσα στο αρχείο courses:

```
const router = express.Router({mergeParams: true});

router.route('/').get(
    authenticate(),
    authorize(),
    validate(blueprint.courses),
    lift(method.courses),
    respond
);

module.exports = router;
```

Αρχικά με την πρώτη γραμμή κώδικα γίνεται required το αρχείο που περιέχει των κώδικα που πρέπει να εκτελεστεί στο συγκεκριμένο end point δηλαδή όταν ένας χρήστης επιθυμεί να δει όλα τα διαθέσιμα μαθήματα της σχολής. Στο αρχείο αυτό έχει γίνει export η μεταβλητή-μέθοδος router. Έπειτα με τη χρήση του router.use ορίζεται πως αυτη η μέθοδος θα εκτελεστεί όταν στον server φτάσει ένα request στο endpoint “/:id/courses”. Στη μέθοδο αυτή ορίζεται η σειρα εκτέλεσης πέντε βασικών middleware. Εφόσον πραγματοποιηθούν έλεγχοι για authentication, authorization και

validation στη συνέχεια εκτελείται το middleware lift το οποίο αναλαμβάνει να εκτελέσει την κατάλληλη μέθοδο για αυτό το request, περνώντας τις απαραίτητες παραμέτρους και μόλις αυτό ολοκληρωθεί προωθεί το αντικείμενο response και οδηγεί τον έλεγχο στο middleware respond το οποίο αναλαμβάνει την κατάλληλη μορφοποίηση της απάντησης προς τον client και την αποστολή.

Υλοποίηση του sequelize μοντέλου για μία οντότητα στη βάση

Δίνεται ένα παράδειγμα ορισμού του μοντέλου για τα έγγραφα των καθηγητών στα μαθήματα

```
const define = (database, types) => {
    // Note.
    const Note = database.define('Note', {

      // The ID of the note.
      id: {
        type: types.INTEGER,
        autoIncrement: true,
        primaryKey: true,
        field: 'id',
      },

      // The external UUID of the note.
      uuid: {
        type: types.UUID,
        field: 'uuid',
      },

      // the title of the note
      title: {
        type: types.STRING(128),
        allowNull: true,
        field: 'title',
      },
```

Αρχικά, ορίζονται μια προς μια όλες οι παράμετροι-ιδιότητες αυτής της μεταβλητής-μοντέλου σε πλήρη αντιστοιχία ένα προς ένα με τις ιδιότητες (τα columns) κάθε εγγραφής (του αντίστοιχου πίνακα) στη βάση. Με τον ορισμό των singular και plural names υπάρχει μεγαλύτερη ευκολία στην αναφορά στο αντικείμενο ενώ με το “sorting criterion” ορίζονται τα values με τα βάση οποία αν το απιθυμήμουμε το sequelize μπορει να ταξινομήσει τα αντικειμενα που επιστραφούν από τη βάση μετά απο ενα request .

```
// the path where the file exist
      path: {
        type: types.STRING(256),
        allowNull: false,
        field: 'path',
      },
    }, {
      name: {
        singular: 'note',
        plural: 'notes',
      },
      tableName: 'note',
    });

    // Define Sorting criterions.
    Note.SortingCriterion = [
      'title',
      'createdAt',
      'updatedAt',
    ];

    // Define associations (for autoassociations on sql queries).
    Note.associate = (models) => {
      models.Note.belongsTo(models.User, {foreignKey: 'userId',
targetKey: 'id'});
      models.Note.belongsTo(models.Course, {foreignKey: 'courseId',
targetKey: 'id'});
    };

    // eslint-disable-next-line func-names
    Note.prototype.toJSON = function() {
      const values = Object.assign({ }, this.get());
      delete values.id;
      delete values.userId;
      delete values.courseId;
      return values;
    };

    // eslint-disable-next-line func-names
    Note.prototype.toReductedObject = function(input) {
      // Add new entries to object.
      const values = Object.assign(this.get(), input);
      delete values.id;
      delete values.userId;
      delete values.courseId;
      return values;
    };

    return Note;
  };

  module.exports = define;
```

Έπειτα, ορίζονται τα association αυτού του μοντέλου με τα υπόλλοιπα μοντέλα της βάσης τα οποία θα χρησιμοποιηθούν από το sequelize για τα απαραίτητα joins στη βάση δεδομένων όταν στα request χρησιμοποιείται “include”. Τέλος μορφοποιείται κατάλληλα το αντικείμενο που θα επιστραφεί στον client (front-end app) ώστε για παράδειγμα να παραληφθούν ευαίσθητα δεδομένα οπως ids τα οποία απαιτούνται για την εύρυθμη λειτουργία αλλά δεν θέλουμε να διαρρεύσουν εκτός του server του συστήματος.

Packages-Modules

- **Express**

Είναι ένα απλό και ευέλικτο framework εφαρμογών NodeJs που παρέχει ένα ισχυρό σύνολο λειτουργιών για εφαρμογές κινητών και διακομιστή. Παρέχει πληθώρα μεθόδων, που αξιοποιούν το πρωτόκολλο http(s) και διευκολύνει την ανάπτυξη των εφαρμογών.

- **Winston**

Χρησιμοποιήθηκε για καταγραφή πληροφοριών, errors, και μηνυμάτων αποσφαλμάτωσης.

- **Body-parser**

Για ανάγνωση και ανάλυση του σώματος των requests. Οταν στέλνονται δεδομένα σε μορφή json με τη χρήση Post request τότε τα δεδομένα μπαίνουν στο request.body. Οταν τα δεδομένα έρχονται με τη μορφή παραμέτρων στο url τότε αποθηκεύονται στο request.query.

- **Cookie-parser**

Τοποθετεί όλα τα cookies που στέλνει ο client στο request.body.

- **Api-doc**

Για τη δημιουργία και αυτόματη μορφοποίηση Documentation.

- **Eslint**

Έλεγχος της μορφής και δομής του κώδικα.

- **Moment** και **moment-timezone**

Για την σωστη μορφοποίηση ημερομηνιών.

- **Uuid**

Για τη δημιουργία καθολικών μοναδικων κωδικών που χρησιμοποιούνται για μορφοποίηση για τις οντότητες της βάσης δεδομένων .

- **Blueprint**

Για τη δημιουργία κλάσεων.

- **Nodemon**

Χρησιμοποιείται για τη παρακολουθηση των αλλαγων στον κωδικα και την αυτόματη επανεκκίνηση του Node server (προσφέρει συμπεριφορά hot-code reloading). Αξιοποιήθηκε για να διευκολύνει τη διαδικασία ανάπτυξης

- **Bluebird**

Για τη χρηση promises.

- **db-migrate / db-migrate-pg**

Για τη δημιουργία και διαχείριση των migrations στη βάση δεδομένων.

- **Jsonwebtoken**

Για τη δημιουργία και τη διαχείριση των tokens.

- **Joi**

Χρησιμοποιείται για τον έλεγχο της μορφής των δεδομένων στα εισερχόμενα requests.

- **Sequelize**

Χρησιμοποιήθηκε για την απεικόνιση-συσχέτιση των οντοτήτων στη βάση με τα nodejs μοντέλα και την ευκολότερη και ασφαλέστερη επικοινωνία με τη βαση.

- **Aws-sdk**

Απαιτείται για την επικοινωνία με το s3 bucket.

- **Bcrypt**

Προσφέρει συναρτήσεις κρυπτογράφησης που χρησιμοποιούνται για τη διαχείριση των passwords.

- **Nodemailer**

Για αποστολή smtp-μηνυμάτων ηλεκτρονικού ταχυδρομείου μέσω της εφαρμογής.

- **Simple-hashtable**

Για τη δημιουργία και διαχείριση πινάκων κατακερματισμού.

5.2.4.1 Ροή εκτέλεσης

Όταν στο NodeJs server φτάσει ένα αίτημα, ελέγχεται αρχικά απο τον router αν το endpoint που ζητήθηκε υπάρχει και αντιστοιχεί στον συγκεκριμένο τύπο του αιτήματος (get, put, post, delete κ.τ.λ). Αν το endpoint που ζητήθηκε είναι έγκυρο, τότε πριν την εκτέλεση της αντίστοιχης μεθόδου, χρησιμοποιούνται τρία middlewares.

- **Authentication**

Το πρώτο κατά σειρά εκτέλεσης middleware αναλαμβάνει τον έλεγχο της αυθεντικότητας του χρήστη. Συγκεκριμένα, κάθε αίτημα πρέπει να συνοδεύεται από έναν κωδικό (bearer token) ο οποίος αντιστοιχεί σε συγκεκριμένο χρήστη της εφαρμογής. Σε περίπτωση που το το αίτημα δεν συνοδεύεται από έναν κωδικό σε αποδεκτή μορφή ή ο κωδικός δεν αντιστοιχεί σε κάποιον χρήστη τότε ο αιτών θα δεχτεί μια απάντηση με κωδικο 401 και μήνυμα τύπου (UNKNOWN ACCESSOR) το οποίο υποδεικνύει πως ο χρήστης είναι άγνωστος και δεν έχει δικαίωμα να στείλει οποιοδήποτε αίτημα στο backend της εφαρμογής.

- **Authorization**

Στη συνέχεια εφόσον εκτελεστούν οι έλεγχοι για της αυθεντικοποίηση του χρήστη εκτελείται το δεύτερο middleware που ελέγχει αν ο χρήστης είναι εξουσιοδοτημένος για να εκτελέσει την ενέργεια που επιθυμεί. Για παράδειγμα, πολλοι χρήστες του myUniLife (καθηγητές και διαχειριστές) είναι εξουσιοδοτημένοι να τροποποιήσουν τις λεπτομέρειες ενός μαθήματος και όλοι οι χρήστες μπορούν να δουν τα διαθέσιμα μαθήματα, αλλά και στις δύο περιπτώσεις θα πρέπει να είναι μέλη του αντίστοιχου τμήματος και πανεπιστημίου. Επιπλέον, το αίτημα αφορά για παράδειγμα την επεξεργασία του προφίλ του χρήστη, αλλά ο χρήστης ο οποίος προσπαθεί να εκτελέσει την ενέργεια δεν είναι ο κάτοχος του προφίλ, τότε το αίτημά του θα απορριφθεί και θα λάβει μια απάντηση με κωδικό 403 και μήνυμα ACCESS_DENIED.

- **Validation**

Το τελευταίο middleware ελέγχου που θα εκτελεστεί αναλαμβάνει τον έλεγχο των παραμέτρων του αιτήματος. Πιο συγκεκριμένα, χρησιμοποιείται το npm module JOI validator και πραγματοποιούνται έλεγχοι, τόσο στις παραμέτρους του url όσο και στο σώμα του request, το οποίο στην πλειοψηφία των περιπτώσεων είναι σε μορφή json. Γίνεται έλεγχος για τον αν όλες οι προαπαιτούμενες παράμετροι έχουν συμπληρωθεί και αποσταλεί, καθώς και ο τύπος και το εύρος τιμών τους. Για παράδειγμα, σε τμήματα με βαθμολογία στην κλίμακα 1-10, ένα αίτημα που αφορά την καταχώρηση βαθμολογίας και το πεδίο του βαθμού απουσιάζει, ή είναι τύπου συμβολοσειράς και οχι ακεραίου δεν θα γίνει αποδεκτό και ο server θα επιστρέψει error 400 MISSING_GRADE ή INVALID_GRADE. Επίσης, αν το

εξάμηνο αναπαρίσταται με έναν ακέραιο, τοτε πραγματοποιείται έλεγχος και στο εύρος τιμών ώστε μια τιμη -2 να μην μπορει να είναι αποδεκτή.

5.2.4.2 Migrations

Για τον ευκολότερο χειρισμό των migrations στην βάση δεδομένων χρησιμοποιήθηκαν τα node packages dm-migrate και db-migrate-pg. Αυτά επιτρέπουν την οργάνωση των εκτελέσιμων αρχείων που εκτελούν τα πραγματικά migrations και κάνουν tracking αυτών σε ένα φάκελο με το όνομα migrations μέσα στον οποίο υπάρχει και ο φάκελος sqls όπου ορίζονται σε psql syntax οι εντολές που θα εκτελεστούν στη βάση. Κάθε migration διαθέτει δύο βασικές λειτουργίες: up και down. Με την πρώτη εκτελείται το migration αν δεν έχει ήδη εφαρμοστεί στη βάση ενώ με το down προσφέρεται η δυνατότητα για rollback.

Συνεπώς, ο βασικός χειρισμός των migration μπορεί να περιοριστεί στις ακόλουθες τρεις βασικές εντολές cli:

1. **db-migrate create <migration_name> --sql-file**

Δημιουργία νέου migration.

2. **db-migrate up**

Εκτέλεση τυχόν υπαρχόντων migrations που δεν έχουν εκτελεστεί.

3. **db-migrate down [-c x]**

Rollback του τελευταίου (η των x τελευταίων) migrations.

5.2.5 Deployment στην υπηρεσία Okeanos

Στο πλαίσιο της διπλωματικής εργασίας η υπηρεσία του okeanos αξιοποιήθηκε ως περιβάλλον ανάπτυξης για το deploy του server, προκειμένου να μπορεί μια κινητή συσκευή iOS (iphone) να επικοινωνήσει με το backend μέσω ενός restful API.

Supervisor configuration

Ο node server ρυθμίστηκε ως supervisorctl service ώστε να λειτουργεί ως υπηρεσια του μηχανήματος και να επιτρέπεται η διακοπή, εκκίνηση και επανεκκίνηση του από το σύστημα.

Για να πραγματοποιηθεί αυτό δημιουργήθηκε το ακόλουθο αρχείο στο φάκελο */etc/supervisor/conf.d*

```
command=npm run start:dev
autostart=true
autorestart=true
stderr_logfile=/var/log/myUniLife-backend.err.log
stdout_logfile=/var/log/myUniLife-backend.out.log
```

Η πρώτη γραμμή ορίζει την εντολή που θα εκτελεστεί. Η συγκεκριμένη εντολή (npm run start:dev) είναι ορισμένη μέσα στο package.json και εκτελεί το εξής:

"*nodemon --inspect --exec 'node -r dotenv/config' server.js*".

Με τις εντολές αυτές ενεργοποιείται το περιβάλλον με την κατάλληλη έκδοση node και npm και τις αντίστοιχες παραμέτρους και στη συνέχεια εκτελείται ο κώδικας στο αρχείο server.js όπου και αρχικοποιείται και εκτελείται ο listener της εφαρμογής ο οποίος θα περιμένει να δεχτεί εισερχόμενες συνδέσεις. Η δεύτερη και τρίτη γραμμή χρησιμοποιείται ώστε η υπηρεσία να μπορεί να ξεκινήσει αυτόματα κατα την εκκίνηση του εικονικού μηχανήματος, αλλά και να γίνει αυτόματη επανεκκίνηση σε περίπτωση που για οποιοδήποτε λόγο η υπηρεσια τερματίσει.
Στην τέταρτη γραμμή ορίζεται το αρχείο στο οποίο θα καταγράφονται τα errors του server ενώ αντίστοιχα στην πέμπτη γραμμή ορίζεται το αρχείο για την καταγραφή όλων των υπόλοιπων outputs της εφαρμογής που δεν αποτελούν error.(π.χ. μηνύματα αποσφαλμάτωσης κ.τ.λ).

Apache configuration

Επιπλέον προκειμένου να δρομολογείται η κίνηση που φτάνει στο μηχάνημα του okeanos και αφορά την υπηρεσία του myUniLife προς το εικονικό host και port στο οποίο “ακούει” ο myUniLife server χρησιμοποιήθηκε ο apache server ως proxy. Για να επιτευχθεί αυτό δημιουργήθηκε το ακόλουθο αρχείο στο φάκελο /etc/apache2/ sites-available.

```
<VirtualHost okeanos-ip:80 *:80>
  ServerAdmin myUniLife@admins.com
  ServerName myUniLife.local
  # Ensure that encoded slashes are not decoded but left in their
encoded state.
  # http://doc.gitlab.com/ce/api/projects.html#get-single-project
  # AllowEncodedSlashes NoDecode

  ServerSignature Off
  ProxyPreserveHost On
  ProxyRequests Off

  <Proxy *>
     Order deny,allow
     Allow from all
  </Proxy>

    ProxyPass / http://myUniLife.local:4242/
    ProxyPassReverse / http://myUniLife.local:4242/
</VirtualHost>

<VirtualHost *:80>
  ServerName myunilife
</VirtualHost>
```

Στο παραπάνω αρχείο ορίζονται δύο εικονικοί διακομιστές. Ο πρώτος αναλαμβάνει να προωθήσει τα εισερχόμενα αιτήματα που καταφθάνουν στο μηχάνημα του okeanos στη θύρα 80 στην εσωτερική διεύθυνση http://myUniLife.local:4242 οπου φιλοξενείται ο server της υπηρεσίας myUniLife.
Ο δεύτερος εικονικός διακομιστής υπάρχει ώστε να επιτυγχάνεται επικοινωνία με το τον τοπικό διακομιστή ο οποίος λειτουργεί ως s3-bucket.

Τέλος στο αρχείο /etc/hosts προστέθηκαν οι ακόλουθες γραμμές:

```
127.0.0.2       myUniLife.local
127.0.0.2       myunilife
```

Deploy script

Για την αυτοματοποίηση της διαδικασίας deploy του server γράφτηκε και έγινε χρήση του ακόλουθου script για μηχανήματα με λειτουργικό σύστημα βασισμένο σε Unix.

```
#! /bin/bash
#title          :deploy.sh
#description    :Script to deploy myUniLife (development) to okeanos .
#author     :Nikos Koukoutis
#date           :2019-10-11
#version        :0.1

#output colors
RED='\033[0;31m' # RED color
NC='\033[0m' # No Color

cd /home/myunilife/myUniLife-backend

# just to be sure nothing has been changed that would prevent from git
pull
git checkout -f
git checkout develop
#echo -e "${RED}Make sure that github is reachable from okeanos...${NC}"
#echo "nameserver 8.8.8.8" | sudo tee /etc/resolv.conf > /dev/null

echo -e "\n${RED}Pull the latest develop version from github${NC}"
git pull origin develop

. env/bin/activate

echo -e "\n${RED}Install new npm packages...${NC}"
npm install

echo -e "\n${RED}Execute latest migrations...${NC}"
npx db-migrate up
docker-compose up -d

echo -e "\n${RED}Restart the service...${NC}"
sudo supervisorctl restart myUniLife-backend

echo -e "\n${RED}Seed the database [s3]...${NC}"
npm run seed

echo -e "\n${RED}Seed the database [professors]...${NC}"
npm run seed-professors

echo -e "\n${RED}Seed the database [flows]...${NC}"
npm run seed-flows

echo -e "\n${RED}Seed the database [courses]...${NC}"
npm run seed-courses
#Until to solve the problem with watches for node kill all processes
#myUniLife-backend service will automatically restart
sudo pkill -f node
```

Αρχικά γίνεται download της τελευταίας έκδοσης του κώδικα που φιλοξενείται στο github.

Εγκαθίστανται τα απαιτούμενα πακέτα (τα οποία βρίσκονται ορισμένα μέσα στο package.json) με τη χρήση του npm install. Έπειτα, εκτελούνται τυχόν υπάρχοντα migrations για τη βάση δεδομένων και εκτελείται η εντολή docker-compose up -d. Με αυτή την εντολή και τη χρήση του αρχείου docker-compose.yml θα γίνει εκκίνηση της βάσης δεδομένων και του s3-bucket localstack. Στη συνέχεια, επανεκκινείται η υπηρεσία του node και τρέχουν τρία αρχεία κώδικα που θα γεμίσουν τη βάση με καθηγητές, μαθήματα και ροές μαθημάτων.

5.2.6 Ασφάλεια

Όπως σε κάθε εφαρμογή σήμερα, έτσι και στο myUniLife ιδιαίτερης προσοχής χρήζει η ασφάλεια των δεδομένων των χρηστών. Τα ακαδημαϊκά δεδομένα, αλλά και λοιπά προσωπικά στοιχεία που μπορεί να αποθηκεύονται, είναι πιθανό να αποτελέσουν στόχο κακόβουλων χρηστών, κάτι το οποίο θα έβλαπτε τόσο τους χρήστες όσο και την αξιοπιστία της εφαρμογής. Αρχικά, για κάθε request προς το backend του συστήματος απαιτείται ένα bearer token προκείμενου αυτό να χρησιμοποιηθεί για το authentication του χρήστη από το αντίστοιχο middleware, πριν ακόμα εκτελεστεί οποιαδήποτε μέθοδος που έχει πρόσβαση στα δεδομένα του χρήστη. Ακόμη, μέσω ενός endpoint στο nodeJS server του συστήματος παρέχεται η δυνατότητα στο χρήστη να ακυρώσει όλα τα υπάρχοντα tokens, πραγματοποιώντας αποσύνδεση από όλες τις συσκευές που μπορεί να είναι συνδεδεμένες με το λογαριασμό του, οποιαδήποτε στιγμή και απαιτώντας ο χρήστης να συνδεθεί ξανά εισάγοντας το username και το password του, ώστε να μπορεί να εκτελέσει οποιοδήποτε request. Τόσο η υπηρεσία s3, που χρησιμοποιήθηκε για την αποθήκευση των αρχείων, όσο και η posrgreSQL που χρησιμοποιήθηκε για τα στοιχεία των χρηστών, έχουν σχεδιαστεί με γνώμονα την ασφάλεια των δεδομένων, ενώ τα password των χρηστών αποθηκεύονται κρυπτογραφημένα μέσω μη συμμετρικής συνάρτησης (μονής κατεύθυνσης), ώστε να μην μπορούν να χρησιμοποιηθούν, και καθίστανται άχρηστα στα χέρια κάποιου που τα έχει υποκλέψει. Η παράμετρος που χρησιμοποιείται ως salt για τη συνάρτηση κρυπτογράφησης είναι μοναδική για κάθε χρήστη, με αποτέλεσμα τα δεδομένα να είναι ασφαλή ακόμα και αν αποκαλυφθεί η συνάρτηση κρυπτογράφησης και το κρυπτογραφημένο password. Ωστόσο, επιπλέον μέτρα ασφαλείας μπορούν να ληφθούν προκειμένου να προστατευθούν τα δεδομένα από κάποιον επιτιθέμενο που αποκτά απευθείας πρόσβαση στη βάση και όχι μέσω της εφαρμογής. Μία από τις πιο έμπιστες λύσεις είναι η κρυπτογράφηση όλων των δεδομένων στη βάση, καθιστώντας την με τον τρόπο αυτό μη αναγνώσιμη, χωρίς την αντίστοιχη συνάρτηση και τις παραμέτρους της. Τέλος, επιπλέον τακτικές που θα μπορούσαν να συμβάλουν στην ασφάλεια του συστήματος είναι:

- Ο διαχωρισμός της βάσης από τον server. Με τον τρόπο αυτό αν κάποιος παραβιάσει τον admin account του μηχανήματος όπου φιλοξενείται το ένα, δεν αποκτά αυτόματα πρόσβαση και στο δεύτερο.
- Συχνό update των πακέτων και των βιβλιοθηκών που χρησιμοποιούνται, αλλά και του λειτουργικού συστήματος του μηχανήματος που φιλοξενεί την εφαρμογή, ώστε να καλύπτονται πιθανά κενά ασφαλείας.
- Περιοδική αλλαγή της συνάρτησης κρυπτογράφησης και των παραμέτρων της, ώστε να μειώσει η πιθανότητα επιθέσεων τύπου brute force, αλλά και να αντιμετωπιστούν καταστάσεις όπου ήδη έχει πραγματοποιηθεί η διαρροή σχετικών πληροφοριών.

5.3 Παραδείγματα χρήσης

5.3.1 Διαχείριση μαθημάτων

Στην ενότητα αυτή, θα περιγραφεί και θα απεικονιστεί με στιγμιότυπα οθονών, το σενάριο στο οποίο, η γραμματεία, που έχει αναλάβει τη διαχείριση των μαθημάτων ενός τμήματος, χρησιμοποιεί την εφαρμογή myUniLife, προκειμένου να προσθέσει ένα καινούργιο μάθημα στα διδασκόμενα της σχολής. Στη συνέχεια, ο καθηγητής στον οποίο έχει ανατεθεί το μάθημα, από τη γραμματεία, θα ανανεώσει την περιγραφή του μαθήματος, ώστε να ενημερώσει τους φοιτητές για το περιεχόμενο του. Τέλος, ο φοιτητής μέσω της καρτέλας των μαθημάτων, επιλέγει να ακολουθήσει το καινούργιο μάθημα και να δει πληροφορίες σχετικές με αυτό.

Αρχικά, και οι τρεις χρήστες θα πρέπει να συνδεθούν στην εφαρμογή, χρησιμοποιώντας το username και το password, που αντιστοιχούν στο λογαριασμό τους. Έπειτα, η γραμματεία μέσω της καρτέλας "Μαθήματα" χρησιμοποιεί την επιλογή "Νέο Μάθημα +". Αμέσως μετά, θα οδηγηθεί στην φόρμα δημιουργίας νέου μαθήματος, όπου και θα συμπληρώσει τα εξής στοιχεία:

- Κωδικό μαθήματος, όπως αυτός αναγράφεται στον οδηγό σπουδών της σχολής
- Τίτλος μαθήματος
- Εξάμηνο, κατά το οποίο διδάσκεται το μάθημα
- Κατηγορία
- Κυριότητα
- Αν ανήκει σε κατεύθυνση ροής ή στον κορμό διδασκαλίας
- Τοποθεσία διεξαγωγής της διάλεξης

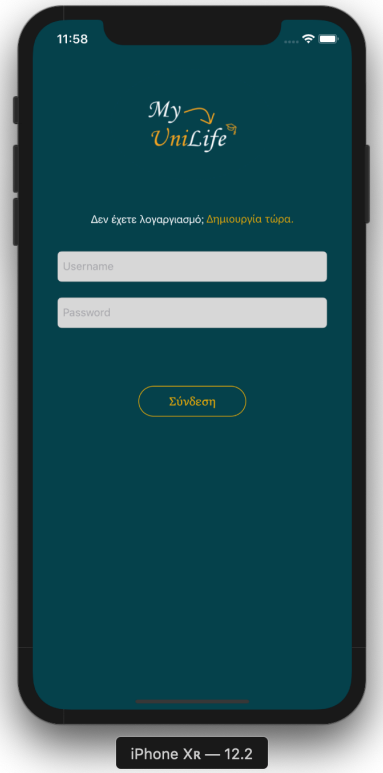
Στη συνέχεια, πατώντας το κουμπί "Προσθήκη Καθηγητών", θα της εμφανιστούν όλοι οι διαθέσιμοι καθηγητές της σχολής, από όπου και θα επιλέξει τον διδάσκοντα του μαθήματος. Με τη χρήση της επιλογής "Προσθήκη μέρας" μπορεί να δηλώσει τις ημέρες και ώρες διεξαγωγής του μαθήματος, σε εβδομαδιαία βάση, και πατώντας το κουμπί "Αποθήκευση" να κατοχυρώσει τις επιλογές της. (**Φάση 1**)

Με τη σειρά του ο καθηγητής θα μπορεί πλέον, από την καρτέλα "Μαθήματα" να δει το νέο μάθημα που του ανατέθηκε. Επιλέγοντας το, θα εμφανιστούν τα γενικά στοιχεία αυτού, και έπειτα, του παρέχεται η δυνατότητα να ανανεώσει την περιγραφή, πατώντας το κουμπί "Επεξεργασία" καθώς και το κουμπί "Αποθήκευση" για να την αποθηκεύσει. (**Φάση 2**)

Τέλος, ο φοιτητής που ενδιαφέρεται για το συγκεκριμένο μάθημα, περιηγούμενος στη καρτέλα "Μαθήματα" και επιλέγοντας το κουμπί "Νέο Μάθημα" μπορεί να δει σε μια οθόνη όλα τα διαθέσιμα μαθήματα της σχολής προς εγγραφή και

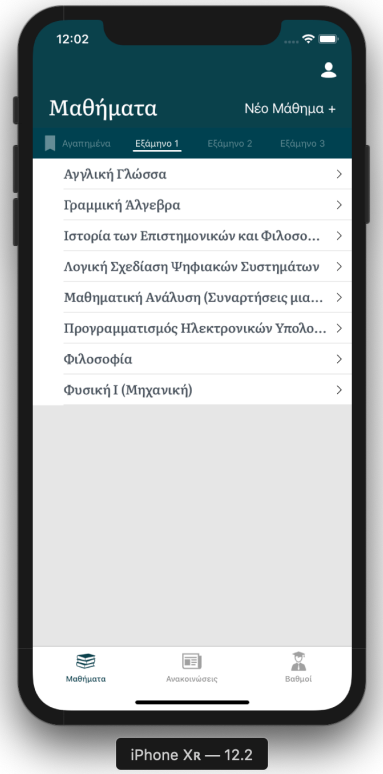
απεγγραφή. Αφού, διαλέξει το νέο μάθημα, και πατήσει το κουμπί “Τέλος”, οι επιλογές του αποθηκεύονται και το νέο μάθημα προστίθεται στα μαθήματα του. (**Φάση 3**)

Εκτέλεση σεναρίου

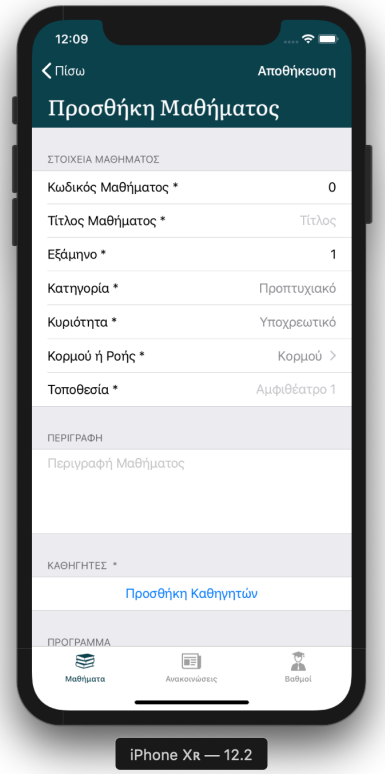


Οθόνη σύνδεσης όλων των χρηστών

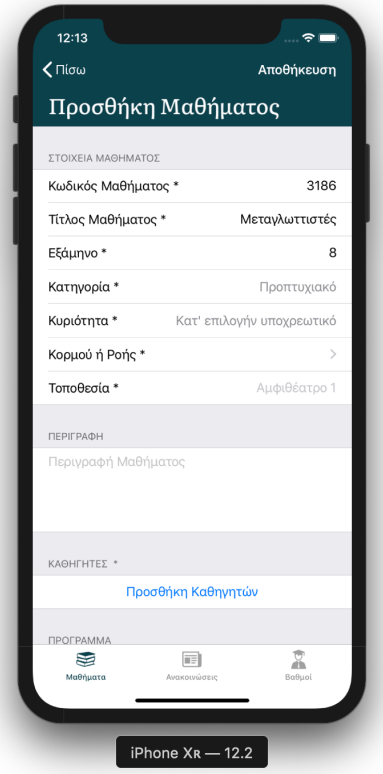
Φάση 1 (Γραμματεία)



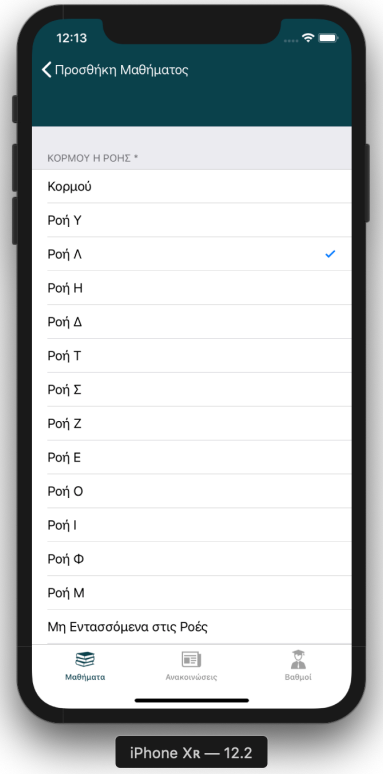
Καρτέλα μαθημάτων



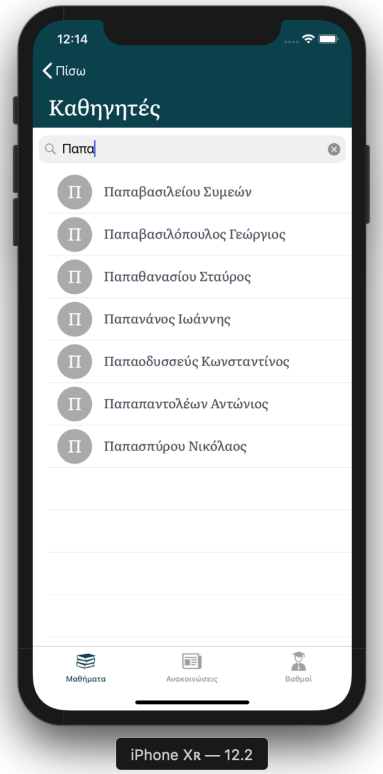
Φόρμα προσθήκης μαθήματος



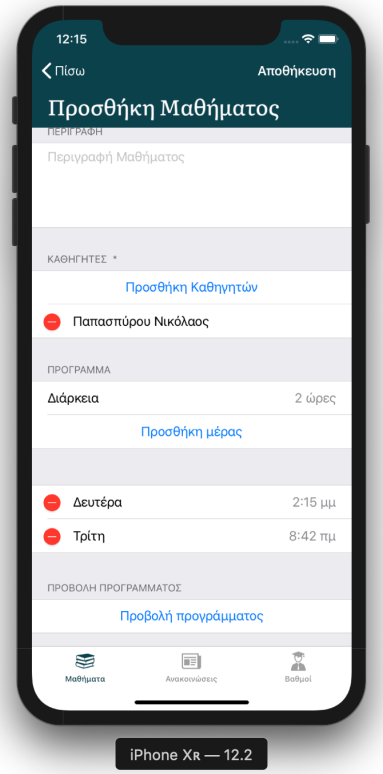
Συμπλήρωση φόρμας



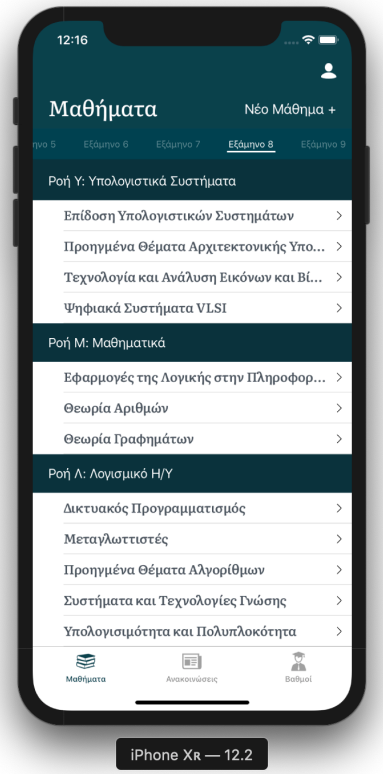
Επιλογή ροής



Επιλογή καθηγητή

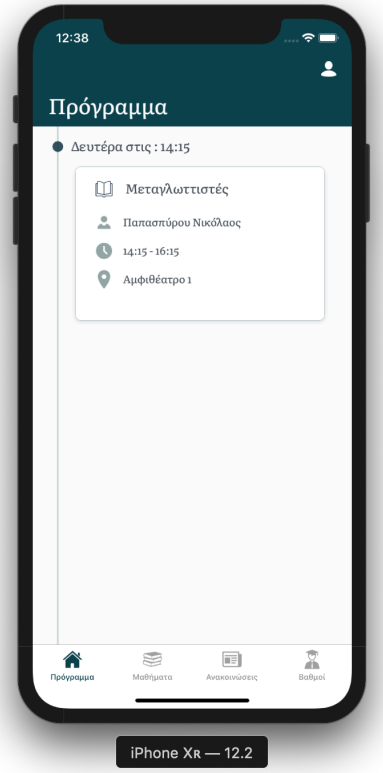


Προσθήκη προγράμματος

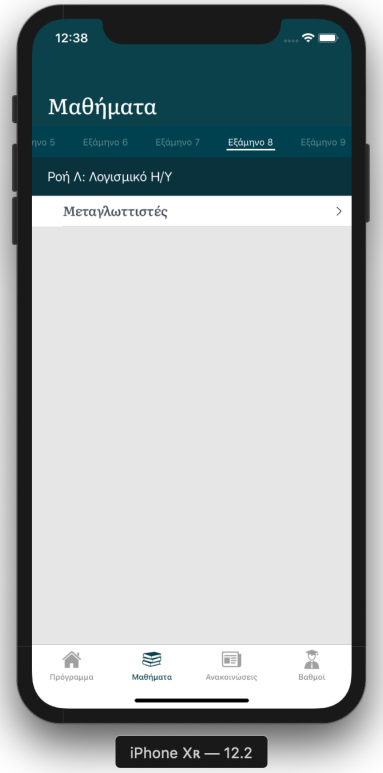


Προβολή νέου μαθήματος

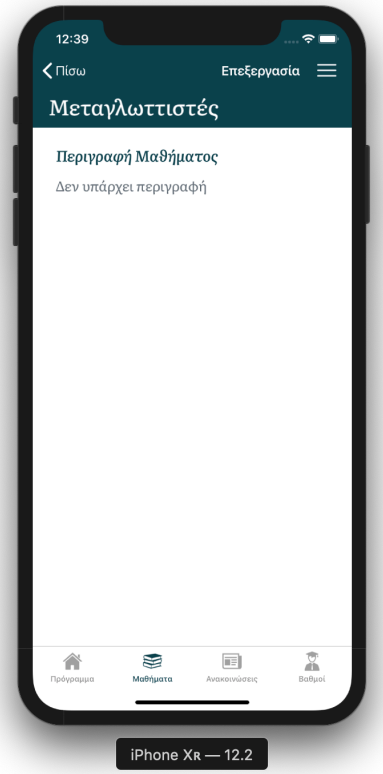
Φάση 2 (Καθηγητής)



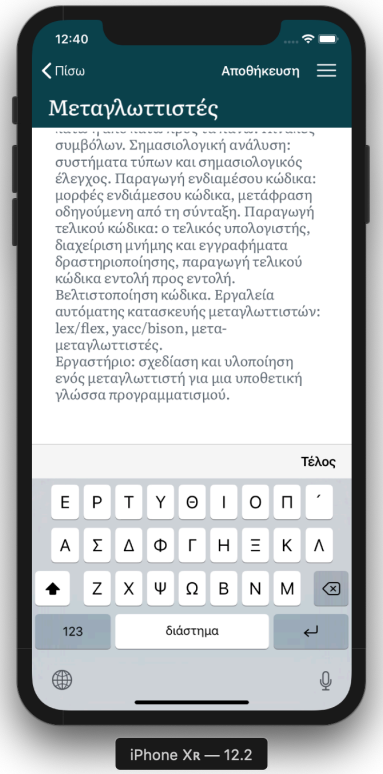
Αρχική οθόνη



Οθόνη μαθημάτων

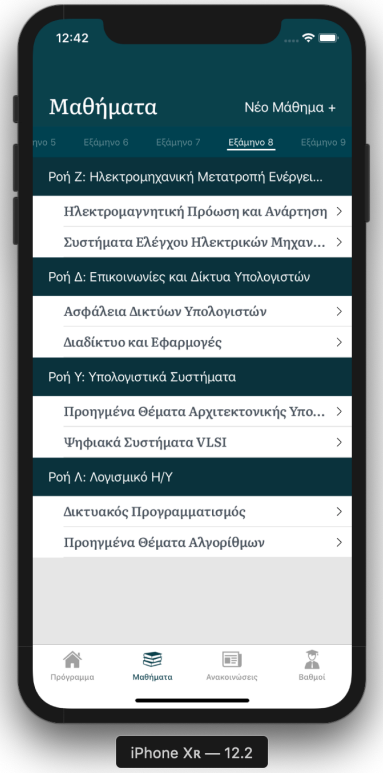


Στοιχεία μαθήματος

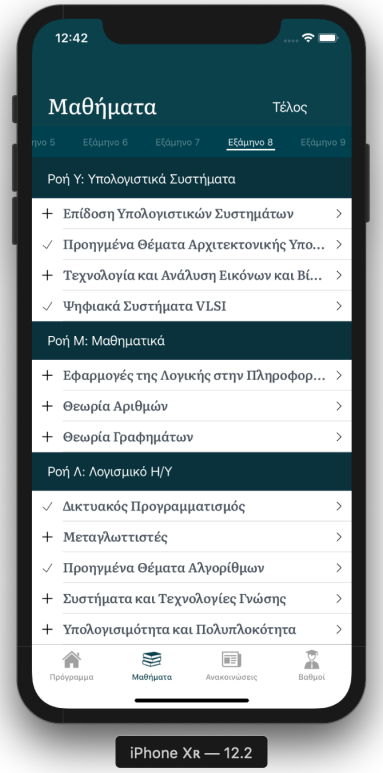


Επεξεργασία μαθήματος

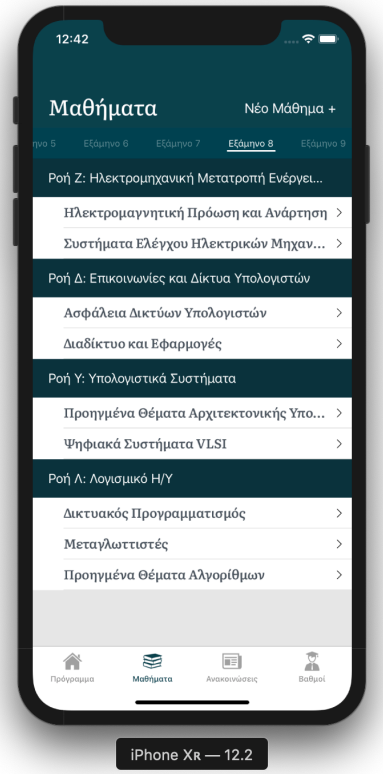
Φάση 3 (Φοιτητής)



Καρτέλα μαθημάτων



Προσθήκη νέου μαθήματος



Καρτέλα μαθημάτων με το νέο μάθημα

Στοιχεία μαθήματος

5.3.2 Διαχείριση εργασιών

Στην ενότητα αυτή, παρουσιάζεται το σενάριο διαχείρισης εργασιών από τον καθηγητή και τους φοιτητές ενός μαθήματος.

Ο καθηγητής εφόσον συνδεθεί στην εφαρμογή, πλοηγείται στην καρτέλα "Μαθήματα", όπου θα δει όλα τα μαθήματα στα οποία είναι διδάσκων, ομαδοποιημένα σύμφωνα με το εξάμηνο διδασκαλίας, και επιλέγει το μάθημα το οποίο επιθυμεί να επεξεργαστεί. Έπειτα, αφού επιλέξει το στοιχείο "Εργασίες" από το μενού, οδηγείται στην οθόνη όπου προβάλλονται οι εργασίες που έχουν αναρτηθεί, τόσο από τον ίδιο (εκφωνήσεις), όσο και από τους φοιτητές του μαθήματος. Χρησιμοποιώντας την επιλογή "Νέα Εργασία +", ανοίγει ο επιλογέας αρχείων του κινητού, μέσω του οποίου επιλέγει το αρχείο προς ανάρτηση. (**Φάση 1**)

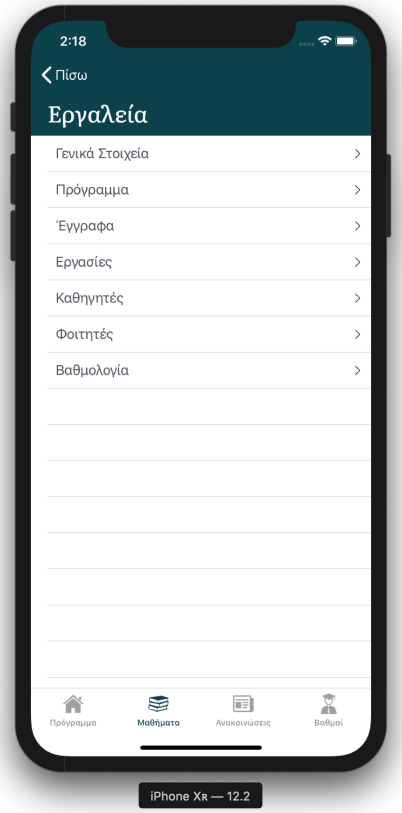
Ο φοιτητής, από το μενού του μαθήματος και από την επιλογή "Εργασίες" κατεβάζει όποια εκφώνηση επιθυμεί. Όταν κάποια στιγμή θελήσει να υποβάλει τη λύση του σε μια εργασία, επιλέγει το κουμπί "Υποβολή" και επιλέγει το αρχείο προς ανάρτηση. (**Φάση 2**)

Έπειτα, ο καθηγητής κατεβάζει τη λύση του συγκεκριμένου φοιτητή, και υποβάλλει την βαθμολογία πατώντας το κουμπί "Βαθμολόγηση". (**Φάση 3**)

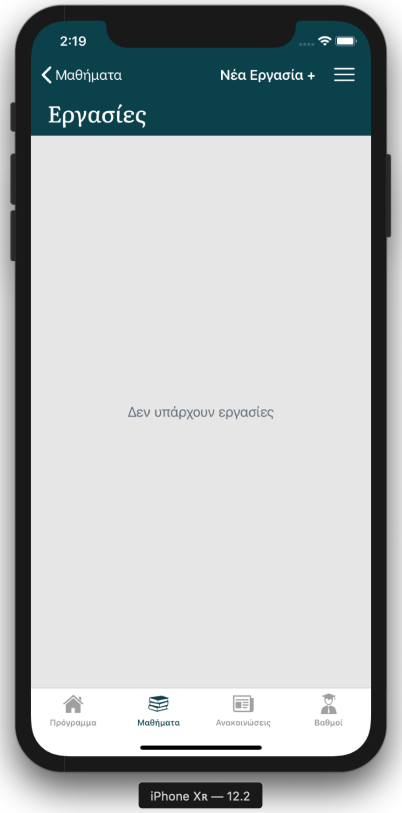
Τέλος, ο φοιτητής βλέπει τη βαθμολογία της εργασίας του. (**Φάση 4**)

Φάση 1 (Καθηγητής)

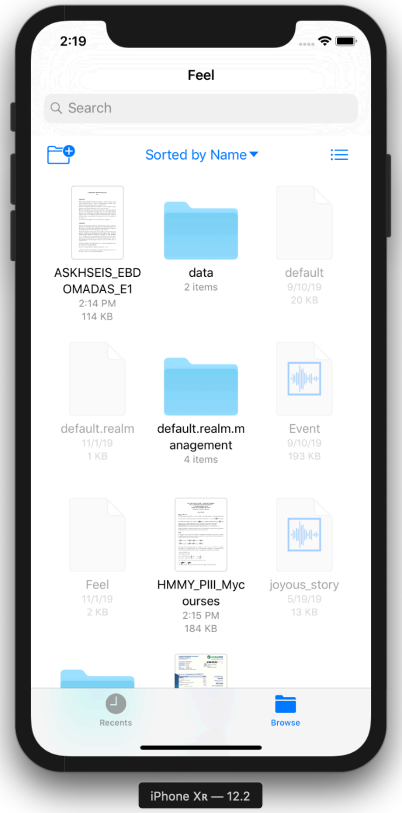
Οθόνη μαθημάτων



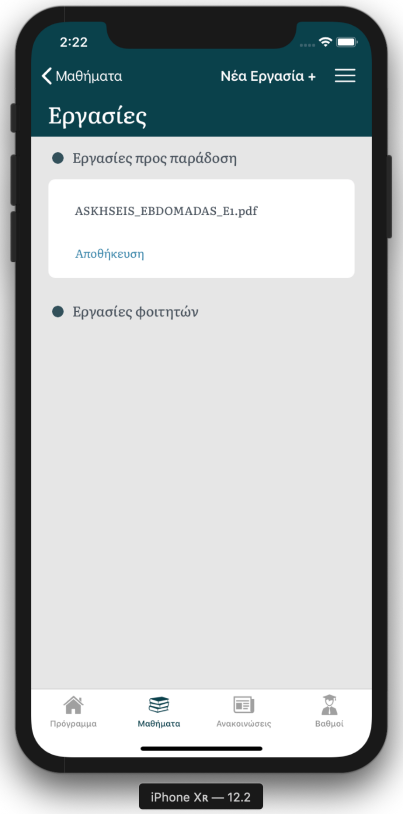
Μενού μαθήματος



Οθόνη εργασιών



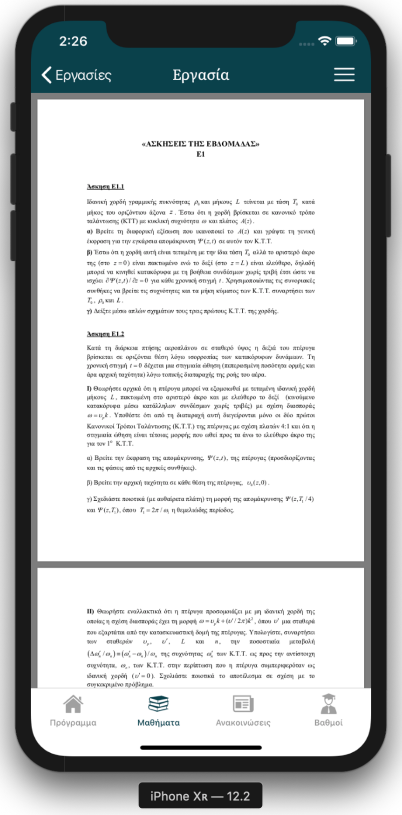
Επιλογή εκφώνησης



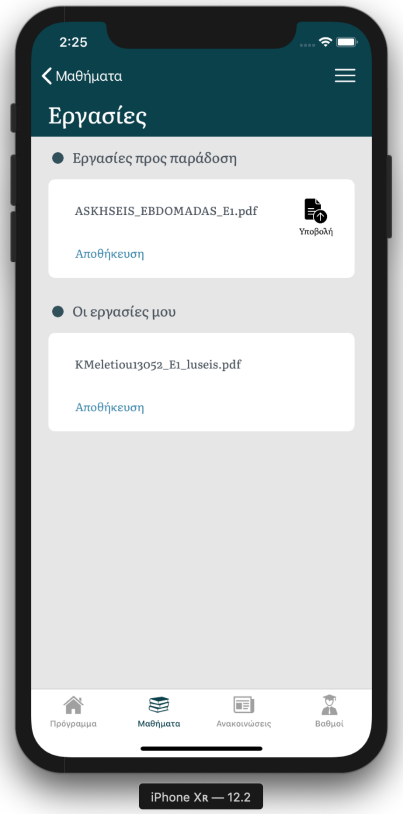
Προβολή αναρτημένης εκφώνησης

Φάση 2 (Φοιτητής)

Οθόνη εργασιών προς παράδοση

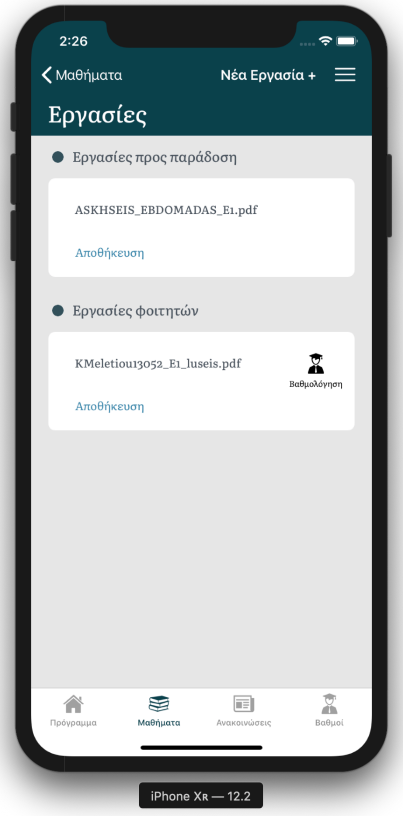


Προβολή εκφώνησης

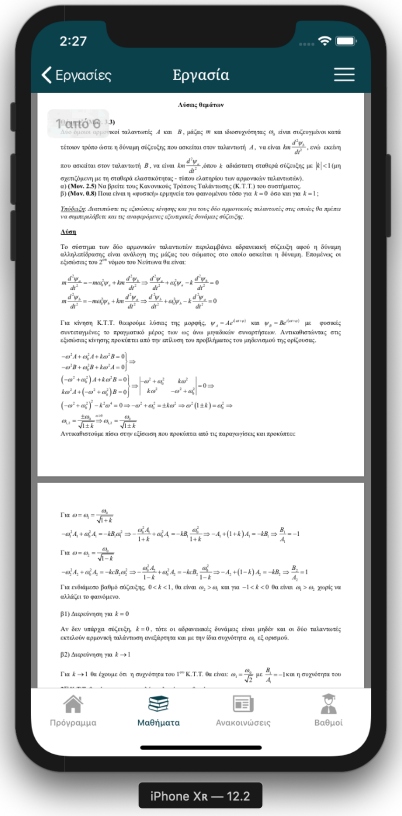


Προβολή αναρτημένων λύσεων

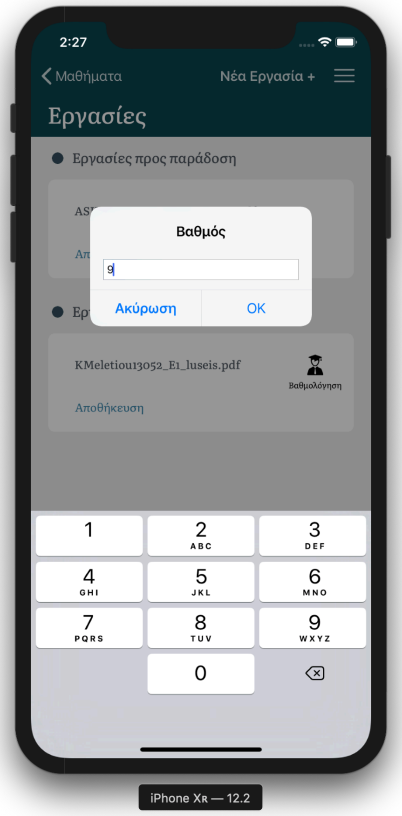
Φάση 3
(Καθηγητής)



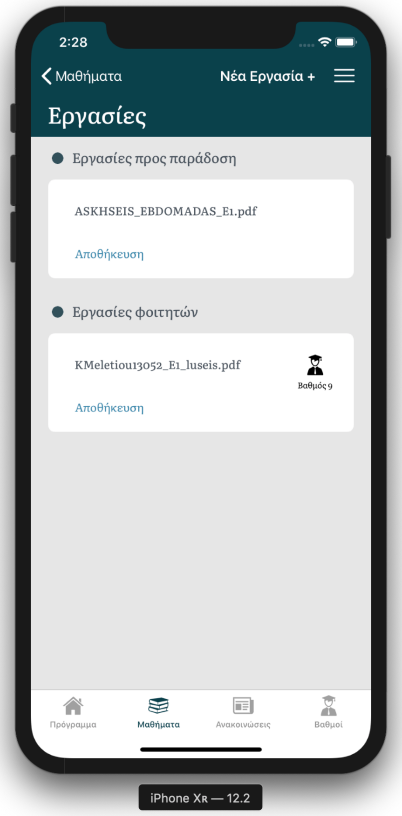
Οθόνη εργασιών προς βαθμολόγηση



Προβολή λύσεων

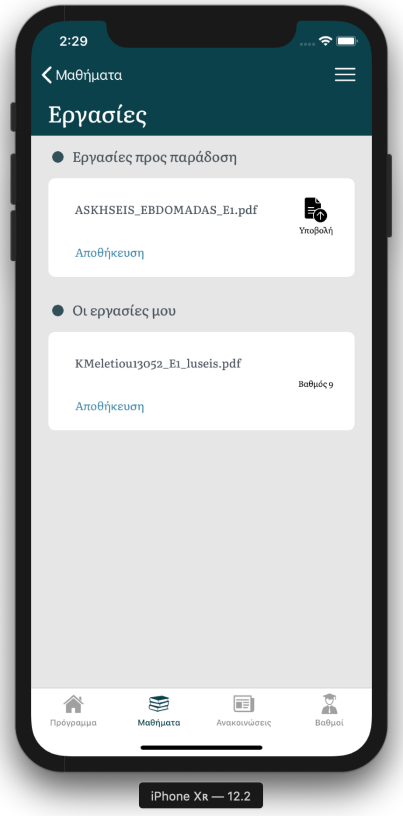


Προσθήκη βαθμού



Προβολή βαθμού

Φάση 4 (Φοιτητής)



Προβολή βαθμού εργασίας

6

Επίλογος

6.1 Συμπεράσματα

Η όλη διαδικασία της διπλωματικής εργασίας, αλλά και της δημιουργίας του συστήματος μας, προσέφερε μια πρωτόγνωρη εμπειρία ενασχόλησης με νέες τεχνολογίες, αλλά και με τη διαδικασία που πρέπει να ακολουθήσει κάποιος αν επιθυμεί να δημιουργήσει μια επιτυχημένη εφαρμογή, που αποσκοπεί στην ευρεία αποδοχή της από τους χρήστες.

Αντιληφθήκαμε ότι με τα σημερινά εργαλεία και τις τεχνολογίες, ουσιαστικά δεν υπάρχουν όρια στο τι μπορεί να δημιουργήσει κάποιος για να λύσει ένα πρόβλημα σε οποιονδήποτε τομέα και αν ανήκει αυτό. Ωστόσο, πολλές είναι επίσης και οι προκλήσεις, με τις οποίες θα έρθει αντιμέτωπος κατά την σχεδίαση και υλοποίηση του. Στη σημερινή ιδιαίτερα εποχή, με τις κυριολεκτικά χιλιάδες εφαρμογές και τους πολλούς τρόπους να αντιμετωπιστεί ένα πρόβλημα, για να πετύχει μία εφαρμογή απαιτείται προσοχή στις λεπτομέρειες σχεδίασης και στις τεχνολογίες που θα χρησιμοποιηθούν. Μία εφαρμογή που απλά επιλύει το πρόβλημα για το οποίο σχεδιάστηκε μπορεί να γνωρίσει μερική ή και καθόλου αποδοχή. Είναι αναγκαίο να δοθεί προσοχή στην εμπειρία που θα έχει ο χρήστης, όταν την χρησιμοποιεί, καθώς πρέπει να του προσφέρεται ένα φιλικό και εύχρηστο περιβάλλον, μέσα από το οποίο θα μπορεί να αξιοποιήσει όλες της δυνατότητες του συστήματος. Η πληροφορία θα πρέπει να παρέχεται στον τελικό χρήστη με τρόπο απλό και κατανοητό, ακόμα και για έναν χρήστη χωρίς εμπειρία,και το ίδιο το σύστημα να πρέπει να τον βοηθά και να τον καθοδηγεί στην εκμάθηση και σωστή χρήση και αξιοποίηση του. Ακόμα, το όλο σύστημα θα πρέπει να είναι αξιόπιστο και ασφαλές και να παρέχει τις υπηρεσίες του άμεσα και δίχως διακυμάνσεις, ανάλογα με τη ζήτηση. Σήμερα, υπάρχουν αρκετές εφαρμογές που χειρίζονται, εν μέρη, τα φοιτητικά στοιχεία ενός φοιτητή και η σχολή μας διαθέτει ιστότοπους από τους οποίους οι φοιτητές και οι καθηγητές μοιράζονται υλικο σχετικό με τα μαθήματα. Μια εφαρμογή πλήρης διαχείρισης όλου του συστήματος, όπως προσπαθήσαμε να δημιουργήσουμε είναι αναγκαίο να μην υστερεί των παραπάνω, να προσφέρει τα ίδια πλεονεκτήματα, να λύνει τα προβλήματα τους και να επεκτείνει τις δυνατότητες αυτών. Επιπλέον, καταλήξαμε στο συμπέρασμα ότι είναι πολύ σημαντικό για τον σημερινό φοιτητή να έχει μία μόνο πλατφόρμα που θα του προσφέρει υπηρεσίες για όλες τις ακαδημαϊκές του υποχρεώσεις, ακόμα και αν κάποιες από τις λειτουργίες θα ήταν πιο συμβατές με μια desktop ή web έκδοση του συστήματος, σίγουρα η mobile

έκδοση είναι επιτακτική λόγω της κυρίαρχης θέσης που κατέχει το έξυπνο κινητό τηλέφωνο στη ζωή και καθημερινότητα όλων των ανθρώπων.

6.2 Μελλοντικές επεκτάσεις

Το σύστημα myUniLife σχεδιάστηκε και υλοποιήθηκε με σκοπό να προσφέρει υποστήριξη στη διαχείριση μαθημάτων για τη σχολή ηλεκτρολόγων μηχανικών και μηχανικών υπολογιστών. Ωστόσο, και στις δύο προαναφερθείσες φάσεις υπήρχε πάντα μέριμνα για τις μελλοντικές προεκτάσεις που μπορεί να υποστηρίξει και να προσφέρει. Η προσθήκη μοντέλων αλλά και εγγραφών στη βάση για "department" και "organization" θέτει τις βάσεις, ώστε η εφαρμογή να μπορεί να αξιοποιηθεί και απο τα υπόλοιπα τμήματα του Εθνικού Μετσόβιου Πολυτεχνείου, αλλά και από διαφορετικά πανεπιστήμια της χώρας. Για τις ανάγκες της διπλωματικής και την υποστήριξη της σχολης ΗΜΜΥ έχουν εισαχθεί στη βάση δεδομένων όλοι οι καθηγητές της σχολής με τα στοιχεία τους. Παράλληλα όμως, το back end της εφαρμογής υποστηρίζει μεσω restful apis την δημιουργία και προσθήκη νέων καθηγητών, έτσι ώστε με τις ανάλογες προσθήκες στο user interface της εφαρμογής, να παρέχεται η δυνατότητα αυτή στις γραμματείες των σχολών που ενδιαφέρονται. Παρά το γεγονός ότι η εφαρμογή σχεδιάστηκε ως standalone σύστημα, το οποίο οι διαχειριστές, καθηγητές και φοιτητές των σχολών θα χρησιμοποιούν κατά αποκλειστικότητα, για λόγους ομαλότερης μετάβασης από τα ήδη υπάρχοντα συστήματα, αποθηκεύεται στη βάση δεδομένων το api κάθε σχολής (σε περίπτωση που κάτι τέτοιο υπάρχει). Έπειτα, κατα την εγγραφή του φοιτητή στο myUniLife χρησιμοποιώντας τα ακαδημαικα του διαπιστευτήρια, ο server της εφαρμογής επικοινωνεί με το αντίστοιχο api και λαμβάνει για τον φοιτητή όλα τα στοιχεία που η σχολή του μπορεί να παρέχει, όπως τα μαθήματα στα οποία είναι ήδη εγγεγραμμένος, καθώς και τους βαθμούς του σε αυτά.

Ακόμη, μπορεί να δοθεί στη γραμματεία, αλλά και στους καθηγητές η δυνατότητα ανάρτησης ανακοινώσεων και σε όλους τους χρήστες η λειτουργία push notifications, ώστε να ενημερώνονται για αυτές. Θα μπορούσε επίσης, να προστεθεί στις υπηρεσίες που παρέχει το σύστημα και μια ενσωματωμένη πλατφόρμα επικοινωνίας μεταξύ καθηγητών και φοιτητών, διευρύνοντας το δίκτυο γνώσεων και ευκαιριών, καταστώντας ευκολότερη την ανταλλαγη όχι μόνο απόψεων, ερωτήσεων και απαντήσεων, αλλά και υλικού. Τέλος, μία πλήρης λύση στο αρχικό πρόβλημα που επιδιώκει να αντιμετωπίσει το σύστημα myUniLife και συνεπώς, μία πιο ολοκληρωμένη εμπειρία χρήστη, ίσως απαιτεί, τόσο την επέκταση τους συστήματος σε συσκευές λογισμικού android, όσο και μία desktop (web) έκδοση αυτού, προσβάσιμη μέσω ενός φυλλομετρητή ιστού.

7

Βιβλιογραφία

[1] Swift, https://www.wikiwand.com/en/Swift_(programming_language)

[2] Apple Developer Documentation https://developer.apple.com/documentation/

[3] Layers of the iOS architecture, https://subscription.packtpub.com/book/application_development/9781849691307/1/ch01lvl1sec12/layers-of-the-ios-architecture)

[4] iOS design patterns, https://medium.com/swlh/ios-design-patterns-a9bd07818129

[5] Coordinator pattern, https://medium.com/@saad.eloulladi/ios-coordinator-pattern-in-swift-39a15aa3b01b

[6] J. Bucanek, Learn iOS 8 App Development, 2nd edition, **ISBN-10:** 1484202090

[7] Xcode, https://www.iclarified.com/70073/apple-releases-xcode-102-with-support-for-swift-5

[8] Node JS, https://el.wikipedia.org/wiki/Nodejs

[9] Node JS, https://tutorialslink.com/Tutorials/Node-Js-Recipes-for-Beginners/24

[10] Sequelize, http://blog.devitpl.com/sequelize/

[11] Sequelize, https://sequelize.org/v5/

[12] PostgreSQL, https://el.wikipedia.org/wiki/PostgreSQL

[13] Υπηρεσία Okeanos, https://okeanos.grnet.gr/home/

[14] Git, https://el.wikipedia.org/wiki/Git

[15] UUID, https://en.wikipedia.org/wiki/Universally_unique_identifier

[16] Amazon s3, https://aws.amazon.com/s3/

[17] Docker, https://en.wikipedia.org/wiki/Docker_(software)

[18] Docker compose, https://docs.docker.com/compose/

[19] Single responsibility principle, https://en.wikipedia.org/wiki/Single_responsibility_principle

[20] Postman, www.getpostman.com

[21] Cocoapods, https://cocoapods.org/

Παράρτημα

1.1 Οδηγίες για τον Server

Οδηγίες για χρήση του NodeJS server τοπικά σε linux (Unix based) υπολογιστή.

CLONE

```
git clone git@github.com:nikoskkt/myUniLife-backend.git
```

CREATE THE VIRTUAL ENVIRONMENT

```
cd myUniLife-backend
nodeenv -n 10.16.0 --prebuilt env
```

ACTIVATE THE VIRTUAL ENVIRONMENT

```
. env/bin/activate
```

DEACTIVATE THE VIRTUAL ENVIRONMENT

```
deactivate_node
```

INSTALL THE DEPENDENCIES

```
npm install
```

CONFIGURE

Create `.env` based on `.env.example`.

```
cp .env.example .env
Edit `.env`.
```

Create docker-compose.yml based on docker-compose.yml.example.

```
cp docker-compose.yml.example docker-compose.yml
Edit `docker-compose.yml`.
```

RUN (DEVELOPMENT)

```
npm run start:dev
```

DOCKER

The following command will use the docker-compose.yml file to create and docker containers for database and s3-bucket (for file storage).

```
docker-compose up -d
```

DATABASE

CREATE DATABASE MIGRATION

```
npx db-migrate create MIGRATION_NAME --sql-file
```

APPLY DATABASE MIGRATIONS

```
npx db-migrate up
```

ROLLBACK THE LATEST MIGRATION

```
npx db-migration down -c 1
```

SEED THE APPLICATION

[s3]

```
npm run seed
```

[flows]

```
npm run seed-flows
```

[professors]

```
npm run seed-professors
```

[courses]

```
npm run seed-courses
```

RESET ALL DATABASE MIGRATION

```
npx db-migrate reset
```

GENERATE THE DOCUMENTATION

```
npm run gen-doc
```

1.2 Οδηγίες για την εφαρμογή iOS

Παρακάτω δίνονται οι εντολές που απαιτούνται για να μπορέσει κανείς να τρέξει την εφαρμογή σε σύστημα MAC OS.

CLONE

```
git clone git@github.com:constantinosMel/myUniLife-ios
```

CHANGE DIRECTORY

```
cd myUniLife-ios
```

INSTALL COCOAPODS

```
sudo gem install cocoapods
```

INSTALL COCOAPODS DEPENDECIES

```
pod install
```

CREATE ENVIRONMENT.XCCONFIG

```
cp Environment.xcconfig.example Environment.xcconfig
```

OPEN XCODE AND RUN

```
open ./myUniLife.xcworkspace/
```