ΠΑΝΕΠΙΣΤΗΜΙΟ ΜΑΚΕΔΟΝΙΑΣ

Διατμηματικό Πρόγραμμα Μεταπτυχιακών Σπουδών στα Πληροφοριακά Συστήματα

Complete CRUD Application with Angular, JSON Server and React Form

Ονοματεπώνυμο: Πετρόπουλος Στέφανος

Πατρώνυμο: Κωνσταντίνος

Επιβλέπων: Γεώργιος Ευαγγελίδης

Οκτώβριος 2022

Αφιέρωση

Αφιερώνω τη διπλωματική μου εργασία στην οικογένειά μου, για την ηθική και οικονομική υποστήριξη που μου παρείχαν κατά τη διάρκεια των σπουδών μου στη Θεσσαλονίκη.

Ευχαριστίες

Από αυτήν τη θέση θα ήθελα να ευχαριστήσω θερμά τον επιβλέποντα καθηγητή μου κ. Γεώργιο Ευαγγελίδη που μου έδωσε την ευκαιρία να ασχοληθώ με ένα τόσο ενδιαφέρον θέμα, καθώς και για την εμπιστοσύνη που μου έδειξε και για την αφιέρωση του πολύτιμου χρόνου του, με σκοπό την ολοκλήρωση της εργασίας μου.

Τέλος θα ήθελα να ευχαριστήσω όλους εκείνους που συνέβαλαν στην περάτωση της εργασίας αυτής. Ο καθένας με την ξεχωριστή ιδιότητά του και με τον δικό του τρόπο έβαλε το λιθαράκι του για την ολοκλήρωσή της.

Περίληψη

Ο σκοπός αυτής της διπλωματικής εργασίας είναι η πλήρης υλοποίηση μιας εφαρμογής CRUD Application με στόχο τη διαχείριση ενός συστήματος φοιτητών, οι οποίοι θα είναι οι βασικοί χρήστες του συστήματος. Για τον λόγο αυτό, κρίθηκε απαραίτητη η αναζήτηση και μελέτη σχετικής βιβλιογραφίας και tutorial, με αποτέλεσμα ο ερευνητής να αποκτήσει μια πληρέστερη εικόνα γύρω από το τι είναι ένα CRUD application, τις δυνατότητες του και τη χρησιμότητά που αυτό παρουσιάζει.

Έπειτα πραγματοποιήθηκε έρευνα για την εύρεση των κατάλληλων λογισμικών και της γλώσσας προγραμματισμού καθώς και της βάσης δεδομένων που είναι περισσότερο χρήσιμη και αποδοτική, έτσι ώστε να επιτευχθεί ένα πλήρως λειτουργικό αποτέλεσμα. Μετά την πραγματοποίηση της επιλογής των λογισμικών τα οποία θα χρησιμοποιηθούν για την υλοποίηση του CRUD application, ο ερευνητής επιλέγει τα δεδομένα με τα οποία θα ασχοληθεί και τα συμπληρώνει στον JSON-SERVER. Στην συνέχεια, μελετάται από τον ερευνητή η αρχιτεκτονική του κώδικα καθώς και τα API services, τα οποία δίνουν τη δυνατότητα στον χρήστη διαχείρισης του front end και του back end.

Τέλος, πραγματοποιήθηκε η υλοποίηση του CRUD application, η οποία αναλύεται διεξοδικά παρακάτω, και δίνονται περαιτέρω πληροφορίες για τη χρήση της εφαρμογής που το καθιστούν περισσότερο εύχρηστο.

Abstract

The purpose of this thesis is the implementation of a CRUD application, in order to manage a system, in which students are the users. The search and study of relevant bibliography and tutorials by the researcher, was necessary in order to thoroughly understand a CRUD application, its capabilities and its usefulness.

Afterwards, a research is carried out in order to find the appropriate software and programming language as well as the database that would be more useful and efficient, to achieve a fully functional result. The researcher has to select the software that he will use for the implementation of the CRUD application, chose the data and fill them in the JSON-SERVER. It is also required, the study of the architecture of the code and the API services, which enable the users to handle the front end and back end.

Finally, the implementation of the CRUD application is carried out, which is analyzed elaborately further down and more information is given about the application in order to facilitate and improve its use.

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΑΤΑΛΟΓΟΣ ΕΙΚΟΝΩΝ

ΚΑΤΑΛΟΓΟΣ ΠΙΝΑΚΩΝ

1. ΣΤΟΧΟΣ

1.1 Στόχος

Στόχος της παρούσας εργασίας είναι η δημιουργία λογισμικού διαχείρισης του προγράμματος κάθε φοιτητή. Βέβαια η εφαρμογή μπορεί να υποστηρίξει ταυτόχρονα και άλλες δυνατότητες . Το σύστημα υποστηρίζει και απευθύνεται σε φοιτητές, διδάσκοντες, στη γραμματεία ή γενικά σε εξωτερικούς χρήστες (επισκέπτες), που έχουν τη δυνατότητα πραγματοποίησης προσπέλασης και ανάκτησης δεδομένων.

1.2 Χρησιμότητα

Η σχεδίαση και ανάπτυξη του συγκεκριμένου CRUD Application αποτέλεσε το έναυσμα για την εκμάθηση νέων εργαλείων και την εμβάθυνση σε αυτά, τα οποία βρίσκονται σήμερα στη διάθεση ενός προγραμματιστή. Για την υλοποίηση της εργασίας χρησιμοποιήθηκε η γλώσσα προγραμματισμού Angular Typescript, μια σύγχρονη γλώσσα για προγραμματισμό διαδικτύου και με πολλές δυνατότητες. Πρωταρχικό ρόλο στην δημιουργία του κώδικα και στην οργάνωσή αυτού, κατείχε το Visual Studio Code , το οποίο αποτελεί ένα ολοκληρωμένο περιβάλλον ανάπτυξης IDE από την Microsoft και δίνει την δυνατότητα για μια σωστή και ελεγχόμενη σύνταξη του κώδικα. Επίσης, χρησιμοποιήθηκε η Bootstrap**,** μια συλλογή εργαλείων ανοιχτού κώδικα για τη δημιουργία ιστοσελίδων και διαδικτυακών εφαρμογών. Περιέχει HTML και CSS για τις μορφές τυπογραφίας, κουμπιά πλοήγησης και άλλων στοιχείων του περιβάλλοντος, καθώς και προαιρετικές επεκτάσεις JavaScript. Τέλος χρησιμοποιήθηκε ο διακομιστής JSON. Πρόκειται για μια μονάδα κόμβου, η οποία χρησιμοποιείται για τη δημιουργία επίδειξης υπηρεσιών **REST JSON** μέσα σε σύντομο χρονικό διάστημα μερικών λεπτών. Μοναδική προϋπόθεση αποτελεί η παρουσία ενός αρχείου JSON, ως δείγμα δεδομένων. Στην συγκεκριμένη εφαρμογή υπάρχουν δυο **db.json** αρχεία ένα για τους χρήστες, δηλαδή τους φοιτητές, και ένα για τα subjects.

1.3 Επισκόπηση

Η επισκόπηση αποτελεί έναν οδηγό του συστήματος που αναπτύχθηκε. Περιλαμβάνει τα παρακάτω κεφάλαια:

2.Ανάλυση

3.Σχεδίαση

4.Συμπεράσματα

5.Βιβλιογραφία

Η ανάλυση περιλαμβάνει τον προσδιορισμό των αναγκών του συστήματος διαχείρισης φοιτητών και εγγραφής μαθημάτων που αναπτύχθηκε. Επιπλέον περιλαμβάνει λεπτομερή ανάλυση των διαφόρων κατηγοριών χρηστών του συστήματος, και των διαφόρων λειτουργιών που είναι διαθέσιμες στους χρήστες (admin- user).

Η σχεδίαση αφορά την περιγραφή θεμάτων σχετικών με την σχεδίαση του λογισμικού διαχείρισης εγγραφής φοιτητών και επιλογής προγράμματος του κάθε φοιτητή. Πιο αναλυτικά, αρχικά παρατίθεται μια λεπτομερής περιγραφή της βάσης δεδομένων που χρησιμοποιήθηκε. Στην συνέχεια παρατίθεται το σχέδιο της βάσης δεδομένων και η αναλυτική σχεδίαση των πινάκων. Τέλος ακολουθεί το τμήμα που περιγράφει τις κυριότερες οθόνες της διεπιφάνειας του συστήματος με επεξήγηση της λειτουργίας τους και των βασικών στοιχείων υλοποίησης τους.

2. ΑΝΑΛΥΣΗ

2.1 Εισαγωγή ανάλυσης

2.1.1 Σκοπός

Ο σκοπός αυτού του τμήματος του εγγράφου, είναι ο προσδιορισμός των αναγκών και των περιορισμών του συστήματος που υλοποιήθηκε καθώς και η ανάλυση των χρηστών και των κυριότερων λειτουργιών που αυτοί έχουν τη δυνατότητα να επιτελέσουν.

2.1.2 Γενική Περιγραφή

Η συγκεκριμένη εφαρμογή αποτελεί ένα σύστημα διαχείρισης της έγγραφης των φοιτητών στο τμήμα παρακολούθησης διαδικτυακών μαθημάτων, επιλέγοντας οι ίδιοι τις ώρες και ημέρες τις οποίες επιθυμούν. Ο φοιτητής έχει την δυνατότητα να επιλέξει μέσω μιας πλατφόρμας μαθημάτων, τα μαθήματα τα οποία θέλει να παρακολουθήσει και στην συνέχεια να κάνει ένα μικρό τεστ με κάποιες ερωτήσεις, με σκοπό ο καθηγητής να δει τα ποσοστά του και το επίπεδο γνώσης του. Ο καθηγητής έχει την δυνατότητα να παρακολουθήσει τα ποσοστά και το πρόγραμμα κάθε φοιτητή με τα δεδομένα τα οποία συλλέγει από την βάση json.db. Αρχικά γίνεται η σχεδίαση της βάσης δεδομένων με προσδιορισμό των πινάκων, των πεδίων τους και του τύπου δεδομένων του καθενός, καθώς και οι σχέσεις μεταξύ τους. Κεντρικός άξονας της σχεδίασης είναι η αποτελεσματική υποστήριξη των λειτουργιών που θα επιτελεί το σύστημα. Στην συνέχεια γίνεται ο σχεδιασμός των διεπιφανειών για κάθε κατηγορία χρήστη που υποστηρίζει το σύστημα, με κριτήριο σχεδιασμού πάντα την ευχρηστία του συστήματος σε σχέση με τα χαρακτηριστικά των διαφόρων τύπων χρηστών. Τρεις είναι οι κατηγορίες χρήστη ο Φοιτητής, ο Διδάσκων και ο admin. Και οι τρεις κατηγορίες κατέχουν δικαίωμα στην προσθήκη, αφαίρεση και τροποποίηση δεδομένων του συστήματος μέσω αντίστοιχων λειτουργιών. Ωστόσο ο admin έχει και επιπλέον δυνατότητες στο σύστημα.

2.1.3 Ανάλυση Συστήματος

Η εφαρμογή προορίζεται για το διαδίκτυο και θα εκτελείται σε ένα διακομιστή, συνεπώς ο χρήστης χρειάζεται να έχει σύνδεση στο Internet. Η τωρινή μορφή της εφαρμογής είναι σε demo.

2.1.4 Εγκατάσταση της εφαρμογής

2.1.4.1 Εγκατάσταση Angular

Αρχικά πραγματοποιείται επιλογή μιας γλώσσας προγραμματισμού για την δημιουργία δυναμικών ιστοσελίδες. Μια γλώσσα που είναι ευρέως διαδεδομένη σήμερα είναι η Angular Typescript η οποία υποστηρίζει και τη σύνδεση με βάσεις δεδομένων, γεγονός πολύ σημαντικό για μια web εφαρμογή. Το Angular είναι μια πλατφόρμα ανάπτυξης, που βασίζεται στην Typescript. Ως πλατφόρμα, το Angular περιλαμβάνει:

- Ένα πλαίσιο βασισμένο σε στοιχεία για τη δημιουργία επεκτάσιμων εφαρμογών Ιστού
- Μια συλλογή από καλά ενσωματωμένες βιβλιοθήκες που καλύπτουν μια μεγάλη ποικιλία λειτουργιών, όπως δρομολόγηση, διαχείριση φορμών, επικοινωνία πελάτη-διακομιστή και άλλα

Το Angular βασίζει τα περιβάλλοντα κατασκευής του στο **Node.js** και πολλές από τις δυνατότητες του εξαρτώνται από πακέτα **NPM .** Αρχικά, ο πελάτης **Node Package Manager (NPM)** είναι μέρος του προεπιλεγμένου πακέτου Node.js. Για να επιβεβαιώσουμε ότι η εγκατάσταση του node.js έχει γίνει με επιτυχία, μπορούμε να περάσουμε την εντολή **node -v** στο terminal του Visual Studio ή στο cmd, Power shell.

```
PS C:\Users\theos\Desktop\sliderbar-admin\myblog> npm -v
6.14.8
```

(image 1)

Επιπλέον πρέπει να εγκαταστήσουμε το Angular CLI, το οποίο μας επιτρέπει να αρχικοποιήσουμε, να αναπτύξουμε και να διαχειριστούμε τις Angular εφαρμογές μας. Μπορούμε να χρησιμοποιήσουμε τη διαχείριση πακέτων NPM για να εγκαταστήσουμε το Angular CLI. Πληκτρολογώντας την παρακάτω εντολή: npm install -g @angular/cli στο terminal του Visual Studio. Κατά τη διαδικασία εγκατάστασης, το σύστημα μας ρωτά εάν θέλετε να μοιραστείτε δεδομένα χρήσης με την ομάδα Angular. Είστε ελεύθεροι να απαντήσετε είτε **Ναι** είτε **Όχι** , καθώς δεν επηρεάζει τη λειτουργικότητα.

```
? Would you like to share anonymous usage data with the Angular Team at Google under
Google's Privacy Policy at https://policies.google.com/privacy? For more details and
how to change this setting, see http://angular.io/analytics. Yes

Thank you for sharing anonymous usage data. If you change your mind, the following
command will disable this feature entirely:

    ng analytics off

+ @angular/cli@10.0.8
added 281 packages from 206 contributors in 59.541s
```

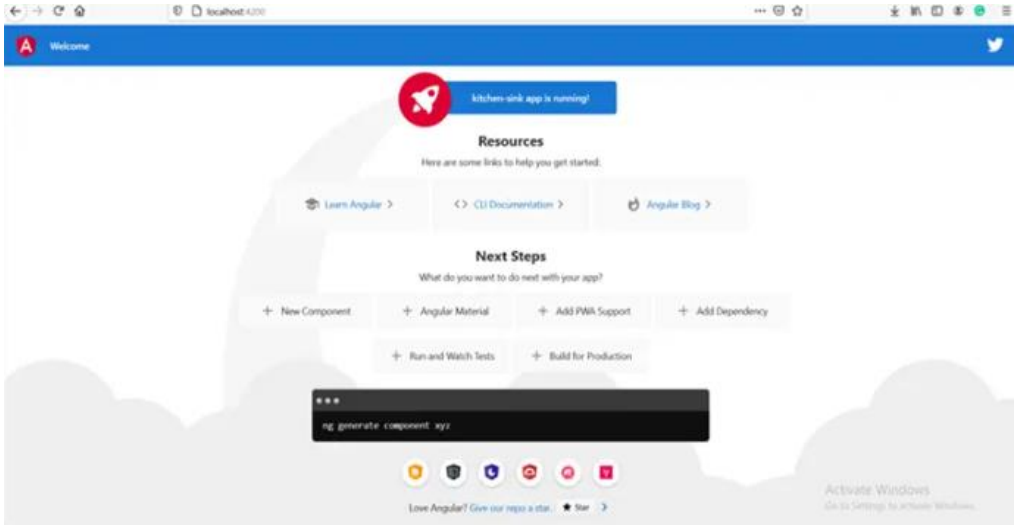
(image2)

Τέλος πληκτρολογούμε στο terminal την εντολή ng service. Το σύστημα προχωρά στη δημιουργία του περιβάλλοντος για την Angular εφαρμογή σας.

```
chunk {main} main.js, main.js.map (main) 60.6 kB [initial] [rendered]
chunk {polyfills} polyfills.js, polyfills.js.map (polyfills) 141 kB [initial] [rendered]
chunk {runtime} runtime.js, runtime.js.map (runtime) 6.15 kB [entry] [rendered]
chunk {styles} styles.js, styles.js.map (styles) 12.4 kB [initial] [rendered]
chunk {vendor} vendor.js, vendor.js.map (vendor) 2.66 MB [initial] [rendered]
Date: 2020-08-31T12:29:57.907Z - Hash: 5ca9e69cabc3fb6c3e85 - Time: 17096ms
** Angular Live Development Server is listening on localhost:4200, open your browser on http://localhost:4200/ **
: Compiled successfully.
```

(image3)

Χρησιμοποιούμε οποιοδήποτε πρόγραμμα περιήγησης για πρόσβαση στον τοπικό διακομιστή ανάπτυξης Angular στο **localhost:4200** .



(image4)

2.1.4.2 Εγκατάσταση JSON-Server

Όπως αναφέρθηκε και παραπάνω θα χρησιμοποιηθεί JSON-Server. Ο JSON Server μας βοηθάει να ρυθμίσουμε ένα REST API με λειτουργίες CRUD πολύ γρήγορα. Ο διακομιστής JSON είναι διαθέσιμος ως πακέτο NPM. Η εγκατάσταση μπορεί να γίνει χρησιμοποιώντας τον διαχειριστή πακέτων Node.js: $ npm install -g json-server. Προσθέτοντας την -g επιλογή βεβαιωνόμαστε ότι το πακέτο είναι εγκατεστημένο καθολικά στο σύστημα. Δημιουργώντας ένα νέο αρχείο με όνομα *db.json*. Αυτό το αρχείο περιέχει τα δεδομένα που πρέπει να εκτεθούν από το REST API. Για αντικείμενα που περιέχονται στη δομή JSON, τα entpoints CRUD δημιουργούνται αυτόματα.

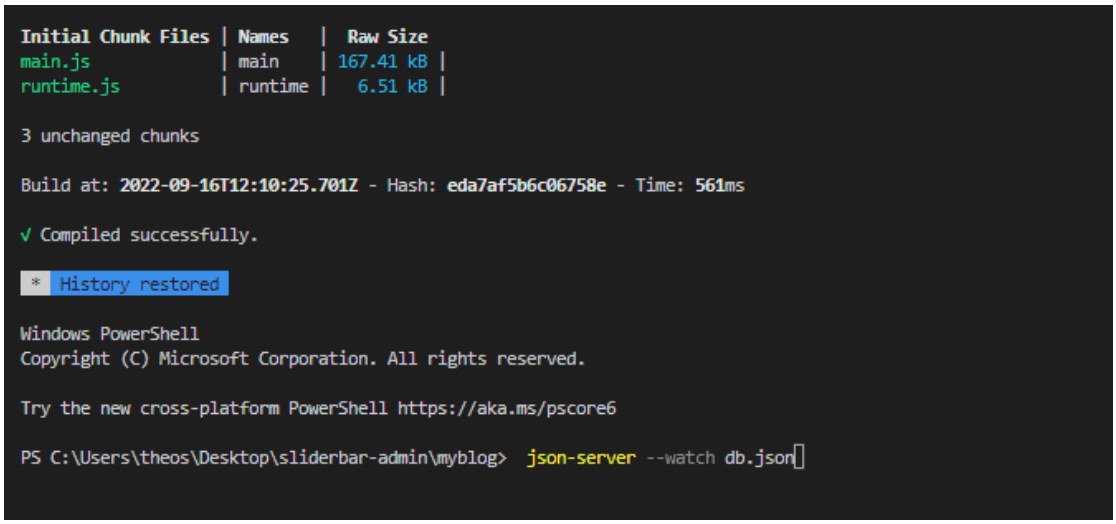
Βλέποντας το αρχείο db.json:


```
"students": [
  {
    "id": 1,
    "name": "giannis",
    "afm": "5364",
    "email": "giannisgmail.com",
    "phone": 6980468831,
    "address": "eteokleous"
  },
  {
    "id": 2,
    "name": "kwstas",
    "email": "kwstaspapakis@1234gmail.com",
    "phone": 6977363688,
    "address": "eteokleous"

  },

  {
    "id": 3,
    "name": "giannis",
    "email": "giannisptr1996@gmail.com",
    "phone": 6980484848,
    "address": "martiou"

  },
```

(image5)

Η δομή JSON αποτελείται από ένα αντικείμενο student στο οποίο έχουν εκχωρηθεί τρία σύνολα δεδομένων. Κάθε αντικείμενο student αποτελείται από πέντε ιδιότητες: *id* , *name* ,email *, phone , hours*.

Εκτελώντας τον διακομιστή JSON πρέπει να περάσουμε στο terminal του Visual Studio



(image6)

Ως παράμετρος πρέπει να περάσουμε πάνω από το αρχείο που περιέχει τη δομή JSON μας (db.json). Επιπλέον, χρησιμοποιούμε την παράμετρο — *watch.* Χρησιμοποιώντας αυτήν την παράμετρο, διασφαλίζουμε ότι ο διακομιστής έχει ξεκινήσει σε λειτουργία παρακολούθησης, πράγμα που σημαίνει ότι παρακολουθεί για αλλαγές αρχείων και ενημερώνει ανάλογα το εκτεθειμένο API.pi2.

2.1.5 Αρχιτεκτονική MVC

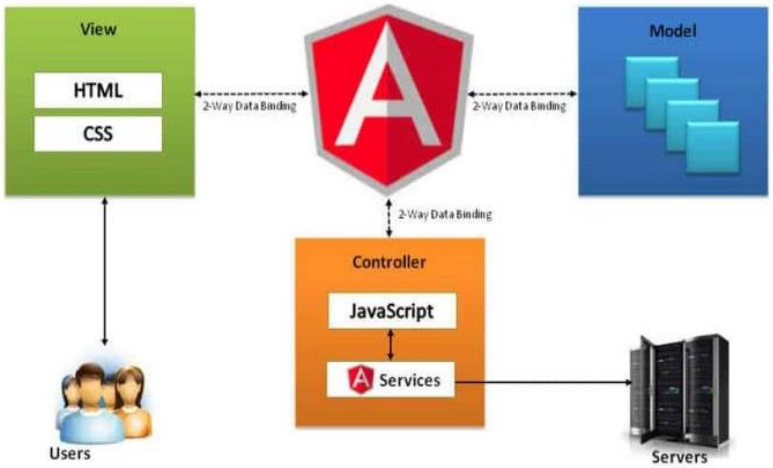
Το MVC (Model - View - Controller) περιγράφηκε για πρώτη φορά το 1979 από τον Trygve Reenskaug, και αργότερα χρησιμοποιήθηκε για Smalltalk στο Xerox PARC. Η αρχική υλοποίηση περιγράφεται σε βάθος στο "Applications Programming in Smalltalk-80: How to use Model–View–Controller".

Η AngularTs έχει πλέον καθιερωθεί ως μία από τις πιο αξιόπιστες, εύχρηστες και κατανοητές γλώσσες προγραμματισμού για να κατασκευάσει κάποιος μία web εφαρμογή. Τα τελευταία χρόνια έχουν αρχίσει να αναπτύσσονται ορισμένα εργαλεία που έχουν ως κύριο στόχο να διευκολύνουν τον προγραμματιστή. Τα εργαλεία αυτά βασίζονται σε JavaScript κυρίως που υποστηρίζεται πλήρως από την Angular και δίνουν την ευκαιρία στον προγραμματιστή να εξελίξει ακόμη περισσότερο τον κώδικα του.

Ένα άλλο σημαντικό στοιχείο των εργαλείων της Angular είναι ότι στηρίζονται πάνω στο πρότυπο MVC (Model View Controller). Με λίγα λόγια το πρότυπο αυτό βοηθά στην διατήρηση ενός οργανωμένου κώδικά έτσι ώστε να είναι πιο κατανοητός και πιο εύκολα συντηρούμενος. Το Model είναι το τμήμα που διαχειρίζεται λειτουργίες πάνω σε δεδομένα της βάσης (πχ CRUD functions), το View έχει να κάνει με το τι φαίνεται στο χρήστη, δηλαδή οι σελίδες της εφαρμογής και το Controller που βρίσκεται ενδιάμεσα, διαχειρίζεται τις ενέργειες του χρήστη, τις αναγνωρίζει και τις μεταφέρει στο κατάλληλο σημείο του κώδικα όπου θα επιτελεστεί μία συγκεκριμένη λειτουργία. Επίσης, δέχεται τα δεδομένα που προκύπτουν από τα Models και τα μεταφέρει στο κατάλληλο View.

2.1.6 Παράδειγμα της αρχιτεκτονικής του MVC

Χρησιμοποιώντας το μοντέλο σχεδιασμού MVC, το πρόγραμμα χωρίζεται σε τρία βασικά μέρη: 1. Το μοντέλο (Model) αναπαριστά τα δεδομένα της εφαρμογής 2. Η προβολή (View) παράγει μια παρουσίαση των δεδομένων του μοντέλου 3. Ο ελεγκτής (Controller) διαχειρίζεται και κατευθύνει τις αιτήσεις που κάνει ο χρήστης



(image7)

Θέτοντας ως παράδειγμα πως ένα τερματικό που το χειρίζεται ο User, έκανε κλικ στο δεσμό «Αγόρασε ένα κατά παραγγελία ηλεκτρικό ποδήλατο» στην κεντρική σελίδα της εφαρμογής σας.

- Ο user κάνει κλικ στον σύνδεσμο http://www.example.com/hlektrikopodilato/buy και ο φυλλομετρητής (browser) του στέλνει μια αίτηση στον εξυπηρετή (server) σας.
- Υπάρχει και ο διαβιβαστής (dispatcher) ο οποίος ελέγχει το url /hlektrikopodilato/buy και αποστέλλει το αίτημα στον κατάλληλο ελεγκτή (controller).
- Ο ελεγκτής εκτελεί την λογική της εφαρμογής. Για παράδειγμα μπορεί να ελέγχει αν ο user έχει κάνει εγγραφή (login) στο σύστημα.
- Ο ελεγκτής χρησιμοποιεί επίσης τα μοντέλα (models) για να αποκτήσει πρόσβαση στα δεδομένα της εφαρμογής. Τα μοντέλα συνήθως αναπαριστούν πίνακες της βάσης δεδομένων.
- Εφόσον ο ελεγκτής έχει ολοκληρώσει την εργασία του με τα δεδομένα, τα στέλνει στην προβολή (View). Η προβολή παίρνει τα δεδομένα από τον ελεγκτή και τα προετοιμάζει για παρουσίαση στον χρήστη. Οι προβολές συνήθως είναι σε μορφή HTML αλλά η προβολή μπορεί να είναι και ένα PDF, ένα XML αρχείο, ή ένα αντικείμενο JSON ανάλογα με τις ανάγκες σας.
- Μετά την δημιουργία της αναπαράστασης των δεδομένων από την προβολή, η αναπαράσταση αυτή αποστέλλεται στον user.

2.2 Ανάλυση χρηστών

2.2.1 Χαρακτηριστικά - Ικανότητες - Επιθυμίες χρηστών

Είναι γνωστό ότι για κάθε προϊόν λογισμικού, πρώτα από όλα καταγράφονται οι λειτουργικές απαιτήσεις του. Αυτό ορίζει το σημείο όπου πρέπει να καθοριστούν οι χρήστες του λογισμικού, καθώς και οι τρόποι με τους οποίους αλληλεπιδρούν με την εφαρμογή. Περιγράφουμε με άλλα λόγια τις δυνατότητες που υποστηρίζει η εφαρμογή. Όταν μιλάμε για τον προσδιορισμό των χρηστών προσδιορίζουμε τους τυπικούς χρήστες του συστήματος και στην συνέχεια την ανάλυση των χαρακτηριστικών τους, από την οποία θα προκύψουν χρήσιμα συμπεράσματα για τον σχεδιασμό του συστήματος με τρόπο που να ικανοποιεί τις ανάγκες των διαφόρων κατηγοριών χρηστών. Συγκεκριμένα καταγράφονται τα ατομικά χαρακτηριστικά, οι επιθυμίες τους, οι ικανότητες τους και σαφώς οι περιορισμοί που μπορεί να έχει κάθε χρήστης.

Στο σύστημα παρουσιάζονται δυο κατηγορίες χρήστη

- Καθηγητής (ο οποίος μπορεί να θεωρηθεί και ως admin)
- Φοιτητής

Για κάθε φοιτητή θεωρείται αυτονόητο η εξοικείωση με τη χρήση υπολογιστών καθώς και με πανομοιότυπες εφαρμογές διαδικτύου. Γενικά, οι φοιτητές όπως και κάθε άλλος χρήστης, επιθυμούν το περιβάλλον του συστήματος να είναι εύχρηστο, εύκολα κατανοητό, με απλούς τρόπους πλοήγησης παρόμοιους με αυτούς που έχουν συνηθίσει μέχρι σήμερα. Συγκεκριμένα, κάθε φοιτητής θα χρησιμοποιεί την πλατφόρμα για να εγγραφεί σε μαθήματα προγραμματισμού, να μπορεί να απαντήσει σε συγκεκριμένες ερωτήσεις μαθημάτων που τις επιλέγει ο καθηγητής του κάθε μαθήματος και να επιλέγει τις ώρες και την ημέρα που επιθυμεί να παρακολουθήσει κάποιο μάθημα. Επιπλέον, μπορεί να εγγραφεί σε νέο μάθημα ή να επεξεργαστεί κάποια ώρα ή ημέρα προκειμένου να αλλάξει το πρόγραμμα του ή να διαγράψει κάποια συμμετοχή του σε μάθημα πριν αλλά και μετά την έναρξη του μαθήματος.

Για την κατηγορία καθηγητής το περιβάλλον είναι επίσης πολύ φιλικό. Μπορεί να διαχειρίζεται τα στοιχεία του φοιτητή που επιβλέπει, να επιβλέπει τα μαθήματα που εγγράφετε ο φοιτητής καθώς και το πρόγραμμα του, να επεξεργάζεται την ημερομηνία που επιλέγει ο φοιτητής αν υπάρχει κάποιο πρόβλημα με την συγκεκριμένη επιλογή. Μπορεί επίσης να διαγράψει κάποιον φοιτητή αν αυτός σταματήσει να παρακολουθεί κάποιο μάθημα του ή τελειώσει η περίοδος παρακολούθησης του.

Φυσικά, όλες αυτές οι λειτουργίες θα πρέπει να είναι απλές στην εκτέλεση, εύκολες στην εκμάθηση και κατανοητές, διότι αποτελούν τακτικές εργασίες του προσωπικού και θα πρέπει να γίνονται με τρόπο που να εξασφαλίζεται η προστασία των δεδομένων από μη εξουσιοδοτημένη πρόσβαση.

2.2.2 Συντηρησιμότητα και Επεκτασιμότητα

Είναι σχεδόν βέβαιο ότι κάποια στιγμή στο μέλλον θα υπάρξει κάποια συντήρηση ή επέκταση του λογισμικού, για τον λόγο αυτό το λογισμικό πρέπει από την αρχή να έχει δομημένο κώδικα και πλήρως κατανοητό. Για να επιτευχθεί αυτό, απαιτείται η επεξήγηση του κώδικα με προσθήκη σχολίων που θα διευκολύνουν στην κατανόηση του, σε όποια

σημεία του κώδικα κρίνεται απαραίτητο. Βέβαια υπάρχει και η περίπτωση που ίσως η βάση db.json δεν καλύπτει αρκετές μελλοντικές ανάγκες. Στην περίπτωση αυτή απαιτείται αλλαγή της βάσης σε (SQL , mongo dB) οι οποίες αναμφισβήτητα δίνουν τις παραπάνω δυνατότητες.

Ευχρηστία Συστήματος

Το σύστημα είναι αρκετά απλό και κατανοητό στη χρήση του έτσι ώστε να είναι δυνατή η πλοήγηση και ενός χρήστη με ελάχιστη εμπειρία, ακολουθώντας τις οδηγίες χωρίς κανένα πρόβλημα.

Αξιοπιστία Συστήματος

Ακόμη και στην περίπτωση στην οποία ο χρήστης θα εισάγει λανθασμένα στοιχεία, το σύστημα θα είναι πλήρως λειτουργικό. Η αντικατάσταση των λανθασμένων στοιχείων με τα σωστά θα μπορεί να γίνει και σε δεύτερο χρόνο από τον χρήστη.

Ασφάλεια

Το σύστημα χρησιμοποιεί Secure HTTP (HTTPS), συνεπώς όλα τα δεδομένα που ανταλλάσσονται είναι κρυπτογραφημένα. Επίσης, το σύστημα απαγορεύει τη μη εξουσιοδοτημένη πρόσβαση σε λειτουργίες διαχείρισης δεδομένων όπως δημιουργία και ανάθεση εργασιών, εμφάνιση βαθμολογίας κλπ. Αυτό επιτυγχάνεται μέσω ελέγχου πρόσβασης του χρήστη με χρήση συνθηματικού, κωδικού πρόσβασης και με συνόδους ελέγχου.

2.3 Ανάλυση των λειτουργιών

Σε αυτό το σημείο το σημείο αναφέρονται οι διάφορες διεργασίες της εφαρμογής ανάλογα με τον χρήστη.

Φοιτητής

- Εγγραφή στο σύστημα Signup.
- Σύνδεση στην πλατφόρμα.
- Επεξεργασία στοιχείων του χρήστη στο σύστημα.
- Διαγραφή στοιχείων του χρήστη από το σύστημα.
- Εγγραφή σε μάθημα στο οποίο έχει επιλέξει.
- Προσθήκη νέου μαθήματος σε νέο πρόγραμμα (με διαφορετική ημερομηνία και ώρα).
- Επεξεργασία του μαθήματος και ημερομηνία/ώρας.
- Διαγραφή του μαθήματος από το πρόγραμμα του.
- Αναζήτηση ημερομηνίας και ώρας στο πρόγραμμα.
- Επιλογή ημερομηνίας και ώρας στο πρόγραμμα.
- Δυνατότητα απαντήσεων σε ερωτήσεις σχετικές με το μάθημα.
- Προβολή σωστών και λανθασμένων απαντήσεων.
- Αποσύνδεση από την πλατφόρμα.

Καθηγητής

- Εγγραφή στο σύστημα Signup.
- Σύνδεση στην πλατφόρμα.
- Επεξεργασία στοιχείων του φοιτητή στο σύστημα.
- Επεξεργασία μαθήματος του φοιτητή στο σύστημα.
- Διαγραφή φοιτητή από την βάση δεδομένων.
- Διαγραφή μαθήματος από την βάση δεδομένων.
- Ορισμός ημερομηνίας και ώρας του μαθήματος (το μάθημα παρακολουθείτε καθημερινές).
- Προσθήκη νέου φοιτητή στο τμήμα.
- Προσθήκη και αλλαγή μαθήματος με κάποιο άλλο.
- Επεξεργασία ερωτήσεων σχετικά με το κάθε μάθημα που πρόκειται να απαντηθούν από το σύνολο των φοιτητών.
- Αποσύνδεση από την πλατφόρμα.

3.ΣΧΕΔΙΑΣΗ

3.1 Περιγραφή Βάσης Δεδομένων

Με στόχο την λειτουργικότητα της εφαρμογής και την παρουσίαση της κατάλληλης πληροφορίας, πραγματοποιείται διαχείριση δυναμικών δεδομένων. Τα δεδομένα αυτά αποθηκεύονται σε μια βάση δεδομένων με την οποία η εφαρμογή επικοινωνεί. Για το λόγο αυτό χρησιμοποιείται βάση δεδομένων JSON, που αποτελεί ένα τύπο μη σχεσιακής βάσης δεδομένων που έχει σχεδιαστεί για να αποθηκεύει και να αναζητά δεδομένα ως έγγραφα JSON, αντί να κανονικοποιεί δεδομένα σε πολλούς πίνακες, ο καθένας με μια μοναδική και σταθερή δομή, όπως σε μια σχεσιακή βάση δεδομένων. Οι βάσεις δεδομένων εγγράφων JSON χρησιμοποιούν την ίδια μορφή εγγράφου-μοντέλου που χρησιμοποιούν οι προγραμματιστές στον κώδικα της εφαρμογής τους, γεγονός που διευκολύνει την αποθήκευση και την αναζήτηση δεδομένων. Παρακάτω παρατίθενται αναλυτικά οι πίνακες της βάσης. Στην ενότητα αυτή γίνεται μια αναλυτική περιγραφή των πινάκων της βάσης δεδομένων και των πεδίων που περιλαμβάνει ο καθένας τους. Στη συνέχεια παρατίθεται μια περιγραφή του σχεδιασμού και της λειτουργίας κάθε μιας από τις κύριες οθόνες του συστήματος.

3.2 Βάση Δεδομένων και Πίνακες

Τα δεδομένα τα οποία καταχωρούν οι χρήστες αποθηκεύονται σε κάποιους πίνακες, οι οποίοι είναι αποθηκευμένοι με την σειρά τους στην JSON.db που χρησιμοποιούμε. Αυτοί οι πίνακες είναι οι εξής:

Πίνακας Signup Users

Δομή:

Όνομα πεδίου	Τύπος	Πρόσθετα
Id	**Int**	**Αυξανόμενος αριθμός**
Full name	**Varchar**	
ΑΦΜ	**Int**	
Mobile	**Number**	
Email	**Varchar**	
Password	**Int**	

- Το Id είναι μοναδικός αριθμός κάθε εγγραφής χρήστη. Κάθε φορά που πραγματοποιείται μια εγγραφή, αυτός ο αριθμός αυξάνεται αυτόματα.
- Το Full name είναι το όνομα με το οποίο κάνει εγγραφή ο χρήστης στην εφαρμογή.
- Το ΑΦΜ του φοιτητή θεωρείται βασικό πεδίο και θεωρητικά είναι το κύριο κλειδί που ενώνει τον πίνακα του φοιτητή με τον πίνακα μαθημάτων.
- Το mobile είναι ο αριθμός τηλεφώνου του χρήστη το οποίο θεωρείται και αυτό ένα υποχρεωτικό πεδίο στην συμπλήρωση της φόρμας εγγραφής.
- Το Email είναι το προσωπικό email για επικοινωνία του χρήστη και αποτελεί υποχρεωτικό πεδίο του πίνακα.
- Το password είναι ο κωδικός με τον οποίο ο χρήστης θα κάνει login στην πλατφόρμα, ένα πολύ σημαντικό πεδίο του πίνακα το οποίο θέλει ιδιαίτερη προσοχή κατά την εισαγωγή του από το χρήστη, ώστε να εισάγουν έναν ασφαλή κωδικό.

Πίνακας Students: ο πίνακας αυτός αναφέρεται στους μαθητές οι οποίοι έχουν εγγραφεί και αυτόματα τα στοιχεία τους καταχωρούνται στον πίνακα που διαχειρίζεται ο καθηγητής, με εξαίρεση τον κωδικό πρόσβασης.

Δομή:

Όνομα πεδίου	Τύπος	Πρόσθετα
Id	**Int**	**Αυξανόμενος αριθμός**
Name	**Varchar**	
ΑΦΜ	**Number**	
Email	**Varchar**	
Phone	**Number**	
Address	**Varchar**	

- Το Id είναι μοναδικός αριθμός κάθε εγγραφής φοιτητή. Κάθε φορά που πραγματοποιείται μια εγγραφή, ο αριθμός αυτός αυξάνεται αυτόματα.
- Το name είναι το όνομα με το οποίο κάνει εγγραφή ο φοιτητής στην εφαρμογή.
- Το ΑΦΜ του φοιτητή θεωρείται βασικό πεδίο και θεωρητικά είναι το κύριο κλειδί που ενώνει τον πίνακα του φοιτητή με τον πίνακα μαθημάτων.
- Το phone είναι ο αριθμός τηλεφώνου το φοιτητή το οποίο θεωρείται και αυτό ένα υποχρεωτικό πεδίο στην συμπλήρωση της φόρμας εγγραφής.
- Το Email είναι και αυτό ένα υποχρεωτικό πεδίο του πίνακα, όπου ο φοιτητής πρέπει να εισάγει ένα προσωπικό του email για επικοινωνία.

Πίνακας Subjects: Αναφέρεται στα μαθήματα που μπορεί να επιλέξει ο φοιτητής και τις ώρες και μέρες παρακολούθησης τους, τις οποίες ο φοιτητής είναι υποχρεωμένος να συμπληρώσει.

Δομή:

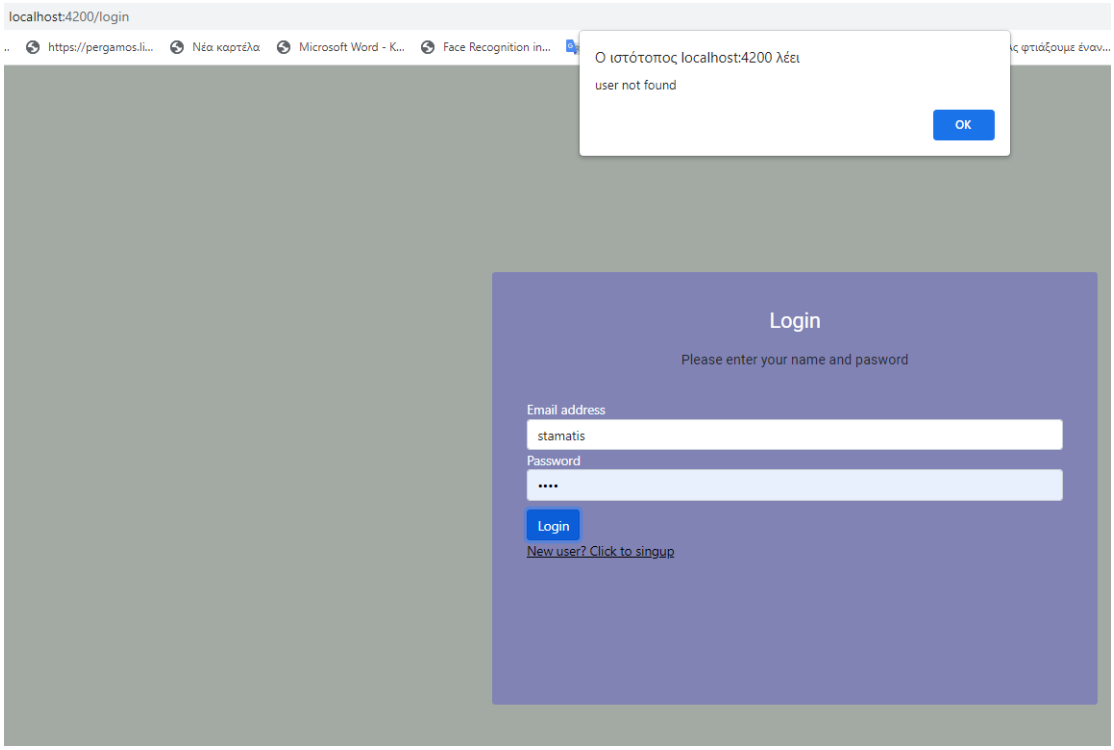
Όνομα πεδίου	Τύπος	Πρόσθετα
Id	**Int**	**Αυξανόμενος αριθμός**
Name	**Varchar**	
Subject	**Number**	
Date	**Date**	
Hours	**Number**	

- Το Id είναι μοναδικός αριθμός κάθε μαθήματος. Κάθε φορά που θα γίνεται μια προσθήκη νέου μαθήματος από τον φοιτητή, αυτός ο αριθμός θα αυξάνεται αυτόματα
- Το Name είναι το όνομα με το οποίο κάνει εγγραφή ο φοιτητής στο μάθημα.
- Το Subject είναι το όνομα του μαθήματος το οποίο είναι μοναδικό για κάθε μάθημα.
- Το Date είναι η ημερομηνία που επιλέγει να παρακολουθήσει το μάθημα ο φοιτητής.
- Το Hours είναι οι ώρες παρακολούθησης ενός μαθήματος και είναι κλειδωμένες σε δύο τιμές 12:00-14:00 και 16:00-18:00.

3.3 Η Μορφή της εφαρμογής

Για την λειτουργικότητα της εφαρμογής, όπως αυτή περιγράφηκε παραπάνω σχεδιάστηκαν κατάλληλες σελίδες και τμήματα σελίδων, μεταξύ των οποίων πλοηγείται ο χρήστης αλληλεπιδρώντας με την εφαρμογή ώστε να επιτευχθεί το επιθυμητό αποτέλεσμα. Η γενική ιδέα για τη διεπιφάνεια της εφαρμογής, συνίσταται στη χρήση συστατικών που τοποθετούνται στις σελίδες της και της προσδίδουν την απαιτούμενη λειτουργικότητα. Συστατικά όπως φόρμες, κουμπιά, μενού επιλογής και πίνακες, γνώριμα στους χρήστες του διαδικτύου, τα οποία δομούνται ακολουθώντας ένα νοητό περίγραμμα (template) ώστε να συνθέσουν μια διεπαφή φιλική στον χρήστη. Οι φόρμες χρησιμοποιούνται όπου προβλέπεται – αναμένεται είσοδος από το χρήστη, τα κουμπιά για την αλληλεπίδραση του χρηστή με την εφαρμογή, επιτρέποντας την πυροδότηση γεγονότων, οι πίνακες για παρουσίαση δεδομένων, ενώ τα μενού επιλογής σε σημεία όπου η είσοδος που αναμένεται από τον χρήστη είναι τυποποιημένη, οπότε του δίνεται η δυνατότητα να επιλέξει παρά να πληκτρολογήσει.

Σε περίπτωση κάποιας λανθασμένης επιλογής του χρήστη, όπως ένα λανθασμένο password ή email. Ο χρήστης ενημερώνεται με αυτοματοποιημένο μήνυμα όπως θα δούμε και στην image 8.



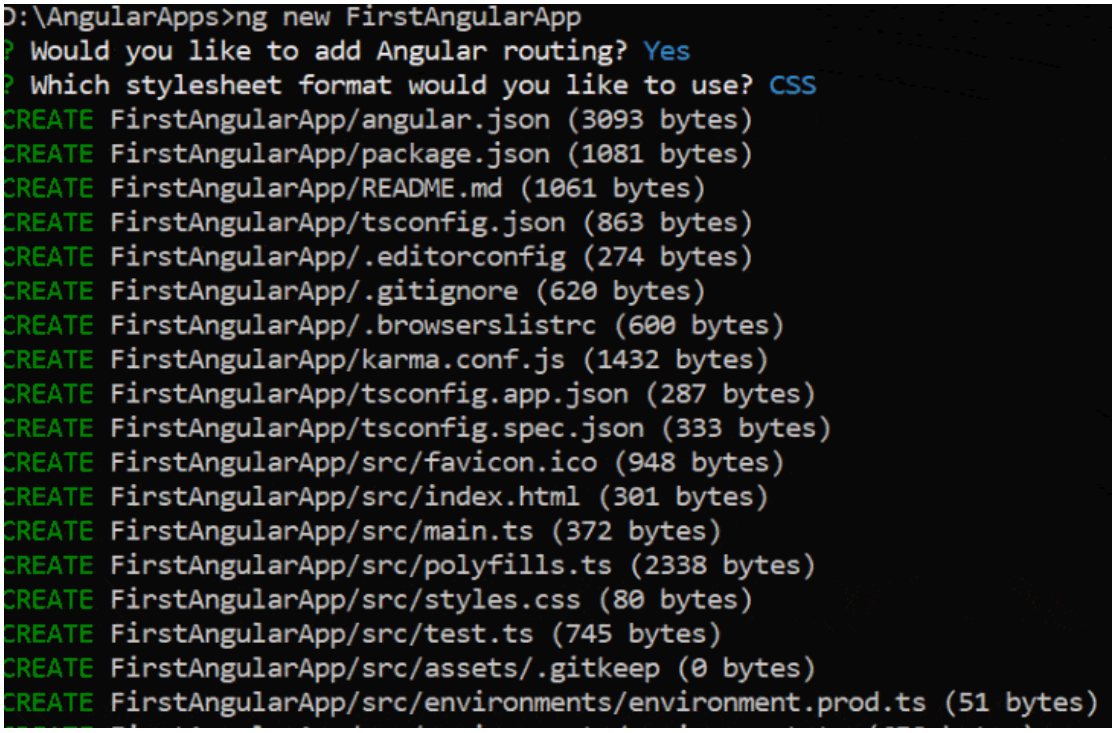
(image8)

Στην αρχική σελίδα, λοιπόν, παρουσιάζεται φόρμα εισόδου στην υπηρεσία, όπου οι χρήστες πρέπει να επιλέξουν την ιδιότητα τους. Στο σημείο αυτό, πρέπει να ενημερώσουμε ότι η ιδιότητα του φοιτητή είναι ως απλός χρήστης και μπορεί με την εγγραφή του να επιλέξει με ένα check τον ρόλο του σαν χρήστης. Ανάλογα με την επιλογή που θα κάνουν θα συνδεθούν στην εφαρμογή ως admin (καθηγητής), είτε θα τους ζητήσει να εισάγουν το όνομα χρήστη και τον μυστικό κωδικό πρόσβασης και εφόσον οι κωδικοί αυτοί επαληθευτούν (τοπικά ή μη), αναγνωρίζεται ο χρήστης και μεταφερόμαστε στη σελίδα του χρήστη, το λεγόμενο Student Profile. Η ίδια ακριβώς διαδικασία γίνεται και στην περίπτωση που ο χρήστης είναι ο καθηγητής, τότε εισάγει το email και το password του και έπειτα μπορεί να συνδεθεί στην εφαρμογή εφόσον οι κωδικοί αυτοί επαληθευτούν (τοπικά ή μη), αναγνωρίζεται ο χρήστης και μεταφερόμαστε στη σελίδα του χρήστη, το λεγόμενο Management System.

3.4 Αναφορική παρουσίαση του Κώδικα

Σε αυτό το τμήμα παρουσιάζεται η δημιουργία μιας εφαρμογής Angular χρησιμοποιώντας το Angular CLI. Το Angular CLI βοηθά στη γρήγορη ρύθμιση ενός χώρου εργασίας και μιας αρχικής εφαρμογής, η οποία περιλαμβάνει τις απαραίτητες βιβλιοθήκες NPM και άλλες εξαρτήσεις για την εφαρμογή. Για τη δημιουργία μιας αρχικής εφαρμογής, γίνεται μετάβαση στον φάκελο όπου επιθυμείται να δημιουργηθεί μια εφαρμογή και εκτελείται η ng new <My Blog> στο terminal του visual studio.

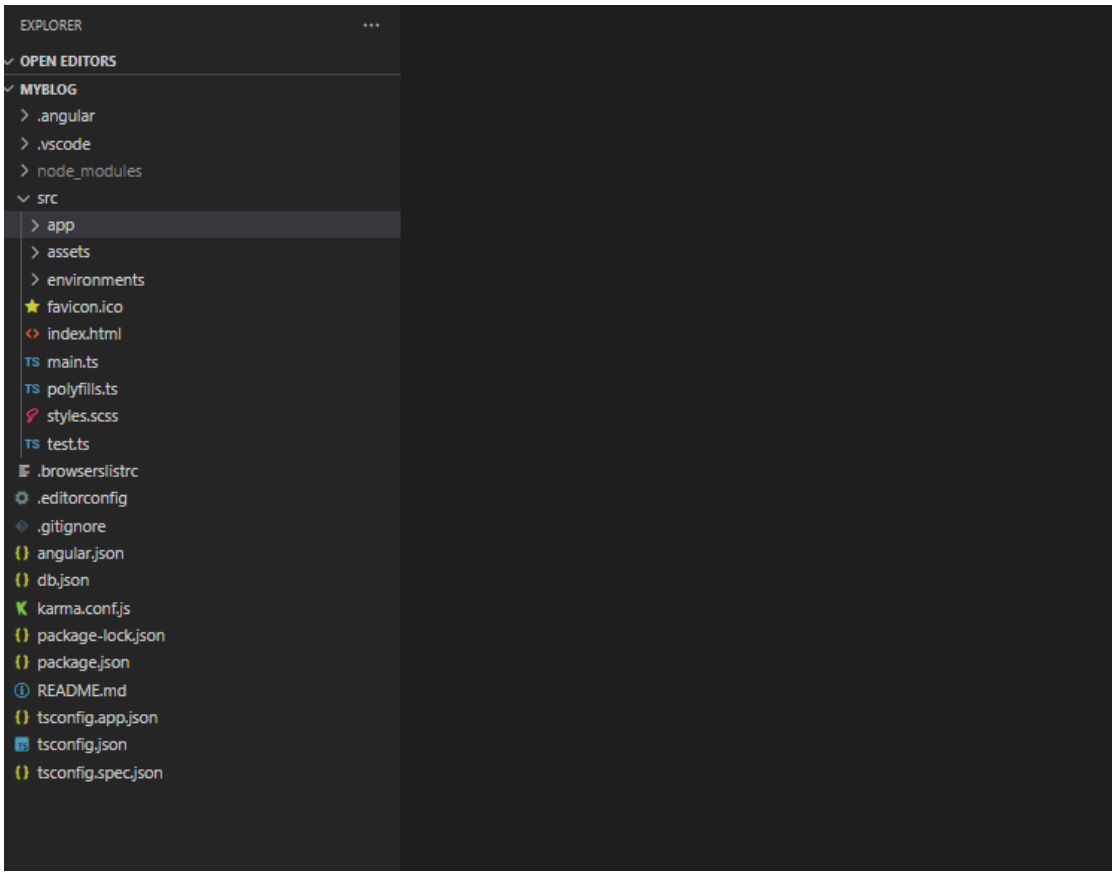
Η ng new εντολή ζητά πληροφορίες σχετικά με λειτουργίες που πρέπει να συμπεριληφθούν στο αρχικό έργο της εφαρμογής. Υπάρχει η δυνατότητα αποδοχής των προεπιλογών, πατώντας ένα πλήκτρο Enter, όπως φαίνεται παρακάτω.



(image9)

Η παραπάνω εντολή μπορεί να διαρκέσει 2-3 λεπτά για τη δημιουργία ενός έργου και την εγκατάσταση των απαραίτητων βιβλιοθηκών. Για άνοιγμα αυτού του έργου σε VS Code, γίνεται μετάβαση στο φάκελο του έργου στο παράθυρο τερματικού/εντολών και πληκτρολογείτε code..

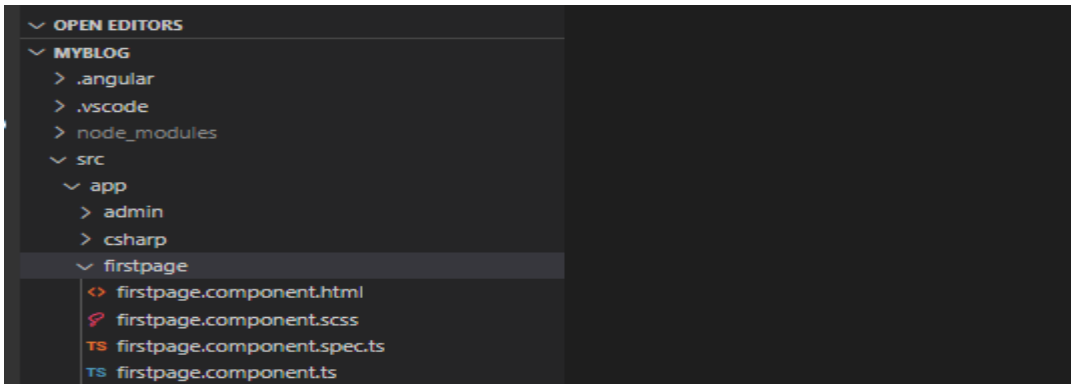
Το νέο έργο εμφανίζεται με την παρακάτω μορφή



(image 10)

Το αριστερό παράθυρο στο VS Code παρουσιάζει τα αρχεία και τους φακέλους που δημιουργήθηκαν από το Angular CLI.

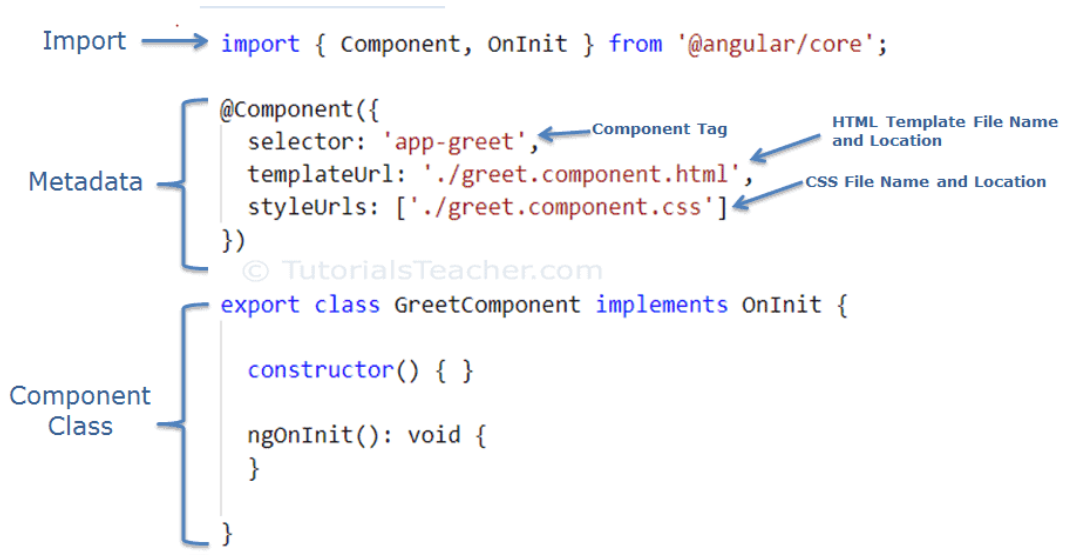
Αρχεία για ένα στοιχείο μπορούν να δημιουργηθούν με μη αυτόματο τρόπο ή χρησιμοποιώντας την εντολή Angular CLI. Το Angular CLI μειώνει τον χρόνο ανάπτυξης. Έτσι χρησιμοποιούμε το Angular CLI για τη δημιουργία ενός νέου στοιχείου. Χρησιμοποιώντας την ακόλουθη εντολή ng generate component **login** δημιουργείται ένας νέος φάκελος "greet" και φάκελος εφαρμογής και δημιουργούνται τέσσερα αρχεία, όπως φαίνεται παρακάτω.



(image11)

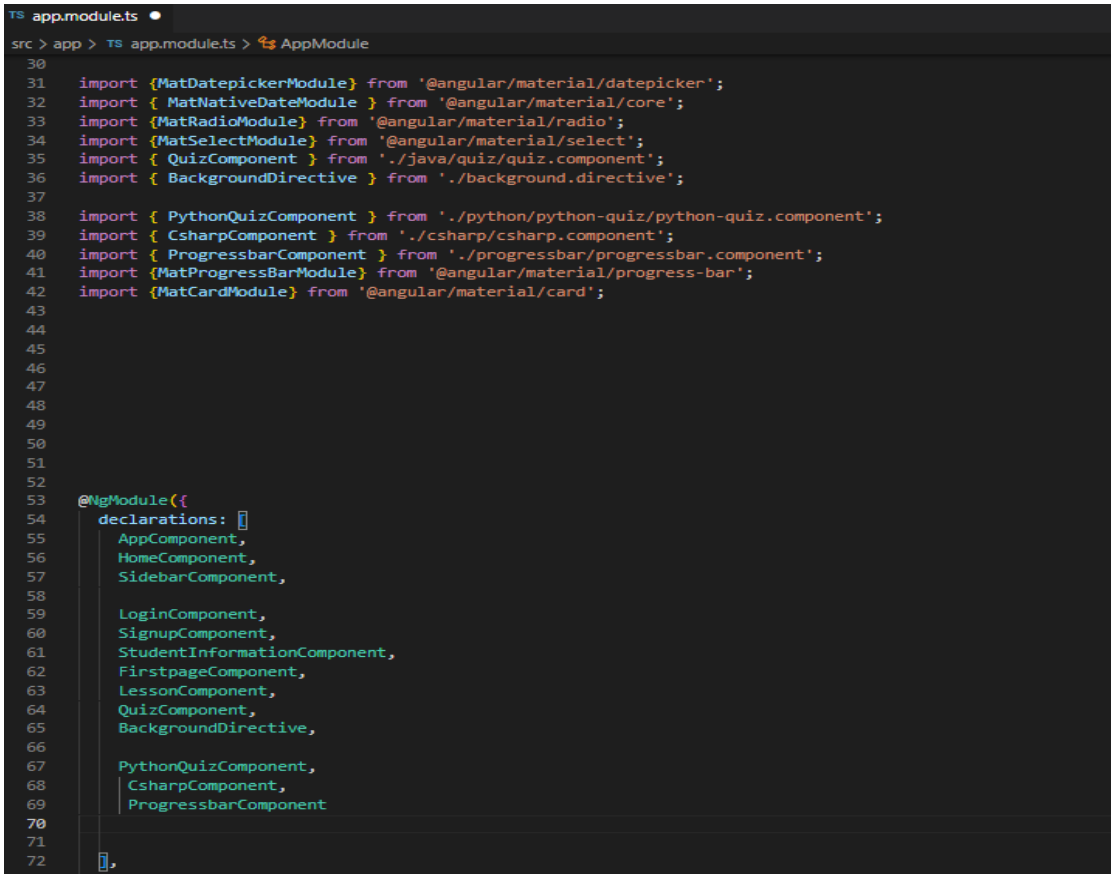
Παραπάνω, το firstpage.component.css είναι ένα αρχείο CSS για το στοιχείο, το firstpage.component.html είναι ένα αρχείο HTML για το στοιχείο όπου θα γράψουμε HTML, το firstpage.component.spec.ts είναι ένα δοκιμαστικό αρχείο όπου μπορούμε να γράψουμε μοναδιαίες δοκιμές για ένα στοιχείο και το firstpage.component.ts είναι το αρχείο κλάσης για ένα στοιχείο.

Στο παρακάτω σχήμα απεικονίζει το σημαντικό μέρος της κατηγορίας component



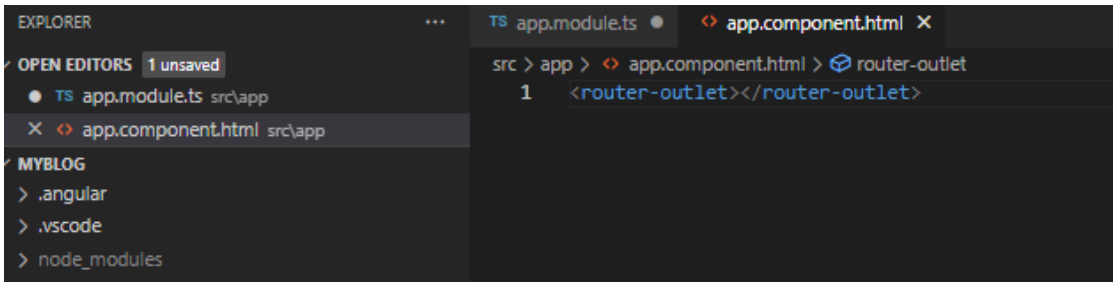
Component Class: firstpage.component είναι η κλάση στοιχείων. Περιέχει ιδιότητες και μεθόδους αλληλεπίδρασης με την προβολή μέσω ενός Angular API. Υλοποιεί τη OnInit διεπαφή, η οποία είναι ένα άγκιστρο κύκλου ζωής.

Component Metadata: Το @component είναι ένας διακοσμητής που χρησιμοποιείται για τον καθορισμό των μεταδεδομένων για την κατηγορία στοιχείων που ορίζεται αμέσως κάτω από αυτό. Είναι μια συνάρτηση και μπορεί να περιλαμβάνει διαφορετικές ρυθμίσεις παραμέτρων για το στοιχείο. Καθοδηγεί την Angular πού να πάρει τα απαιτούμενα αρχεία για το στοιχείο, να δημιουργήσει και να αποδώσει το στοιχείο. Όλα τα εξαρτήματα Angular πρέπει να έχουν @component είναι διακοσμητή πάνω από την κατηγορία εξαρτημάτων. Η δήλωση εισαγωγής λαμβάνει την απαιτούμενη δυνατότητα από τη Angular ή άλλες βιβλιοθήκες. Το import μας επιτρέπει να χρησιμοποιούμε εξωτερικά μέλη από εξαγόμενα modules. Στην συνέχεια πρέπει να δηλώσουμε το firstpagecomponent στο αρχείο root γιατί το root module πρέπει να το γνωρίζει. Έτσι εμείς προσθέτουμε κάθε νέο component στο declaration array.



(image12)

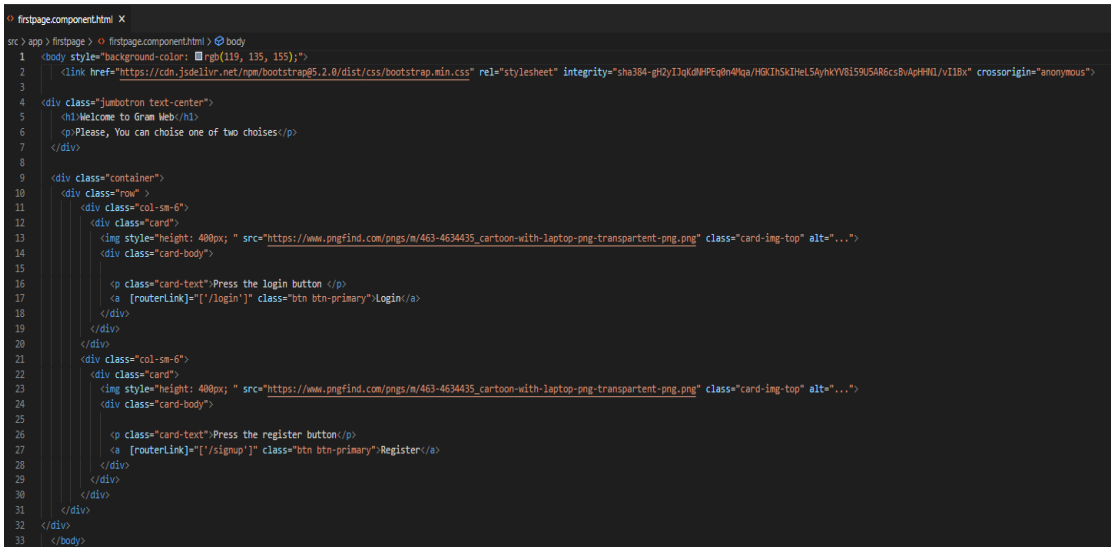
Αφού γίνει η προσθήκη το component στο declaration array του app.module προσθέτουμε το <router-outlet></ router-outlet> στο app.component.html.



(image13)

First Page Component

Με την ολοκλήρωση της διαδικασίας αυτής μπορούμε να ξεκινήσουμε να δημιουργούμε τον κώδικα μας στο firstpage.component.html και στην συνέχεια στο firstpage.component.ts αρχείο μας.

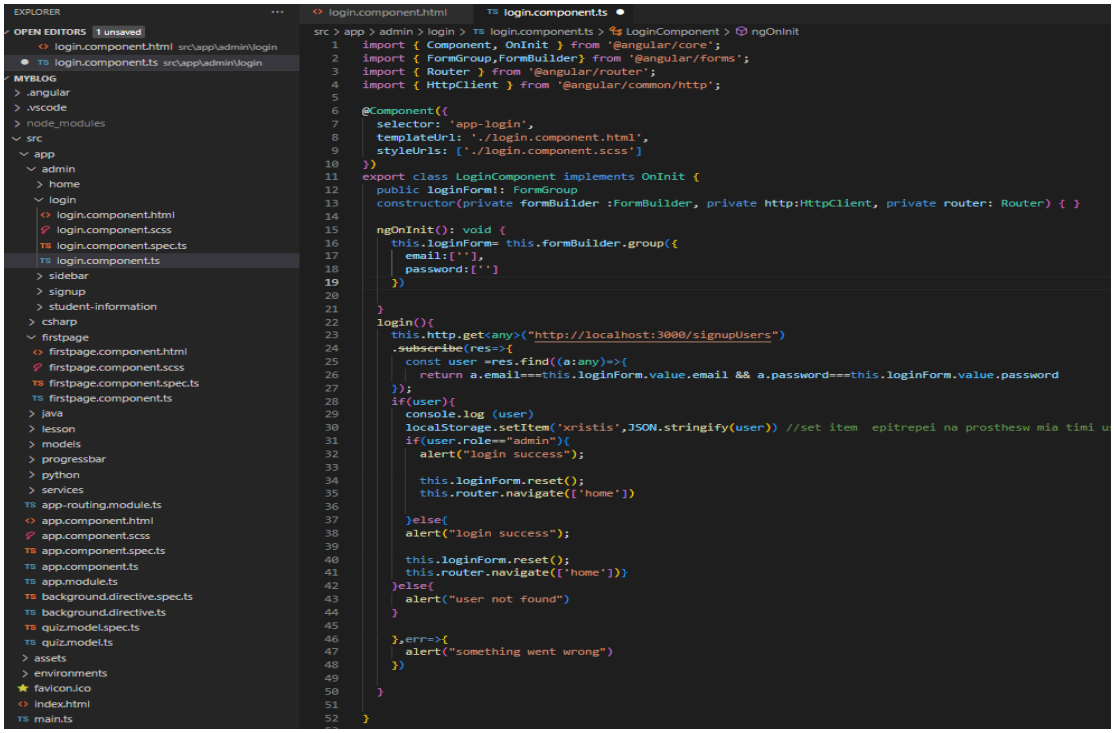


(image13)

Στην παραπάνω εικόνα παρουσιάζεται η δημιουργία δυο cards, το ένα αφορά τον χρήστη που έχει εγγραφεί και θέλει να συνδεθεί απευθείας στην πλατφόρμα και η δεύτερη κάρτα τον χρήστη που θέλει να εγγραφεί. Κάθε μία κάρτα περιέχει ένα routerLink που συνδέει απευθείας, είτε με την login page, είτε με την signup page.

Login Component

Το δεύτερο component το οποίο δημιουργήσαμε είναι το login.component με το οποίο θα χτίσουμε την σελίδα, μέσω της οποίας ο χρήστης μπορεί να συνδεθεί στην εφαρμογή μας.



(image14)

Όπως βλέπουμε στην εικόνα 14 δημιουργώντας το νέο component login δημιουργήθηκαν τρία αρχεία css, html, ts. Αρχικά στο login.component.ts εισάγουμε τις βιβλιοθήκες για angular material form-filed, εισάγουμε το Router από την υπηρεσία angular Router, η οποία επιτρέπει την πλοήγηση από τη μια προβολή στην άλλη καθώς οι χρήστες εκτελούν εργασίες εφαρμογής. Επιπλέον για να μπορέσουμε να εκτελέσουμε τα αιτήματα από το api.services για το οποίο θα μιλήσουμε στην συνέχεια. Το HttpClient χρησιμοποιείται για την εκτέλεση αιτημάτων HTTP και εισάγεται από τη μορφή @angular/common/http.

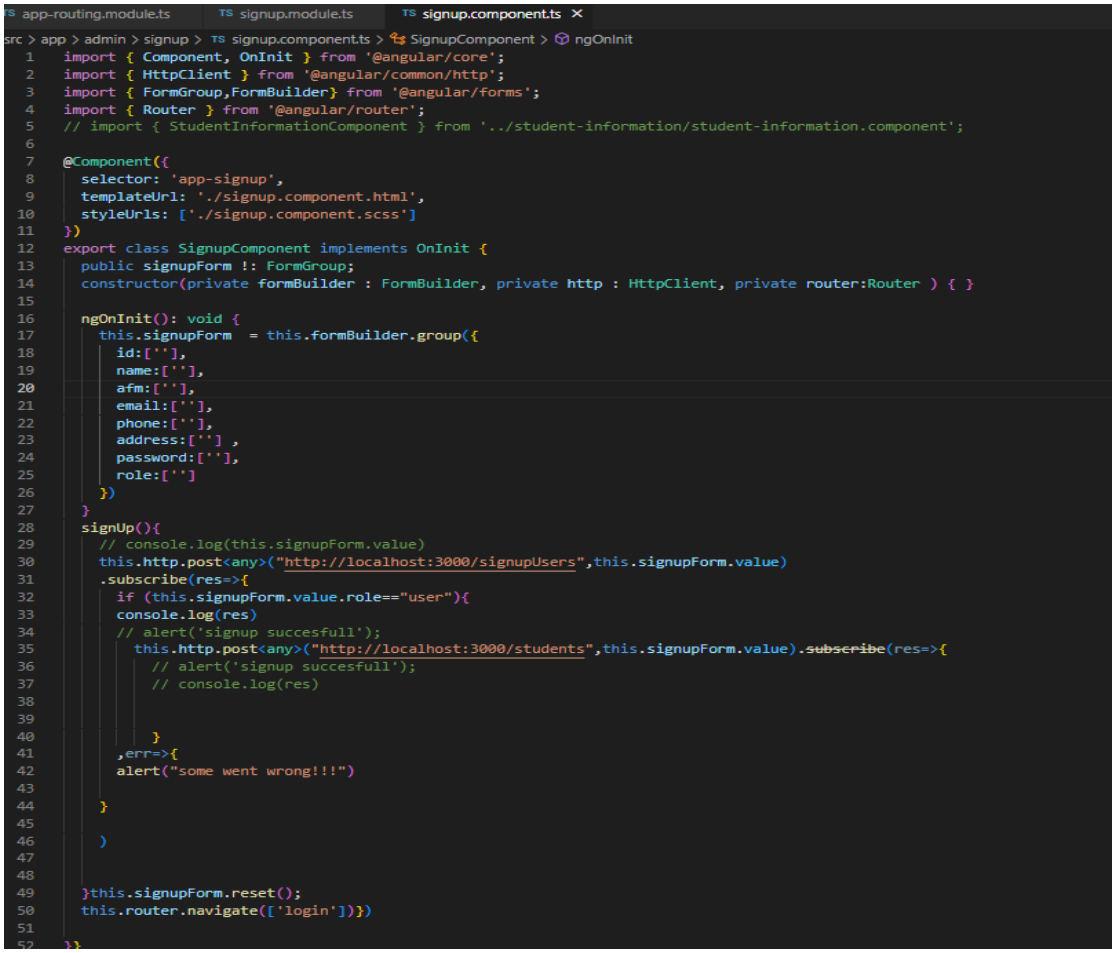
Στο αρχείο login.component.ts δημιουργούμε την δήλωση μίας φόρμας στην οποία εισάγουμε το email και password. Με την δημιουργία της μεθόδου login() κάνουμε εγγραφή όλα τα στοιχεία του signupUsers.db στο res και στην συνέχεια ελέγχουμε αν τα αντίστοιχα στοιχεία που συμπληρώνει ο χρήστης στην φόρμα που δημιουργούμε στην html αντιστοιχούν με το email και password του αντίστοιχου χρήστη. Αν αντιστοιχούν τότε η σύνδεση στην φόρμα login γίνετε με επιτυχία και ο χρήστης συνδέεται με την πλατφόρμα, αλλιώς εμφανίζεται μήνυμα σφάλματος και ο χρήστης δοκιμάζει ξανά να συμπληρώσει τα σωστά στοιχεία του.

(image15)

Στο αρχείο login.component.html δημιουργούμε τα πεδία της φόρμας login η οποία περιέχει δύο πεδία το email και το password και ενεργοποιείται με την μέθοδο login() που αναλύσαμε παραπάνω. Αφού δημιουργήσαμε τα πεδία, έπειτα δημιουργούμε ένα button με το οποίο συνδεόμαστε στην πλατφόρμα. Στην περίπτωση που δεν έχουμε εγγραφεί υπάρχει και ένα δεύτερο link που μας συνδέει στην φόρμα εγγραφής.

SignUp Component

Το τρίτο component που δημιουργήσαμε για την πλατφόρμα μας με σκοπό να εισάγουμε στο αρχείο signup.component.ts την φόρμα με την οποία ο χρήστης θα κάνει εγγραφή στην πλατφόρμα και θα μπορεί μέσω της μεθόδου signUp() να περνάει δεδομένα με τη μέθοδο post στον SignupUsers.json με σκοπό η μέθοδος login() να μπορεί να κάνει get τα στοιχεία αυτά όπως αναφέραμε παραπάνω.

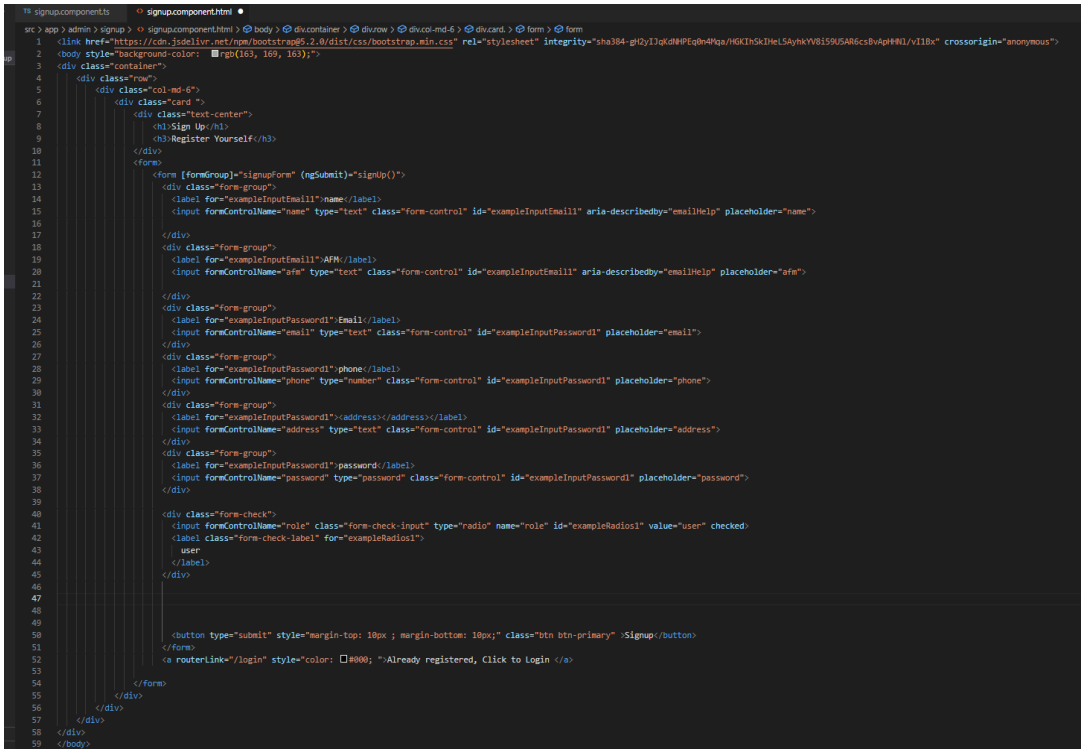


(image16)

Στην συνέχεια αφού κάνουμε post όλα τα value της signup φόρμας, ελέγχουμε αν ο ρόλος του χρήστη είναι user η admin. Αν ο ρόλος είναι user θεωρείται ως φοιτητής και περνάμε την πληροφορία αυτή στο studens.db .

Με την μέθοδο .reset καθαρίζουμε τη φόρμα με σκοπό στην επόμενη εγγραφή νέου χρήστη να μην αποθηκεύει τα παλιά στοιχεία άλλου χρήστη. Τέλος με το .navigate μεταφερόμαστε στην login page όπου θα εισάγουμε το email και password.

Στο αρχείο signup.component.html δημιουργούμε τα πεδία της φόρμας όπως φαίνεται στην εικόνα 17.

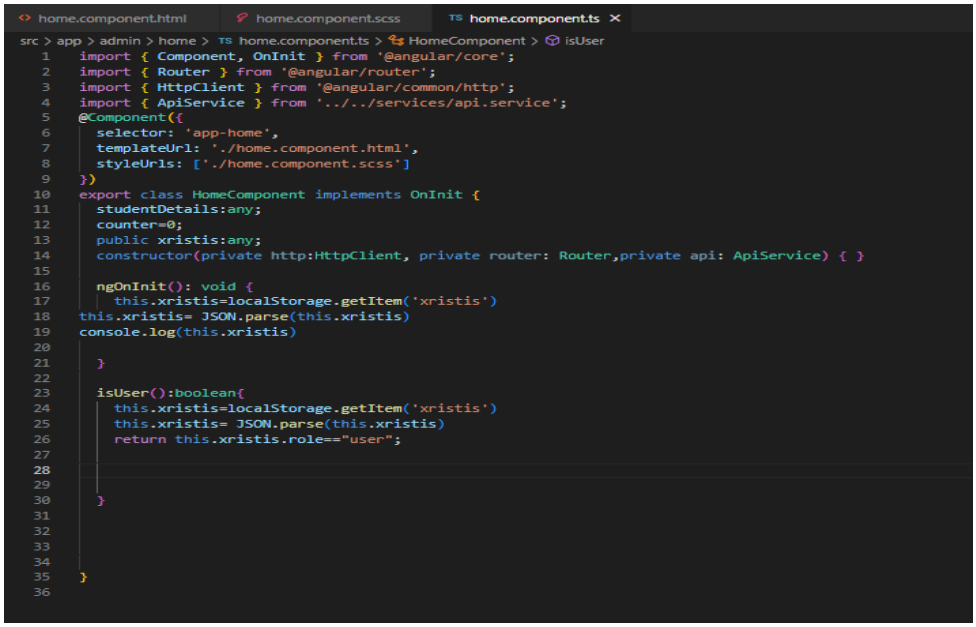


(image 17)

Στο αρχείο signup.component.html δημιουργούμε τα πεδία της φόρμας signup η οποία περιέχει επτά πεδία το name,afm,email,phone,address,password και role και ενεργοποιείται με την μέθοδο signup() που αναλύσαμε παραπάνω. Αφού δημιουργήσαμε τα πεδία, έπειτα δημιουργούμε ένα button με το οποίο συνδεόμαστε στη φόρμα login.

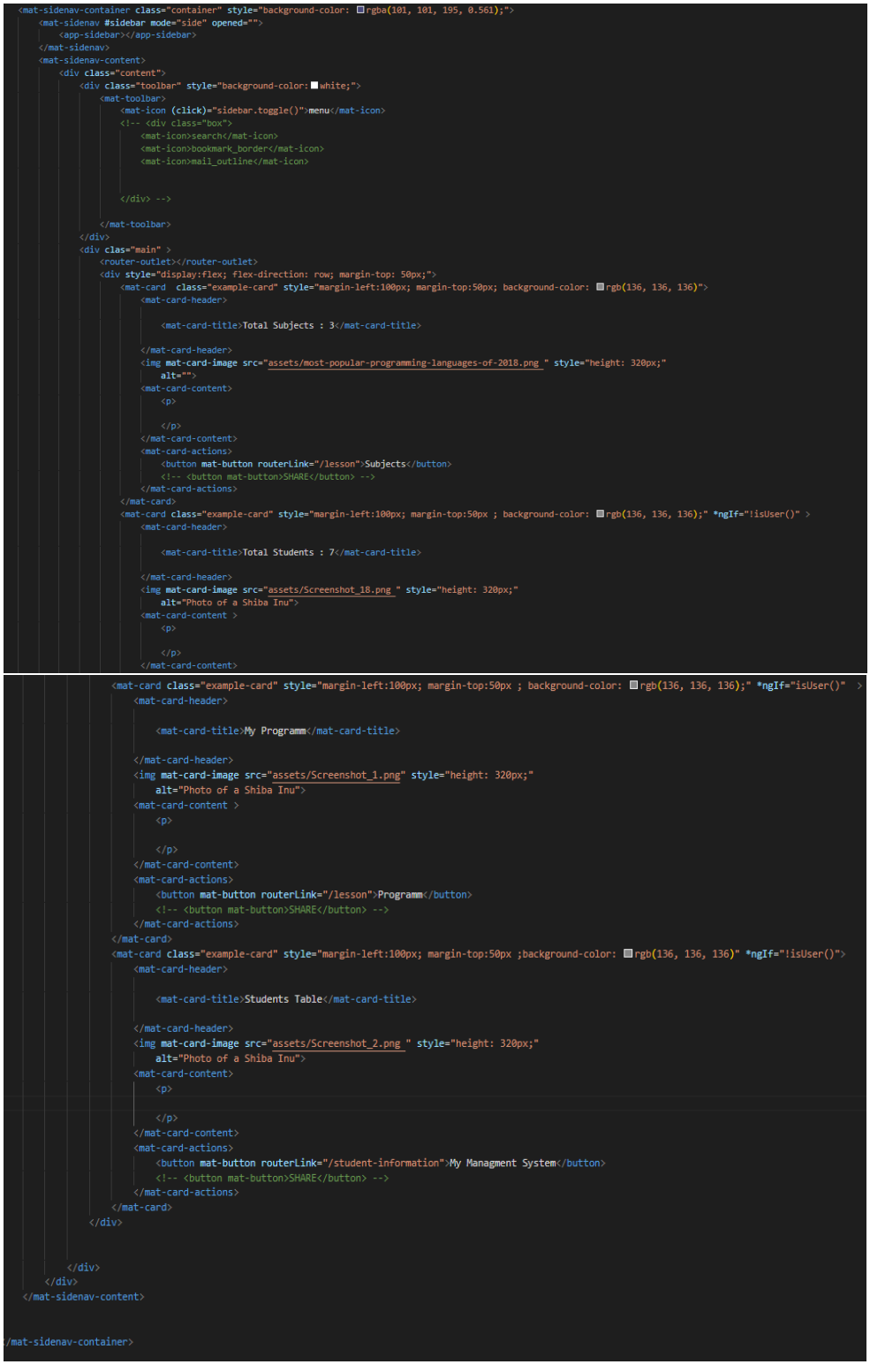
Home page Component

Αφού ολοκληρωθεί η διαδικασία της εγγραφής και την σύνδεσης πρέπει να δημιουργήσουμε μια σελίδα στην οποία θα μεταβαίνει ο χρήστης μετά την σύνδεση. Αυτή η σελίδα θα είναι η home page και αναλόγως αν ο χρήστης είναι φοιτητής ή καθηγητής θα υποστηρίζει συγκεκριμένο μενού.



(image18)

Δηλώνοντας το public xristis με τύπο any, το χρησιμοποιούμε για να αποθηκεύσουμε όλες τις τιμές που έχει πάρει το κλειδί ‘xristis’ και στην συνέχεια επιστρέφουμε σε καθαρή τιμή την συμβολοσειρά στο this.xristis. Στη γραμμή 23 δημιουργούμε μια μέθοδο isUser(), την οποία θα την χρησιμοποιήσουμε όταν ο χρήστης είναι φοιτητής να αλλάζει το menu της home page, ανάλογα με τις δυνατότητες που έχει ο κάθε χρήστης.



(image19)

Αναλύοντας το homepage.component.html στην γραμμή 2-4 εισάγουμε το <app-sidebar> την πλαϊνή μπάρα, η οποία όπως θα δούμε παρακάτω περιέχει το μενού στο οποίο μπορεί να κινηθεί ο χρήστης ανάλογα με τον ρόλο που έχει στην πλατφόρμα. Στην συνέχεια, παρατηρούμε ότι δημιουργείται ένα μενού που περιέχει τρεις κάρτες σαν επιλογή όταν ο χρήστης είναι admin και δύο όταν ο χρήστης είναι user, αυτό μπορούμε να το υλοποιήσουμε με την δυνατότητα που μας δίνει η ngIf.

Sidebar Component

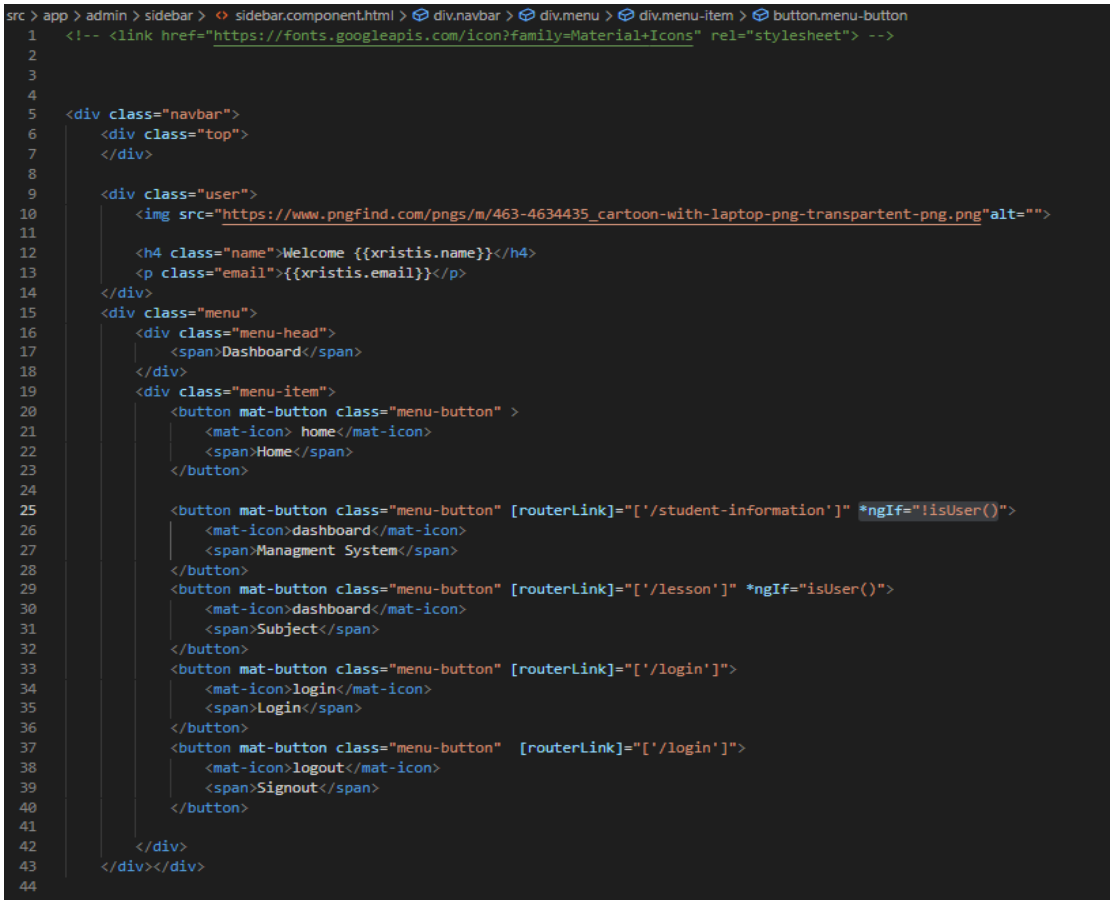
Όπως αναφέραμε παραπάνω το sidebar δημιουργήθηκε για να μας βοηθήσει στην καλύτερη λειτουργία του homepage μας, καθώς δίνει την δυνατότητα να ορίσουμε με πιο όμορφο τρόπο ποιές ενέργειες μπορεί να κάνει ο χρήστης (φοιτητής) και ποιες ο admin (καθηγητής), βλέποντας τον κώδικα από το sidebar.component.ts μπορούμε να εξηγήσουμε κάποιες βασικές ενέργειες.

src > app > admin > sidebar > TS sidebar.component.ts > SidebarComponent

```
1   import { Component, OnInit } from '@angular/core';
2   import { UrlSerializer } from '@angular/router';
3
4
5   @Component({
6     selector: 'app-sidebar',
7     templateUrl: './sidebar.component.html',
8     styleUrls: ['./sidebar.component.scss']
9   })
10  export class SidebarComponent implements OnInit {
11
12    constructor() { }
13  public xristis:any;
14    ngOnInit(): void {
15  this.xristis=localStorage.getItem('xristis')
16  this.xristis= JSON.parse(this.xristis)
17  console.log(this.xristis)
18    }
19
20    isUser():boolean{
21      this.xristis=localStorage.getItem('xristis')
22      this.xristis= JSON.parse(this.xristis)
23      return this.xristis.role=="user";
24
25
26
27    }
28
29  }
```

(image20)

Δηλώνοντας το public xristis με τύπο any, τον χρησιμοποιούμε για να αποθηκεύσουμε όλες τις τιμές που έχει πάρει το κλειδί ‘xristis’ και στην συνέχεια επιστρέφουμε σε καθαρή τιμή την συμβολοσειρά στο this.xristis. Στη γραμμή 23 δημιουργούμε μια μέθοδο isUser(), την οποία θα την χρησιμοποιήσουμε όταν ο χρήστης είναι φοιτητής να αλλάζει το menu της home page, ανάλογα με τις δυνατότητες που έχει ο κάθε χρήστης.



(image21)

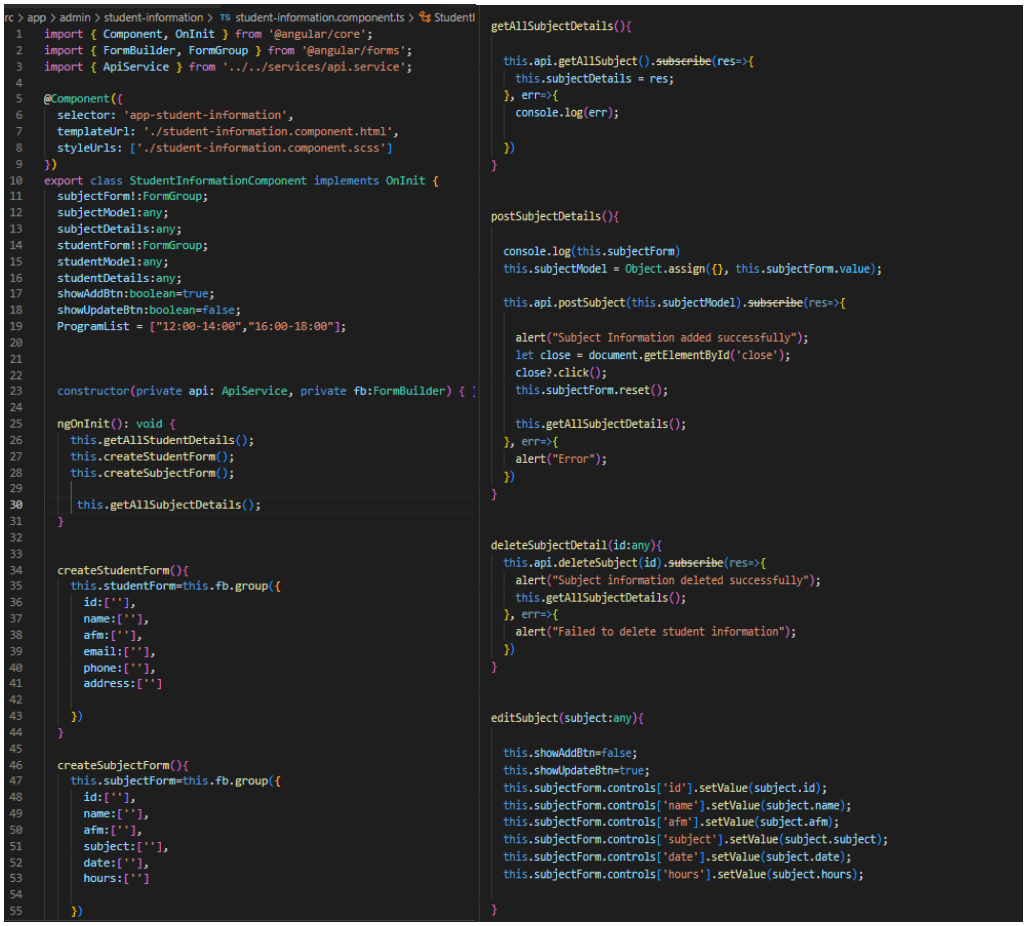
Βλέποντας το αρχείο html δημιουργούμε μια κλάση user με ένα icon user και στην συνέχεια δίνουμε στην κλάση name το όνομα του χρήστη που είναι συνδεδεμένος. Αυτό γίνετε με το διπλό {{ xristis.name}} και αντίστοιχα για το email του.

Μέσα στην κλάση menu δίνουμε την δυνατότητα με ένα ngIf αναλόγως τον user, να μπορεί να βλέπει άλλη κατηγορία. Για παράδειγμα όπως φαίνεται και στον κώδικα αν ο χρήστης είναι admin έχει δυνατότητα πρόσβασης στο system management μενού, ενώ δεν έχει πρόσβαση στο μενού subject εκεί έχει πρόσβαση μόνο ο φοιτητής, ο οποίος δεν έχει πρόσβαση στο system management.

Η εντολή routerLink δίνει την δυνατότητα στο button που θα επιλέξουμε να μας συνδέσει με την αντίστοιχη σελίδα. Για παράδειγμα [routerLink]=[/student-information] μας οδηγεί στην σελίδα student information που θα δούμε στην συνέχεια.

Student-information Component

Στο component αυτό πέρα από τα αρχεία css, html, ts θα δούμε και το αρχείο api.services με τα οποία μπορούμε να πάρουμε δεδομένα, να στείλουμε δεδομένα, να διαγράψουμε και να επεξεργαστούμε δεδομένα από την βάση μας.



```
rc > app > admin > student-information > TS student-information.component.ts > StudentI
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { ApiService } from '../../services/api.service';

@Component({
  selector: 'app-student-information',
  templateUrl: './student-information.component.html',
  styleUrls: ['./student-information.component.scss']
})
export class StudentInformationComponent implements OnInit {
  subjectForm!:FormGroup;
  subjectModel:any;
  subjectDetails:any;
  studentForm!:FormGroup;
  studentModel:any;
  studentDetails:any;
  showAddBtn:boolean=true;
  showUpdateBtn:boolean=false;
  ProgramList = ["12:00-14:00","16:00-18:00"];



  constructor(private api: ApiService, private fb:FormBuilder) { }

  ngOnInit(): void {
    this.getAllStudentDetails();
    this.createStudentForm();
    this.createSubjectForm();

    this.getAllSubjectDetails();
  }


  createStudentForm(){
    this.studentForm=this.fb.group({
      id:[''],
      name:[''],
      afm:[''],
      email:[''],
      phone:[''],
      address:['']

    })
  }

  createSubjectForm(){
    this.subjectForm=this.fb.group({
      id:[''],
      name:[''],
      afm:[''],
      subject:[''],
      date:[''],
      hours:['']

    })
```

```
getAllSubjectDetails(){

  this.api.getAllSubject().subscribe(res=>{
    this.subjectDetails = res;
  }, err=>{
    console.log(err);

  })
}


postSubjectDetails(){

  console.log(this.subjectForm)
  this.subjectModel = Object.assign({}, this.subjectForm.value);

  this.api.postSubject(this.subjectModel).subscribe(res=>{

    alert("Subject Information added successfully");
    let close = document.getElementById('close');
    close?.click();
    this.subjectForm.reset();

    this.getAllSubjectDetails();
  }, err=>{
    alert("Error");
  })
}


deleteSubjectDetail(id:any){
  this.api.deleteSubject(id).subscribe(res=>{
    alert("Subject information deleted successfully");
    this.getAllSubjectDetails();
  }, err=>{
    alert("Failed to delete student information");
  })
}


editSubject(subject:any){

  this.showAddBtn=false;
  this.showUpdateBtn=true;
  this.subjectForm.controls['id'].setValue(subject.id);
  this.subjectForm.controls['name'].setValue(subject.name);
  this.subjectForm.controls['afm'].setValue(subject.afm);
  this.subjectForm.controls['subject'].setValue(subject.subject);
  this.subjectForm.controls['date'].setValue(subject.date);
  this.subjectForm.controls['hours'].setValue(subject.hours);

}
```

(image 22)

```
updateSubjectDetails(){
  this.subjectModel = Object.assign({}, this.subjectForm.value);

  this.api.updateSubject(this.subjectModel, this.subjectModel.id).subscribe(res=>{
    alert("Subject information updated successfully");
    let close = document.getElementById('close');
    close?.click();
    this.getAllSubjectDetails();
    this.subjectForm.reset();
    this.subjectModel={};
  }, err=>{
    alert("Error in updating subject information");
  })

}

resetSubject(){
  this.subjectForm.reset();
  this.subjectModel={};
}


getAllStudentDetails(){
  this.api.getAllStudent().subscribe(res=>{
    this.studentDetails = res;
    // console.log(this.studentDetails)
  }, err=>{
    console.log(err);

  })
}

onAddClick(){
  this.showAddBtn=true;
  this.showUpdateBtn=false;
}
postStudentDetails(){
  this.studentModel = Object.assign({}, this.studentForm.value); // antigarfei twn value sto studentmodel


  this.api.postStudent(this.studentModel).subscribe(res=>{ //eggrafh tou neou xrhsth sthn res
    // console.log(res)

    alert("Student Information added successfully");
    let close = document.getElementById('close');
    close?.click();
    this.studentForm.reset();

    this.getAllStudentDetails();
  }, err=>{
    alert("Error");
  })
}
```

(image 23)

Στο σημείο αυτό θα προσπαθήσουμε να αναλύσουμε όσο γίνεται πιο συγκεκριμένα κάθε τμήμα του παραπάνω κώδικα

Αρχικά δηλώνουμε τους μαθητές και τα μαθήματα καθώς και τις αντίστοιχες φόρμες τους. Επιπλέον δηλώνω έναν πίνακα programList με συγκεκριμένες ώρες που θα τις χρησιμοποιήσω για το πρόγραμμα των μαθημάτων.

Μέσα στην ngOnInit δηλώνω τις μεθόδους που θα δημιουργήσω πιο κάτω. Ξεκινώντας από την γραμμή 34 δημιουργούμε τη φόρμα του φοιτητή και τη φόρμα του μαθήματος με τον ίδιο τρόπο όπως τη φόρμα του login και signup.

- Μέθοδος **getAllSubjectDetails:** στην μέθοδο αυτή ζητάμε να γίνει εγγραφή όλων των μαθημάτων που έχουν καταχωρηθεί στην /localhost:3000/subjects ή αλλιώς subject.db στην **res** και από την res να καταχωρηθούν στην subjectDetails, σε περίπτωση αποτυχίας να εμφανιστεί μήνυμα λάθους.
- Μέθοδος **postSubjectDetails:** γράφοντας τη μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της subjectForm σε ένα subjectModel το οποίο θα είναι μια αντίστοιχη φόρμα. Αυτό γίνεται γιατί για να κάνουμε post στην βάση μας πρέπει να ανοίγει μια νέα φόρμα όπου θα συμπληρώνουμε τα στοιχεία μας. Ύστερα κάνουμε εγγραφή τις τιμές αυτές στο res και εμφανίζουμε μήνυμα επιτυχίας και κλείνει η φόρμα και στην συνέχεια με την .reset() καθαρίζεται για την επόμενη εγγραφή.
- Μέθοδος **deleteSubjectDetails:** από το όνομα της μεθόδου καταλαβαίνουμε ότι η μέθοδος αυτή χρησιμοποιείται για διαγραφή κάποιου μαθήματος, για τον λόγο αυτό εισάγουμε (id:any). Με το api.deleteSubject(id) δίνουμε εντολή στο αρχείο api.services να επικοινωνήσει με την βάση μας και να διαγράψει το μάθημα με το συγκεκριμένο id και κάνουμε εγγραφή του αποτελέσματος αυτού στο res. Στην περίπτωση που υπάρχει κάποιο λάθος εμφανίζεται κατάλληλο μήνυμα.
- Μέθοδος **editSubject:** με την μέθοδο αυτή δεν καλούμε κάποιο api το μόνο που κάνουμε είναι να μπορούμε να αλλάξουμε τις τιμές της φόρμας.
- Μέθοδος **updateSubjectDetails:** με τη μέθοδο αυτή μπορούμε να ενημερώνουμε κάποια από τις τιμές της subjectModel για συγκεκριμένη τιμή πεδίου πχ name. Γράφοντας τη μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της subjectForm σε ένα subjectModel, στην συνέχεια μέσω του api.updateSubject από το αρχείο api.services ενημερώνουμε το συγκεκριμένο πεδίο της subjectModel και κλείνουμε την φόρμα. Έτσι ενημερώνετε η βάση μας.

Η ίδια ακριβώς διαδικασία γίνεται και για το subject

- Μέθοδος **getAllStudentsDetails:** στη μέθοδο αυτή ζητάμε να γίνει εγγραφή όλων των μαθημάτων που έχουν καταχωρηθεί στην /localhost:3000/students ή αλλιώς students.db στην **res** και από την res να καταχωρηθούν στην studentsDetails, σε περίπτωση αποτυχίας να εμφανιστεί μήνυμα λάθους.
- Μέθοδος **postStudentsDetails:** γράφοντας της μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της studentForm σε ένα studentModel το οποίο θα είναι μια αντίστοιχη φόρμα. Αυτό γίνεται γιατί για να κάνουμε post στην βάση μας πρέπει να ανοίγει μια νέα φόρμα όπου θα συμπληρώνουμε τα στοιχεία μας. Ύστερα κάνουμε εγγραφή τις τιμές αυτές στο res και

εμφανίζουμε μήνυμα επιτυχίας και κλείνει η φόρμα και στη συνέχεια με την .reset() καθαρίζεται για την επόμενη εγγραφή.

- Μέθοδος **deleteStudentDetails:** από το όνομα της μεθόδου καταλαβαίνουμε ότι η μέθοδος αυτή χρησιμοποιείται για διαγραφή κάποιου μαθητή, για τον λόγο αυτό εισάγουμε (id:any). Με το api.deleteStudent(id) δίνουμε εντολή στο αρχείο api.services να επικοινωνήσει με τη βάση μας και να διαγράψει το μάθημα με το συγκεκριμένο id και κάνουμε εγγραφή του αποτελέσματος αυτού στο res. Στην περίπτωση που υπάρχει κάποιο λάθος εμφανίζεται κατάλληλο μήνυμα.
- Μέθοδος **editStudent:** με τη μέθοδο αυτή δεν καλούμε κάποιο api το μόνο που κάνουμε είναι να μπορούμε να αλλάξουμε τις τιμές της φόρμας.
- Μέθοδος **updateStudentDetails:** με τη μέθοδο αυτή μπορούμε να ενημερώνουμε κάποια από τις τιμές της studentModel για συγκεκριμένη τιμή πεδίου πχ name. Γράφοντας τη μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της studentForm σε ένα studentModel, στη συνέχεια μέσω του api.updateStudent από το αρχείο api.services ενημερώνουμε το συγκεκριμένο πεδίο της studentModel και κλείνουμε τη φόρμα, έτσι ενημερώνετε η βάση μας.

API SERVICES

c > app > services > TS api.service.ts > ApiService

```
import { HttpClient } from '@angular/common/http';
import { Injectable } from '@angular/core';
import { environment } from 'src/environments/environment';
import { map, Observable } from 'rxjs';
import { MapType } from '@angular/compiler';

@Injectable({
  providedIn: 'root'
})
export class ApiService {

  constructor(private http:HttpClient) { }
  baseUrl : string = environment.baseUrl;

  postStudent(data:any){
    return this.http.post<any>(this.baseUrl+'/students', data);
  }

  getAllStudent(){
    return this.http.get<any>(this.baseUrl+'/students');
  }

  deleteStudent(id:any){
    return this.http.delete<any>(this.baseUrl+'/students/'+ id);
  }

  updateStudent(data:any, id:number){
    return this.http.put<any>(this.baseUrl+'/students/'+id, data);
  }
```

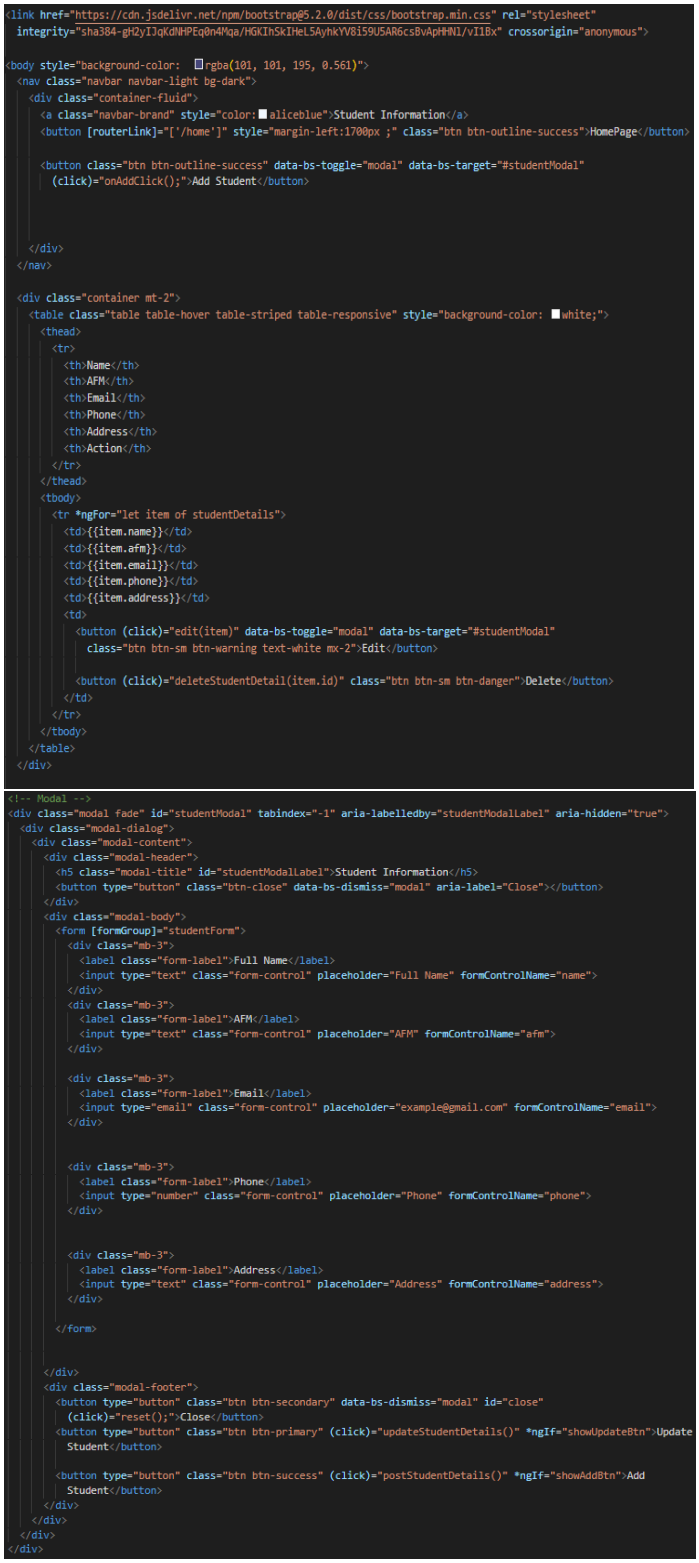
(image 24)

```
  //subject

  postSubject(data:any){
    return this.http.post<any>(this.baseUrl+'/subjects', data);
  }


  getAllSubject(){
    return this.http.get<any>(this.baseUrl+'/subjects/');
  }

  deleteSubject(id:any){
    return this.http.delete<any>(this.baseUrl+'/subjects/'+ id);
  }

  updateSubject(data:any, id:number){
    return this.http.put<any>(this.baseUrl+'/subjects/'+id, data);
  }


getSubject(name:any)
{
  return this.http.get<any>(this.baseUrl+'/subjects?name='+name);
}



}
```

(image 25)

Το αρχείο api.services δίνει τη δυνατότητα στο χρήστη να πάρει, να δώσει, να επεξεργαστεί και να διαγράψει όποιο μάθημα ή μαθητή θέλει από την βάση subject.db και students.db. Μέσω των μεθόδων getSubject, που τραβάει το μάθημα από την βάση και το εμφανίζει στο frontend. Με τη μέθοδο post, η βάση μας τραβάει τις τιμές των πεδίων και τις αποθηκεύει στη βάση. Με τη μέθοδο delete, η βάση μας τραβάει τις τιμές των πεδίων και τις αποθηκεύει στη βάση. Ενώ με τη μέθοδο update ενημερώνει τη βάση για το συγκεκριμένο id στο οποίο έγινε αλλαγή της τιμής.

Δημιουργώντας το student-information.component.html να δημιουργήσουμε δύο πίνακες ο πρώτος θα αφορά τα στοιχεία του φοιτητή και ο δεύτερος το πρόγραμμα του φοιτητή με βάση το μάθημα που έχει επιλέξει. Βλέποντας το αρχείο html στις παρακάτω εικόνες θα αναλύσουμε την ανάπτυξη του πίνακα student καθώς και το modal το οποίο δημιουργήσαμε για να εκτελέσουμε τις εντολές της εγγραφής και ενημέρωσης.



(image 26)

Αναλύοντας τον html κώδικα του student-information, στην αρχή δημιουργούμε δύο buttons μέσα στο nav-bar, έτσι ώστε να μας δρομολογούν στην αρχική σελίδα και στην προσθήκη νέου φοιτητή. Ο πίνακας ο οποίος αναφέρεται στον φοιτητή θα έχει τα πεδία name , afm , email, phone , address και τα actions buttons. Βάζοντας μια επαναληπτική

ngFor για κάθε χρήστη μειώνουμε τις άσκοπες επαναλήψεις για κάθε νέο χρήστη σε ξεχωριστό πίνακα.

Τέλος ο πίνακας περιέχει δύο buttons το edit και το delete τα οποία συνδέονται με το #studentModal. Το modal αυτό ανοίγει κάνοντας κλικ στο edit και στο add student, υπάρχει μια φόρμα η studentForm, την οποία έχουμε δηλώσει και στο αρχείο ts με τα συγκεκριμένα πεδία που θα πρέπει να συμπληρωθούν και έπειτα να αποθηκευτούν στον πίνακα. Κάνοντας κλικ στο κουμπί close η φόρμα εκτελεί τη μέθοδο reset() και καθαρίζει με σκοπό στην επόμενη νέα εγγραφή μαθητή η φόρμα να είναι καθαρή.

Η ίδια ακριβώς διαδικασία εκτελείτε και για τα μαθήματα ας δούμε παρακάτω.

```
<nav class="navbar navbar-light bg-dark">

  <button class="btn btn-outline-success" data-bs-toggle="modal" data-bs-target="#subjectModal"
    style="margin-left:1990px" (click)="onAddClick();">Add Programm</button>
</nav>

<div class="container mt-5 ">

  <div class="row">

    <div class="container mt-2">
      <table class="table table-hover table-striped table-responsive" style="background-color: white;">
        <thead>
          <tr>
            <th>name </th>
            <th>AFM </th>
            <th>subject</th>
            <th>Date</th>
            <th>Hours</th>
            <th>Action</th>

          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let item of subjectDetails">
            <td>{{item.name}}</td>
            <td>{{item.afm}}</td>

            <td>{{item.subject}}</td>
            <td>{{item.date|date}}</td>
            <td>{{item.hours}}</td>


            <td>

              <button (click)="editSubject(item)" data-bs-toggle="modal" data-bs-target="#subjectModal"
                class="btn btn-sm btn-warning text-white mx-2">Edit</button>

              <button (click)="deleteSubjectDetail(item.id)" class="btn btn-sm btn-danger">Delete</button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

```
<div class="modal fade" id="subjectModal" tabindex="-1" aria-labelledby="subjectModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="subjectModalLabel">Subject Information</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <form [formGroup]="subjectForm">

          <div class="mb-3">
            <mat-form-field appearance="outline">
              <mat-label>subject name</mat-label>
              <select formControlName="subject" matNativeControl required>

                <option value="java">java</option>
                <option value="angular">angular</option>
                <option value="c sharp">c sharp</option>

              </select>

            </mat-form-field>
          </div>

          <div class="mb-3">

            <mat-form-field appearance="outline">
              <mat-label>Choose a date</mat-label>
              <input formControlName="date" matInput [matDatepicker]="picker">
              <mat-hint>MM/DD/YYYY</mat-hint>
              <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
              <mat-datepicker #picker></mat-datepicker>
            </mat-form-field>

            <mat-label for="">
              <h1>choose your hours</h1>
            </mat-label>
            <mat-radio-group formControlName="hours" class="example-radio-group">
              <mat-radio-button class="example-radio-group" *ngFor="let hours of ProgramList" [value]="hours">
                {{hours}}
              </mat-radio-button>
            </mat-radio-group>
          </div>

        </form>

        <div class="modal-footer">
          <button type="button" class="btn btn-secondary" data-bs-dismiss="modal" id="close"
            (click)="resetSubject();">Close</button>
          <button type="button" class="btn btn-primary" (click)="updateSubjectDetails()" *ngIf="showUpdateBtn">Update
            Subject</button>

          <button type="button" class="btn btn-success" (click)="postSubjectDetails()" *ngIf="showAddBtn">Add
            Subject</button>
        </div>
      </div>
    </div>
  </div>
</div>
```

(image 27)

Αναλύοντας τον html κώδικα του student-information στην αρχή δημιουργούμε δύο buttons μέσα στο nav-bar έτσι ώστε να μας δρομολογούν στην αρχική σελίδα και στην προσθήκη νέου μαθήματος. Ο πίνακας ο οποίος αναφέρεται στον φοιτητή θα έχει τα subject name , date , hours και τα actions buttons. Βάζοντας μια επαναληπτική ngFor για κάθε χρήστη μειώνουμε τις άσκοπες επαναλήψεις για κάθε νέο subject σε ξεχωριστό πίνακα.

Τέλος ο πίνακας περιέχει δύο buttons το edit και το delete τα οποία συνδέονται με το #subjectModal. Το modal αυτό ανοίγει κάνοντας κλικ στο edit και στο add subject , υπάρχει μια φόρμα η subjectForm την οποία έχουμε δηλώσει και στο αρχείο ts με τα συγκεκριμένα πεδία που θα πρέπει να συμπληρωθούν και έπειτα να αποθηκευτούν στον πίνακα. Κάνοντας κλικ στο κουμπί close η φόρμα εκτελεί την μέθοδο reset() και καθαρίζει με σκοπό στην επόμενη νέα εγγραφή subject η φόρμα να είναι καθαρή.

Lesson Component

Αρχικά δηλώνουμε τα μαθήματα καθώς και την αντίστοιχη φόρμα, επιπλέον δηλώνω έναν πίνακα programList με συγκεκριμένες ώρες που θα τις χρησιμοποιήσω για το πρόγραμμα των μαθημάτων.

Μέσα στην ngOnInit δηλώνω τις μεθόδους που θα δημιουργήσω πιο κάτω. Ξεκινώντας από την γραμμή 29 δημιουργούμε τη φόρμα του φοιτητή και τη φόρμα του μαθήματος με τον ίδιο τρόπο όπως τη φόρμα του login και signup.

- Μέθοδος **getSubjectDetails:** στη μέθοδο αυτή ζητάμε να γίνει εγγραφεί όλων των μαθημάτων που έχουν καταχωρηθεί στην /localhost:3000/subjects ή αλλιώς subject.db στην **res** και από την res να καταχωρηθούν στην subjectDetails, σε περίπτωση αποτυχίας να εμφανιστεί μήνυμα λάθους.
- Μέθοδος **postSubjectDetails:** γράφοντας της μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της subjectForm σε ένα subjectModel το οποίο θα είναι μια αντίστοιχη φόρμα αυτό γίνεται γιατί για να κάνουμε post στη βάση μας πρέπει να ανοίγει μια νέα φόρμα όπου θα συμπληρώνουμε τα στοιχεία μας. Ύστερα κάνουμε εγγραφή τις τιμές αυτές στο res και εμφανίζουμε μήνυμα επιτυχίας και κλείνει η φόρμα και στη συνέχεια με την .reset() καθαρίζεται για την επόμενη εγγραφή.
- Μέθοδος **deleteSubjectDetails:** από το όνομα της μεθόδου καταλαβαίνουμε ότι η μέθοδος αυτή χρησιμοποιείται για διαγραφή κάποιου μαθήματος, για τον λόγο αυτό εισάγουμε (id:any). Με το api.deleteSubject(id) δίνουμε εντολή στο αρχείο api.services να επικοινωνήσει με τη βάση μας και να διαγράψει το μάθημα με το συγκεκριμένο id και κάνουμε εγγραφή του αποτελέσματος αυτού στο res. Στην περίπτωση που υπάρχει κάποιο λάθος εμφανίζεται κατάλληλο μήνυμα.
- Μέθοδος **editSubject:** με τη μέθοδο αυτή δεν καλούμε κάποιο api το μόνο που κάνουμε είναι να μπορούμε να αλλάξουμε τις τιμές της φόρμας.
- Μέθοδος **updateSubjectDetails:** με τη μέθοδο αυτή μπορούμε να ενημερώνουμε κάποια από τις τιμές της subjectModel για συγκεκριμένη τιμή πεδίου πχ name. Γράφοντας της μέθοδο αυτή χρησιμοποιούμε το object.assign({})) το οποίο δίνει την δυνατότητα να αντιγράψουμε όλες τις τιμές της subjectForm σε ένα subjectModel, στη συνέχεια μέσω του api.updateSubject από το αρχείο api.services.
- Η μέθοδος reset() χρησιμοποιείται όπως αναφέραμε και παραπάνω για να καθαρίζει τα πεδία της φόρμας μας όταν ο χρήστης ολοκληρώνει είτε την εγγραφή είτε την αλλαγή

στοιχείων του, με σκοπό ο νέος χρήστης που θα ανοίξει την ίδια φόρμα να έχει καθαρά τα πεδία ώστε να μπορεί να περάσει τα δικά του στοιχεία.

```
src > app > lesson > TS lesson.component.ts > LessonComponent > readonly
import { validateVerticalPosition } from '@angular/cdk/overlay';
import { Component, OnInit } from '@angular/core';
import { FormBuilder, FormGroup } from '@angular/forms';
import { ApiService } from '../services/api.service';
import { ActivatedRoute } from '@angular/router';

@Component({
  selector: 'app-lesson',
  templateUrl: './lesson.component.html',
  styleUrls: ['./lesson.component.scss']
})
export class LessonComponent implements OnInit {

  ProgramList = ["12:00-14:00","16:00-18:00"];
  subjectForm!:FormGroup;
  subjectModel:any;
  subjectDetails:any;
  showAddBtn:boolean=true;
  showUpdateBtn:boolean=false;
  disabled:boolean=false;
  readonly: boolean=true;
id:any
data:any
  route: any;

  constructor(private api: ApiService, private fb:FormBuilder, private activ:ActivatedRoute) { }
  public xristis:any;

  ngOnInit(): void {
    this.xristis=localStorage.getItem('xristis')
    this.xristis= JSON.parse(this.xristis)
console.log(this.xristis)
  //  this.id=this.activ1.snapshot.params['id'];
   this.getSubjectDetails();
    //  this.getAllSubjectDetails();
    this.createSubjectForm();



  }









  createSubjectForm(){
    this.subjectForm=this.fb.group({
      id:[''],
      name:[''],
      afm:[''],
      subject:[''],
      date:[''],
      hours:['']

    })
  }
```

```
getSubjectDetails(){

  this.api.getSubject(this.xristis.name ).subscribe(res=>{


   this.subjectDetails=res;


  }, err=>{
    console.log(err.message);

  })
}

onAddClick(){
  this.showAddBtn=true;
  this.showUpdateBtn=false;
}
postSubjectDetails(){
  console.log(this.subjectForm)
  this.subjectForm.patchValue({

      name:this.xristis.name,
      afm:this.xristis.afm


  })
  console.log(this.subjectForm)
  this.subjectModel = Object.assign({}, this.subjectForm.value);

  this.api.postSubject(this.subjectModel).subscribe(res=>{

    alert("Subject Information added successfully");
    let close = document.getElementById('close');
    close?.click();
    this.subjectForm.reset();

    this.getSubjectDetails();
  }, err=>{
    alert("Error");
  })
}

deleteSubjectDetail(id:any){
  this.api.deleteSubject(id).subscribe(res=>{
    alert("Subject information deleted successfully");
    this.getSubjectDetails();
  }, err=>{
    alert("Failed to delete student information");
  })
}
```

```
edit(subject:any){

  this.showAddBtn=false;
  this.showUpdateBtn=true;
  this.subjectForm.controls['id'].setValue(subject.id);
  this.subjectForm.controls['name'].setValue(subject.name);
  this.subjectForm.controls['afm'].setValue(subject.afm);
  this.subjectForm.controls['subject'].setValue(subject.subject);
  this.subjectForm.controls['date'].setValue(subject.date);
  this.subjectForm.controls['hours'].setValue(subject.hours);

}

updateSubjectDetails(){
  this.subjectModel = Object.assign({}, this.subjectForm.value);

  this.api.updateSubject(this.subjectModel, this.subjectModel.id).subscribe(res=>{
    alert("Subject information updated successfully");
    let close = document.getElementById('close');
    close?.click();
    this.getSubjectDetails();
    this.subjectForm.reset();
    this.subjectModel={};
  }, err=>{
    alert("Error in updating subject information");
  })

}

reset(){
  this.subjectForm.reset();
  this.subjectModel={};
}


isUser():boolean{
  this.xristis=localStorage.getItem('xristis')
  this.xristis= JSON.parse(this.xristis)
  return this.xristis.role=="user";



}
}
```

(image 28)

Αναλύοντας τον html κώδικα του lesson.component.html στη αρχή χρησιμοποιούμε μια ngIf και ορίζουμε τα σημεία τα οποία μπορεί να βλέπει ένας φοιτητής ή ένας καθηγητής

,ορίζουμε δύο buttons μέσα στο nav-bar έτσι ώστε να μας δρομολογούν στην αρχική σελίδα και στην προσθήκη νέου μαθήματος. Πριν δημιουργήσουμε τον πίνακα που αφορά τα μαθήματα , δημιουργούμε ένα μενού από κάρτες οι οποίες σχετίζονται με τα μαθήματα που μπορεί να επιλέξει ο χρήστης. Αυτές οι κάρτες περιέχουν ένα σύνολο ερωτήσεων που θα αναλύσουμε αργότερα. Στην συνέχεια δημιουργούμε τον πίνακα μαθημάτων ο οποίος αναφέρεται στον φοιτητή θα έχει τα subject name , date , hours και τα actions buttons. Βάζοντας μια επαναληπτική ngFor για κάθε χρήστη μειώνουμε τις άσκοπες επαναλήψεις για κάθε νέο subject σε ξεχωριστό πίνακα.

```
<div class="p-5 bg-dark text-white text-center" *ngIf="isUser()">
  <h1>My Programm</h1>
  <h4 class="name">{{xristis.name}}</h4>


</div>

<nav class="navbar navbar-expand-sm bg-dark navbar-dark">
  <div class="container-fluid">
    <ul class="navbar-nav">
      <li class="nav-item">
        <a class="nav-link active" href="home">Home</a>
      </li>
      <li class="nav-item">
        <a class="nav-link" href="#">Link</a>
      </li>

    </ul>
  </div>
</nav>

<div class="container mt-5">
  <div class="row" style="margin-bottom:15px;">
    <h2>Your subjects in this exam</h2>
    <div class="col-sm-4">
      <div class="card" style="width: 18rem;">
        <img src="assets/java-logo-vert-blk.png" class="card-img-top" alt="...">
        <div class="card-body">
          <h5 class="card-title">JAVA</h5>
          <p class="card-text">First Exam</p>
          <a href="quiz" class="btn btn-primary">Go question</a>
        </div>
      </div>
    </div>
    <div class="col-sm-4">
      <div class="card" style="width: 18rem;">
        <img src="assets/Python.svg.png" class="card-img-top" alt="...">
        <div class="card-body">
          <h5 class="card-title">Python</h5>
          <p class="card-text">Fist Exams</p>
          <a href="python-quiz" class="btn btn-primary">Go question</a>

        </div>
      </div>
    </div>
    <div class="col-sm-4">
      <div class="card" style="width: 18rem;">
        <img src="assets/unnamed.png" class="card-img-top" alt="...">
        <div class="card-body">
          <h5 class="card-title">Pythos</h5>
          <p class="card-text">First Exams</p>
          <a href="csharp" class="btn btn-primary">Go somewhere</a>
        </div>
      </div>
    </div>
  </div>
</div>
```

```
<div class="p-5 bg-dark text-white text-center" *ngIf="isUser()">
  <h1>you can select your program</h1>
</div>
<div class="p-5 bg-dark text-white text-center" *ngIf="!isUser()">

</div>




<div class="container mt-5 " *ngIf="isUser()" >
  <button class="btn btn-outline-success" data-bs-toggle="modal" data-bs-target="#subjectModal"
    (click)="onAddClick();">Add Programm</button>
  <div class="row">
    <h2>Your subjects in this exam</h2>
    <div class="container mt-2">
      <table class="table table-hover table-striped table-responsive" >
        <thead>
          <tr>
            <th>Name </th>
            <th>AFM</th>
            <th>Subject</th>
            <th>Date</th>
            <th>Hours</th>
            <th>Action</th>

          </tr>
        </thead>
        <tbody>
          <tr *ngFor="let item of subjectDetails" >
            <td>{{item.name}}</td>
            <td>{{item.afm}}</td>
            <td>{{item.subject}}</td>
            <td>{{item.date|date}}</td>
            <td>{{item.hours}}</td>


            <td>
              <!-- <button type="button" class="btn btn-success" (click)="postSubjectDetails()" *ngIf="show

              <button (click)="edit(item)" data-bs-toggle="modal" data-bs-target="#subjectModal"
                class="btn btn-sm btn-warning text-white mx-2">Edit</button>

              <button (click)="deleteSubjectDetail(item.id)" class="btn btn-sm btn-danger">Delete</button>
            </td>
          </tr>
        </tbody>
      </table>
    </div>
  </div>
</div>
```

(image 29)

Τέλος ο πίνακας περιέχει δύο buttons το edit και το delete τα οποία συνδέονται με το #subjectModal. Το modal αυτό ανοίγει κάνοντας κλικ στο edit και στο add subject , υπάρχει μια φόρμα η subjectForm την οποία έχουμε δηλώσει και στο αρχείο ts με τα συγκεκριμένα πεδία που θα πρέπει να συμπληρωθούν και έπειτα να αποθηκευτούν στον πίνακα. Κάνοντας κλικ στο κουμπί close η φόρμα εκτελεί την μέθοδο reset() και καθαρίζει με σκοπό στην επόμενη νέα εγγραφή subject η φόρμα να είναι καθαρή.

```
<div class="modal fade" id="subjectModal" tabindex="-1" aria-labelledby="subjectModalLabel" aria-hidden="true">
  <div class="modal-dialog">
    <div class="modal-content">
      <div class="modal-header">
        <h5 class="modal-title" id="subjectModalLabel">Subject Information</h5>
        <button type="button" class="btn-close" data-bs-dismiss="modal" aria-label="Close"></button>
      </div>
      <div class="modal-body">
        <form [formGroup]="subjectForm">
          <div class="mb-3">



          <div class="mb-3">
            <mat-form-field appearance="outline">
              <mat-label>subject name</mat-label>
              <select formControlName="subject" matNativeControl required>
                <!-- <mat-select formControlName="subject" placeholder="subject"> -->
                <option value="java">java</option>
                <option value="angular">angular</option>
                <option value="c sharp">c sharp</option>

              </select>

            </mat-form-field>

            <!-- <input type="text" class="form-control"  placeholder="subject" formControlName="subject"> -->
          </div>

          <div class="mb-3">

            <mat-form-field appearance="outline">
              <mat-label>Choose a date</mat-label>
              <input formControlName="date" matInput [matDatepicker]="picker">
              <mat-hint>MM/DD/YYYY</mat-hint>
              <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
              <mat-datepicker #picker></mat-datepicker>
            </mat-form-field>

            <mat-label for="">
              <h1>choose your hours</h1>
            </mat-label>
            <mat-radio-group formControlName="hours" class="example-radio-group">
              <mat-radio-button class="example-radio-group" *ngFor="let hours of ProgramList" [value]="hours">
                {{hours}}
              </mat-radio-button>
            </mat-radio-group>
          </div>


          </div>

        </form>
```

```
          <div class="mb-3">

            <mat-form-field appearance="outline">
              <mat-label>Choose a date</mat-label>
              <input formControlName="date" matInput [matDatepicker]="picker">
              <mat-hint>MM/DD/YYYY</mat-hint>
              <mat-datepicker-toggle matSuffix [for]="picker"></mat-datepicker-toggle>
              <mat-datepicker #picker></mat-datepicker>
            </mat-form-field>

            <mat-label for="">
              <h1>choose your hours</h1>
            </mat-label>
            <mat-radio-group formControlName="hours" class="example-radio-group">
              <mat-radio-button class="example-radio-group" *ngFor="let hours of ProgramList" [value]="hours">
                {{hours}}
              </mat-radio-button>
            </mat-radio-group>
          </div>


          </div>

        </form>


      </div>
      <div class="modal-footer">
        <button type="button" class="btn btn-secondary" data-bs-dismiss="modal" id="close"
          (click)="reset();">Close</button>
        <button type="button" class="btn btn-primary" (click)="updateSubjectDetails()" *ngIf="showUpdateBtn">Update
          Subject</button>

        <button type="button" class="btn btn-success" (click)="postSubjectDetails()" *ngIf="showAddBtn">Add
          Subject</button>
      </div>
    </div>
  </div>
</div>
```

(image 30)

Java Quiz Component

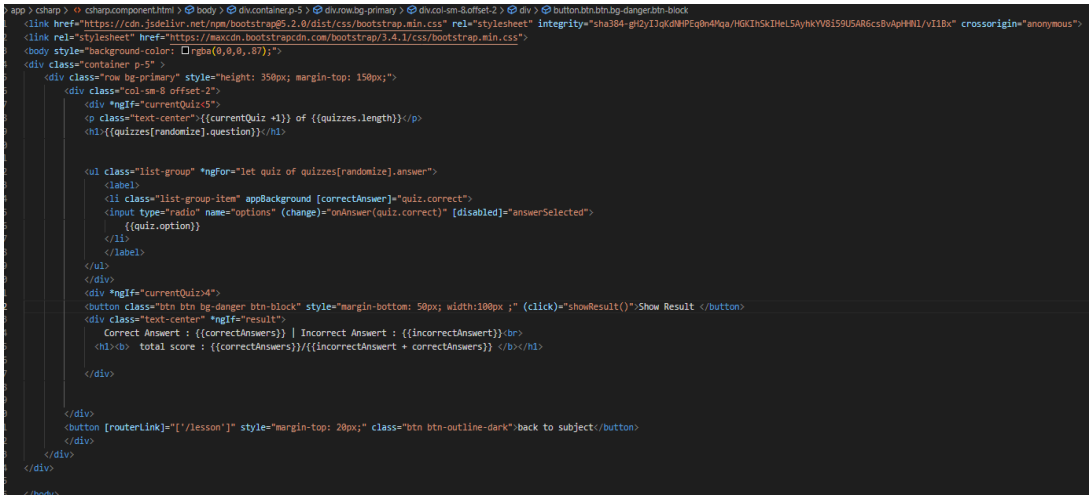
Όπως αναφέραμε και προηγουμένως στην σελίδα lesson ο φοιτητής και ο καθηγητής θα μπορούν να βλέπουν κάποιες καρτέλες με μαθήματα. Κάνοντας κλικ στις καρτέλες αυτές εμφανίζεται ανάλογα με το μάθημα και κάποιες ερωτήσεις του μαθήματος που έχει επιλέξει ο καθηγητής.

```
import { Component, OnInit } from '@angular/core';
import { JavaQuizService } from 'src/app/services/java-quiz.service'
import { Quiz } from 'src/app/quiz.model';
@Component({
  selector: 'app-quiz',
  templateUrl: './quiz.component.html',
  styleUrls: ['./quiz.component.scss']
})
export class QuizComponent implements OnInit {
quizzes: Quiz[]= [];
currentQuiz=0;
answerSelected= false;
correctAnswers=0;
incorrectAnswert=0;
result=false;
randomize:any;
  constructor(private quizService:JavaQuizService) { }

  ngOnInit(): void {
    this.quizzes=this.quizService.getQuizzes();

    this.randomize = Math.floor(Math.random()* this.quizzes.length)
  }

  onAnswer(option:boolean){
    this.answerSelected=true;
    setTimeout(()=>{
      this.currentQuiz++;
      this.randomize = Math.floor(Math.random()* this.quizzes.length

      this.answerSelected=false;

    },2000)

    if(option)
  {
    this.correctAnswers++;
  } else{
    this.incorrectAnswert++;
  }

  }

  showResult(){
    this.result=true;
  }
```

(image 31)

Αναλύοντας τον κώδικα typescript βλέπουμε ότι δηλώνουμε έναν πίνακα quizzers, δηλώνουμε ως σωστή απάντηση την τιμή true και ως λάθος την τιμή false, επιπλέον βάζουμε τυχαία επιλογή ερωτήσεων και έναν μετρητή counter.

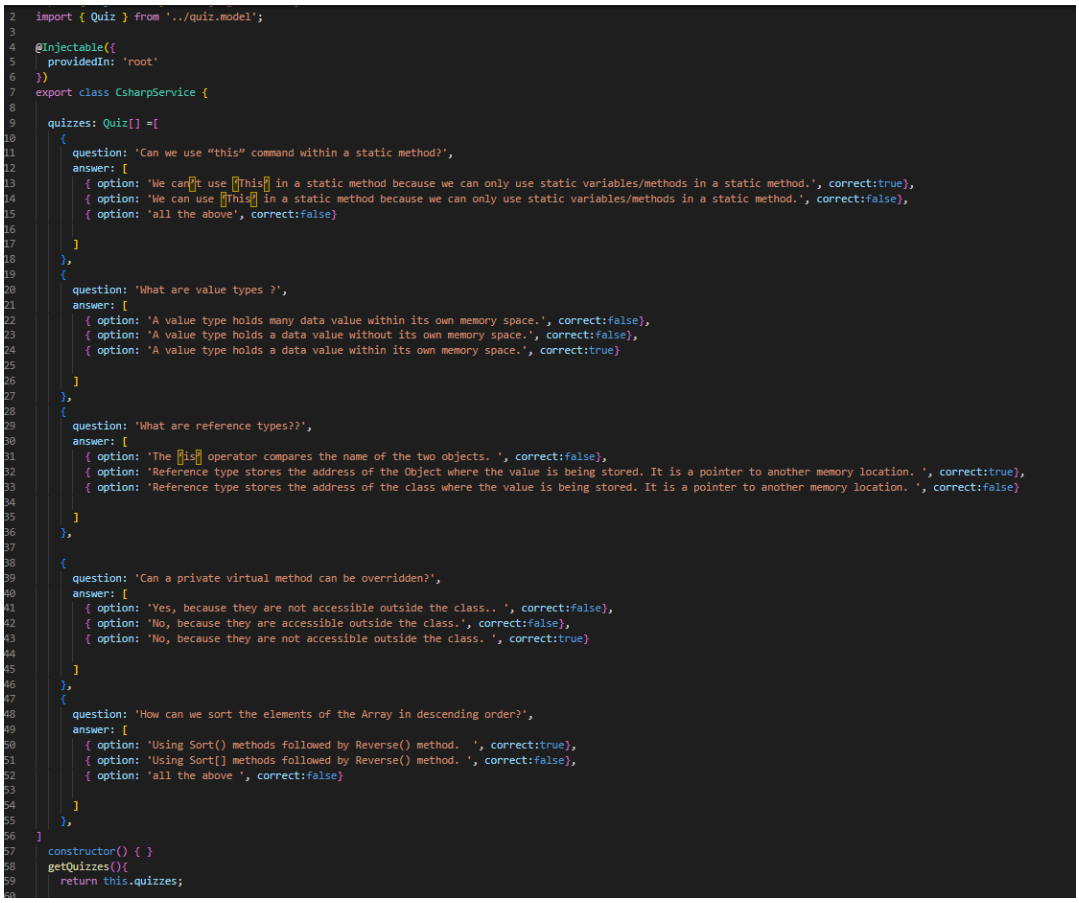
Μέσα στην ngOnit () καλούμε την μέθοδο getQuizzes από τα services. Στη μέθοδο onAnswer δηλώνουμε τη σωστή απάντηση ως true και στη συνέχεια μέσα σε μια συνάρτηση timeout δημιουργούμε μια εναλλαγή των ερωτήσεων με τυχαία σειρά κάθε φορά ανά 2 δευτερόλεπτα αφού ο χρήστης επιλέξει την απάντηση. Αν η απάντηση είναι σωστή ο μετρητής για τις σωστές απαντήσεις αυξάνεται κατά ένα σε διαφορετική περίπτωση αυξάνεται ο μετρητής για τις λάθος απαντήσεις. Τέλος με τη μέθοδο showResult καλούμε να μας δείξει το συνολικό αποτέλεσμα.



```
app > csharp > csharp.component.html > body > div.container.p-5 > div.row.bg-primary > div.col-sm-8.offset-2 > div > button.btn.btn.bg-danger.btn-block
<link href="https://cdn.jsdelivr.net/npm/bootstrap@5.2.0/dist/css/bootstrap.min.css" rel="stylesheet" integrity="sha384-gH2yIJqKdNHPEq0n4Mqa/HGKIhSkIHeL5AyhkYV8i59U5AR6csBvApHHNl/vI1Bx" crossorigin="anonymous">
<link rel="stylesheet" href="https://maxcdn.bootstrapcdn.com/bootstrap/3.4.1/css/bootstrap.min.css">
<body style="background-color: rgba(0,0,0,.87);">
<div class="container p-5" >
    <div class="row bg-primary" style="height: 350px; margin-top: 150px;">
        <div class="col-sm-8 offset-2">
            <div *ngIf="currentQuiz<5">
            <p class="text-center">{{currentQuiz +1}} of {{quizzes.length}}</p>
            <h1>{{quizzes[randomize].question}}</h1>


            <ul class="list-group" *ngFor="let quiz of quizzes[randomize].answer">
                <label>
                <li class="list-group-item" appBackground [correctAnswer]="quiz.correct">
                <input type="radio" name="options" (change)="onAnswer(quiz.correct)" [disabled]="answerSelected">
                    {{quiz.option}}
                </li>
                </label>
            </ul>
            </div>
            <div *ngIf="currentQuiz>4">
            <button class="btn btn bg-danger btn-block" style="margin-bottom: 50px; width:100px ;" (click)="showResult()">Show Result </button>
            <div class="text-center" *ngIf="result">
                Correct Answert : {{correctAnswers}} | Incorrect Answert : {{incorrectAnswert}}<br>
                <h1><b>  total score : {{correctAnswers}}/{{incorrectAnswert + correctAnswers}} </b></h1>

            </div>


        </div>
        <button [routerLink]="['/lesson']" style="margin-top: 20px;" class="btn btn-outline-dark">back to subject</button>
        </div>
    </div>
</div>

</body>
```

(image 32)

Στο αρχείο html του quiz βάζοντας μια ngIf ορίζουμε τον αριθμό των ερωτήσεων ότι δε ξεπερνούν τις 5, ξεκινώντας το μέτρημα από το 0 και αυξάνοντας τη σειρά κατά μια ερώτηση και βάζοντας μία ngFor τραβάμε τυχαία κάθε ερώτηση κάθε φορά. Κλείνοντας την ngIf ορίζουμε ότι ο αριθμός των ερωτήσεων είναι μεγαλύτερος του 4, καθώς γνωρίζουμε εξαρχής τον αριθμό από τις ερωτήσεις που έχουμε περάσει δυναμικά μέσα στο αρχείο services.



```
import { Quiz } from '../quiz.model';

@Injectable({
  providedIn: 'root'
})
export class CsharpService {

  quizzes: Quiz[] =[
    {
      question: 'Can we use “this” command within a static method?',
      answer: [
        { option: 'We can't use 'This' in a static method because we can only use static variables/methods in a static method.', correct:true},
        { option: 'We can use 'This' in a static method because we can only use static variables/methods in a static method.', correct:false},
        { option: 'all the above', correct:false}

      ]
    },
    {
      question: 'What are value types ?',
      answer: [
        { option: 'A value type holds many data value within its own memory space.', correct:false},
        { option: 'A value type holds a data value without its own memory space.', correct:false},
        { option: 'A value type holds a data value within its own memory space.', correct:true}

      ]
    },
    {
      question: 'What are reference types??',
      answer: [
        { option: 'The 'is' operator compares the name of the two objects. ', correct:false},
        { option: 'Reference type stores the address of the Object where the value is being stored. It is a pointer to another memory location. ', correct:true},
        { option: 'Reference type stores the address of the class where the value is being stored. It is a pointer to another memory location. ', correct:false}

      ]
    },

    {
      question: 'Can a private virtual method can be overridden?',
      answer: [
        { option: 'Yes, because they are not accessible outside the class.. ', correct:false},
        { option: 'No, because they are accessible outside the class.', correct:false},
        { option: 'No, because they are not accessible outside the class. ', correct:true}

      ]
    },
    {
      question: 'How can we sort the elements of the Array in descending order?',
      answer: [
        { option: 'Using Sort() methods followed by Reverse() method.  ', correct:true},
        { option: 'Using Sort[] methods followed by Reverse() method. ', correct:false},
        { option: 'all the above ', correct:false}

      ]
    },
]
  constructor() { }
  getQuizzes(){
    return this.quizzes;
```

(image 33)

Στο αρχείο αυτό δημιουργούμε τις ερωτήσεις που θα έχει ο πίνακας quiz, γράφοντας την ερώτηση και στη συνέχεια μέσα σε option τις απαντήσεις στις οποίες δίνουμε τιμή true or false. Τέλος δημιουργούμε τη μέθοδο getQuizzes από την οποία παίρνουμε τις ερωτήσεις και τα αποτελέσματα των ερωτήσεων και τα επιστρέφουμε στο αρχείο ts.

Python Quiz Component

Αναλύοντας τον κώδικα typescript βλέπουμε ότι δηλώνουμε έναν πίνακα quizzers δηλώνουμε ως σωστή απάντηση την τιμή true και ως λάθος την τιμή false, επιπλέον βάζουμε τυχαία επιλογή ερωτήσεων και έναν μετρητή counter.

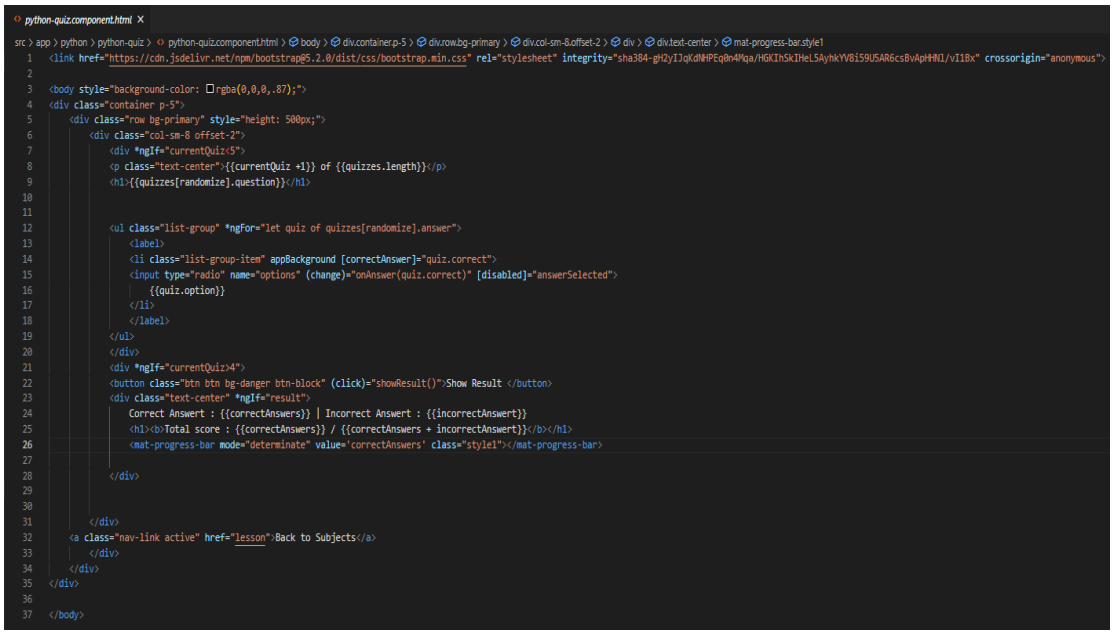
Μέσα στην ngOnit () καλούμε την μέθοδο getQuizzes από τα services. Στη μέθοδο onAnswer δηλώνουμε την σωστή απάντηση ως true και στη συνέχεια μέσα σε μια συνάρτηση timeout δημιουργούμε μια εναλλαγή των ερωτήσεων με τυχαία σειρά κάθε φορά ανά 2 δευτερόλεπτα αφού ο χρήστης επιλέξει την απάντηση. Αν η απάντηση είναι σωστή ο μετρητής για τις σωστές απαντήσεις αυξάνεται κατά ένα σε διαφορετική περίπτωση αυξάνεται ο μετρητής για τις λάθος απαντήσεις. Τέλος με τη μέθοδο showResult καλούμε να μας δείξει το συνολικό αποτέλεσμα.

```
import { Component, OnInit } from '@angular/core';
import { Quiz } from 'src/app/quiz.model';
import { PythonQuizService } from 'src/app/services/python-quiz.service';
@Component({
  selector: 'app-python-quiz',
  templateUrl: './python-quiz.component.html',
  styleUrls: ['./python-quiz.component.scss']
})
export class PythonQuizComponent implements OnInit {
  quizzes: Quiz[]= [];
currentQuiz=0;
answerSelected= false;
correctAnswers=0;
incorrectAnswert=0;
result=false;
randomize:any;

  constructor(private quizService:PythonQuizService) { }

  ngOnInit(): void {
    this.quizzes=this.quizService.getQuizzes();

    this.randomize = Math.floor(Math.random()* this.quizzes.length)
  }
  onAnswer(option:boolean){
    this.answerSelected=true;
    setTimeout(()=>{
      this.currentQuiz++;
      this.randomize = Math.floor(Math.random()* this.quizzes.length)

      this.answerSelected=false;

    },3000)

    if(option)
  {
    this.correctAnswers++;
  } else{
    this.incorrectAnswert++;
  }

  }

  showResult(){
    this.result=true;
  }

}
```

(image 34)

Στο αρχείο html του quiz βάζοντας μια ngIf ορίζουμε τον αριθμό των ερωτήσεων ότι δε ξεπερνούν τις 5 ξεκινώντας το μέτρημα από το 0 και αυξάνοντας τη σειρά κατά μια ερώτηση και βάζοντας μία ngFor τραβάμε τυχαία κάθε ερώτηση κάθε φορά.

Κλείνοντας την ngIf ορίζουμε ότι ο αριθμός των ερωτήσεων είναι μεγαλύτερος του 4, καθώς γνωρίζουμε εξαρχής τον αριθμό από τις ερωτήσεις που έχουμε περάσει δυναμικά μέσα στο αρχείο services.



(image 35)

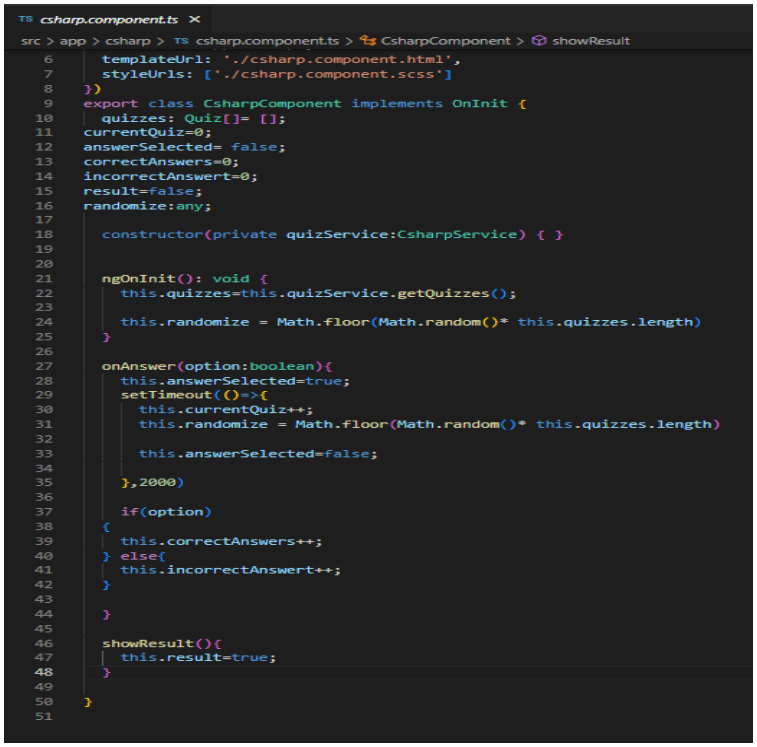
Στο αρχείο services δημιουργούμε τις ερωτήσεις που θα έχει ο πίνακας quiz, γράφοντας την ερώτηση και στην συνέχεια μέσα σε option τις απαντήσεις στις οποίες δίνουμε τιμή true or false. Τέλος δημιουργούμε την μέθοδο getQuizzes από την οποία παίρνουμε τις ερωτήσεις και τα αποτελέσματα των ερωτήσεων και τα επιστρέφουμε στο αρχείο ts.

C sharp Quiz Component

Αναλύοντας τον κώδικα typescript βλέπουμε ότι δηλώνουμε έναν πίνακα quizzers δηλώνουμε ως σωστή απάντηση την τιμή true και ως λάθος την τιμή false, επιπλέον βάζουμε τυχαία επιλογή ερωτήσεων και έναν μετρητή counter.

Μέσα στην ngOnit () καλούμε την μέθοδο getQuizzes από τα services. Στη μέθοδο onAnswer δηλώνουμε τη σωστή απάντηση ως true και στη συνέχεια μέσα σε μια συνάρτηση timeout δημιουργούμε μια εναλλαγή των ερωτήσεων με τυχαία σειρά κάθε φορά ανά 2 δευτερόλεπτα αφού ο χρήστης επιλέξει την απάντηση.

Αν η απάντηση είναι σωστή ο μετρητής για τις σωστές απαντήσεις αυξάνεται κατά ένα σε διαφορετική περίπτωση αυξάνεται ο μετρητής για τις λάθος απαντήσεις. Τέλος με τη μέθοδο showResult καλούμε να μας δείξει το συνολικό αποτέλεσμα.



(image 36)

Στο αρχείο html του quiz βάζοντας μια ngIf ορίζουμε τον αριθμό των ερωτήσεων ότι δε ξεπερνούν τις 5 ξεκινώντας το μέτρημα από το 0 και αυξάνοντας τη σειρά κατά μια ερώτηση και βάζοντας μία ngFor τραβάμε τυχαία κάθε ερώτηση κάθε φορά.

Κλείνοντας την ngIf ορίζουμε ότι ο αριθμός των ερωτήσεων είναι μεγαλύτερος του 4, καθώς γνωρίζουμε εξαρχής τον αριθμό από τις ερωτήσεις που έχουμε περάσει δυναμικά μέσα στο αρχείο services.

(image 37)

Τέλος προσθέτουμε δύο buttons το πρώτο ονομάζεται showResult και επιστρέφει τα αποτελέσματα που μας δίνει η μέθοδος getResult(). Το δεύτερο button μας δίνει τη δυνατότητα να επιστρέψουμε στη σελίδα lesson μέσω του routerLink.

Models

Το Angular είναι ένα πλαίσιο διεπαφής που κρατά τις πληροφορίες από το πρόγραμμα περιήγησης και τις στέλνει στη βάση δεδομένων. Για την αποθήκευση των πληροφοριών δημιουργούνται κλάσεις μοντέλου. Οι κλάσεις μοντέλων μπορεί να είναι τύπου διεπαφής ή κλάσεων. Ας δούμε πώς μπορούμε να δημιουργήσουμε και να προσθέσουμε χαρακτηριστικά με μη αυτόματο τρόπο.

Μπορούμε να δημιουργήσουμε ένα μοντέλο χρησιμοποιώντας γωνιακό CLI ή χειροκίνητα σε γραφομηχανή. Με τον πρώτο τρόπο απλά δημιουργούμε έναν φάκελο models μέσα στο src και εκεί δημιουργούμε το αρχείο που θέλουμε στη δικιά μας περίπτωση τα models (student , subject και questions), μπορούμε βέβαια να χρησιμοποιήσουμε την εντολή της angular : **ng g class (name) –type=model**

```
export class Subject {
    id:any;
    name:any;
    afm:any;
    subject:any;
    date:any;
    hours:any;
}
```

(image 38)

```
export class Quiz {
    question!: string;
    answer!: { option: any ,correct:boolean } [];

}
```

(image 39)

```
app > models > TS student.ts > Student
export class Student {
    id:any;
    name:any;
    afm:any;
    age:any;
    mobile:any;
    email:any;
    address:any;
    subject:any;
    hours:any;

}
```

(image 40)

Ο student , subject , questions είναι κλάσεις μοντέλου που περιέχει δεδομένα μιας κλάσης student, subject και questions αντίστοιχα. Πώς προσθέτετε ιδιότητες στην κλάση μοντέλου;

- Πρέπει να προσθέσετε setter/getter ή constructor για να αρχικοποιήσετε αυτήν την κλάση

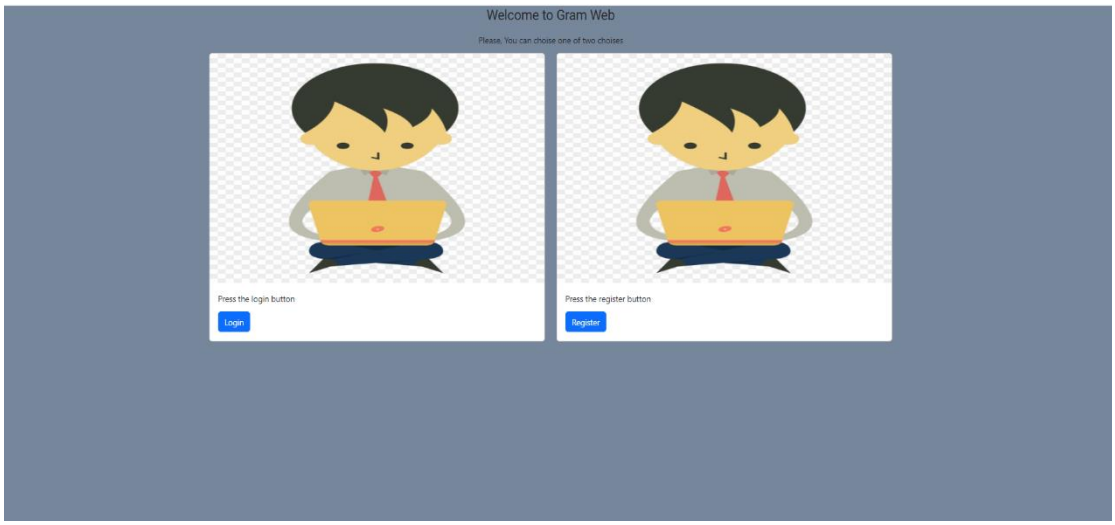
Κατάλογος CSS

Κάθε component αποτελείται όπως είδαμε παραπάνω από τρία βασικά αρχεία το component.ts , component.html και το component.css. Ωστόσο δεν έγινε κάποια αναφορά για τα αρχεία css τα οποία αποτελούν πολύ σημαντικό κομμάτι του κώδικα μας για την μορφοποίηση του UI μας. Είναι τα αρχεία, στα οποία μπορούμε να αλλάξουμε διαστάσεις, μέγεθος και χρώμα στις φόρμες και πινάκες μας. Μπορούμε να μορφοποιήσουμε τις κάρτες μας, ανάλογα με την διάσταση της σελίδας μας και γενικά να δώσουμε μια εντελώς διαφορετική μορφοποίηση της πλατφόρμας μας. Είναι γνωστό πλέον ότι το εργαλείο της bootstrap όπως και τα material της angular μας έχουν βγάλει από την δύσκολη θέση του να προσπαθούμε να κάνουμε το ui της πλατφόρμας μας εφαρμόσιμο σε κάθε διάσταση οθόνης και πολλές φορές στηριζόμαστε μόνο σε αυτά τα εργαλεία θεωρώντας την css κάτι πλέον το περιττό. Αυτό δεν μπορεί να συμβεί καμία από τις παραπάνω βιβλιοθήκες δεν μπορεί να καλύψει απόλυτα την css , ωστόσο είναι πολύ σημαντικές για την δομή του ui μας.

4. ΣΧΕΔΙΑΣΗ ΔΙΕΠΑΦΩΝ

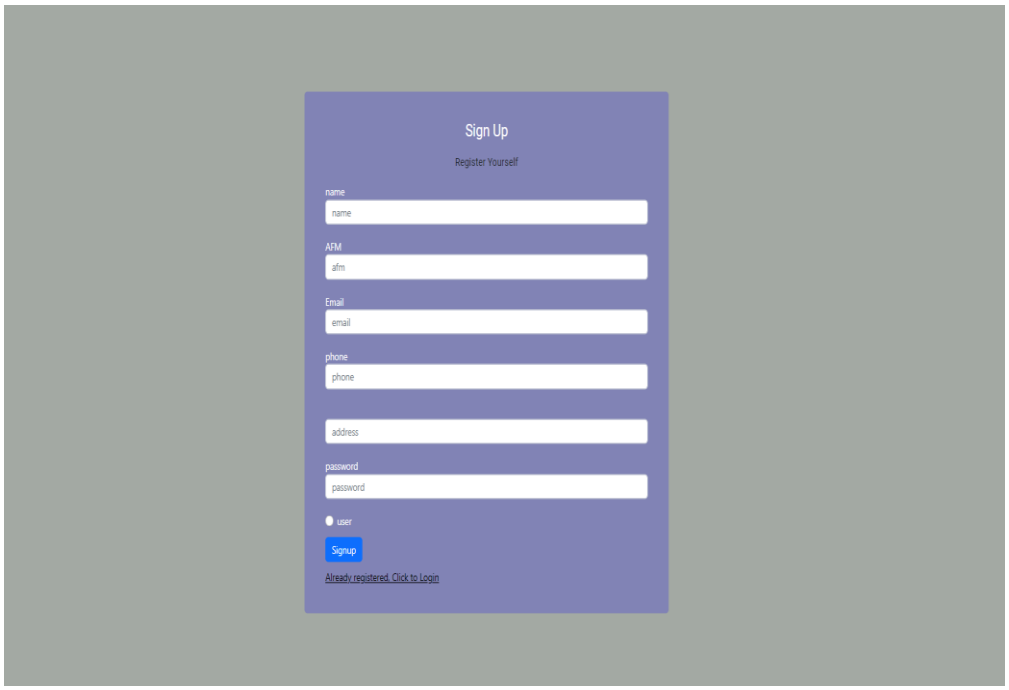
4.1 Log In Page

Η πρώτη σελίδα που θα συναντήσει ο χρήστης όταν πληκτρολόγηση το url : http://www.blog:4200 είναι η σελίδα σύνδεσης στην εφαρμογή που του δίνει την δυνατότητα είτε να κάνει απευθείας σύνδεση αν είναι εγγεγραμμένος ή να κάνει εγγραφή κάνοντας κλικ στην κάρτα register.



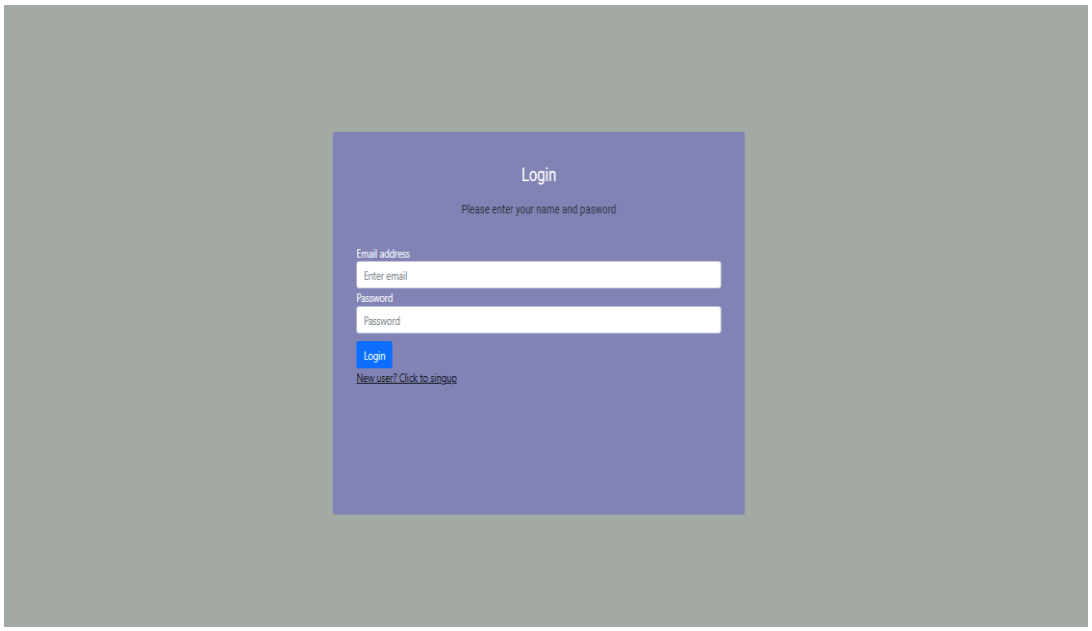
(image 41)

Στην περίπτωση όπου ο χρήστης δεν έχει κάνει εγγραφεί, κάνοντας κλικ στην κάρτα register, ανοίγει μια φόρμα εγγραφής στην οποία πρέπει να συμπληρώσει τα πεδία, που είναι υποχρεωτικά.



(image 42)

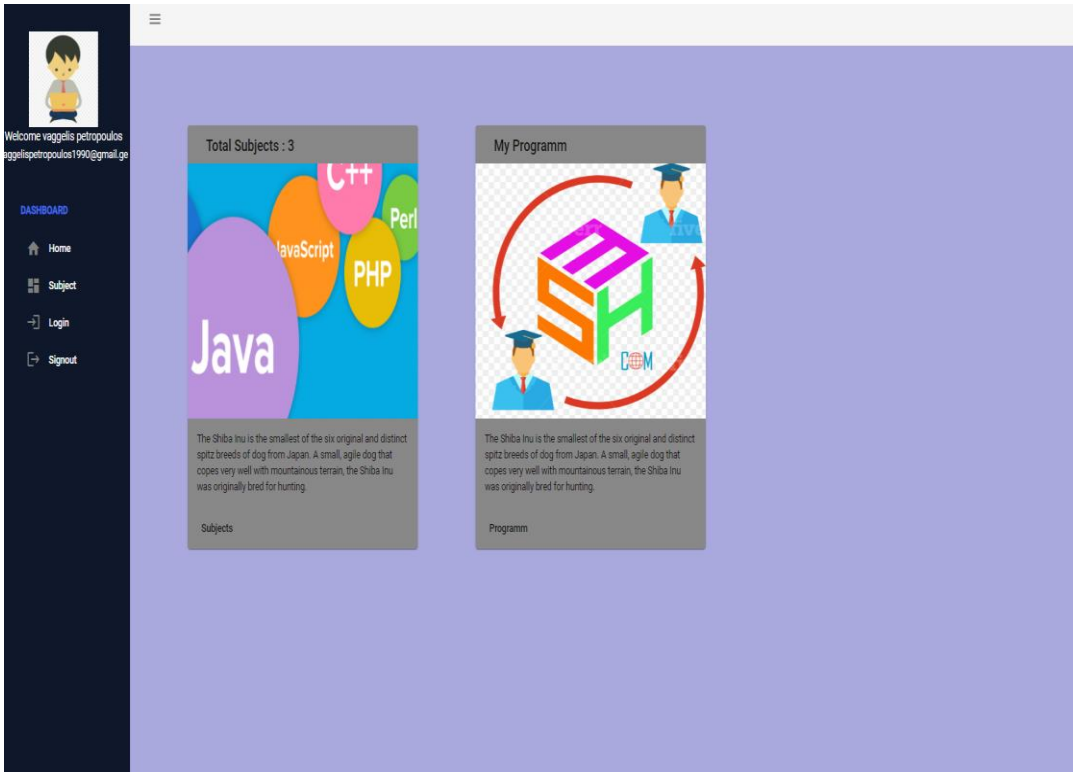
Τα πεδία αυτά είναι όνομα, ΑΦΜ (θεωρείται μοναδικό), email , τηλέφωνο, διεύθυνση και κωδικός. Τέλος ο χρήστης πρέπει να τσεκάρει το user ώστε να θεωρηθεί φοιτητής. Πατώντας το signup συνδέεται αυτόματα στην σελίδα login.



(image 43)

Στην σελίδα login ο χρήστης πληκτρολογεί το email και password με τα οποία έκανε εγγραφή και αποθηκεύτηκαν στην βάση δεδομένων με στοιχεία χρήστη. Πατώντας το κουμπί login συνδέετε στο homepage.

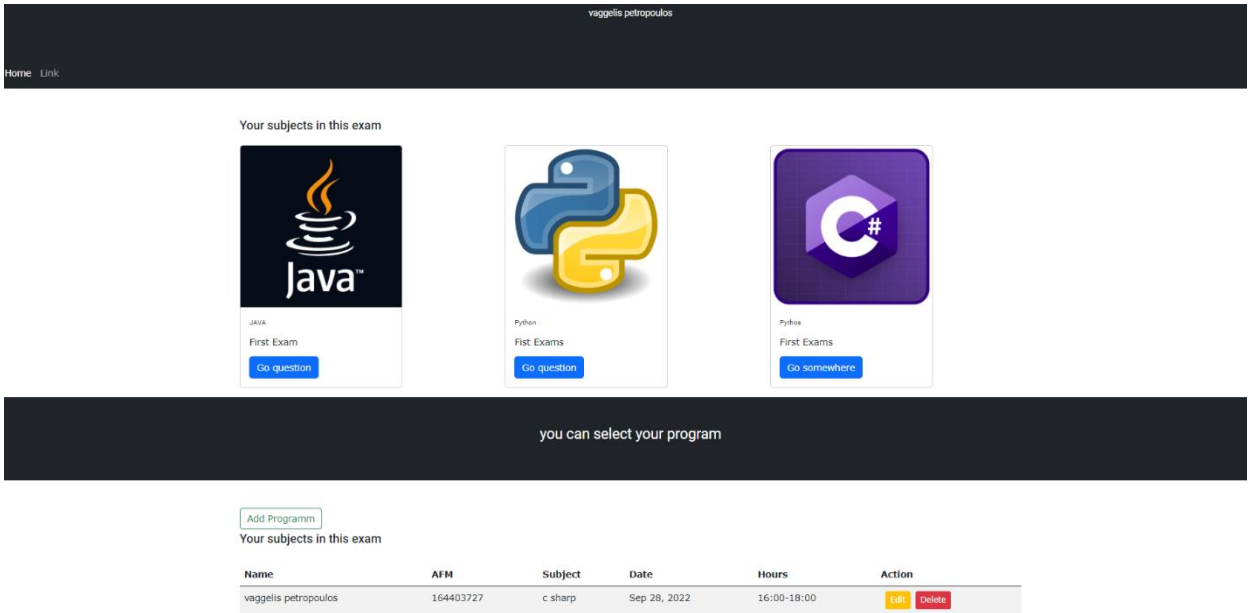
4.2 Home Page



(image 44)

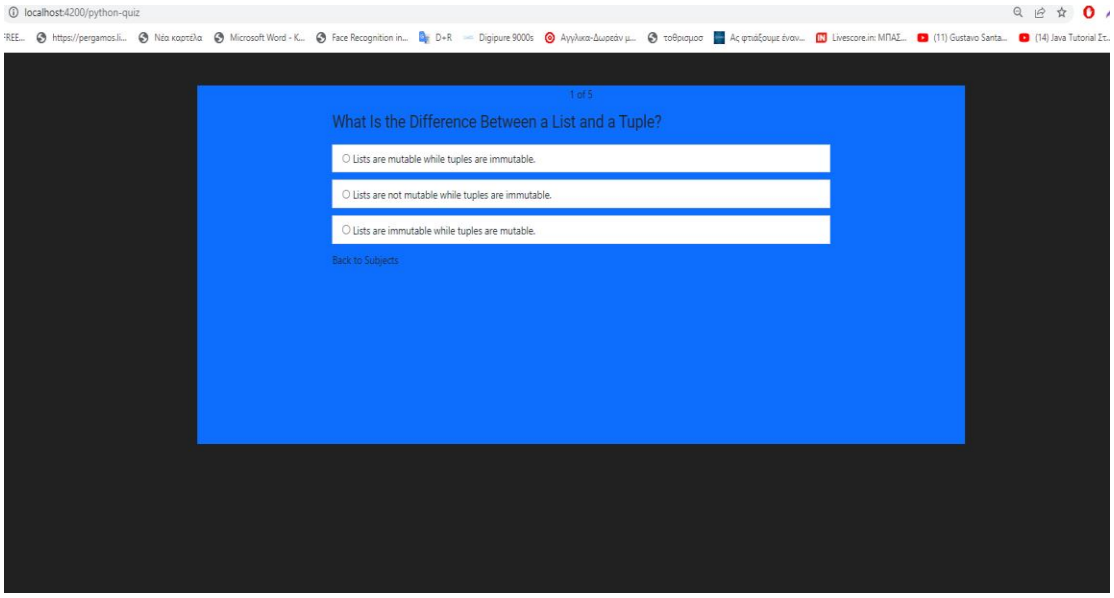
Στο homepage ο χρήστης έχει διαθέσιμο ένα menu “dashboard” το οποίο βρίσκεται στο εσωτερικό ενός slider bar. Περιέχει το menu στο οποίο μπορεί να κινηθεί ο φοιτητής.

Το home, τον οδηγεί στην αρχική του σελίδα εκεί δηλαδή που βρίσκεται όταν κάνει login. Το subject μεταφέρει τον φοιτητή στη σελίδα μαθημάτων.



(image 45)

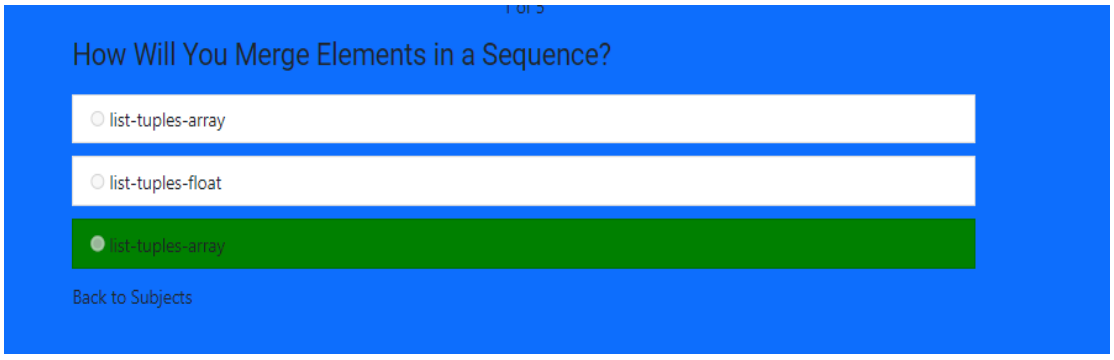
Στη σελίδα αυτή ο χρήστης μπορεί να δει τα διαθέσιμα μαθήματα του εξαμήνου και να επιλέξει έστω δύο από αυτά. Στην παραπάνω εικόνα βλέπουμε τρία μαθήματα java , python και c sharp, για τα οποία κάνοντας κλικ σε μια από τις τρεις καρτέλες, ο χρήστης μπορεί να δει και να απαντήσει κάποιες ερωτήσεις που έχει βάλει ο καθηγητής σχετικά με το μάθημα. Κάνοντας κλικ ο φοιτητής στην καρτέλα της python ανοίγει η σελίδα python-question-page.



(image 46)

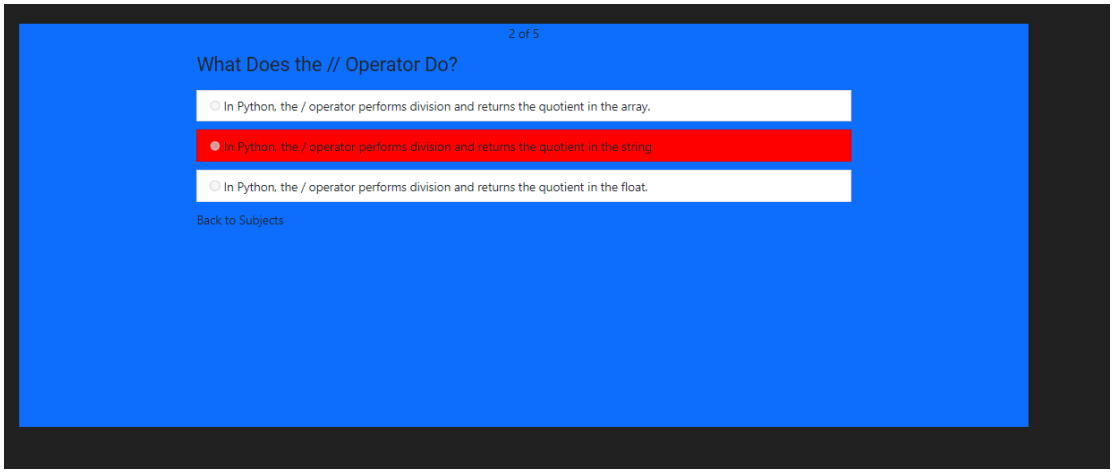
Έχουμε συνδεθεί σε ένα ερωτηματολόγιο στο οποίο πρέπει να απαντήσουμε πέντε ερωτήσεις σχετικές με την δομή και λειτουργία της python. Οι ερωτήσεις αυτές έχουν τυχαία επιλογή κάθε φορά και ο χρήστης έχει την δυνατότητα αφού επιλέξει μια ερώτηση να αλλάξει την επιλογή του μέσα σε τρία δευτερόλεπτα, αλλιώς προχωράει αυτόματα στη δεύτερη ερώτηση.

Αν η ερώτηση είναι σωστή εμφανίζεται με πράσινο χρώμα σε αντίθετη περίπτωση εμφανίζεται με κόκκινο χρώμα.



(image 47)

Η δεύτερη εικόνα δείχνει μια λάθος απάντηση από τον χρήστη.

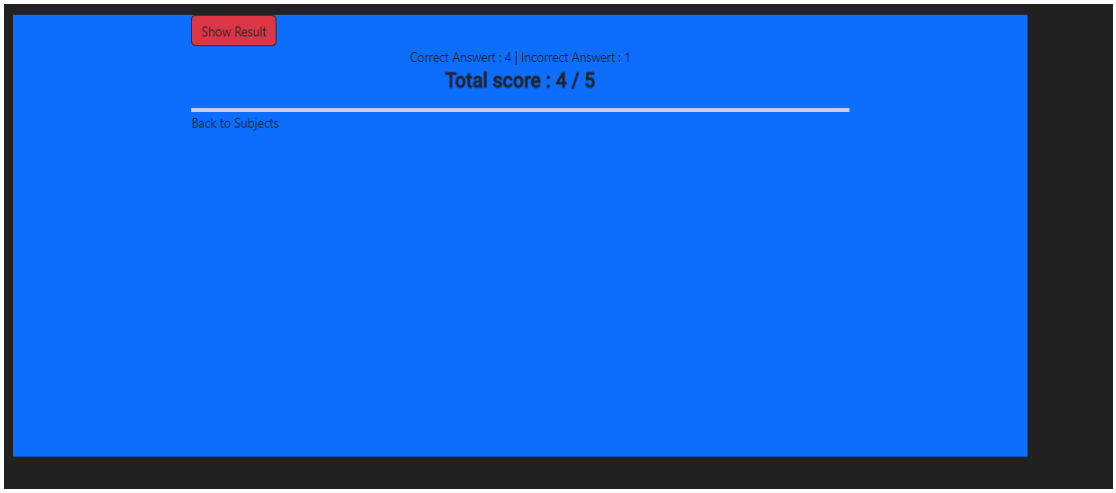


(image 48)

Αν ο φοιτητής θέλει να αποχωρήσει από τις ερωτήσεις υπάρχει ένα button back to Subject κάτω από τα πεδία επιλογών όπου ο χρήστης μπορεί να το κλικάρει και να συνδεθεί ξανά στη σελίδα μαθημάτων. Στην περίπτωση όπου ο φοιτητής συνεχίσει να απαντάει στις ερωτήσεις, μετά την πέμπτη ερώτηση θα του εμφανιστεί το συνολικό του σκορ όπως φαίνεται παρακάτω.

(image 49)

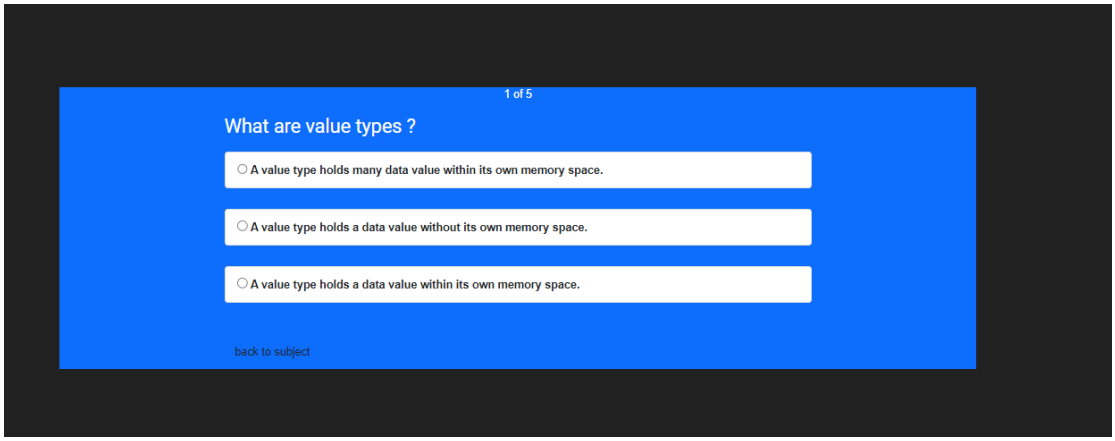
Πατώντας το button show result ο φοιτητής μπορεί να δει τα αποτελέσματα των απαντήσεων.



(image 50)

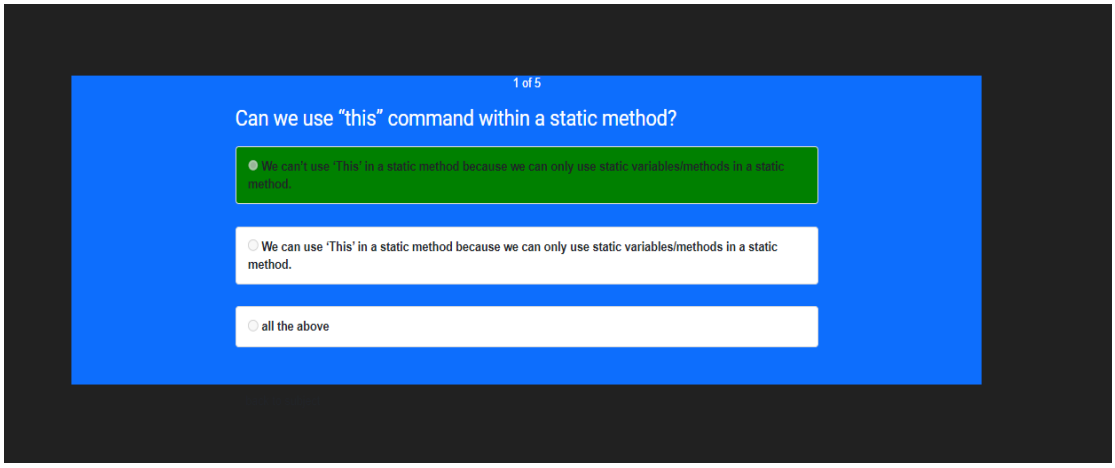
Όπως φαίνεται στην παραπάνω εικόνα ο συγκεκριμένος φοιτητής έχει απάντηση σωστά σε τέσσερις από τις πέντε ερωτήσεις του.

Επιλέγοντας την δεύτερη καρτέλα της c sharp ο χρήστης κάνει ακριβώς την ίδια διαδικασία, επιλέγει αρχικά την καρτέλα, κάνει κλικ στο button go question και στην συνέχεια επιλέγει τις ερωτήσεις.



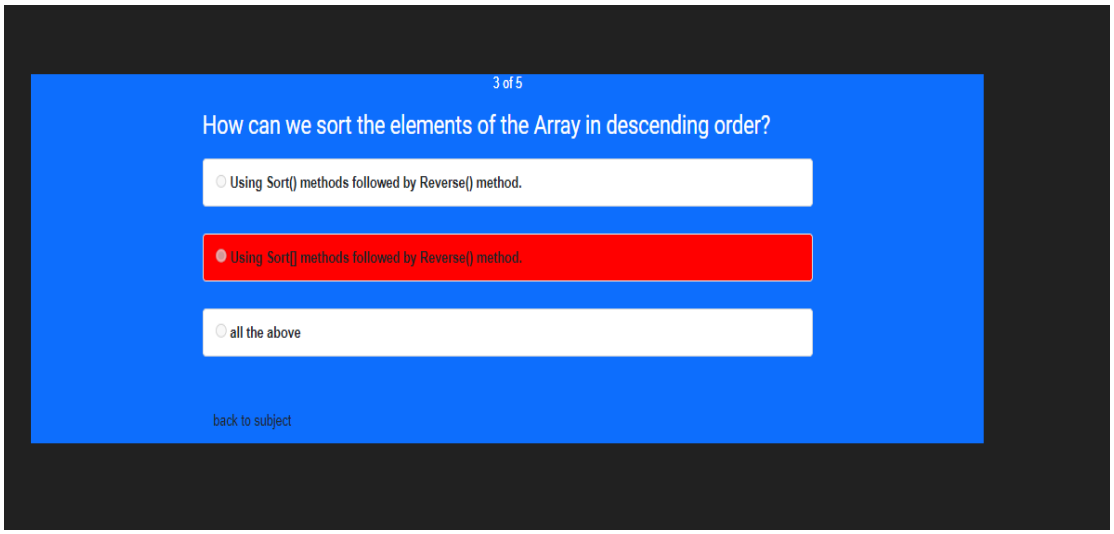
(image 51)

Αν η ερώτηση είναι σωστή εμφανίζεται με πράσινο χρώμα σε αντίθετη περίπτωση εμφανίζεται με κόκκινο χρώμα.



(image 52)

Η δεύτερη εικόνα δείχνει μια λάθος απάντηση από τον χρήστη.

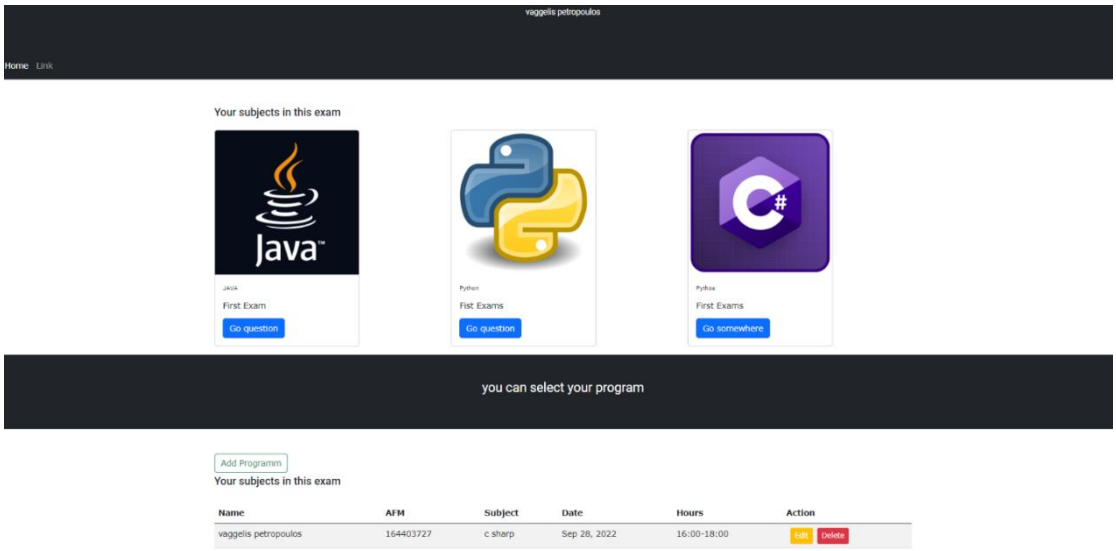


(image 53)

Η ίδια διαδικασία γίνεται με την καρτέλα java όπου ο φοιτητής απαντάει γενικές ερωτήσεις σχετικά με την δομή και την λειτουργία της java. Με αυτόν τον τρόπο αποκτά μια σχετική γενική γνώση για το περιεχόμενο των μαθημάτων που θα παρακολουθήσει. Αξίζει να σημειωθεί ότι κάθε εβδομάδα ο καθηγητής του μαθήματος ανανεώνει τις ερωτήσεις, ώστε οι φοιτητές που θέλουν να απαντήσουν στις ερωτήσεις να έχουν διαφορετικό υλικό κάθε εβδομάδα.

(image 54)

Στην συνέχεια ο φοιτητής μπορεί να επιστρέψει στην αρχική του σελίδα , πατώντας το button back to subject με σκοπό να επιλέξει το πρόγραμμα του.



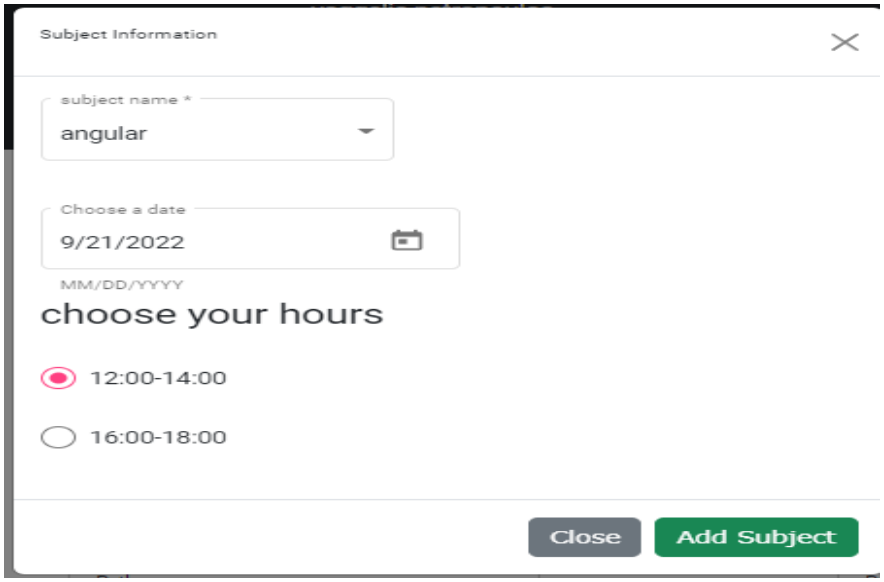
(image 55)

Όπως βλέπουμε στην παραπάνω εικόνα ο φοιτητής Vaggelis Petropoulos έχει επιλέξει το μάθημα c sharp στις 28/10/22 και ώρα 16:00-18:00. Ας αναλύσουμε πως μπορεί ένας φοιτητής να επιλέξει το πρόγραμμα του.

Αρχικά πατά το button add Program

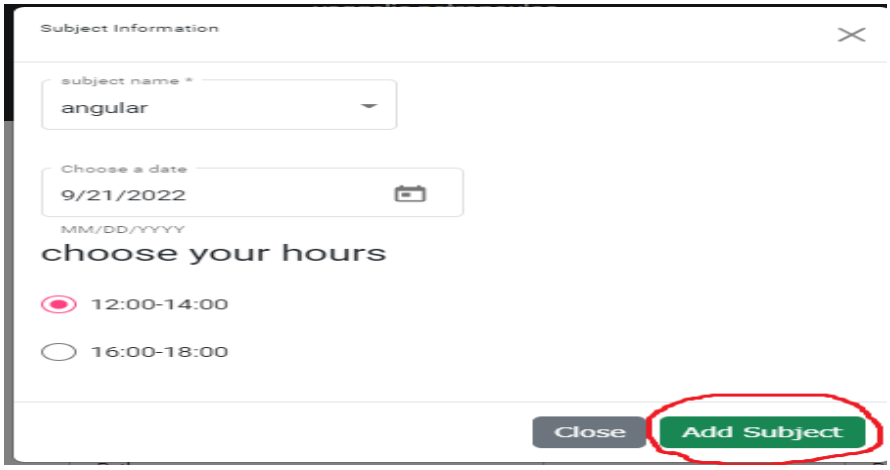
(image 56)

Στην συνέχεια ανοίγει μια φόρμα εισαγωγής μαθημάτων, επιλογής ημερομηνίας μέσω ενός date picker και τέλος επιλέγει την ώρα που θέλει να παρακολουθήσει. Οι ώρες είναι επιλεγμένες 12:00-14:00 και 16:00-18:00.



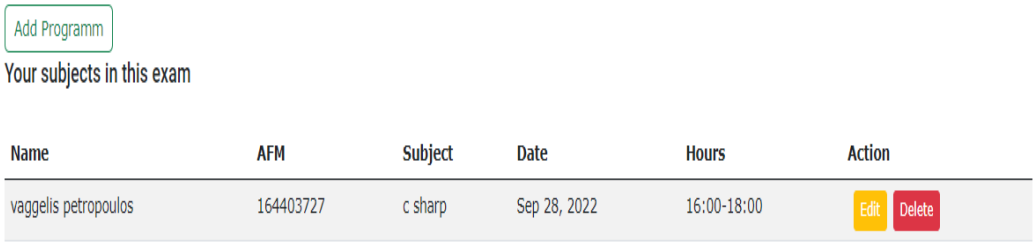
(image 57)

Στο συγκεκριμένο παράδειγμα της παραπάνω εικόνας ο φοιτητής επιλέγει το μάθημα angular την ημερομηνία 9/21/22 και ώρα 12:00-14:00. Αν ο φοιτητής θέλει να αποθηκεύσει αυτό το πρόγραμμα πατάει το κουμπί add Subject.



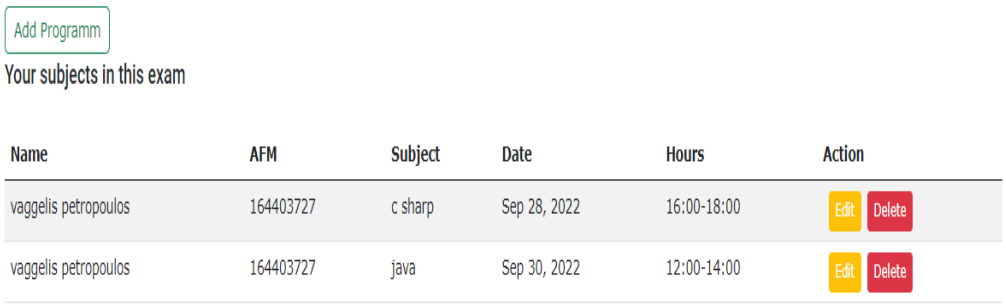
(Image 58)

Και στη συνέχεια το μάθημα , η ημερομηνία και η ώρα αποθηκεύονται στον προσωπικό του πίνακα.



(image 59)

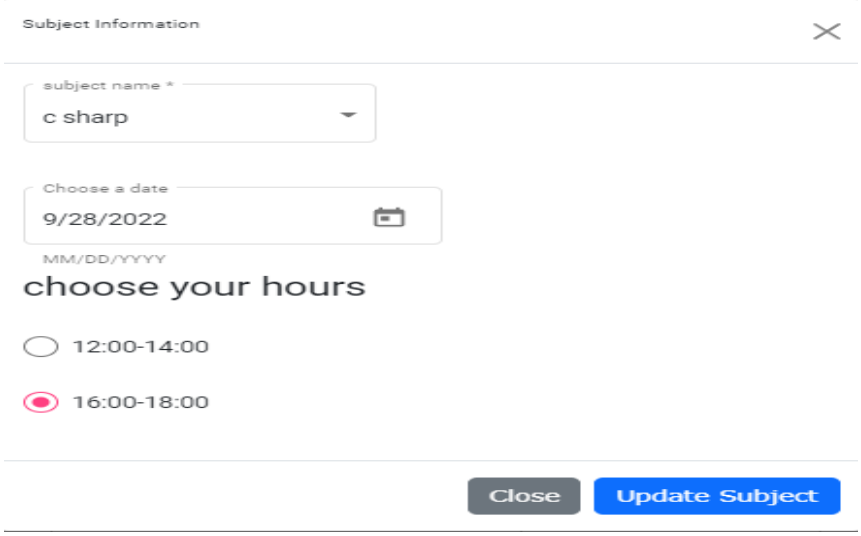
Επιπλέον ο χρήστης μπορεί να επιλέξει επιπλέον νέο μάθημα με διαφορετική ώρα και ημέρα, όπως στο παρακάτω παράδειγμα.



(image 60)

Βλέπουμε ότι με την ίδια ακριβώς διαδικασία ο φοιτητής έχει επιλέξει στο προσωπικό του πρόγραμμα νέο μάθημα την java με διαφορετική ημερομηνία και ώρα.

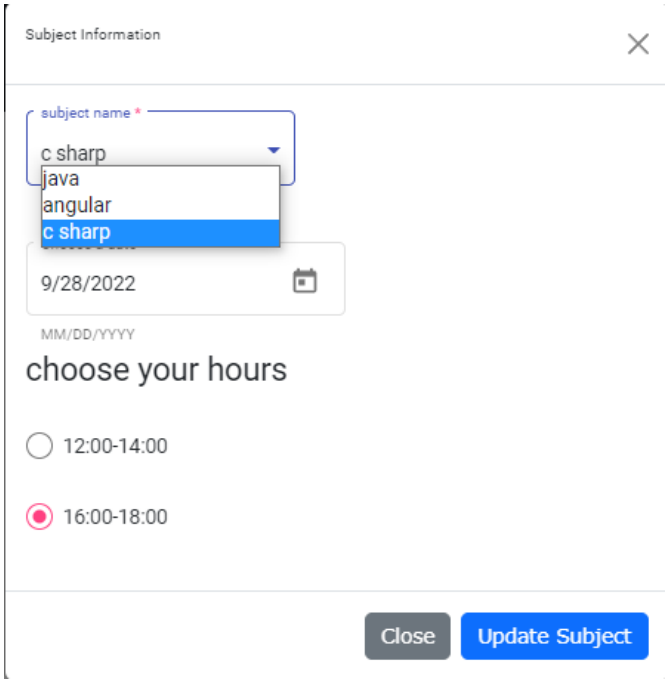
Όπως μπορούμε να παρατηρήσουμε στον πίνακα προγράμματος κάθε φοιτητή υπάρχει δυνατότητα επεξεργασίας ή διαγραφής κάποιου μαθήματος του σε ένα συγκεκριμένο χρονικό διάστημα έως και πέντε μέρες πριν. Πατώντας το edit ανοίγει η παρακάτω φόρμα, η οποία είναι ήδη συμπληρωμένη με το πρόγραμμα που έχει επιλέξει ο χρήστης όπως φαίνεται παρακάτω.



(image 61)

Ο φοιτητής μπορεί να αλλάξει το μάθημα επιλέγοντας το πεδίο του μαθήματος και αλλάζοντας την c sharp σε java ή αλλάζοντας την ημερομηνία από 9/28/2022 σε κάποια άλλη και την ώρα επίσης. Τέλος πατώντας το update Subject ενημερώνεται η βάση για την αλλαγή και επιστρέφει στον πίνακα του προγράμματος τα αλλαγμένα πεδία.

Ένα παράδειγμα όπου ο χρήστης αλλάζει το μάθημα που είχε επιλέξει αρχικά.



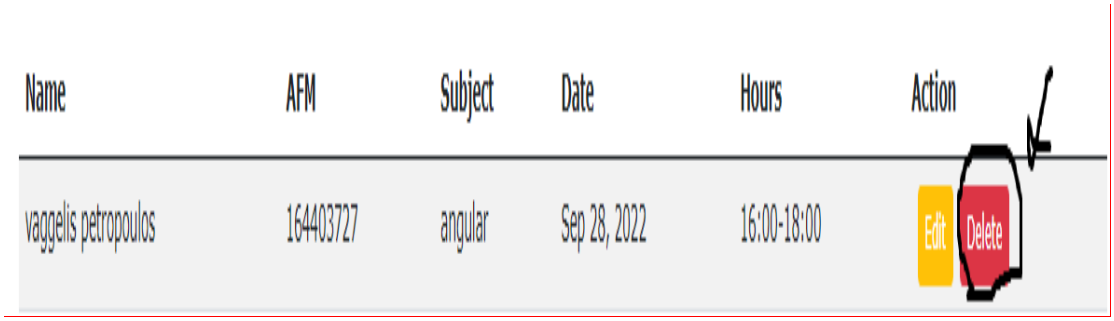
(image 62)

Ενημερώνοντας την βάση επιστρέφει στον πίνακα την angular που θα ήθελε να παρακολουθήσει.

(image 63)

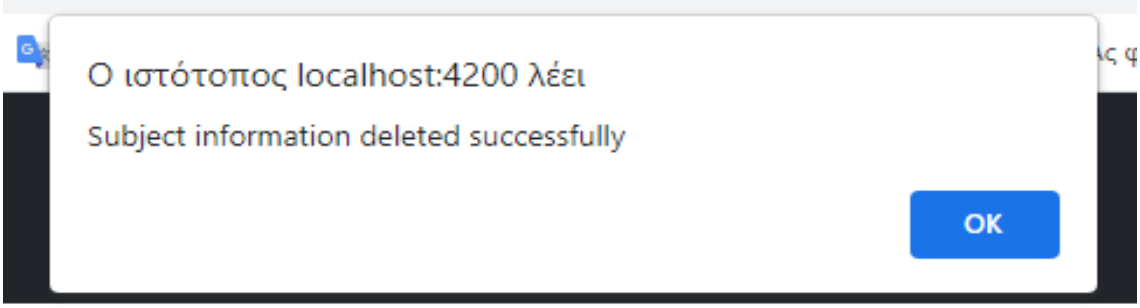
Στην συνέχεια υπάρχει και η περίπτωση όπου ο χρήστης θέλει να διαγράψει ένα μάθημα από το πρόγραμμα του, είτε επειδή δεν μπορεί να το παρακολουθήσει είτε επειδή δεν είναι διαθέσιμος εκείνη την ώρα ή μέρα.

Αυτό γίνεται αρκετά εύκολα, επιλέγει το delete .



(image 64)

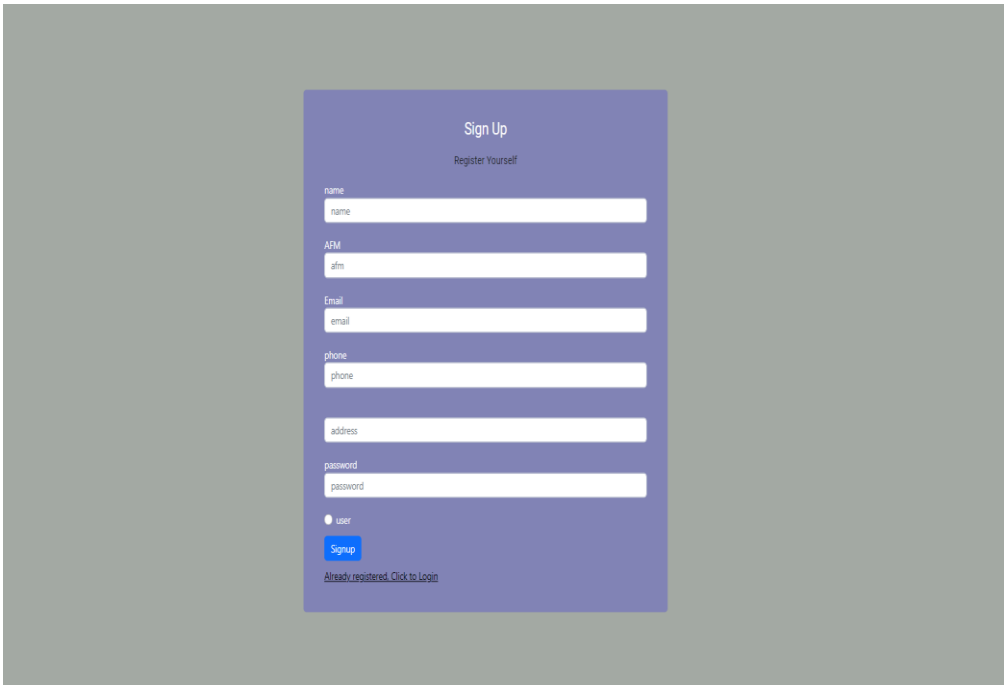
Και εμφανίζεται μήνυμα επιβεβαίωσης διαγραφής.



(image 65)

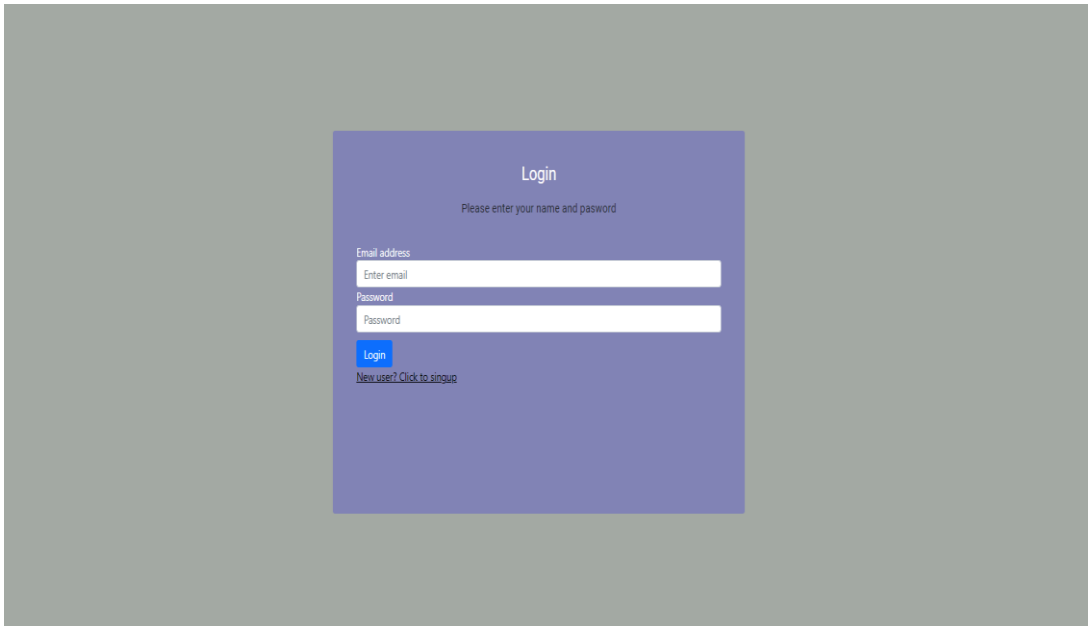
Κάνοντας κλικ στο ok το μάθημα διαγράφεται από τη βάση και επιτυχώς από τον πίνακα προγράμματος. Στην περίπτωση όπου ο φοιτητής αλλάξει γνώμη για τη διαγραφή πρέπει να επιστρέψει στο πρόγραμμα και να επιλέξει ξανά το μάθημα, την ώρα και ημερομηνία την οποία μπορεί να είχε. Αυτό γίνεται γιατί σε περίπτωση διαγραφής η βάση ενημερώνεται αυτόματα και δεν υπάρχει επιλογή διόρθωσης ή ακύρωσης της διαγραφής.

Στο σημείο αυτό θα αναλύσουμε τις δυνατότητες που έχει ο καθηγητής κάνοντας σύνδεση στην πλατφόρμα. Ξεκινώντας με τη διαδικασία εγγραφής, ο καθηγητής συμπληρώνει τα στοιχεία του. Ωστόσο δεν πρέπει να κάνει τσεκ το πεδίο user.



(image 66)

Πατώντας το signup συνδέεται αυτόματα στην σελίδα login.



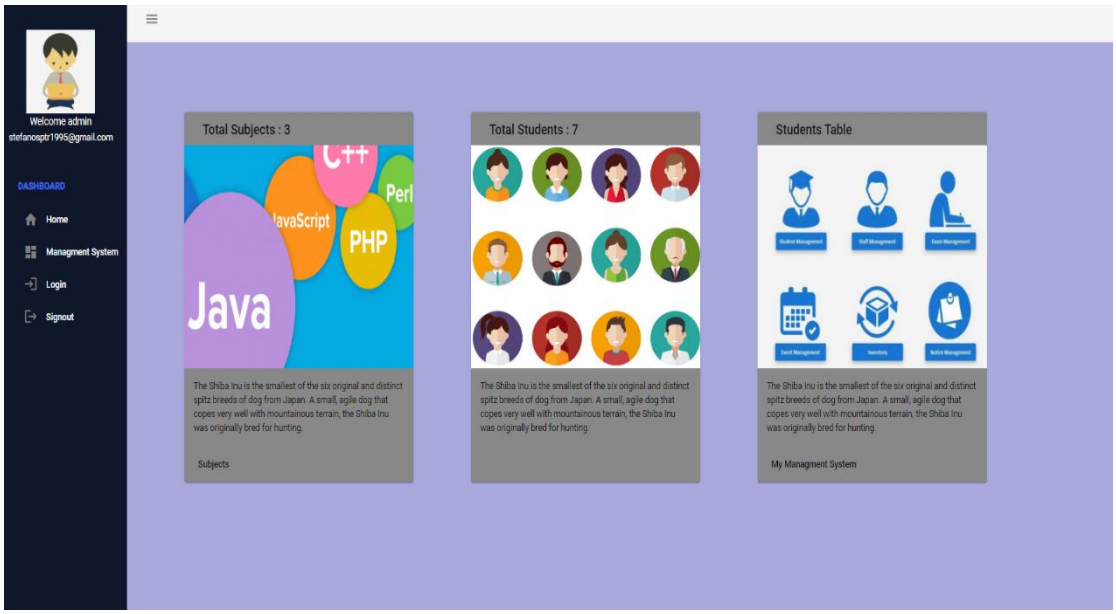
(image 67)

Στην σελίδα login ο καθηγητής πληκτρολογεί το email και password με τα οποία έκανε εγγραφή και αποθηκεύτηκαν στην βάση δεδομένων με στοιχεία χρήστη. Πατώντας το κουμπί login συνδέετε στο homepage.

Home page

Στο homepage ο καθηγητής έχει διαθέσιμο ένα menu “dashboard” το οποίο βρίσκεται στο εσωτερικό ενός slider bar. Περιέχει το menu στο οποίο μπορεί να κινηθεί ο καθηγητής.

Το home τον οδηγεί στην αρχική του σελίδα εκεί δηλαδή που βρίσκετε όταν κάνει login.

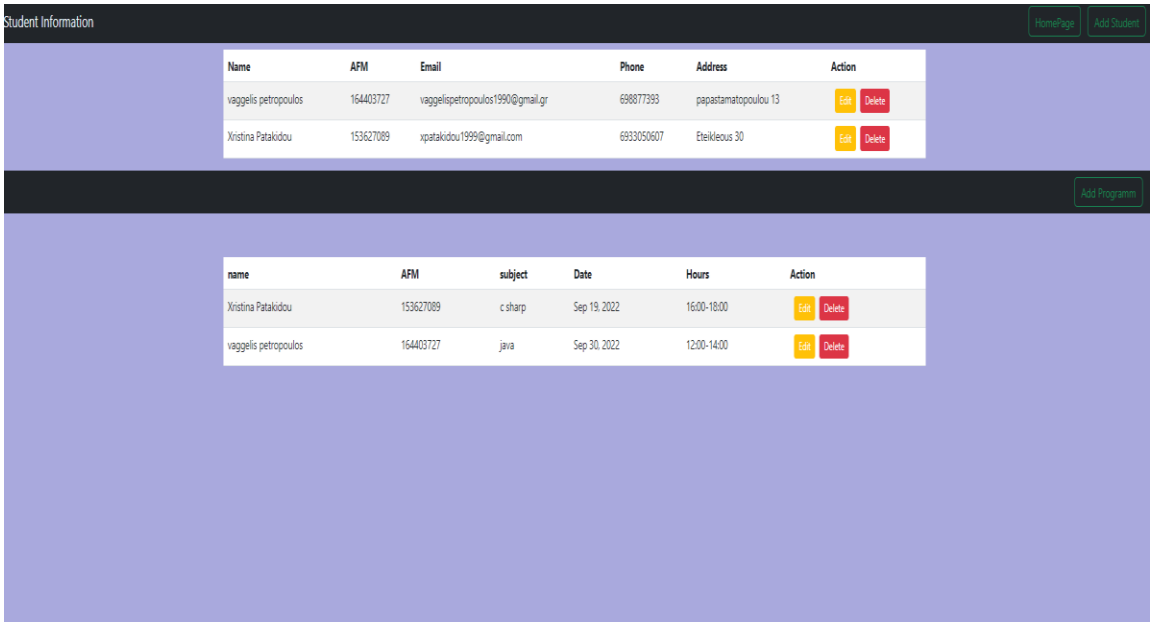


(image 68)

Το dashboard του καθηγητή περιέχει το management system, με το οποίο διαχειρίζεται τον πίνακα των μαθητών και τον πίνακα των μαθημάτων που θα δούμε στην συνέχεια. Επιπλέον περιέχει το login και signout και σύνδεση και αποσύνδεση από την πλατφόρμα.

Management System

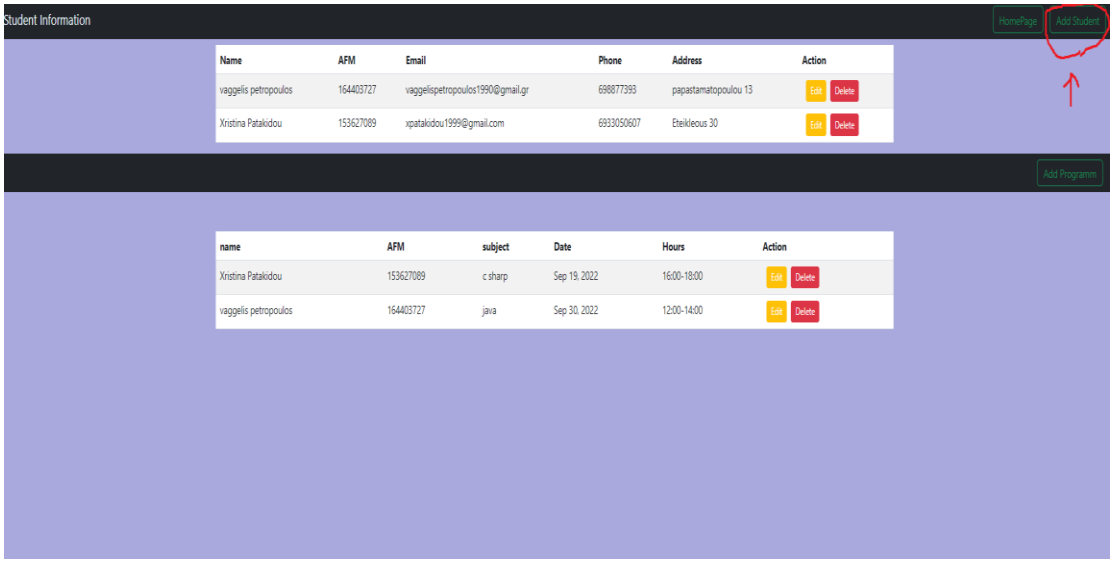
Κάνοντας επιλογή στο management system ο καθηγητής μπορεί να δει τους παρακάτω πίνακες.



(image 69)

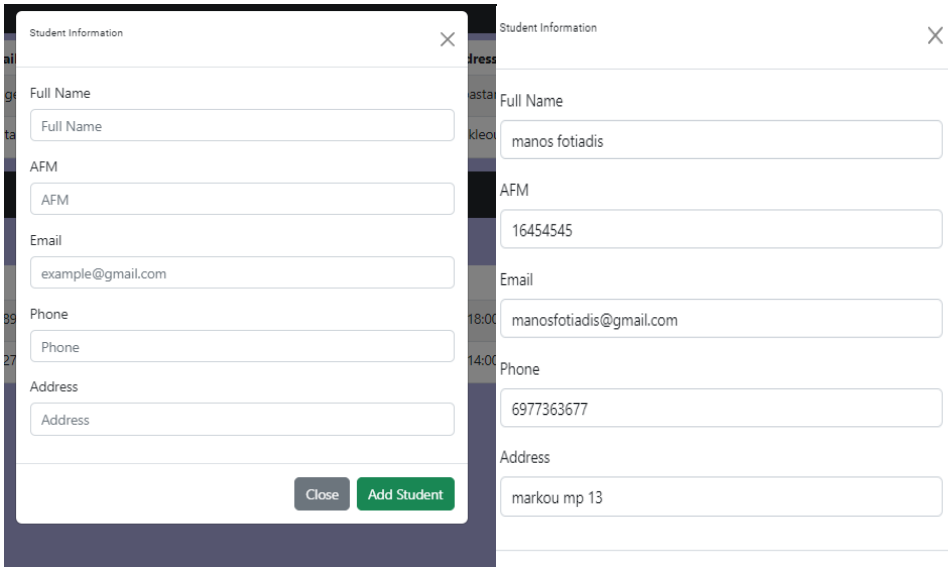
Οι παραπάνω πίνακες παίρνουν τα στοιχεία τους από την βάση των φοιτητών και από την βάση των μαθημάτων. Ο πρώτος πίνακας καταγράφει τα στοιχεία των φοιτητών μετά από κάθε νέα εγγραφή έτσι ώστε να μπορεί ο καθηγητής να δει την συμμετοχή των μαθητών στην πλατφόρμα καθώς και τα στοιχεία τους αν χρειαστεί να τους ενημερώσει για κάτι σχετικό με το μάθημα.

Στο nav-bar υπάρχει το button add student με το οποίο ο καθηγητής μπορεί να εισάγει ένα νέο φοιτητή στην πλατφόρμα.



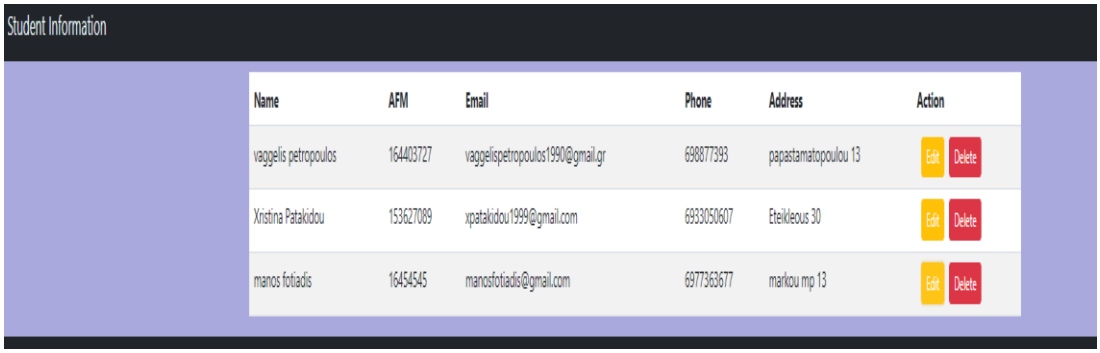
(image 70)

Πατώντας το κουμπί αυτό ανοίγει μια φόρμα στην οποία ο καθηγητής συμπληρώνει τα στοιχεία του νέου φοιτητή. Όνομα , ΑΦΜ , email , τηλέφωνο , διεύθυνση.



(image 71)

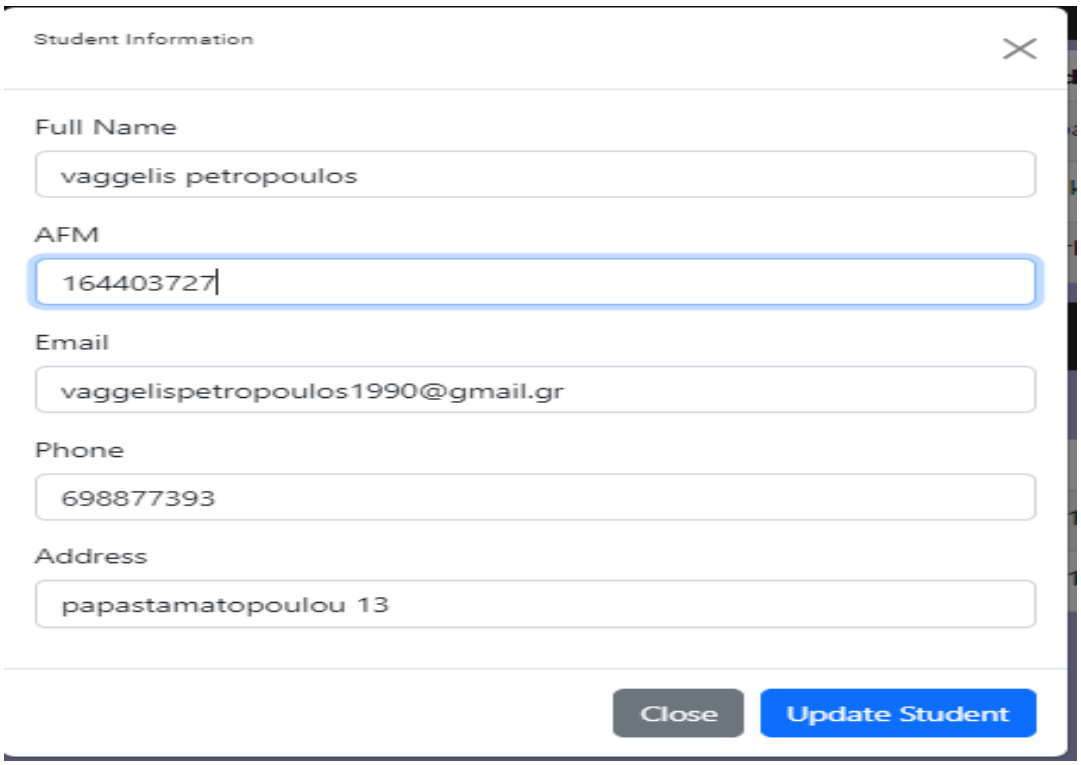
Πατώντας το add student ενημερώνεται η βάση του φοιτητή και ο πίνακας κάνει εγγραφή από την βάση τον νέο φοιτητή. Όπως βλέπουμε και στην παρακάτω εικόνα έχει γίνει εγγραφή του φοιτητή Manos Fotiadis.



(image 72)

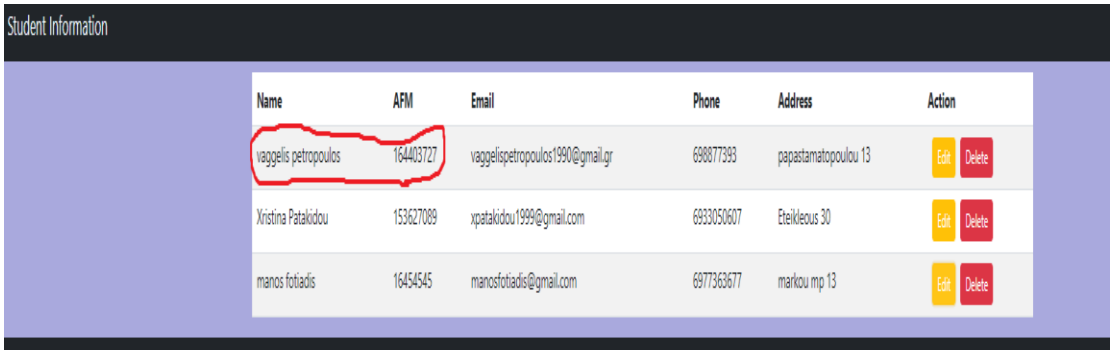
Στην περίπτωση αυτή ο φοιτητής ενημερώνεται από τον καθηγητή για την εγγραφή του στην πλατφόρμα και αλλάζει το password του έχει δημιουργήσει ο καθηγητής σε ένα νέο δικό του για να μπορεί να κάνει σύνδεση στην πλατφόρμα.

Ο πίνακας διαχείρισης φοιτητή περιέχει δύο ενέργειες το edit και delete. Πατώντας το edit ο καθηγητής μπορεί να επεξεργαστεί τα στοιχεία κάποιου μαθητή, για παράδειγμα μπορεί να αλλάξει κάποιο στοιχείο όπως το τηλέφωνο επικοινωνίας μετά από συζήτηση με τον μαθητή. Η περίπτωση που ο χρήστης Vaggelis Petropoulos έχει λάθος κάποιο πεδίο πχ ΑΦΜ, ο καθηγητής μπορεί να αλλάξει το ΑΦΜ του και να περάσει το σωστό.



(image 73)

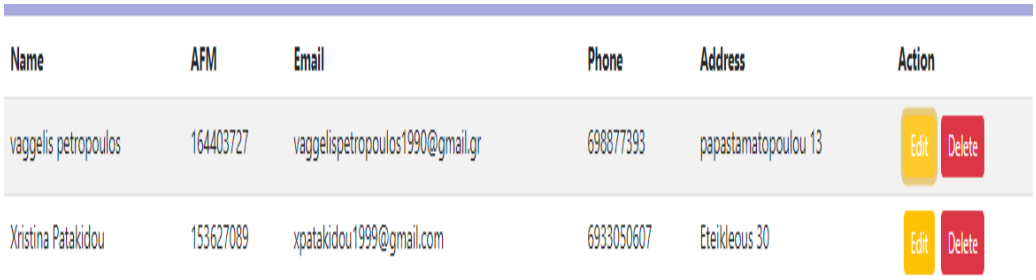
Κάνοντας update student ενημερώνεται αυτόματα η βάση δεδομένων του φοιτητή και αποθηκεύεται το σωστό ΑΦΜ στον πίνακα του φοιτητή.



(image 74)

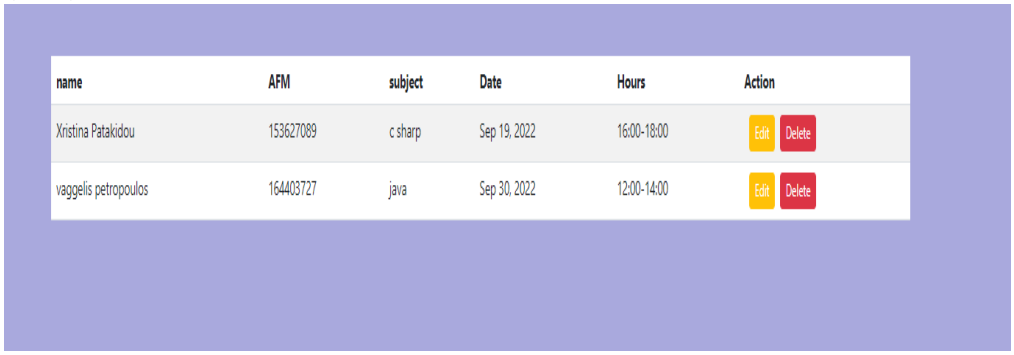
Επιπλέον υπάρχει και η δυνατότητα διαγραφής κάποιου φοιτητή από των πίνακα διαχείρισης φοιτητών. Αυτό συνήθως γίνεται σε περίπτωση που κάποιος φοιτητής ενημερώσει καθυστερημένα για την μη παρακολούθηση του προγράμματος και θέλει να ακυρώσει την εγγραφή από το σύστημα, τότε ενημερώνει τον καθηγητή ώστε να τον διαγράψει.

Για παράδειγμα αν ο Manos Fotiadis θέλει να ακυρώσει την εγγραφή από την πλατφόρμα, ο καθηγητής κάνοντας κλικ στο delete διαγράφει επιτυχώς τον συγκεκριμένο φοιτητή. Η βάση ενημερώνεται και επιστρέφει στον πίνακα τον φοιτητών όλους τους φοιτητές πέραν του Manos Fotiadis.



(image 75)

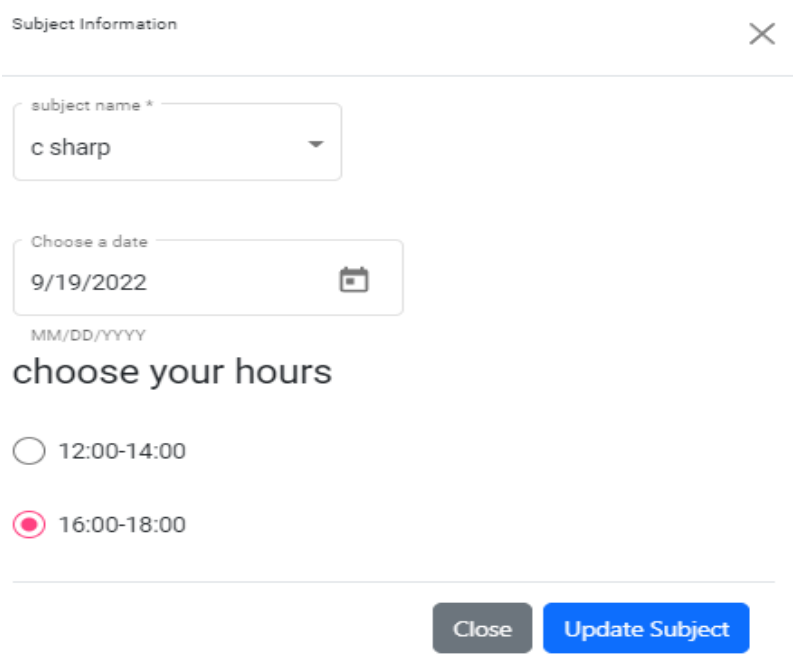
Την ίδια διαδικασία ακολουθεί ο καθηγητής για των πίνακα των μαθημάτων των φοιτητών.



(image 76)

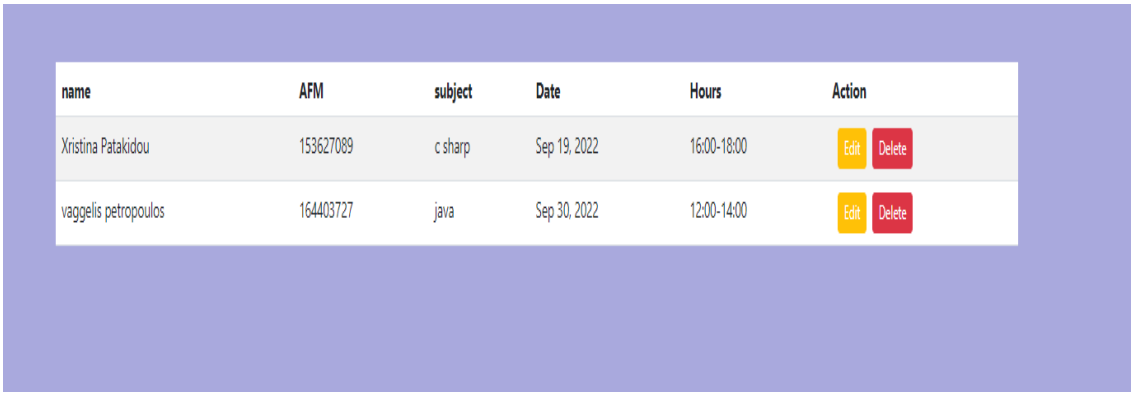
Στον πινάκα αυτόν ο καθηγητής βλέπει τα μαθήματα, τις ημερομηνίες και ώρες που έχει επιλέξει ένας μαθητής να παρακολουθήσει ένα συγκεκριμένο μάθημα, να δει το μάθημα που έχει επιλέξει και την ώρα προκειμένου να φτιάξει ένα τμήμα για εκείνη την ημέρα.

Σε περίπτωση που ένα τμήμα δεν συμπληρώσει τα άτομα που χρειάζεται ο καθηγητής ενημερώνει τον φοιτητή να αλλάξει την ημέρα και ώρα σε μια συγκεκριμένη ημέρα και ώρα με περισσότερα άτομα ή ενημερώνει τον φοιτητή ότι έχει αλλάξει ένα από τα παραπάνω πεδία.



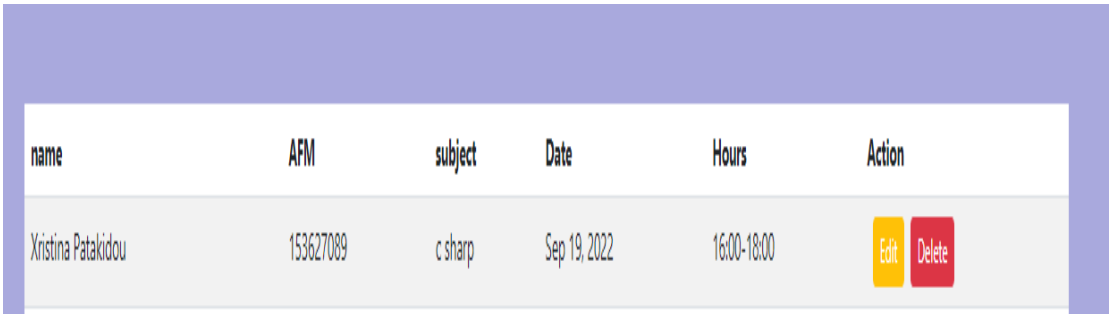
(image 77)

Στην παραπάνω εικόνα ο καθηγητής κάνει edit στον φοιτητή Xristina Patakidou και αλλάζει την ώρα του μαθήματος από 12:00-14:00 σε 16:00-18:00. Κάνει update subject και ενημερώνει τη βάση μαθημάτων για την αλλαγή ώρας. Η αλλαγή φαίνεται αυτόματα στον πίνακα μαθημάτων.



(image 78)

Στον παραπάνω πίνακα υπάρχει και η ενέργεια delete στην περίπτωση αυτή ο καθηγητής έχει την δυνατότητα να διαγράψει το μάθημα κάποιου φοιτητή ο οποίος είτε σταμάτησε να παρακολοθυεί το μάθημα είτε έχει αποχωρήση εντελώς από το τμήμα. Στις περιπτώσεις αυτές ο καθηγητής κάνοντας διαγραφή, αφαιρεί το όνομα του μαθητή καθώς και το πρόγραμμα το οποίο έχει επιλέξει από τον πίνακα.

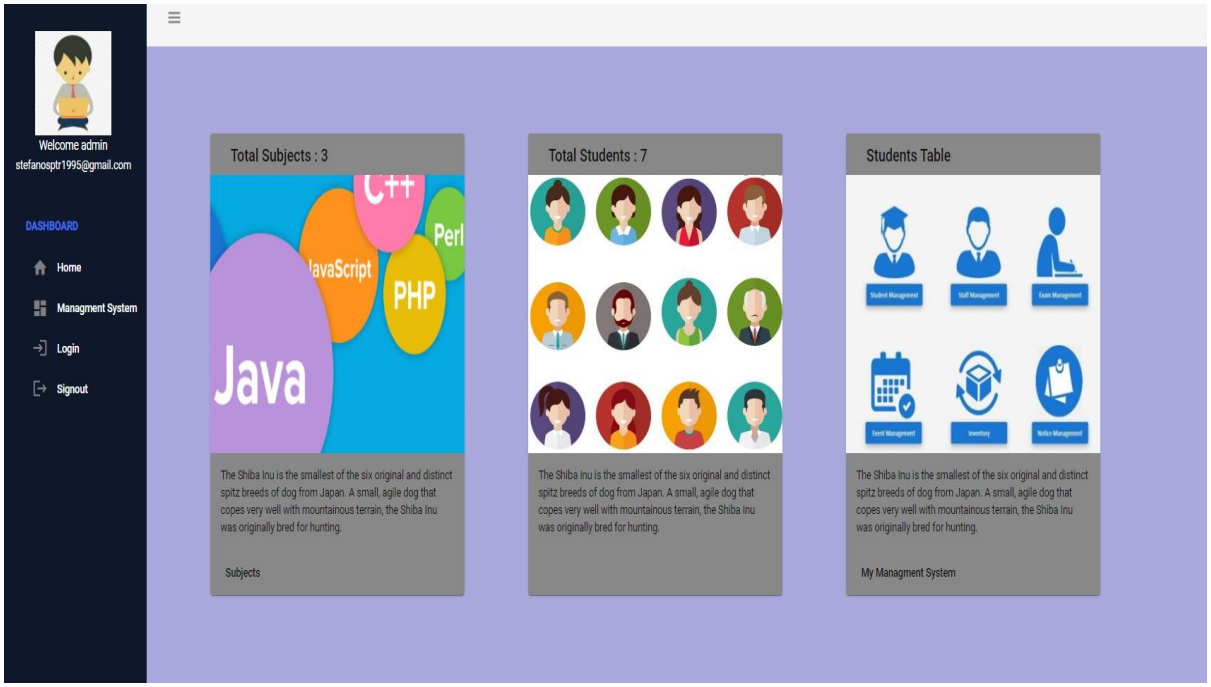


(image 79)

Στην παραπάνω εικόνα ο καθηγητής έχει διαγράψει τον φοιτητή Vaggeli Petropoulo από τον πίνακα μαθημάτων και έχει μείνει μόνο η φοιτήτρια Xristina Patakidou.

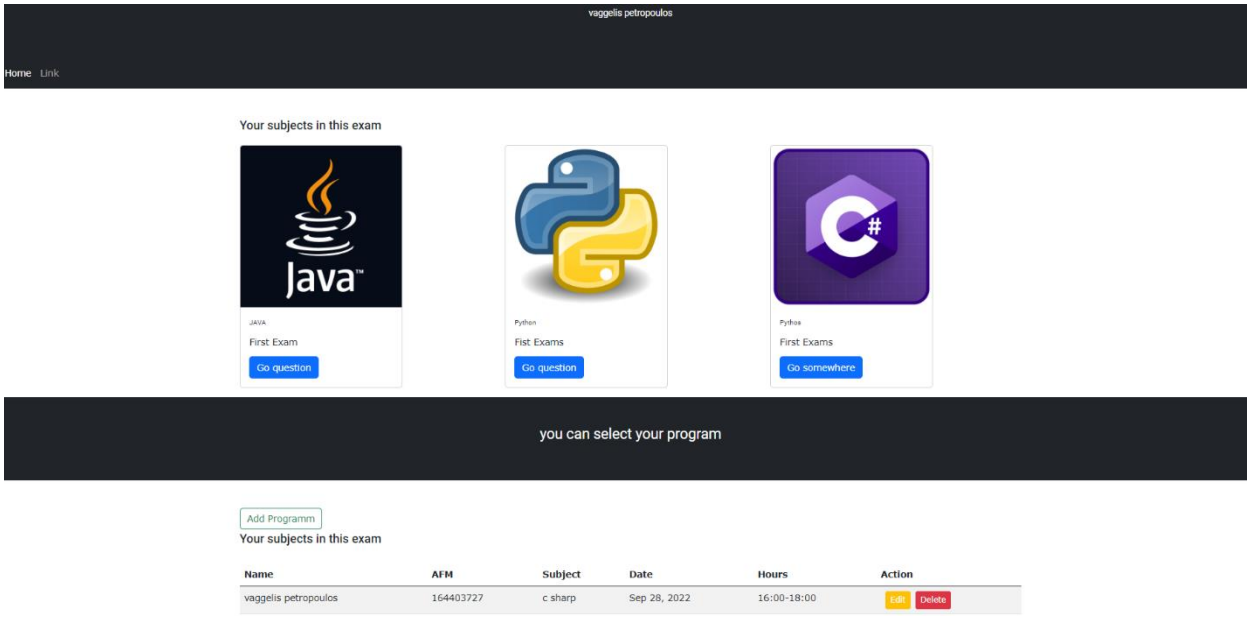
Με κάθε αλλαγή που κάνει ο καθηγητής στον πίνακα του προγράμματος για κάθε φοιτητή , η αλλαγή αυτή παρουσιάζεται και στον πίνακα του προγράμματος του φοιτητή. Για παράδειγμα αύτη την στιγμή αν ο χρήστης vaggelis Petropoulos συνδεθεί στην πλατφόρμα θα δει ότι στον πίνακα των μαθημάτων του δεν υπάρχει το πρόγραμμα που έχει επιλέξει. Αυτο γίνεται καθώς ενημερώνεται η βάση μαθημάτων και στην συνέχεια στέλνει τα αποτελέσματα στον πίνακα του προγράμματος.

Στο menu του καθηγητή παρουσιάζονται και τρεις καρτέλες μια για το σύνολο των μαθητών , μια για το σύνολο των μαθημάτων και μία για την προβολή των ερώτησεων τις οποίες έχει επιλέξει ο καθηγητής να παρουσιάζεται σε κάθε φοιτητή.



(image 80)

Κάνοντας κλικ στην πρώτη καρτέλα ο καθηγητής έχει πρόσβαση στον πίνακα των μαθητών και μπορεί να επεξεργαστεί τον πίνακα αυτον όπως αναφέραμε παραπάνω. Το ίδιο συμβαίνει και με τον πίνακα των φοιτητών. Με την επιλογή της καρτέλας student card ο καθηγητής έχει πρόσβαση και στις ερωτήσεις που επιλέγει για να παρουσιάσει τους φοιτητές.

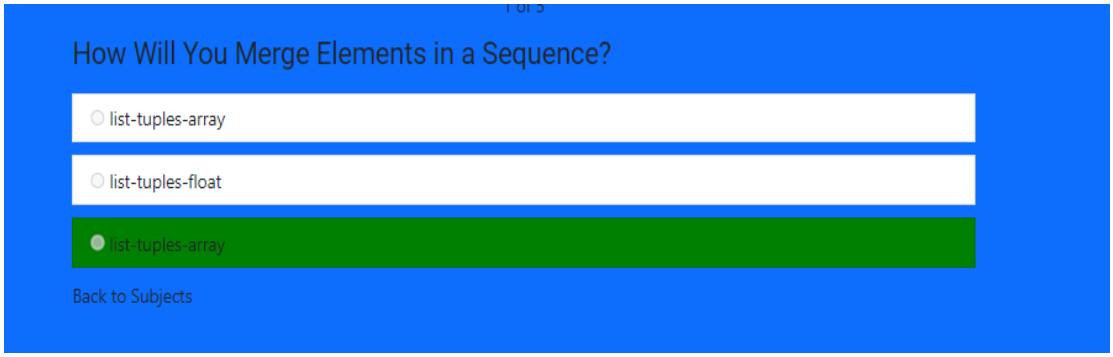


(image 81)

Στην σελίδα αυτή ο καθηγητής μπορεί να δεί ποιά μαθήματα είναι διαθέσημα αυτό το εξάμηνο και να επιλέξει κάποιες ερωτήσεις τις οποίες θέλει να προβάλονται στα μαθήματα του φοιτητή. Στην παραπάνω εικόνα βλέπουμε τρία μαθήματα java , python και c sharp τα οποία κάνοντας κλίκ σε μια απο τις τρεις καρτέλες , ο καθηγητής μπορεί να δεί και να αλλάξει η να προσθέσει κάποιες ερωτήσεις που έχει βάλει πριν λίγο καιρό σχετικά με το μάθημα. Κάνοντας κλικ ο καθηγητής στην καρτέλα της python ανοίγει η σελίδα python-question-page.

Έχουμε συνδεθεί σε ένα ερωτηματολόγιο στο οποίο πρέπει να απαντήσουμε πέντε ερωτήσεις σχετικές με την δομή και λειτουργία της python. Οι ερωτήσεις αυτές έχουν τυχαία επιλογή κάθε φορά και ο χρήστης έχει την δυνατότητα αφου επιλέξει μια ερώτηση να αλλάξει την επιλογή του μέσα σε τρία δευτερόλεπτα ,αλλίως προχωράει αυτόματα στην δεύτερη ερώτηση.

Αν η ερώτηση είναι σώστη εμφανίζεται με πράσινο χρώμα σε αντίθετη περίπτωση εμφανίζεται με κόκκινο χρώμα.



(image 82)

Η δεύτερη εικόνα δείχνει μια λάθος απάντηση απο τον χρήστη.

(image 83)

Αν ο καθηγητής θέλει να αποχωρήσει από τις ερωτήσεις κάθως βλέπει ότι η νέα προσθήκη των ερωτήσεων που έβαλε πέρασαν επιτυχώς υπάρχει ένα button back to

Subject κάτω απο τα πεδία επιλογών όπου ο καθηγητής μπόρει να το κλικάρει και να συνδεθεί ξανά στο home page. Η ίδια διαδικασία ισχύει και για τα υπόλοιπα μαθήματα ,στα οποία ο καθηγητής μπορεί να επιλέξει τις ερωτήσεις που θέλει , να τις αλλάξει απο την βάση των ερωτήσεων. Με αυτο το τρόπο ο φοιτητής έχει στην διαθεσή του κάθε φορά νέες ερωτήσεις και μπορεί να τις απαντήσει και να δει τα αποτελέσματα του. Με αυτό τον τρόπο ολοκληρώνετε η παρουσίαση της εφαρμογής για κάθε χρήστη. Στην συνέχεια θα αναφερθούμε σε συμπεράσματα που προέκειψαν απο την σχεδίαση και υλοποίηση της πλατφόρμας.

5.Συμπεράσματα και αποτελέσματα

Στόχοι της παρούσας εργασίας ήταν η ανάπτυξη του λογισμικού διαχείρισης του προγράμματος των φοιτητών και της εγγραφής φοιτητών στην πλατφόρμα. Η συγκεκριμένη τεχνολογία, λοιπόν, χρησιμοποιήθηκε ως το μέσο ή το εργαλείο για την διευκόλυσνη της οργάνωσης του τμήματος και την καλύτερη διαχείρηση των φοιτητών , με κύριο στόχο την ικανοποίηση του κάθε φοιτητή που έχει εγγραφεί στην πλατφόρμα.

Η συγκεκριμένη τεχνολογία, λοιπόν, χρησιμοποιήθηκε ως το μέσο ή το εργαλείο για την ηλεκτρονική δραστηριοποίηση στο χώρο αυτό, μέσα από τη διαδικτυακή εφαρμογή που αναπτύχθηκε.

Από την άλλη, επιτεύχθηκε η κατανόηση όλων των συντεταγμένων της τεχνολογίας της Angular από το τεχνικό υπόβαθρο και τα βασικά χαρακτηριστικά της, μέχρι την εύστοχη χρήση όλων των δυνατοτήτων της. Άλλωστε, αντιμετωπίστηκαν ζητήματα που μόνο μέσα από την εφαρμογή γίνονται απόλυτα αντιληπτά, όπως θέματα απαιτήσεων λογισμικού και διασυνδεσιμότητας.

Επιπλέον, έγιναν αντιληπτά και τα πλεονεκτήματα από τη χρήση της εν λόγω τεχνολογίας:

- Μειωμένες απαιτήσεις σε κώδικα
- Δυνατότητα ανάπτυξης της λογικής ανεξάρτητα από τις σελίδες
- Εύκολη σχεδίαση των σελίδων με χρήση έτοιμων βιβλιοθηκών και χρήση bootstrap

Φτάνοντας στα παραδοτέα της διπλωματικής μου εργασίας, πέρα από το παρόν έγγραφο που αποτελεί και την τεκμηρίωσή της, περιλαμβάνεται σε αυτά cd που περιέχει το λογισμικό, την βάση καθώς και τα εργαλεία τα οποία χρειάστηκαν για την κατασκευή της συγκεκριμένης εφαρμογής.

Συγκεκριμένα, στο usb περιέχονται:

Φάκελος αρχείων της εφαρμογής, όπως ακριβώς αυτή αναπτύχθηκε με τον Visual studio code, τα αρχεία κώδικα , τις σελίδες, τις βιβλιοθήκες, τη διαμόρφωση και γενικότερα όλα τα απαιτούμενα αρχεία.

Τέλος, όσον αφορά την υλοποίηση της εφαρμογής, επικεντρωθήκαμε στην τεχνολογία ανάπτυξης και στην κάλυψη των λειτουργικών προδιαγραφών, οπότε και υπάρχει χώρος για βελτιώσεις και προσθήκες. Άλλωστε, τέτοιου είδους εφαρμογές πάντοτε εξελίσσονται και προσαρμόζονται με τον καιρό σε νέα δεδομένα.

Παραθέτουμε, λοιπόν, σε αυτό το σημείο ιδέες και προτάσεις για περαιτέρω εργασία:

Ενημέρωση και άλλων χρηστών ότι υπάρχει αυτός ο ιστότοπος έστω και απλά για περιήγηση στην πλατφόρμα (ως επισκέπτες).

Θα μπορούσε να προστεθεί η δυνατότητα εμφάνισης στατιστικών στοιχείων όπως για παράδειγμα διάγραμμα ανα έτος ή ανα μήνα των φοιτητών που έκαναν νέα εγγραφή.

Θα μπορούσε να προστεθεί η δυνατότητα εμφάνισης στατιστικών στοιχείων όπως για παράδειγμα progress bar για κάθε φοιτητή σχετικά με τα αποτελέσματα του στις ερωτήσεις των μαθημάτων, τα οποία αποτελέσματα θα μπορεί να τα δει και ο καθηγητής στον δικό του πίνακα.

Θα μπορούσε να προστεθεί η δυνατότητα εμφάνισης κάποιον νέων παιχνιδιών ετσι ώστε η πλατφόρμα να γίνει πιο ελκυστική για τους φοιτητές.

Τέλος η γραμματεία θα μπορούσε να αποτελέσει ένα νέο χρήστη , ο οποιος θα έχει συγκεκριμένες αρμοδιότητες σχετικά με την οργάνωση των φοιτητών και τον μαθημάτων, με σκοπό τον καλύτερο συντονισμό του τμήματος.

6. Βιβλιογραφία

ΑΡΘΡΑ

[1] Joshi, B. (2019). Angular. In: Beginning Database Programming Using ASP.NET Core 3

- Published12 December 2019
- Publisher NameApress, Berkeley, CA

[2] Kotaru, V.K. (2020). Angular: HTTP Client. In: Angular for Material Design. Apress, Berkeley,

- Published10 December 2019
- Publisher NameApress, Berkeley, CA

[3] Chasseur, C. et al. – Enabling JSON Document Stores in Relational Systems, WebDB, 2013.

ΒΙΒΛΙΑ

[4] Angular Services "Design state-of-the-art applications with customized Angular services.

- Published February 2017
- Publisher Name by Packt Publishing Ltd

ΚΕΙΜΕΝΟ ΣΕ ΔΙΑΚΤΥΑΚΟ ΤΟΠΟ

[5] https://angular.io/guide/typescript-configuration

[6] https://getbootstrap.com/docs/5.2/getting-started/introduction/

[7] https://www.w3schools.com/

[8] https://www.digitalocean.com/community/tutorials/json-server