ΠΑΝΕΠΙΣΤΗΜΙΟ ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΑΞΙΟΛΟΓΗΣΗ ΚΑΙ ΠΙΣΤΟΠΟΙΗΣΗ ΓΝΩΣΕΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVASCRIPT

Κωνσταντίνος Καπνιάς

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

ΥΠΕΥΘΥΝΟΣ

Τζιάλλας Γρηγόριος
Καθηγητής

Λαμία, 4 Μαρτίου έτος 2024

ΠΑΝΕΠΙΣΤΗΜΙΟ
ΘΕΣΣΑΛΙΑΣ

ΣΧΟΛΗ ΘΕΤΙΚΩΝ ΕΠΙΣΤΗΜΩΝ

ΤΜΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

# ΔΙΑΔΙΚΤΥΑΚΗ ΕΦΑΡΜΟΓΗ ΓΙΑ ΤΗΝ ΑΞΙΟΛΟΓΗΣΗ ΚΑΙ ΠΙΣΤΟΠΟΙΗΣΗ ΓΝΩΣΕΩΝ ΠΡΟΓΡΑΜΜΑΤΙΣΜΟΥ JAVASCRIPT

Κωνσταντίνος Καπνιάς


UNIVERSITY OF
THESSALY

SCHOOL OF SCIENCE

DEPARTMENT OF COMPUTER SCIENCE & TELECOMMUNICATIONS

# WEB APPLICATION FOR ASSESSMENT AND KNOWLEDGE CERTIFICATION OF JAVASCRIPT EXERCISES

Konstantinos Kapnias

FINAL THESIS

ADVISOR

Tziallas Grigorios
Professor

Lamia, March 4th year 2024

*«Με ατομική μου ευθύνη και γνωρίζοντας τις κυρώσεις [(1)], που προβλέπονται από της διατάξεις της παρ. 6 του άρθρου 22 του Ν. 1599/1986, δηλώνω ότι:*

*1. Δεν παραθέτω κομμάτια βιβλίων ή άρθρων ή εργασιών άλλων αυτολεξεί **χωρίς να τα περικλείω σε εισαγωγικά** και χωρίς να αναφέρω το συγγραφέα, τη χρονολογία, τη σελίδα. Η αυτολεξεί παράθεση χωρίς εισαγωγικά χωρίς αναφορά στην πηγή, είναι λογοκλοπή. Πέραν της αυτολεξεί παράθεσης, λογοκλοπή θεωρείται και η παράφραση εδαφίων από έργα άλλων, συμπεριλαμβανομένων και έργων συμφοιτητών μου, καθώς και η παράθεση στοιχείων που άλλοι συνέλεξαν ή επεξεργάσθηκαν, χωρίς αναφορά στην πηγή. Αναφέρω πάντοτε με πληρότητα την πηγή κάτω από τον πίνακα ή σχέδιο, όπως στα παραθέματα.*

*2. Δέχομαι ότι η αυτολεξεί **παράθεση χωρίς εισαγωγικά**, ακόμα κι αν συνοδεύεται από αναφορά στην πηγή σε κάποιο άλλο σημείο του κειμένου ή στο τέλος του, είναι αντιγραφή. Η αναφορά στην πηγή στο τέλος π.χ. μιας παραγράφου ή μιας σελίδας, δεν δικαιολογεί συρραφή εδαφίων έργου άλλου συγγραφέα, έστω και παραφρασμένων, και παρουσίασή τους ως δική μου εργασία.*

*3. Δέχομαι ότι υπάρχει επίσης περιορισμός στο μέγεθος και στη συχνότητα των παραθεμάτων που μπορώ να εντάξω στην εργασία μου εντός εισαγωγικών. Κάθε μεγάλο παράθεμα (π.χ. σε πίνακα ή πλαίσιο, κλπ), προϋποθέτει ειδικές ρυθμίσεις, και όταν δημοσιεύεται προϋποθέτει την άδεια του συγγραφέα ή του εκδότη. Το ίδιο και οι πίνακες και τα σχέδια*

*4. Δέχομαι όλες τις συνέπειες σε περίπτωση λογοκλοπής ή αντιγραφής.*

Ημερομηνία: 4/4/2024

Ο – Η Δηλ.
*Κωνσταντίνος Καπνιάς*

*(1) «Όποιος εν γνώσει του δηλώνει ψευδή γεγονότα ή αρνείται ή αποκρύπτει τα αληθινά με έγγραφη υπεύθυνη δήλωση*
*του άρθρου 8 παρ. 4 Ν. 1599/1986 τιμωρείται με φυλάκιση τουλάχιστον τριών μηνών. Εάν ο υπαίτιος αυτών των πράξεων*
*σκόπευε να προσπορίσει στον εαυτόν του ή σε άλλον περιουσιακό όφελος βλάπτοντας τρίτον ή σκόπευε να βλάψει άλλον, τιμωρείται με κάθειρξη μέχρι 10 ετών.»*

## ΠΕΡΙΛΗΨΗ

Σκοπός της πτυχιακής αυτής είναι η δημιουργία μίας διαδικτυακής εφαρμογής η οποία αξιολογεί εργασίες JavaScript ως προς τη σύνταξη και την ορθότητα των αποτελεσμάτων τους. Στοχεύει τη διευκόλυνση καθηγητών να βαθμολογούν εφαρμογές φοιτητών, προσφέροντας μια φιλική και γρήγορη διεπαφή (UI) στο χρήστη. Η εφαρμογή είναι γραμμένη με React.js για το front end, Express.js για το back end, και MySQL για τη βάση δεδομένων.

## ABSTRACT

The goal of this thesis is to create an online application that evaluates JavaScript assignments based on their syntax and result correctness. It aims to assist teachers in grading student applications, providing a user-friendly and fast interface (UI). The application is built with React.js for the front end, Express.js for the back end, and MySQL for the database.

## Table of Contents

# ΚΕΦΑΛΑΙΟ 1 Εισαγωγή

Αντικείμενο της πτυχιακής εργασίας είναι η κατασκευή και ανάπτυξη μίας διαδικτυακής εφαρμογής που μπορεί να χρησιμοποιείται από φοιτητές και καθηγητές μίας σχολής, προκειμένου να ανεβαίνουν και να αξιολογούνται αυτόματα εργασίες της γλώσσας JavaScript. Κύριος στόχος αυτής της εργασίας είναι η αυτοματοποίηση του ελέγχου απλών προγραμμάτων, καλύπτοντας όσο περισσότερες περιπτώσεις είναι δυνατό και δίνοντας έμφαση στην ασφάλεια σε όλα τα επίπεδα του προγραμματισμού.

Η εργασία κατασκευάστηκε με TypeScript για μεγαλύτερη ευκρίνεια σε όλα τα στάδια του προγραμματισμού, καθώς και frameworks για το back end και το front end, Express και React αντίστοιχα. Η βάση δεδομένων είναι γραμμένη σε MySQL με το επώνυμο της Workbench.

# ΚΕΦΑΛΑΙΟ 2 Προγραμματισμός Διαδικτύου

## (Υποκεφάλαιο 2.1) JavaScript και TypeScript

Η εποχή του διαδικτύου είναι πολύ συχνά συνώνυμη με την εποχή της JavaScript, η οποία είναι μια υψηλού επιπέδου, client-side γλώσσα προγραμματισμού που ευθύνεται για τη λογική και την λειτουργικότητα των ιστοσελίδων, δίνοντας μεγάλη ευελιξία στη κατασκευή τους.

Ιστορικά, η JavaScript ξεκίνησε με την ιδέα να αφαιρεθεί από το διαδίκτυο ο περιορισμός των ιστοσελίδων στο να προσφέρουν ένα διαδραστικό περιβάλλον το 1995. Ως τότε, οι σελίδες του διαδικτύου δε μπορούσαν να αλλάξουν περιεχόμενο με το που φόρτωναν την πρώτη φορά. Παρά την ίσως παραπλανητική της ονομασία, η JavaScript δε συσχετίζεται με τη Java, αλλά στα σχέδια των πρώτων προγραμματιστών της ήταν η ενσωμάτωση της Java στο διαδίκτυο.[1]

Η JavaScript είναι weakly typed γλώσσα, πράγμα το οποίο σημαίνει πως δεν έχει τύπους στις μεταβλητές της. Το type casting γίνεται με έμμεσο τρόπο, απλά αναθέτοντας μια νέα μεταβλητή σε ήδη υπάρχουσα άλλου τύπου, όπως φαίνεται στην παρακάτω εικόνα. Εδώ μία μεταβλητή μετατρέπεται από αλφαριθμητική, σε αριθμό, σε boolean.[2]

```
1   let x = "hello world"
2   x = 5
3   x = true
```

Αυτή η ευελιξία στη σύνταξη οδήγησε στη δυσκολία χρήσης της γλώσσας σε μεγάλης έκτασης πρότζεκτ, και την ανάγκη για μια λύση που δεν έσπαγε τη συμβατότητα και την ιδεολογία της γλώσσας. Για αυτό και βγήκε μια “επέκταση” της, που προσθέτει στατικούς τύπους και μεταβλητές. Αυτή η γλώσσα, γνωστή ως TypeScript ή “JavaScript με τύπους”, έχει κατασκευαστεί από τη Microsoft δεκαεπτά χρόνια μετά την πρώτη κυκλοφορία της JavaScript.[3] Ένα παράδειγμα χρήσης της με τον κώδικα που χρησιμοποιήθηκε παραπάνω ακολουθεί σε εικόνα, όπου όπως φαίνεται, μετά τον αρχικό ορισμό του **x** σαν αλφαριθμητική μεταβλητή, ο μεταγλωττιστής αναγνωρίζει ως λάθος το έμμεσο type casting.

```
1   let x: string = "hello world"
2   x = 5
3   x = true
```

περαιτέρω, η διασημότητα των δύο γλωσσών οδήγησε στη δημιουργία πολλαπλών frameworks που τις επεκτείνουν, μερικά από τα πιο γνωστά στη σήμερα εποχή παραδείγματα να είναι η Angular, jQuery, React και Node.

## (Υποκεφάλαιο 2.2) Τεχνολογίες που χρησιμοποιήθηκαν

Η εφαρμογή που διαπραγματεύεται η πτυχιακή κάνει χρήση των τελευταίων δύο προαναφερόμενων frameworks, ονόματι React και Node, και πιο συγκεκριμένα ένα framework του Node, το Express.

### (Ενότητα 2.2.1) Node και Express

Το Node.js είναι ένα περιβάλλον εκτέλεσης JavaScript ανοικτού κώδικα που χρησιμοποιείται για την εκτέλεση κώδικα από πλευράς εξυπηρετητή (server-side). Χρησιμοποιείται για την κατασκευή και ανάπτυξη ενός back end που υποστηρίζει την ιδεολογία “παντού JavaScript” ενώνοντας τον διαδίκτυο σε μία γλώσσα.[5]

Η κύρια ιδέα είναι η χρήση ενός ασύγχρονου I/O (Input/Output) το οποίο κάνει δυνατό τον χειρισμό πολλαπλών ταυτόχρονων αιτημάτων. Με άλλα λόγια, το καθιστά ιδανικό και γρήγορο για εφαρμογές που απαιτούν υψηλή απόδοση και κλιμακωσιμότητα σε πραγματικό χρόνο.[4]

Όπως και η JavaScript, έτσι και το Node έχει frameworks που διευκολύνουν την ανάπτυξη API, μερικά δημοφιλή εκ των οποίων Express, Koa και Nest. Στα πλαίσια της πτυχιακής έχει γίνει χρήση του Express, που είναι και το πιο γνωστό.

Το Express έχει χτιστεί με την ιδεολογία εφαρμογών που κάνουν χρήση της αρχιτεκτονικής RESTful API. Παρέχει όλα τα απαραίτητα εργαλεία και τις βιβλιοθήκες για τη δημιουργία ενός API, και τη σύνδεση του με ένα διακομιστή (server), όπως δρομολογητές, middleware και διαχείρηση στατικών αρχείων HTML και CSS.[6]

### (Ενότητα 2.2.2) React

Παρόλο που η React δεν είναι —τεχνικά μιλώντας— framework αλλά βιβλιοθήκη, η έκταση της την κάνει να αντιμετωπίζεται ευρέως ως framework. Δημιουργήθηκε από την Meta (πρώην Facebook) και προσφέρει έναν αποτελεσματικό τρόπο για την κατασκευή επαναχρησιμοποιήσιμων UI στοιχείων. Χρησιμοποιείται για τη δημιουργία σύγχρονων εφαρμογών ιστού, όπου οι αλλαγές στα δεδομένα αντανακλώνται αυτόματα στον χρήστη χωρίς την ανάγκη ανανέωσης της σελίδας.[7]

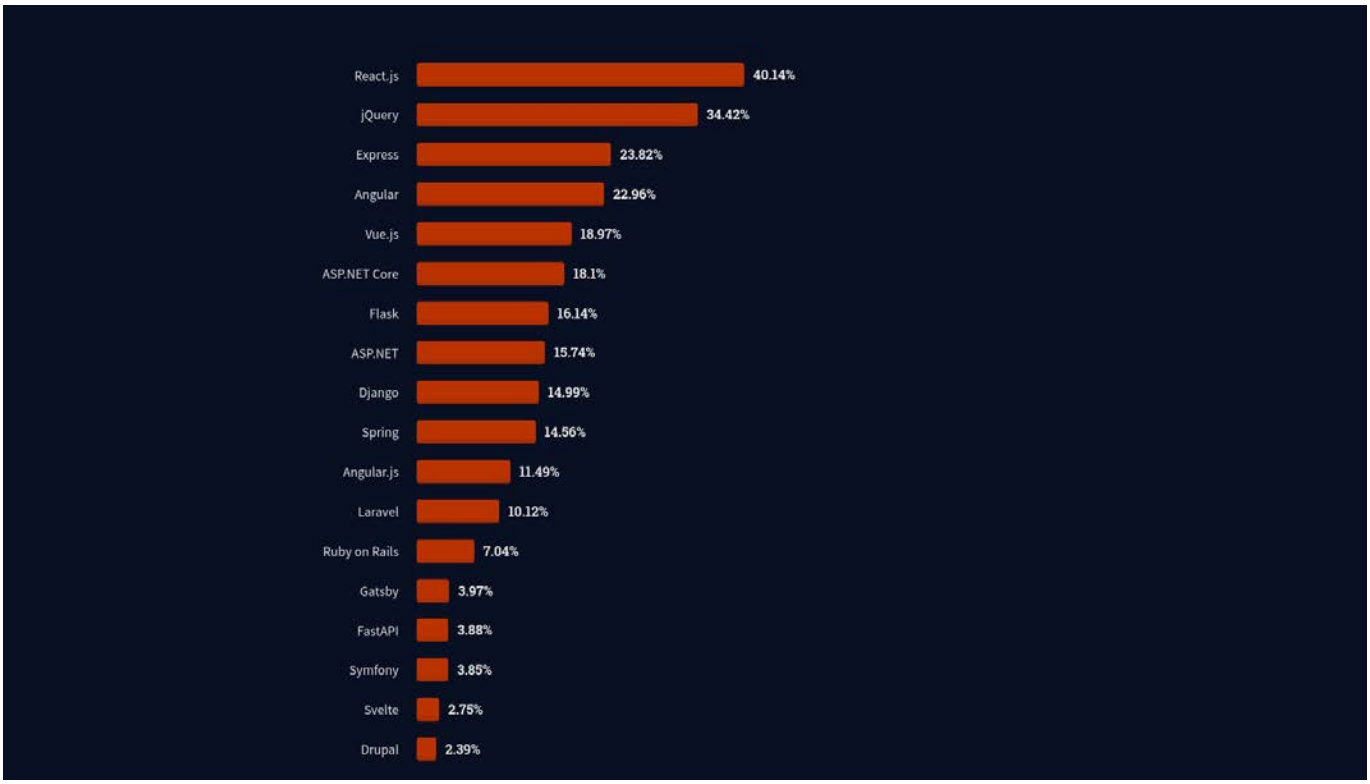
Η React συνεργάζεται συχνά με άλλα frameworks, όπως το Next.js, αλλά δεν είναι επίσης σπάνια και η κατασκευή ιστοσελίδων που βασίζονται μόνο στη React και τις βιβλιοθήκες της.

Ακολουθεί επίσης την προσέγγιση του δηλωτικού προγραμματισμού (declarative programming), σύμφωνα με την οποία οι προγραμματιστές καθορίζουν τις επιθυμητές καταστάσεις ή αποτελέσματα του προγράμματος, και η εκτελέσιμη λογική πίσω από αυτές

υλοποιείται από το σύστημα. Αυτό σημαίνει ότι ο προγραμματιστής καθορίζει δηλώσεις, κανόνες ή περιγραφές για τις επιθυμητές συμπεριφορές του προγράμματος, και ο εκτελεστής (ή άλλο μέσο) επιλύει τις λεπτομέρειες για το πώς ακριβώς θα πραγματοποιηθεί η υλοποίηση.

Περαιτέρω βασίζεται σε ένα μοντέλο ανάπτυξης που ονομάζεται βασισμένο σε στοιχεία (component-based), όπου μικρά στοιχεία (components) αντιπροσωπεύουν ένα συγκεκριμένο κομμάτι της λειτουργικότητας ή του περιεχομένου μιας εφαρμογής (όπως ένα κουμπί, ένα μενού ή μια φόρμα). Ο σκοπός των στοιχείων είναι όπως προαναφέρθηκε η “ανακύκλωση” και επαναχρησιμοποίηση τους σε διάφορα μέρη μιας εφαρμογής, κάνοντας τον κώδικα πιο οργανωμένο.[8]

Μέχρι και σήμερα, η React παραμένει ένα από τα κορυφαία frameworks για JavaScript. Συγκεκριμένα, με έρευνα του 2021 σε 80.000 προγραμματιστές διαφόρων χωρών, τα στατιστικά έβαλαν τη React ως κορυφαίο web framework με 40,14% των χρηστών να την προτιμούν.[9]




## (Ενότητα 2.2.3) MySQL

Η MySQL είναι ανοιχτού κώδικα σύστημα διαχείρισης σχεσιακών βάσεων δεδομένων που κυκλοφόρησε αρχικά το 1995, και χρησιμοποιείται από αρκετούς μεγάλους οργανισμούς σαν το Facebook, το X (πρώην Twitter) και την Amazon.[10]

Όπως εξηγείται και στο όνομά της, η MySQL κάνει χρήση της γλώσσας ερωτημάτων SQL (Structured Query Language) για τον χειρισμό των δεδομένων. Το όνομα της βέβαια δεν

έρχεται από το άρθρο ιδιοκτησίας “my” αλλά από το όνομα της κόρης του συνιδρυτή Monty Widenius, τη My.[11]

Μέχρι και το Φεβρουάριο 2024, η MySQL είναι ένα από τα κορυφαία και πιο δημοφιλή συστήματα διαχείρισης βάσεων δεδομένων, παραμένοντας σε δεύτερη θέση κατάταξης σε έρευνες όπως και φαίνεται παρακάτω.[12]

| Rank | | | DBMS | Database Model | Score | | |
|---|---|---|---|---|---|---|---|
| Feb 2024 | Jan 2024 | Feb 2023 | | | Feb 2024 | Jan 2024 | Feb 2023 |
| 1. | 1. | 1. | Oracle | Relational, Multi-model | 1241.45 | -6.05 | -6.08 |
| 2. | 2. | 2. | MySQL | Relational, Multi-model | 1106.67 | -16.79 | -88.78 |
| 3. | 3. | 3. | Microsoft SQL Server | Relational, Multi-model | 853.57 | -23.03 | -75.52 |
| 4. | 4. | 4. | PostgreSQL | Relational, Multi-model | 629.41 | -19.55 | +12.90 |
| 5. | 5. | 5. | MongoDB | Document, Multi-model | 420.36 | +2.88 | -32.41 |
| 6. | 6. | 6. | Redis | Key-value, Multi-model | 160.71 | +1.33 | -13.12 |
| 7. | 7. | ↑ 8. | Elasticsearch | Search engine, Multi-model | 135.74 | -0.33 | -2.86 |
| 8. | 8. | ↓ 7. | IBM Db2 | Relational, Multi-model | 132.23 | -0.18 | -10.74 |
| 9. | 9. | ↑ 12. | Snowflake | Relational | 127.45 | +1.53 | +11.80 |
| 10. | ↑ 11. | ↓ 9. | SQLite | Relational | 117.28 | +2.08 | -15.38 |

Υπάρχει στη συνέχεια και το επίσημο εργαλείο σχεδιασμού βάσεων δεδομένων MySQL Workbench το οποίο προσφέρει μια γραφική διεπαφή στο χρήστη για τη σχεδίαση, ανάπτυξη και συντήρηση μιας βάσης SQL, αυτοματοποιώντας πολλές διαδικασίες και φέρνοντας τον προγραμματιστή σε πολύ λιγότερη επαφή με την ίδια την SQL. Βγήκε ως διάδοχος του DBDesigner 4 της fabForce.net και συμβολικά για αυτό, η αρίθμηση των εκδόσεων του ξεκίνησε από το 5.0.[13]

# ΚΕΦΑΛΑΙΟ 3 Εφαρμογές Αξιολόγησης Προγραμματισμού

## (Υποκεφάλαιο 3.1) State of the art

Καθώς η επιστήμες εξελίσσονται, τόσο εξελίσσεται και η ανάγκη να έχουν οι μαθητευόμενοι κάποιο υπόβαθρο και πληθώρα ικανοτήτων. Διαρκώς βγαίνουν εργαλεία που βοηθούν τόσο τους εκπαιδευόμενους όσο και τους εκπαιδευτές στο να σχηματίσουν και να διατηρήσουν αυτό το υπόβαθρο. Ένα από αυτά είναι οι εφαρμογές αυτόματης αξιολόγησης εργασιών, δηλαδή προγράμματα που μέσω αλγορίθμων ελέγχουν την ορθότητα των απαντήσεων σε προβλήματα τόσο θεωρητικά όσο και σε πρακτικά. Το φάσμα τέτοιων εφαρμογών επεκτείνεται από αυτές που ελέγχουν ερωτήσεις πολλαπλών επιλογών ή συμπλήρωσης κενού, μέχρι και εκείνες που τρέχουν συγκεκριμένες ασκήσεις δοσμένες από τον εκπαιδευτή, οι οποίες μέσω ενός αλγορίθμου προσπαθούν να προβλέψουν τα λάθη του εκπαιδευόμενου.

Οι κλάδοι που έχουν εφαρμογές αυτόματης αξιολόγησης είναι άφθονοι, κι ένας από αυτούς που είναι άξιος μελέτης είναι ο προγραμματισμός. Καθώς οι τεχνολογίες στην πληροφορική αλλάζουν τακτικά, εφευρίσκονται και διάφοροι τρόποι να ελέγχεται κώδικας, όπως για παράδειγμα μέσω testing frameworks, τα οποία βοηθάνε τον προγραμματιστή να βρει συντακτικά και λογικά λάθη. Όμως κάτι τέτοιο πέρα από εργαλείο για προγραμματιστές θα ήταν χρήσιμο κι ως εργαλείο για εκπαίδευση το οποίο ελέγχει και βαθμολογεί τον κώδικα φοιτητών. Ο σκοπός τέτοιων εργαλείων είναι τόσο η διευκόλυνση ενός εκπαιδευόμενου να μάθει πιο γρήγορα από τα λάθη του, καθώς οι ασκήσεις διορθώνονται με μεγαλύτερη ταχύτητα και, ανάλογα τον αλγόριθμο, σε πραγματικό χρόνο είτε με ελάχιστη επιμέλεια από κάποιον διδάσκων, όσο και η διευκόλυνση του διδάσκων στο να διαχειρίζεται τεράστιους όγκους εργασιών σε προσιτό χρόνο, αφού ένας αλγόριθμος μπορεί να τον κατευθύνει στα μέρη που χρειάζεται να προσηλωθεί.

Σε ρεαλιστικά σενάρια βέβαια, είναι ανέφικτο από ένα πρόγραμμα που ελέγχει πολύπλοκα προβλήματα όπως μαθηματικά ή κώδικα να καλύψει τις άπειρες περιπτώσεις προβλημάτων που υπάρχουν, καθώς η αλγοριθμοποίηση της αξιολόγησης στοχεύει στην πρόβλεψη λαθών σε συγκεκριμένα σενάρια. Έτσι υπάρχει πάντα ο κίνδυνος να μην υπολογιστεί σωστά ένα σενάριο ή το να υπάρξει ανθρώπινο λάθος σε μια πρόβλεψη. Γιαυτό κιόλας δεν υπάρχουν ακόμα τέλεια προγράμματα αυτόματης αξιολόγησης κώδικα για όλες τις γλώσσες, κι όλα κοιτάνε να καλύψουν ένα μέρος του φάσματος του προγραμματισμού, γυρνώντας γύρω από μία ή μερικές γλώσσες αντί για όλες.

## (Υποκεφάλαιο 3.2) Παράδειγμα Εφαρμογής: JS-test

Μία απλή εφαρμογή αυτόματης αξιολόγησης είναι το ανοιχτού κώδικα js-test που είναι ανεβασμένη στο GitHub. Η εφαρμογή είναι φτιαγμένη για να τρέχει ως ιστοσελίδα γραμμένη με JavaScript, και όπως επεξηγεί το όνομα του εστιάζει σε αυτόματη αξιολόγηση της JavaScript μονάχα, και πιο συγκεκριμένα σε διαγωνίσματα ή μικρά κουίζ.

Λειτουργεί με τον εξής τρόπο: Ο καθηγητής φτιάχνει τις ερωτήσεις, απαντήσεις και προαιρετικά έναν αλγόριθμο βαθμολόγησης σε ένα εξωτερικό αρχείο .md το οποίο ακολουθεί μια συγκεκριμένη σύνταξη. Είναι δυνατό να υπάρξουν πολλές εκδοχές της ίδιας ερώτησης ώστε να εμφανίζονται με τυχαία σειρά στους φοιτητές για να αποφευχθεί η αντιγραφή. Στη συνέχεια, ο κώδικας διαβάζει το αρχείο και ανεβάζει τις ερωτήσεις στο front end που βλέπει ο μαθητής, όπου και μπορούν να συμπληρωθούν. Αποτελούνται είτε από πολλαπλής επιλογής, είτε συγγραφή απλών εντολών κώδικα.

Καθώς η εφαρμογή δεν έχει κάποιο δικό της back end ή ανεβασμένη βάση που να την υποστηρίζει, δίνεται η δυνατότητα να ανέβουν τα διαγωνίσματα σε δύο συγκεκριμένα back ends. Πιο συγκεκριμένα, ένα από αυτά είναι το Firebase, στο οποίο ο διδάσκων συνδέει την εφαρμογή και φτιάχνει μια βάση δεδομένων, και στη συνέχεια αφού δώσει τα απαραίτητα link στους φοιτητές, μπορεί να κατεβάσει τις απαντήσεις τους σε μορφή αρχείου JSON. Ένα άλλο back end είναι το Google Forms, από το οποίο ο διδάσκων μπορεί να πάρει τις απαντήσεις των φοιτητών σε αρχείο CSV το οποίο θα χρειαστεί στη συνέχεια να μετατρέψει σε JSON.

Αφού συλλεχθούν οι απαντήσεις των φοιτητών και μετατραπούν σε JSON, ο καθηγητής μπορεί με μια εντολή στο τερματικό να τις περάσει από αυτόματη βαθμολόγηση είτε βάσει κάποιου σταθερού σκορ ανά σωστή απάντηση, είτε βάσει του αλγόριθμου που έχει φτιάξει. Ο αλγόριθμος που φτιάχτηκε από τον καθηγητή είναι χρήσιμος σε σενάρια με κώδικα όπου η βαθμολόγηση έχει κάποια βαρύτητα ανάλογα το πόσο κοντά φτάνει ο φοιτητής στη λύση, και δεν είναι απόλυτη με σωστό ή λάθος.[14]

# ΚΕΦΑΛΑΙΟ 4 React Grading System

## (Υποκεφάλαιο 4.1) Περιγραφή

Το React Grading System είναι μια πλατφόρμα αξιολόγησης εργασιών JavaScript, στην οποία μπορούν να εγγράφονται καθηγητές και φοιτητές. Στόχος της είναι όχι μόνο η γρήγορη βαθμολόγηση εργασιών, αλλά και η δυνατότητα ο κάθε χρήστης να μπορεί να ανεβάζει εργασίες (αν είναι φοιτητής ή να τις δημιουργεί (αν είναι καθηγητής) σε δικό του χρόνο. Προσομοιώνει δηλαδή μια παρόμοια λειτουργία με αυτή των εργασιών στο eClass, όπου ο καθηγητής δημιουργεί νέα πρότζεκτ με κάποια προθεσμία και προαπαιτούμενα, και οι φοιτητές μπορούν να ανεβάζουν και να κατεβάζουν τις εργασίες τους όσο βρίσκονται μέσα στην προθεσμία.

Η αξιολόγηση των εργασιών γίνεται στο front end, περνώντας τον κώδικα του φοιτητή από τεστ που έχουν γραφτεί από τον καθηγητή κατά τη δημιουργία της εργασίας, δηλαδή ζευγάρια από αναμενόμενες εισόδους και αναμενόμενες εξόδους. Προσομοιώνει δηλαδή σε ένα φιλικό προς το χρήστη περιβάλλον τη διαδικασία που του debugging με testing frameworks.

Η σελίδα επικεντρώνεται στην ασφάλεια τόσο των χρηστών, όσο και του εαυτού της, καθώς παίρνει διάφορα μέτρα για να καταπολεμήσει επιθέσεις. Ελέγχει για JavaScript και SQL injections, καθώς πρέπει να είναι σίγουρο πως ο κώδικας του κάθε φοιτητή που βαθμολογεί είναι ασφαλής.

Δίνει παρ όλη την αυτοματοποίηση βέβαια τη δυνατότητα στον καθηγητή να δει και να τροποποιήσει τον κώδικα ενός φοιτητή πριν τον τρέξει, καθώς μπορεί να διαπιστωθεί η ύπαρξη κακόβουλου κώδικα, ή όπως προαναφέρθηκε στο κεφάλαιο 3.1, να μην καλύπτεται κάποιο συγκεκριμένο σενάριο από τον αλγόριθμο.

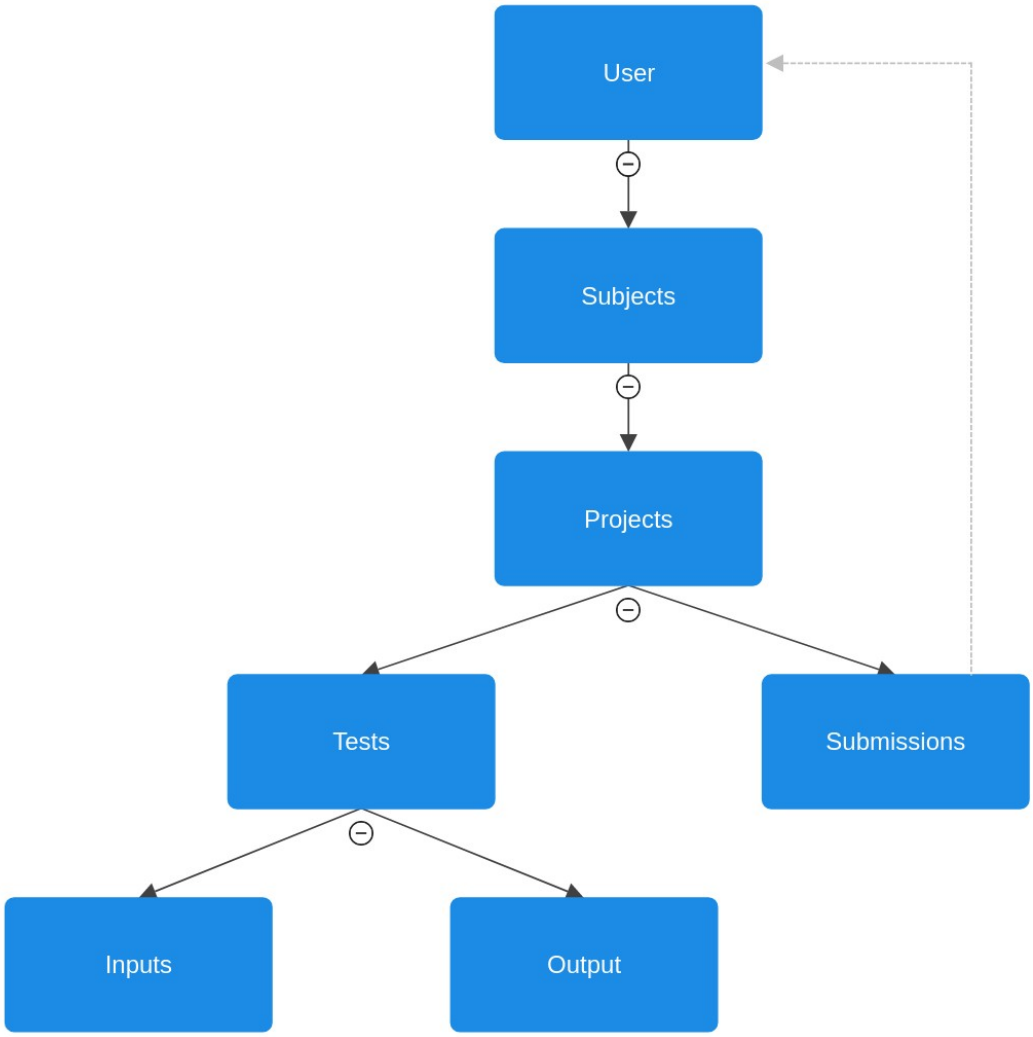
Πέρα από τα δύο frameworks τα οποία χρησιμοποιήθηκαν για την κατασκευή, έχουν βοηθήσει πολλές βιβλιοθήκες, και το στυλάρισμα έγινε με το framework SCSS της κλασικής CSS. Μερικές από τις επώνυμες βιβλιοθήκες είναι: Ο διαδικτυακός επεξεργαστής κώδικα CodeMirror, το πρότυπο δημιουργίας ασφαλών αυθεντικοποιημένων τεκμηρίων JSON Web Token, και ο διαχειριστής περιβαλλοντικών παραμέτρων dotenv. Θα γίνει πιο αναλυτική εξήγηση του καθενός παρακάτω.

Για τα δεδομένα, έχει ακολουθηθεί ένα συγκεκριμένο σχεσιακό μοντέλο που τηρείται σε όλα τα στάδια της ανάπτυξης. Το μοντέλο αυτό αποσκοπεί να ενώσει τα δεδομένα, διατηρώντας μια ιεραρχία “από πάνω προς τα κάτω”, όπου πατρικές οντότητες έχουν άμεση πρόσβαση στα δεδομένα των πρώτων παιδιών τους, και μέσω αυτών μπορούν να δουν και να τροποποιήσουν και τα υπόλοιπα.

Στην κορυφή της ιεραρχίας βρίσκεται ο χρήστης (user), ο οποίος μπορεί να έχει πολλαπλά μαθήματα (subjects), είτε ως φοιτητής είτε ως καθηγητής. Τα μαθήματα έχουν πρότζεκτ (projects), και αυτά έχουν τις εργασίες φοιτητών (submissions) και προαιρετικά τεστ (tests) για τον αλγόριθμο βαθμολόγησης. Τα τεστ με τη σειρά τους έχουν εισόδους

(inputs) και εξόδους (outputs). Όπως είναι λογικό, οι εργασίες φοιτητών πρέπει να είναι συνδεδεμένες με τους φοιτητές, αλλά για να υπάρχει μια ξεκάθαρη ροή δεδομένων σε όλα τα στάδια της ανάπτυξης, δεν αποθηκεύονται οι εργασίες ως πεδίο στον χρήστη, αλλά αποθηκεύονται αναφορές στον χρήστη μέσα στις εργασίες.

Ένας γράφος της μοντελοποίησης βρίσκεται παρακάτω.




Τέλος, η ιστοσελίδα δε φιλοξενείται σε κάποιο διακομιστή, αλλά έχει όλα τα προαπαιτούμενα ώστε να το κάνει. Για την εκτέλεση της, είναι αναγκαίο να ανοιχθούν δύο θύρες σε τοπικό διακομιστή (localhost), οι οποίες δεν είναι τυχαίες και αναλαμβάνονται από το package.json. Συγκεκριμένα, δίνεται η θύρα 8000 στο back end και 3000 στο front end.

Για την εκκίνηση στις θύρες αυτές, έχουν δημιουργηθεί και για τα δύο μέρη της εργασίας σενάρια npm για Windows και Linux, ώστε να θέτεται σωστά και μη χειροκίνητα η θύρα που διαβάζεται. Πιο συγκεκριμένα:

- Για τα Windows ο χρήστης τρέχει **`npm run start-win`**
- Για τα Linux ο χρήστης τρέχει **`npm run start-lin`**

## (Υποκεφάλαιο 4.2) Ιεραρχία

Η εφαρμογή είναι χτισμένη γύρω από μια ιεραρχία δικαιωμάτων, όπου ο κάθε χρήστης ανάλογα με το ρόλο του έχει πρόσβαση σε συγκεκριμένες σελίδες και λειτουργίες. Δε μπορούν για δηλαδή όλοι οι χρήστες να πραγματοποιήσουν όλες τις ενέργειες ή να περιηγηθούν σε όλες τις σελίδες.

Έχει χρησιμοποιηθεί ένα πεδίο **role** στη βάση δεδομένων που παίρνει τιμές μεταξύ 3 και 0 και διαχωρίζει τους χρήστες αντίστοιχα. Πιο συγκεκριμένα:

- **3 = επισκέπτης.** Ένας ρόλος που δεν αξιοποιήθηκε και έμεινε για έναν υποθετικό χρήστη ο οποίος δεν έχει τη δυνατότητα να εγγράφεται σε μαθήματα ή να ανεβάζει εργασίες. Ο σκοπός του θα ήταν να μπορεί να βλέπει μόνο τις σελίδες των μαθημάτων και των εργασιών τους για να διαβάζει τις πληροφορίες τους.
- **2 = φοιτητής.** Μπορεί να κάνει ότι κι ο επισκέπτης, και έχει επίσης τη δυνατότητα να εγγράφεται σε μαθήματα και να βλέπει τα πρότζεκτ τους, καθώς και να ανεβάζει τις εργασίες του σε αυτά. Μπορεί να βλέπει μόνο το δικό του προφίλ και τις δικές του βαθμολογίες.
- **1 = καθηγητής.** Μπορεί να κάνει ότι κι ο φοιτητής, και έχει επίσης τη δυνατότητα να επισκέπτεται τις σελίδες δημιουργίας νέου μαθήματος, νέου πρότζεκτ, και βαθμολόγησης εργασιών. Σε πρότζεκτ μαθημάτων που επιβλέπει, του δίνεται η επιλογή να αλλάξει την προθεσμία και επίσης μπορεί να βλέπει τα προφίλ όλων των άλλων χρηστών, καθώς και τις βαθμολογίες τους,
- **0 = διαχειριστής.** Μπορεί να κάνει ότι κι ο καθηγητής. Σε μια επέκταση της ιστοσελίδας, είχε σχεδιαστεί να υπάρχει πάνελ διαχειριστή πίσω από το οποίο μπορεί να αλλάζει τους ρόλους των άλλων χρηστών, έτσι φτιάχνοντας άλλους φοιτητές, καθηγητές ή και διαχειριστές.

Πέρα από τα δικαιώματα του κάθε χρήστη στο front end, έχουν μπει και περιορισμοί ώστε να απαγορεύεται η χρήση λειτουργιών σε κάποιο χρήστη με λιγότερα δικαιώματα. Για παράδειγμα, αν ένας φοιτητής δοκιμάσει από τη μπάρα συνδέσμων να δει τη σελίδα δημιουργίας νέου πρότζεκτ, αυτή θα τον ανακατευθύνει στην αρχική.

## (Υποκεφάλαιο 4.3) Front End

### (Ενότητα 4.3.1) Περιγραφή

Το front end της εφαρμογής είναι υπεύθυνο τόσο για μια φιλική εμφάνιση προς το χρήστη, όσο και για μεγάλο μέρος της λογικής και του αλγορίθμου βαθμολόγησης. Αναλαμβάνει τις κλήσεις προς το διακομιστή μέσω fetch API, και μοντελοποιεί τα δεδομένα που λαμβάνει από αυτόν σε κλάσεις (classes) ώστε να είναι έτοιμα για εμφάνιση στη διεπαφή (view).

### (Ενότητα 4.3.2) Μοντελοποίηση MVVM

Για τη κατασκευή του front end ακολουθήθηκε το αρχιτεκτονικό πρότυπο Model-View-ViewModel (MVVM ή Μοντέλο-Προβολή-Μοντέλο Προβολής) το οποίο αποσκοπεί στο διαχωρισμό της λογικής της εφαρμογής από την προβολή των δεδομένων και αποτελείται από τα τρία βασικά στοιχεία από τα οποία και ονομάζεται.[15]

Πρώτο από τα στοιχεία του MVVM, το **μοντέλο (model)** είναι μια σειρά από κλάσεις (classes) και διεπαφές (interfaces) που αντιπροσωπεύουν τα δεδομένα και τη λογική της εφαρμογής μόνο. Το μοντέλο δεν πρέπει να γνωρίζει τίποτα για το UI και συνήθως περιλαμβάνει κλάσεις που αποθηκεύουν δεδομένα και παρέχουν μεθόδους για την πρόσβαση και την επεξεργασία αυτών. Ένα παράδειγμα είναι η αντιστοίχιση δεδομένων από ένα αρχείο JSON του back end σε μία επώνυμη κλάση του front end.

Για την εφαρμογή υπάρχουν 7 κλάσεις για το μοντέλο, ακολουθώντας τη δομή και την ιεραρχία του γράφου που προαναφέρθηκε. Ακολουθεί μια επεξήγηση της κάθε μίας, μαζί με τα πεδία τους.

| class UserModel<br>implements IUser | Το μοντέλο ενός προφίλ χρήστη. Αντιπροσωπεύει τα δεδομένα του εν ενεργεία χρήστη, ή των μαθητών ενός καθηγητή |
|---|---|
| id: number | Το id του χρήστη |
| username: string | Το όνομα που χρησιμοποιείται στο login |
| firstName: string | Το πρώτο όνομα του χρήστη |
| lastName: string | Το επίθετο του χρήστη |
| role: number | Ο ρόλος του χρήστη. 0 για διαχειριστή, 1 για καθηγητή και 2 για φοιτητή |
| subjects: SubjectModel[] | Τα μαθήματα στα οποία είναι μέλος ο |

| | |
|---|---|
| | χρήστης |
| averageGrade: number \| null | Ο μέρος όρος του χρήστη |

| class SubjectModel<br>implements ISubject | Το μοντέλο ενός μαθήματος. Αντιπροσωπεύει τόσο τα δεδομένα ενός μαθήματος που ανήκει στο χρήστη, όσο και αυτά που φορτώνονται γενικά. |
|---|---|
| id: number | Το id του μαθήματος |
| name: string | Το όνομα του μαθήματος |
| description: string | Το περιγραφή του μαθήματος |
| projects: ProjectModel[] | Το πρότζεκτ του μαθήματος. |
| semester: number | Το εξάμηνο του μαθήματος |
| supervisorId: number | Το id του διαχειριστή του μαθήματος. Αντικατοπτρίζει το id του καθηγητή που το έφτιαξε |
| userGrade: number \| null | Ο βαθμός ενός χρήστη. Αυτό το πεδίο συμπληρώνεται μόνο αν το μάθημα ανήκει σε έναν χρήστη και έχει φορτωθεί από αυτόν. |
| averageGrade: number \| null | Ο μέσος όρος του βαθμού των φοιτητών του μαθήματος. Αυτό το πεδίο συμπληρώνεται μόνο αν το μάθημα ανήκει στον καθηγητή. |

| class ProjectModel<br>implements IProject | Το μοντέλο ενός πρότζεκτ. Αντιπροσωπεύει τα δεδομένα ενός πρότζεκτ από μάθημα που ανήκει στο χρήστη. |
|---|---|
| id: number | Το id του πρότζεκτ |
| name: string | Το όνομα του πρότζεκτ |
| description: string | Το περιγραφή του πρότζεκτ |
| deadline: Date \| string | Η προθεσμία του πρότζεκτ |
| submissions: SubmissionModel[] | Οι καταθέσεις εργασιών του πρότζεκτ |
| tests: TestModel[] | Τα τεστ του πρότζεκτ που περνάνε από |

| | |
|---|---|
| | τον αλγόριθμο αυτόματης αξιολόγησης |
| averageGrade: number \| null | Ο μέσος όρος του βαθμού των φοιτητών του πρότζεκτ. Αυτό το πεδίο συμπληρώνεται μόνο αν το μάθημα ανήκει στον καθηγητή. |

| class SubmissionModel<br>implements ISubmission | Το μοντέλο μιας εργασίας. Αντιπροσωπεύει τα δεδομένα μιας ανεβασμένης εργασίας φοιτητή που περνάει από τον αλγόριθμο αξιολόγησης. |
|---|---|
| id: number | Το id της εργασίας |
| name: string | Το όνομα της εργασίας |
| student?: UserModel | Μια αναφορά στον φοιτητή που ανέβασε την εργασία |
| code: string | Ο κώδικας της εργασίας |
| date: Date | Η ημερομηνία κατάθεσης της εργασίας |
| grade: number \| null | Ο βαθμός της εργασίας που δίνεται από τον καθηγητή μετά την αξιολόγηση |

| class TestModel<br>implements ITest | Το μοντέλο ενός τεστ για κάποιο πρότζεκτ. Αντιπροσωπεύει τα δεδομένα του τεστ και κρατάει αναφορές στις διάφορες εισόδους και εξόδους. |
|---|---|
| id: number | Το id του τεστ |
| mainFunction: string | Η κύρια συνάρτηση την οποία θα τρέξει το τεστ κατά την εκτέλεση του. |
| inputs: ITestInput[] | Οι είσοδοι που το τεστ θα δώσει στην κύρια συνάρτηση κατά την εκτέλεση του |
| output: ITestOutput | Η έξοδος που περιμένει το τεστ μετά την εκτέλεση του |

| class TestInputModel<br>implements ITestInput | Το μοντέλο μίας εισόδου για κάποιο τεστ. |
|---|---|

| | |
|---|---|
| id: number | Το id της εισόδου |
| code: string | Ο κώδικας της εισόδου |
| isMainParam: boolean; | Μια μεταβλητή που κρίνει αν η είσοδος θα χρησιμοποιηθεί σαν παράμετρος στην κύρια συνάρτηση του τεστ. Το πεδίο αυτό χρησιμοποιείται μόνο για σκοπούς debugging και δεν έχει κάποια άλλη χρησιμότητα |

| | |
|---|---|
| class TestOutputModel<br>implements ITestOutput | Το μοντέλο μίας εξόδου για κάποιο τεστ. |
| id: number | Το id της εισόδου |
| code: string | Ο κώδικας της εξόδου |

Διάφορες από τις προαναφερόμενες κλάσεις χρησιμοποιούν επίσης και βοηθητικές μεταβλητές, οι οποίες δεν είναι μέρη της μοντελοποίησης, όπως για παράδειγμα η μεταβλητή isValidJSONorArray?: boolean που χρησιμοποιείται προαιρετικά στα TestInputModel και TestOutputModel για να ελέγξει αν ο κώδικας που δόθηκε είναι αντικείμενο JSON ή πίνακας. Αυτές οι μεταβλητές χρησιμοποιούνται μόνο σε ένα ή μερικά μέρη της μοντελοποίησης και δεν αφορούν αυτή, οπότε δεν αναφέρθηκαν παραπάνω.

Δεύτερο στοιχείο του MVVM, η **προβολή (view)** αντιπροσωπεύει το UI της εφαρμογής. Περιλαμβάνει τα γραφικά στοιχεία όπως κουμπιά, πεδία κειμένου και λοιπά στοιχεία που ο χρήστης βλέπει και αλληλεπιδρά μαζί τους. Η προβολή από μόνη της παρόλα αυτά δεν περιέχει τα δεδομένα που έχουν οριστεί στο μοντέλο, καθώς η δουλειά της είναι απλά η εμφάνιση της ιστοσελίδας.

Στα πλαίσια της React, η προβολή μπορεί να αναπαρασταθεί με συναρτησιακά components (functional components) ή κλάσεις που περιέχουν JSX/TSX (JavaScript / TypeScript XML), επεκτάσεις των αντίστοιχων γλωσσών που επιτρέπουν στο χρήστη την ενσωμάτωση σύνταξης HTML μέσα σε κώδικα JavaScript. Τα JSX/TSX χρησιμοποιούνται συχνά από προγραμματιστές ώστε ο κώδικας να είναι ευανάγνωστος και να ακολουθεί το πρότυπο “παντού JavaScript”.

Η προβολή του React Grading System αποτελείται από 9 σελίδες οι οποίες ακολουθούν την ιεραρχία 3-0. Σε περίπτωση που γίνει προσπάθεια επίσκεψης σε σελίδα που κάποιος χρήστης δεν έχει δικαίωμα να δει, ανακατευθύνεται στη σελίδα σύνδεσης (/) ή στην αρχική (/home), ανάλογα αν δεν είναι ή είναι συνδεδεμένος αντίστοιχα. Περαιτέρω, κάποιες από

τις σελίδες δέχονται παραμέτρους ώστε να φορτώσουν στην προβολή διαφορετικά δεδομένα. Μία από τις κοινές παραμέτρους είναι τα φίλτρα δεδομένων, τα οποία μπορούν να πάρουν τιμές -1 για αρνητικό, 0 για ουδέτερο και 1 για θετικό. Ακολουθεί μια λίστα με τις σελίδες και τις σχετικές πληροφορίες τους.

- **Login Page**
  - Relative Link: /
  - Περιγραφή: Σύνδεση χρήστη και δημιουργία νέου λογαριασμού
  - Προσβασιμότητα: Σε όλους όσους δεν είναι συνδεδεμένοι.
- **Home Page**
  - Relative Link: /home
  - Περιγραφή: Η αρχική οθόνη που περιέχει τα μαθήματα του χρήστη και τα πρότζεκτ που δεν έχει παραδώσει.
  - Προσβασιμότητα: Σε όλους όσους είναι συνδεδεμένοι.
- **Profile Page**
  - Relative Link: /profile
  - Περιγραφή: Το προφίλ του χρήστη που περιέχει διάφορα στατιστικά στοιχεία του και όλα τα μαθήματα με τις βαθμολογίες που πήραν.
  - Προσβασιμότητα: Σε όλους όσους είναι συνδεδεμένοι.
  - Παράμετροι
    - id: Αν δοθεί φορτώνει το προφίλ του χρήστη με αυτό το id. Έχει νόημα μόνο για καθηγητές και διαχειριστές, καθώς οι φοιτητές δε μπορούν να δουν άλλα προφίλ πέρα από το δικό τους
- **Subjects Page**
  - Relative Link: /subjects
  - Περιγραφή: Η σελίδα των μαθημάτων, που περιέχει μια λίστα με όλα τα μαθήματα στο μισό της οθόνης και ένα επιλεγμένο μάθημα στο άλλο μισό.
  - Προσβασιμότητα: Σε όλους όσους είναι συνδεδεμένοι.
  - Παράμετροι
    - id: Το id του μαθήματος. Φορτώνει σε μέρος της σελίδας τις πληροφορίες του αντίστοιχου πρότζεκτ
    - joined: Φίλτρο που φορτώνει μαθήματα βάσει του αν συμμετέχει ο χρήστης
    - supervising: Έχει νόημα μόνο για καθηγητές και διαχειριστές. Φίλτρο που φορτώνει μαθήματα βάσει του αν τα διαχειρίζεται ο χρήστης.
- **Projects Page**
  - Relative Link: /projects
  - Περιγραφή: Η σελίδα των πρότζεκτ, που περιέχει μια λίστα με όλα τα πρότζεκτ των μαθημάτων που είναι εγγεγραμμένος ο χρήστης στο μισό της οθόνης και ένα επιλεγμένο πρότζεκτ στο άλλο μισό.
  - Προσβασιμότητα: Σε όλους όσους είναι συνδεδεμένοι.
  - Παράμετροι

        - id: Το id του πρότζεκτ. Φορτώνει σε μέρος της σελίδας τις πληροφορίες του αντίστοιχου μαθήματος
        - submitted: Φίλτρο που φορτώνει πρότζεκτ βάσει του αν τα έχει παραδώσει ο χρήστης
        - graded: Φίλτρο που φορτώνει πρότζεκτ βάσει του αν έχουν βαθμολογηθεί
        - deadline: Φίλτρο που φορτώνει τα πρότζεκτ βάσει του αν είναι εμπρόθεσμα
        - supervising: Έχει νόημα μόνο για καθηγητές και διαχειριστές. Φίλτρο που φορτώνει πρότζεκτ βάσει του αν τα διαχειρίζεται ο χρήστης.
- **Submissions Page**
    - Relative Link: /submissions
    - Περιγραφή: Η σελίδα των ανεβασμένων εργασιών. Εδώ φορτώνονται για ένα συγκεκριμένο πρότζεκτ όλες οι εργασίες των φοιτητών. Σε μέρος είναι φορτωμένη η εκφώνηση του πρότζεκτ και οι εργασίες, και σε άλλο μέρος ο κώδικας για βαθμολόγηση.
    - Προσβασιμότητα: Σε καθηγητές και διαχειριστές
    - Παράμετροι
        - project: Το id του πρότζεκτ. Αν δε δοθεί, η σελίδα είναι κενή.
        - Id: Το id της εργασίας. Φορτώνει σε μέρος της σελίδας τον κώδικα του φοιτητή για βαθμολόγηση.
- **New Subject Page**
    - Relative Link: /new-subject
    - Περιγραφή: Σελίδα για τη δημιουργία νέων μαθημάτων
    - Προσβασιμότητα: Σε καθηγητές και διαχειριστές
- **New Project Page**
    - Relative Link: /new-project
    - Περιγραφή: Σελίδα για τη δημιουργία νέων πρότζεκτ και των αντίστοιχων τεστ τους.
    - Προσβασιμότητα: Σε καθηγητές και διαχειριστές
- **Error Page**
    - Relative Link: /* (* = wildcard)
    - Περιγραφή: Αν δεν έχει αναφερθεί πιο πάνω, η σελίδα παραπέμπει εδώ, όπου ειδοποιείται ο χρήστης πως έκανε λάθος.
    - Προσβασιμότητα: Σε όλους

Αξιοσημείωτο είναι πως η πλοήγηση από τη μία σελίδα στην άλλη γίνεται μέσω του React Router, το οποίο είναι μια βιβλιοθήκη που αναλαμβάνει εφαρμογές React και φροντίζει να υπάρχει γρήγορη εναλλαγή σελίδων χωρίς επαναφορτώσεις. Οι περισσότερες πλοηγήσεις, με εξαίρεση το login/logout δε χρειάζονται επαναφόρτωση, και σε αυτές τις περιπτώσεις η επαναφόρτωση είναι υποχρεωτική λόγω μιας αλλαγής στα cookies, η οποία θα αναλυθεί παρακάτω.

Τρίτο στοιχείο του MVVM, το **πρότυπο προβολής (ViewModel)**, είναι η σύνδεση του μοντέλου και της προβολής. Είναι ένα ενδιάμεσο στρώμα από συναρτήσεις που επιτρέπουν την προβολή να επικοινωνεί με το μοντέλο χωρίς να γνωρίζει τη λεπτομερή υλοποίηση του. Συνήθως επίσης περιέχει τη λογική που αφορά την εμφάνιση και τη συμπεριφορά των στοιχείων της προβολής.

Η σύνδεση μεταξύ του μοντέλου, του προτύπου προβολής και της προβολής συνήθως γίνεται μέσω δεσμεύσεων δεδομένων (data bindings) που επιτρέπουν την αυτόματη ενημέρωση των γραφικών στοιχείων της προβολής όταν τα δεδομένα αλλάζουν στο μοντέλο.

Αν και το πρότυπο προβολής μοιάζει με τον ελεγκτή (controller) της αρχιτεκτονικής MVC (Model-View-Controller), έχοντας μια ενδιάμεση λειτουργία και συνδέοντας το μοντέλο και την προβολή, τα δύο μεταξύ τους έχουν βασικές διαφορές. Μία από αυτές είναι πως ο ελεγκτής επεξεργάζεται τα δεδομένα που λαμβάνονται από τον χρήστη (μέσω του UI) και να προωθεί τα αντίστοιχα αιτήματα προς το μοντέλο ή την προβολή ενώ το πρότυπο προβολής είναι υπεύθυνο για τη διαχείριση των δεδομένων που εμφανίζονται στην προβολή. Σε αρκετές περιπτώσεις βέβαια, και ειδικά σε μεγάλες εφαρμογές, χρησιμοποιείται και πρότυπο προβολής αλλά και ελεγκτής παράλληλα.

Αν και επίσης συνηθίζεται σε μεγάλες εφαρμογές να υπάρχει διαχωρισμός του μοντέλου και του προτύπου προβολής, στο React Grading System το πρότυπο προβολής ενσωματώθηκε μέσα στο μοντέλο για να βοηθήσει στην αναγνωσιμότητα λόγω της μικρής έκτασης της εφαρμογής και την έλλειψη πολυπλοκότητας των συναρτήσεων του προτύπου.

Γενικά στην ιστοσελίδα το πρότυπο προβολής ενός μοντέλου είτε δεν είχε καμία συνάρτηση, είτε είχε μία για την ανάκτηση δεδομένων από το back end (fetching from the back end) και την αντιστοίχηση τους στην κλάση, είτε είχε ελάχιστες για το φιλτράρισμα των δεδομένων στο view.

## (Ενότητα 4.3.2) Ροή Δεδομένων

Η ορολογία dataflow (ή dataflow programming στον προγραμματισμό) αναφέρεται στον τρόπο με τον οποίο τα δεδομένα ρέουν μέσα σε ένα πρόγραμμα ή σύστημα. Στο dataflow programming, οι διεργασίες εκτελούνται όταν τα δεδομένα είναι διαθέσιμα για επεξεργασία, αντί να εκτελούνται με βάση μια σειρά εντολών. Αυτό σημαίνει ότι οι εκφράσεις εκτελούνται μόνο όταν υπάρχουν δεδομένα για τις οποίες μπορούν να εφαρμοστούν, και μπορεί να γίνει παράλληλη εκτέλεση διαφορετικών τμημάτων του προγράμματος που επεξεργάζονται διαφορετικά δεδομένα.

Όσον αφορά το front end του React Grading System, τα δεδομένα περνάνε από διάφορα στάδια μέχρι να εμφανιστούν στην οθόνη του χρήστη, ακολουθώντας τους κανόνες του MVVM.

Η ανάκτηση δεδομένων από το back end (fetching from the back end) γίνεται με χρήση του Fetch API, μιας ασύγχρονης συνάρτησης για επικοινωνία με τον διακομιστή. Η

συνάρτηση αυτή κάνει κλήση (Request) σε συγκεκριμένο πόρο του back end, προαιρετικά στέλνοντας (post) δεδομένα, και περιμένει απάντηση (Response), η οποία περιέχει ένα κωδικό κατάστασης HTTP (HTTP status code) και προαιρετικά κάποια δεδομένα, τα οποία στα πλαίσια της εφαρμογής είναι αρχεία JSON.[17] Οι κωδικοί κατάστασης HTTP που λαμβάνονται υπόψη είναι:

- 200: Ok
- 401: Unauthorized
- 403: Forbidden
- 404: Not Found

Καθώς οι συναρτήσεις του Fetch API είναι ασύγχρονες, δεν εκτελούνται ακολουθιακά και υπάρχει ένα αμελητέο χρονικό περιθώριο αναμονής μέχρι να δοθεί Response, στο οποίο η εφαρμογή περιμένει.

Οι συναρτήσεις που ζητούν (GET) και δίνουν (POST) δεδομένα στο back end είναι υλοποιημένες ξεχωριστά και η κάθε μία αναλαμβάνει μία κλήση και μία οντότητα. Για παράδειγμα, μια κλήση για να ανακτηθούν τα δεδομένα του χρήστη επιστρέφει όσα δεδομένα είναι πεδία στη βάση δεδομένων και δεν επιστρέφει άλλες οντότητες όπως τα μαθήματα ή τα πρότζεκτ του χρήστη.

Περαιτέρω, η διαδικασία των κλήσεων ακολουθεί το πρωτόκολλο της Αυθεντικοποίησης με Token (Token-Based Authentication). Σύμφωνα με το προτόκολο, με το που κάποιος χρήστης συνδεθεί, δημιουργείται ένα JSON Web Token (JWT) από το back end που ισχύει για ένα συγκεκριμένο αριθμό ωρών και αποθηκεύεται τοπικά ως cookie το οποίο διαγράφεται στην αποσύνδεση. Το JWT είναι ένα πρότυπο για τη δημιουργία ασφαλών αυθεντικοποιημένων τεκμηρίων που επιτρέπει στο χρήστη να κάνει κλήσεις στους υπόλοιπους πόρους της εφαρμογής. Χωρίς αυτό, δε μπορεί κάποιος να ζητήσει να κάνει οποιαδήποτε άλλη κλήση πέρα από απόπειρα σύνδεσης και δημιουργία λογαριασμού. Η ύπαρξη του προστατεύει την εφαρμογή από κακόβουλες επιθέσεις που προσπαθούν να υποκλέψουν δεδομένα. Μετά τη λήξη του JWT, ένας συνδεδεμένος χρήστης θα μεταφερθεί ξανά στη σελίδα σύνδεσης όπου και θα του ζητηθεί να ξανασυνδεθεί.[16]

Στην περίπτωση που ένας αυθεντικοποιημένος χρήστης ζητήσει δεδομένα από το back end για μια οντότητα, η εφαρμογή αντιστοιχεί την απάντηση με ένα νέο instance κλάσης του μοντέλου για αυτή την οντότητα. Κάθε κλάση έχει περαιτέρω μια συνάρτηση ονόματι setup η οποία δίνει την δυνατότητα να συνεχίσουν να γίνονται ανακτήσεις δεδομένων που ανήκουν σε δικές τους οντότητες, ακολουθώντας την ιεραρχία που δόθηκε στον γράφο του κεφαλαίου 4.2. Το βάθος των ανακτήσεων ορίζεται από την εφαρμογή ανά περίπτωση, έτσι κάνοντας και τον ελάχιστο δυνατό αριθμό κλήσεων και παίρνοντας μόνο τα απαραίτητα δεδομένα.

Όταν τελειώσει η αντιστοίχιση, το νέο instance καταχωρείται σε κάποιο state στην προβολή από την οποία έγινε η κλήση, και τα δεδομένα εμφανίζονται στην οθόνη του χρήστη.

Τέλος, η αποστολή δεδομένων προς το back end προαπαιτεί το να έχει το πακέτο την κατάλληλη μορφή, δηλαδή να στείλει ένα σωστό JSON. Οι περιπτώσεις αποστολών είναι πολύ πιο λίγες, με την πιο σημαντική ίσως να είναι η αποστολή κώδικα στο back end. Αυτό γίνεται κάνοντας τον κώδικα string στο ανέβασμα του αρχείου χωρίς να το τρέξει. Δε γίνεται σε αυτό το σημείο κάποιος έλεγχος για το περιεχόμενο του κώδικα, και αφήνεται για το επόμενο στάδιο, δηλαδή κατά την ανάκτηση του και την αξιολόγηση εργασιών.

## (Ενότητα 4.3.3) Αξιολόγηση Εργασιών

Η ιδεολογία του αλγόριθμου αξιολόγησης είναι πως τρέχει κώδικα JavaScript από μία συγκεκριμένη κύρια συνάρτηση με συγκεκριμένες εισόδους, και περιμένοντας συγκεκριμένες εξόδους. Το σύνολο των δεδομένων που αποτελούν την κύρια συνάρτηση, τις εισόδους και την αναμενόμενη έξοδο δημιουργούν ένα τεστ. Κατά συνέπεια, αυτό σημαίνει πως για την αξιοποίηση του αλγορίθμου αξιολόγησης, ένα πρότζεκτ μπορεί να περιέχει οτιδήποτε όσο αυτό είναι μια σειρά από συναρτήσεις JavaScript που επιστρέφουν ένα αποτέλεσμα.

Διάφορα τεστ μπορούν να οριστούν με αυτό τον τρόπο από τον καθηγητή κατά τη δημιουργία ενός πρότζεκτ. Είναι σημαντικό για αυτό το λόγο, να γίνεται ξεκάθαρο από την εκφώνηση ποια είναι η απαραίτητη ονομασία των συναρτήσεων που πρέπει να γράψει ο φοιτητής.

Σε ένα απλό παράδειγμα αυτού, ας πούμε πως έχει ανέβει πρότζεκτ η οποία ζητάει τη δημιουργία μιας συνάρτησης που παίρνει δύο αριθμούς και βρίσκει το μέσο όρο τους. Όταν ο αλγόριθμος αξιολόγησης περάσει την εργασία, θα πρέπει να ξέρει σε ποια συνάρτηση να δοκιμάσει τις εισόδους του και να ελέγξει αν παίρνει το επιθυμητό αποτέλεσμα. Έστω λοιπόν πως ζητείται σαν όνομα κύριας συνάρτησης το **calculateAverage**, και παραδίδονται οι παρακάτω 3 εργασίες:

Εργασία 1

```
function calculateAverage(num1, num2) {
    const sum = num1 + num2;
    const average = sum / 2;
    return average;
}
```

Στον έλεγχο, ο αλγόριθμος θα αναγνωρίσει τη συνάρτηση **calculateAverage** και θα περάσει τις δύο εισόδους του στις παραμέτρους **num1** και **num2**.

Εργασία 2

```
function calculateAverage(num1) {
    const sum = num1 + 10;
    const average = sum / 2;
    return average;
}
```

Ο αλγόριθμος θα αναγνωρίσει τη συνάρτηση **calculateAverage** αλλά δε θα μπορέσει να περάσει τις δύο εισόδους του, καθώς βλέπει μόνο μία. Έτσι, αν ο καθηγητής είχε ορίσει σαν εισόδους για παράδειγμα το 3 και 5, θα κληθεί η **calculateAverage(3, 5)**, και θα βγάλει συντακτικό λάθος.

Εργασία 3

```
function main(num1, num2) {
    const sum = num1 + num2;
    const average = sum / 2;
    return average;
}
```

Ο αλγόριθμος δε θα βρει τη συνάρτηση **calculateAverage** οπότε δε θα ξέρει που να τεστάρει τις παραμέτρους του. Αν και η εργασία είναι σωστή, θα αναγνωριστεί σαν λάθος.

Αν και η λειτουργία της κύριας συνάρτησης φαίνεται περιοριστική εξ' αρχής, το πρόβλημα υπάρχει μόνο σε μικρά πρότζεκτ σαν αυτό του παραδείγματος. Σε πρότζεκτ με δεκάδες γραμμές κώδικα είναι σημαντικό να όχι μόνο να διαχωρίζεται ποια είναι η κύρια συνάρτηση και ποιες οι βοηθητικές, αλλά και να δίνεται η δυνατότητα στον καθηγητή να φτιάξει τεστ για πολλές από αυτές τις συναρτήσεις ώστε να ελέγξει περισσότερα τμήματα του κώδικα.

Σε ένα παράδειγμα όπου θέλουμε να φτιαχτούν δύο δισδιάστατοι πίνακες K x N και N x M και μετά να πολλαπλασιαστούν, μπορεί να χρειαστούμε διάφορες συναρτήσεις όπως **randomiseArrays, multiplyArrays** και **printArrays**. Η κάθε μία από αυτές αναλαμβάνει ένα κομμάτι του κώδικα το οποίο αν υπήρχε μόνο σε μία κύρια συνάρτηση θα ήταν αδύνατο να ελεγχθεί. Δίνουμε έτσι μεγαλύτερο έλεγχο στον αλγόριθμο ως προς το ποια κομμάτια να αξιολογήσει.

Η μέθοδος αξιολόγησης περιλαμβάνει το **Function()** constructor της JavaSript, που χρησιμοποιείται για τη δημιουργία νέων συναρτήσεων δυναμικά κατά τη διάρκεια εκτέλεσης του κώδικα. Δηλαδή, για κάθε τεστ, δημιουργείται μια συνάρτηση με Function() που επιστρέφει την κύρια συνάρτηση του τεστ με τις εισόδους του. Το αντίστοιχο απόσπασμα από τον κώδικα απεικονίζεται παρακάτω.

```
let finalCode = `${code}
return ${test.mainFunction}(${inputCode});
`


try{
    let result = Function(finalCode)()
```

Όπως φαίνεται από την εικόνα, το Function() χρησιμοποιείται σε συνδυασμό με Immediately-Invoked Function Expression (IIFE), το οποίο μπορεί να εκφραστεί και ως **Function()()** και είναι ένα πρότυπο για την άμεση εκτέλεση μιας συνάρτησης κατά τη δημιουργία της.[25]

Είναι σημαντικό να αναφερθεί πως παρά την ευελιξία του, ο Function() constructor πρέπει να χρησιμοποιείται με προσοχή λόγω του δυνητικού κινδύνου ασφάλειας, όπως JavaScript injections, τα οποία θα αναλυθούν παρακάτω.

Το Function() constructor είναι ο διάδοχος της πασίγνωστης συνάρτησης **eval()** που ενώ είχε παρόμοια λειτουργία, είχε πολύ μεγαλύτερο ρίσκα, τόσο που οδήγησε στη χρήση της φράσης *"Eval is evil!"*[18] Οι βασικές διαφορές ασφάλειας είναι κυρίως πως ενώ η eval() εκτελεί τον κώδικα που της δίνεται ως συμβολοσειρά, η Function() δημιουργεί μια νέα συνάρτηση κατά το runtime βάσει της παρεχόμενης συμβολοσειράς κώδικα. Με άλλα λόγια, στη Function() δε μπορεί κάποιος να δώσει κώδικα ο οποίος δεν είναι JavaScript και τρέχει σε διαφορετικό scope.

Αφού λοιπόν η Function() τρέξει τον κώδικα, υπάρχουν 4 σενάρια που μπορεί να ακολουθηθούν:

- Ο κώδικας τρέχει σωστά και επιστρέφει το αναμενόμενο αποτέλεσμα. Το τεστ είναι πετυχημένο.
- Ο κώδικας τρέχει σωστά και δεν επιστρέφει το αναμενόμενο αποτέλεσμα. Το τεστ είναι αποτυχημένο.
- Ο κώδικας τρέχει άλλα δεν επιστρέφει κάποιο αποτέλεσμα. Αυτό συμβαίνει όταν ο φοιτητής δεν ονόμασε την κύρια συνάρτηση του κατάλληλα ή δεν έχει ορίσει κάποιο αποτέλεσμα να επιστρέφεται. Το τεστ είναι αποτυχημένο.
- Ο κώδικας δεν τρέχει. Αυτό συμβαίνει όταν υπάρχει κάποιο συντακτικό λάθος, ή δεν είναι κώδικας JavaScript, ή σε άλλες απρόσμενες περιπτώσεις.

Η βαθμολογία της εργασίας υπολογίζεται από το μέσο όρο των πετυχημένων τεστ και εμφανίζεται στην οθόνη του καθηγητή με το που τρέξει τον αλγόριθμο, μαζί με μια συνοπτική εξήγηση του κάθε τεστ (παράμετροι, αναμενόμενο αποτέλεσμα, πραγματικό αποτέλεσμα).

Πριν περάσει από τον αλγόριθμο, ο κώδικας του φοιτητή ανεβαίνει σε ένα playground που υλοποιήθηκε με το διαδικτυακό επεξεργαστή κώδικα CodeMirror, όπου μπορεί να τροποποιηθεί από τον καθηγητή. Ο σκοπός ύπαρξης μίας τέτοιας λειτουργίας είναι η αποφυγή λάθος βαθμολόγησης σε περίπτωση που έχει γίνει κάποιο λάθος από τη μεριά του καθηγητή ή να εξεταστούν περιπτώσεις που ο καθηγητής δε θέλει ο βαθμός του φοιτητή να είναι απόλυτος από τον αλγόριθμο αξιολόγησης.

Κοινές περιπτώσεις λαθών από μεριάς καθηγητή είναι τα τυπογραφικά λάθη, όπως για παράδειγμα να ορίσει σαν κύρια συνάρτηση **calculateAray** αντί για **calculateArray**, και τα λογικά λάθη, όπως για παράδειγμα εσφαλμένος υπολογισμός αναμενόμενης εξόδου.

Καθώς τέτοια λάθη δεν είναι δυνατό να αποφευχθούν, δίνεται η δυνατότητα στον καθηγητή να τρέξει τον αλγόριθμο είτε ανεβάζοντας τη βαθμολογία της εργασίας απευθείας, είτε χωρίς, ανάλογα με το αν θέλει να κάνει και χειροκίνητο έλεγχο.

## (Υποκεφάλαιο 4.4) Back End

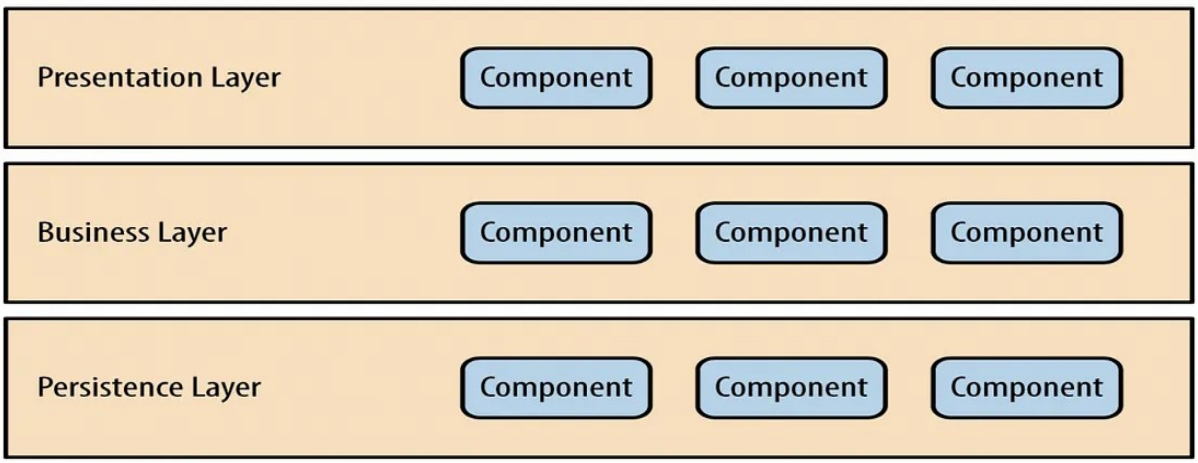
### (Ενότητα 4.4.1) Περιγραφή

Το back end της εφαρμογής είναι τόσο υπεύθυνο για μια γρήγορη επεξεργασία των δεδομένων, όσο και για την επικοινωνία με τη βάση δεδομένων. Συνδέεται στη βάση με τη βιβλιοθήκη της **mysql** και επικοινωνεί μαζί της χρησιμοποιώντας transactions για ακεραιότητα δεδομένων. Παίρνει επίσης διάφορα μέτρα ασφαλείας ως προς την κωδικοποίηση ευαίσθητων δεδομένων, όπως κωδικοί.

### (Ενότητα 4.4.2) Layered Architecture

Για την κατασκευή του back end χρησιμοποιήθηκε η Αρχιτεκτονική Διαστρωμάτωσης (Layered Architecture), ένας τρόπος οργάνωσης του κώδικα σε επίπεδα, κάθε ένα από τα οποία αναλαμβάνει συγκεκριμένες λειτουργίες. Κάθε επίπεδο εξαρτάται μόνο από αυτά που βρίσκονται από πάνω του.

Αξιοσημείωτο είναι πως αυτό το μοντέλο δεν είναι αποκλειστικό στην κατασκευή ενός back end και συνδυάζεται συνήθως με άλλα, παρόλα αυτά εδώ χρησιμοποιήθηκε μόνο για αυτό. Πιο συγκεκριμένα, σε μια τέτοια αρχιτεκτονική συνηθίζεται να υπάρχουν 3 διαφορετικά επίπεδα: Presentation Layer, Business Layer και Persistance Layer.[19]




**Presentation Layer ή View Layer**, είναι το επίπεδο που είναι υπεύθυνο για την παρουσίαση των δεδομένων στο χρήστη. Όταν αναφερόμαστε σε full stack development, αυτό μπορεί να σημαίνει οτιδήποτε βλέπει ο χρήστης όπως το γραφικό περιβάλλον μιας εφαρμογής ή ιστοσελίδας.

Στην περίπτωση του back end του React Grading System, το Presentation Layer αφορά το REST API, το οποίο είναι το κοντινότερο σε διεπαφή με το χρήστη υπάρχει.

Το REST API (Representational State Transfer Application Programming Interface) είναι ένας τρόπος σχεδιασμού και ανάπτυξης εφαρμογών διαδικτύου. Είναι το κομμάτι του κώδικα που επιτρέπει την επικοινωνία μεταξύ του front end και του back end μέσω του πρωτοκόλλου HTTP.

Αποτελεί το σύνολο των endpoints, των διεπαφών δηλαδή που αποτελούν τις διαδρομές URL τις οποίες καλεί το front end. Ο κάθε πόρος μπορεί να χρησιμοποιείται για διαφορετικές ενέργειες, από τις οποίες η εφαρμογή χρησιμοποιεί:

- **GET:** Χρησιμοποιείται για την ανάκληση δεδομένων από τη βάση και την παράδοση τους στο front end.
- **POST:** Χρησιμοποιείται για την δημιουργία νέων πόρων ή την αποστολή δεδομένων από το front end στη βάση.
- **PUT:** Χρησιμοποιείται για την ενημέρωση ενός πόρου, όπου τα ενημερωμένα δεδομένα δίνονται από το front end.
- **DELETE:** Χρησιμοποιείται για τη διαγραφή εγγραφών από τη βάση.

**Business Layer**, το επίπεδο που περιλαμβάνει την κύρια επιχειρηματική λογική της εφαρμογής. Είναι το ενδιάμεσο επίπεδο που καλείται από το REST API και περιέχει τις λειτουργίες που αφορούν την επεξεργασία και τη διαχείριση των δεδομένων. Συνήθως περιέχει ένα σύνολο κλάσεων που ονομάζονται Managers οι οποίοι περιέχουν όλες τις συναρτήσεις που ελέγχουν για τυχόν λάθη και δίνουν το πράσινο φως στο να προχωρήσει ο κώδικας.

Στο React Grading System, οι περισσότερες συναρτήσεις των Managers ασχολούνται με τον έλεγχο του JWT και τον εντοπισμό λαθών στις παραμέτρους που ήρθαν από τα HTTP requests. Αν και το JWT και Αυτές αφορούν τις οντότητες του χρήστη, των μαθημάτων, των πρότζεκτ, των εργασιών και των τεστ. Υπάρχουν βέβαια και μερικές ειδικές περιπτώσεις οι οποίες θα αναλυθούν.

Το Token Manager αναλαμβάνει τη δημιουργία νέων JWT tokens με διάρκεια ζωής 168 ώρες, δηλαδή μίας εβδομάδας, και την επαλήθευση των token που ήδη υπάρχουν για το αν έχουν λήξει. Ένα token επίσης μπορεί να περιέχει πέρα από οτιδήποτε έχει προαναφερθεί, και μερικά δεδομένα που συνήθως χρησιμοποιούνται μαζί του. Στην προκειμένη της εργασίας έχουν αποθηκευτεί το id του χρήστη και ο ρόλος του.

Οι συναρτήσεις register και login, όταν εκτελεσθούν επιτυχημένα, δημιουργούν νέα tokens, επιστρέφοντας το πίσω στο front end, και πιο συγκεκριμένα το register περνάει τον κωδικό που έδωσε ο χρήστης από κρυπτογράφηση (hashing). Η κρυπτογράφηση αυτή γίνεται με τη βοήθεια του αλγόριθμου bcrypt ο οποίος έχει σχεδιαστεί ώστε να είναι αρκετά αργός και ανθεκτικός στις επιθέσεις brute-force. Λειτουργεί περνώντας τον κωδικό του χρήστη από μια τυχαία τιμή που ονομάζουμε salt, για έναν αριθμό επαναλήψεων που ονομάζουμε cost factor.[20]

Τέλος υπάρχει το ειδικό Transaction Manager που δεν διαχειρίζεται κάποια οντότητα, αλλά ασχολείται με τις “συναλλαγές” που γίνονται στη βάση δεδομένων, πιο συγκεκριμένα εξασφαλίζοντας τη συνοχή και την ακεραιότητα των δεδομένων. Μια συναλλαγή είναι μια ακολουθία ενεργειών (queries, commits, rollbacks) που εκτελούνται ατομικά και είτε εφαρμόζονται όλες μαζί, είτε καμία.[23]

Όταν δηλαδή το back end είναι έτοιμο να κάνει κλήσεις (queries) στη βάση δεδομένων, διαβάζοντας ή γράφοντας δεδομένα σε αυτή, ξεκινάει μία συναλλαγή και μετά εκτελεί την κλήση του. Αν τα πάντα ολοκληρωθούν πετυχημένα, θα επικυρώσει τις αλλαγές (commit), κι αν αποτύχει σε κάποιο βήμα θα γυρίσει στο αρχικό στάδιο χωρίς να κάνει καμία αλλαγή (rollback).

Περαιτέρω μέσω συναλλαγών διασφαλίζεται πως δε γίνονται παράλληλες αλλαγές στη βάση δεδομένων, όπως στην περίπτωση ύπαρξης πολλών χρηστών που ανεβάζουν δεδομένα ταυτόχρονα, αποφεύγοντας έτσι μπερδέματα.

Ένα ακόμα χαρακτηριστικό του Transaction Manager είναι πως χρησιμοποιεί Dependency Injections (DI) ώστε να είναι διαθέσιμος τόσο στο Business Layer, όσο και στο Persistence Layer που θα αναλυθεί παρακάτω, στο οποίο κιόλας είναι πιο χρήσιμος, καθώς είναι πιο κοντά στη βάση παρά στο ενδιάμεσο στρώμα.

Το Dependency Injection που αναφέρθηκε, είναι μια προγραμματιστική τεχνική που αποσκοπεί στο να μειωθούν οι βαθμοί σύνδεσης μεταξύ των διάφορων στοιχείων του κώδικα. Η λογική της είναι πως αντικείμενα ή συναρτήσεις που είναι απαραίτητα κάπου, έρχονται από κάποιο εξωτερικό παράγοντα (όπως ως παράμετροι) αντί να δημιουργούνται εσωτερικά.[21] Αυτό διαχωρίζει το ποια κομμάτια του κώδικα είναι ικανά στο να κατασκευάσουν άλλα, και κάνει τον κώδικα πιο συντηρήσιμο.

**Persistence Layer**, το επίπεδο που είναι υπεύθυνο τόσο για τη δημιουργία και διαχείριση των οντοτήτων (entities) όσο και για τα Data Access Objects (DAO) και τις κλήσεις στη βάση δεδομένων. Πιο συγκεκριμένα, οι οντότητες αντιπροσωπεύουν τα δεδομένα που αποθηκεύονται στη βάση δεδομένων και αποτελούν τις αντιστοιχίσεις των πινάκων, κάτι παρόμοιο με το μοντέλο στο MVVM του front end, και τα DAO είναι κλάσεις που χρησιμοποιούνται για την ανάκτηση και την αποθήκευση δεδομένων από και προς τη βάση δεδομένων, με άλλα λόγια το σύνολο των εντολών SQL και οι συναρτήσεις που στέλνουν queries.

Σε μεγάλες και πολύπλοκες εφαρμογές που το back end δεν κάνει ανάκτηση και αποθήκευση μόνο από και προς μια βάση δεδομένων, γίνεται χρήση Repositories, τα οποία έχουν παρόμοια λειτουργία με τα DAO, με τη μόνη διαφορά πως τα DAO κάνουν συναλλαγές μόνο με τη βάση δεδομένων, ενώ τα Repositories όχι. Με άλλα λόγια τα DAO είναι ένα υποσύνολο των Repositories.

Στο React Grading System, τα DAO της αντιπροσωπεύουν τις 7 Οντότητες (Entities) που υπάρχουν, δηλαδή το χρήστη, τα μαθήματα, τα πρότζεκτ, τις εργασίες, τα τεστ, τις εισόδους και τις εξόδους των τεστ. Οι οντότητες έχουν παρόμοιο ρόλο και πεδία με το

μοντέλο του front end, αντιστοιχίζοντας σε μια κλάση τα δεδομένα που έρχονται μετά από ένα ερώτημα στη βάση.

Στην ειδική περίπτωση του login, πέρα από κλήσεις στη βάση δεδομένων, η αντίστοιχη συνάρτηση ελέγχει τον κωδικό που δόθηκε με τον κρυπτογραφημένο κωδικό που βρίσκεται στη βάση.

Καθώς εν συνεχεία τα DAO είναι υπεύθυνα για τις εντολές που στέλνονται στη βάση και αναλύουν τις εισόδους του χρήστη, είναι και το κομμάτι του κώδικα που είναι πιο ευάλωτο σε SQL και JavaScript Injections. Τα Injections αυτά είναι επιθέσεις που πατάνε στην έλλειψη ασφαλείας και στοχεύουν να τρέξουν κακόβουλο κώδικα.

Τα SQL Injections συγκεκριμένα είναι όταν εισάγονται εντολές SQL σε κάποια φόρμα ή παράμετρο URL, όπως μια εντολή που ρίχνει τη βάση ή κάποιο table αυτής, ή ζητάει τους κωδικούς όλων των χρηστών. Ενώ τα JavaScript Injections είναι αντίστοιχα όταν εισάγεται κώδικας JavaScript σε μια ιστοσελίδα μέσω μιας φόρμας για διάφορες λειτουργίες, όπως η εκτέλεση κάποιας συνάρτησης ή η υποκλοπή δεδομένων πληκτρολογίου.[22] Είναι σημαντικό για μια ιστοσελίδα στα πλαίσια του React Grading System, όπου οι φοιτητές ανεβάζουν τον κώδικα των εργασιών τους οι οποίες πρέπει να αναλυθούν στη συνέχεια από έναν αλγόριθμο, να γίνεται έλεγχος για οποιοδήποτε Injection, το οποίο εν τέλει μπορεί να έγινε και κατά λάθος.

Για την αποφυγή Injections, γίνεται χρήση Παραμετροποιημένων Queries, τα οποία χρησιμοποιούν παραμέτρους για την παροχή δεδομένων στο ερώτημα αντί να τα ενσωματώνει απευθείας σε αυτό. Έτσι, οι εισαγωγές δεδομένων δεν εκτελούνται ως κώδικας SQL αλλά ως απλές τιμές δεδομένων.

Στην περίπτωση React Grading System, το Transaction Manager κάνει χρήση της βιβλιοθήκης **mysql** στις συναλλαγές της, το οποίο στη σύνταξη του αντικαθιστά τα μέρη όπου υπάρχουν τιμές στη γραμμή της SQL με ερωτηματικά (**?**). Ακολουθεί ένα απλό παράδειγμα χωρίς και με Παραμετροποιημένα Queries, όπου **tm** είναι το Transaction Manager που έχει δοθεί με Dependency Injection σε ένα μία συνάρτηση του Repository, και **.query** είναι η συνάρτηση που στέλνει ένα ερώτημα στη βάση:

```
let userInput = `'; DROP TABLE users; --`
let sqlQuery = `SELECT * FROM users WHERE username = '${userInput}';`
tm.query(sqlQuery)
```

Ο παραπάνω κώδικας κλείνει το ερώτημα που στέλνεται στη βάση, και ξεκινάει ένα δεύτερο κακόβουλο, μετατρέποντας επίσης το επόμενο μέρος του κώδικα σε σχόλιο. Ως αποτέλεσμα θα τρέξει η παρακάτω γραμμή στη βάση.

```
SELECT * FROM users WHERE username = ''; DROP TABLE users; --';
```

Με τη χρήση Παραμετροποιημένων Queries αυτό αποφεύγεται, καθώς ο κώδικας εισάγεται με τον παρακάτω τρόπο:

```
let userInput = `'; DROP TABLE users; --`
let sqlQuery = `SELECT * FROM users WHERE username = ?;`
tm.query(sqlQuery, userInput)
```

Τέλος, ένα μέρος ακόμα του Persistence Layer είναι και το configuration της βάσης δεδομένων που εισάγεται σαν παράμετρος στο Transaction Manager. Το configuration παίρνει παραμέτρους τις παρακάτω:

- **host**, η διεύθυνση του διακομιστή
- **user**, το όνομα του χρήστη που συνδέεται στο διακομιστή
- **password**, ο κωδικός του προαναφερόμενου χρήστη
- **port**, το port στο οποίο θα γίνει η σύνδεση
- **database**, σε ποια βάση θα συνδεθεί

Οι παράμετροι αυτοί έρχονται από ένα αρχείο Περιβαλλοντικών Μεταβλητών **.env** της αντίστοιχης βιβλιοθήκης **dotenv**. Ο σκοπός των Περιβαλλοντικών Μεταβλητών σαν ιδεολογία είναι τόσο η εύκολη χρήση τους από οποιοδήποτε μέρος του κώδικα, όσο και η προστασία ευαίσθητων πληροφοριών όταν ο κώδικας ανεβαίνει σε δημόσιους χώρους (πχ GitHub) καθώς το αρχείο τους δεν ανεβαίνει.[24]

Αν και στα πλαίσια της πτυχιακής εργασίας που τρέχει τοπικά στο περιβάλλον κάποιου χρήστη δεν υπάρχει κίνδυνος υποκλοπής ευαίσθητων πληροφοριών, προσομοιώνεται όσο περισσότερο γίνεται το μοντέλο μιας πραγματικής εφαρμογής σε πραγματικό περιβάλλον.

Υπάρχει Περαιτέρω ένα δεύτερο αρχείο **.env.example** το οποίο έχει τις μεταβλητές που μπορεί να χρησιμοποιηθούν για να δουλέψει σωστά η εφαρμογή με λίγα βήματα κατά την κλωνοποίηση της από το GitHub.

### (Ενότητα 4.4.3) Ροή Δεδομένων

Στο back end του React Grading System, τα δεδομένα περνάνε από διάφορα στάδια μέχρι να παραδοθούν στο front end ή στη βάση, ακολουθώντας τους κανόνες της Αρχιτεκτονικής Διαστρωμάτωσης και τις μορφές που ειπώθηκαν παραπάνω.

Συγκεκριμένα, όταν το View Layer λαμβάνει μια κλήση από το front end, τρέχει μια συνάρτηση ονόματι **transact** η οποία είναι υπεύθυνη να ξεκινήσει μία νέα συναλλαγή και να τη δει ως το τέλος της. Τα δύο σενάρια είναι σε επιτυχία να τρέξει τη συνάρτηση **commit** της συναλλαγής που είναι υπεύθυνη να επικυρώσει τις αλλαγές στη βάση και να επιστρέψει κάποιο αποτέλεσμα, και σε σφάλμα να τρέχει τη συνάρτηση **rollback** της συναλλαγής και να γυρίσει τη βάση στο στάδιο που βρισκόταν πριν την κλήση, επιστρέφοντας κατάλληλο κωδικό σφάλματος.

Η transact δέχεται σαν παραμέτρους ένα νέο instance κάποιου Manager, και μάλιστα πιο συγκεκριμένα μια συγκεκριμένη συνάρτηση του. Ο Manager που δημιουργείται παίρνει σαν παράμετρο τη νέα συναλλαγή και με τη σειρά του δημιουργεί στον constructor του ένα instance του DAO που θα χρειαστεί να καλέσει, δίνοντας του τη συναλλαγή.

Το DAO χρησιμοποιεί τη συναλλαγή για να τρέξει μια συνάρτηση του Transaction Manager ονόματι **query** που εκτελεί τις εντολές της SQL. Αρχικά, η query ελέγχει το αν η συναλλαγή είναι ανοιχτή ή ολοκληρωμένη, και στην πρώτη περίπτωση πραγματοποιεί το ερώτημα στη βάση, ενώ στη δεύτερη επιστρέφει σφάλμα, έτσι απαγορεύοντας να τρέξουν παράλληλα ερωτήματα.

Αν η query ολοκληρωθεί με επιτυχία, τα ενδεχόμενα είναι διαφορετικά ανάλογα με το τι ενέργεια έχει πραγματοποιηθεί. Αν η ενέργεια ήταν GET (δηλαδή αν έχουν ζητηθεί δεδομένα, το αποτέλεσμα του ερωτήματος που είναι ένα γενικό αρχείο τύπου JSON ή ένας πίνακας, αντιστοιχείται σε μια κλάση Οντότητας ή σε ένα Πίνακα Οντοτήτων πριν επιστραφεί. Αν η ενέργεια ήταν POST, PUT ή DELETE (δηλαδή αν ανέβηκαν, τροποποιήθηκαν ή διαγράφτηκαν δεδομένα), δεν επιστρέφεται τίποτα, και η επιβεβαίωση του αν το ερώτημα ολοκληρώθηκε βασίζεται στην επιτυχία της συναλλαγής.

## (Υποκεφάλαιο 4.5) Database

### (Ενότητα 4.5.1) Εγκατάσταση Βάσης

Η βάση δεδομένων έχει εγκατασταθεί και παραμετροποιηθεί μέσω MySQL Workbench και σε τυχόν κλωνοποίηση από το GitHub είναι απαραίτητο να ακολουθηθούν κάποια βήματα για την ορθή λειτουργία της, τα οποία περιλαμβάνονται και ως οδηγίες στο αρχείο README.md στο φάκελο του back end. Πιο συγκεκριμένα:

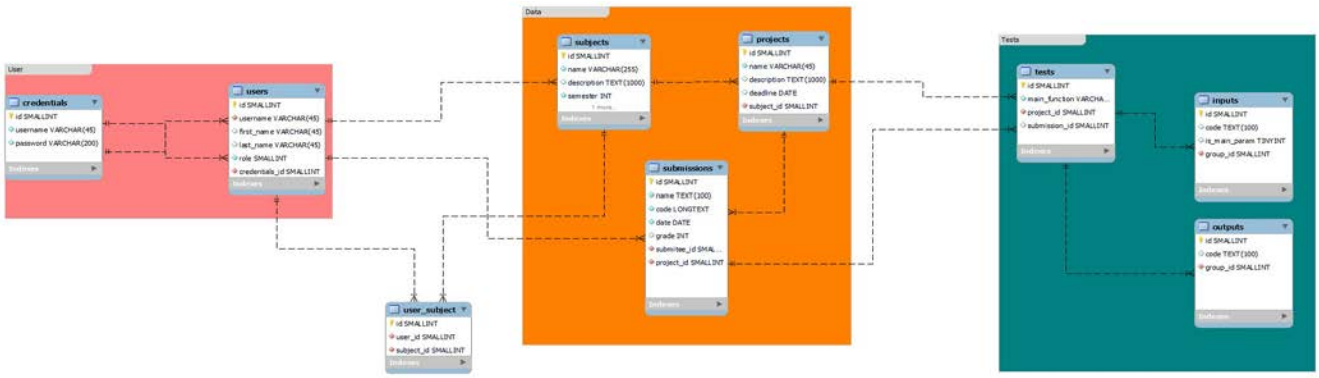
1. Μετά το άνοιγμα του MySQL Workbench, ο χρήστης πρέπει να πλοηγηθεί στο MySQL Connections. Από εκεί μετά να δημιουργήσει μία νέα σύνδεση ονόματι **GradingSystem** και να αφήσει όλες τις υπόλοιπες ρυθμίσεις ως έχουν.
2. Πριν τη δημιουργία θα ζητηθεί να εισαχθούν τα αναγνωριστικά του root. Το όνομα χρήστη είναι root, και όσον αφορά τον κωδικό πρόσβασης, αν είναι η πρώτη σύνδεση στο Workbench, ο χρήστης ακολουθεί τα παρακάτω βήματα ανάλογα το λογισμικό:
    - Σε περιβάλλον Windows παρατηρήθηκε πως ο κωδικός είναι συνήθως κενός, δηλαδή ο χρήστης μπορεί να συνδεθεί πατώντας μόνο το όνομα χρήστη.
    - Σε περιβάλλον Linux υπάρχει ένας προσωρινός πρώτος κωδικός ο οποίος μπορεί να μαθευτεί πληκτρολογώντας την παρακάτω εντολή στο bash `**sudo grep 'temporary password' /var/log/mysqld.log**`

3. Μετά θα ζητηθεί να πληκτρολογηθεί ένας νέος κωδικός πρόσβασης, ο οποίος στα πλαίσια της εργασίας πρέπει να είναι ο ίδιος με αυτόν που αναφέρεται στο πεδίο PASSWORD του .env.example αρχείου του back end.
4. Αφού ολοκληρωθεί η σύνδεση, πριν την ανοίξει ο χρήστης, πρέπει να πλοηγηθεί από την αρχική σελίδα στα Models όπου και πρέπει να φορτώσει το μοντέλο της βάσης όπως αποθηκεύτηκε στο φάκελο **/backend/database/schema.mwb** της εφαρμογής.
5. Στο νέο μοντέλο, ο χρήστης πλοηγείται από την μπάρα στην κορυφή στο **Database > Forward Engineer** και επιλέγει στη σύνδεση το GradingSystem που ορίστηκε προηγουμένως. Στην συνέχεια πατάει στο Next μέχρι να τελειώσει.
6. Από την αρχική σελίδα, ο χρήστης πλοηγείται στη σύνδεση, δημιουργεί ένα νέο query και επικολλεί τα inserts που υπάρχουν στο αρχείο **/backend/database/db.sql** και το τρέχει
7. Αν όλα πήγαν καλά, σε ένα νέο query τρέχει τις παρακάτω εντολές
    - **ALTER USER 'root'@'localhost' IDENTIFIED WITH mysql_native_password BY 'xxx';** (όπου xxx ο κωδικός που βρίσκεται στο .env.example)
    - **flush privileges;**

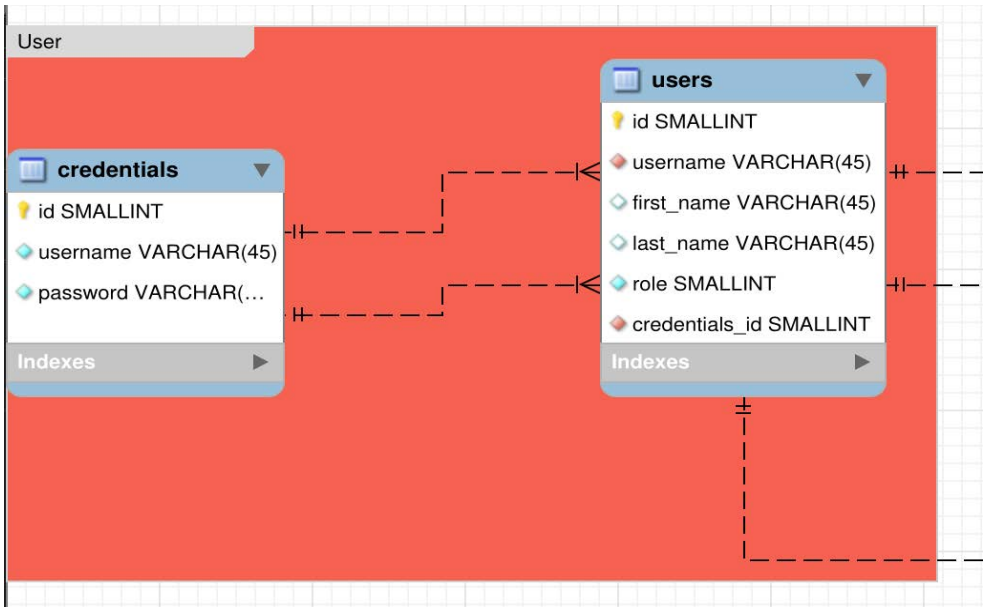
    Αυτό σιγουρεύει πως με το που ξεκινήσει το back end από τα σενάρια που δόθηκαν στο 4.2 δε θα πέσουμε σε ER_NOT_SUPPORTED_AUTH_MODE. Αυτό το σφάλμα δημιουργείται λόγω του ότι το back end χρησιμοποιεί τη βιβλιοθήκη mysql αντί για mysql2 η οποία θεωρείται παλαιότερη και μπορεί να συμβιβάζει μερικά θέματα ασφαλείας. Είναι γενικά πιο ασφαλές να αναβαθμιστεί το back end σε mysql2, αλλά στα πλαίσια της πτυχιακής που τρέχει σε τοπικό διακομιστή δεν αποτελεί κάποιο σοβαρό θέμα.

## (Ενότητα 4.5.2) Σχήμα Βάσης

Όπως προαναφέρθηκε, το σχήμα της βάσης είναι σχεσιακό. Αποτελείται από 9 πίνακες οι οποίοι είναι όλοι συνδεδεμένοι μεταξύ τους. Ακολουθεί το Διάγραμμα EER και επεξήγηση των πινάκων:



Όπως βλέπουμε, υπάρχουν 3 βασικές ομαδοποιήσεις των πινάκων. Με το κόκκινο απεικονίζονται οι πίνακες που αφορούν τους χρήστες, με πορτοκαλί οι πίνακες που αφορούν τα βασικά δεδομένα της σελίδας, και με πράσινο οι πίνακες μου αφορούν τα τεστ. Ο πίνακας που βρίσκεται εκτός των ομάδων είναι Πίνακας Συσχετίσεων (Association Table). Τα πεδία των πινάκων είναι κατά μεγάλο βαθμό και τα ίδια που λαμβάνουν οι οντότητες του back end και το μοντέλο του front end, με μικρές διαφορές όπως σε foreign keys. Ακολουθεί μια πιο κοντινή εικονογράφηση των τριών ομάδων των πινάκων μαζί με επεξήγηση τους.



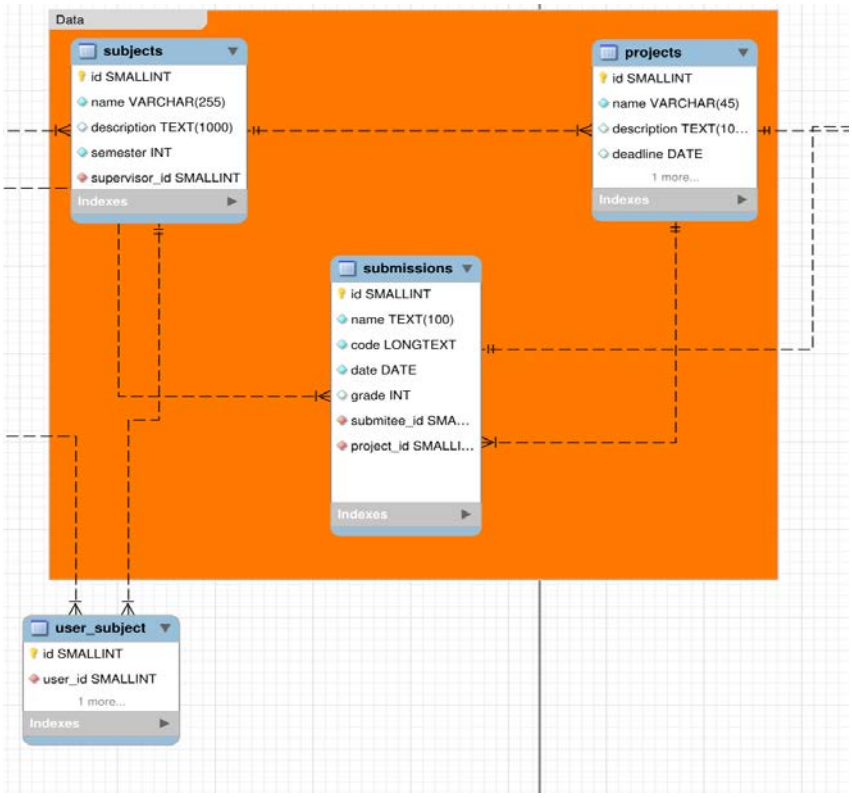

Στην ομάδα των χρηστών υπάρχουν δύο πίνακες, **credentials** και **users**. Η αντιστοίχιση τους είναι ένας προς έναν, καθώς δεν υπάρχει περίπτωση τα credentials ποτέ να ανήκουν σε πάνω από έναν χρήστες.

Ο σκοπός του διαχωρισμού τους είναι πως ο πρώτος έχει αποθηκευμένα μόνο τα δεδομένα που χρησιμοποιούνται για το login, ο δεύτερος έχει όλες τις πληροφορίες του χρήστη. Κάτι τέτοιο είναι απαραίτητο για την ασφάλεια των δεδομένων, ειδικά αφού το login είναι το μόνο endpoint του back end που μπορεί οποιοσδήποτε μη συνδεδεμένος χρήστης να καλέσει, έτσι "κλειδώνοντας" την πρόσβαση σε ευαίσθητες πληροφορίες. Τα πεδία των πινάκων είναι παρακάτω:

1. **credentials**
   - **id**
     - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
     - Το μοναδικό id
   - **username**
     - VARCHAR(45), NOT NULL, UNIQUE
     - Το όνομα χρήστη

- **password**
  - VARCHAR(200), NOT NULL
  - Ο κωδικός πρόσβασης

2. **users**
   - **id**
     - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
     - Το μοναδικό id
   - **username**
     - FOREIGN KEY του username του credentials
   - **first_name**
     - VARCHAR(45)
     - Το μικρό όνομα του χρήστη
   - **last_name**
     - VARCHAR(45)
     - Το επίθετο του χρήστη
   - **role**
     - SMALLINT, NOT NULL
     - Ο ρόλος του χρήστη
   - **credentials_id**
     - FOREIGN KEY του id του credentials

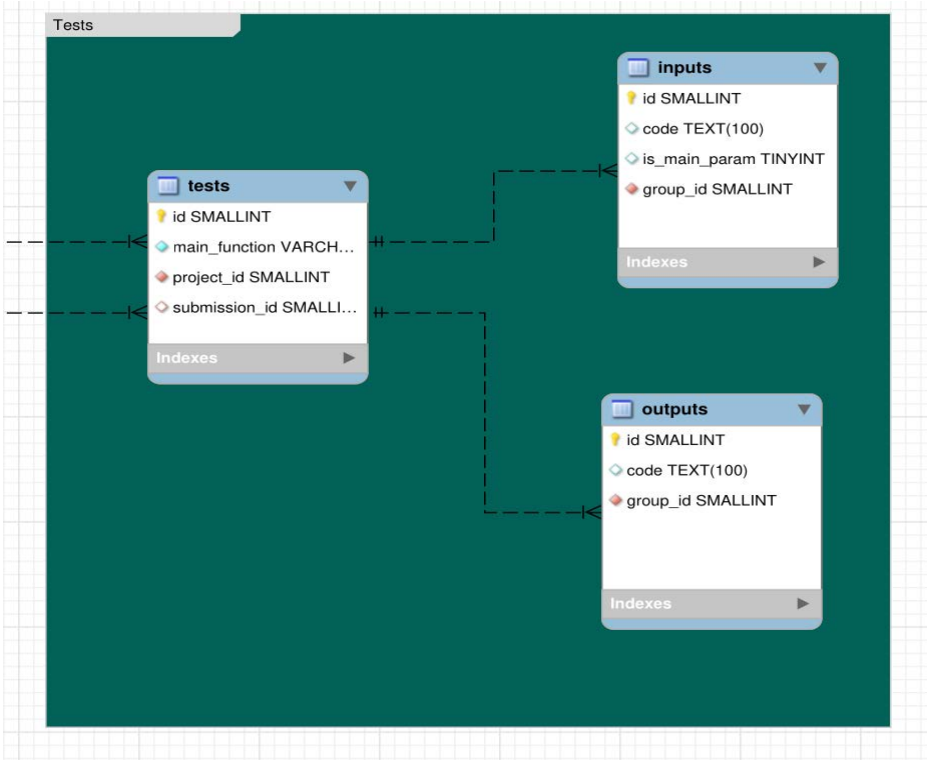


Στην ομάδα των δεδομένων υπάρχουν τρεις πίνακες, **subjects**, **projects** και **submissions**. Και οι τρεις αφορούν όσα πράγματα εν τέλει εμφανίζονται στη διεπαφή του χρήστη. Η αντιστοίχιση τους είναι ένας προς πολλά, καθώς το κάθε μάθημα μπορεί να έχει πολλά πρότζεκτ και αυτά με τη σειρά τους να έχουν πολλές ανεβασμένες εργασίες. Τα πεδία των πινάκων είναι παρακάτω:

1. **subjects**
   - **id**
     - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
     - Το μοναδικό id
   - **name**
     - VARCHAR(255), NOT NULL
     - Το όνομα του μαθήματος
   - **description**
     - TEXT(1000)
     - Η περιγραφή του μαθήματος
   - **semester**
     - VARCHAR(45)
     - Το εξάμηνο του μαθήματος.
   - **supervisor_id**
     - FOREIGN KEY του id του users
     - Αφορά το ποιος είναι ο διδάσκων του μαθήματος.
2. **projects**
   - **id**
     - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
     - Το μοναδικό id
   - **name**
     - VARCHAR(255), NOT NULL
     - Το όνομα του πρότζεκτ
   - **description**
     - TEXT(1000)
     - Η περιγραφή του πρότζεκτ
   - **subject_id**
     - FOREIGN KEY του id του subject
     - Αφορά το σε ποιο μάθημα ανήκει το πρότζεκτ
3. **submissions**
   - **id**
     - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
     - Το μοναδικό id

- **name**
  - TEXT(100), NOT NULL
  - Το όνομα του παραδοτέου αρχείου. Σε αντίθεση με άλλα ονόματα, δόθηκε TEXT στο πεδίο για να μπορεί σε δεύτερο χρόνο να μεγαλώσει εύκολα το μέγιστο μέγεθος όπως είναι και πολύ πιθανό να χρειαστεί.
- **code**
  - LONGTEXT, NOT NULL
  - Ο κώδικας της εργασίας
- **date**
  - DATE, NOT NULL
  - Η ημερομηνία παράδοσης της εργασίας
- **grade**
  - INT
  - Ο βαθμός της εργασίας
- **submitee_id**
  - FOREIGN KEY του id του users
  - Αφορά τον φοιτητή που παρέδωσε
- **project_id**
  - FOREIGN KEY του id του projects
  - Αφορά το για ποιο πρότζεκτ ανέβηκε η εργασία

Στην ομάδα αυτή μπορεί να κατηγοριοποιηθεί και ο Πίνακας Συσχετίσεων **user_subjects**, ο οποίος κρατάει το id ενός χρήστη κι ενός μαθήματος, ώστε να γνωρίζει η βάση ποιος χρήστης παρακολουθεί ποια μαθήματα.



Στην ομάδα των τεστ υπάρχουν τρεις πίνακες, **tests**, **inputs** και **outputs**. Η σχέση τους με την ομάδα των δεδομένων είναι ένα προς πολλά, καθώς το κάθε πρότζεκτ μπορεί να έχει πολλά τεστ. Η αντιστοίχιση στην δική τους ομάδα είναι ένα προς πολλά για τα τεστ και τις εισόδους, καθώς το κάθε τεστ μπορεί να έχει πολλές εισόδους, και ένα προς ένα για τις εξόδους, καθώς το κάθε τεστ έχει μόνο μία έξοδο. Τα πεδία των πινάκων είναι παρακάτω:

1. **tests**
    - **id**
        - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
        - Το μοναδικό id
    - **main_function**
        - VARCHAR(45), NOT NULL
        - Το όνομα της κύριας συνάρτησης πάνω στην οποία θα δράσει το τεστ
    - **project_id**
        - FOREIGN KEY του id του projects
        - Αφορά το πρότζεκτ στο οποίο ανήκει το τεστ
    - **submission_id** (αχρησιμοποίητο)
        - SMALLINT
        - Ένα πεδίο που θα χρησιμοποιούταν σε μια επέκταση της εφαρμογής στην οποία ο καθηγητής θα μπορούσε προαιρετικά να αλλάζει τα τεστ των πρότζεκτ και να τα αποθηκεύει αλλαγμένα για την κάθε εργασία ξεχωριστά
2. **inputs**
    - **id**
        - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
        - Το μοναδικό id
    - **code**
        - TEXT(100)
        - Ο κώδικας εισόδου
    - **is_main_param** (development only)
        - TINYINT
        - Ένα πεδίο που χρησιμοποιήθηκε στο development για σκοπούς debugging και δεν έχει πραγματική χρήση. Μετράει το αν το input θα μπει σαν παράμετρος στην κύρια συνάρτηση του τεστ ή όχι
    - **group_id**
        - FOREIGN KEY του id του tests
        - Αφορά το τεστ στο οποίο ανήκει η είσοδος
3. **outputs**

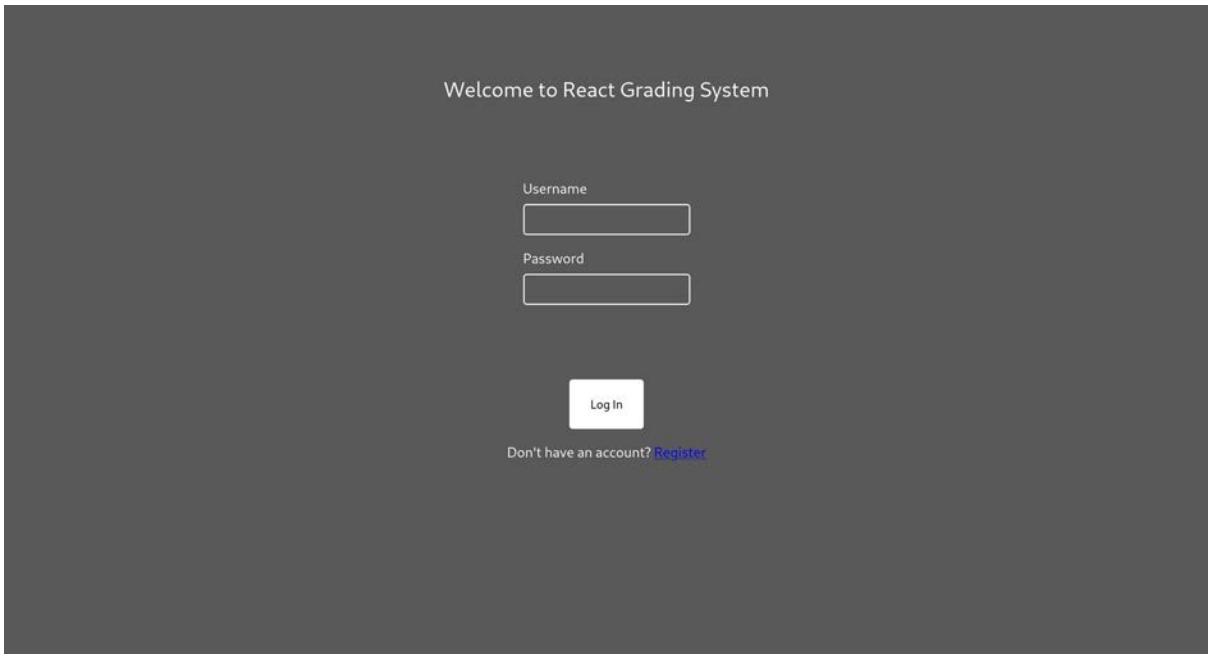
- **id**
  - SMALLINT, PRIMARY KEY, NOT NULL, AUTO_INCREMENT
  - Το μοναδικό id
- **code**
  - TEXT(100)
  - Ο κώδικας εξόδου
- **group_id**
  - FOREIGN KEY του id του tests
  - Αφορά το τεστ στο οποίο ανήκει η είσοδος

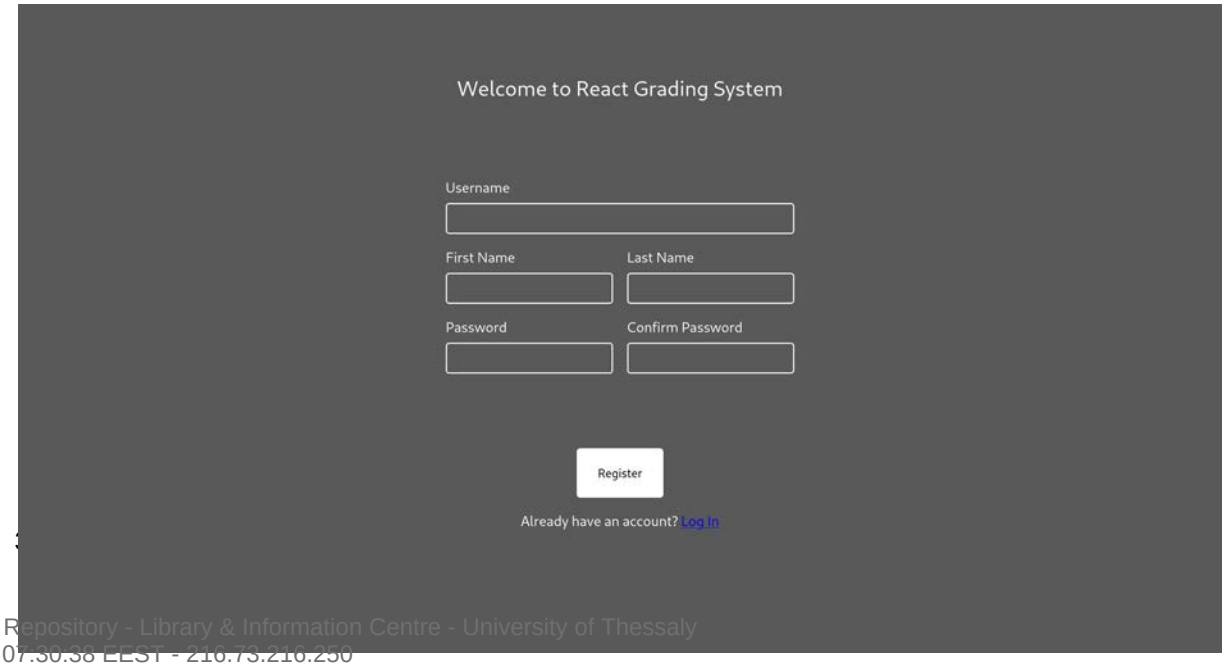
# ΚΕΦΑΛΑΙΟ 5 Επίδειξη React Grading System

Σε αυτό το κεφάλαιο θα επικεντρωθούμε σε μια επίδειξη με απεικονίσεις και επεξηγήσεις της εφαρμογής και των διαφόρων λειτουργιών που αναλύθηκαν προηγουμένως.

## (Υποκεφάλαιο 5.1) Σύνδεση & Αρχική Οθόνη

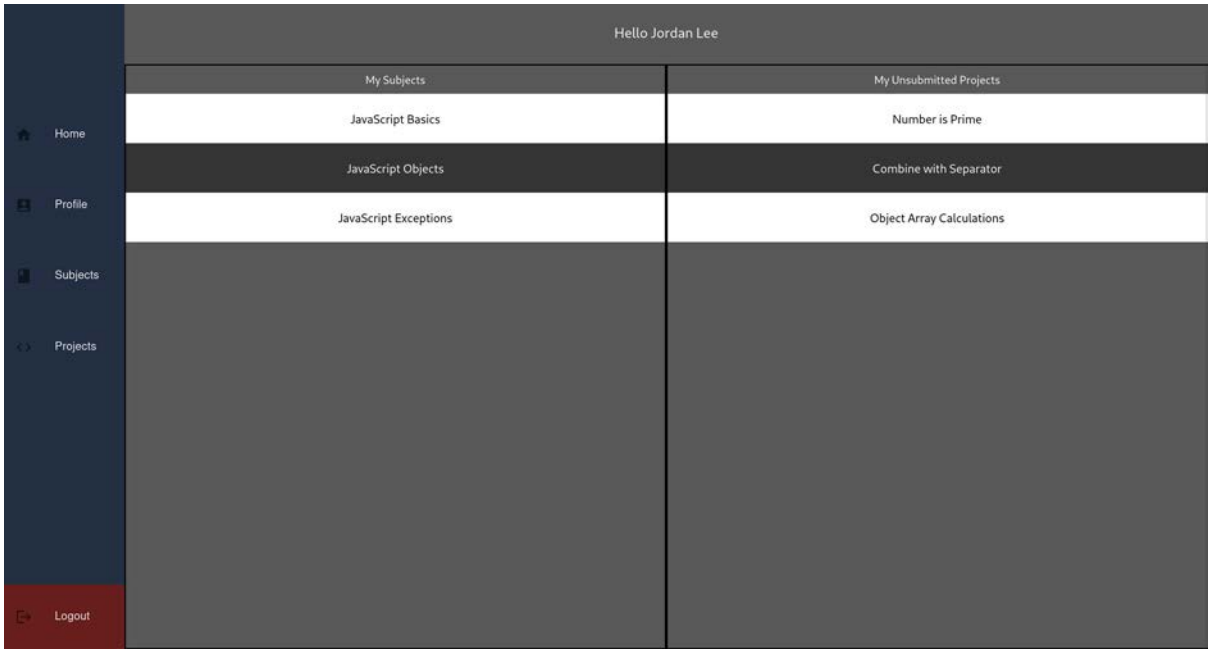
Στην αρχική οθόνη της εφαρμογής περιέχεται μία φόρμα για Log In ή Register. Αξιοσημείωτο είναι πως η αλλαγή στη φόρμα γίνεται μέσω του state και δεν προκαλεί επαναφόρτωση της σελίδας. Αυτό γλιτώνει χρόνο και επιπλέον διατηρεί το πληκτρολογημένο username και password αν αλλαχθεί η μορφή από Log In σε Register.



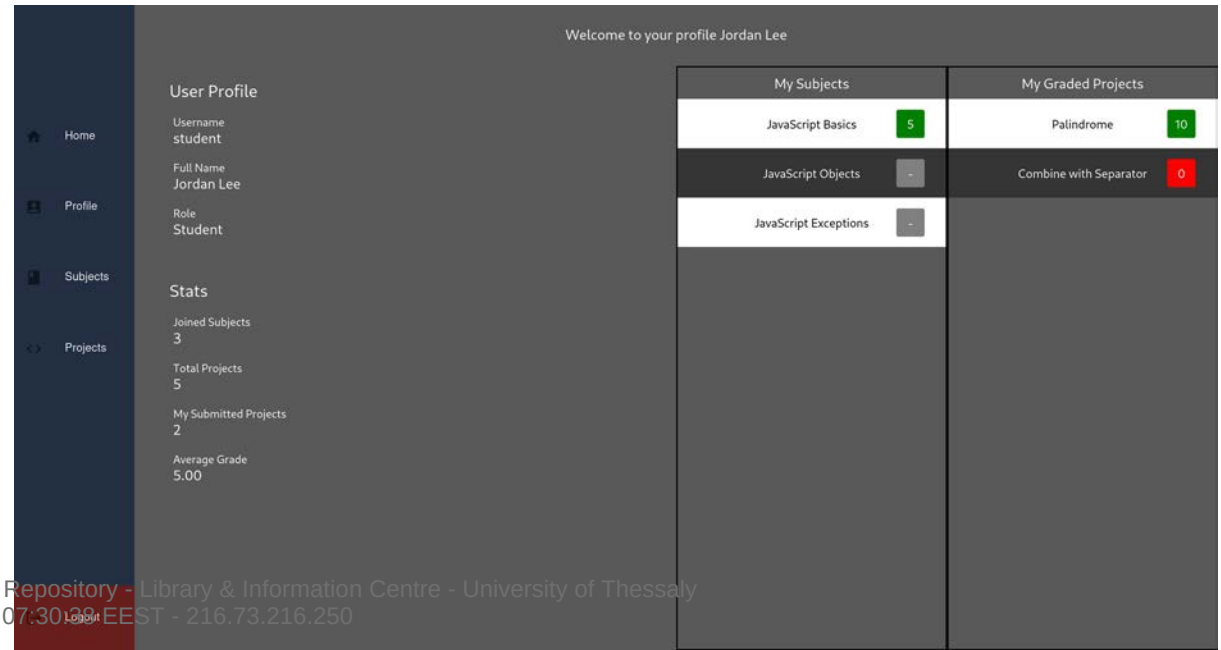



Έστω πως ο χρήστης συνδέεται ως φοιτητής. Από τη στιγμή που συνδεθεί, η σελίδα θα τον ανακατευθύνει στο **/home** που είναι και η αρχική σελίδα.

Η αρχική σελίδα έχει συνοπτικά ένα καλωσόρισμα, τα μαθήματα στα οποία ο χρήστης είναι εγγεγραμμένος, και τα πρότζεκτ αυτών των μαθημάτων που δεν έχουν παραδοθεί. Πατώντας σε οποιοδήποτε οδηγεί το χρήστη στην αντίστοιχη σελίδα τους που θα εξηγηθεί παρακάτω. Επιπλέον, υπάρχει ένα sidebar στα αριστερά που παραμένει σε όλες τις σελίδες και αποσκοπεί στην πλοήγηση στα υπόλοιπα μέρη της ιστοσελίδας.




Πατώντας στο **Profile** της μπάρας, η σελίδα πλοηγείται στο προφίλ του χρήστη, όπου φαίνονται μερικά στατιστικά στοιχεία, καθώς και τα επιλεγμένα του μαθήματα και τα βαθμολογημένα του πρότζεκτ. Και στα μαθήματα και στα πρότζεκτ εμφανίζεται από δίπλα ο βαθμός, και στα μαθήματα είναι ο μέσος όρος των βαθμολογημένων πρότζεκτ που του αντιστοιχούν. Καθώς τα μαθήματα δεν έχουν πάντα πρότζεκτ, αν δεν έχουν δεν εμφανίζεται βαθμός.



Στην περίπτωση βέβαια του καθηγητή είναι λίγο διαφορετικό. Καθώς ο καθηγητής μπορεί να δει τα προφίλ των φοιτητών, μπορεί να δει αυτή τη σελίδα, αλλά δεν υπάρχει νόημα στο να εμφανίζεται βαθμολογία ή βαθμολογημένα πρότζεκτ στο προφίλ του. Για αυτό υπάρχει μόνο η λίστα με τα μαθήματα που διαχειρίζεται καθώς και ξεχωριστά στατιστικά που αφορούν μόνο καθηγητές.

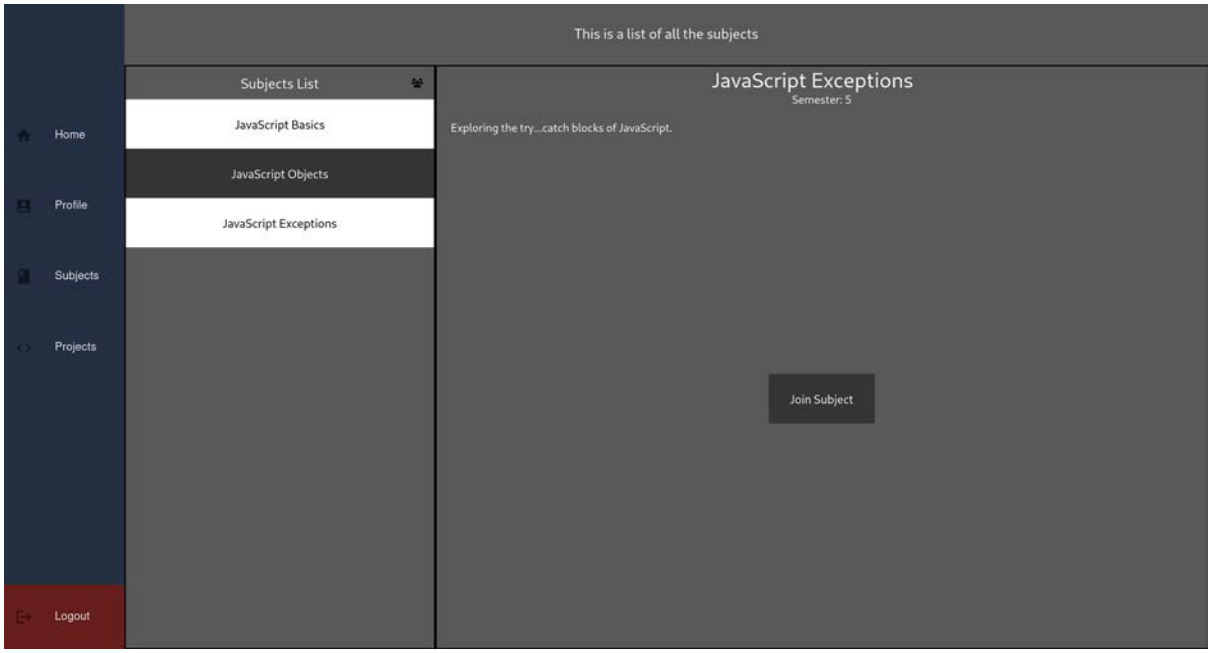

Παρατηρούμε πως στην παραπάνω εικόνα παρόλο που είναι συνδεδεμένος καθηγητής υπάρχουν και στατιστικά για Joined Subjects και για Supervising Subjects. Αυτό οφείλεται στο γεγονός πως ένας καθηγητής μπορεί να συμμετάσχει σε μαθήματα που δεν διαχειρίζεται και να βλέπει τα πρότζεκτ τους, απλά χωρίς να παραδίδει εργασίες. Στην εικόνα τυγχάνει οι δύο αριθμοί να είναι οι ίδιοι, πράγμα το οποίο σημαίνει πως καθώς ο καθηγητής δε συμμετέχει σε παραπάνω μαθήματα από αυτά που διαχειρίζεται.

## (Υποκεφάλαιο 5.2) Σελίδες Διαχείρισης Δεδομένων

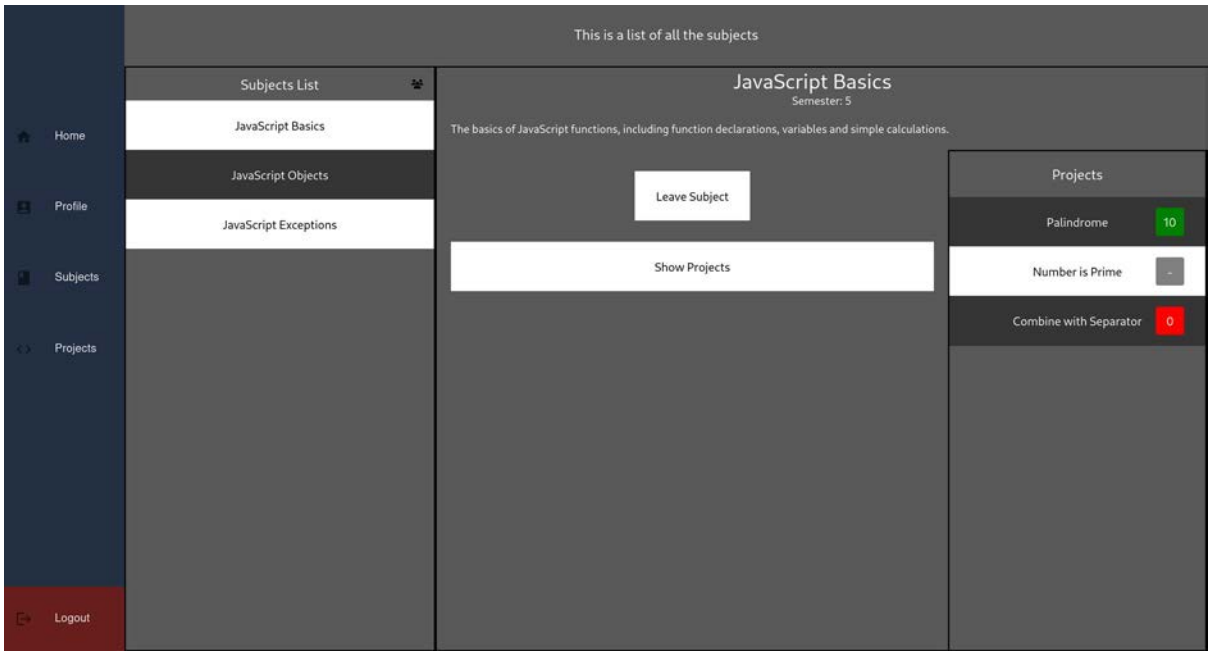
Οι σελίδες διαχείρισης δεδομένων αφορούν αυτές που δείχνουν τις πληροφορίες των μαθημάτων και των πρότζεκτ. Αυτές οι δύο σελίδες μοιάζουν αρκετά τόσο στην εμφάνιση όσο και στη λειτουργικότητά τους, παρουσιάζοντας μέρος των δεδομένων και προσφέροντας διαδραστικότητα, φιλτράρισμα ή επεξεργασία αυτών.

Η σελίδα **Subjects** που αφορά τα μαθήματα, έχει μια λίστα με τα μαθήματα στα αριστερά, από τα οποία μπορεί ο χρήστης να επιλέξει οποιοδήποτε και να διαβάσει τις πληροφορίες του. Δίπλα στη λίστα υπάρχει και ένα κουμπί που ορίζει τιμή στο φίλτρο που αναφέρθηκε στο 4.3.1

Δίνεται η δυνατότητα επίσης, αν ο χρήστης δεν ανήκει στο μάθημα, να γίνει μέλος του.

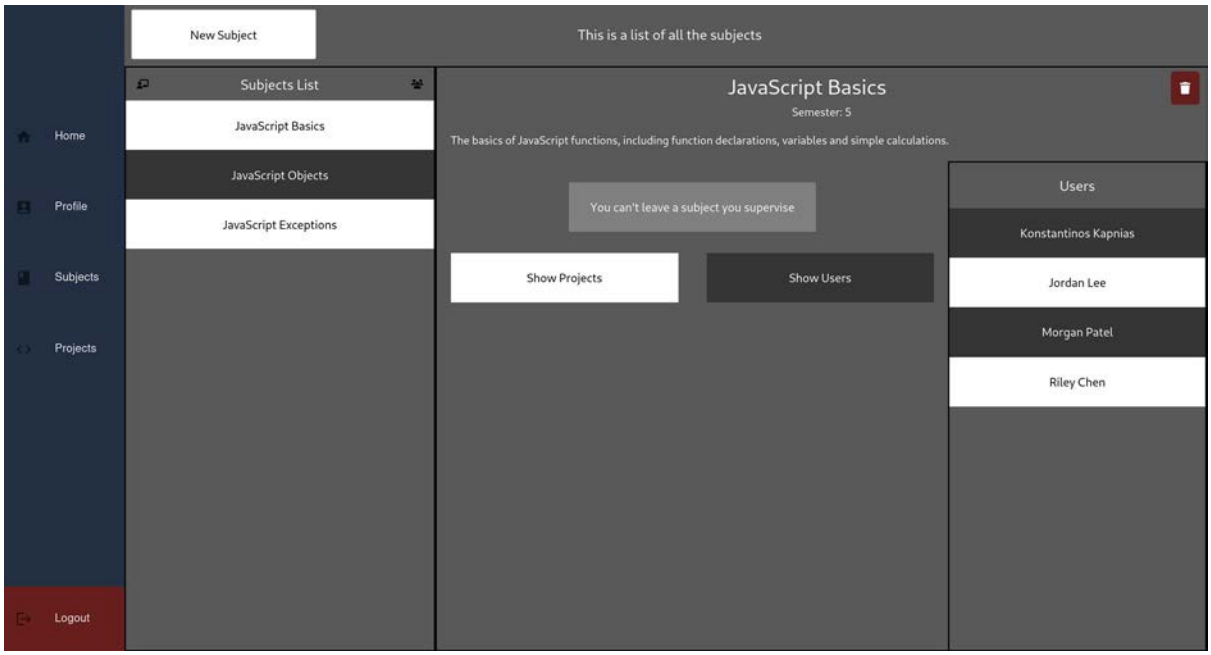



Αφού ο φοιτητής γίνει μέλος του μαθήματος, μπορεί να δει τα πρότζεκτ που είναι διαθέσιμα. Τα πρότζεκτ αυτά θα εμφανιστούν τόσο στην αρχική του σελίδα όσο και στη σελίδα των πρότζεκτ που θα αναλυθεί παρακάτω. Τα πρότζεκτ που εμφανίζονται εδώ έχουν δίπλα και το βαθμό, από τη στιγμή που ο φοιτητής παρέδωσε εργασία για αυτά.

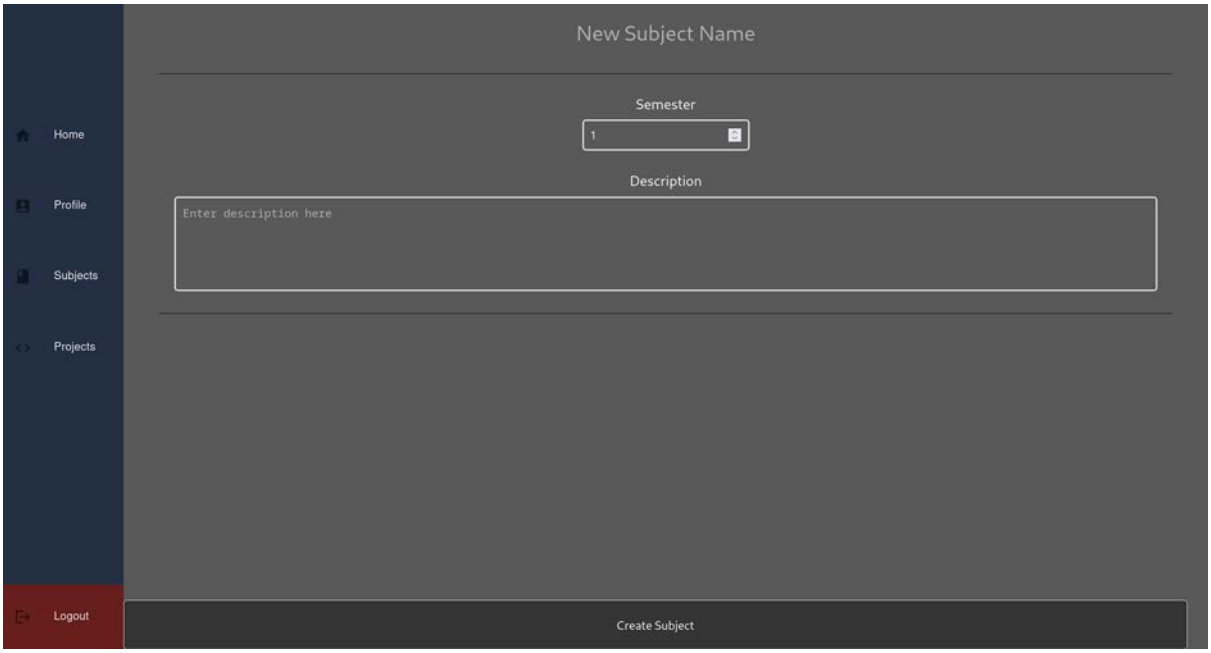



Για τον καθηγητή η διεπαφή είναι λίγο διαφορετική. Το κουμπί για να γίνει μέλος γίνεται μη διαθέσιμο από τη στιγμή που διαχειρίζεται το μάθημα, και στη θέση του προστίθεται ένα κουμπί διαγραφής δίπλα στον τίτλο του μαθήματος. Επιπλέον,

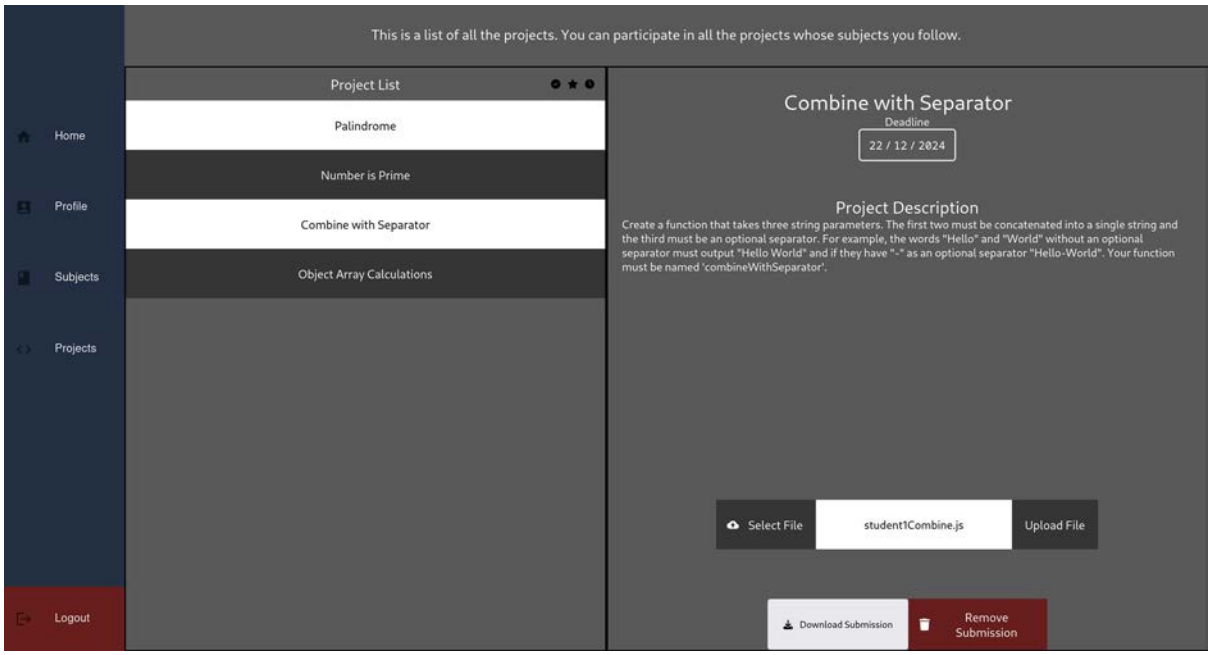
προστίθεται και λίστα με τους φοιτητές που είναι μέλη σε αυτό το μάθημα, των οποίων τα προφίλ μπορεί ο καθηγητής να ελέγξει, και ένα κουμπί για τη δημιουργία νέου μαθήματος.



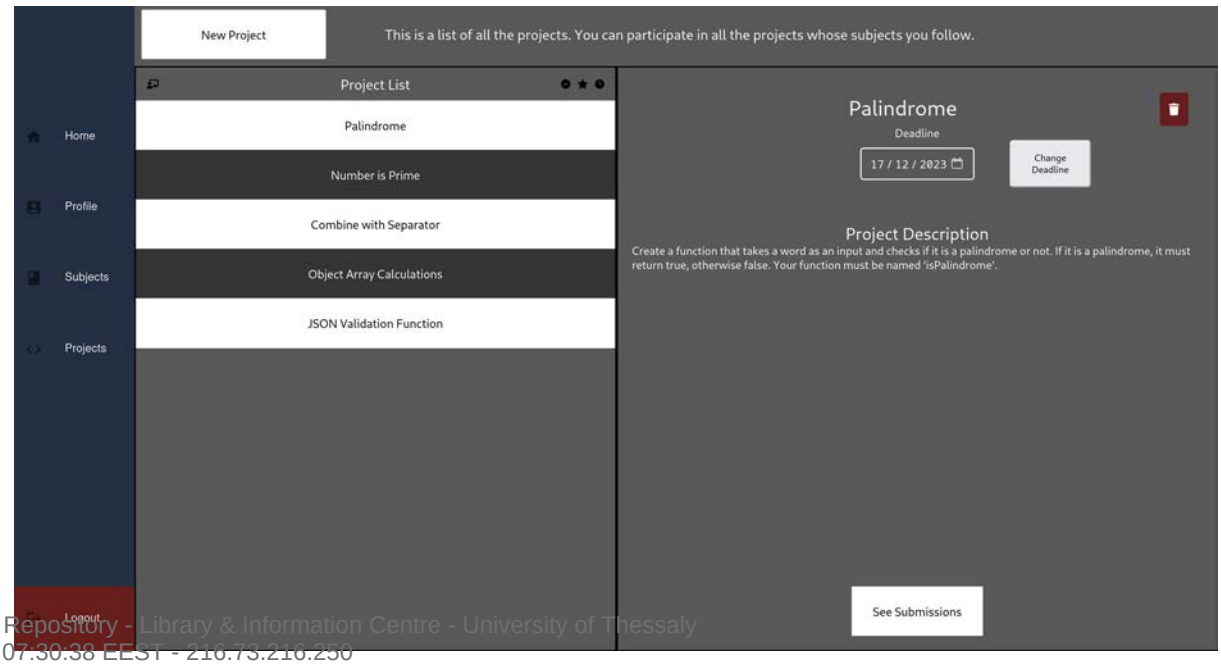

Η δημιουργία ενός νέου μαθήματος είναι απλή, ζητώντας μόνο έναν τίτλο, το εξάμηνο του μαθήματος και μια περιγραφή. Από τη στιγμή που δοθούν αυτά τα πεδία το μάθημα προστίθεται τόσο στη λίστα με όλα τα μαθήματα, όσο και στη λίστα των μαθημάτων που διαχειρίζεται ο καθηγητής.



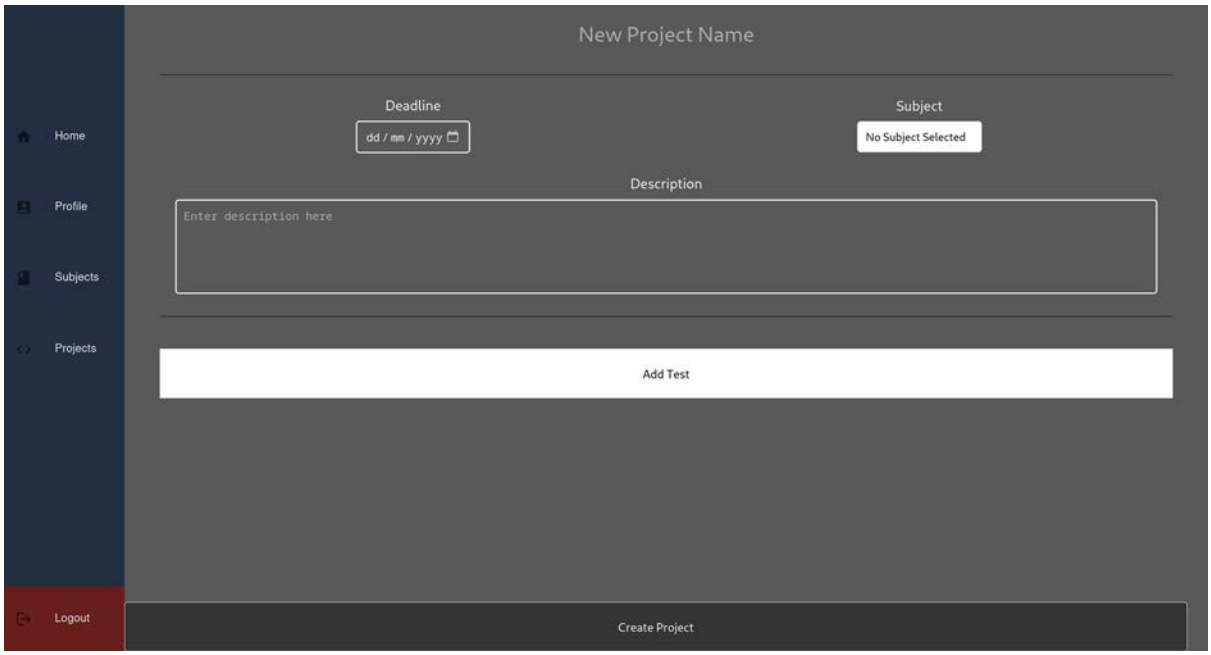
Προχωρώντας από τα μαθήματα, πλοηγώντας στα **Projects**, ο χρήστης βλέπει μια παρόμοια οθόνη, με τη μόνη διαφορά ότι δείχνει όλα τα πρότζεκτ που ανήκουν στα μαθήματα που είναι μέλος ο χρήστης. Μαζί με το όνομα και την περιγραφή δίνεται η ημερομηνία παράδοσης του πρότζεκτ και η φόρμα αποστολής αρχείου εργασίας. Σε αυτή τη φόρμα ο φοιτητής επιλέγει την εργασία του και μπορεί σε δεύτερο χρόνο όσο είναι μέσα στα πλαίσια της προθεσμίας, είτε να την κατεβάσει, είτε να την διαγράψει, είτε να ανεβάσει άλλη. Από τη στιγμή που η προθεσμία λήξει αυτές οι δυνατότητες δεν παρέχονται άλλο.




Και πάλι για τον καθηγητή η διεπαφή διαφέρει. Αντί να υπάρχει φόρμα αποστολής αρχείου, υπάρχει ένα κουμπί προβολής των εργασιών που έχουν ήδη ανέβει, που οδηγεί στη σελίδα βαθμολόγησης. Δίνεται επίσης η δυνατότητα στον καθηγητή τόσο να διαγράψει το πρότζεκτ όσο και να αλλάξει την προθεσμία της εργασίας και να φτιάξει ένα νέο πρότζεκτ.

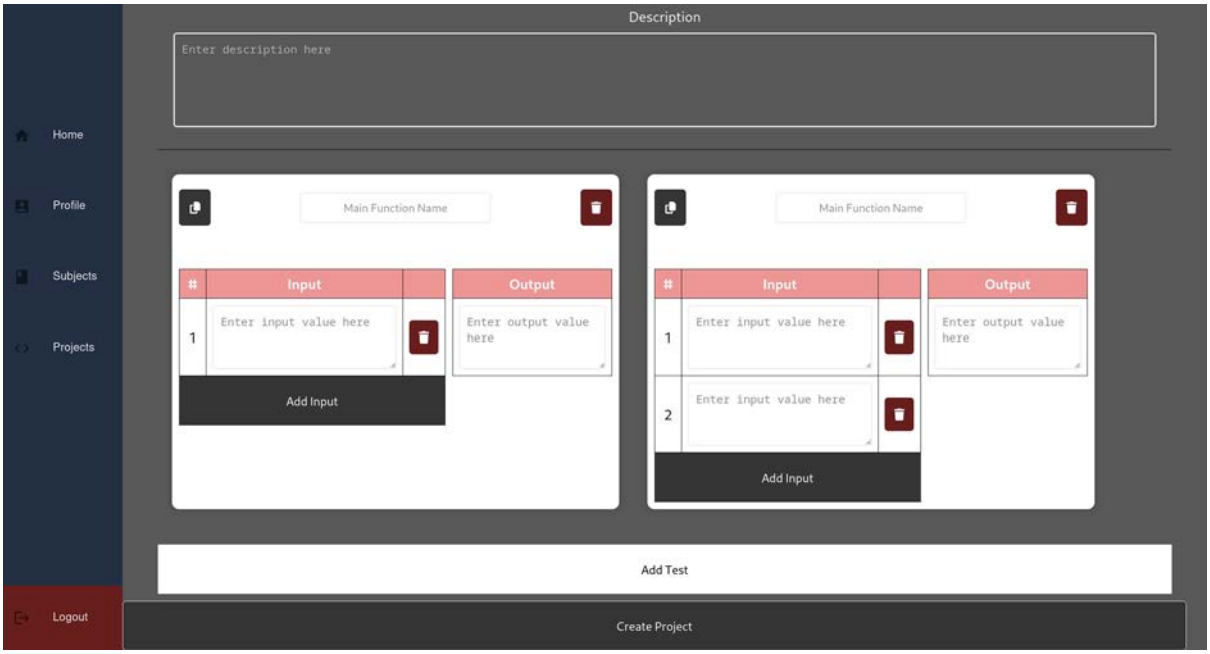


Η σελίδα νέου πρότζεκτ απαιτεί τη συμπλήρωση ονόματος, αρχικής προθεσμίας, σε ποιο μάθημα θα ανήκει, και περιγραφή. Δίνεται επίσης η επιλογή για τη δημιουργία νέων τεστ.



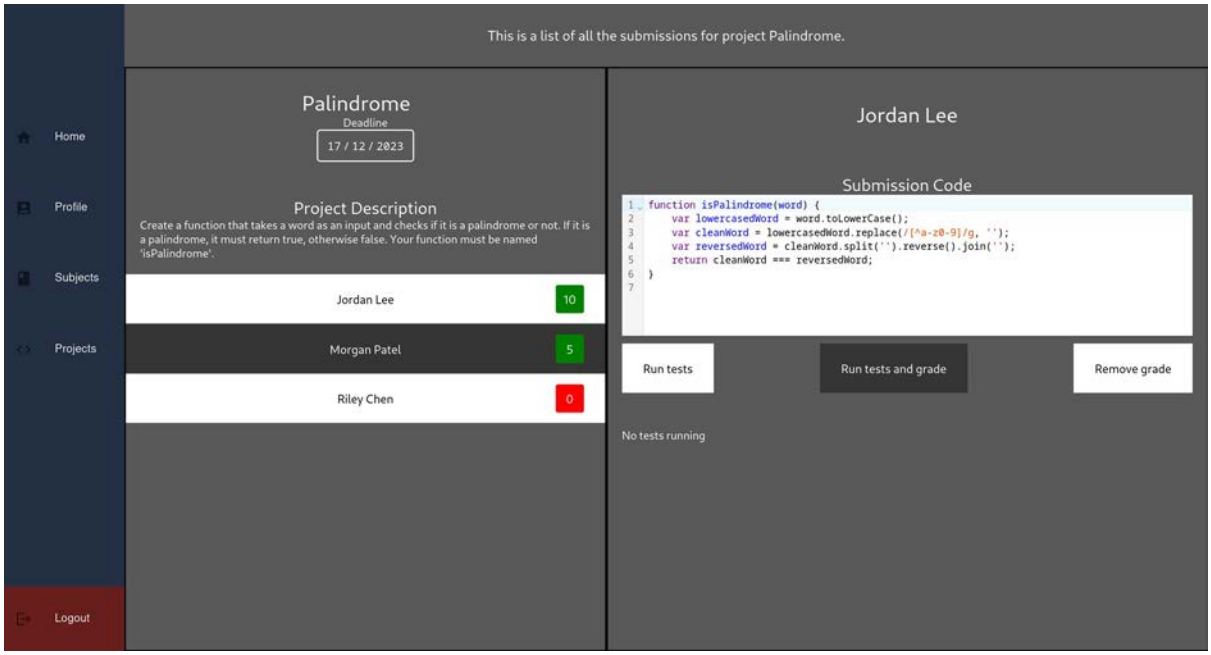

Ένα νέο τεστ χρειάζεται τη συμπλήρωση των εξής: Μιας κύριας συνάρτησης, από μηδέν έως άπειρες εισόδους, και μία έξοδο. Για τη διευκόλυνση του καθηγητή κατά τη συγγραφή των τεστ, δίνεται η δυνατότητα αντιγραφής ενός, καθώς πολλές φορές τεστάρονται οι ίδιες συναρτήσεις με μικρές διαφορετικές στις εισόδους.

Ο αλγόριθμος επίσης αναγνωρίζει αν έχει δοθεί σαν είσοδος ή έξοδος κάποιος πίνακας ή αντικείμενο, στέλνοντας μήνυμα στον καθηγητή αν έκανε κάποιο συντακτικό λάθος όπως η έλλειψη μιας αγκύλης.




## (Υποκεφάλαιο 5.3) Αξιολόγηση Εργασιών

Το τελευταίο κομμάτι της εφαρμογής αφορά μόνο τους καθηγητές και είναι η αξιολόγηση των εργασιών. Αφού πατηθεί το κουμπί **See Submissions** κάτω από ένα πρότζεκτ, ο καθηγητής έρχεται σε μια νέα οθόνη η οποία μοιάζει με τις προηγούμενες. Στην αριστερή της μεριά υπάρχουν συνοπτικά οι πληροφορίες του πρότζεκτ και από κάτω τους τα ονόματα των φοιτητών που παρέδωσαν εργασία, μαζί με τη βαθμολογία τους αν έχουν βαθμολογηθεί. Πατώντας σε οποιονδήποτε φοιτητή φορτώνει στη δεξιά μεριά τον επεξεργαστή κώδικα CodeMirror μαζί με τρία κουμπιά για το πέρασμα του κώδικα από τα τεστ χωρίς βαθμολόγηση, το πέρασμα με βαθμολόγηση, και τη διαγραφή βαθμολόγησης για μια εργασία.



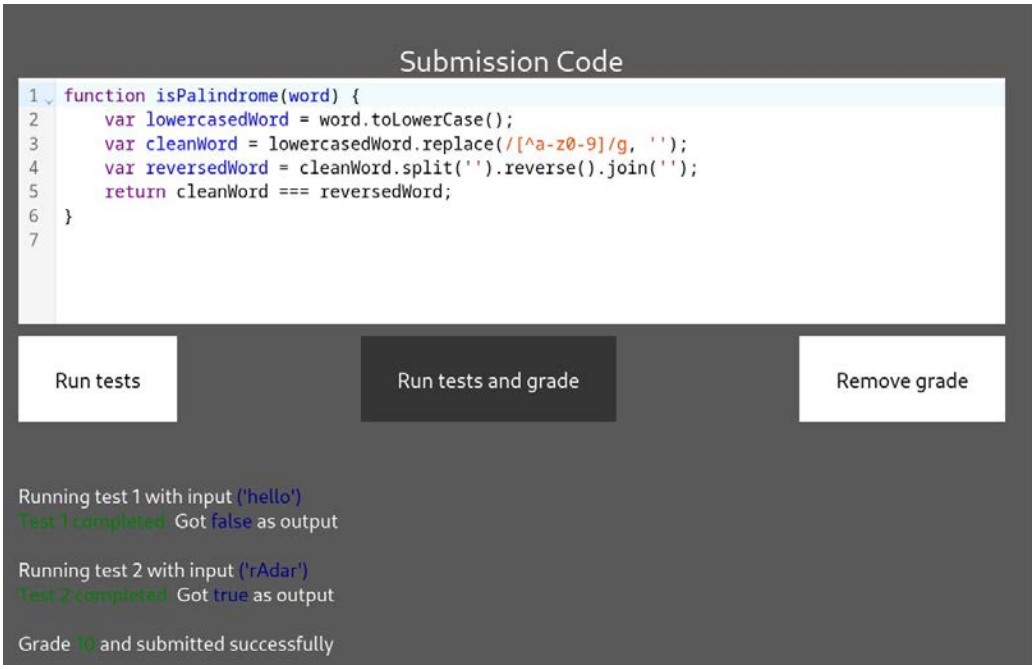

Στο παράδειγμα που θα δούμε, η εκφώνηση του πρότζεκτ είναι η εξής:

*Φτιάξτε μια συνάρτηση που παίρνει έναν λέξη ως είσοδο και ελέγχει αν είναι παλίνδρομη ή όχι. Αν είναι παλίνδρομη, πρέπει να επιστρέφει true, αλλιώς false. Η συνάρτησή σας πρέπει να ονομάζεται 'isPalindrome'.*

Για αυτή τη συνάρτηση ο καθηγητής έχει φτιάξει δύο τεστ:

- Τεστ 1
  - Είσοδος: hello
  - Έξοδος: false
- Test 2
  - Είσοδος: rAdar
  - Έξοδος: true

Ο πρώτος φοιτητής παρέδωσε την παρακάτω συνάρτηση:



Η συνάρτηση παίρνει μια λέξη, τη μετατρέπει σε μικρά γράμματα, στη συνέχεια αφαιρεί όσους χαρακτήρες δεν είναι αλφαριθμητικοί, μετά την αντιστρέφει, και τέλος ελέγχει αν η τελική αυτή λέξη είναι η ίδια με την αρχική.

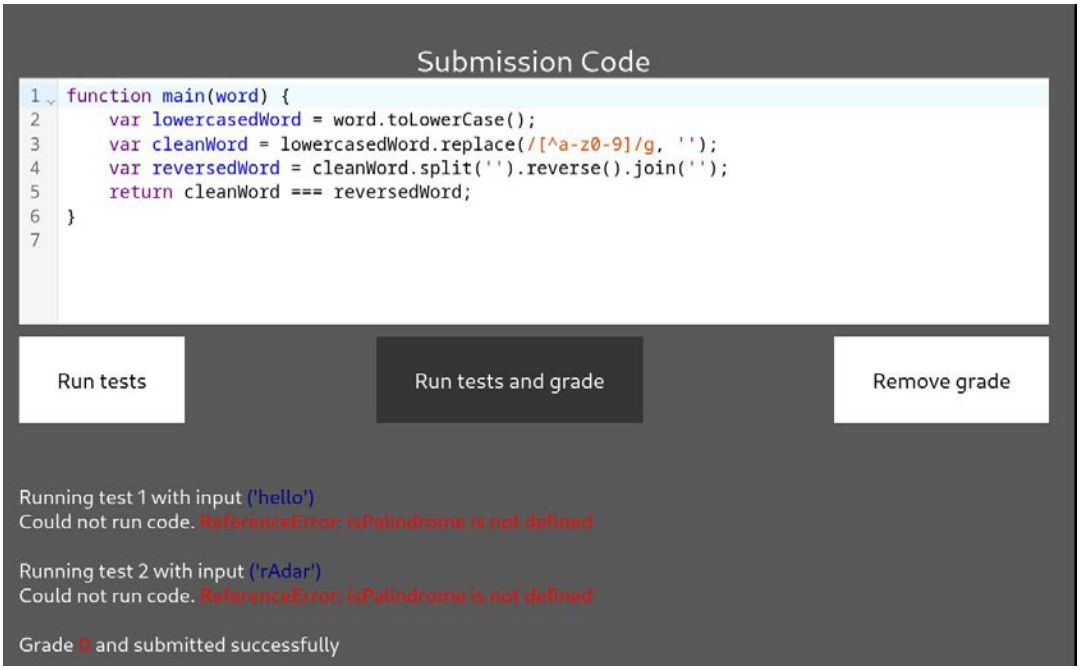
Περνώντας τη συνάρτηση αυτή από τις δύο ενδεχόμενες εισόδους του καθηγητή, τα αποτελέσματα είναι τα αναμενόμενα, άρα η βαθμολόγηση είναι 10, καθώς το κάθε τεστ πιάνει από 5 βαθμούς.

Ο δεύτερος φοιτητής παρέδωσε την παρακάτω συνάρτηση:


Η συνάρτηση αυτή ενώ κάνει ουσιαστικά την ίδια προεργασία με την προηγούμενη, δεν φροντίζει να μετατρέψει τα κεφαλαία γράμματα σε μικρά. Έτσι πέφτει στην παγίδα του καθηγητή που έκανε το πρώτο Α κεφαλαίο στη λέξη radar, κι ενώ η λέξη είναι παλίνδρομη, δε την αναγνωρίζει.

Η εργασία βαθμολογείτε με 5 καθώς πέρασε μόνο από ένα τεστ.

Ο τρίτος φοιτητής παρέδωσε την παρακάτω συνάρτηση:




Ενώ η συνάρτηση ήταν ίδια στον κώδικα με την πρώτη, ο φοιτητής δεν όρισε όπως ζητούσε η εκφώνηση σωστά το όνομα της συνάρτησης. Έτσι ενώ ο αλγόριθμος έψαχνε τη συνάρτηση **isPalindrome**, διάβαζε μόνο **main** και δεν πέρασε κανένα τεστ, επιστρέφοντας 0 στη βαθμολόγηση.

Η δυνατότητα του καθηγητή να πειράξει τον κώδικα πριν ή αφού τον βαθμολογήσει γίνεται χρήσιμη εδώ, καθώς η ξεκάθαρη προβολή των λαθών αποκαλύπτει αν το λάθος ήταν αμελητέο. Θα μπορούσε για παράδειγμα ο φοιτητής να γράψει **isPalindome**, το οποίο είναι ένα ειλικρινές τυπογραφικό λάθος που ο καθηγητής θα κρίνει πως αξίζει να διορθωθεί για μια δίκαιη βαθμολόγηση.

# ΚΕΦΑΛΑΙΟ 6 Επίλογος

## (Υποκεφάλαιο 6.1) Πιθανές Βελτιστοποιήσεις

Το React Grading System παρ όλη την προσοχή που έχει δοθεί στα διάφορα μέρη του, θα μπορούσε να έχει μερικές βελτιστοποιήσεις αν χρησιμοποιούταν σε μεγάλη κλίμακα, τόσο γ την ασφάλεια όσο και για την ευκολία των χρηστών. Θα αναφερθούν μερικές από αυτές ενδεικτικά.

Η σελιδοποίηση των δεδομένων θα βοηθούσε στην ταχύτητα εμφάνισης δεδομένων. Στα παραδείγματα που δόθηκαν, η φόρτωση και το φιλτράρισμα των λιστών γίνονται από το front end της εφαρμογής, στο οποίο φορτώνονται όλα όσα έχει η βάση εκείνη τη στιγμή. Αν και αυτό λειτουργεί με μικρό αριθμό δεδομένων, γρήγορα γίνεται πολύ αργό σε μεγάλους αριθμούς, όπως για παράδειγμα αν υπάρχουν 20 μαθήματα με 5 πρότζεκτ το καθένα. Η εμφάνιση 100 πρότζεκτ τα οποία με ένα κουμπί φιλτράρονται θα γινόταν πολύπλοκη και θα ήταν πιο χρήσιμο να γίνεται από τη βάση, αφού η SQL θεωρείται πολύ πιο γρήγορη σε θέματα διαχείρισης δεδομένων από ότι η JavaScript.

Μια λειτουργία που θα βοηθούσε τους καθηγητές με πιο λεπτομερή βαθμολόγηση θα ήταν να δίνεται βαρύτητα στα τεστ όπως στο JS-test στο κεφάλαιο 3.2, αντί να είναι όλα τα τεστ ισόβαθμα. Αυτό φαίνεται και στο παράδειγμα που δόθηκε στο κεφάλαιο 5.3, όπου τα δύο τεστ αντιστοιχούσαν σε διαφορετικά επίπεδα δυσκολίας και λεπτομέρειας, καθώς η είσοδος ‘hello’ ελέγχει το αν η εργασία έγινε σωστά, η είσοδος ‘rAdar’ ελέγχει την προσοχή που έδωσε ο φοιτητής στις διάφορες περιπτώσεις που μπορεί να προκύψουν.

Μια βελτιστοποίηση ακόμα είναι η αξιοποίηση του ρόλου του διαχειριστή, ο οποίος όπως αναφέρθηκε στο κεφάλαιο 4.2 δεν είναι παρά ένας ακόμα χρήστης με δικαιώματα καθηγητή. Ο διαχειριστής θα μπορούσε να αλλάζει τους ρόλους των χρηστών καθώς και να απενεργοποιεί χρήστες, πράγμα που θα έκανε την εφαρμογή τελείως ανεξάρτητη από το MySQL Workbench κατά τη χρήση της, καθώς αυτή τη στιγμή κάθε νέος χρήστης αντιμετωπίζεται σαν φοιτητής και ο μόνος τρόπος αλλαγής του είναι από το Workbench.

Και τέλος, η ύπαρξη ρυθμίσεων θα βοηθούσε όλους τους χρήστες, ώστε να μπορούν να αλλάζουν τα στοιχεία του λογαριασμού τους.

## (Υποκεφάλαιο 6.2) Συμπεράσματα

Η κατασκευή μιας ιστοσελίδας είναι μια πολύπλοκη διαδικασία με πολλά στάδια τα οποία απαιτούν προσοχή, κυρίως θέματα ασφαλείας ως προς τους χρήστες και τη βάση δεδομένων. Οι τεχνολογίες του Node.js και της React.js προσφέρουν μια μοντέρνα λύση σε θέματα πολυπλοκότητας και λόγω της διασημότητας τους συνεχίζουν να αναπτύσσονται ραγδαία, προσφέροντας στον προγραμματιστή τη δυνατότητα να σκέφτεται περισσότερο τα σοβαρά ζητήματα όπως τη λογική αλγορίθμων και τη αντιμετώπιση κακόβουλων επιθέσεων, παρά τις μικρές τεχνικές λεπτομέρειες. Μπορούν και γεννιούνται έτσι πιο

σύγχρονες, διαδραστικές και ανεπτυγμένες ιστοσελίδες που δίνουν διάφορες λειτουργίες στο χρήστη πέρα από μια παρουσίαση.

Όσο αναλυτικός και προσεγμένος κι αν είναι ένας αλγόριθμος βαθμολόγησης κώδικα, με τα επίπεδα τεχνολογίας που έχουμε σήμερα είναι αδύνατη η απόλυτη πρόβλεψη όλων των περιπτώσεων και ο απόλυτος αυτοματισμός της διαδικασίας. Ό,τι αλγόριθμος και να γραφτεί έχει συγκεκριμένο τρόπο λειτουργίας που αυτοματοποιεί μέρος της διαδικασίας αξιολόγησης, αλλά ποτέ ολόκληρη, καθώς κάθε περίπτωση είναι διαφορετική. Σκοπός λοιπόν τέτοιων αλγορίθμων είναι η πρόβλεψη όσο το δυνατό περισσότερων σεναρίων και η συνεχής ανάπτυξή και προσαρμογή του σε νέες τεχνολογίες. Ένας σωστός αλγόριθμος δεν αποσκοπεί στο να πάρει τη δουλειά ενός βαθμολογητή/καθηγητή, αλλά στο να μειώσει τη δυσκολία και το φόρτο εργασίας του, καθώς και να βοηθήσει έναν μαθητή/φοιτητή να λαμβάνει πιο γρήγορα αποτελέσματα και να βελτιώνεται. Σε οποιαδήποτε περίπτωση, παρόλο που τέτοιες πλατφόρμες επιτρέπουν στους χρήστες να ασχοληθούν με τις εργασίες στο δικό τους χρόνο όσο βρίσκεται μέσα σε μια προθεσμία, είναι σίγουρο πως μειώνουν την διαπροσωπική επαφή μεταξύ καθηγητή και φοιτητή, κάνοντας τα αποτελέσματα πιο απρόσωπα και γυρνώντας τη διαδικασία της μάθησης γύρω από την πλατφόρμα παρά γύρω από ανθρώπους. Είναι σημαντικό η χρήση τέτοιων εργαλείων να μην αντικαθιστά πλήρως την διαδικασία της μάθησης, αλλά να τη συμπληρώνει και να την ενισχύει.

## ΒΙΒΛΙΟΓΡΑΦΙΑ


1. Korolev, M. (2019, March 1). *JavaScript quirks in one image from the internet*. DEV Community. https://dev.to/mkrl/javascript-quirks-in-one-image-from-the-internet-52m7
2. Chapter 4. How JavaScript was created. https://exploringjs.com/es5/ch04.html
3. Wikipedia. TypeScript. https://en.wikipedia.org/wiki/TypeScript
4. Node.js — about Node.js®. https://nodejs.org/en/about
5. Gcuomo. JavaScript Everywhere and the Three Amigos (WebSphere: Into the wild BLUE yonder!). https://web.archive.org/web/20131114212619/https://www.ibm.com/developerworks/community/blogs/gcuomo/entry/javascript_everywhere_and_the_three_amigos?lang=en
6. Wikipedia. Express.js. https://en.wikipedia.org/wiki/Express.js
7. React js archived docs https://web.archive.org/web/20180408084010/https://reactjs.org/
8. Baer, E. *What react is and why it matters*. O'Reilly Online Learning. https://www.oreilly.com/library/view/what-react-is/9781491996744/ch01.html
9. Stack Overflow Developer Survey 2021. Stack Overflow. https://insights.stackoverflow.com/survey/2021#section-most-popular-technologies-web-frameworks
10. MySQL :: MySQL 8.0 Reference Manual :: 1.2.1 What is MySQL? https://dev.mysql.com/doc/refman/8.0/en/what-is-mysql.html
11. MySQL :: MySQL 8.0 Reference Manual :: 1.2.3 History of MySQL. https://dev.mysql.com/doc/refman/8.0/en/history.html
12. DB-Engines ranking. DB-Engines. https://db-engines.com/en/ranking/relational+dbms
13. Wikipedia. MySQL workbench. https://en.wikipedia.org/wiki/MySQL_Workbench
14. Adrienjoly. GitHub – adrienjoly/js-test https://github.com/adrienjoly/js-test
15. Wikipedia. Model–view–viewmodel. https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93viewmodel
16. RFC 7519: JSON Web Token (JWT). IETF Datatracker. https://datatracker.ietf.org/doc/html/rfc7519
17. Fetch API - Web APIs | MDN. (2023, April 1). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/API/Fetch_API

18. Khonde, A. (2021, June 4). Eval is evil - Why we should not use eval in JavaScript. DEV Community. https://dev.to/amitkhonde/eval-is-evil-why-we-should-not-use-eval-in-javascript-1lbh
19. Derma team (Yihan Duan, Lishi Wang Stephen Lau, Omar Qadri). Layered architecture.https://cs.uwaterloo.ca/~m2nagapp/courses/CS446/1195/Arch_Design_Activity/Layered.pdf
20. Wikipedia. Bcrypt. https://en.wikipedia.org/wiki/Bcrypt
21. Wikipedia. Dependency injection. https://en.wikipedia.org/wiki/Dependency_injection
22. *Hacking NodeJS and MongoDB | Websecurify blog*. https://blog.websecurify.com/2014/08/hacking-nodejs-and-mongodb
23. Wikipedia. Database transaction. https://en.wikipedia.org/wiki/Database_transaction
24. *npm: dotenv*. Npm. https://www.npmjs.com/package/dotenv?S_TACT=333AC01A
25. *Function() constructor - JavaScript | MDN*. (2023, September 7). MDN Web Docs. https://developer.mozilla.org/en-US/docs/Web/JavaScript/Reference/Global_Objects/Function/Function