ΠΑΝΕΠΙΣΤΗΜΙΟ ΠΕΛΟΠΟΝΝΗΣΟΥ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΗΛΕΚΤΡΟΛΟΓΩΝ ΜΗΧΑΝΙΚΩΝ

ΚΑΙ ΜΗΧΑΝΙΚΩΝ ΥΠΟΛΟΓΙΣΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

“Σχεδιασμός και ανάπτυξη βοηθητικού web based συστήματος καταχώρησης βαθμολογιών μαθημάτων”

ΜΠΑΛΚΟΝΗΣ ΓΙΩΡΓΟΣ

ΘΟΔΩΡΗΣ ΤΣΙΡΩΝΗΣ

ΕΠΙΝΒΛΕΠΩΝ: ΣΩΤΗΡΙΟΣ ΧΡΙΣΤΟΔΟΥΛΟΥ

ΠΑΤΡΑ 2024

ΠΕΡΙΕΧΟΜΕΝΑ

ΚΕΦΑΛΑΙΟ 1: ΕΙΣΑΓΩΓΗ

1.1 Γενική Εισαγωγή

Στην εποχή της πληροφορίας, η ψηφιακή διαχείριση των πληροφοριών έχει γίνει μέρος της καθημερινότητάς μας και εκπαίδευσή μας σίγουρα όχι εξαίρεση. Μια από τις εργασίες που είναι απαιτητικές και είναι να παίρνουμε περισσότερες πληροφορίες σχετικά με τους μαθητές μας και να τα βάζουμε όλα μαζί, οι βαθμολογίες των θεωρητικών μαθημάτων, οι βαθμολογίες των εργαστηριακών επιδόσεων, η πραγματοποίηση ενός project, αυτά αποτελούν ενα σύνολο για να δημιουργήσουμε μια εικόνα σχετικά με την ακαδημαϊκή επίδοση τους. Αυτή η πτυχιακή εργασία επιλέγει την ανάπτυξη μιας online εφαρμογής: της εισαγωγής και επεξεργασίας των σχολικών αποδόσεων των φοιτητών.

1.2 Περίληψη

Η παρούσα πτυχιακή εργασία επικεντρώνεται στην ανάπτυξη μιας διαδικτυακής εφαρμογής για τη διαχείριση ακαδημαϊκών δεδομένων φοιτητών, συμπεριλαμβανομένων των βαθμολογιών, των παρουσιών και της οργάνωσης εργαστηριακών ομάδων. Στόχος της εφαρμογής είναι η αυτοματοποίηση της διαδικασίας καταχώρησης και ανάλυσης των επιδόσεων των φοιτητών, μειώνοντας τα σφάλματα και βελτιώνοντας τη διαφάνεια. Η εφαρμογή αποτελείται από ένα backend βασισμένο στο Node.js, το οποίο χρησιμοποιεί PostgreSQL για την αποθήκευση δεδομένων και το TypeORM για τη διαχείριση της βάσης. Στο frontend, χρησιμοποιείται Next.js και TailwindCSS, προσφέροντας μια responsive και φιλική προς τον χρήστη εμπειρία. Το σύστημα περιλαμβάνει πέντε βασικές σελίδες: διαχείριση φοιτητών, εργαστηριακές ομάδες, καταγραφή απουσιών, εισαγωγή

βαθμολογιών και ρυθμίσεις υπολογισμού τελικών βαθμών. Ιδιαίτερη έμφαση δόθηκε στη διαχείριση των βαθμολογιών μέσω ενός ευέλικτου μοντέλου, όπου οι καθηγητές μπορούν να καθορίσουν διαφορετικές φόρμουλες υπολογισμού των τελικών βαθμών ανά εργαστηριακή ομάδα. Επιπλέον, η εφαρμογή προσφέρει δυνατότητα μαζικής εισαγωγής δεδομένων μέσω αρχείων CSV, διασφαλίζοντας την εύκολη διαχείριση μεγάλου όγκου δεδομένων. Κατά την ανάπτυξη, προκλήσεις όπως η δυναμική ενημέρωση δεδομένων και η επικύρωση εισόδων επιλύθηκαν μέσω της χρήσης του React Query και της βιβλιοθήκης Zod. Η επιτυχής ολοκλήρωση του έργου επιβεβαιώνει την αποτελεσματικότητα του συστήματος στη βελτίωση της ακαδημαϊκής διαχείρισης και αποτελεί μια ισχυρή βάση για περαιτέρω ανάπτυξη και επέκταση των λειτουργιών του.

1.3 Παρουσίαση του Προβλήματος

Η μέθοδος που ακολουθείται όλον αυτο τον καιρό για την καταγραφή των βαθμών και τον υπολογισμό του τελικού βαθμου ή των εργαστιριακών βαθμων, τη εισαγωγή των απουσιών στα εργαστήρια αλλα και η διαχήρηση των εργαστηριακών ομάδων, παρουσιάζει διάφορες προκλήσεις όπως:

- Η διαχείρηση μεγάλου όγκου δεδομένων είναι χρονοβόρα και μη αποτελεσματική.
- Η έλλειψη κεντρικής αποθήκευσης δεδομένων δυσχεραίνει την πρόσβαση και την ανάληση της πληροφορίας.
- Ο υπολογισμός του τελικού βαθμού μπορει να γίνει λαθανσμένα.

Αυτές είναι μερικές απο τις προκλήσεις που δημιούργησαν την ανάγκη δημιουργίας ενός αυτοματοποιημένου συστήματος που θα βελτιώσει την ακρίβεια και την αποδοτηκότητα της διαδηκασίας αυτής.

1.4 Σημασία και Ανάγκη της Εφαρμογής

Η ανάπτηξη μιας τέτοιας εφαρμογής για την καταχώρηση και τον υπολογισμό των βαθμών προσφέρει πολλά οφέλη:

- **Αποδοτικότητα**: Εξοικονόμηση χρόνου μέσω της αυτοματοποιημένης διαδικασίας υπολογισμού του τελικού βαθμού και καταχώρησης των δεδομένων.
- **Ανάλυση Δεδομένων**: Δυνατότητα γρήγορης ανάλυσης και επεξεργασίας των δεδομένων.
- **Διαφάνεια**: Παροχή εύκολης και γρήγορης πρόσβασης στα δεδομένα των φοιτητών
- **Ακρίβεια**: Ελαχιστοποίηση των λαθών μέσω αυτοματοποιημένων υπολογισμών.

1.5 Δομή της Πτυχιακής Εργασίας

Η παρούσα πτυχιακή εργασία αποτελείται απο έξι κεφάλαια:

- **Κεφάλαιο 1: Εισαγωγή** - Παρουσιάζει το θέμα της εργασίας, το πρόβλημα οπου αντιμετοπίζεται και την σημασία της εφαρμογής.
- **Κεφάλαιο 2: Σκοπός και Στόχοι** - Περιγράφει το σκοπό και τους στόχους της εργασίας, καθώς και τα αναμενόμενα αποτελέσματα.
- **Κεφάλαιο 3: Μεθοδολογίες** - Αναλύει τις μεθοδολογίες που χρησημοποιήθηκαν για την ανάπτυξη της εφαρμογής, συμπεριλαμβανομένων των τεχνολογιών και εργαλείων.
- **Κεφάλαιο 4: Αποτελέσματα** - Παρουσιάζει τα αποτελέσματα της εφαρμογής.
- **Κεφάλαιο 5: Συζήτηση των αποτελεσμάτων** - Συζητά τα αποτελέσματα, τα πλεονεκτήματα και τους περιορισμούς της εφαρμογής.
- **Κεφάλαιο 6: Συμπεράσματα και Προτάσεις Περαιτέρω Ανάπτυξης** - Συμπεραινει την εργασία και προτείνει μελλοντικές βελτιώσεις.

Με την παραπάνω δομή, στόχος της πτυχιακής εργασίας είναι να παρέχει μια ολοκληρομένη και σαφή εικόνα της ανάπτυξης και της λειτουργείας της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2: ΣΚΟΠΟΣ ΚΑΙ ΣΤΟΧΟΙ

2.1 Σκοπός της Εργασίας

Ο σκοπός της παρούσας πτυχιακής εργασίας είναι η ανάπτυξη μιας Single Page Application (SPA) διαδικτυακής εφαρμογής όπου διευκολύνει την καταχώρηση, τον υπολογισμό και την επεξεργασία στα στοιχεία και τους βαθμούς των φοιτητών. Η εφαρμογή στοχεύει να παρέχει ένα ενιαίο και εύχρηστο περιβάλλον για τους καθηγητές, επιτρέποντας την διαχείριση των βαθμών, των εργαστηριακών ομάδων και των παρουσιών των φοιτητών.

2.2 Ειδικοί Στόχοι

Οι ειδικοί στόχοι της εφαρμογής περιλαμβάνουν την ανάπτυξη ενός λειτουργικού πρωτοτύπου που θα επιτρέπει στους καθηγητές να καταχωρούν και να διαχειρίζονται τις βαθμολογίες των φοιτητών τους. Η εφαρμογή θα παρέχει τη δυνατότητα διαχωρισμού των φοιτητών σε διαφορετικές κατηγορίες, όπως αυτές των εργαστηριακών ομάδων, διευκολύνοντας έτσι την οργάνωση των μαθημάτων. Επίσης, θα περιλαμβάνει αυτοματοποιημένες διαδικασίες για τον υπολογισμό των τελικών βαθμών, λαμβάνοντας υπόψη τις βαθμολογίες από οποιοδήποτε είδος βαθμού με το κατάλληλο βάρος όπου μπορεί να ορίσει ο ίδιος ο καθηγητής. Ένας σημαντικός στόχος είναι η ενσωμάτωση λειτουργικότητας για την καταχώρηση των παρουσιών των φοιτητών στα εργαστήρια και η αυτόματη ενημέρωση της καρτέλας κάθε φοιτητή. Ιδιαίτερη έμφαση δίνεται στην αυτόματη διαχείριση των απουσιών, όπου το σύστημα θα ειδοποιεί και θα απορρίπτει αυτόματα τους φοιτητές από τα εργαστήρια σε περίπτωση υπέρβασης του επιτρεπόμενου αριθμού απουσιών. Επιπλέον, η εφαρμογή θα λειτουργεί σε τοπικό επίπεδο, απαιτώντας την εγκατάσταση της βάσης δεδομένων PostgreSQL καθώς και του Node.js για την λειτουργεία του εξυπηρετητή (server).

2.3 Αναμενόμενα Αποτελέσματα

Αναμένεται ότι η ανάπτυξη και η υλοποίηση της εφαρμογής να οδηγήσουν σε σημαντική αύξηση της ακρίβειας στη διαδικασία βαθμολόγησης, μειώνοντας τα λάθη που προκύπτουν από χειροκίνητες διαδικασίες. Επιπλέον, θα επιτευχθεί βελτίωση της αποδοτικότητας, καθώς η εφαρμογή θα επιτρέπει στους καθηγητές να εξοικονομήσουν χρόνο μέσω ενός εύχρηστου και αυτοματοποιημένου συστήματος. Η ενίσχυση της διαφάνειας στη βαθμολογία είναι επίσης ένα αναμενόμενο αποτέλεσμα, με τους καθηγητές να έχουν εύκολη πρόσβαση στα δεδομένα τους. Τέλος, θα διασφαλιστεί η ασφάλεια των δεδομένων μέσω αξιόπιστων τεχνικών αποθήκευσης, με τη δυνατότητα λειτουργίας της εφαρμογής σε τοπικό επίπεδο μέσω της χρήσης της βάσης δεδομένων PostgreSQL.

ΚΕΦΑΛΑΙΟ 3: ΜΕΘΟΔΟΛΟΓΙΕΣ

Στο κεφάλαιο αυτό θα παρουσιαστούν οι μέθοδοι και οι τεχνολογίες που χρησιμοποιήθηκαν για την ανάπτυξη της εφαρμογής, καθώς και η διαδικασία υλοποίησης της. Η εφαρμογή σχεδιάστηκε ως μια Single Page Application (SPA), αξιοποιώντας σύγχρονες τεχνολογίες frontend και backend για να προσφέρει μια ομαλή και αποτελεσματική εμπειρία χρήστη. Η επιλογή των τεχνολογιών έγινε με βάση την ευελιξία, την ασφάλεια και την ευκολία συντήρησης.

3.1 Ανάλυση Απαιτήσεων

Η πρώτη φάση της ανάπτυξης περιλάμβανε την αναλυτική καταγραφή των απαιτήσεων του συστήματος. Οι λειτουργικές απαιτήσεις αφορούσαν τη δυνατότητα καταχώρησης βαθμολογιών, την οργάνωση των φοιτητών σε εργαστηριακές ομάδες, την παρακολούθηση των παρουσιών, καθώς και την αυτοματοποίηση του υπολογισμού των τελικών βαθμών.

3.2 Σχεδιασμός του Συστήματος

Ο σχεδιασμός του συστήματος περιλάμβανε την αρχιτεκτονική του, την οποία αποτελεί ένας κλασικός τριών επιπέδων διαχωρισμός: το frontend, το backend και η βάση δεδομένων. Το

frontend υλοποιήθηκε με χρήση του Next.js 13 σε συνδυασμό με TypeScript και Tailwind CSS για τη διαχείριση του UI. Το backend αναπτύχθηκε με Node.js και TypeScript, ενώ η διασύνδεση με τη βάση δεδομένων έγινε μέσω του ORM TypeORM. Η βάση δεδομένων PostgreSQL επιλέχθηκε για την αποθήκευση και διαχείριση των δεδομένων.

Στον σχεδιασμό της βάσης δεδομένων, δόθηκε έμφαση στην ομαλή οργάνωση των δεδομένων που αφορούν τους φοιτητές, των κατηγοριών τους, τις βαθμολογίες, τις παρουσίες, την διαχώριση των εργαστηριακών ομάδων και την φόρμα υπολογισμού του τελικού βαθμού. Δημιουργήθηκαν οι σχετικοί πίνακες και η διασύνδεσή τους με χρήση κατάλληλων κλειδιών για τη διασφάλιση της ακεραιότητας των δεδομένων.

3.3 Τεχνολογίες και Εργαλεία

Η ανάπτυξη της εφαρμογής βασίστηκε σε μια σειρά από σύγχρονες τεχνολογίες και εργαλεία. Η γλώσσα προγραμματισμού που χρησιμοποιήθηκε για την υλοποίηση της εφαρμογής είναι η TypeScript. Επιλέχθηκε αυτή καθώς προσφέρει αυστηρή ταυτοποίηση του κώδικα, αυξάνοντας την ασφάλεια, την αποφυγή λαθών στην μεταφορά της πληροφορίας στην εφαρμογή καθώς παρέχει τεχνικές ελέγχου (type checkers) αλλά και συντηρησιμότητα. Βασικό ρόλο της υποδομής του συστήματος έχει το Node.js, το οποίο είναι υπεύθυνο για την ένωση και τη λειτουργία των τεχνολογιών. Για τους σκοπούς της ανάπτυξης (development) και τις απαιτήσεις του συστήματος, χρησιμοποιήθηκε ο διαχειριστής πακέτου Bun. Ως βάση δεδομένων αποφασίσαμε να χρησιμοποιήσουμε τη PostgreSQL για λόγους ευελιξίας. Για το frontend επιλέξαμε το Next.js, ένα meta-framework της γνωστής σε όλους React, το οποίο προσφέρει πολλές δυνατότητες για την απλοποίηση του development.

3.3.1 Backend

Το backend της εφαρμογής αποτελεί τη ραχοκοκαλιά της, ενσωματώνοντας όλες τις κρίσιμες λειτουργίες και ελέγχους που απαιτούνται για την ορθή λειτουργία του συστήματος. Στον πυρήνα αυτής της αρχιτεκτονικής βρίσκεται το Node.js, μια πλατφόρμα εκτέλεσης JavaScript που επιτρέπει την ανάπτυξη εξαιρετικά επεκτάσιμων, αποδοτικών και ταυτόχρονων διακομιστών. Το Node.js βασίζεται στο V8, την JavaScript μηχανή της Google Chrome, που εκτελεί JavaScript κώδικα εξαιρετικά γρήγορα. Χάρη στο event-driven, non-blocking I/O μοντέλο του, το Node.js επιτρέπει τη διαχείριση μεγάλου αριθμού ταυτόχρονων συνδέσεων χωρίς να επηρεάζεται η απόδοση, καθιστώντας το ιδανικό για εφαρμογές πραγματικού

χρόνου, όπως διαδικτυακές εφαρμογές που απαιτούν γρήγορη και αποτελεσματική επεξεργασία δεδομένων.

Ένα από τα μεγάλα πλεονεκτήματα του Node.js είναι η χρήση της JavaScript τόσο στο frontend όσο και στο backend, προσφέροντας έναν συνεπή τρόπο προγραμματισμού σε όλη την εφαρμογή. Ωστόσο, για να αξιοποιηθεί πλήρως η ισχύς της TypeScript, η οποία είναι ένα υπερσύνολο της JavaScript και προσθέτει στατική τυποποίηση, είναι απαραίτητη η μετατροπή του TypeScript κώδικα σε JavaScript που μπορεί να εκτελεστεί από το Node.js.

Για τη μεταγλώττιση και συσκευασία του TypeScript κώδικα χρησιμοποιούμε το Bun. Το Bun είναι ένα εξαιρετικά γρήγορο εργαλείο για τη διαχείριση εξαρτήσεων, το bundling και την εκτέλεση TypeScript χωρίς την ανάγκη για εξωτερική μεταγλώττιση. Χρησιμοποιούμε το Bun για τη διαχείριση των εξαρτήσεων κατά τη διάρκεια του development, ενώ για το χτίσιμο (build) της εφαρμογής, το Bun παρέχει ενσωματωμένες δυνατότητες για τη δημιουργία παραγωγικού κώδικα, εξασφαλίζοντας γρήγορη και αποδοτική διαδικασία. Το Bun αναλαμβάνει όλο το φορτίο του bundling και της μεταγλώττισης TypeScript απευθείας στην παραγωγή, χωρίς την ανάγκη πρόσθετων εργαλείων όπως το esbuild ή το webpack. Το Bun είναι ιδιαίτερα γρήγορο και βελτιστοποιημένο για σύγχρονα έργα JavaScript και TypeScript, προσφέροντας ταχύτατη εγκατάσταση και εκτέλεση.

Αν και το Bun χρησιμοποιείται για τη μεταγλώττιση του κώδικα, για τη διαχείριση των εξαρτήσεων και των πακέτων του έργου επιλέχθηκε ο ίδιος Bun, ο οποίος προσφέρει ακόμα πιο γρήγορες επιδόσεις από τα παραδοσιακά εργαλεία όπως το Yarn και το npm. Είναι γραμμένο σε TypeScript και παρέχει ένα απλό API για την εγκατάσταση, το bundling και τη διαχείριση των εξαρτήσεων, με έμφαση στη μειωμένη κατανάλωση πόρων και την αυξημένη απόδοση. Ειδικότερα, το Bun προσφέρει χρόνο εκκίνησης σημαντικά πιο γρήγορο και καλύτερη υποστήριξη για το ESM (ECMAScript Modules), καθιστώντας το ιδανικό για σύγχρονα έργα JavaScript και TypeScript.

Η εφαρμογή χρησιμοποιεί επίσης την PostgreSQL, μια ισχυρή και αξιόπιστη σχεσιακή βάση δεδομένων ανοιχτού κώδικα, για την αποθήκευση και διαχείριση των δεδομένων. Η PostgreSQL είναι γνωστή για την επεκτασιμότητα και την αξιοπιστία της, καθώς και για τις προηγμένες δυνατότητές της, όπως υποστήριξη για περίπλοκα ερωτήματα και διαχείριση JSON δεδομένων. Στην εφαρμογή αυτή, η PostgreSQL επιτρέπει την αποθήκευση και ανάκτηση δεδομένων με αποδοτικό τρόπο, εξασφαλίζοντας παράλληλα την ακεραιότητα των δεδομένων.

Η διασύνδεση του Node.js με την PostgreSQL επιτυγχάνεται μέσω του TypeORM, ενός Object Relational Mapper (ORM) που επιτρέπει τη χαρτογράφηση των πινάκων της βάσης

δεδομένων σε κλάσεις TypeScript. Το TypeORM διευκολύνει τη διαχείριση των δεδομένων μέσω αντικειμενοστραφών προσεγγίσεων, επιτρέποντας την αλληλεπίδραση με τη βάση δεδομένων χωρίς την ανάγκη για γραφή ακατέργαστων SQL ερωτημάτων. Μέσω του TypeORM, το σύστημα μπορεί να ελέγχει την ύπαρξη των πινάκων στη βάση δεδομένων και, αν χρειάζεται, να τους δημιουργεί αυτόματα, διασφαλίζοντας τη συνοχή και την ακεραιότητα της βάσης δεδομένων.

Η ασφάλεια και η ακεραιότητα των δεδομένων ενισχύονται περαιτέρω με τη χρήση του Zod, μιας βιβλιοθήκης δήλωσης σχημάτων τύπου για TypeScript, η οποία προσφέρει αυστηρούς μηχανισμούς επικύρωσης των δεδομένων. Το Zod επιτρέπει την αυστηρή τυποποίηση των δεδομένων, διασφαλίζοντας ότι μόνο τα δεδομένα που πληρούν τις καθορισμένες προδιαγραφές εισάγονται στο σύστημα, μειώνοντας έτσι τον κίνδυνο σφαλμάτων και βελτιώνοντας την ασφάλεια των δεδομένων κατά τη μεταφορά τους.

Η επικοινωνία μεταξύ του frontend και του backend επιτυγχάνεται μέσω RESTful APIs (Representational State Transfer Application Programming Interfaces). Τα REST APIs προσφέρουν έναν αποδοτικό τρόπο για την ανταλλαγή δεδομένων μέσω HTTP, επιτρέποντας στο frontend να αποστέλλει αιτήματα στο backend και να λαμβάνει απαντήσεις με επεξεργασμένα δεδομένα. Αυτή η προσέγγιση επιτρέπει την ανάπτυξη κλιμακούμενων και ευέλικτων υπηρεσιών που ανταποκρίνονται στις ανάγκες των χρηστών.

Συνολικά, η χρήση του Node.js σε συνδυασμό με TypeScript, Bun, PostgreSQL, TypeORM, Zod και REST APIs προσφέρει μια ισχυρή και ευέλικτη υποδομή για την ανάπτυξη της εφαρμογής. Το Node.js και το TypeScript παρέχουν μια σταθερή βάση για τη διαχείριση των αιτημάτων και την ασφάλεια των δεδομένων, ενώ το Bun εξασφαλίζει την ταχεία και αποδοτική μεταγλώττιση του κώδικα και τη διαχείριση των εξαρτήσεων. Η PostgreSQL και το TypeORM επιτρέπουν την ασφαλή και αποτελεσματική διαχείριση της βάσης δεδομένων. Το Zod διασφαλίζει την ακρίβεια και την ασφάλεια των δεδομένων, και τα REST APIs επιτρέπουν την απρόσκοπτη επικοινωνία μεταξύ του frontend και του backend, δημιουργώντας μια ολοκληρωμένη και αξιόπιστη λύση.

3.3.2 Frontend

Το frontend της εφαρμογής είναι εξίσου κρίσιμο για την ομαλή και αποτελεσματική λειτουργία της, καθώς αποτελεί το τμήμα που αλληλεπιδρά άμεσα με τους χρήστες, παρέχοντας τους μια φιλική και αποδοτική εμπειρία χρήσης. Για την ανάπτυξη του frontend,

χρησιμοποιήθηκε το Next.js 13, ένα ισχυρό framework που βασίζεται στη React και παρέχει προηγμένες δυνατότητες για τη δημιουργία Single Page Applications (SPAs) και δυναμικών ιστοσελίδων. Το Next.js προσφέρει πολλά πλεονεκτήματα, όπως την αυτόματη βελτιστοποίηση της απόδοσης, τη δυνατότητα server-side rendering (SSR) και static site generation (SSG), καθώς και μια εύκολη στη χρήση αρχιτεκτονική που διευκολύνει την ανάπτυξη σύγχρονων και κλιμακούμενων εφαρμογών.

Με το Next.js 13, η εφαρμογή επωφελείται από την πιο πρόσφατη τεχνολογία στη React ανάπτυξη, επιτρέποντας τη δημιουργία δυναμικών και γρήγορων ιστοσελίδων που ανταποκρίνονται άμεσα στις ενέργειες των χρηστών. Η υποστήριξη για server-side rendering βελτιώνει την απόδοση και την SEO (Search Engine Optimization) των σελίδων, καθιστώντας την εφαρμογή πιο προσβάσιμη και αποδοτική. Επιπλέον, το Next.js προσφέρει δυνατότητες για εύκολη διαχείριση των ρυθμίσεων διαδρομών (routing) και δυναμικών σελίδων, επιτρέποντας την ανάπτυξη πολύπλοκων δομών με ευκολία και αποδοτικότητα.

Για το σχεδιασμό του frontend, χρησιμοποιήθηκε το Tailwind CSS, ένα utility-first CSS framework που διευκολύνει τη δημιουργία ευέλικτων και επαναχρησιμοποιήσιμων στιλ για την εφαρμογή. Το Tailwind CSS διαφέρει από τα παραδοσιακά CSS frameworks, καθώς αντί να παρέχει προκαθορισμένα στοιχεία, επιτρέπει στους προγραμματιστές να χτίζουν εξατομικευμένα σχέδια απευθείας από τη βάση, χρησιμοποιώντας utility classes. Αυτό καθιστά το Tailwind CSS ιδιαίτερα αποδοτικό για τη δημιουργία προσαρμοσμένων σχεδίων, καθώς οι προγραμματιστές μπορούν να γράφουν λιγότερο κώδικα και να αποφεύγουν την υπερβολική χρήση γραφής CSS. Το αποτέλεσμα είναι ένα καθαρό και συντηρήσιμο UI, το οποίο προσαρμόζεται εύκολα στις απαιτήσεις της εφαρμογής.

Η χρήση του Tailwind CSS επιτρέπει επίσης ταχύτερη ανάπτυξη, καθώς οι προγραμματιστές μπορούν να δημιουργούν responsive layouts και να προσαρμόζουν το στυλ των στοιχείων της σελίδας άμεσα, χωρίς να χρειάζεται να γράψουν παραδοσιακό CSS κώδικα από την αρχή. Το Tailwind CSS υποστηρίζει επίσης την εύκολη συντήρηση του κώδικα, καθώς η εφαρμογή μπορεί να προσαρμοστεί γρήγορα σε νέες ανάγκες και αλλαγές, χωρίς να επηρεάζεται η συνολική συνέπεια του σχεδιασμού.

Η TypeScript χρησιμοποιήθηκε επίσης στο frontend, παρέχοντας στατική τυποποίηση και βελτιωμένη ασφάλεια κώδικα, μειώνοντας τα σφάλματα και βελτιώνοντας τη συντηρησιμότητα. Η TypeScript ενσωματώθηκε αρμονικά με το Next.js και το Tailwind CSS, διασφαλίζοντας ότι ο κώδικας παραμένει σαφής, δομημένος και εύκολα επεκτάσιμος.

Η συνδυασμένη χρήση του Next.js 13, του Tailwind CSS και της TypeScript στο frontend της εφαρμογής παρέχει τα απαραίτητα εργαλεία για τη δημιουργία σύγχρονων, γρήγορων και εύκολα επεκτάσιμων διεπαφών χρήστη. Η ευελιξία και η δύναμη αυτών των τεχνολογιών

συνέβαλαν καθοριστικά στην υλοποίηση ενός φιλικού προς τον χρήστη περιβάλλοντος, το οποίο μπορεί να προσαρμοστεί και να εξελιχθεί ανάλογα με τις ανάγκες της εφαρμογής.

3.3.3 Testing

Η διαδικασία δοκιμών (testing) αποτελεί αναπόσπαστο μέρος της ανάπτυξης της εφαρμογής, εξασφαλίζοντας ότι όλες οι λειτουργίες λειτουργούν όπως αναμένεται και ότι το σύστημα είναι απαλλαγμένο από σφάλματα. Για τον σκοπό αυτό, χρησιμοποιήθηκε το Postman, ένα δημοφιλές εργαλείο για τη δοκιμή API (Application Programming Interface). Το Postman παρέχει ένα ισχυρό περιβάλλον για τη δημιουργία, την εκτέλεση και την αυτοματοποίηση δοκιμών HTTP αιτημάτων, επιτρέποντας στους προγραμματιστές να διασφαλίσουν ότι τα RESTful APIs της εφαρμογής ανταποκρίνονται σωστά και επιστρέφουν τα αναμενόμενα δεδομένα.

Με το Postman, ήταν δυνατό να δοκιμαστούν όλες οι βασικές λειτουργίες του backend, όπως η καταχώρηση, η ενημέρωση, η ανάκτηση και η διαγραφή δεδομένων. Το εργαλείο αυτό επέτρεψε την εύκολη δημιουργία σεναρίων δοκιμών και την επαλήθευση της απόδοσης και της ασφάλειας των API. Επιπλέον, το Postman υποστηρίζει την αυτοματοποίηση των δοκιμών, γεγονός που διευκολύνει την επαναλαμβανόμενη εκτέλεση δοκιμών κατά τη διάρκεια της ανάπτυξης, εξασφαλίζοντας ότι οποιεσδήποτε αλλαγές στον κώδικα δεν επηρεάζουν αρνητικά τη λειτουργικότητα της εφαρμογής.

Η χρήση του Postman επέτρεψε την ταχεία ανίχνευση και επίλυση σφαλμάτων, διασφαλίζοντας ότι η εφαρμογή παραμένει σταθερή και αξιόπιστη καθ' όλη τη διάρκεια της ανάπτυξης. Επιπλέον, η δυνατότητα τεκμηρίωσης των API μέσω του Postman διευκόλυνε την επικοινωνία και τη συνεργασία μεταξύ των μελών της ομάδας, καθώς όλα τα μέρη της εφαρμογής μπορούσαν να επικυρώσουν και να κατανοήσουν τις αλληλεπιδράσεις του συστήματος με τα δεδομένα.

Η υιοθέτηση του Postman ως εργαλείο δοκιμών αποδείχθηκε ιδιαίτερα χρήσιμη για την ανάπτυξη της εφαρμογής. Μέσω της εκτεταμένης χρήσης του, καταφέραμε να διασφαλίσουμε ότι το backend λειτουργεί αξιόπιστα και αποδοτικά, αντιμετωπίζοντας αποτελεσματικά τυχόν σφάλματα και αστοχίες. Το Postman παρείχε ένα πλαίσιο για την αυτοματοποίηση των δοκιμών και τη διασφάλιση της ομαλής λειτουργίας των API, κάτι που τελικά συνέβαλε καθοριστικά στη σταθερότητα και την ποιότητα του συστήματος που αναπτύχθηκε.

3.4. Ανάπτυξη και Υλοποίηση

3.4.1. Backend Ανάπτυξη

Η ανάπτυξη του backend της εφαρμογής ξεκίνησε με τη ρύθμιση του Node.js server, ο οποίος ελέγχει την ύπαρξη της βάσης δεδομένων και των σχετικών πινάκων κατά την εκκίνηση. Εάν η βάση δεδομένων δεν υπάρχει ή λείπουν πίνακες, ο server είναι ρυθμισμένος να τους δημιουργεί αυτόματα. Αυτό το βήμα εξασφαλίζει ότι το σύστημα είναι πάντα έτοιμο για χρήση χωρίς την ανάγκη χειροκίνητης παρέμβασης.

Μετά την επιτυχημένη ρύθμιση του server και της βάσης δεδομένων, η ανάπτυξη προχώρησε με τη δημιουργία των βασικών πινάκων χρησιμοποιώντας το TypeORM. Οι κύριοι πίνακες που δημιουργήθηκαν είναι οι εξής:

1. Μαθητής (Student):

Ο πίνακας Μαθητής αποτελεί τον πυρήνα της αποθήκευσης δεδομένων για τους μαθητές. Σε αυτόν τον πίνακα αποθηκεύονται βασικές πληροφορίες για κάθε μαθητή, όπως το όνομα, το επώνυμο, το ακαδημαϊκό έτος, και η ομάδα εργαστηρίου στην οποία ανήκει. Επίσης, ο πίνακας περιλαμβάνει πληροφορίες σχετικά με την πρόοδο του μαθητή, όπως η κατάστασή του στο εργαστήριο και στη θεωρία, καθώς και οι τελικοί βαθμοί που έχει λάβει.

Ο πίνακας Student συνδέεται άμεσα με τους πίνακες Grade και LabAttendance μέσω σχέσεων OneToMany. Αυτό επιτρέπει την ευέλικτη διαχείριση των δεδομένων που αφορούν τις βαθμολογίες και τις παρουσίες των μαθητών στα εργαστήρια.

```
@Entity()
export class Student {
  @PrimaryColumn({ type: "integer" })
  studentId: number;

  @Column({ type: "varchar" })
  name: string;

  @Column({ type: "varchar" })
  surname: string;
```

```
  @Column({ type: "integer" })
  academicYear: number;

  @Column({ type: "varchar", default: "no-group" })
  labGroup: string;

  @Column({ type: "varchar", default: "no-type" })
  studentType: string;

  @Column({ type: "varchar", default: server.student.labStatus.Enum.pending })
  labStatus: server.student.LabStatus;

  @Column({ type: "varchar", default: server.student.theoryStatus.Enum.pending })
  theoryStatus: server.student.TheoryStatus;

  @Column({ type: "varchar", default: "N/A" })
  finalGrade: string;

  @Column({ type: "varchar", default: "N/A" })
  labGrade: string;

  @OneToMany(() => Grade, (grade) => grade.student)
  grades: Grade[];

  @OneToMany(() => LabAttendance, (labAttendance) => labAttendance.student)
  labAttendances: LabAttendance[];
}
```

2. Βαθμός (Grade)

Ο πίνακας Grade χρησιμοποιείται για την αποθήκευση των επιμέρους βαθμολογιών που λαμβάνει κάθε μαθητής σε διαφορετικά μαθήματα ή δραστηριότητες. Κάθε εγγραφή σε αυτόν τον πίνακα περιλαμβάνει τον τύπο του βαθμού (π.χ., θεωρία, εργασίες), τη βαθμολογία που έλαβε ο μαθητής, και την ημερομηνία που καταγράφηκε η βαθμολογία.

Η σύνδεση με τον πίνακα Student επιτρέπει την αποθήκευση πολλαπλών βαθμολογιών για κάθε μαθητή, προσφέροντας έτσι ένα πλήρες ιστορικό της ακαδημαϊκής του πορείας. Αυτή η δομή είναι κρίσιμη για την ορθή λειτουργία του συστήματος υπολογισμού τελικών βαθμών, καθώς επιτρέπει την εύκολη πρόσβαση σε όλες τις σχετικές βαθμολογίες κατά τη διάρκεια του υπολογισμού.

```
@Entity()
export class Grade {
  @PrimaryGeneratedColumn({ type: "integer" })
  id: number;

  @Column({ type: "varchar" })
  gradeType: string;

  @Column("decimal", { precision: 3, scale: 2 })
  score: number;

  @Column({ type: "varchar" })
  date: string;

  @ManyToOne(() => Student, (student) => student.grades)
  @JoinColumn({ name: "studentId" })
  student: Student;
}
```

3. Παροθσίες Εργαστηρίου (LabAttendance):

Ο πίνακας LabAttendance διαχειρίζεται την καταγραφή των παρουσιών των μαθητών στα εργαστήρια. Για κάθε παρουσία, αποθηκεύεται αν ο μαθητής παρευρέθηκε ή όχι, καθώς και η ημερομηνία της παρουσίας. Αυτή η πληροφορία είναι σημαντική για την παρακολούθηση της συμμετοχής των μαθητών στα εργαστήρια και για την αυτόματη ενημέρωση της κατάστασής τους (π.χ., αν έχουν αποτύχει λόγω υπέρβασης του επιτρεπόμενου αριθμού απουσιών).

Ο πίνακας αυτός συνδέεται επίσης με τον πίνακα Student, επιτρέποντας τη συσχέτιση των παρουσιών με τους αντίστοιχους μαθητές. Αυτή η συσχέτιση βοηθά στον αυτοματοποιημένο έλεγχο των παρουσιών και στη διασφάλιση ότι οι μαθητές που υπερβαίνουν τα επιτρεπτά όρια απουσιών μαρκάρονται σωστά.

```
@Entity()
export class LabAttendance {
  @PrimaryGeneratedColumn({ type: "integer" })
  id: number;
```

```
  @Column({ type: "boolean", nullable: true })
  attended: boolean;

  @Column({ type: "varchar" })
  date: string;

  @ManyToOne(() => Student, (student) => student.labAttendances)
  @JoinColumn({ name: "studentId" })
  student: Student;
}
```

4. Υπολογισμός Βαθμου (GradeFormula):

Ο πίνακας GradeFormula είναι ο πιο κρίσιμος για τον υπολογισμό του τελικού βαθμού των μαθητών. Περιέχει τις φόρμουλες που χρησιμοποιούνται για τον υπολογισμό του τελικού βαθμού, οι οποίες καθορίζονται από τον τύπο του μαθητή και το όνομα της φόρμουλας. Το πεδίο της φόρμουλας αποθηκεύεται ως JSON και περιλαμβάνει τα βάρη που πρέπει να δοθούν σε κάθε είδος βαθμολογίας.

Αυτή η ευέλικτη προσέγγιση επιτρέπει στους καθηγητές να ορίσουν διαφορετικές φόρμουλες υπολογισμού για διαφορετικούς τύπους μαθητών, ανάλογα με τις ανάγκες τους. Κατά τη διάρκεια του υπολογισμού του τελικού βαθμού, η συνάρτηση υπολογισμού αναζητά την κατάλληλη φόρμουλα στον πίνακα GradeFormula και εφαρμόζει τα κατάλληλα βάρη στις βαθμολογίες του μαθητή, οδηγώντας στον υπολογισμό του τελικού βαθμού.

Η λειτουργία του πίνακα GradeFormula σε συνδυασμό με τους πίνακες Student, Grade, και LabAttendance, επιτρέπει ένα ολοκληρωμένο και ακριβές σύστημα διαχείρισης και αξιολόγησης των μαθητών. Η σχεδίαση αυτή εξασφαλίζει ότι οι μαθητές αξιολογούνται δίκαια και με βάση τις προκαθορισμένες ακαδημαϊκές προδιαγραφές.

```
@Entity()
export class GradeFormula {
  @PrimaryGeneratedColumn()
  id: number;

  @Column({ type: "varchar" })
  studentType: string;
```

```
  @Column({ type: "varchar" })
  formulaName: string;

  @Column({ type: "json" })
  formula: Record<string, number>;
}
```

Μετά την επιτυχή δημιουργία των πινάκων, ανέπτυξα τις βασικές CRUD λειτουργίες για κάθε πίνακα μέσω REST APIs. Τα APIs αυτά δοκιμάστηκαν και επαληθεύτηκαν μέσω του Postman, για να διασφαλιστεί η σωστή λειτουργία τους.

Ένα από τα μεγαλύτερα challenges κατά την ανάπτυξη του backend ήταν η υλοποίηση της φόρμουλας υπολογισμού του τελικού βαθμού, που τελικά επιλύθηκε με τη χρήση του πίνακα GradeFormula. Αυτή η προσέγγιση επιτρέπει ευελιξία στον τρόπο υπολογισμού των βαθμών, ανάλογα με τον τύπο μαθητή και τις απαιτήσεις του καθηγητή.

Τέλος, το σύστημα εξασφαλίζει την ακρίβεια και την ασφάλεια των δεδομένων μέσω της χρήσης της βιβλιοθήκης ZOD για type checking, διασφαλίζοντας ότι τα δεδομένα που εισάγονται στο σύστημα είναι έγκυρα και συμβατά με τις καθορισμένες προδιαγραφές.

3.4.2. Frontend Ανάπτυξη

Η ανάπτυξη του frontend της εφαρμογής πραγματοποιήθηκε με τη χρήση του Next.js και της TailwindCSS, προσφέροντας ένα γρήγορο, ευέλικτο και responsive περιβάλλον χρήστη. Η εφαρμογή περιλαμβάνει πέντε κύριες σελίδες που επιτρέπουν τη διαχείριση των φοιτητών, των εργαστηριακών ομάδων, των απουσιών, των βαθμολογιών και των ρυθμίσεων των φόρμουλων υπολογισμού των τελικών βαθμών.

Κατά την ανάπτυξη του frontend, δόθηκε ιδιαίτερη έμφαση στη βελτιστοποίηση της εμπειρίας χρήστη μέσω ενός σαφούς και λειτουργικού UI. Τα δεδομένα λαμβάνονται και αποστέλλονται στο backend μέσω API requests, διασφαλίζοντας την άμεση και ομαλή ενημέρωση των πληροφοριών σε πραγματικό χρόνο.

Δομή της Εφαρμογής:

Διαχείριση Φοιτητών

Η αρχική σελίδα της εφαρμογής εμφανίζει όλους τους φοιτητές με τις αντίστοιχες πληροφορίες τους. Οι χρήστες έχουν τη δυνατότητα να προσθέσουν νέους φοιτητές είτε χειροκίνητα είτε μέσω μαζικής εισαγωγής από CSV αρχείο. Επιπλέον, παρέχεται η επιλογή επεξεργασίας ή διαγραφής υφιστάμενων φοιτητών, διευκολύνοντας τη διαχείριση των δεδομένων.

Διαχείριση Εργαστηριακών Ομάδων

Στη δεύτερη σελίδα, οι φοιτητές μπορούν να ταξινομηθούν σε εργαστηριακές ομάδες. Οι ομάδες αυτές δημιουργούνται δυναμικά, επιτρέποντας ευελιξία στην κατανομή των φοιτητών. Μέσω ενός φιλικού UI, οι χρήστες μπορούν να μετακινούν φοιτητές μεταξύ των ομάδων, διασφαλίζοντας την ορθολογική κατανομή τους.

Παρακολούθηση Απουσιών

Η τρίτη σελίδα επιτρέπει την καταγραφή των απουσιών των φοιτητών. Για κάθε εργαστηριακή συνεδρία, οι χρήστες μπορούν να μαρκάρουν αν ένας φοιτητής ήταν παρών ή απών. Αυτή η λειτουργία βοηθά στην παρακολούθηση της συμμετοχής των φοιτητών και επιτρέπει τον αυτόματο υπολογισμό της κατάστασής τους βάσει των κανόνων απουσιών.

Καταχώρηση Βαθμολογιών

Στην τέταρτη σελίδα, οι καθηγητές μπορούν να εισάγουν βαθμολογίες για τις εργασίες, τις εξετάσεις και άλλες ακαδημαϊκές δραστηριότητες. Το UI είναι σχεδιασμένο έτσι ώστε να επιτρέπει γρήγορη και εύκολη εισαγωγή δεδομένων, ενώ οι βαθμολογίες αποθηκεύονται και συγχρονίζονται άμεσα με το backend.

Ρυθμίσεις Βαθμολογικών Φόρμουλων

Στην πέμπτη σελίδα, οι χρήστες μπορούν να ορίσουν το ποσοστό που αντιστοιχεί σε κάθε δραστηριότητα για τον υπολογισμό του τελικού βαθμού. Η εφαρμογή προσφέρει δυναμική διαχείριση των βαρών των εργασιών ανά εργαστηριακή ομάδα, επιτρέποντας έτσι προσαρμογή των κριτηρίων αξιολόγησης.

3.5. Διαχείριση Έργου

Η διαχείριση του έργου για την ανάπτυξη της εφαρμογής βασίστηκε σε δύο κύρια εργαλεία: το GitHub και το Postman. Αυτά τα εργαλεία διασφαλίζουν την οργανωμένη και αποδοτική ανάπτυξη του λογισμικού, επιτρέποντας την εύκολη συνεργασία μεταξύ των μελών της ομάδας και τη διασφάλιση της ποιότητας του κώδικα.

3.5.1 Χρήση GitHub

Το GitHub αποτέλεσε το κεντρικό αποθετήριο για τον κώδικα της εφαρμογής. Η επιλογή του GitHub προσφέρει μια σειρά από πλεονεκτήματα που το καθιστούν ιδανικό για τη διαχείριση του έργου. Αρχικά, το GitHub επιτρέπει την εύκολη παρακολούθηση των αλλαγών στον κώδικα μέσω του συστήματος ελέγχου εκδόσεων Git. Κάθε αλλαγή που γίνεται στον κώδικα καταγράφεται σε μια νέα έκδοση, επιτρέποντας έτσι την επιστροφή σε προηγούμενες εκδόσεις αν χρειαστεί. Αυτό προσφέρει ασφάλεια και δυνατότητα διόρθωσης λαθών χωρίς να διακινδυνεύει η σταθερότητα του έργου.

Επιπλέον, το GitHub διευκολύνει τη συνεργασία μεταξύ των μελών της ομάδας. Μέσω του συστήματος pull requests, τα μέλη της ομάδας μπορούν να προτείνουν αλλαγές στον κώδικα, οι οποίες στη συνέχεια αναθεωρούνται και εγκρίνονται πριν ενσωματωθούν στο κύριο αποθετήριο. Αυτή η διαδικασία διασφαλίζει ότι ο κώδικας που προστίθεται είναι σωστά δοκιμασμένος και πληροί τα πρότυπα ποιότητας της ομάδας.

Το GitHub χρησιμοποιήθηκε επίσης για την οργάνωση των εργασιών και των εκδόσεων του έργου. Μέσω της χρήσης εργαλείων όπως τα Issues και τα Milestones, η ομάδα μπορούσε να διαχειρίζεται τις διάφορες εργασίες που έπρεπε να ολοκληρωθούν, να παρακολουθεί την πρόοδο του έργου, και να διασφαλίζει ότι το έργο προχωρά σύμφωνα με το χρονοδιάγραμμα.

3.5.2 Χρήση Postman

Το Postman αποτέλεσε το βασικό εργαλείο για τη δοκιμή και την τεκμηρίωση των API που αναπτύχθηκαν στο backend της εφαρμογής. Για να υποστηριχθεί η ανάπτυξη του frontend, δημιουργήθηκε μια πλήρης σουίτα δοκιμών στο Postman που περιλαμβάνει όλες τις βασικές λειτουργίες CRUD (Create, Read, Update, Delete).

Αυτή η σουίτα δοκιμών προσφέρει πολλαπλά οφέλη. Αφενός, παρέχει παραδείγματα για το πώς μπορούν να γίνουν οι κλήσεις στο API, κάτι που είναι ιδιαίτερα χρήσιμο για τους προγραμματιστές που εργάζονται στο frontend. Μέσω των παραδειγμάτων αυτών, οι προγραμματιστές μπορούν να δουν πώς λειτουργεί το API στην πράξη και να προσαρμόσουν τον κώδικά τους ανάλογα. Αφετέρου, η σουίτα δοκιμών επιτρέπει τη διενέργεια αυτοματοποιημένων δοκιμών, διασφαλίζοντας ότι οι βασικές λειτουργίες του API λειτουργούν σωστά ακόμα και μετά από αλλαγές στον κώδικα.

Η χρήση του Postman για τη δημιουργία και τη διαχείριση αυτής της σουίτας δοκιμών εξασφαλίζει ότι το API παραμένει συνεπές και αξιόπιστο καθ' όλη τη διάρκεια της ανάπτυξης του έργου. Επιπλέον, η τεκμηρίωση που παρέχεται από το Postman είναι άμεσα διαθέσιμη και εύκολα προσβάσιμη, διευκολύνοντας έτσι τη συνεργασία μεταξύ των μελών της ομάδας και τη συντήρηση του κώδικα στο μέλλον.

ΚΕΦΑΛΑΙΟ 4: ΑΠΟΤΕΛΕΣΜΑΤΑ

Στο κεφάλαιο αυτό παρουσιάζονται τα αποτελέσματα που προέκυψαν από την ανάπτυξη και εφαρμογή του συστήματος διαχείρισης βαθμολογιών και παρακολούθησης φοιτητών. Οι αξιολογήσεις και οι δοκιμές που πραγματοποιήθηκαν αποσκοπούν στη διασφάλιση της λειτουργικότητας και της απόδοσης του συστήματος, καθώς και στην εξασφάλιση της αξιοπιστίας και της ορθότητας των δεδομένων που διαχειρίζεται.

4.1 Δοκιμές και Αποτελέσματα

Κατά τη διαδικασία ανάπτυξης της εφαρμογής, δόθηκε ιδιαίτερη έμφαση στη δοκιμή των λειτουργιών και των δυνατοτήτων του συστήματος, ώστε να διασφαλιστεί ότι όλες οι προδιαγραφές και οι απαιτήσεις πληρούνται πλήρως. Οι δοκιμές που πραγματοποιήθηκαν είχαν ως στόχο να αξιολογήσουν την ακρίβεια των βασικών λειτουργιών, όπως η καταχώρηση και επεξεργασία βαθμολογιών, η παρακολούθηση παρουσιών στα εργαστήρια και ο υπολογισμός των τελικών βαθμών των φοιτητών.

Ένα σημαντικό μέρος των δοκιμών επικεντρώθηκε στη διασφάλιση της ορθότητας των δεδομένων. Χρησιμοποιώντας το Postman, δημιουργήθηκε μια πλήρης σουίτα δοκιμών για τις βασικές λειτουργίες CRUD (Create, Read, Update, Delete) των APIs του συστήματος. Αυτή η σουίτα περιλάμβανε σενάρια που καλύπτουν όλους τους πιθανούς τρόπους αλληλεπίδρασης με το σύστημα, επιτρέποντας έτσι την ανακάλυψη και την επιδιόρθωση τυχόν σφαλμάτων που θα μπορούσαν να προκύψουν κατά την καταχώρηση ή την επεξεργασία δεδομένων.

Επιπλέον, οι δοκιμές αυτές βοήθησαν στη διασφάλιση της ασφάλειας και της αξιοπιστίας του συστήματος. Με την προσομοίωση διαφόρων σεναρίων χρήσης και με την εκτέλεση εντατικών δοκιμών, το σύστημα αξιολογήθηκε για την ικανότητά του να διαχειρίζεται τα

δεδομένα με ασφάλεια, αποτρέποντας την ανεπίτρεπτη πρόσβαση και εξασφαλίζοντας την ακεραιότητα των πληροφοριών.

4.1.2 Αποτελέσματα

Τα αποτελέσματα των δοκιμών έδειξαν ότι το σύστημα λειτουργεί όπως αναμενόταν, με όλες τις βασικές λειτουργίες να εκτελούνται σωστά και χωρίς σφάλματα. Οι βασικές λειτουργίες καταχώρησης, ανάκτησης, ενημέρωσης και διαγραφής δεδομένων εκτελέστηκαν με επιτυχία σε όλες τις περιπτώσεις δοκιμής. Επίσης, το σύστημα απέδειξε την ικανότητά του να υπολογίζει τους τελικούς βαθμούς των φοιτητών με ακρίβεια, χρησιμοποιώντας τις φόρμουλες που ορίζονται για κάθε τύπο φοιτητή.

Οι δοκιμές απόδοσης ανέδειξαν ότι το σύστημα ανταποκρίνεται με ταχύτητα και ακρίβεια στις απαιτήσεις των χρηστών. Οι χρόνοι απόκρισης ήταν ικανοποιητικοί, ακόμα και όταν το σύστημα υποβλήθηκε σε αυξημένο φόρτο εργασίας. Αυτό αποδεικνύει την ικανότητα του συστήματος να διαχειρίζεται πολλαπλές αιτήσεις ταυτόχρονα, χωρίς να επηρεάζεται η απόδοσή του.

4.2 Ανάλυση Απόδοσης

Η ανάλυση της απόδοσης του συστήματος βασίστηκε στα αποτελέσματα των δοκιμών και στην αξιολόγηση της συμπεριφοράς του υπό διαφορετικές συνθήκες λειτουργίας. Οι μετρήσεις που πραγματοποιήθηκαν αφορούσαν την ταχύτητα απόκρισης του συστήματος, τη χρήση πόρων και την ικανότητά του να διαχειρίζεται μεγάλο όγκο δεδομένων και ταυτόχρονες αιτήσεις.

Το σύστημα έδειξε να ανταποκρίνεται αποτελεσματικά στις απαιτήσεις των χρηστών, με τους χρόνους απόκρισης να παραμένουν εντός των προκαθορισμένων ορίων. Η χρήση των διαθέσιμων πόρων, όπως η μνήμη και η επεξεργαστική ισχύς, ήταν αποδοτική, ενώ το σύστημα διατήρησε τη σταθερότητά του ακόμα και όταν υποβλήθηκε σε συνθήκες υψηλής φόρτωσης.

ΚΕΦΑΛΑΙΟ 5: ΣΥΖΗΤΗΣΗ ΤΩΝ ΑΠΟΤΕΛΕΣΜΑΤΩΝ

Σε αυτό το κεφάλαιο, αναλύονται τα αποτελέσματα της ανάπτυξης και εφαρμογής του συστήματος διαχείρισης βαθμολογιών και παρακολούθησης φοιτητών. Η συζήτηση επικεντρώνεται στην ερμηνεία των αποτελεσμάτων, στις συγκρίσεις με άλλες υπάρχουσες λύσεις, καθώς και στα πλεονεκτήματα και τους περιορισμούς που παρουσιάζει η εφαρμογή.

5.1 Ερμηνεία Αποτελεσμάτων

Τα αποτελέσματα από τις δοκιμές και την αξιολόγηση της απόδοσης της εφαρμογής δείχνουν ότι το σύστημα πέτυχε τους στόχους του, προσφέροντας ένα αποτελεσματικό και αξιόπιστο εργαλείο για τη διαχείριση βαθμολογιών και παρουσιών φοιτητών. Η εφαρμογή αποδείχθηκε ικανή να διαχειριστεί μεγάλο όγκο δεδομένων και να εκτελέσει τις βασικές λειτουργίες καταχώρησης, ενημέρωσης και ανάκτησης δεδομένων χωρίς προβλήματα.

Η διαδικασία υπολογισμού των τελικών βαθμών, που βασίζεται σε φόρμουλες προσαρμοσμένες στον τύπο του φοιτητή και τις συγκεκριμένες ανάγκες του εκπαιδευτικού προγράμματος, λειτούργησε με ακρίβεια. Αυτό επιβεβαιώνεται από τις επιτυχείς δοκιμές που πραγματοποιήθηκαν, οι οποίες περιλάμβαναν διαφορετικά σενάρια και τύπους δεδομένων. Η δυνατότητα του συστήματος να προσαρμόζεται σε διαφορετικές φόρμουλες υπολογισμού δείχνει την ευελιξία του, επιτρέποντας στους χρήστες να καθορίζουν οι ίδιοι τα κριτήρια και τις προτεραιότητες βάσει των οποίων θα αξιολογούνται οι φοιτητές.

Οι δοκιμές απόδοσης ανέδειξαν την ικανότητα του συστήματος να ανταποκρίνεται άμεσα στις απαιτήσεις των χρηστών, ακόμα και σε περιπτώσεις αυξημένου φόρτου. Η χρήση τεχνολογιών όπως το Node.js για το backend και το PostgreSQL για τη διαχείριση της βάσης δεδομένων συνέβαλε σημαντικά σε αυτό. Οι χρόνοι απόκρισης ήταν σταθερά χαμηλοί,

υποδεικνύοντας ότι το σύστημα μπορεί να υποστηρίξει πολλούς ταυτόχρονους χρήστες χωρίς να επηρεάζεται η απόδοση.

Επιπλέον, η αξιοπιστία και η ασφάλεια του συστήματος δοκιμάστηκαν μέσω πολλαπλών σεναρίων που περιελάμβαναν τόσο τυπικές λειτουργίες όσο και περιπτώσεις λάθους. Το σύστημα απέδειξε ότι μπορεί να διαχειριστεί σφάλματα εισόδου και μη αναμενόμενες καταστάσεις χωρίς να καταρρεύσει, διασφαλίζοντας την ακεραιότητα των δεδομένων και την εμπειρία του χρήστη.

5.2 Συγκρίσεις με Άλλες Εφαρμογές

Η σύγκριση της αναπτυγμένης εφαρμογής με άλλες υπάρχουσες λύσεις στην αγορά δείχνει ότι η παρούσα εφαρμογή προσφέρει σημαντικά πλεονεκτήματα τόσο στη λειτουργικότητα όσο και στην προσαρμοστικότητα. Σε αντίθεση με πολλές παραδοσιακές λύσεις που είναι περιορισμένες σε προκαθορισμένες λειτουργίες, η εφαρμογή αυτή επιτρέπει την προσαρμογή των λειτουργιών και των φόρμουλων υπολογισμού των βαθμών ανάλογα με τις συγκεκριμένες ανάγκες κάθε εκπαιδευτικού προγράμματος.

Η χρήση του Next.js και του Tailwind CSS για το frontend προσφέρει μια μοντέρνα, responsive διεπαφή χρήστη που είναι εύκολη στην πλοήγηση και χρήση, κάτι που αποτελεί σημαντικό πλεονέκτημα σε σύγκριση με πιο παραδοσιακές ή στατικές εφαρμογές. Η δυνατότητα της εφαρμογής να εκτελείται και σε τοπικό επίπεδο χωρίς την ανάγκη για σύνδεση στο διαδίκτυο, προσθέτει επίσης αξία για περιβάλλοντα με περιορισμένη πρόσβαση στο διαδίκτυο ή για χρήση σε απομακρυσμένες περιοχές.

Ωστόσο, σε σύγκριση με μεγάλες εμπορικές πλατφόρμες διαχείρισης εκπαιδευτικού περιεχομένου, η παρούσα εφαρμογή έχει κάποιους περιορισμούς όσον αφορά τη δυνατότητα κλιμάκωσης και την ενσωμάτωση με άλλα συστήματα. Πολλές εμπορικές λύσεις προσφέρουν δυνατότητες cloud-based, που επιτρέπουν την εύκολη κλιμάκωση και την ενσωμάτωση με άλλες εφαρμογές και υπηρεσίες. Επίσης, αυτές οι πλατφόρμες συχνά περιλαμβάνουν πιο εξελιγμένα εργαλεία ανάλυσης δεδομένων και αναφορών, που μπορεί να είναι απαραίτητα σε μεγάλους οργανισμούς.

5.3 Πλεονεκτήματα και Περιορισμοί της Εφαρμογής

Η εφαρμογή που αναπτύχθηκε προσφέρει αρκετά πλεονεκτήματα, τα οποία την καθιστούν εξαιρετικά χρήσιμη για την διαχείριση βαθμολογιών και παρουσιών φοιτητών. Ένα από τα κύρια πλεονεκτήματα είναι η αυτοματοποίηση των διαδικασιών, η οποία εξοικονομεί χρόνο και μειώνει τα σφάλματα που προκύπτουν από χειροκίνητες καταχωρήσεις. Η ευελιξία του συστήματος επιτρέπει στους χρήστες να προσαρμόζουν τις φόρμουλες υπολογισμού και τις διαδικασίες αξιολόγησης, ανάλογα με τις ανάγκες τους. Η ενσωμάτωση αυτών των λειτουργιών σε ένα ενιαίο σύστημα προσφέρει μια ολοκληρωμένη λύση που διευκολύνει την παρακολούθηση και την αξιολόγηση των φοιτητών.

Επιπλέον, η χρήση τεχνολογιών όπως το TypeScript και το Zod προσφέρει αυξημένη ασφάλεια και αξιοπιστία, διασφαλίζοντας ότι μόνο έγκυρα δεδομένα εισάγονται και επεξεργάζονται στο σύστημα. Η αρχιτεκτονική του συστήματος είναι σχεδιασμένη με τρόπο που να επιτρέπει την εύκολη επέκταση και συντήρηση, ενώ η τεκμηρίωση και τα εργαλεία δοκιμής που αναπτύχθηκαν κατά τη διάρκεια της υλοποίησης, εξασφαλίζουν ότι το σύστημα μπορεί να συντηρηθεί και να αναπτυχθεί περαιτέρω με ευκολία.

Ωστόσο, υπάρχουν ορισμένοι περιορισμοί που πρέπει να ληφθούν υπόψη. Η εφαρμογή, αν και εξαιρετικά λειτουργική σε τοπικό περιβάλλον, ενδέχεται να χρειαστεί προσαρμογή για χρήση σε πιο ευρεία κλίμακα, όπως σε κατανεμημένα περιβάλλοντα ή σε περιβάλλοντα cloud. Επιπλέον, ενώ το σύστημα προσφέρει μια βασική σειρά εργαλείων αναφορών και ανάλυσης, μπορεί να χρειαστεί περαιτέρω ανάπτυξη για να ανταποκριθεί πλήρως στις ανάγκες οργανισμών που απαιτούν πιο σύνθετες αναλύσεις δεδομένων και εξελιγμένα εργαλεία αναφοράς.

Συνολικά, η εφαρμογή που αναπτύχθηκε παρέχει μια δυνατή και ευέλικτη βάση για τη διαχείριση βαθμολογιών και παρουσιών φοιτητών, προσφέροντας σημαντικά πλεονεκτήματα σε σχέση με πιο παραδοσιακές λύσεις. Παράλληλα, αναγνωρίζονται και τα

σημεία βελτίωσης και οι μελλοντικές ευκαιρίες για περαιτέρω ανάπτυξη, που θα μπορούσαν να επεκτείνουν τη λειτουργικότητα και την κλιμάκωση του συστήματος.

ΚΕΦΑΛΑΙΟ 6: ΟΔΗΓΙΕΣ ΕΓΚΑΤΑΣΤΑΣΗΣ ΚΑΙ ΧΡΗΣΗΣ

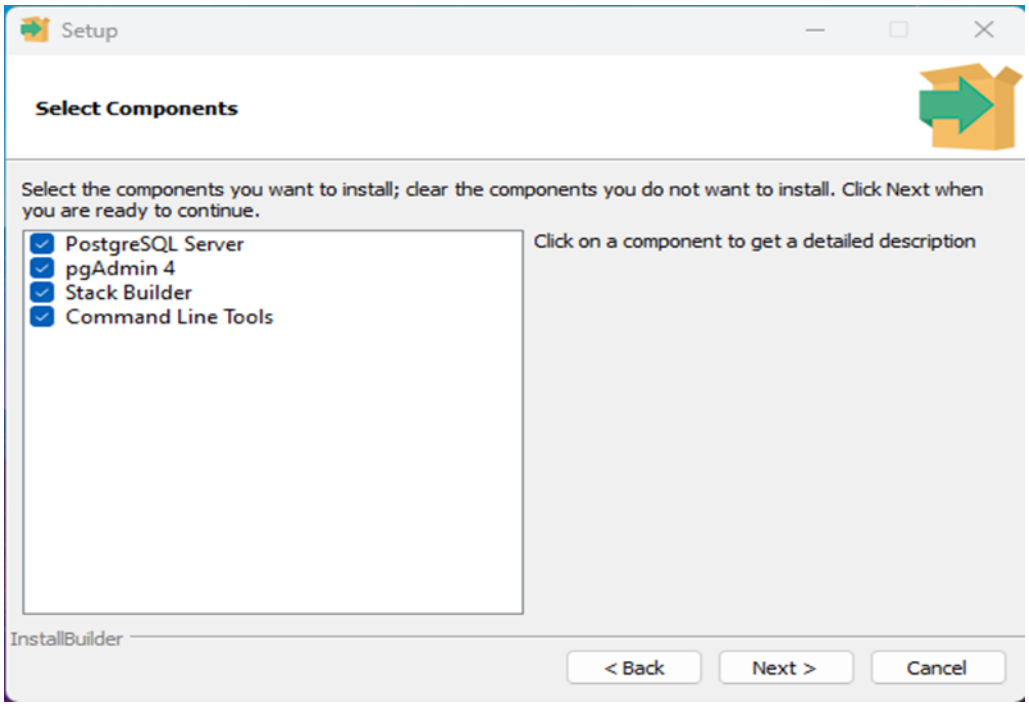
6.1 Οδηγίες Εγκατάστασης

Εγκατάσταση Απαραίτητων Εργαλείων

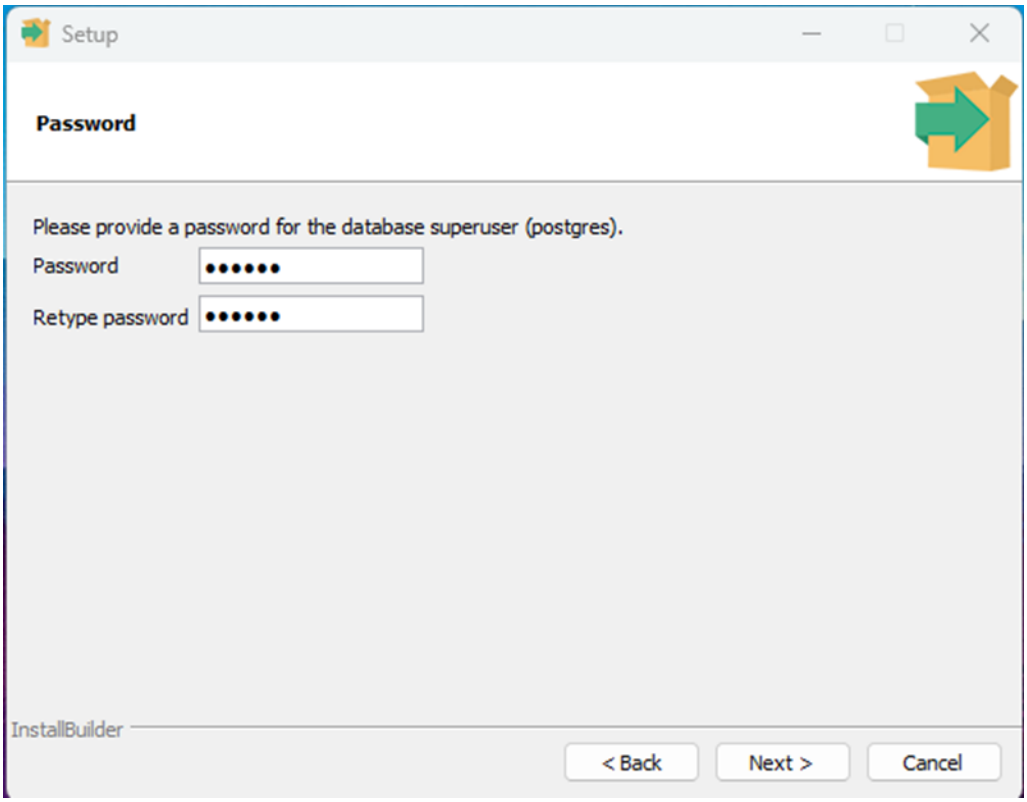
- Node.js (recommended version v20.18.0 (LTS))
- Postgres
- pgAdmin

Ρύθμιση Postgres

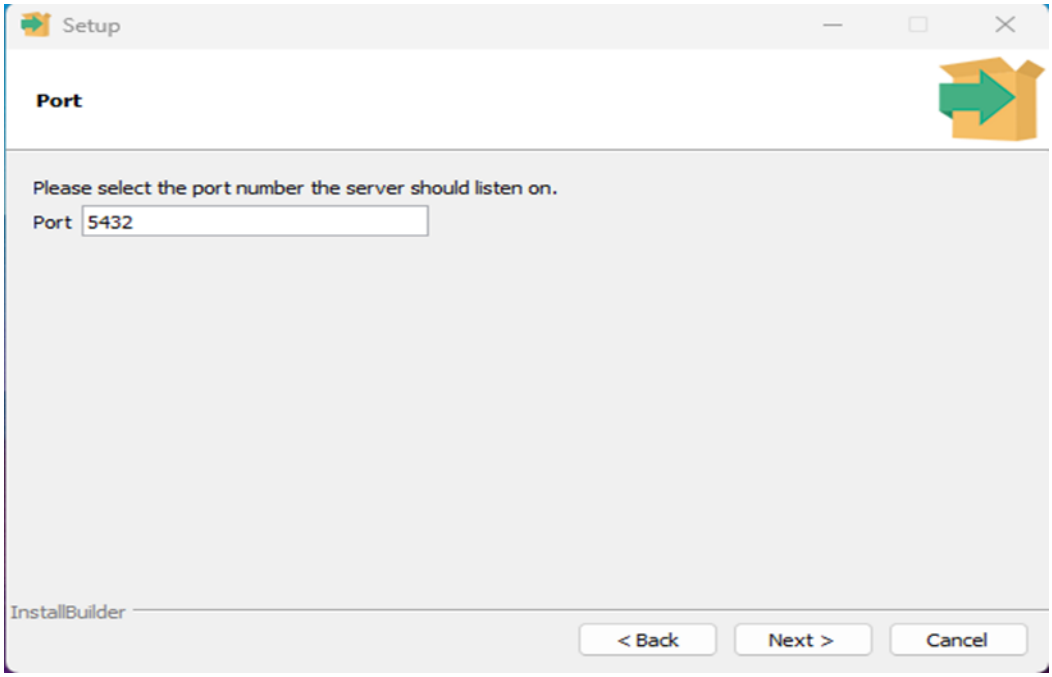
Ακολουθήστε τα παρακάτω βήματα για την ρύθμιση της postgres.



- Δημιουργούμε έναν χρήστη με κωδικό που θα μας επιτρέπει πρόσβαση στην Postgres.

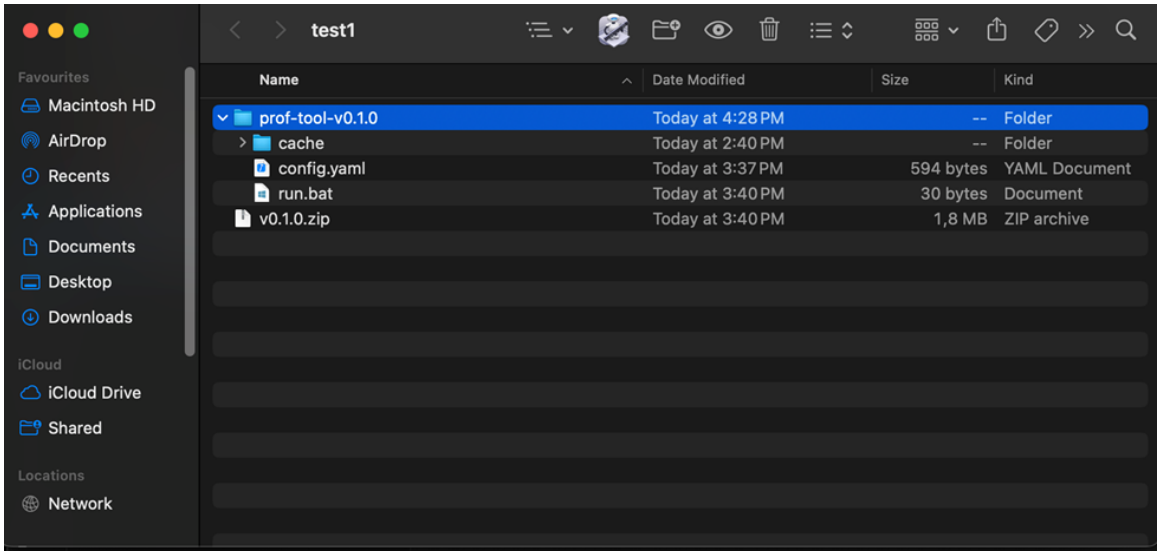


- Ορίζουμε το port που θέλουμε να χρησιμοποιεί η Postgres.

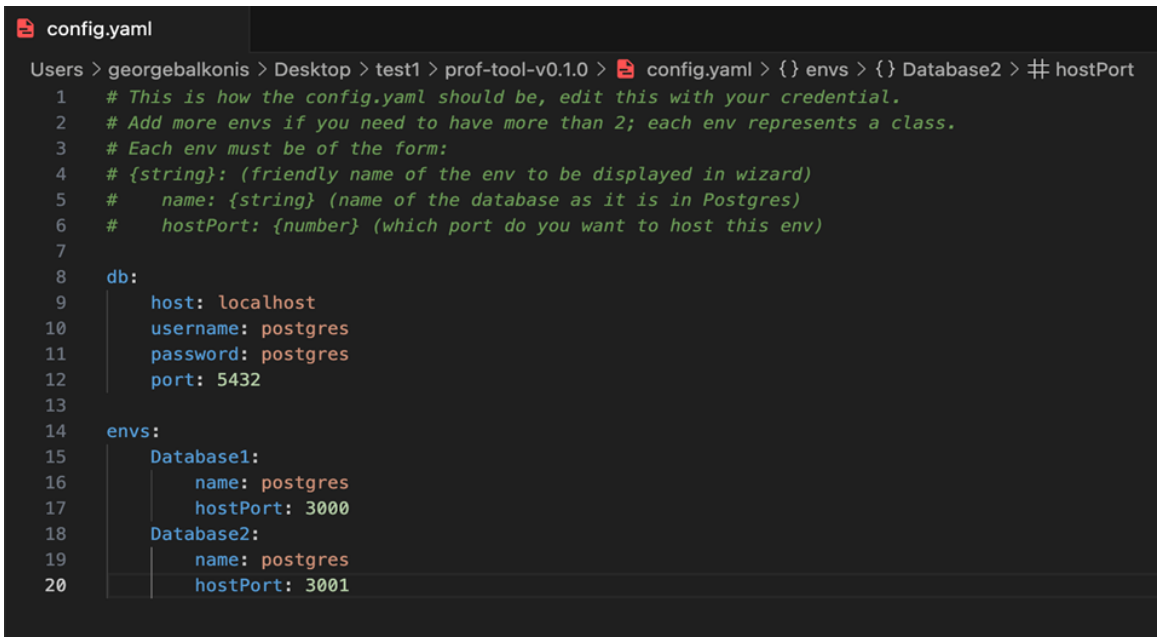


Εκτέλεση Αρχείου

- Κατεβάζουμε το αρχείο .zip και το αποσυμπιέζουμε. Το αρχείο που περιμένουμε είναι ο φάκελος prof-tool-v0.1.0.



- Ανοίγουμε το αρχείο config.yaml και εισάγουμε τα στοιχεία της βάσης που μόλις δημιουργήσαμε.



- Στοιχεία του Postgres server κάτω από το db:
  - Host: Το όνομα του Postgres server που χρησιμοποιήσαμε όταν ρυθμίσαμε το pgAdmin.
  - Username: Το όνομα του superuser της βάσης.

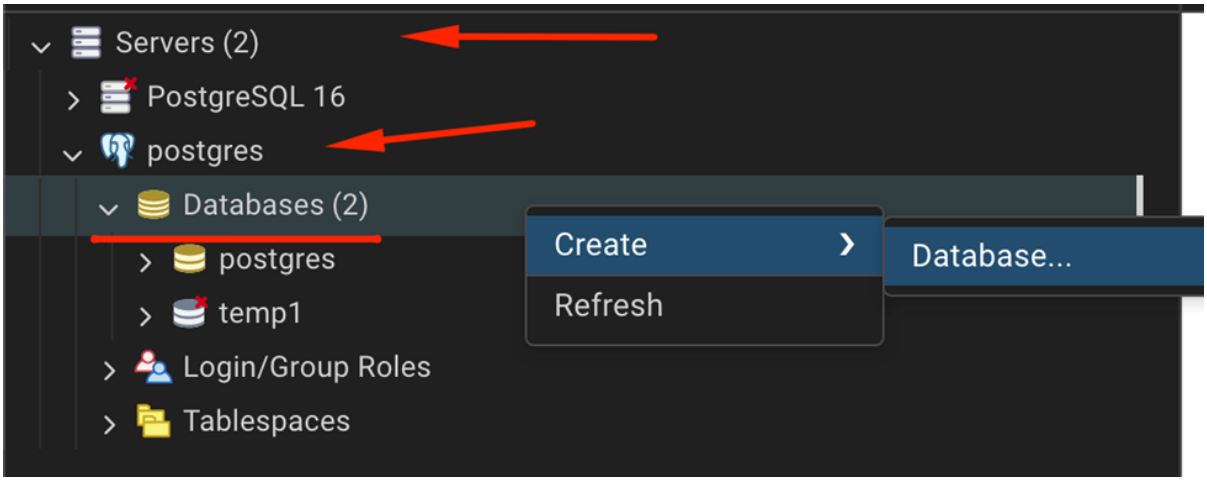
- Password: Ο κωδικός του superuser της βάσης.
- Port: Το port που επιλέξαμε για να τρέχει η βάση δεδομένων μας.

- Στοιχεία κάτω από τα envs (για την ρύθμιση της σελίδας μας):
  - Database1: Ψευδώνυμο για να αναγνωρίζουμε ποια βάση είναι στο μενού επιλογών.
  - Name: Το όνομα της βάσης δεδομένων.
  - HostPort: Το port στο οποίο θέλουμε να σερβίρεται η σελίδα αυτής της βάσης.

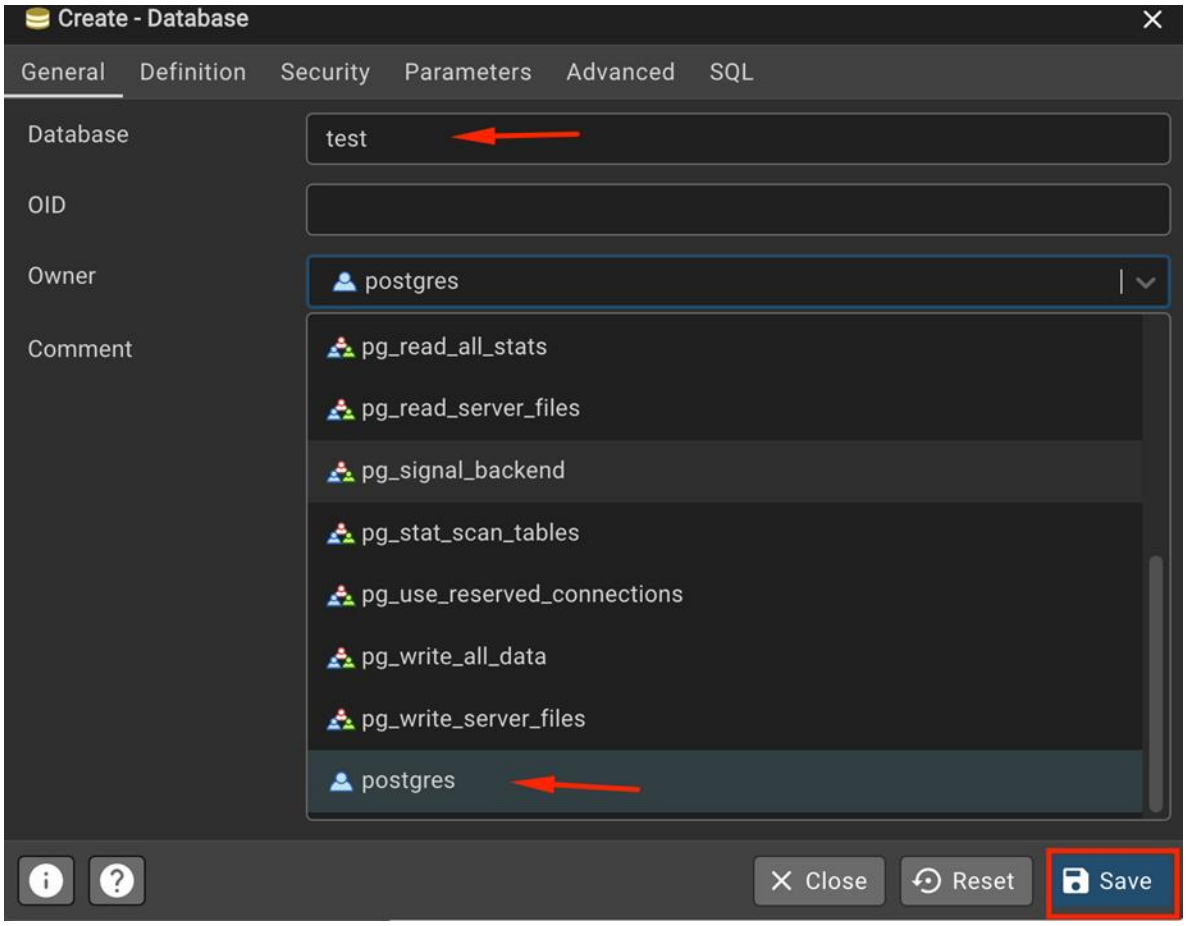
Δημιουργία επιπλέον βάσης (μαθήματος)

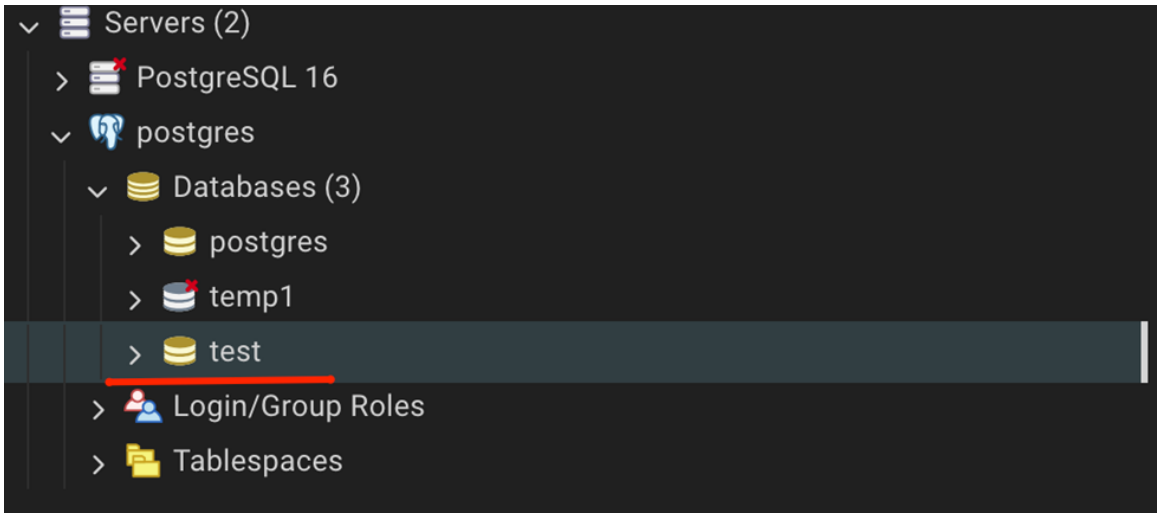
Δημιουργούμε τη βάση δεδομένων στον Postgres μέσω του pgAdmin.

- Από το μενού, κάνουμε δεξί κλικ στο Databases > Create > Database.



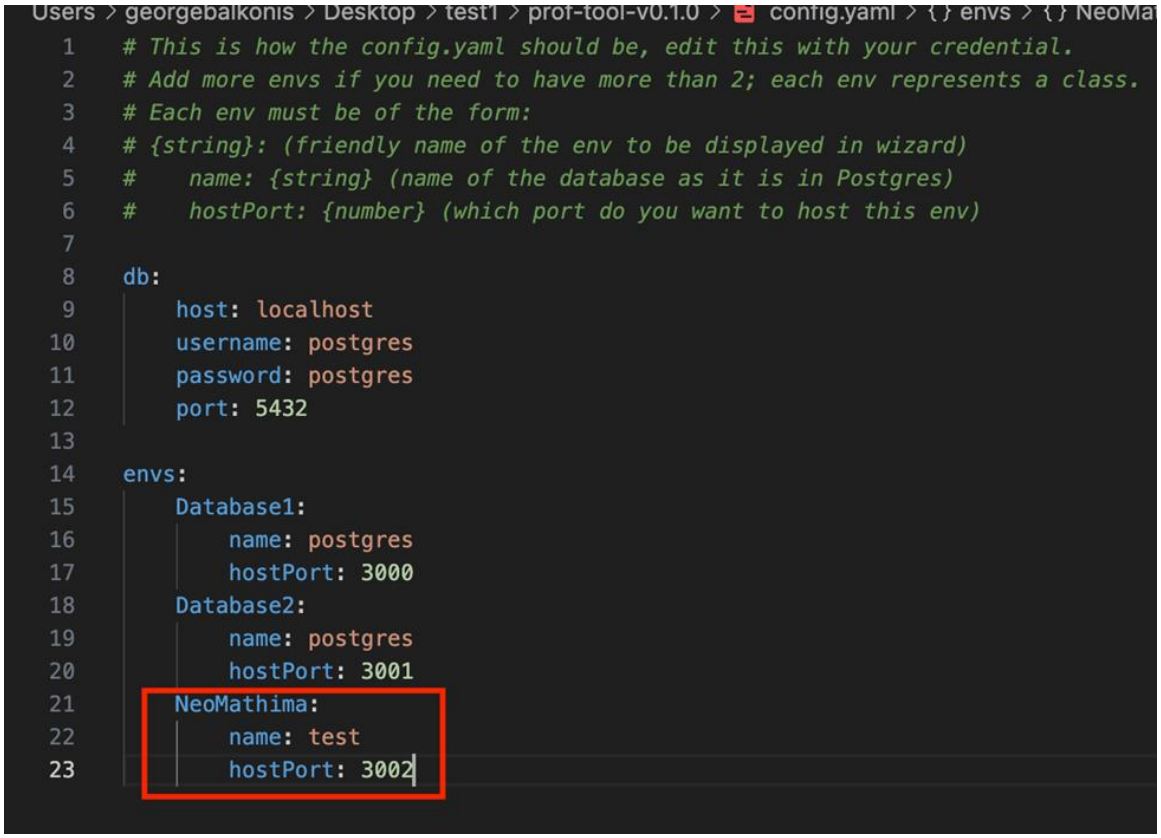
- Δίνουμε μια ονομασία για τη βάση. Σιγουρευόμαστε ότι ο Owner είναι ο superuser μας και πατάμε Save.



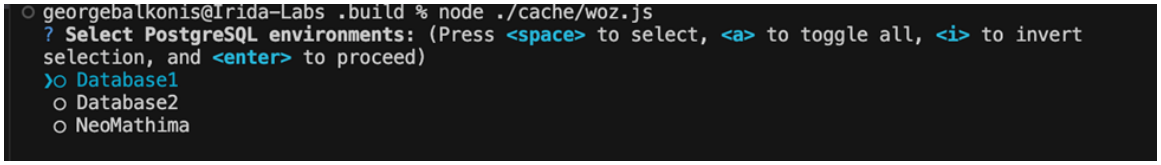


Προσθήκη Νέας Βάσης στο config.yaml

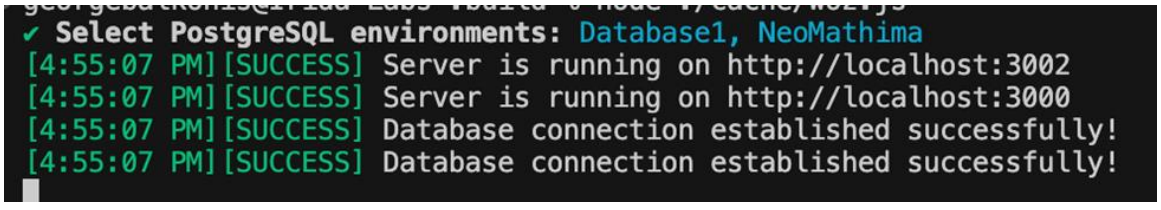
- Ανοίγουμε το αρχείο config.yaml και δημιουργούμε ένα νέο env για τη νέα βάση. Προσέχουμε να μην έχουμε δύο ίδιες θύρες (ports).



- Όταν προσθέσουμε τη νέα βάση, τρέχουμε το run.bat.
- Επιλέγουμε ποιά μαθήματα (envs) θέλουμε να τρέξουμε.



- Μεταβείτε στις ανάλογες διευθύνσεις για να έχετε πρόσβαση στα μαθήματα που επιλέξατε.



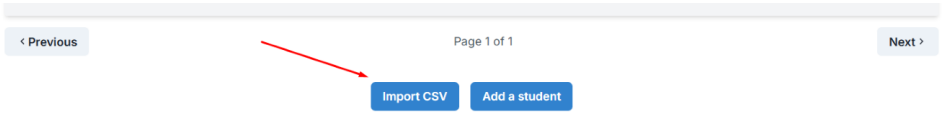
Με αυτές τις ρυθμίσεις η εφαρμογή θα είναι έτοιμη για χρήση.

6.2 Οδηγίες Χρήσης

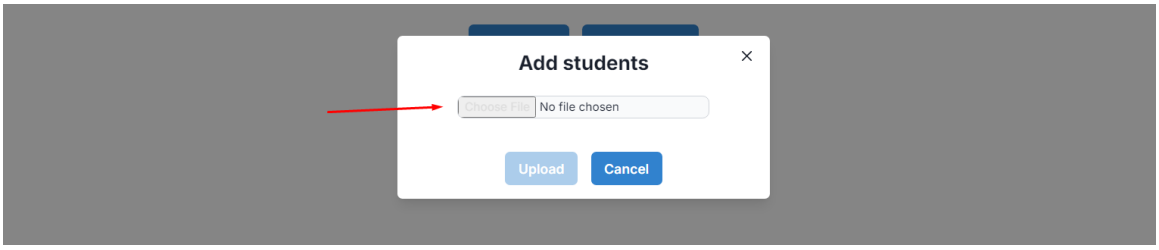
Σελίδα Students

Ανέβασμα Φοιτητών (Εισαγωγή CSV):

Για να ανεβάσετε μια λίστα φοιτητών, κάντε κλικ στο κουμπί "Import CSV".



Θα εμφανιστεί ένα παράθυρο διαλόγου όπου μπορείτε να επιλέξετε και να ανεβάσετε ένα αρχείο CSV.



Το αρχείο CSV πρέπει να περιέχει τις εξής στήλες με την ακριβή σειρά:

	A	B	C	D	E
1	studentId	name	surname	academicYear	studentType
2	3091	Thodoris	Tsironis	2017	second-time
3	3093	Thodoris	Lappas	2017	first-time
4	3092	Giwrgos	Balkonis	2016	first-time
5					

StudentId: Ο μοναδικός αριθμός ταυτότητας του φοιτητή.

Name: Το όνομα του φοιτητή.

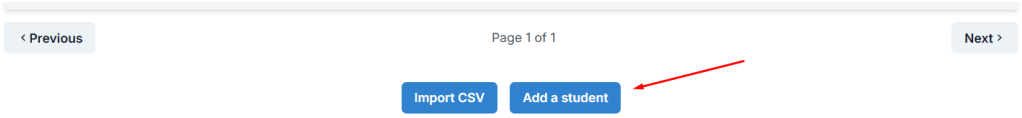
Surname: Το επώνυμο του φοιτητή.

AcademicYear: Το τρέχον ακαδημαϊκό έτος του φοιτητή.

StudentType: Ο τύπος του φοιτητή (π.χ. δίνει πρώτη φορά, δεύτερη φορά κτλπ.).

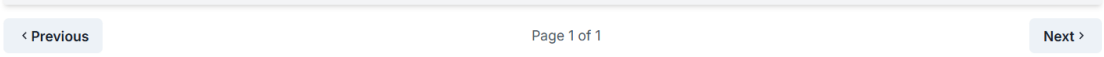
Προσθήκη Φοιτητή Χειροκίνητα:

Αν θέλετε να προσθέσετε έναν φοιτητή χειροκίνητα, κάντε κλικ στο κουμπί "Add a student" και συμπληρώστε τα απαραίτητα στοιχεία.



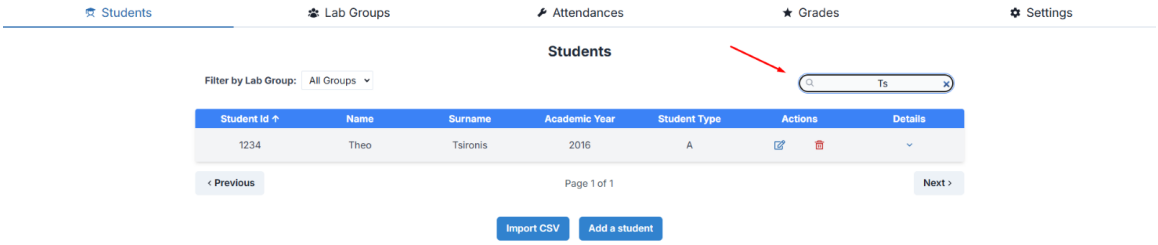
Πλοήγηση στις Σελίδες:

Αν η λίστα φοιτητών είναι μεγάλη, μπορείτε να πλοηγηθείτε σε διαφορετικές σελίδες χρησιμοποιώντας τα κουμπιά Next και Previous στο κάτω μέρος της σελίδας.



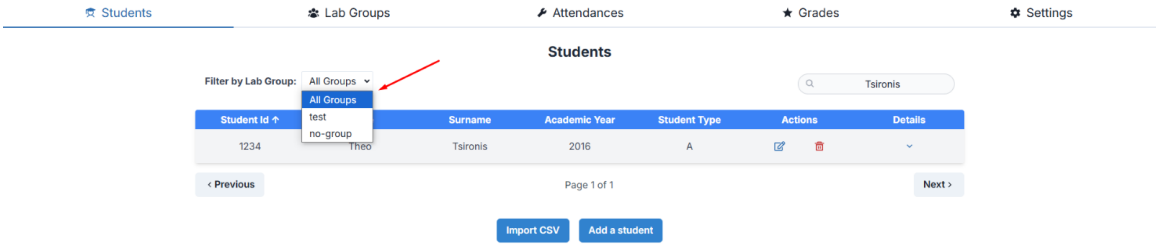
Αναζήτηση Φοιτητή:

Μπορείτε να αναζητήσετε έναν συγκεκριμένο φοιτητή εισάγοντας το όνομά του ή τον αριθμό μητρώου του στο πεδίο αναζήτησης που βρίσκεται δεξιά.



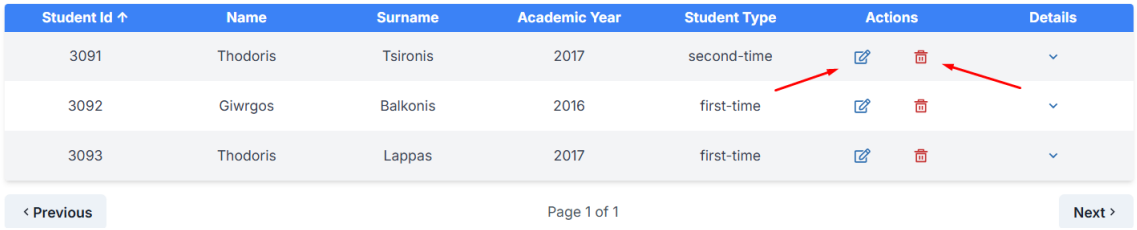
Φιλτράρισμα κατά Ομάδα Εργαστηρίου:

Μπορείτε να φιλτράρετε τους φοιτητές ανάλογα με την ομάδα εργαστηρίου επιλέγοντας την αντίστοιχη ομάδα από το αναδυόμενο μενού δίπλα από το "Filter by Lab Group".



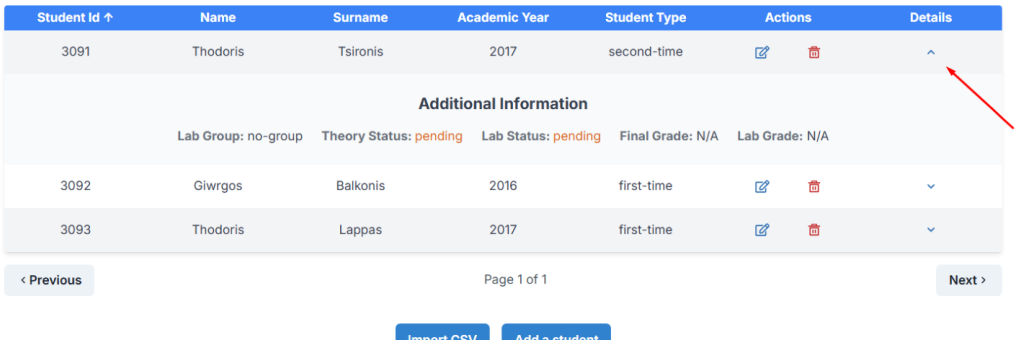
Επεξεργασία / Διαγραφή Φοιτητή:

Μπορείτε να επεξεργαστείτε τα στοιχεία του κάθε φοιτητή ή να τον διάγραψετε απο τα ανάλογα εικονίδια στον πίνακα



Επιπλέον Πληροφορίες:

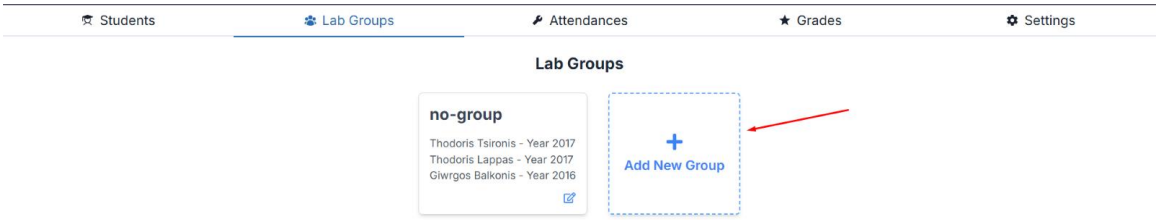
Μπορείτε να δείτε επιπλέον πληροφορίες για τους φοιτητές πατώντας το κάτω βελάκι στην δεξιά πλευρά του πίνακα για τον κάθε φοιτητή.



Σελίδα Lab Groups

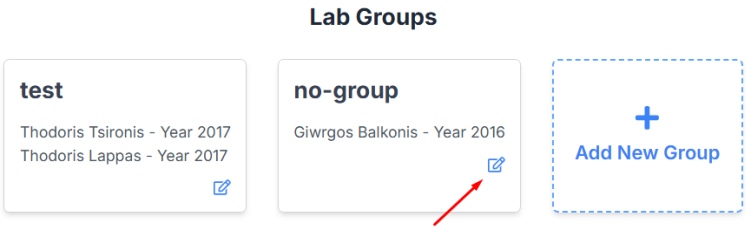
Δημιουργία Εργαστηριακής Ομάδας:

Αν θέλετε να δημιουργήσετε μία εργαστηριακή ομάδα, κάντε κλικ στο κουμπί "Add New Group" και σημπληρώστε το όνομα και τους φοιτητές της συγκεκριμένης ομάδας.



Επεξεργασία Εργαστηριακής Ομάδας:

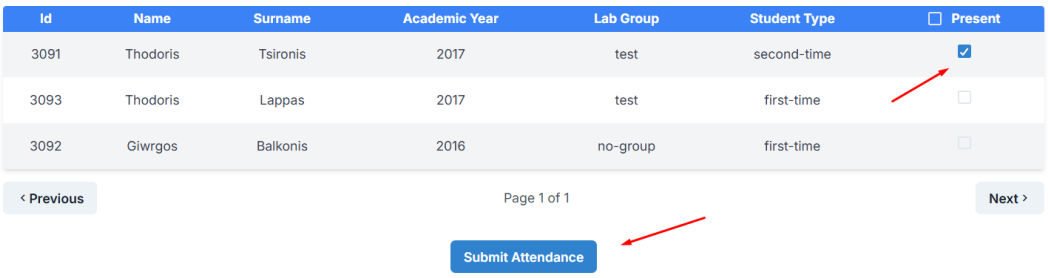
Αν θέλετε να επεξεργαστείτε μία ήδη υπάρχουσα εργαστηριακή ομάδα, κάντε κλικ στο εικονίδιο στην κάτω δεξιά γωνία αυτής της εργαστηριακής ομάδας.



Σελίδα Attendances

Συμπλήρωση απουσιών παρουσιών:

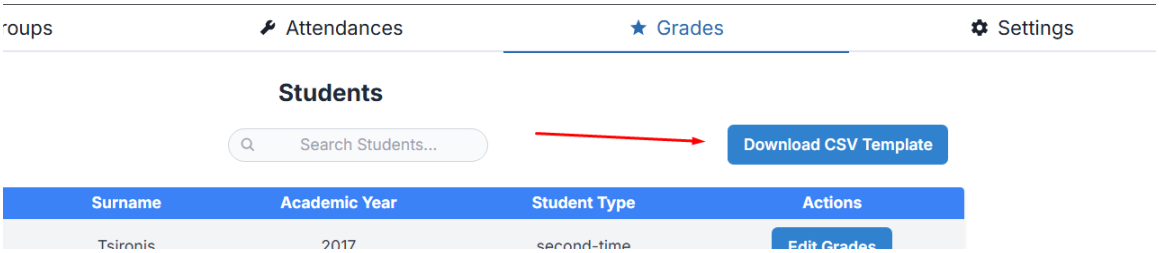
Πατήστε check στο κουτάκι των φοιτητών που είναι παρόντες στο μάθημα και κάντε κλικ στο κουμπί "Submit Attendance"



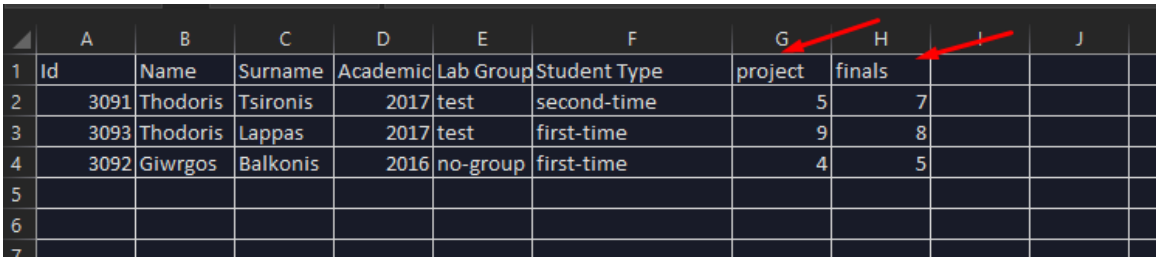
Σελίδα Grades

Ανέβασμα Βαθμών(Εισαγωγή CSV):

Κάντε κλικ στο κουμπί “Download CSV Template” για να κατεβάσετε το csv αρχείο με τους φοιτητές.

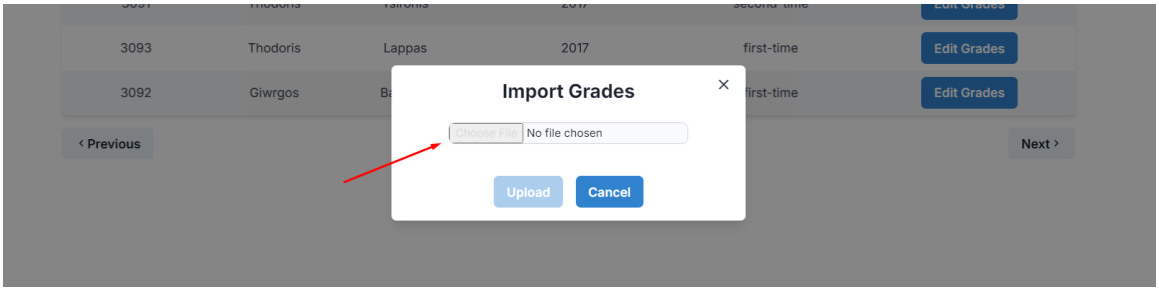


Προσθέστε τους βαθμούς βάζοντας το όνομα του βαθμού για τίτλο και δίπλα από κάθε φοιτητή τον ανάλογο βαθμό.



	A	B	C	D	E	F	G	H	I	J
1	Id	Name	Surname	Academic	Lab Group	Student Type	project	finals		
2	3091	Thodoris	Tsironis	2017	test	second-time	5	7		
3	3093	Thodoris	Lappas	2017	test	first-time	9	8		
4	3092	Giwrgos	Balkonis	2016	no-group	first-time	4	5		
5										
6										

Για να ανεβάσετε τους βαθμούς, κάντε κλικ στο κουμπί "Import CSV". Θα εμφανιστεί ένα παράθυρο διαλόγου όπου μπορείτε να επιλέξετε και να ανεβάσετε το αρχείο.



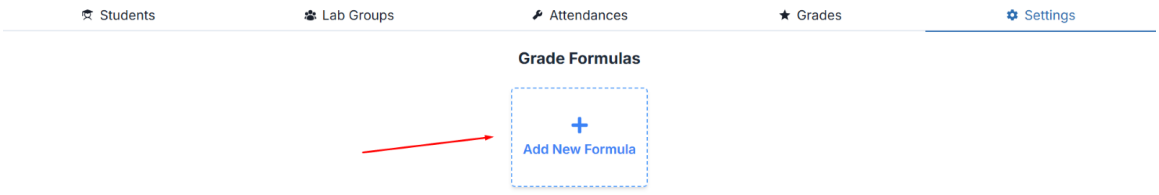
Επεξεργασία Βαθμών:

Κάντε κλικ στο κουμπί “Edit Grades” δίπλα από κάθε φοιτητή για να επεξεργαστείτε, να διαγράψετε τους βαθμούς του ή να προσθέσετε καινούριους.

Σελίδα Settings

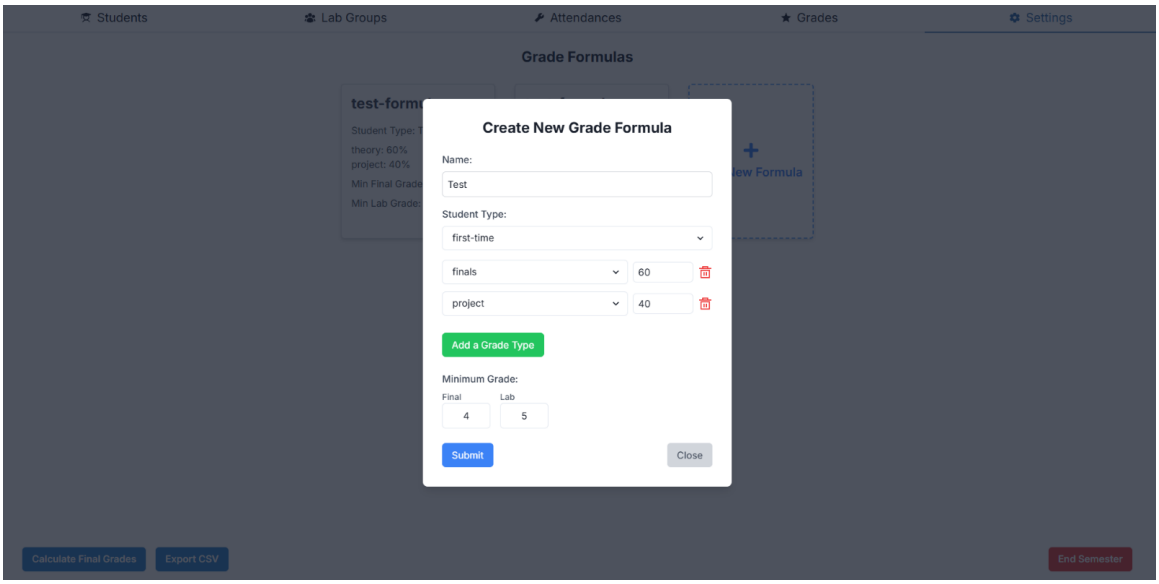
Δημιουργία καινούριου τύπου βαθμολόγησης:

Κάντε κλικ στο κουμπί “Add new Formula” για να προσθέσετε έναν καινούριο τύπο βαθμολόγησης.



Επιλέξτε την ονομασία του τύπου βαθμολόγησης, τον τύπο φοιτητή στον οποίο θα εφαρμόζεται, τους βαθμούς που θα περιλαμβάνονται στον τελικό υπολογισμό με τα αντίστοιχα ποσοστά τους, καθώς και τον ελάχιστο απαιτούμενο βαθμό για να περάσει το

μάθημα.



Υπολογισμός Τελικού Βαθμού:

Κάντε κλικ στο κουμπί “Calculate Final Grades” για να υπολογίσετε τους τελικού βαθμούς των φοιτητών, ανάλογα με τους τύπους που έχετε δημιουργήσει.

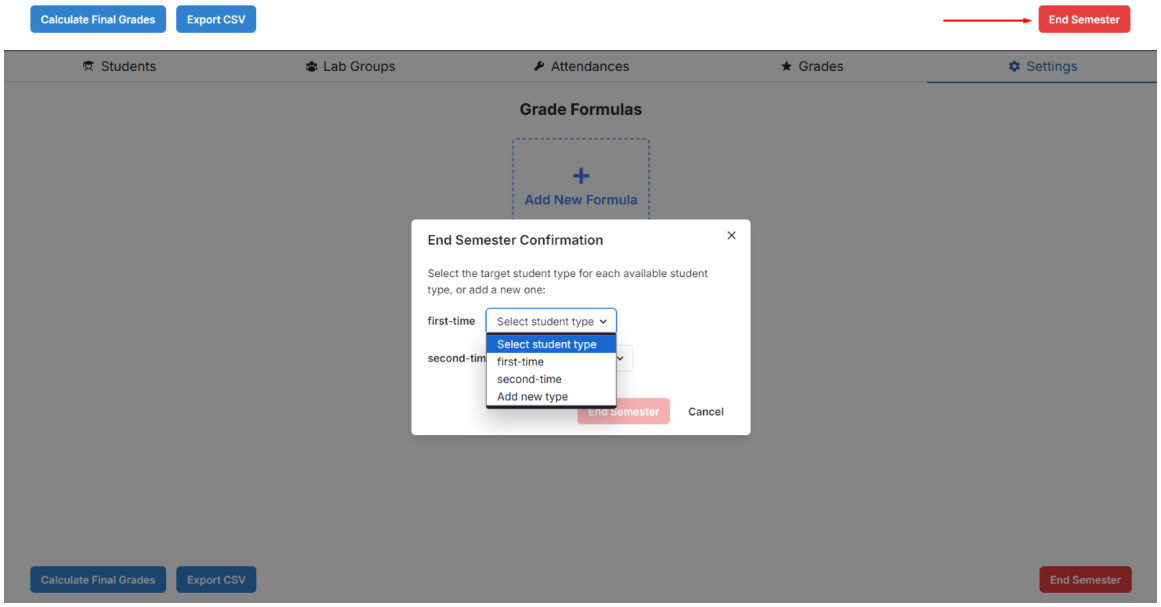
Κατέβασμα φοιτητών:

Κάντε κλικ στο κουμπί “Export CSV” για να κατεβάσετε τους φοιτητές που χρειάζεστε χρησιμοποιώντας τα φίλτρα στο παράθυρο διαλόγου που θα ανοίξει. Εάν επιθυμείτε να κατεβάσετε όλους τους φοιτητές αφήστε τα φίλτρα κενά.

Τέλος εξαμήνου:

- Κάντε κλικ στο κουμπί “End Semester” για να τελειώσετε το εξάμηνο. Θα σας ζητηθεί να επιλέξετε σε τι θα αλλάξουν οι τύποι των φοιτητών (π.χ. πρωτοετής ->

δευτεροετής κλπ.) και τέλος αφού πατήσετε ξανά “End Semester” , θα διαγραφθούν όλοι οι φοιτητές που έχουν περάσει το μάθημα, ενώ όσοι κόπηκαν θα συνεχίσουν να υπάρχουν. Ταυτόχρονα θα κατέβουν δύο αρχεία csv, ένα με τους φοιτητές που πέρασαν και ένα με τους φοιτητές που κόπηκαν.



ΚΕΦΑΛΑΙΟ 7: ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ ΠΕΡΑΙΤΕΡΩ ΑΝΑΠΤΥΞΗΣ

Στο κεφάλαιο αυτό παρουσιάζονται τα συνολικά συμπεράσματα της πτυχιακής εργασίας και διατυπώνονται προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις της εφαρμογής διαχείρισης βαθμολογιών και παρακολούθησης φοιτητών. Η εργασία αυτή ανέδειξε τις δυνατότητες της τεχνολογικής καινοτομίας στην εκπαίδευση, παρέχοντας ένα ισχυρό εργαλείο για την ενίσχυση της διαφάνειας και της αποτελεσματικότητας της εκπαιδευτικής διαδικασίας.

7.1 Συμπεράσματα

Η ανάπτυξη της εφαρμογής είχε ως πρωταρχικό στόχο τη δημιουργία ενός ολοκληρωμένου συστήματος που διευκολύνει τη διαχείριση των βαθμολογιών και των παρουσιών των φοιτητών. Το σύστημα σχεδιάστηκε για να παρέχει ένα εύχρηστο και αποδοτικό περιβάλλον εργασίας για τους καθηγητές, επιτρέποντάς τους να καταχωρούν, να επεξεργάζονται και να παρακολουθούν τις βαθμολογίες και τις παρουσίες με ακρίβεια και ευκολία. Τα αποτελέσματα των δοκιμών επιβεβαίωσαν ότι η εφαρμογή πέτυχε τους αρχικούς της στόχους, αποδεικνύοντας ότι μπορεί να διαχειριστεί μεγάλο όγκο δεδομένων και να εκτελεί τις απαιτούμενες λειτουργίες με υψηλή ακρίβεια.

Η εφαρμογή χρησιμοποιεί σύγχρονες τεχνολογίες, όπως το Node.js για το backend και το Next.js για το frontend, προσφέροντας μια φιλική και μοντέρνα διεπαφή χρήστη, η οποία ανταποκρίνεται άμεσα στις ανάγκες των χρηστών. Το σύστημα επιτρέπει επίσης την προσαρμογή των φόρμουλων υπολογισμού των βαθμών, γεγονός που προσδίδει ευελιξία στη διαχείριση και αξιολόγηση των φοιτητών, βάσει των συγκεκριμένων αναγκών κάθε εκπαιδευτικού προγράμματος.

Παρά την επιτυχία της εφαρμογής στην επίτευξη των στόχων της, η ανάλυση και η σύγκριση με άλλες λύσεις ανέδειξαν ορισμένα σημεία για βελτίωση και επέκταση. Υπάρχει η δυνατότητα να εμπλουτιστεί περαιτέρω το σύστημα, ώστε να εξυπηρετεί όχι μόνο τους καθηγητές αλλά και τους φοιτητές ως χρήστες.

7.2 Προτάσεις για Μελλοντική Έρευνα ή Ανάπτυξη

Για την περαιτέρω ανάπτυξη και βελτίωση της εφαρμογής, προτείνονται οι ακόλουθες επεκτάσεις και λειτουργίες:

1. Ενσωμάτωση Φοιτητών ως Χρήστες της Εφαρμογής

Μια σημαντική πρόταση για μελλοντική ανάπτυξη είναι η δυνατότητα προσθήκης φοιτητών ως χρήστες του συστήματος. Αυτή η επέκταση θα επέτρεπε στους φοιτητές να έχουν προσωπικούς λογαριασμούς στην εφαρμογή, μέσω των οποίων θα μπορούν να βλέπουν τις βαθμολογίες τους, τις παρουσίες τους στα εργαστήρια και άλλες σημαντικές πληροφορίες για την ακαδημαϊκή τους πορεία.

Η προσθήκη αυτής της λειτουργίας θα ενισχύσει τη διαφάνεια και την εμπιστοσύνη στο εκπαιδευτικό σύστημα, καθώς οι φοιτητές θα έχουν άμεση πρόσβαση στα δεδομένα τους και θα μπορούν να παρακολουθούν την πρόοδό τους σε πραγματικό χρόνο. Επιπλέον, αυτή η δυνατότητα θα μειώσει το διοικητικό φόρτο για τους καθηγητές και το προσωπικό, καθώς οι φοιτητές θα μπορούν να λαμβάνουν τις πληροφορίες που χρειάζονται απευθείας από την εφαρμογή, χωρίς να απαιτείται προσωπική επικοινωνία ή επιπλέον διαχείριση.

2. Αποστολή Ειδοποιήσεων μέσω Email στους Φοιτητές

Μια επιπλέον δυνατότητα που θα μπορούσε να ενσωματωθεί στο σύστημα είναι η αποστολή αυτόματων ειδοποιήσεων μέσω email στους φοιτητές, κάθε φορά που αναρτώνται οι βαθμολογίες ή καταχωρούνται σημαντικές αλλαγές που τους αφορούν. Αυτή η λειτουργία θα εξασφάλιζε ότι οι φοιτητές ενημερώνονται άμεσα για τις νέες εξελίξεις και τις αλλαγές στην ακαδημαϊκή τους κατάσταση, ενισχύοντας έτσι την επικοινωνία και την ενημέρωση.

Η αυτόματη αποστολή ειδοποιήσεων θα μπορούσε να υλοποιηθεί μέσω ενός συστήματος email notification που θα ενσωματώνεται με το backend της εφαρμογής. Χρησιμοποιώντας γνωστές υπηρεσίες email όπως το SendGrid ή το Amazon SES (Simple Email Service), το σύστημα θα μπορούσε να στέλνει εξατομικευμένα μηνύματα στους φοιτητές, εξοικονομώντας χρόνο και διασφαλίζοντας την έγκαιρη ενημέρωση των ενδιαφερομένων.

3. Μεταφορά του Συστήματος σε Περιβάλλον Cloud (AWS)

Ένα ακόμα σημαντικό βήμα για τη βελτίωση και επέκταση της εφαρμογής είναι η μεταφορά του συστήματος σε περιβάλλον cloud, όπως το Amazon Web Services (AWS). Η χρήση του cloud θα προσφέρει πολλαπλά οφέλη, όπως η αυξημένη κλιμάκωση, η βελτιωμένη διαθεσιμότητα και η μεγαλύτερη ασφάλεια.

Με τη μεταφορά της εφαρμογής στο AWS, θα ήταν δυνατό να εκμεταλλευτούμε πλήρως τις δυνατότητες κλιμάκωσης και ευελιξίας που προσφέρει το cloud. Για παράδειγμα, η βάση δεδομένων θα μπορούσε να μεταφερθεί σε Amazon RDS (Relational Database Service), ενώ το backend της εφαρμογής θα μπορούσε να φιλοξενηθεί σε Amazon EC2 (Elastic Compute Cloud) ή σε υπηρεσίες serverless όπως το AWS Lambda. Επιπλέον, η χρήση υπηρεσιών όπως το AWS Elastic Load Balancer θα εξασφαλίσει την ομαλή διαχείριση του φόρτου εργασίας και τη βελτιστοποίηση της απόδοσης.

Η μεταφορά στο cloud θα επιτρέψει επίσης την εύκολη ενσωμάτωση με άλλες υπηρεσίες και πλατφόρμες, διευκολύνοντας τη διασύνδεση με εξωτερικά συστήματα και την επέκταση των λειτουργιών του συστήματος. Επιπλέον, η χρήση προηγμένων χαρακτηριστικών ασφαλείας που παρέχονται από το AWS θα αυξήσει την προστασία των δεδομένων, διασφαλίζοντας την ιδιωτικότητα και την ασφάλεια των ευαίσθητων πληροφοριών των φοιτητών και των καθηγητών.

4. Υποστήριξη Πολλαπλών Μαθημάτων σε Μία Βάση Δεδομένων

Μια ακόμα σημαντική πρόταση για μελλοντική ανάπτυξη είναι η δυνατότητα υποστήριξης πολλών μαθημάτων σε μία ενιαία βάση δεδομένων. Αυτή η αλλαγή θα διευκόλυνε τη διαχείριση δεδομένων, εξαλείφοντας την ανάγκη ύπαρξης ξεχωριστών βάσεων για κάθε μάθημα, ενώ παράλληλα θα διατηρούσε την οργάνωση και την αποτελεσματικότητα του συστήματος.

Η υλοποίηση αυτής της δυνατότητας θα μπορούσε να βασιστεί στη δημιουργία μιας νέας δομής στη βάση δεδομένων, η οποία θα περιλαμβάνει μια κεντρική οντότητα για τα μαθήματα. Η σχέση μεταξύ μαθημάτων, φοιτητών, βαθμολογιών και παρουσιών θα διατηρείται μέσω ενός μοντέλου σχεσιακής βάσης δεδομένων. Με αυτόν τον τρόπο, θα είναι δυνατό να αποθηκεύονται όλα τα δεδομένα σε μία ενιαία βάση, χωρίς να απαιτείται η δημιουργία νέας βάσης δεδομένων για κάθε μάθημα.

Αυτή η αλλαγή θα προσφέρει πολλά πλεονεκτήματα, όπως η μείωση της πολυπλοκότητας στη διαχείριση του συστήματος, η βελτίωση της απόδοσης κατά την πρόσβαση στα δεδομένα και η ευκολότερη ενσωμάτωση νέων λειτουργιών. Επιπλέον, η ανάπτυξη κατάλληλων λειτουργιών φιλτραρίσματος και παρουσίασης στο frontend θα επιτρέψει στους χρήστες της εφαρμογής να βλέπουν και να διαχειρίζονται δεδομένα πολλαπλών μαθημάτων με ευκολία.

Με την προσθήκη αυτής της λειτουργίας, το σύστημα θα γίνει πιο ευέλικτο, επιτρέποντας τη χρήση του σε μεγαλύτερα εκπαιδευτικά περιβάλλοντα και ικανοποιώντας τις ανάγκες διαχείρισης περισσότερων μαθημάτων από το ίδιο περιβάλλον.

7.3 Τελικές Παρατηρήσεις

Η πτυχιακή εργασία αυτή απέδειξε ότι η ανάπτυξη μιας εφαρμογής διαχείρισης βαθμολογιών και παρακολούθησης φοιτητών μπορεί να αποτελέσει ένα ισχυρό εργαλείο για την εκπαιδευτική κοινότητα, βελτιώνοντας τη διαφάνεια, την αποδοτικότητα και την ακρίβεια της εκπαιδευτικής διαδικασίας. Η επιτυχής υλοποίηση των βασικών λειτουργιών και η θετική αξιολόγηση της απόδοσης της εφαρμογής δείχνουν ότι το σύστημα είναι έτοιμο να χρησιμοποιηθεί σε πραγματικές συνθήκες.

Οι προτάσεις για μελλοντική ανάπτυξη που περιγράφηκαν σε αυτό το κεφάλαιο αναδεικνύουν τις δυνατότητες επέκτασης και βελτίωσης του συστήματος, ώστε να ανταποκρίνεται καλύτερα στις ανάγκες των χρηστών και να προσαρμόζεται στις απαιτήσεις ενός συνεχώς εξελισσόμενου εκπαιδευτικού περιβάλλοντος. Με τη συνέχιση της ανάπτυξης και της βελτίωσης, η εφαρμογή μπορεί να εξελιχθεί περαιτέρω, παρέχοντας περισσότερα χαρακτηριστικά και δυνατότητες που θα ενισχύσουν τη συνολική εμπειρία των χρηστών και θα προσφέρουν αξία στην εκπαιδευτική κοινότητα.

Βιβλιογραφία

1. Node.js Foundation. (n.d.). Node.js. Πηγή: https://nodejs.org

2. PostgreSQL Global Development Group. (n.d.). PostgreSQL: The world's most advanced open source database. Πηγή: https://www.postgresql.org

3. TypeScript Team. (n.d.). TypeScript: JavaScript With Syntax For Types. Πηγή: https://www.typescriptlang.org

4. Next.js Team. (n.d.). Next.js: The React Framework for Production. Πηγή: https://nextjs.org

5. Tailwind Labs. (n.d.). Tailwind CSS: Rapidly build modern websites without ever leaving your HTML. Πηγή: https://tailwindcss.com

6. Postman. (n.d.). Postman: Simplifying API Development. Πηγή: https://www.postman.com

7. Zod. (n.d.). Zod: TypeScript-first schema declaration and validation library. Πηγή: https://zod.dev

8. TypeORM. (n.d.). TypeORM: ORM for TypeScript and JavaScript (ES7, ES6, ES5). Πηγή: https://typeorm.io

9. Bun. (n.d.). Bun: A fast all-in-one JavaScript runtime. Πηγή: https://bun.sh

10. Chakra UI. (n.d.). Chakra UI: Simple, modular, and accessible component library for React. Available at: https://chakra-ui.com