Διεθνές Πανεπιστήμιο Ελλάδος
Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

Ανάπτυξη Διαδικτυακού Συστήματος Προγραμματισμού Μαθημάτων και Διαχείρισης Αιθουσών Διδασκαλίας για Εκπαιδευτικά Ιδρύματα

Φοιτητής:
Αλέξανδρος Γκατζόπουλος
Αριθμός Μητρώου: 175135

Επιβλέπων:
Αντώνης Σιδηρόπουλος

26 Μαΐου 2024

Τίτλος Π.Ε. Ανάπτυξη και Εφαρμογή ενός Διαδικτυακού Συστήματος Προγραμματισμού Μαθημάτων και Διαχείρισης Αιθουσών με Χρήση για Εκπαιδευτικά Ιδρύματα
Κωδικός Π.Ε. 23262
Ονοματεπώνυμο φοιτητή/τών Αλέξανδρος Γκατζόπουλος
Ονοματεπώνυμο εισηγητή Αντώνης Σιδηρόπουλος
Ημερομηνία ανάληψης Π.Ε. 04-10-2023
Ημερομηνία περάτωσης Π.Ε. 26-05-2024

Βεβαιώνω ότι είμαι ο συγγραφέας αυτής της εργασίας και ότι κάθε βοήθεια την οποία είχα για την προετοιμασία της είναι πλήρως αναγνωρισμένη και αναφέρεται στην εργασία. Επίσης, έχω καταγράψει τις όποιες πηγές από τις οποίες έκανα χρήση δεδομένων, ιδεών, εικόνων και κειμένου, είτε αυτές αναφέρονται ακριβώς είτε παραφρασμένες. Επιπλέον, βεβαιώνω ότι αυτή η εργασία προετοιμάστηκε από εμένα προσωπικά, ειδικά ως πτυχιακή εργασία, στο Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του ΔΙ.ΠΑ.Ε.

Αφιέρωση

Θα ήθελα να εκφράσω τις ευχαριστίες μου προς όλους όσους με υποστήριξαν κατά τη διάρκεια της εκπόνησης αυτής της εργασίας, ειδικά προς τον επιβλέποντα καθηγητή μου για τις πολύτιμες συμβουλές και την καθοδήγηση.

Πρόλογος

Η εκπαίδευση αποτελεί έναν από τους πιο κρίσιμους τομείς της σύγχρονης κοινωνίας, επηρεάζοντας άμεσα την ανάπτυξη και την πρόοδο των ατόμων και των κοινωνιών στο σύνολό τους. Με την εξέλιξη της τεχνολογίας, οι εκπαιδευτικοί οργανισμοί καλούνται να προσαρμοστούν σε νέες προκλήσεις και να εκμεταλλευτούν τις καινοτόμες λύσεις που προσφέρει η ψηφιακή εποχή.

Ένας από τους σημαντικότερους τομείς που απαιτούν προσοχή και βελτίωση είναι ο προγραμματισμός των μαθημάτων και η διαχείριση των αιθουσών διδασκαλίας. Οι παραδοσιακές μέθοδοι διαχείρισης, αν και χρησιμοποιούνται ευρέως, συχνά αποδεικνύονται αναποτελεσματικές και ανεπαρκείς για να καλύψουν τις ολοένα και αυξανόμενες ανάγκες των εκπαιδευτικών ιδρυμάτων.

Αυτή η εργασία επικεντρώνεται στην ανάπτυξη και εφαρμογή ενός διαδικτυακού συστήματος προγραμματισμού μαθημάτων και διαχείρισης αιθουσών, που έχει ως στόχο να αυτοματοποιήσει και να βελτιώσει τις υπάρχουσες διαδικασίες. Η λύση που προτείνεται επιδιώκει να διευκολύνει τους καθηγητές στην οργάνωση των μαθημάτων τους, να παρέχει στους φοιτητές εύκολη και άμεση πρόσβαση στις πληροφορίες που χρειάζονται και να βελτιώσει τη συνολική αποδοτικότητα της χρήσης των πόρων του ιδρύματος.

Ελπίζω ότι η εργασία αυτή θα αποτελέσει ένα χρήσιμο εργαλείο για όσους ενδιαφέρονται να βελτιώσουν τις διαδικασίες προγραμματισμού και διαχείρισης σε εκπαιδευτικούς οργανισμούς και θα συμβάλει στην ευρύτερη συζήτηση για τη χρήση της τεχνολογίας στην εκπαίδευση.

Περίληψη

Η παρούσα εργασία ασχολείται με την ανάπτυξη και εφαρμογή ενός διαδικτυακού συστήματος προγραμματισμού μαθημάτων και διαχείρισης αιθουσών για εκπαιδευτικά ιδρύματα. Το σύστημα στοχεύει στην αυτοματοποίηση και βελτίωση των διαδικασιών προγραμματισμού μαθημάτων και διαχείρισης αιθουσών, προσφέροντας μια ολοκληρωμένη λύση για καθηγητές και μαθητές.

Η εφαρμογή επιτρέπει στους καθηγητές να προγραμματίζουν, να τροποποιούν και να παρακολουθούν τα μαθήματα εντός των διαθέσιμων αιθουσών, διασφαλίζοντας ότι οι αλλαγές στο πρόγραμμα ενημερώνονται αυτόματα στους συμμετέχοντες. Οι μαθητές έχουν πρόσβαση σε πραγματικό χρόνο στο ημερολόγιο των μαθημάτων τους, διευκολύνοντας την παρακολούθηση και οργάνωση των ακαδημαϊκών τους υποχρεώσεων.

Η εφαρμογή αξιοποιεί σύγχρονες τεχνολογίες όπως το React για το frontend, το Node.js για το backend, και το Prisma για τη διαχείριση της βάσης δεδομένων. Η χρήση αυτών των εργαλείων διασφαλίζει υψηλή απόδοση, ασφάλεια και ευελιξία, καθιστώντας την εφαρμογή μια αξιόπιστη λύση για την αποτελεσματική διαχείριση των ακαδημαϊκών διαδικασιών.

Συνολικά, η εργασία αυτή καταδεικνύει την αποτελεσματικότητα της χρήσης σύγχρονων τεχνολογιών για την ανάπτυξη διαδικτυακών συστημάτων που βελτιώνουν τις ακαδημαϊκές διαδικασίες και διευκολύνουν τη διαχείριση των εκπαιδευτικών πόρων.

Abstract

This project focuses on the development and implementation of an online scheduling and room management system for educational institutions. The system aims to automate and improve the processes of scheduling courses and room management, offering a comprehensive solution for teachers and students.

The application allows teachers to schedule, modify, and monitor courses within the available rooms, ensuring that any changes to the schedule are automatically updated. Students have real-time access to their course calendar, facilitating the tracking and organization of their academic responsibilities.

The application leverages modern technologies such as React for the frontend, Node.js for the backend, and Prisma for database management. The use of these tools ensures high performance, security, and flexibility, making the application a reliable solution for the efficient management of academic processes.

Overall, this project demonstrates the effectiveness of using modern technologies to develop online systems that enhance academic processes and facilitate the management of educational resources.

Περιεχόμενα

Κεφάλαιο 1

Εισαγωγή

1.1 Περιγραφή του Προβλήματος

Στον σύγχρονο εκπαιδευτικό κόσμο, η αποτελεσματική διαχείριση του προγραμματισμού μαθημάτων και της χρήσης των αιθουσών αποτελεί σημαντική πρόκληση. Εκπαιδευτικά ιδρύματα αντιμετωπίζουν συχνά δυσκολίες στην οργάνωση των μαθημάτων τους, ώστε να εξυπηρετούνται οι ανάγκες τόσο των καθηγητών όσο και των φοιτητών, προσπαθώντας παράλληλα να αξιοποιήσουν τον διαθέσιμο χώρο των αιθουσών με τον πιο αποδοτικό τρόπο. Προκλήσεις όπως οι συγκρούσεις ωραρίων, η διαχείριση των αλλαγών τελευταίας στιγμής και η επικοινωνία των ανακοινώσεων, απαιτούν μια δυναμική και ευέλικτη προσέγγιση.

1.2 Επισκόπηση της Τρέχουσας Κατάστασης και των Υφιστάμενων Λύσεων

Πολλά ιδρύματα συνεχίζουν να βασίζονται σε παραδοσιακές μεθόδους για τη διαχείριση του προγραμματισμού μαθημάτων και των αιθουσών, όπως χειρόγραφα ημερολόγια και ηλεκτρονικά υπολογιστικά φύλλα. Αν και υπάρχουν διαθέσιμες ψηφιακές λύσεις, πολλές από αυτές δεν καλύπτουν πλήρως τις ανάγκες των εκπαιδευτικών οργανισμών ή είναι δύσκολο να προσαρμοστούν στις ειδικές απαιτήσεις κάθε ιδρύματος. Επιπροσθέτως, η έλλειψη ενσωμάτωσης μεταξύ διαφορετικών συστημάτων συχνά προκαλεί προβλήματα συντονισμού και αποδοτικότητας.

1.3 Σκοπός της Εφαρμογής

Η ανάπτυξη και εφαρμογή ενός διαδικτυακού συστήματος προγραμματισμού μαθημάτων και διαχείρισης αιθουσών στοχεύει στην αυτοματοποίηση και βελτίωση των υφιστάμενων διαδικασιών. Η εφαρμογή προσφέρει στους καθηγητές τη δυνατότητα να προγραμματίζουν, τροποποιούν και παρακολουθούν τα μαθήματά τους με ευκολία, ενώ παράλληλα επιτρέπει στους φοιτητές να έχουν πρόσβαση σε πραγματικό χρόνο στο ημερολόγιο των μαθημάτων τους. Αυτό εξασφαλίζει την έγκαιρη και αποτελεσματική ενημέρωση όλων των εμπλεκομένων σε περίπτωση αλλαγών.

1.4 Ανάλυση Αναγκών

Η ανάγκη για ένα ολοκληρωμένο σύστημα διαχείρισης προκύπτει από τις καθημερινές δυσκολίες που αντιμετωπίζουν οι καθηγητές, οι φοιτητές και οι διαχειριστές. Οι καθηγητές χρειάζονται ένα εργαλείο που θα τους επιτρέπει να προγραμματίζουν τα μαθήματά τους εύκολα και γρήγορα, ενώ οι φοιτητές επιζητούν άμεση πρόσβαση σε ενημερωμένα προγράμματα. Οι διαχειριστές, από την πλευρά τους, χρειάζονται ένα σύστημα που θα διευκολύνει τον συντονισμό και την αποτελεσματική διαχείριση των πόρων.

1.5 Στόχοι και Σκοποί

Η εφαρμογή στοχεύει στη δημιουργία ενός εύχρηστου και αποδοτικού συστήματος που θα καλύπτει τις ανάγκες όλων των χρηστών του εκπαιδευτικού ιδρύματος. Συγκεκριμένα, επιδιώκεται η διευκόλυνση του προγραμματισμού μαθημάτων, η μείωση των συγκρούσεων ωραρίων, η βελτίωση της επικοινωνίας και η αποτελεσματική χρήση των αιθουσών.

1.6 Μεθοδολογία

Για την ανάπτυξη της εφαρμογής ακολουθήθηκε μια μεθοδολογία που περιλαμβάνει τη χρήση σύγχρονων τεχνολογιών και εργαλείων, όπως το React για το frontend, το Remix για τη διαχείριση των δεδομένων, το Tailwind CSS για τη σχεδίαση, το Node.js για το backend και το Prisma για τη διαχείριση της βάσης δεδομένων. Η μεθοδολογία περιλαμβάνει επίσης τον έλεγχο και την επικύρωση δεδομένων, καθώς και την ανάπτυξη APIs για την επικοινωνία μεταξύ των διαφόρων συστημάτων.

1.7 Οργανωτική Δομή της Εργασίας

Η εργασία δομείται σε κεφάλαια που αναλύουν τη θεωρία, την ανάπτυξη, την υλοποίηση και τα αποτελέσματα του συστήματος. Το πρώτο κεφάλαιο αποτελεί την εισαγωγή, ενώ τα επόμενα κεφάλαια εμβαθύνουν στη σχεδίαση της βάσης δεδομένων, την ανάπτυξη των APIs, τη διαχείριση των χρηστών και των δικαιωμάτων, καθώς και την ανάλυση των αποτελεσμάτων και τη συζήτηση των συμπερασμάτων.

1.8 Καινοτομίες και Συνεισφορά

Η εφαρμογή εισάγει καινοτομίες στη διαχείριση των ακαδημαϊκών προγραμμάτων και των αιθουσών, όπως η δυνατότητα πραγματικού χρόνου ενημέρωσης. Η συνεισφορά της έρευνας έγκειται στη βελτίωση της αποδοτικότητας και της ευελιξίας των εκπαιδευτικών διαδικασιών, προσφέροντας ένα εργαλείο που ανταποκρίνεται στις σύγχρονες ανάγκες των εκπαιδευτικών ιδρυμάτων.

1.9 Αναμενόμενα Αποτελέσματα

Τα αναμενόμενα αποτελέσματα της εφαρμογής περιλαμβάνουν τη βελτίωση της αποδοτικότητας στη διαχείριση των μαθημάτων και των αιθουσών, την καλύτερη ενημέρωση των φοιτητών και τη διευκόλυνση της διοίκησης των εκπαιδευτικών ιδρυμάτων. Η εφαρμογή αναμένεται να μειώσει τις συγκρούσεις ωραρίων, να αυξήσει την ακρίβεια των προγραμμάτων και να διευκολύνει την επικοινωνία μεταξύ των εμπλεκόμενων μερών.

1.10 Περιορισμοί και Προκλήσεις

Κατά την υλοποίηση και τη χρήση της εφαρμογής, μπορεί να προκύψουν περιορισμοί και προκλήσεις, όπως η ενσωμάτωση με υπάρχοντα συστήματα, η εκπαίδευση των χρηστών και η αντιμετώπιση τεχνικών προβλημάτων. Η επιτυχής αντιμετώπιση αυτών των προκλήσεων είναι κρίσιμη για την ομαλή λειτουργία και την αποδοχή της εφαρμογής από τους χρήστες.

Κεφάλαιο 2

Αρχιτεκτονική και ανάλυση τεχνολογιών του συστήματος

2.1 Απαιτήσεις του συστήματος

2.1.1 Ημερολόγιο και Προγραμματισμός Μαθημάτων

Το σύστημα επιτρέπει στους καθηγητές να προγραμματίζουν, να επεξεργάζονται και να ακυρώνουν μαθήματα, καθώς και η δυνατότητα για τους μαθητές να παρακολουθούν το ημερολόγιο των μαθημάτων τους.

Οι καθηγητές μπορούν να προγραμματίζουν μαθήματα, καθορίζοντας την ημέρα, την ώρα και την αίθουσα διεξαγωγής του μαθήματος. Έχουν τη δυνατότητα να επεξεργάζονται τα προγραμματισμένα μαθήματα, αλλάζοντας την ημερομηνία ή την ώρα διεξαγωγής.

Οι μαθητές έχουν τη δυνατότητα να παρακολουθούν το ημερολόγιο των μαθημάτων τους σε πραγματικό χρόνο, βλέποντας όλες τις προγραμματισμένες ώρες και ημερομηνίες διδασκαλίας. Ενημερώνονται άμεσα για οποιεσδήποτε αλλαγές στο πρόγραμμα των μαθημάτων τους. Επιπλέον, μπορούν να προσαρμόζουν το ημερολόγιό τους, προσθέτοντας προσωπικές σημειώσεις και υπενθυμίσεις για τις ώρες διδασκαλίας.

2.1.2 Διαχείριση Αιθουσών

Η διαχείριση αιθουσών είναι μια κρίσιμη λειτουργία του συστήματος, επιτρέποντας την αποδοτική χρήση των διαθέσιμων πόρων και την εξασφάλιση ότι οι αίθουσες χρησιμοποιούνται κατάλληλα.

Οι καθηγητές μπορούν να προβαίνουν σε κρατήσεις αιθουσών βάσει των απαιτήσεων των μαθημάτων τους, διασφαλίζοντας ότι οι αίθουσες είναι διαθέσιμες κατά τις ώρες διδασκαλίας.

Μπορούν να ελέγχουν τη διαθεσιμότητα των αιθουσών σε πραγματικό χρόνο, λαμβάνοντας υπόψη τη χωρητικότητα και τον εξοπλισμό της κάθε αίθουσας.

Το σύστημα προσφέρει λεπτομερείς πληροφορίες για κάθε αίθουσα, περιλαμβάνοντας τη χωρητικότητα της αίθουσας, δηλαδή τον αριθμό των ατόμων που μπορεί να φιλοξενήσει, τον διαθέσιμο εξοπλισμό, όπως πίνακες, προβολείς, συστήματα ήχου και άλλα εργαλεία διδασκαλίας, καθώς και τη θέση της αίθουσας εντός του εκπαιδευτικού ιδρύματος, βοηθώντας τους καθηγητές και τους μαθητές να την εντοπίζουν εύκολα.

Με αυτές τις δυνατότητες, το σύστημα διασφαλίζει την αποτελεσματική διαχείριση των αιθουσών και τη βελτιστοποίηση της χρήσης των πόρων του εκπαιδευτικού ιδρύματος.

2.1.3 Διαχείριση Χρηστών και Εξουσιοδοτήσεων

Το σύστημα υποστηρίζει πολλαπλούς ρόλους χρηστών, καθένας με διαφορετικό επίπεδο πρόσβασης και δικαιωμάτων. Οι κύριοι ρόλοι περιλαμβάνουν τους καθηγητές, τους μαθητές και τους διαχειριστές.

Οι καθηγητές έχουν τη δυνατότητα να δημιουργούν, να τροποποιούν και να διαγράφουν μαθήματα και γεγονότα. Συγκεκριμένα, οι καθηγητές μπορούν να δημιουργούν νέα μαθήματα και να καθορίζουν τις λεπτομέρειες τους, όπως το όνομα του μαθήματος και το πρόγραμμα διδασκαλίας. Επίσης, μπορούν να τροποποιούν υπάρχοντα μαθήματα, προσθέτοντας ή αφαιρώντας πληροφορίες, αλλάζοντας το πρόγραμμα. Επιπλέον, έχουν τη δυνατότητα να προγραμματίζουν τις ώρες διδασκαλίας και τις ημερομηνίες διεξαγωγής των μαθημάτων.

Οι μαθητές έχουν πρόσβαση στα μαθήματα στα οποία είναι εγγεγραμμένοι και μπορούν να παρακολουθούν το ημερολόγιο των μαθημάτων τους, βλέποντας τις προγραμματισμένες ώρες και ημερομηνίες διδασκαλίας. Ενημερώνονται σε πραγματικό χρόνο για αλλαγές στο πρόγραμμα των μαθημάτων. Επιπλέον, μπορούν να βλέπουν πληροφορίες για τα μαθήματα, όπως περιγραφές, από ποιους καθηγητές διδάσκεται, σε ποιο εξάμηνο διδάσκεται, και άλλες σχετικές πληροφορίες.

Οι διαχειριστές έχουν πλήρη πρόσβαση και δικαιώματα επί του συστήματος, συμπεριλαμβανομένης της δημιουργίας, της διαγραφής και της ενημέρωσης των στοιχείων τους. Επιπλέον, διαχειρίζονται τις αίθουσες διδασκαλίας, καθορίζοντας τις λεπτομέρειες κάθε αίθουσας, όπως η χωρητικότητα και ο εξοπλισμός. Οι διαχειριστές μπορούν επίσης να παρακολουθούν και να τροποποιούν τα προγράμματα διδασκαλίας, διασφαλίζοντας ότι δεν υπάρχουν συγκρούσεις ωραρίων και ότι οι αίθουσες χρησιμοποιούνται αποτελεσματικά.

2.2 Αρχιτεκτονική του συστήματος

Η αρχιτεκτονική του προτεινόμενου συστήματος προγραμματισμού μαθημάτων και διαχείρισης αιθουσών βασίζεται σε ένα μοντέλο τριών στρωμάτων, το οποίο περιλαμβάνει το επί-

πεδο παρουσίασης, το επίπεδο λογικής εφαρμογής και το επίπεδο διαχείρισης δεδομένων. Στο επίπεδο παρουσίασης, χρησιμοποιούνται σύγχρονες τεχνολογίες front-end, όπως, JavaScript frameworks (React.js), για τη δημιουργία μιας διαδραστικής και προσβάσιμης διεπαφής χρήστη. Το επίπεδο λογικής εφαρμογής υλοποιείται μέσω ενός back-end framework (Node.js), το οποίο επιτρέπει την επεξεργασία αιτημάτων, τη διαχείριση δεδομένων και την επικοινωνία με τη βάση δεδομένων. Η βάση δεδομένων, το τρίτο επίπεδο, σχεδιάζεται για την αποθήκευση, ανάκτηση και διαχείριση όλων των απαραίτητων δεδομένων σχετικά με μαθήματα, χρήστες, αίθουσες και προγραμματισμούς.

2.3 Ανάλυση Τεχνολογιών

2.3.1 FrontEnd

React

Η React είναι μια βιβλιοθήκη JavaScript που χρησιμοποιείται για την κατασκευή user interfaces (UI). Αναπτύχθηκε από το Facebook και κυκλοφόρησε για πρώτη φορά το 2013 [1]. Επικεντρώνεται στη δημιουργία component-based αρχιτεκτονικής, που σημαίνει ότι το UI χωρίζεται σε μικρά, ανεξάρτητα και επαναχρησιμοποιήσιμα κομμάτια, τα οποία ονομάζονται components. Κάθε component είναι υπεύθυνο για ένα συγκεκριμένο μέρος του UI και μπορεί να περιέχει τη δική του λογική και στυλ.

Τα χαρακτηριστικά της Node.js συνοψίζονται παρακάτω:

Virtual DOM

Το Virtual DOM είναι μια ελαφριά αναπαράσταση του πραγματικού DOM. Αντί να ενημερώνει απευθείας το πραγματικό DOM με κάθε αλλαγή, το React δημιουργεί ένα εικονικό DOM και υπολογίζει τις διαφορές μεταξύ του εικονικού και του πραγματικού DOM (diffing). Στη συνέχεια, ενημερώνει μόνο τα μέρη του πραγματικού DOM που έχουν αλλάξει [2].

- Βελτιστοποιεί την απόδοση, διότι οι ενημερώσεις στο πραγματικό DOM είναι πιο αργές από τις ενημερώσεις στο Virtual DOM [3].
- Μειώνει τον αριθμό των αναγκαίων DOM ενημερώσεων, κάνοντας την εφαρμογή πιο γρήγορη και αποδοτική.

Component-based Architecture

Το React ενθαρρύνει τη δημιουργία επαναχρησιμοποιήσιμων components. Κάθε component είναι μια αυτόνομη μονάδα που μπορεί να περιέχει τη δική της λογική, στυλ και markup [4].

- Επαναχρησιμοποίηση: Μπορείς να επαναχρησιμοποιήσεις components σε διάφορα μέρη της εφαρμογής σου, μειώνοντας έτσι την ανάγκη για επαναλαμβανόμενο κώδικα.
- Συντήρηση: Είναι πιο εύκολο να διαχειρίζεσαι και να συντηρείς κώδικα που είναι διαχωρισμένος σε μικρά, αυτόνομα κομμάτια.
- Ανεξαρτησία: Κάθε component μπορεί να αναπτυχθεί, να δοκιμαστεί και να ενσωματωθεί ανεξάρτητα από τα άλλα.

JSX (JavaScript XML)

Το JSX είναι μια σύνταξη που επιτρέπει τη χρήση HTML-like markup μέσα σε JavaScript. Αν και δεν είναι απαραίτητο να χρησιμοποιήσεις JSX, είναι ευρέως χρησιμοποιούμενο στο React λόγω της αναγνωσιμότητας και της ευκολίας που προσφέρει [5].

- Αναγνωσιμότητα: Κάνει τον κώδικα πιο ευανάγνωστο και ευκολότερο στην κατανόηση, καθώς βλέπεις το layout και τη λογική μαζί.
- Προσαρμοστικότητα: Επειδή είναι JavaScript, μπορείς να χρησιμοποιήσεις τη δυναμική φύση της γλώσσας για να χειρίζεσαι το markup.

State and Props

- **State:** Είναι μια εσωτερική κατάσταση που μπορεί να αλλάξει και να επηρεάσει τη συμπεριφορά του component. Κάθε component μπορεί να έχει το δικό του state [6].
- **Props:** Είναι ιδιότητες που μεταφέρονται από το γονικό component στο παιδικό component. Τα props είναι αμετάβλητα και χρησιμοποιούνται για τη μετάδοση δεδομένων και λειτουργιών.

Lifecycle Methods

Το React παρέχει μια σειρά από μεθόδους που εκτελούνται σε διαφορετικά στάδια του κύκλου ζωής ενός component (π.χ., όταν δημιουργείται, όταν ενημερώνεται, όταν καταστρέφεται) [7].

- Έλεγχος: Επιτρέπουν την εκτέλεση κώδικα σε συγκεκριμένα σημεία του κύκλου ζωής ενός component, παρέχοντας περισσότερο έλεγχο πάνω στην εφαρμογή.

Hooks

Τα Hooks είναι ειδικές λειτουργίες που εισήχθησαν στο React 16.8 και επιτρέπουν τη χρήση state και άλλων χαρακτηριστικών του React χωρίς να χρειάζεται να γράψεις κλάσεις [8].

- Απλότητα: Κάνουν τα functional components πιο ισχυρά και ευέλικτα.
- Κοινή λογική: Επιτρέπουν την επαναχρησιμοποίηση λογικής μεταξύ διαφορετικών components χωρίς την ανάγκη για HOCs (Higher Order Components) ή render props.

Μεγάλη Κοινότητα και Υποστήριξη

Το React έχει μια από τις μεγαλύτερες κοινότητες στον κόσμο του web development, με πληθώρα διαθέσιμων πόρων, όπως βιβλιοθήκες, εργαλεία, tutorials, και forums [9].

- Βοήθεια: Εύκολη πρόσβαση σε υποστήριξη και συμβουλές από άλλους προγραμματιστές.
- Εργαλεία και βιβλιοθήκες: Πληθώρα εργαλείων και βιβλιοθηκών που μπορούν να βοηθήσουν στην επιτάχυνση της ανάπτυξης και την προσθήκη προηγμένων λειτουργιών στην εφαρμογή σου.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του React. Συνθέτουν μια ισχυρή και ευέλικτη πλατφόρμα για την ανάπτυξη μοντέρνων web εφαρμογών.

Remix

Το Remix είναι ένα σύγχρονο web framework βασισμένο στο React, σχεδιασμένο για την κατασκευή full-stack web εφαρμογών. Αναπτύχθηκε από τους δημιουργούς του React Router και επικεντρώνεται στην απόδοση και την εμπειρία χρήστη [10]. Το Remix προσφέρει ισχυρά εργαλεία για τη διαχείριση του routing, τη φόρτωση δεδομένων και την απόδοση της εφαρμογής.

Server-side Rendering (SSR): Το SSR επιτρέπει στον server να renderάρει το React component tree και να στείλει το HTML στον πελάτη πριν φορτωθεί το JavaScript. Αυτό βελτιώνει την αρχική απόδοση και τη SEO [11].

- Γρήγορη αρχική φόρτωση: Ο χρήστης βλέπει περιεχόμενο γρηγορότερα καθώς δεν χρειάζεται να περιμένει να φορτωθεί και να εκτελεστεί το JavaScript.
- SEO βελτιστοποίηση: Οι μηχανές αναζήτησης μπορούν να ανιχνεύσουν και να ευρετηριάσουν το περιεχόμενο πιο αποτελεσματικά.

Static Site Generation (SSG): Το SSG δημιουργεί στατικό HTML κατά τη διαδικασία build, που μπορεί να σερβίρεται απευθείας από ένα CDN. Είναι ιδανικό για σελίδες που δεν αλλάζουν συχνά.

- Υψηλή απόδοση: Οι στατικές σελίδες φορτώνουν εξαιρετικά γρήγορα από το CDN.
- Ασφάλεια: Μειωμένος κίνδυνος επιθέσεων, καθώς δεν υπάρχουν δυναμικά backend scripts που μπορούν να εκτελεστούν.

Ενσωματωμένη διαχείριση routing

Το Remix χρησιμοποιεί τον React Router για τη διαχείριση των routes. Κάθε route αντιστοιχεί σε ένα component και μπορεί να έχει δική του λογική και φόρτωση δεδομένων [12].

- Απλότητα: Η διαχείριση των routes γίνεται απευθείας στο component tree, κάνοντας την αρχιτεκτονική του κώδικα πιο απλή και ευανάγνωστη.
- Συνεπής λογική: Ενσωματωμένη διαχείριση της κατάστασης του URL, δυναμικά segments, nested routes, κ.λπ.

Βελτιστοποιημένη φόρτωση δεδομένων

Το Remix επιτρέπει τη φόρτωση δεδομένων σε επίπεδο routes, γεγονός που βοηθά στην αποφυγή περιττών requests και στη βελτίωση της απόδοσης [13].

- Data pre-fetching: Το Remix μπορεί να φορτώνει δεδομένα πριν το component εμφανιστεί στο UI, διασφαλίζοντας ότι όλα είναι έτοιμα όταν το χρήστης φτάνει στη σελίδα.
- Data caching: Μηχανισμοί caching που μειώνουν τα αιτήματα στο server και βελτιώνουν την απόδοση.

Εξαιρετική εμπειρία προγραμματιστή

Το Remix προσφέρει εργαλεία και χαρακτηριστικά που κάνουν την ανάπτυξη εφαρμογών πιο ευχάριστη και αποτελεσματική [14].

- Hot Module Replacement (HMR): Άμεσες ενημερώσεις στον browser καθώς γράφετε κώδικα, χωρίς να χάνετε την κατάσταση της εφαρμογής.
- Λιγότερος boilerplate κώδικας: Η ενσωμάτωση πολλών λειτουργιών σε ένα ενιαίο framework μειώνει την ανάγκη για πρόσθετες βιβλιοθήκες και κώδικα.

Progressive Enhancement

Το Remix εφαρμόζει την αρχή του progressive enhancement, που σημαίνει ότι οι βασικές λειτουργίες της εφαρμογής πρέπει να λειτουργούν ανεξάρτητα από το JavaScript [15].

- Ανθεκτικότητα: Η εφαρμογή παραμένει λειτουργική ακόμα και αν το JavaScript αποτύχει να φορτωθεί.

- Προσβασιμότητα: Βελτιώνει την προσβασιμότητα για χρήστες με περιορισμένες δυνατότητες ή παλαιότερες συσκευές.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του Remix, που συνθέτουν ένα ισχυρό και ευέλικτο πλαίσιο για την ανάπτυξη σύγχρονων web εφαρμογών με έμφαση στην απόδοση, την εμπειρία χρήστη και την ευκολία ανάπτυξης.

Tailwind CSS

Το Tailwind CSS είναι ένα utility-first CSS framework που επιτρέπει τη γρήγορη και αποτελεσματική δημιουργία προσαρμοσμένων σχεδίων. Σε αντίθεση με τα παραδοσιακά CSS frameworks που προσφέρουν προδιαμορφωμένα components, το Tailwind προσφέρει μικρές, χαμηλού επιπέδου CSS classes (utilities) που μπορούν να συνδυαστούν για τη δημιουργία σύνθετων σχεδίων απευθείας στο HTML [16].

Χρήση classes για στυλιζάρισμα απευθείας στο HTML

Το Tailwind παρέχει μια πληθώρα από μικρές, χαμηλού επιπέδου CSS classes που μπορούν να χρησιμοποιηθούν απευθείας στο HTML για να εφαρμόσουν συγκεκριμένα στυλ σε στοιχεία.

- Ταχύτητα ανάπτυξης: Δεν χρειάζεται να γράψετε ξεχωριστά CSS αρχεία, καθώς το styling γίνεται απευθείας στο HTML με την προσθήκη των κατάλληλων classes [17].
- Συνοχή σχεδίου: Η χρήση προκαθορισμένων classes εξασφαλίζει ότι το στυλ παραμένει συνεπές σε όλη την εφαρμογή.
- Αποφυγή CSS υπερφόρτωσης: Δεν υπάρχει ανάγκη για την εγγραφή επαναλαμβανόμενων CSS κανόνων, μειώνοντας έτσι το μέγεθος του CSS.

Εξαιρετικά προσαρμόσιμο

Το Tailwind είναι εξαιρετικά προσαρμόσιμο, επιτρέποντάς σας να προσαρμόσετε ή να επεκτείνετε τα default styles του framework [18].

- Θέματα (Themes): Μπορείτε να δημιουργήσετε και να εφαρμόσετε custom themes που προσαρμόζουν χρώματα, γραμματοσειρές και άλλες παραμέτρους για να ταιριάζουν στις ανάγκες της εφαρμογής σας.
- Configuration: Το tailwind.config.js αρχείο σας επιτρέπει να προσαρμόσετε τα default breakpoints, spacing, colors, κ.λπ., προσφέροντας πλήρη έλεγχο πάνω στο σχεδιασμό.
- Plugins: Υπάρχει δυνατότητα για προσθήκη custom plugins που μπορούν να επεκτείνουν τη λειτουργικότητα του Tailwind με νέες utilities ή components.

Καλές επιδόσεις και μικρό μέγεθος

Το Tailwind CSS είναι σχεδιασμένο για να παράγει μικρά και αποδοτικά CSS αρχεία, ειδικά όταν χρησιμοποιείται σε συνδυασμό με εργαλεία όπως το PurgeCSS [19].

- PurgeCSS Integration: Το Tailwind μπορεί να συνδυαστεί με το PurgeCSS για να αφαιρέσει όλες τις αχρησιμοποίητες CSS classes από την παραγωγή build, μειώνοντας έτσι δραματικά το μέγεθος του τελικού CSS αρχείου.
- Απόδοση: Τα μικρά μεγέθη CSS αρχείων οδηγούν σε ταχύτερους χρόνους φόρτωσης της σελίδας, βελτιώνοντας την απόδοση και την εμπειρία χρήστη.
- Ευελιξία: Ακόμα και με πολλά utilities, το τελικό μέγεθος του CSS μπορεί να διατηρηθεί μικρό και αποδοτικό.

Utility-First Approach

Το utility-first σημαίνει ότι το Tailwind προσφέρει χαμηλού επιπέδου classes που εφαρμόζουν ένα μόνο στυλ, όπως p-4 για padding ή text-center για κεντράρισμα κειμένου [20].

- Γρήγορη ανάπτυξη: Η χρήση utility classes επιτρέπει την γρήγορη και εύκολη προσθήκη στυλ σε HTML, χωρίς την ανάγκη για γραφή custom CSS.
- Μικρότερη πολυπλοκότητα: Η προσέγγιση αυτή μειώνει την πολυπλοκότητα και την αλληλεπίδραση μεταξύ διαφορετικών CSS classes.
- Προβλεψιμότητα: Κάθε class έχει συγκεκριμένο και προβλέψιμο αποτέλεσμα, διευκολύνοντας την κατανόηση και τη συντήρηση του κώδικα.

Responsive Design

Το Tailwind ενσωματώνει breakpoints για responsive design απευθείας στις utility classes του [21].

- Ευκολία στη χρήση: Μπορείτε να προσθέσετε responsive styles απλά προσθέτοντας prefixes όπως sm:, md:, lg: , πριν από οποιαδήποτε utility class.
- Συνεκτική responsive σχεδίαση: Οι responsive classes επιτρέπουν τη διατήρηση ενός συνεκτικού και προβλέψιμου responsive σχεδίου σε όλη την εφαρμογή.

Πλούσιο οικοσύστημα και κοινότητα

Το Tailwind έχει αναπτύξει μια μεγάλη κοινότητα και ένα πλούσιο οικοσύστημα εργαλείων και plugins [22].

- Υποστήριξη και πόροι: Υπάρχουν άφθονα tutorials, άρθρα και εργαλεία που μπορούν να βοηθήσουν στην επίλυση προβλημάτων και στην επιτάχυνση της ανάπτυξης.
- Επέκταση λειτουργιών: Πολλά διαθέσιμα plugins που προσθέτουν νέες δυνατότητες και utilities στο Tailwind, διευκολύνοντας την εφαρμογή συγκεκριμένων στυλιστικών απαιτήσεων.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του Tailwind CSS. Παρέχει ένα ισχυρό και ευέλικτο εργαλείο για τη γρήγορη ανάπτυξη προσαρμοσμένων και αποδοτικών web σχεδίων.

2.3.2 Backend

Node.js

Το Node.js είναι ένα runtime περιβάλλον για την εκτέλεση JavaScript στον server. Αναπτύχθηκε από τον Ryan Dahl το 2009 και βασίζεται στη μηχανή JavaScript V8 της Google, η οποία χρησιμοποιείται επίσης στο Google Chrome. Η κύρια καινοτομία του Node.js είναι η χρήση ενός event-driven, non-blocking I/O μοντέλου, που το καθιστά ιδανικό για την κατασκευή κλιμακούμενων και υψηλής απόδοσης εφαρμογών [23].

Event-driven και non-blocking I/O

Το Node.js χρησιμοποιεί ένα event-driven αρχιτεκτονικό πρότυπο, όπου οι λειτουργίες εκτελούνται ως αντιδράσεις σε συμβάντα. Επιπλέον, το non-blocking I/O σημαίνει ότι οι λειτουργίες εισόδου/εξόδου (I/O) δεν μπλοκάρουν την εκτέλεση άλλου κώδικα. Αντίθετα, όταν μια λειτουργία ολοκληρωθεί, ένα callback (ή promise) εκτελείται για να χειριστεί το αποτέλεσμα [24].

- Υψηλή απόδοση: Οι λειτουργίες I/O μπορούν να εκτελούνται παράλληλα, αυξάνοντας την απόδοση της εφαρμογής.
- Κλιμάκωση: Επειδή το μοντέλο είναι event-driven, το Node.js μπορεί να διαχειριστεί μεγάλο αριθμό ταυτόχρονων συνδέσεων με λίγους πόρους.

Μεγάλη κοινότητα και πλούσιο οικοσύστημα πακέτων (npm)

Το npm (Node Package Manager) είναι το προεπιλεγμένο εργαλείο διαχείρισης πακέτων για το Node.js, παρέχοντας πρόσβαση σε εκατοντάδες χιλιάδες πακέτα ανοικτού κώδικα που μπορούν να χρησιμοποιηθούν για την ανάπτυξη εφαρμογών [25].

- Ευκολία ανάπτυξης: Μπορείτε να βρείτε και να ενσωματώσετε έτοιμες λύσεις για σχεδόν οποιοδήποτε πρόβλημα ή λειτουργία.
- Υποστήριξη: Η μεγάλη κοινότητα προσφέρει άφθονους πόρους, όπως βιβλιοθήκες, εργαλεία, tutorials και forums.
- Συντήρηση: Η συνεχής ενημέρωση και ανάπτυξη των πακέτων από την κοινότητα εξασφαλίζει ότι μπορείτε να έχετε πρόσβαση σε σύγχρονες και ασφαλείς βιβλιοθήκες.

Υψηλή απόδοση για εφαρμογές που απαιτούν real-time λειτουργικότητα

Το Node.js είναι ιδιαίτερα κατάλληλο για εφαρμογές που απαιτούν real-time λειτουργικότητα, όπως chat εφαρμογές, multiplayer games, ή εφαρμογές που χρειάζονται ταυτόχρονη επεξεργασία δεδομένων [26].

- WebSockets: Το Node.js υποστηρίζει WebSockets, επιτρέποντας τη δημιουργία διαρκών συνδέσεων μεταξύ client και server, που είναι απαραίτητες για real-time επικοινωνία.
- Αμεσότητα: Το event-driven μοντέλο επιτρέπει άμεση απόκριση στα αιτήματα των χρηστών, μειώνοντας την καθυστέρηση και βελτιώνοντας την εμπειρία χρήστη.
- Streaming Data: Το Node.js μπορεί να χειριστεί streaming δεδομένα πολύ αποτελεσματικά, κάτι που είναι κρίσιμο για εφαρμογές όπως οι video και audio streaming υπηρεσίες.

Ενιαία γλώσσα για server και client

Με το Node.js, μπορείτε να χρησιμοποιήσετε JavaScript τόσο στον client (frontend) όσο και στον server (backend), επιτρέποντας την επαναχρησιμοποίηση κώδικα και κοινή λογική μεταξύ των δύο μερών της εφαρμογής [27].

- Κοινή γνώση: Οι προγραμματιστές μπορούν να εργαστούν τόσο στο frontend όσο και στο backend, διευκολύνοντας τη συνεργασία και μειώνοντας την ανάγκη για ειδικούς σε διαφορετικές γλώσσες.
- Κοινή λογική: Μπορείτε να μοιράζεστε λειτουργίες και utilities μεταξύ server και client, μειώνοντας την πολυπλοκότητα και τον χρόνο ανάπτυξης.

Microservices και API Development

Το Node.js είναι ιδανικό για την ανάπτυξη microservices και APIs, λόγω της ευελιξίας και της δυνατότητας εύκολης κλιμάκωσης [28].

- Κλιμακούμενη αρχιτεκτονική: Η δυνατότητα δημιουργίας μικρών, ανεξάρτητων υπηρεσιών που μπορούν να αναπτυχθούν και να κλιμακωθούν ανεξάρτητα.
- RESTful και GraphQL APIs: Υπάρχουν πολλές βιβλιοθήκες και εργαλεία για τη δημιουργία και διαχείριση APIs, όπως το Express για RESTful APIs και το Apollo για GraphQL.

Ευρεία υποστήριξη πλατφορμών και εργαλείων

Το Node.js υποστηρίζεται από πολλές πλατφόρμες και εργαλεία ανάπτυξης, κάνοντας το εύκολα προσαρμόσιμο σε διαφορετικά περιβάλλοντα και διαδικασίες ανάπτυξης [29].

- DevOps και CI/CD: Υποστήριξη για εργαλεία DevOps και συνεχούς ολοκλήρωσης/παράδοσης (CI/CD), όπως το Docker, Kubernetes, Jenkins, και άλλα.
- Hosting και Deployment: Πολλές πλατφόρμες φιλοξενίας υποστηρίζουν το Node.js, όπως το Heroku, AWS, Google Cloud, και Azure, διευκολύνοντας την ανάπτυξη και τη διαχείριση εφαρμογών.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του Node.js. Παρέχει ένα ισχυρό και ευέλικτο περιβάλλον για την ανάπτυξη κλιμακούμενων, υψηλής απόδοσης και real-time εφαρμογών.

Prisma

Το Prisma είναι ένα σύγχρονο ORM (Object-Relational Mapping) εργαλείο για την αλληλεπίδραση με βάσεις δεδομένων. Σχεδιάστηκε για να βοηθήσει τους προγραμματιστές να γράφουν ασφαλή και αποδοτική βάση δεδομένων κώδικα, χρησιμοποιώντας έναν type-safe τρόπο. Το Prisma λειτουργεί ως ένα layer μεταξύ της εφαρμογής σας και της βάσης δεδομένων, παρέχοντας ένα ισχυρό API για τη διαχείριση και την πρόσβαση στα δεδομένα [30].

Type-safe queries

Το Prisma παρέχει ένα API που ενσωματώνει type safety. Αυτό σημαίνει ότι οι ερωτήσεις προς τη βάση δεδομένων είναι type-checked κατά τη διάρκεια της ανάπτυξης, μειώνοντας τα πιθανά runtime errors [31].

- Ασφάλεια τύπων: Βοηθά στην αποφυγή σφαλμάτων που σχετίζονται με τύπους δεδομένων, καθώς οι ερωτήσεις ελέγχονται κατά τη σύνταξη.
- Αναγνωσιμότητα κώδικα: Βελτιώνει την αναγνωσιμότητα και την ανιχνευσιμότητα του κώδικα, διευκολύνοντας την κατανόηση και τη συντήρηση.
- Autocompletion: Παρέχει autocompletion στα IDEs, κάνοντας τη συγγραφή κώδικα ταχύτερη και με λιγότερα λάθη.

Υποστήριξη για πολλαπλές βάσεις δεδομένων

Το Prisma υποστηρίζει πολλαπλές βάσεις δεδομένων, όπως PostgreSQL, MySQL, SQLite, SQL Server, και MongoDB [32].

- Ευελιξία: Σας επιτρέπει να επιλέξετε τη βάση δεδομένων που ταιριάζει καλύτερα στις ανάγκες της εφαρμογής σας.
- Μεταφορά δεδομένων: Διευκολύνει τη μετάβαση από μία βάση δεδομένων σε άλλη, εάν οι ανάγκες της εφαρμογής σας αλλάξουν με την πάροδο του χρόνου.
- Ενιαίο API: Παρέχει ένα ενιαίο API για την αλληλεπίδραση με διαφορετικούς τύπους βάσεων δεδομένων, μειώνοντας την πολυπλοκότητα του κώδικα.

Εύκολη διαχείριση μεταναστεύσεων (migrations)

Το Prisma παρέχει εργαλεία για τη διαχείριση μεταναστεύσεων της βάσης δεδομένων, διευκολύνοντας την ενημέρωση του schema της βάσης δεδομένων με ασφαλή και συντονισμένο τρόπο [33].

- Αυτοματοποιημένες μεταναστεύσεις: Δημιουργεί αυτόματα αρχεία μεταναστεύσεων από αλλαγές στο schema, μειώνοντας τον κίνδυνο λαθών και την ανάγκη για χειροκίνητες διαδικασίες.
- Ιστορικό μεταναστεύσεων: Καταγράφει το ιστορικό των αλλαγών στη βάση δεδομένων, επιτρέποντας την παρακολούθηση και την επαναφορά σε προηγούμενες εκδόσεις αν χρειαστεί.
- Συνεργασία: Επιτρέπει σε ομάδες ανάπτυξης να συνεργάζονται καλύτερα, συντονίζοντας τις αλλαγές στη βάση δεδομένων.

Prisma Client

Το Prisma Client είναι μια αυτόματα παραγόμενη και type-safe βιβλιοθήκη που δημιουργείται από το Prisma schema. Παρέχει ένα ισχυρό API για την εκτέλεση ερωτήσεων και μεταλλάξεων στη βάση δεδομένων [34].

- Ευκολία χρήσης: Παρέχει έναν εύκολο τρόπο για την εκτέλεση ερωτήσεων χωρίς την ανάγκη να γράψετε πολύπλοκες SQL queries.
- Type safety: Ενσωματώνει type safety σε κάθε βήμα της αλληλεπίδρασης με τη βάση δεδομένων, διασφαλίζοντας ότι οι ερωτήσεις σας είναι σωστές.
- Απόδοση: Βελτιστοποιεί τις ερωτήσεις για απόδοση, επιτρέποντας ταχύτερη πρόσβαση στα δεδομένα.

Prisma Studio

Το Prisma Studio είναι ένα GUI εργαλείο που σας επιτρέπει να διαχειρίζεστε και να εξερευνάτε τα δεδομένα σας εύκολα [35].

- Ευκολία διαχείρισης: Παρέχει μια γραφική διεπαφή για την προβολή και επεξεργασία των δεδομένων σας, καθιστώντας τη διαχείριση των δεδομένων πιο διαισθητική.
- Αναλυτικά δεδομένα: Διευκολύνει την αναζήτηση και την ανάλυση των δεδομένων σας χωρίς να χρειάζεται να γράψετε SQL queries.
- Χρήσιμο για debugging: Βοηθά στον έλεγχο και την αποσφαλμάτωση των δεδομένων σας κατά τη διάρκεια της ανάπτυξης.

Ενσωμάτωση με TypeScript

Το Prisma σχεδιάστηκε για να ενσωματώνεται άψογα με TypeScript, παρέχοντας πλήρη υποστήριξη για types και διασφαλίζοντας type safety σε όλο το application [36].

- Type inference: Το Prisma μπορεί να παρέχει type inference για τα μοντέλα σας, καθιστώντας την ανάπτυξη ταχύτερη και πιο ασφαλή.
- Βελτιωμένη εμπειρία ανάπτυξης: Η χρήση TypeScript με Prisma βελτιώνει την εμπειρία ανάπτυξης με καλύτερο autocompletion και λιγότερα σφάλματα.

API και Υποστήριξη για GraphQL

Το Prisma μπορεί να χρησιμοποιηθεί ως ένα layer μεταξύ της εφαρμογής σας και της βάσης δεδομένων σε GraphQL εφαρμογές [37].

- Ευκολία στη δημιουργία GraphQL resolvers: Μπορείτε να δημιουργήσετε GraphQL resolvers που εκτελούν type-safe queries στη βάση δεδομένων μέσω του Prisma Client.
- Ευελιξία: Μπορεί να χρησιμοποιηθεί με οποιοδήποτε GraphQL server, παρέχοντας ευελιξία στον σχεδιασμό της αρχιτεκτονικής σας.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του Prisma. Παρέχει ένα ισχυρό εργαλείο για την αλληλεπίδραση με βάσεις δεδομένων, βελτιώνοντας την ασφάλεια, την απόδοση και την ευκολία διαχείρισης των δεδομένων σας.

2.3.3 Βοηθητικές βιβλιοθήκες και εργαλεία

TypeScript

TypeScript είναι μια υπερσύνολο της JavaScript που προσθέτει στατική τυποποίηση (types). Αναπτύχθηκε από τη Microsoft και κυκλοφόρησε για πρώτη φορά το 2012. Το TypeScript συνδυάζει την ευελιξία της JavaScript με την ισχύ της στατικής ανάλυσης τύπων, βελτιώνοντας την εμπειρία ανάπτυξης και τη συντήρηση του κώδικα [38].

Type safety και δυνατότητα στατικής ανάλυσης κώδικα

Το TypeScript προσθέτει στατικούς τύπους στη γλώσσα JavaScript. Αυτό σημαίνει ότι μπορείτε να δηλώσετε τους τύπους των μεταβλητών, των παραμέτρων συναρτήσεων, των επιστρεφόμενων τιμών και των ιδιοτήτων αντικειμένων [39].

- Type safety: Ο κώδικας ελέγχεται κατά τη διάρκεια της σύνταξης (compile time) για συμβατότητα τύπων, μειώνοντας τον αριθμό των runtime σφαλμάτων.
- Στατική ανάλυση: Παρέχει δυνατότητες στατικής ανάλυσης που βοηθούν στον εντοπισμό σφαλμάτων πριν την εκτέλεση του κώδικα, όπως ασυμβατότητες τύπων και μη δηλωμένες μεταβλητές.
- Αυτοσυμπλήρωση (Autocompletion): Τα IDEs που υποστηρίζουν TypeScript μπορούν να προσφέρουν βελτιωμένη αυτοσυμπλήρωση και τεκμηρίωση, καθιστώντας την ανάπτυξη ταχύτερη και πιο αποδοτική.

Καλύτερη συντήρηση και ανιχνευσιμότητα σφαλμάτων

Με τη χρήση τύπων, ο κώδικας γίνεται πιο σαφής και ευανάγνωστος, διευκολύνοντας τη συντήρηση και την ανιχνευσιμότητα σφαλμάτων [40].

- Καθαρότητα κώδικα: Η σαφήνεια στους τύπους καθιστά τον κώδικα πιο κατανοητό και ευκολότερο στη συντήρηση, ειδικά σε μεγάλα έργα.
- Εργαλεία ανίχνευσης σφαλμάτων: Τα εργαλεία ανάπτυξης μπορούν να εντοπίζουν σφάλματα και ασυνέπειες τύπων κατά τη διάρκεια της ανάπτυξης, διευκολύνοντας την επίλυση προβλημάτων.
- Refactoring: Η δυνατότητα για ασφαλές refactoring είναι ενισχυμένη, καθώς η τυποποίηση επιτρέπει την αναγνώριση όλων των σημείων που επηρεάζονται από μια αλλαγή.

Ευκολία στη συνεργασία σε μεγάλες ομάδες ανάπτυξης

Το TypeScript βοηθά στη συνεργασία μεταξύ προγραμματιστών, ειδικά σε μεγάλες ομάδες, διευκολύνοντας την κατανομή εργασιών και τη διαχείριση του κώδικα [41].

- Κοινή βάση: Οι τύποι λειτουργούν ως κοινή βάση που περιγράφει τις προσδοκίες και τις απαιτήσεις του κώδικα, διευκολύνοντας την κατανόηση μεταξύ των προγραμματιστών.
- Σαφής τεκμηρίωση: Ο τύπος των δεδομένων λειτουργεί ως τεκμηρίωση που περιγράφει τον τρόπο χρήσης των συναρτήσεων και των αντικειμένων.
- Αποφυγή συγκρούσεων: Οι στατικοί τύποι βοηθούν στην αποφυγή συγκρούσεων και ασυμβατότητας κώδικα όταν πολλές ομάδες εργάζονται στον ίδιο κώδικα.

Ενσωμάτωση με υπάρχοντα JavaScript έργα

Το TypeScript είναι πλήρως συμβατό με την JavaScript και μπορεί να ενσωματωθεί σε υπάρχοντα JavaScript έργα σταδιακά [42].

- Σταδιακή υιοθέτηση: Μπορείτε να ξεκινήσετε την υιοθέτηση του TypeScript σταδιακά, προσθέτοντας τύπους σε νέα αρχεία ή αναβαθμίζοντας υπάρχοντα αρχεία JavaScript σε αρχεία TypeScript (.ts).
- Συμβατότητα: Το TypeScript μπορεί να χρησιμοποιηθεί παράλληλα με την JavaScript, επιτρέποντας την ομαλή μετάβαση χωρίς να διακόπτεται η ανάπτυξη.

Προηγμένα χαρακτηριστικά γλώσσας

Το TypeScript προσφέρει προηγμένα χαρακτηριστικά γλώσσας που δεν είναι διαθέσιμα στην κανονική JavaScript, όπως interfaces, generics, και enums [43].

- Interfaces: Τα interfaces επιτρέπουν τον ορισμό των δομών των αντικειμένων, βελτιώνοντας την αναγνωσιμότητα και την ασφάλεια τύπων.
- Generics: Τα generics επιτρέπουν τη δημιουργία επαναχρησιμοποιήσιμων συναρτήσεων και κλάσεων που μπορούν να λειτουργούν με διάφορους τύπους δεδομένων, προσφέροντας ευελιξία και επαναχρησιμοποίηση κώδικα.
- Enums: Τα enums παρέχουν έναν τρόπο για να ορίσετε σύνολα ονομαστικών σταθερών, βελτιώνοντας την καθαρότητα και την αναγνωσιμότητα του κώδικα.

Εργαλεία ανάπτυξης και υποστήριξη IDE

Το TypeScript υποστηρίζεται από μια πληθώρα εργαλείων ανάπτυξης και IDEs, όπως Visual Studio Code, WebStorm, και άλλες [44].

- Autocompletion και Intellisense: Τα εργαλεία ανάπτυξης παρέχουν ισχυρή υποστήριξη για autocompletion και Intellisense, κάνοντας την ανάπτυξη πιο παραγωγική.
- Debugging: Τα IDEs προσφέρουν προηγμένα εργαλεία debugging για TypeScript, διευκολύνοντας την ανίχνευση και την επίλυση σφαλμάτων.
- Refactoring εργαλεία: Υπάρχουν εργαλεία που υποστηρίζουν ασφαλές refactoring για TypeScript, βελτιώνοντας τη συντηρησιμότητα του κώδικα.

Αυτά είναι μερικά από τα κύρια χαρακτηριστικά και οφέλη του TypeScript. Προσφέρει μια ισχυρή λύση για την ανάπτυξη σύγχρονων web εφαρμογών με βελτιωμένη ασφάλεια, συντηρησιμότητα και συνεργασία.

Κεφάλαιο 3

Σχεδιασμός της βάσης δεδομένων

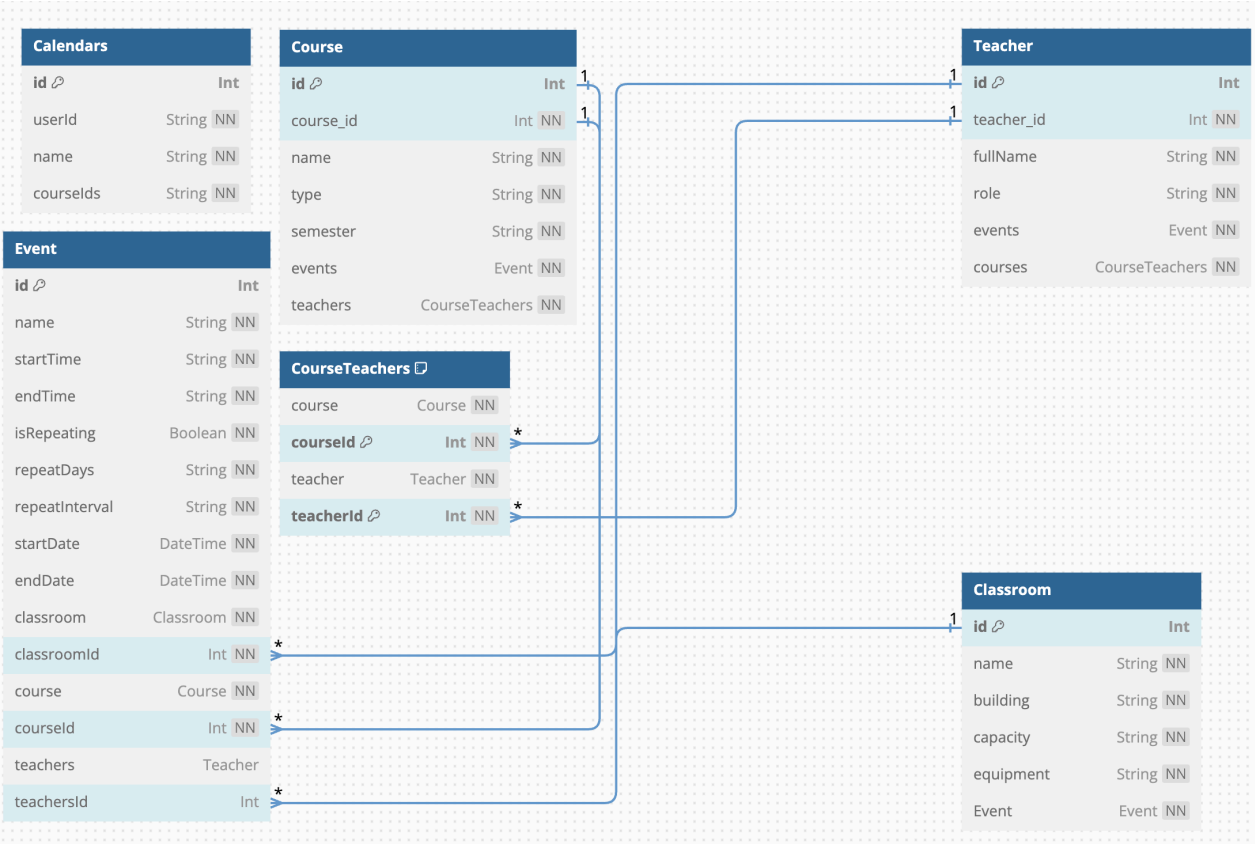
3.1 Δομή της βάσης δεδομένων

Ο σχεδιασμός της βάσης δεδομένων για την εφαρμογή αποτελεί ένα κρίσιμο βήμα για την επίτευξη αποδοτικής και αποτελεσματικής διαχείρισης των εκπαιδευτικών πληροφοριών. Η βάση δεδομένων έχει σχεδιαστεί για να καταγράφει, να οργανώνει και να διαχειρίζεται δεδομένα που σχετίζονται με εκπαιδευτικές δραστηριότητες, μαθήματα, καθηγητές, αίθουσες και προγράμματα σπουδών. Το σχήμα της βάσης δεδομένων περιλαμβάνει διάφορους πίνακες που αλληλεπιδρούν μεταξύ τους μέσω προσεκτικά καθορισμένων σχέσεων, διασφαλίζοντας την ακεραιότητα και τη συνέπεια των δεδομένων.

Η δομή του σχήματος 3.1 περιλαμβάνει τους πίνακες `Event`, `Classroom`, `Course`, `Teacher`, `CourseTeachers` και `Calendars`. Κάθε πίνακας έχει σχεδιαστεί για να εξυπηρετεί συγκεκριμένες λειτουργίες και να καταγράφει σχετικές πληροφορίες, όπως τα γεγονότα, τις αίθουσες διδασκαλίας, τα μαθήματα, τους καθηγητές και τα προγράμματα σπουδών. Οι σχέσεις μεταξύ των πινάκων έχουν οριστεί με τρόπο που να επιτρέπει την πολυδιάστατη αναπαράσταση των δεδομένων, διευκολύνοντας τη δημιουργία σύνθετων ερωτημάτων και την απόδοση των απαραίτητων πληροφοριών με ακρίβεια και ταχύτητα. Με αυτόν τον τρόπο, το σχήμα της βάσης δεδομένων υποστηρίζει τις ανάγκες της εφαρμογής μας και εξασφαλίζει τη βέλτιστη λειτουργία της.

3.2 Ανάλυση Πίνακα ”Course” και Λειτουργία του στην Εφαρμογή

Ο πίνακας `Course` σχεδιάστηκε για να καταγράφει και να διαχειρίζεται τις πληροφορίες που σχετίζονται με τα μαθήματα σε μία βάση δεδομένων. Ο πίνακας περιέχει διάφορα πεδία που επιτρέπουν την καταγραφή βασικών πληροφοριών για κάθε μάθημα, καθώς και την αλληλεπίδρασή του με τους πίνακες `Event` και `CourseTeachers`, οι οποίοι καταγράφουν τα



Σχήμα 3.1: Schema

γεγονότα που αφορούν το μάθημα και τους καθηγητές που το διδάσκουν, αντίστοιχα.

3.2.1 Ανάλυση Στηλών του Πίνακα

- **id (Int)**: Χρησιμοποιείται ως πρωτεύον κλειδί για τον πίνακα. Η τιμή του είναι μοναδική για κάθε εγγραφή και ορίζεται αυτόματα.
- **course_id (Int)**: Αποθηκεύει το μοναδικό αναγνωριστικό του μαθήματος. Το πεδίο αυτό είναι μοναδικό για κάθε μάθημα (`@unique`).
- **name (String)**: Αποθηκεύει το όνομα του μαθήματος, π.χ. ”Δομημένος Προγραμματισμός” ή ” Αντικειμενοστρεφής Προγραμματισμός”.
- **type (String)**: Καταγράφει τον τύπο του μαθήματος, όπως ”Θεωρία” ή ”Εργαστήριο”.
- **semester (String)**: Αποθηκεύει το εξάμηνο στο οποίο προσφέρεται το μάθημα, π.χ. ”Εαρινό” ή ”Χειμερινό”.

3.2.2 Λειτουργία στην Εφαρμογή

Ο πίνακας `Course` επιτρέπει την καταγραφή και διαχείριση των μαθημάτων σε μία βάση δεδομένων. Η δομή του πίνακα επιτρέπει την οργάνωση των μαθημάτων με λεπτομέρεια, συμπεριλαμβανομένων των ονομάτων, των τύπων, των εξαμήνων, και των σχέσεών τους με τα γεγονότα και τους καθηγητές. Αυτό παρέχει τη δυνατότητα ανάπτυξης εφαρμογών που προσφέρουν λεπτομερή διαχείριση των μαθημάτων και των σχετικών εκπαιδευτικών δραστηριοτήτων.

3.2.3 Σχεδιαστικές Αποφάσεις

Οι σχεδιαστικές αποφάσεις πίσω από τη δομή του πίνακα βασίστηκαν στην ανάγκη για αποδοτικότητα και ευελιξία της βάσης δεδομένων. Η χρήση του `Int` ως πρωτεύον κλειδί επιτρέπει την εύκολη και γρήγορη αναζήτηση εγγραφών. Η χρήση του `String` για την αποθήκευση των ονομάτων, των τύπων και των εξαμήνων επιτρέπει την αναπαράσταση των δεδομένων με ακρίβεια. Το πεδίο `course_id` διασφαλίζει τη μοναδικότητα κάθε μαθήματος.

3.2.4 Χρήση των Σχέσεων

Η σχέση με τον πίνακα `Event` επιτρέπει την πολυδιάστατη αναπαράσταση των δεδομένων. Η σχέση ένας προς πολλούς (one-to-many) επιτρέπει την καταγραφή πολλών γεγονότων για κάθε μάθημα, διευκολύνοντας την οργάνωση και τη διαχείριση των εκπαιδευτικών δραστηριοτήτων που αφορούν το μάθημα. Παρομοίως, η σχέση με τον πίνακα `CourseTeachers` επιτρέπει την καταγραφή πολλών καθηγητών για κάθε μάθημα, εξασφαλίζοντας την κατανομή των καθηγητικών πόρων.

3.2.5 Διαχείριση Μαθημάτων και Εξαμήνων

Η στήλη `semester` επιτρέπει την αποθήκευση του εξαμήνου στο οποίο προσφέρεται το μάθημα, γεγονός που βοηθά στην κατανομή και την οργάνωση του προγράμματος σπουδών. Η στήλη `type` επιτρέπει την καταγραφή του τύπου του μαθήματος, διευκολύνοντας την αναγνώριση των θεωρητικών και εργαστηριακών μαθημάτων.

3.2.6 Βελτιστοποίηση και Επεκτασιμότητα

Για μελλοντικές βελτιώσεις και δυνατότητες επεκτασιμότητας του πίνακα, μπορούν να προστεθούν επιπλέον πεδία για την καταγραφή άλλων σχετικών πληροφοριών, όπως οι προαπαιτούμενες γνώσεις ή οι στόχοι του μαθήματος.

3.3 Ανάλυση Πίνακα "Teacher" και Λειτουργία του στην Εφαρμογή

Ο πίνακας `Teacher` σχεδιάστηκε για να καταγράφει και να διαχειρίζεται τις πληροφορίες που σχετίζονται με τους καθηγητές σε μία βάση δεδομένων. Ο πίνακας περιέχει διάφορα πεδία που επιτρέπουν την καταγραφή βασικών πληροφοριών για κάθε καθηγητή, καθώς και την αλληλεπίδρασή του με τους πίνακες `Event` και `CourseTeachers`, οι οποίοι καταγράφουν τα γεγονότα που αφορούν τον καθηγητή και τα μαθήματα που διδάσκει, αντίστοιχα.

3.3.1 Ανάλυση Στηλών του Πίνακα

- **id (Int)**: Χρησιμοποιείται ως πρωτεύον κλειδί για τον πίνακα. Η τιμή του είναι μοναδική για κάθε εγγραφή και ορίζεται αυτόματα.
- **teacher_id (Int)**: Αποθηκεύει το μοναδικό αναγνωριστικό του καθηγητή. Το πεδίο αυτό είναι μοναδικό για κάθε καθηγητή (`@unique`).
- **fullName (String)**: Αποθηκεύει το πλήρες όνομα του καθηγητή, π.χ. "Γεώργιος Παπαδόπουλος".
- **role (String)**: Καταγράφει τον ρόλο του καθηγητή, όπως "Καθηγητής", "Επίκουρος Καθηγητής" ή "Λέκτορας".

3.3.2 Λειτουργία στην Εφαρμογή

Ο πίνακας `Teacher` επιτρέπει την καταγραφή και διαχείριση των καθηγητών σε μία βάση δεδομένων. Η δομή του πίνακα επιτρέπει την οργάνωση των καθηγητών με λεπτομέρεια, συμπεριλαμβανομένων των ονομάτων, των ρόλων, και των σχέσεών τους με τα γεγονότα και τα μαθήματα. Αυτό παρέχει τη δυνατότητα ανάπτυξης εφαρμογών που προσφέρουν λεπτομερή διαχείριση των καθηγητών και των σχετικών εκπαιδευτικών δραστηριοτήτων.

3.3.3 Σχεδιαστικές Αποφάσεις

Οι σχεδιαστικές αποφάσεις πίσω από τη δομή του πίνακα βασίστηκαν στην ανάγκη για αποδοτικότητα και ευελιξία της βάσης δεδομένων. Η χρήση του `Int` ως πρωτεύον κλειδί επιτρέπει την εύκολη και γρήγορη αναζήτηση εγγραφών. Η χρήση του `String` για την αποθήκευση των ονομάτων και των ρόλων επιτρέπει την αναπαράσταση των δεδομένων με ακρίβεια. Το πεδίο `teacher_id` διασφαλίζει τη μοναδικότητα κάθε καθηγητή.

3.3.4 Χρήση των Σχέσεων

Η σχέση με τον πίνακα `Event` επιτρέπει την πολυδιάστατη αναπαράσταση των δεδομένων. Η σχέση ένας προς πολλούς (one-to-many) επιτρέπει την καταγραφή πολλών γεγονότων για κάθε καθηγητή, διευκολύνοντας την οργάνωση και τη διαχείριση των εκπαιδευτικών δραστηριοτήτων που αφορούν τον καθηγητή. Παρομοίως, η σχέση με τον πίνακα `CourseTeachers` επιτρέπει την καταγραφή πολλών μαθημάτων που διδάσκει κάθε καθηγητής, εξασφαλίζοντας την κατανομή των καθηγητικών πόρων.

3.3.5 Διαχείριση Καθηγητών και Ρόλων

Η στήλη `role` επιτρέπει την αποθήκευση του ρόλου του καθηγητή, γεγονός που βοηθά στην κατανομή και την οργάνωση των καθηγητικών αρμοδιοτήτων. Η στήλη `fullName` επιτρέπει την καταγραφή του πλήρους ονόματος του καθηγητή, διευκολύνοντας την αναγνώριση και την οργάνωση των καθηγητών.

3.3.6 Βελτιστοποίηση και Επεκτασιμότητα

Για μελλοντικές βελτιώσεις και δυνατότητες επεκτασιμότητας του πίνακα, μπορούν να προστεθούν επιπλέον πεδία για την καταγραφή άλλων σχετικών πληροφοριών, όπως οι ειδικότητες του καθηγητή.

3.4 Ανάλυση Πίνακα ”Classroom” και Λειτουργία του στην Εφαρμογή

Ο πίνακας `Classroom` σχεδιάστηκε για να καταγράφει και να διαχειρίζεται τις πληροφορίες που σχετίζονται με τις αίθουσες διδασκαλίας σε μία βάση δεδομένων. Ο πίνακας περιέχει διάφορα πεδία που επιτρέπουν την καταγραφή βασικών πληροφοριών για κάθε αίθουσα, καθώς και την αλληλεπίδρασή του με τον πίνακα `Event`, ο οποίος καταγράφει τα γεγονότα που λαμβάνουν χώρα σε κάθε αίθουσα.

3.4.1 Ανάλυση Στηλών του Πίνακα

- **id (Int)**: Χρησιμοποιείται ως πρωτεύον κλειδί για τον πίνακα. Η τιμή του είναι μοναδική για κάθε εγγραφή και ορίζεται αυτόματα.
- **name (String)**: Αποθηκεύει το όνομα της αίθουσας. Το πεδίο αυτό είναι μοναδικό για κάθε αίθουσα (`@unique`).

- **building (String)**: Καταγράφει το κτίριο στο οποίο βρίσκεται η αίθουσα. Παράδειγμα: ”Κτίριο Α”.
- **capacity (String)**: Αποθηκεύει τη χωρητικότητα της αίθουσας, π.χ. ”30 άτομα”.
- **equipment (String)**: Καταγράφει τον εξοπλισμό που διαθέτει η αίθουσα, όπως ”προβολέας, πίνακας”.

3.4.2 Λειτουργία στην Εφαρμογή

Ο πίνακας `Classroom` επιτρέπει την καταγραφή και διαχείριση των αιθουσών διδασκαλίας σε μία βάση δεδομένων. Η δομή του πίνακα επιτρέπει την οργάνωση των αιθουσών με λεπτομέρεια, συμπεριλαμβανομένων των ονομάτων, των κτιρίων, των χωρητικοτήτων, και του εξοπλισμού τους. Αυτό παρέχει τη δυνατότητα ανάπτυξης εφαρμογών που προσφέρουν λεπτομερή διαχείριση των αιθουσών και των διαθέσιμων πόρων.

3.4.3 Σχεδιαστικές Αποφάσεις

Οι σχεδιαστικές αποφάσεις πίσω από τη δομή του πίνακα βασίστηκαν στην ανάγκη για αποδοτικότητα και ευελιξία της βάσης δεδομένων. Η χρήση του `Int` ως πρωτεύον κλειδί επιτρέπει την εύκολη και γρήγορη αναζήτηση εγγραφών. Η χρήση του `String` για την αποθήκευση των ονομάτων, των κτιρίων, της χωρητικότητας και του εξοπλισμού επιτρέπει την αναπαράσταση των δεδομένων με ακρίβεια.

3.4.4 Χρήση των Σχέσεων

Η σχέση με τον πίνακα `Event` επιτρέπει την πολυδιάστατη αναπαράσταση των δεδομένων. Η σχέση ένας προς πολλούς (one-to-many) επιτρέπει την καταγραφή πολλών γεγονότων για κάθε αίθουσα, διευκολύνοντας την οργάνωση και τη διαχείριση των εκπαιδευτικών δραστηριοτήτων που λαμβάνουν χώρα σε κάθε αίθουσα.

3.4.5 Διαχείριση Χωρητικότητας και Εξοπλισμού

Η στήλη `capacity` επιτρέπει την αποθήκευση της χωρητικότητας της αίθουσας, γεγονός που βοηθά στην κατανομή των μαθημάτων και των διαλέξεων ανάλογα με τον αριθμό των μαθητών. Η στήλη `equipment` επιτρέπει την καταγραφή του διαθέσιμου εξοπλισμού, διευκολύνοντας τη διαχείριση και την προετοιμασία των μαθημάτων που απαιτούν συγκεκριμένο εξοπλισμό.

3.4.6 Βελτιστοποίηση και Επεκτασιμότητα

Για μελλοντικές βελτιώσεις και δυνατότητες επεκτασιμότητας του πίνακα, μπορούν να προστεθούν επιπλέον πεδία για την καταγραφή άλλων σχετικών πληροφοριών, όπως η τοποθεσία εντός του κτιρίου ή ειδικές ανάγκες εξοπλισμού.

3.5 Ανάλυση Πίνακα "Event" και Λειτουργία του στην Εφαρμογή

Ο πίνακας `Event` σχεδιάστηκε για να καταγράφει και να διαχειρίζεται τα γεγονότα που σχετίζονται με εκπαιδευτικές δραστηριότητες, όπως μαθήματα ή διαλέξεις, σε μία βάση δεδομένων. Ο πίνακας περιέχει διάφορα πεδία που επιτρέπουν την καταγραφή βασικών πληροφοριών για κάθε γεγονός, καθώς και την αλληλεπίδρασή του με άλλους πίνακες, όπως τις αίθουσες και τα μαθήματα.

3.5.1 Ανάλυση Στηλών του Πίνακα

- **id (Int)**: Χρησιμοποιείται ως πρωτεύον κλειδί για τον πίνακα. Η τιμή του είναι μοναδική για κάθε εγγραφή και ορίζεται αυτόματα.
- **name (String)**: Αποθηκεύει το όνομα του γεγονότος, π.χ. "Αναπλήρωση στο μάθημα Γλώσσες και Τεχνολογίες Ιστού" ή "Διάλεξη Μαθηματικά ΙΙ".
- **startTime (String)**: Καταγράφει την ώρα έναρξης του γεγονότος σε μορφή κειμένου. Παράδειγμα: "08:00".
- **endTime (String)**: Καταγράφει την ώρα λήξης του γεγονότος σε μορφή κειμένου. Παράδειγμα: "10:00".
- **isRepeating (Boolean)**: Δηλώνει αν το γεγονός είναι επαναλαμβανόμενο ή όχι. Η τιμή μπορεί να είναι `true` ή `false`.
- **repeatInterval (String)**: Καταγράφει το διάστημα επανάληψης του γεγονότος. Παράδειγμα: "κάθε εβδομάδα".
- **startDate (DateTime)**: Αποθηκεύει την ημερομηνία έναρξης του γεγονότος σε μορφή `DateTime`.
- **endDate (DateTime)**: Αποθηκεύει την ημερομηνία λήξης του γεγονότος σε μορφή `DateTime`.
- **classroom (Classroom)**: Ορίζει τη σχέση του γεγονότος με την αίθουσα στην οποία θα πραγματοποιηθεί. Η σχέση αυτή ορίζεται μέσω του πεδίου `classroomId`.

- **classroomId (Int)**: Αποθηκεύει το μοναδικό αναγνωριστικό της αίθουσας στην οποία θα πραγματοποιηθεί το γεγονός.
- **course (Course)**: Ορίζει τη σχέση του γεγονότος με το μάθημα που αφορά. Η σχέση αυτή ορίζεται μέσω του πεδίου `courseId`.
- **courseId (Int)**: Αποθηκεύει το μοναδικό αναγνωριστικό του μαθήματος που αφορά το γεγονός.
- **teachers (Teacher)**: Ορίζει τη σχέση του γεγονότος με τον καθηγητή ή τους καθηγητές που θα το διδάξουν. Η σχέση αυτή ορίζεται μέσω του πεδίου `teachersId`.
- **teachersId (Int)**: Αποθηκεύει το μοναδικό αναγνωριστικό του καθηγητή ή των καθηγητών που θα διδάξουν το μάθημα. Το πεδίο μπορεί να είναι κενό (`nullable`).
- **excludedDates (String)**: Αποθηκεύει τις ημερομηνίες που το γεγονός δεν θα πραγματοποιηθεί, σε μορφή κειμένου. Παράδειγμα: `["2022-01-01", "2022-01-02"]`.

3.5.2 Λειτουργία στην Εφαρμογή

Ο πίνακας `Event` επιτρέπει την καταγραφή και διαχείριση όλων των εκπαιδευτικών γεγονότων σε μία βάση δεδομένων. Η δομή του πίνακα επιτρέπει την οργάνωση των μαθημάτων ή διαλέξεων με λεπτομέρεια, συμπεριλαμβανομένων των ωρών, των ημερομηνιών, των επαναλήψεων, και των σχέσεων με τις αίθουσες, τα μαθήματα, και τους καθηγητές. Αυτό παρέχει τη δυνατότητα ανάπτυξης εφαρμογών που προσφέρουν λεπτομερή προγράμματα, εξοικονόμηση χρόνου και καλύτερη διαχείριση των πόρων και των ανθρώπων.

3.5.3 Σχεδιαστικές Αποφάσεις

Οι σχεδιαστικές αποφάσεις πίσω από τη δομή του πίνακα βασίστηκαν στην ανάγκη για αποδοτικότητα και ευελιξία της βάσης δεδομένων. Η χρήση του `Int` ως πρωτεύον κλειδί επιτρέπει την εύκολη και γρήγορη αναζήτηση εγγραφών. Η χρήση των τύπων `String` και `DateTime` επιτρέπει την αναπαράσταση των ωρών και των ημερομηνιών με ακρίβεια. Τα πεδία που σχετίζονται με τις επαναλήψεις (`isRepeating`, `repeatInterval`) επιτρέπουν την εύκολη διαχείριση των επαναλαμβανόμενων γεγονότων.

3.5.4 Χρήση των Σχέσεων

Οι σχέσεις με άλλους πίνακες (`Classroom`, `Course`, `Teacher`) επιτρέπουν την πολυδιάστατη αναπαράσταση των δεδομένων. Η σχέση με τον πίνακα `Classroom` επιτρέπει την καταγραφή της αίθουσας όπου θα πραγματοποιηθεί το γεγονός, ενώ η σχέση με τον πίνακα

`Course` επιτρέπει την καταγραφή του μαθήματος που αφορά το γεγονός. Η σχέση με τον πίνακα `Teacher` επιτρέπει την καταγραφή του καθηγητή ή των καθηγητών που θα διδάξουν το μάθημα.

3.5.5 Επαναλαμβανόμενα Γεγονότα

Η διαχείριση των επαναλαμβανόμενων γεγονότων γίνεται μέσω των πεδίων `isRepeating`, και `repeatInterval`. Αυτά τα πεδία επιτρέπουν την αναπαράσταση των επαναλαμβανόμενων γεγονότων με ευελιξία και ακρίβεια. Για παράδειγμα, αν ένα μάθημα πραγματοποιείται κάθε Δευτέρα το πεδίο `repeatInterval` θα ορίζει το διάστημα επανάληψης.

3.5.6 Εξαιρούμενες Ημερομηνίες

Η στήλη `excludedDates` επιτρέπει την αποθήκευση των ημερομηνιών που το γεγονός δεν θα πραγματοποιηθεί. Αυτό είναι ιδιαίτερα χρήσιμο σε περιπτώσεις αργιών ή ειδικών περιστάσεων, όπου το κανονικό πρόγραμμα των μαθημάτων ή των διαλέξεων μπορεί να αλλάξει. Η αποθήκευση αυτών των ημερομηνιών επιτρέπει την ευέλικτη διαχείριση των γεγονότων και την αποφυγή σύγχυσης.

3.5.7 Βελτιστοποίηση και Επεκτασιμότητα

Για μελλοντικές βελτιώσεις και δυνατότητες επεκτασιμότητας του πίνακα, μπορούν να προστεθούν επιπλέον πεδία για την καταγραφή άλλων σχετικών πληροφοριών.

3.6 Ανάλυση Πίνακα ”Calendars” και Λειτουργία του στην Εφαρμογή

Ο πίνακας `Calendars` σχεδιάστηκε για να καταγράφει και να διαχειρίζεται τις πληροφορίες που σχετίζονται με τα προγράμματα σπουδών (calendars) σε μία βάση δεδομένων. Ο πίνακας περιέχει διάφορα πεδία που επιτρέπουν την καταγραφή βασικών πληροφοριών για κάθε πρόγραμμα, καθώς και την αλληλεπίδρασή του με τους χρήστες και τα μαθήματα.

3.6.1 Ανάλυση Στηλών του Πίνακα

- **id (Int)**: Χρησιμοποιείται ως πρωτεύον κλειδί για τον πίνακα. Η τιμή του είναι μοναδική για κάθε εγγραφή και ορίζεται αυτόματα.

- **userId (String)**: Αποθηκεύει το αναγνωριστικό του χρήστη που δημιούργησε το πρόγραμμα. Παράδειγμα: ”it175135”.
- **name (String)**: Αποθηκεύει το όνομα του προγράμματος, π.χ. ”Πρόγραμμα Σπουδών Χειμερινό Εξάμηνο”.
- **courseIds (String)**: Καταγράφει τα αναγνωριστικά των μαθημάτων που περιλαμβάνονται στο πρόγραμμα, σε μορφή κειμένου. Παράδειγμα: ”1,2,3,4”.

3.6.2 Λειτουργία στην Εφαρμογή

Ο πίνακας `Calendars` επιτρέπει την καταγραφή και διαχείριση των προγραμμάτων σπουδών σε μία βάση δεδομένων. Η δομή του πίνακα επιτρέπει την οργάνωση των προγραμμάτων με λεπτομέρεια, συμπεριλαμβανομένων των ονομάτων, των χρηστών που τα δημιούργησαν, και των μαθημάτων που περιλαμβάνουν. Αυτό παρέχει τη δυνατότητα ανάπτυξης εφαρμογών που προσφέρουν λεπτομερή διαχείριση των προγραμμάτων σπουδών και των σχετικών εκπαιδευτικών δραστηριοτήτων.

3.6.3 Σχεδιαστικές Αποφάσεις

Οι σχεδιαστικές αποφάσεις πίσω από τη δομή του πίνακα βασίστηκαν στην ανάγκη για αποδοτικότητα και ευελιξία της βάσης δεδομένων. Η χρήση του `Int` ως πρωτεύον κλειδί επιτρέπει την εύκολη και γρήγορη αναζήτηση εγγραφών. Η χρήση του `String` για την αποθήκευση των αναγνωριστικών χρηστών και των μαθημάτων επιτρέπει την αναπαράσταση των δεδομένων με ακρίβεια.

3.6.4 Χρήση των Σχέσεων

Η στήλη `userId` επιτρέπει τη σύνδεση κάθε προγράμματος με τον χρήστη που το δημιούργησε, διευκολύνοντας την καταγραφή και την οργάνωση των προγραμμάτων σπουδών ανά χρήστη. Η στήλη `courseIds` επιτρέπει την αποθήκευση των αναγνωριστικών των μαθημάτων που περιλαμβάνονται στο πρόγραμμα, διευκολύνοντας την οργάνωση των μαθημάτων και την αναγνώριση των προγραμμάτων που περιλαμβάνουν συγκεκριμένα μαθήματα.

3.6.5 Διαχείριση Προγραμμάτων Σπουδών

Η στήλη `name` επιτρέπει την αποθήκευση του ονόματος του προγράμματος, γεγονός που βοηθά στην κατανομή και την οργάνωση των προγραμμάτων σπουδών. Η στήλη `courseIds` επιτρέπει την καταγραφή των μαθημάτων που περιλαμβάνονται στο πρόγραμμα, διευκολύνοντας την αναγνώριση και τη διαχείριση των μαθημάτων.

Κεφάλαιο 4

Σχεδιασμός των APIs

Σε αυτό το κεφάλαιο, θα παρουσιάσουμε τον σχεδιασμό των διαφόρων API endpoints που χρησιμοποιούνται στην εφαρμογή για τη διαχείριση γεγονότων, μαθημάτων, αιθουσών και καθηγητών. Τα API endpoints έχουν σχεδιαστεί για να παρέχουν ασφαλή και αποδοτική πρόσβαση στα δεδομένα, επιτρέποντας στους διαχειριστές να διαχειρίζονται εύκολα και γρήγορα τις πληροφορίες του εκπαιδευτικού ιδρύματος.

Τα API endpoints διακρίνονται σε κατηγορίες ανάλογα με τις λειτουργίες που παρέχουν, όπως η δημιουργία, η ενημέρωση, η διαγραφή και η ανάκτηση δεδομένων. Κάθε endpoint συνοδεύεται από λεπτομερή περιγραφή της λειτουργίας του, των παραμέτρων που απαιτούνται και της μορφής της απόκρισης. Επιπλέον, γίνεται αναφορά στις διαδικασίες αυθεντικοποίησης και ελέγχου δικαιωμάτων που εξασφαλίζουν ότι μόνο εξουσιοδοτημένοι χρήστες έχουν πρόσβαση στις αντίστοιχες λειτουργίες.

Endpoint	**Method**
/admin/events	GET
/admin/events/create	POST
/event/:id/update	POST
/admin/events/upload	POST
/event/:id/delete	DELETE
/admin/events/delete	DELETE
/admin/courses	GET
/admin/courses/create	POST
/admin/courses/:id/update	POST
/admin/courses/upload	POST
/admin/courses/:id/delete	DELETE
/admin/courses/delete	DELETE
/admin/classrooms	GET
/admin/classrooms/create	POST
/admin/classrooms/:id/update	POST
/admin/classrooms/upload	POST
/admin/classrooms/:id/delete	DELETE
/admin/classrooms/delete	DELETE
/admin/teachers	GET
/admin/teachers/create	POST
/admin/teachers/:id/update	POST
/admin/teachers/upload	POST
/admin/teachers/:id/delete	DELETE
/admin/teachers/delete	DELETE

4.1 Event

4.1.1 Λήψη Γεγονότων, Μαθημάτων και Αιθουσών

Path	`/admin/events`
Method	`GET`
Request Parameters	Δεν απαιτούνται παράμετροι αιτήματος για αυτό το API.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	JSON αντικείμενο που περιέχει τις πληροφορίες για όλα τα γεγονότα, τα μαθήματα, τις αίθουσες, τα μαθήματα οργανωμένα κατά εξάμηνο, και τις επαναλαμβανόμενες επιλογές.

Το συγκεκριμένο API χρησιμοποιείται για την ανάκτηση όλων των γεγονότων, των μαθημάτων, των αιθουσών και των επιλογών επαναλαμβανόμενων γεγονότων από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε αυτές τις πληροφορίες:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να έχουν πρόσβαση στις πληροφορίες για τα γεγονότα, τα μαθήματα και τις αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Ανάκτηση Γεγονότων, Μαθημάτων και Αιθουσών**: Όλα τα γεγονότα, τα μαθήματα και οι αίθουσες ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Στα μαθήματα περιλαμβάνονται επίσης οι σχέσεις τους με τους καθηγητές.

4. **Οργάνωση Μαθημάτων κατά Εξάμηνο**: Τα μαθήματα ομαδοποιούνται κατά εξάμηνο και δημιουργείται ένα JSON αντικείμενο που περιέχει τις οργανωμένες πληροφορίες.

5. **Προσθήκη Επαναλαμβανόμενων Επιλογών**: Οι επιλογές επαναλαμβανόμενων γεγονότων (π.χ. Daily, Weekly) προστίθενται στο JSON αντικείμενο.

6. **Απόκριση**: Το JSON αντικείμενο επιστρέφεται ως απάντηση.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ανάκτηση πληροφοριών σχετικά με τα γεγονότα, τα μαθήματα και τις αίθουσες από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να δουν όλες τις πληροφορίες για τα γεγονότα, τα μαθήματα και τις αίθουσες, οργανωμένες κατά εξάμηνο, διευκολύνοντας έτσι τη διαχείριση και την κατανομή των πόρων.

4.1.2 Δημιουργία Γεγονότος

Path	`/admin/events/create`
Method	`POST`
Request Body	`name` (string): Το όνομα του γεγονότος. `courseId` (number): Το μοναδικό αναγνωριστικό του μαθήματος που συνδέεται με το γεγονός. `classroomId` (number): Το μοναδικό αναγνωριστικό της αίθουσας. `startTime` (string): Η ώρα έναρξης του γεγονότος. `endTime` (string): Η ώρα λήξης του γεγονότος. `startDate` (string): Η ημερομηνία έναρξης του γεγονότος. `endDate` (string): Η ημερομηνία λήξης του γεγονότος. `repeatInterval` (string): Ο τύπος επανάληψης (π.χ. none, daily, weekly).
Response	Επιτυχής δημιουργία του γεγονότος με μήνυμα επιβεβαίωσης ή μηνύματα σφαλμάτων.

Το συγκεκριμένο API χρησιμοποιείται για τη δημιουργία ενός νέου γεγονότος στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να δημιουργήσουν γεγονότα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`name`, `courseId`, `classroomId`, `startTime`, `endTime`, `startDate`, `endDate`, `repeat`).

4. **Έλεγχος Υπάρχοντος Μαθήματος**: Πραγματοποιείται έλεγχος για να διασφαλιστεί ότι το μάθημα που σχετίζεται με το γεγονός υπάρχει στη βάση δεδομένων. Αν το μάθημα δεν βρεθεί, επιστρέφεται μήνυμα σφάλματος.

5. **Υπολογισμός Ώρας Έναρξης και Λήξης**: Οι ώρες έναρξης και λήξης μετατρέπονται σε αντικείμενα ημερομηνίας και ώρας για περαιτέρω επεξεργασία.

6. **Έλεγχος Διαθεσιμότητας Αίθουσας**: Ελέγχεται αν η αίθουσα είναι διαθέσιμη για το προγραμματισμένο χρονικό διάστημα. Αν υπάρχουν συγκρούσεις, επιστρέφεται μήνυμα σφάλματος.

7. **Δημιουργία του Γεγονότος**: Το νέο γεγονός δημιουργείται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Περιλαμβάνει τη σύνδεση με το αντίστοιχο μάθημα και την αίθουσα.

8. **Απόκριση**: Επιστρέφεται μήνυμα επιβεβαίωσης για την επιτυχή δημιουργία του γεγονότος ή μηνύματα σφαλμάτων αν παρουσιαστούν προβλήματα.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη δημιουργία νέων γεγονότων στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να προσθέτουν νέα γεγονότα, διασφαλίζοντας ότι οι πληροφορίες για τα γεγονότα είναι πάντα ενημερωμένες.

4.1.3 Ενημέρωση Γεγονότος

Path	`/event/:id/update`
Method	`POST`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του γεγονότος που πρόκειται να ενημερωθεί.
RRequest Body	`name` (string): Το όνομα του γεγονότος. `courseId` (number): Το μοναδικό αναγνωριστικό του μαθήματος που συνδέεται με το γεγονός. `classroomId` (number): Το μοναδικό αναγνωριστικό της αίθουσας. `startTime` (string): Η ώρα έναρξης του γεγονότος. `endTime` (string): Η ώρα λήξης του γεγονότος. `startDate` (string): Η ημερομηνία έναρξης του γεγονότος. `endDate` (string): Η ημερομηνία λήξης του γεγονότος. `repeatInterval` (string): Ο τύπος επανάληψης (π.χ. none, daily, weekly).
Response	Ανακατεύθυνση στην αρχική σελίδα `/`.

Το συγκεκριμένο API χρησιμοποιείται για την ενημέρωση των πληροφοριών ενός γεγονότος στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) ή προσωπικού (`eduPersonAffiliation === "staff"`) επιτρέπεται να ενημερώσουν γεγονότα. Αν ο χρήστης δεν έχει τα απαιτούμενα δικαιώματα, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του γεγονότος (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία.

5. **Ενημέρωση του Γεγονότος**: Το γεγονός ενημερώνεται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Αν το μάθημα που σχετίζεται με το γεγονός δεν υπάρχει, επιστρέφεται σφάλμα.

6. **Απόκριση**: Ανακατεύθυνση στην αρχική σελίδα (`/?index`).

Το συγκεκριμένο API χρησιμοποιείται από διαχειριστές και μέλη του προσωπικού της εφαρμογής για την ενημέρωση των πληροφοριών γεγονότων στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με τα απαραίτητα δικαιώματα και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους εξουσιοδοτημένους χρήστες να διατηρούν ενημερωμένες τις πληροφορίες για τα γεγονότα, όπως το όνομα, τις ώρες, τις ημερομηνίες και τις επαναλήψεις τους.

4.1.4 Ανέβασμα Γεγονότων από Αρχείο CSV

Path	`/admin/events/upload`
Method	`POST`
Request Body	Ένα αρχείο CSV που περιέχει τις πληροφορίες για τα γεγονότα που θα ανέβουν στη βάση δεδομένων. Κάθε γραμμή του CSV πρέπει να περιέχει τα πεδία `name`, `startTime`, `endTime`, `isRepeating`, `repeatInterval`, `startDate`, `endDate`, `course_id`, και `classroom_name`.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για την μαζική εισαγωγή γεγονότων στη βάση δεδομένων από ένα αρχείο CSV. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ανεβάσουν το αρχείο CSV. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Διαχείριση Αρχείου CSV**: Το αρχείο CSV αναλύεται και οι πληροφορίες που περιέχει διαβάζονται γραμμή προς γραμμή. Χρησιμοποιείται η βιβλιοθήκη `csv` για την επεξεργασία των δεδομένων. Για κάθε γραμμή του αρχείου, το γεγονός δημιουργείται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την μαζική εισαγωγή γεγονότων στη βάση δεδομένων από ένα αρχείο CSV. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να εισάγουν γεγονότα με μεγαλύτερη ευκολία και αποτελεσματικότητα.

4.1.5 Διαγραφή Γεγονότος

Path	`/event/:id/delete`
Method	`DELETE`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του γεγονότος που πρόκειται να διαγραφεί.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Ανακατεύθυνση στην αρχική σελίδα `/`.

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή ενός γεγονότος από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) ή προσωπικού (`eduPersonAffiliation === "staff"`) επιτρέπεται να διαγράψουν γεγονότα. Αν ο χρήστης δεν έχει τα απαιτούμενα δικαιώματα, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του γεγονότος (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Διαγραφή του Γεγονότος**: Το γεγονός διαγράφεται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

5. **Απόκριση**: Ανακατεύθυνση στην αρχική σελίδα (`/?index`).

Το συγκεκριμένο API χρησιμοποιείται από διαχειριστές και μέλη του προσωπικού της εφαρμογής για τη διαγραφή γεγονότων από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με τα απαραίτητα δικαιώματα και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους εξουσιοδοτημένους χρήστες να διαχειρίζονται τα γεγονότα, διασφαλίζοντας ότι οι πληροφορίες είναι πάντα ενημερωμένες και ακριβείς.

4.1.6 Διαγραφή Όλων των Γεγονότων

Path	`/admin/events/delete`
Method	`DELETE`
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή όλων των γεγονότων (events) από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν όλα τα γεγονότα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Διαγραφή Όλων των Γεγονότων**: Όλα τα γεγονότα διαγράφονται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή όλων των γεγονότων από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν όλα τα γεγονότα, πιθανώς για επανασχεδιασμό της δομής της βάσης δεδομένων ή για εκκαθάριση των δεδομένων.

4.2 Μαθήματα

4.2.1 Λήψη Μαθημάτων και Καθηγητών

Path	`/admin/courses`
Method	`GET`
Request Parameters	Δεν απαιτούνται παράμετροι αιτήματος για αυτό το API.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	JSON αντικείμενο που περιέχει τις πληροφορίες για όλα τα μαθήματα, τους καθηγητές και τα μαθήματα οργανωμένα κατά εξάμηνο.

Το συγκεκριμένο API χρησιμοποιείται για την ανάκτηση όλων των μαθημάτων και των καθηγητών από τη βάση δεδομένων και την οργάνωσή τους κατά εξάμηνο. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε αυτές τις πληροφορίες:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να έχουν πρόσβαση στις πληροφορίες για τα μαθήματα και τους καθηγητές. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Ανάκτηση Μαθημάτων και Καθηγητών**: Όλα τα μαθήματα και οι καθηγητές ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Στα μαθήματα περιλαμβάνονται επίσης οι σχέσεις τους με τους καθηγητές.

4. **Οργάνωση Μαθημάτων κατά Εξάμηνο**: Τα μαθήματα ομαδοποιούνται κατά εξάμηνο και δημιουργείται ένα JSON αντικείμενο που περιέχει τις οργανωμένες πληροφορίες.

5. **Απόκριση**: Το JSON αντικείμενο επιστρέφεται ως απάντηση.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ανάκτηση πληροφοριών σχετικά με τα μαθήματα και τους καθηγητές από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να δουν όλες τις πληροφορίες για τα μαθήματα και τους καθηγητές, οργανωμένες κατά εξάμηνο, διευκολύνοντας έτσι τη διαχείριση και την κατανομή των πόρων.

4.2.2 Δημιουργία Μαθήματος

Path	`/admin/courses/create`
Method	`POST`
Request Body	`course_id` (number): Το μοναδικό αναγνωριστικό του μαθήματος. `type` (string): Ο τύπος του μαθήματος (π.χ. Θεωρία, Εργαστήριο). `name` (string): Το όνομα του μαθήματος. `semester` (string): Το εξάμηνο στο οποίο προσφέρεται το μάθημα. `teacherIds` (string): Αναγνωριστικά των καθηγητών που διδάσκουν το μάθημα, διαχωρισμένα με κόμμα.
Response	Ανακατεύθυνση στη σελίδα `/admin/courses`.

Το συγκεκριμένο API χρησιμοποιείται για τη δημιουργία ενός νέου μαθήματος στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να δημιουργήσουν μαθήματα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`course_id`, `type`, `name`, `semester`, `teacherIds`).

4. **Δημιουργία του Μαθήματος**: Το νέο μάθημα δημιουργείται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Επίσης, οι σχέσεις του μαθήματος με τους καθηγητές δημιουργούνται με την χρήση της μεθόδου `create`.

5. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/courses`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη δημιουργία νέων μαθημάτων στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να προσθέτουν νέα μαθήματα, διασφαλίζοντας ότι οι πληροφορίες για τα μαθήματα είναι πάντα ενημερωμένες.

4.2.3 Ενημέρωση Μαθήματος

Path	`/admin/courses/:id/update`
Method	`POST`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του μαθήματος που πρόκειται να ενημερωθεί.
Request Body	`course_id` (number): Το μοναδικό αναγνωριστικό του μαθήματος. `type` (string): Ο τύπος του μαθήματος (π.χ. Θεωρία, Εργαστήριο). `name` (string): Το όνομα του μαθήματος. `semester` (string): Το εξάμηνο στο οποίο προσφέρεται το μάθημα. `teacherIds` (string): Αναγνωριστικά των καθηγητών που διδάσκουν το μάθημα, διαχωρισμένα με κόμμα.
Response	Ανακατεύθυνση στη σελίδα `/admin/courses`.

Το συγκεκριμένο API χρησιμοποιείται για την ενημέρωση των πληροφοριών ενός μαθήματος στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ενημερώσουν μαθήματα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του μαθήματος (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`course_id`, `type`, `name`, `semester`, `teacherIds`).

5. **Ενημέρωση του Μαθήματος**: Το μάθημα ενημερώνεται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Επίσης, οι σχέσεις του μαθήματος με τους καθηγητές ενημερώνονται με την χρήση της μεθόδου `connectOrCreate`.

6. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/courses`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ενημέρωση των πληροφοριών μαθημάτων στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να διατηρούν ενημερωμένες τις πληροφορίες για τα μαθήματα, όπως το αναγνωριστικό, τον τύπο, το όνομα, το εξάμηνο και τους καθηγητές που τα διδάσκουν.

4.2.4 Ανέβασμα Μαθημάτων από Αρχείο CSV

Path	`/admin/courses/upload`
Method	`POST`
Request Body	Ένα αρχείο CSV που περιέχει τις πληροφορίες για τα μαθήματα που θα ανέβουν στη βάση δεδομένων. Κάθε γραμμή του CSV πρέπει να περιέχει τα πεδία `course_id`, `type`, `desc`, `ex`, και `teacherId`.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για την μαζική εισαγωγή ή ενημέρωση μαθημάτων στη βάση δεδομένων από ένα αρχείο CSV. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ανεβάσουν το αρχείο CSV. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Διαχείριση Αρχείου CSV**: Το αρχείο CSV αναλύεται και οι πληροφορίες που περιέχει διαβάζονται γραμμή προς γραμμή. Χρησιμοποιείται η βιβλιοθήκη `csv` για την επεξεργασία των δεδομένων. Για κάθε γραμμή του αρχείου, το μάθημα είτε δημιουργείται (αν δεν υπάρχει ήδη) είτε ενημερώνεται (αν υπάρχει) στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.Όλα τα μαθήματα και οι καθηγητές ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM. Στα μαθήματα περιλαμβάνονται επίσης οι σχέσεις τους με τους καθηγητές. Οι σχέσεις του μαθήματος με τους καθηγητές δημιουργούνται ή ενημερώνονται επίσης.

4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την μαζική εισαγωγή ή ενημέρωση μαθημάτων στη βάση δεδομένων από ένα αρχείο CSV. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να ενημερώνουν τις πληροφορίες των μαθημάτων με μεγαλύτερη ευκολία και αποτελεσματικότητα.

4.2.5 Διαγραφή Μαθήματος

Path	`/admin/courses/:id/delete`
Method	`DELETE`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του μαθήματος που πρόκειται να διαγραφεί.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή ενός μαθήματος από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν μαθήματα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του μαθήματος (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.
4. **Διαγραφή του Μαθήματος**: Το μάθημα διαγράφεται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
5. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή μαθημάτων από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν μαθήματα που δεν είναι πλέον σε χρήση ή έχουν αντικατασταθεί.

4.2.6 Διαγραφή Όλων των Μαθημάτων

Path	`/admin/courses/delete`
Method	`DELETE`
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή όλων των μαθημάτων από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν όλα τα μαθήματα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Διαγραφή Όλων των Μαθημάτων**: Όλα τα μαθήματα διαγράφονται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή όλων των μαθημάτων από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν όλα τα μαθήματα, πιθανώς για επανασχεδιασμό της δομής της βάσης δεδομένων ή για εκκαθάριση των δεδομένων.

4.3 Αίθουσες

4.3.1 Λήψη Αιθουσών

Path	`/admin/classrooms`
Method	`GET`
Request Parameters	Δεν απαιτούνται παράμετροι αιτήματος για αυτό το API.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	JSON αντικείμενο που περιέχει τις πληροφορίες για όλες τις αίθουσες και τις αίθουσες οργανωμένες κατά κτίριο.

Το συγκεκριμένο API χρησιμοποιείται για την ανάκτηση όλων των αιθουσών από τη βάση δεδομένων και την οργάνωσή τους κατά κτίριο. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε αυτές τις πληροφορίες:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να έχουν πρόσβαση στις αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Ανάκτηση Αιθουσών**: Όλες οι αίθουσες ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

4. **Οργάνωση Αιθουσών κατά Κτίριο**: Οι αίθουσες ομαδοποιούνται κατά κτίριο και δημιουργείται ένα JSON αντικείμενο που περιέχει τις οργανωμένες αίθουσες.

5. **Απόκριση**: Το JSON αντικείμενο επιστρέφεται ως απάντηση.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ανάκτηση πληροφοριών σχετικά με τις αίθουσες από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να δουν όλες τις αίθουσες, οργανωμένες κατά κτίριο, διευκολύνοντας έτσι την διαχείριση και την κατανομή των πόρων.

4.3.2 Δημιουργία Αίθουσας

Path	`/admin/classrooms/create`
Method	`POST`
Request Body	`building` (string): Το κτίριο στο οποίο βρίσκεται η αίθουσα. `capacity` (string): Η χωρητικότητα της αίθουσας. `name` (string): Το όνομα της αίθουσας. `equipment` (string): Ο εξοπλισμός που διαθέτει η αίθουσα.
Response	Ανακατεύθυνση στη σελίδα `/admin/classrooms`.

Το συγκεκριμένο API χρησιμοποιείται για τη δημιουργία μιας νέας αίθουσας στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να δημιουργήσουν αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`building`, `capacity`, `name`, `equipment`).

4. **Δημιουργία της Αίθουσας**: Η νέα αίθουσα δημιουργείται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

5. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/classrooms`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη δημιουργία νέων αιθουσών στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να προσθέτουν νέες αίθουσες, διασφαλίζοντας ότι οι πληροφορίες για τις διαθέσιμες αίθουσες είναι πάντα ενημερωμένες.

4.3.3 Ενημέρωση Αίθουσας

Path	`/admin/classrooms/:id/update`
Method	`POST`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό της αίθουσας που πρόκειται να ενημερωθεί.
Request Body	`building` (string): Το κτίριο στο οποίο βρίσκεται η αίθουσα. `capacity` (string): Η χωρητικότητα της αίθουσας. `name` (string): Το όνομα της αίθουσας. `equipment` (string): Ο εξοπλισμός που διαθέτει η αίθουσα.
Response	Ανακατεύθυνση στη σελίδα `/admin/classrooms`.

Το συγκεκριμένο API χρησιμοποιείται για την ενημέρωση των πληροφοριών μιας αίθουσας στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ενημερώσουν τις αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό της αίθουσας (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`building`, `capacity`, `name`, `equipment`).

5. **Ενημέρωση της Αίθουσας**: Η αίθουσα ενημερώνεται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

6. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/classrooms`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ενημέρωση των πληροφοριών αιθουσών στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να διατηρούν ενημερωμένες τις πληροφορίες για τις αίθουσες, όπως το κτίριο, τη χωρητικότητα, το όνομα και τον εξοπλισμό.

4.3.4 Ανέβασμα Αιθουσών από Αρχείο CSV

Path	`/admin/classrooms/upload`
Method	`GET`
Request Body	Ένα αρχείο CSV που περιέχει τις πληροφορίες για τις αίθουσες που θα ανέβουν στη βάση δεδομένων. Κάθε γραμμή του CSV πρέπει να περιέχει τα πεδία `building`, `capacity`, `name`, και `equipment`.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για την μαζική εισαγωγή ή ενημέρωση αιθουσών στη βάση δεδομένων από ένα αρχείο CSV. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ανεβάσουν το αρχείο CSV. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Διαχείριση Αρχείου CSV**: Το αρχείο CSV αναλύεται και οι πληροφορίες που περιέχει διαβάζονται γραμμή προς γραμμή. Χρησιμοποιείται η βιβλιοθήκη `csv` για την επεξεργασία των δεδομένων. Για κάθε γραμμή του αρχείου, η αίθουσα είτε δημιουργείται (αν δεν υπάρχει ήδη) είτε ενημερώνεται (αν υπάρχει) στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την μαζική εισαγωγή ή ενημέρωση αιθουσών στη βάση δεδομένων από ένα αρχείο CSV. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να ενημερώνουν τις πληροφορίες των αιθουσών με μεγαλύτερη ευκολία και αποτελεσματικότητα.

4.3.5 Διαγραφή Αίθουσας

Path	`/admin/classrooms/:id/delete`
Method	`DELETE`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό της αίθουσας που πρόκειται να διαγραφεί.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή μιας αίθουσας από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό της αίθουσας (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος.
4. **Διαγραφή της Αίθουσας**: Η αίθουσα διαγράφεται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
5. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή αιθουσών από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν αίθουσες που δεν είναι πλέον σε χρήση ή έχουν αντικατασταθεί.

4.3.6 Διαγραφή Όλων των Αιθουσών

Path	`/admin/classrooms/delete`
Method	`DELETE`
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή όλων των αιθουσών από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ενημερώσουν τις αίθουσες. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Διαγραφή Όλων των Αιθουσών**: Όλες οι αίθουσες διαγράφονται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή όλων των αιθουσών από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν όλες τις αίθουσες, πιθανώς για επανασχεδιασμό της δομής της βάσης δεδομένων ή για εκκαθάριση των δεδομένων.

4.4 Καθηγητής

4.4.1 Λήψη Καθηγητών και Συσχετισμένων Μαθημάτων

Path	`/admin/teachers`
Method	`GET`
Request Parameters	Δεν απαιτούνται παράμετροι αιτήματος για αυτό το API.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	JSON αντικείμενο που περιέχει τις πληροφορίες για όλους τους καθηγητές, τους συσχετισμένους με τους καθηγητές μαθήματα και την κατάσταση αυθεντικοποίησης του χρήστη.

Το συγκεκριμένο API χρησιμοποιείται για την ανάκτηση όλων των καθηγητών και των συσχετισμένων με τους καθηγητές μαθημάτων από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε αυτές τις πληροφορίες:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να έχουν πρόσβαση στις πληροφορίες για τους καθηγητές και τα μαθήματα. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Ανάκτηση Καθηγητών και Συσχετισμένων Μαθημάτων**: Όλοι οι καθηγητές και τα συσχετισμένα με τους καθηγητές μαθήματα ανακτώνται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

4. **Οργάνωση Καθηγητών κατά Όνομα**: Οι καθηγητές ομαδοποιούνται κατά πλήρες όνομα και δημιουργείται ένα JSON αντικείμενο που περιέχει τις οργανωμένες πληροφορίες.

5. **Απόκριση**: Το JSON αντικείμενο επιστρέφεται ως απάντηση.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ανάκτηση πληροφοριών σχετικά με τους καθηγητές και τα συσχετισμένα με τους καθηγητές μαθήματα από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να δουν όλες τις πληροφορίες για τους καθηγητές και τα συσχετισμένα μαθήματα, οργανωμένες κατά πλήρες όνομα, διευκολύνοντας έτσι τη διαχείριση και την κατανομή των πόρων.

4.4.2 Δημιουργία Καθηγητή

Path	`/admin/teachers/create`
Method	`POST`
Request Body	`fullName` (string): Το πλήρες όνομα του καθηγητή. `role` (string): Ο ρόλος του καθηγητή (π.χ. Καθηγητής, Επίκουρος Καθηγητής). `teacher_id` (number): Το μοναδικό αναγνωριστικό του καθηγητή.
Response	Ανακατεύθυνση στη σελίδα `/admin/teachers`.

Το συγκεκριμένο API χρησιμοποιείται για τη δημιουργία ενός νέου καθηγητή στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να δημιουργήσουν καθηγητές. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`fullName`, `role`, `teacher_id`).

4. **Δημιουργία του Καθηγητή**: Ο νέος καθηγητής δημιουργείται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

5. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/teachers`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη δημιουργία νέων καθηγητών στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να προσθέτουν νέους καθηγητές, διασφαλίζοντας ότι οι πληροφορίες για τους καθηγητές είναι πάντα ενημερωμένες.

4.4.3 Ενημέρωση Καθηγητή

Path	`/admin/teachers/:id/update`
Method	`POST`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του καθηγητή που πρόκειται να ενημερωθεί.
Request Body	`fullName` (string): Το πλήρες όνομα του καθηγητή. `role` (string): Ο ρόλος του καθηγητή (π.χ. Καθηγητής, Επίκουρος Καθηγητής). `teacher_id` (number): Το μοναδικό αναγνωριστικό του καθηγητή.
Response	Ανακατεύθυνση στη σελίδα `/admin/teachers`.

Το συγκεκριμένο API χρησιμοποιείται για την ενημέρωση των πληροφοριών ενός καθηγητή στη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ενημερώσουν καθηγητές. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του καθηγητή (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Επικύρωση Δεδομένων Φόρμας**: Τα δεδομένα της φόρμας επικυρώνονται χρησιμοποιώντας το zod-form-data schema για να διασφαλιστεί ότι περιλαμβάνουν τα απαιτούμενα πεδία (`fullName`, `role`, `teacher_id`).

5. **Ενημέρωση του Καθηγητή**: Ο καθηγητής ενημερώνεται στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

6. **Απόκριση**: Ανακατεύθυνση στη σελίδα `/admin/teachers`.

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την ενημέρωση των πληροφοριών καθηγητών στη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να διατηρούν ενημερωμένες τις πληροφορίες για τους καθηγητές, όπως το πλήρες όνομα, τον ρόλο και το μοναδικό αναγνωριστικό τους.

4.4.4 Ανέβασμα Καθηγητών από Αρχείο CSV

Path	`/admin/teachers/upload`
Method	`POST`
Request Body	Ένα αρχείο CSV που περιέχει τις πληροφορίες για τους καθηγητές που θα ανέβουν στη βάση δεδομένων. Κάθε γραμμή του CSV πρέπει να περιέχει τα πεδία `teacher_id`, `fullName`, και `role`.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για την μαζική εισαγωγή ή ενημέρωση καθηγητών στη βάση δεδομένων από ένα αρχείο CSV. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να ανεβάσουν το αρχείο CSV. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Διαχείριση Αρχείου CSV**: Το αρχείο CSV αναλύεται και οι πληροφορίες που περιέχει διαβάζονται γραμμή προς γραμμή. Χρησιμοποιείται η βιβλιοθήκη `csv` για την επεξεργασία των δεδομένων. Για κάθε γραμμή του αρχείου, ο καθηγητής είτε δημιουργείται (αν δεν υπάρχει ήδη) είτε ενημερώνεται (αν υπάρχει) στη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για την μαζική εισαγωγή ή ενημέρωση καθηγητών στη βάση δεδομένων από ένα αρχείο CSV. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του επιτρέπει στους διαχειριστές να ενημερώνουν τις πληροφορίες των καθηγητών με μεγαλύτερη ευκολία και αποτελεσματικότητα.

4.4.5 Διαγραφή Καθηγητή

Path	`/admin/teachers/:id/delete`
Method	`DELETE`
Request Parameters	`id` (number): Το μοναδικό αναγνωριστικό του καθηγητή που πρόκειται να διαγραφεί.
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή ενός καθηγητή από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).

2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν καθηγητές. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).

3. **Επικύρωση και Ανάλυση Παραμέτρων**: Το αναγνωριστικό του καθηγητή (`id`) επικυρώνεται και αναλύεται από τις παραμέτρους του αιτήματος χρησιμοποιώντας το zod schema.

4. **Διαγραφή του Καθηγητή**: Ο καθηγητής διαγράφεται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.

5. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή καθηγητών από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν καθηγητές που δεν είναι πλέον σε χρήση ή έχουν αντικατασταθεί.

4.4.6 Διαγραφή Όλων των Καθηγητών

Path	`/admin/teachers/delete`
Method	`DELETE`
Request Body	Δεν απαιτείται σώμα αιτήματος για αυτό το API.
Response	Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται για τη διαγραφή όλων των καθηγητών από τη βάση δεδομένων. Ακολουθεί μια σειρά από βήματα για να διασφαλίσει ότι μόνο οι εξουσιοδοτημένοι χρήστες μπορούν να εκτελέσουν αυτή την ενέργεια:

1. **Αυθεντικοποίηση Χρήστη**: Ο χρήστης που πραγματοποιεί το αίτημα πρέπει να είναι αυθεντικοποιημένος. Αν ο χρήστης δεν είναι αυθεντικοποιημένος, ανακατευθύνεται στη σελίδα εισόδου (`/auth/login`).
2. **Έλεγχος Δικαιωμάτων Χρήστη**: Μόνο οι χρήστες με δικαιώματα διαχειριστή (`isAdmin`) επιτρέπεται να διαγράψουν καθηγητές. Αν ο χρήστης δεν είναι διαχειριστής, ανακατευθύνεται στην αρχική σελίδα (`/`).
3. **Διαγραφή Όλων των Καθηγητών**: Όλοι οι καθηγητές διαγράφονται από τη βάση δεδομένων χρησιμοποιώντας το Prisma ORM.
4. **Απόκριση**: Δεν επιστρέφεται κάποιο σώμα απάντησης (επιστρέφεται `null`).

Το συγκεκριμένο API χρησιμοποιείται αποκλειστικά από διαχειριστές της εφαρμογής για τη διαγραφή όλων των καθηγητών από τη βάση δεδομένων. Είναι διαθέσιμο μόνο για χρήστες με δικαιώματα διαχειριστή και απαιτεί αυθεντικοποίηση. Η χρήση του περιορίζεται σε περιπτώσεις όπου οι διαχειριστές χρειάζεται να αφαιρέσουν όλους τους καθηγητές, πιθανώς για επανασχεδιασμό της δομής της βάσης δεδομένων ή για εκκαθάριση των δεδομένων.

Κεφάλαιο 5

Σχεδιασμός της Διεπιφάνειας Χρήστη

Η διεπιφάνεια χρήστη της εφαρμογής είναι σχεδιασμένη για να παρέχει μια ομαλή και αποτελεσματική εμπειρία τόσο για τους απλούς χρήστες όσο και για τους διαχειριστές. Παρακάτω παρουσιάζονται οι σελίδες που είναι διαθέσιμες στην εφαρμογή και οι λειτουργίες που μπορεί να εκτελέσει ο χρήστης σε κάθε μία από αυτές.

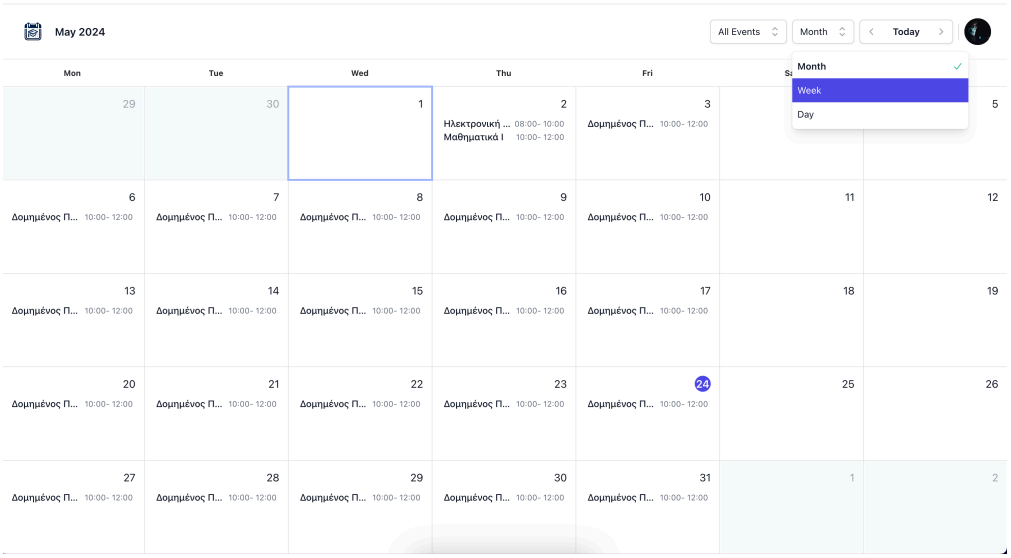
5.1 User

5.1.1 Calendar View (/)

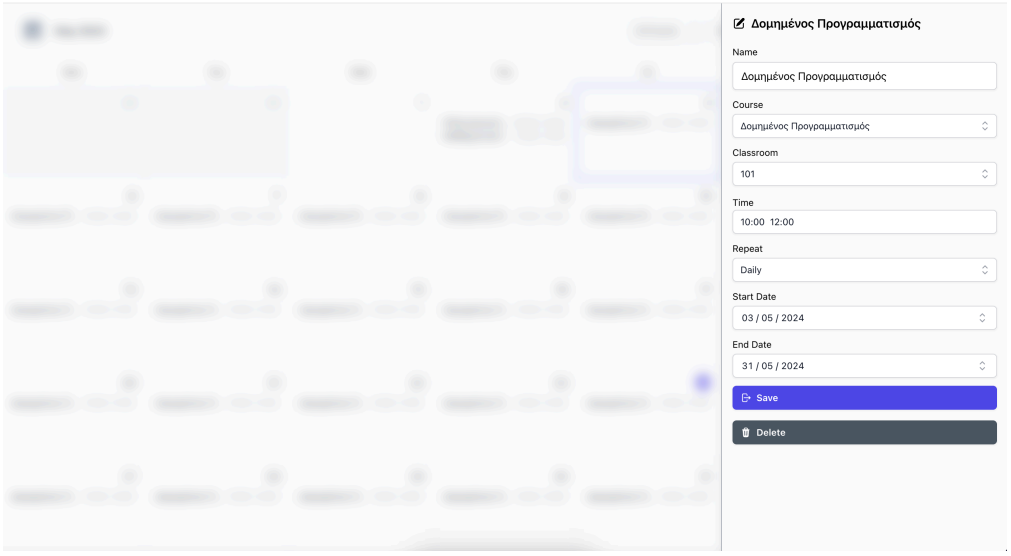
Η κεντρική προβολή ημερολογίου επιτρέπει στον χρήστη να δει όλα τα γεγονότα ανά μήνα, εβδομάδα ή ημέρα σε ένα προσβάσιμο και λειτουργικό ημερολόγιο.

Λειτουργίες:

- Προβολή όλων των γεγονότων ανά μήνα, εβδομάδα ή ημέρα.
- Πλοήγηση με το πληκτρολόγιο, με δυνατότητα επιστροφής στη σημερινή ημέρα.
- Προβολή προφίλ (σε modal).
- Προσθήκη, ενημέρωση και διαγραφή γεγονότων (μόνο για χρήστες με ιδιότητα Staff).



Σχήμα 5.1: Calendar View



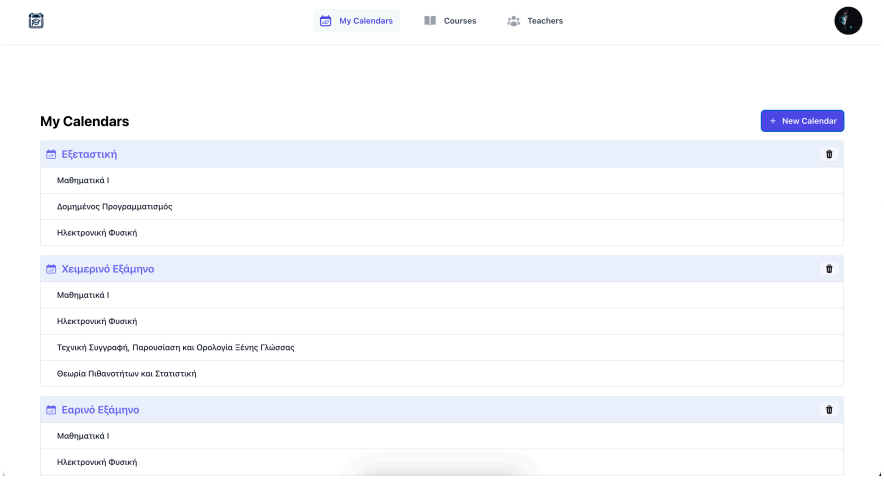
Σχήμα 5.2: Edit Event

5.1.2 My Calendars (user/calendars)

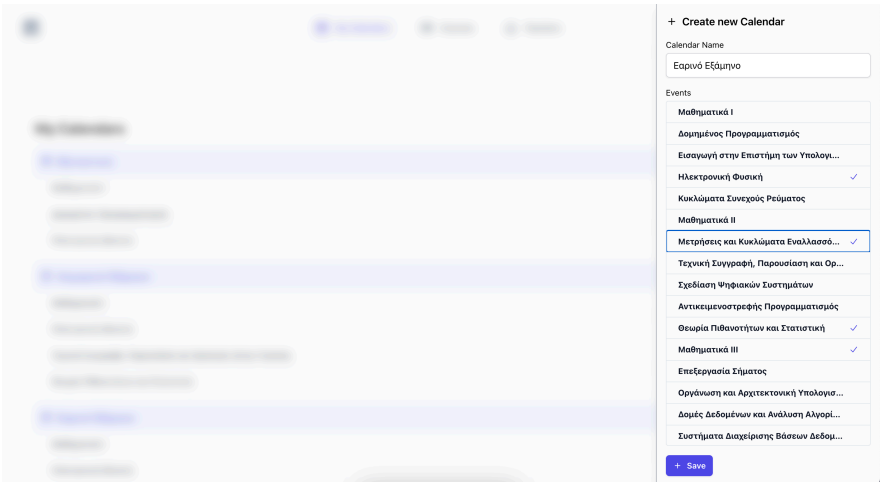
Στην προβολή αυτή, ο χρήστης μπορεί να δει και να διαχειριστεί τα δικά του ημερολόγια.

Λειτουργίες:

- Προβολή των ημερολογίων του χρήστη.
- Δημιουργία νέου ημερολογίου επιλέγοντας ποια μαθήματα θα περιλαμβάνει.
- Διαγραφή ενός ημερολογίου.



Σχήμα 5.3: My Calendar



Σχήμα 5.4: Create New Calendar

5.1.3 Courses (user/courses)

Αυτή η σελίδα επιτρέπει στους χρήστες να δουν μια λίστα όλων των μαθημάτων του Πανεπιστημίου.

Λειτουργίες:

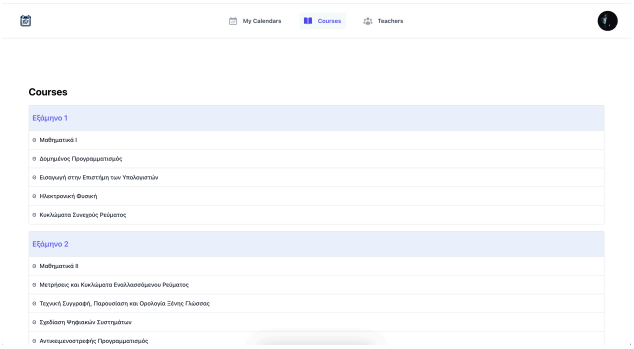
- Προβολή μιας λίστας όλων των μαθημάτων.

5.1.4 Teachers (user/teachers)

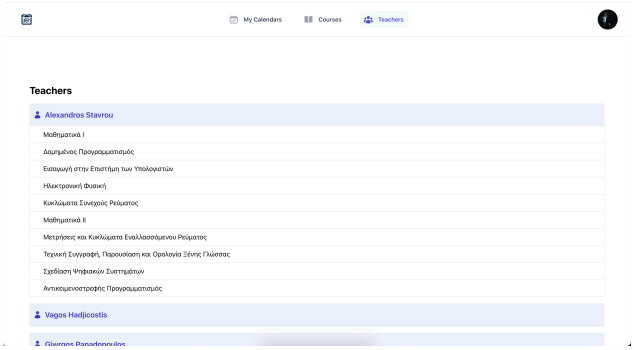
Σε αυτήν την προβολή, οι χρήστες μπορούν να δουν μια λίστα όλων των διδασκόντων και των μαθημάτων που διδάσκουν.

Λειτουργίες:

- Προβολή μιας λίστας όλων των διδασκόντων.
- Προβολή των μαθημάτων που διδάσκει κάθε διδάσκων.



Σχήμα 5.5: Course View



Σχήμα 5.6: Teacher View

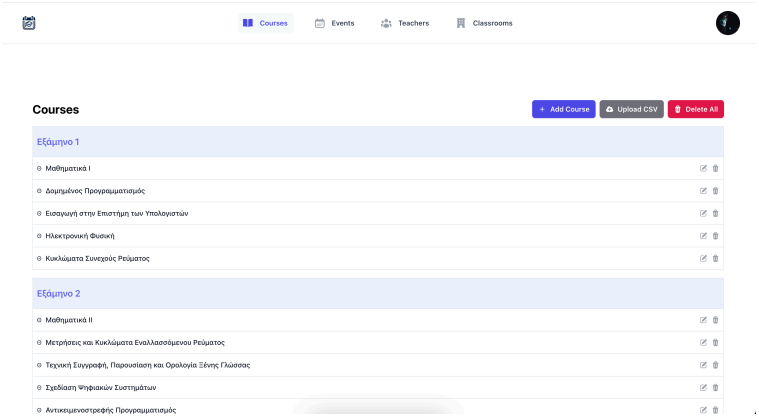
5.2 Admin

5.2.1 Courses (admin/courses)

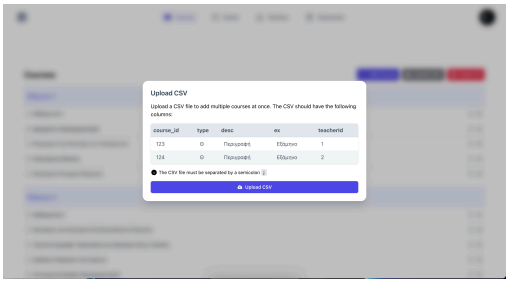
Οι διαχειριστές έχουν πρόσβαση σε μια σελίδα διαχείρισης μαθημάτων, όπου μπορούν να δουν, να προσθέσουν, να ενημερώσουν ή να διαγράψουν μαθήματα.

Λειτουργίες:

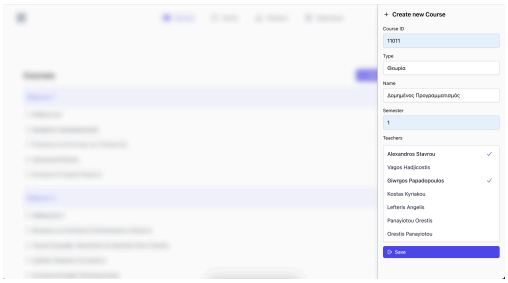
- Προβολή λίστας μαθημάτων.
- Ανέβασμα αρχείων CSV (ανοίγει modal που δείχνει τη μορφή του CSV και επιτρέπει την ανέβασμα του αρχείου).
- Διαγραφή όλων των μαθημάτων.
- Προσθήκη μαθήματος (ανοίγει modal).
- Ενημέρωση μαθήματος (ανοίγει modal).



Σχήμα 5.7: Admin Course View



Σχήμα 5.8: Csv Course Upload



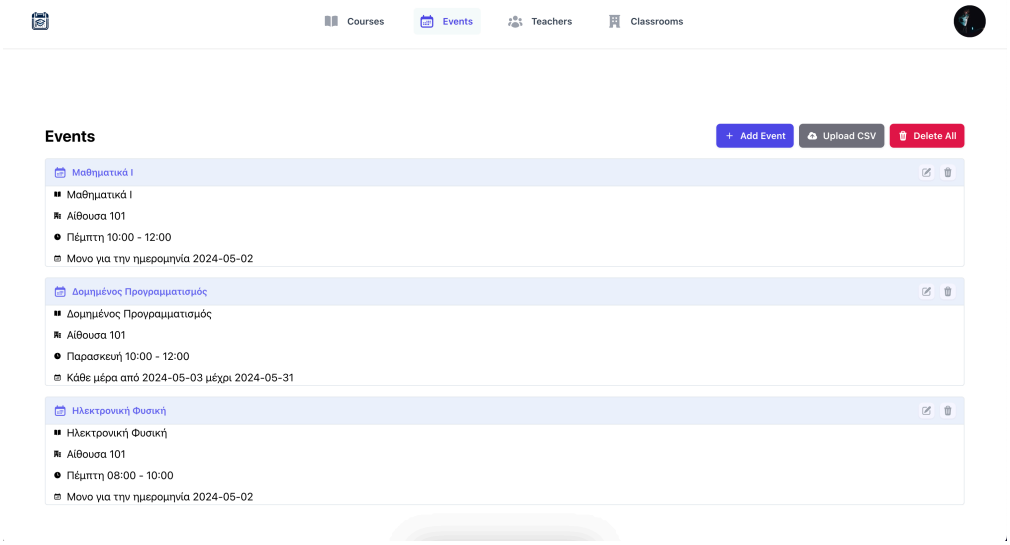
Σχήμα 5.9: Create Course

5.2.2 Events

Σε αυτήν τη σελίδα, οι διαχειριστές μπορούν να διαχειριστούν όλα τα γεγονότα, συμπεριλαμβανομένου της προσθήκης, της ενημέρωσης και της διαγραφής τους.

Λειτουργίες:

- Προβολή λίστας γεγονότων.
- Ανέβασμα αρχείων CSV (ανοίγει modal που δείχνει τη μορφή του CSV και επιτρέπει την ανέβασμα του αρχείου).
- Διαγραφή όλων των γεγονότων.
- Προσθήκη γεγονότος (ανοίγει modal).
- Ενημέρωση γεγονότος (ανοίγει modal).



Σχήμα 5.10: Admin Event View

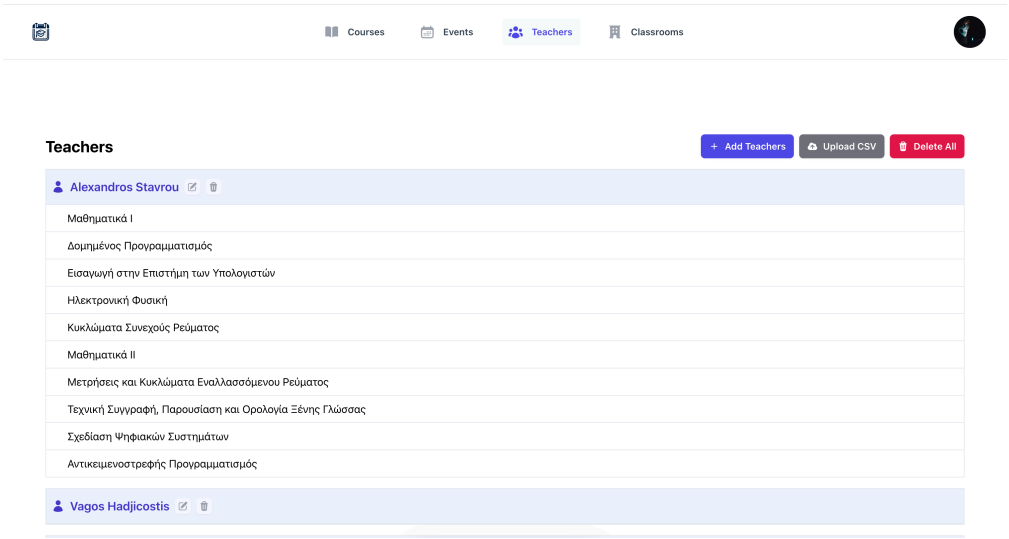
5.2.3 Teachers (admin/teachers)

Η σελίδα αυτή επιτρέπει στους διαχειριστές να διαχειριστούν τους διδάσκοντες, περιλαμβάνοντας λειτουργίες για προσθήκη, ενημέρωση και διαγραφή διδασκόντων.

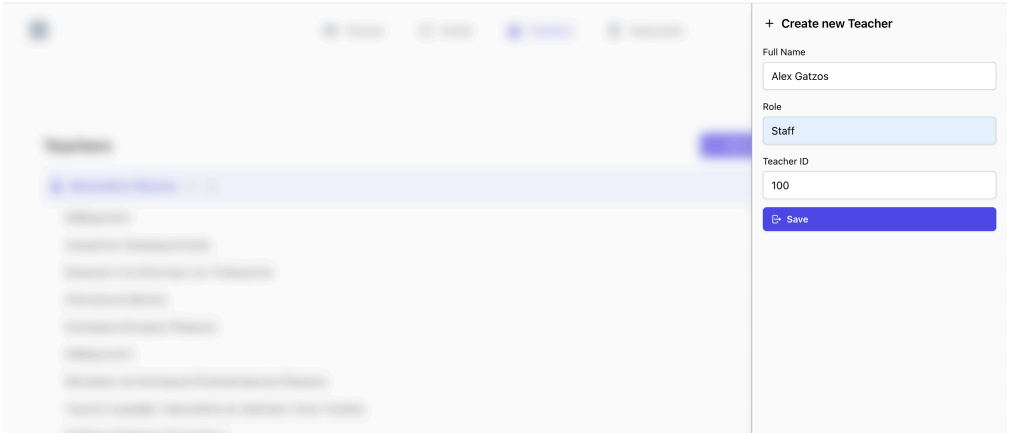
Λειτουργίες:

- Προβολή λίστας διδασκόντων.

- Ανέβασμα αρχείων CSV (ανοίγει modal που δείχνει τη μορφή του CSV και επιτρέπει την ανέβασμα του αρχείου).
- Διαγραφή όλων των διδασκόντων.
- Προσθήκη διδάσκοντα (ανοίγει modal).
- Ενημέρωση διδάσκοντα (ανοίγει modal).



Σχήμα 5.11: Admin Teacher View



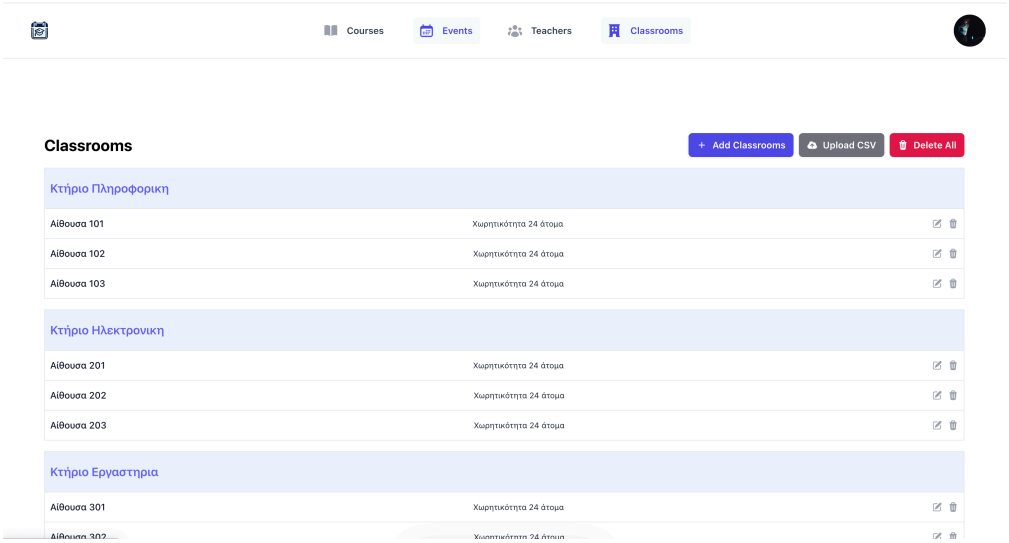
Σχήμα 5.12: Create Teacher

5.2.4 Classrooms (admin/classrooms)

Σε αυτήν τη σελίδα, οι διαχειριστές μπορούν να διαχειριστούν τις αίθουσες διδασκαλίας, περιλαμβάνοντας λειτουργίες για προσθήκη, ενημέρωση και διαγραφή αιθουσών.

Λειτουργίες:

- Προβολή λίστας αιθουσών.
- Ανέβασμα αρχείων CSV (ανοίγει modal που δείχνει τη μορφή του CSV και επιτρέπει την ανέβασμα του αρχείου).
- Διαγραφή όλων των αιθουσών.
- Προσθήκη αίθουσας (ανοίγει modal).
- Ενημέρωση αίθουσας (ανοίγει modal).



Σχήμα 5.13: Admin Classroom View

Κεφάλαιο 6

Συμπεράσματα

Η παρούσα εργασία ανέλυσε τον σχεδιασμό και την ανάπτυξη μιας εφαρμογής διαχείρισης ακαδημαϊκών δεδομένων, εστιάζοντας σε σημαντικές τεχνολογίες και εργαλεία που χρησιμοποιήθηκαν, όπως το React, το Remix, το Tailwind CSS, το Node.js και το Prisma. Καθώς και στην υλοποίηση μιας λειτουργικής και φιλικής προς τον χρήστη διεπιφάνειας χρήστη. Μέσα από την ανάλυση αυτή, προέκυψαν διάφορα συμπεράσματα που αφορούν τις επιλογές τεχνολογίας, τη δομή της βάσης δεδομένων, την ανάπτυξη των APIs και τη σχεδίαση της διεπιφάνειας χρήστη.

6.1 Τεχνολογίες και Εργαλεία

Η επιλογή του React για το frontend απέδειξε ότι είναι εξαιρετικά αποδοτική για την ανάπτυξη μοντέρνων, δυναμικών και επαναχρησιμοποιήσιμων user interfaces. Η χρήση των hooks και του component-based αρχιτεκτονικού προτύπου συνέβαλε στη δημιουργία μιας εύχρηστης και ευέλικτης εφαρμογής. Ο συνδυασμός του React με το Remix παρείχε μια ισχυρή βάση για την ανάπτυξη full-stack εφαρμογών, προσφέροντας εργαλεία για server-side rendering, static site generation και βελτιστοποίηση της φόρτωσης δεδομένων.

Το Tailwind CSS επέτρεψε την ταχεία και αποδοτική δημιουργία προσαρμοσμένων σχεδίων με μια utility-first προσέγγιση. Η ενσωμάτωση του Tailwind CSS βελτίωσε την απόδοση της εφαρμογής και διασφάλισε τη συνοχή του σχεδίου σε όλες τις σελίδες.

Η χρήση του Node.js για το backend επιβεβαίωσε τα πλεονεκτήματα της event-driven και non-blocking I/O αρχιτεκτονικής, επιτρέποντας την ανάπτυξη κλιμακούμενων και υψηλής απόδοσης εφαρμογών. Το Prisma, ως ένα σύγχρονο ORM εργαλείο, απλοποίησε τη διαχείριση της βάσης δεδομένων, προσφέροντας type-safe queries και εύκολη διαχείριση μεταναστεύσεων.

6.2 Δομή της Βάσης Δεδομένων

Η σχεδίαση της βάσης δεδομένων επικεντρώθηκε στη δημιουργία σαφών και λειτουργικών σχέσεων μεταξύ των πινάκων (Event, Classroom, Course, Teacher, CourseTeachers, Calendars), διασφαλίζοντας την ακεραιότητα των δεδομένων και την αποδοτικότητα των ερωτήσεων. Η χρήση του Prisma διευκόλυνε τη διαχείριση των δεδομένων και παρείχε έναν ενιαίο τρόπο αλληλεπίδρασης με διαφορετικές βάσεις δεδομένων.

6.3 Ανάπτυξη APIs

Η ανάπτυξη των APIs ακολούθησε τις καλύτερες πρακτικές για τη δημιουργία ασφαλών και αποδοτικών endpoints, με χρήση των εργαλείων και των προτύπων του Remix. Τα APIs σχεδιάστηκαν για να καλύπτουν τις ανάγκες τόσο των απλών χρηστών όσο και των διαχειριστών, προσφέροντας λειτουργίες όπως προσθήκη, ενημέρωση και διαγραφή δεδομένων, καθώς και διαχείριση αρχείων CSV.

6.4 Σχεδίαση της Διεπιφάνειας Χρήστη

Η διεπιφάνεια χρήστη σχεδιάστηκε με γνώμονα την ευχρηστία και την αποτελεσματικότητα, προσφέροντας μια φιλική και εύχρηστη εμπειρία για τους χρήστες. Οι διαχειριστές έχουν πρόσβαση σε επιπλέον λειτουργίες διαχείρισης, επιτρέποντάς τους να διαχειρίζονται δεδομένα μαθημάτων, γεγονότων, διδασκόντων και αιθουσών διδασκαλίας με ευκολία.

6.5 Τελικές Σκέψεις

Η ανάπτυξη της εφαρμογής αυτής ανέδειξε τη σημασία της σωστής επιλογής τεχνολογιών και εργαλείων, καθώς και της προσεκτικής σχεδίασης της βάσης δεδομένων και των APIs. Η χρήση σύγχρονων τεχνολογιών όπως το React, το Remix, το Tailwind CSS, το Node.js και το Prisma όχι μόνο βελτίωσε την απόδοση και την ασφάλεια της εφαρμογής, αλλά συνέβαλε και στη δημιουργία μιας ευχάριστης εμπειρίας για τον χρήστη.

Η εργασία αυτή καταδεικνύει ότι η υιοθέτηση των κατάλληλων τεχνολογιών και πρακτικών μπορεί να οδηγήσει στην ανάπτυξη αποδοτικών και εύχρηστων εφαρμογών που καλύπτουν πλήρως τις ανάγκες των χρηστών και των διαχειριστών τους.

6.6 Μελλοντικές Βελτιώσεις-Επεκτάσεις

Παρά τη σημαντική πρόοδο που επιτεύχθηκε με την ανάπτυξη της παρούσας εφαρμογής, υπάρχουν σημεία που μπορούν να βελτιωθούν και να επεκταθούν στο μέλλον για να καλύψουν τις ανάγκες των χρηστών και να προσφέρουν επιπλέον λειτουργίες.

Μία από αυτές είναι η αυτόματη δημιουργία και αποστολή ανακοινώσεων μέσω API όταν γίνεται αλλαγή σε ένα μάθημα. Αυτό θα διασφαλίσει ότι οι χρήστες θα ενημερώνονται άμεσα για οποιεσδήποτε αλλαγές, διασφαλίζοντας έτσι την έγκαιρη ενημέρωσή τους και τη συνεχή επικοινωνία μεταξύ των καθηγητών και των φοιτητών.

Η ανάπτυξη μιας native mobile εφαρμογής ή η βελτίωση της υποστήριξης για κινητές συσκευές μέσω τεχνολογιών progressive web app μπορεί να προσφέρει στους χρήστες μια πιο προσβάσιμη και φιλική εμπειρία χρήσης από τα κινητά τους τηλέφωνα και τα tablets.

Βιβλιογραφία

[1] J. Walke. “Introducing React”. (2013), διεύθν.: https://reactjs.org/.

[2] J. Miller. “The Virtual DOM”. (2017), διεύθν.: https://preactjs.com/guide/v10/differences-to-react#virtual-dom.

[3] D. Abramov. “Optimizing Performance in React”. (2018), διεύθν.: https://reactjs.org/docs/optimizing-performance.html.

[4] K. C. Dodds. “Advanced React Component Patterns”. (2020), διεύθν.: https://kentcdodds.com/blog/advanced-react-component-patterns.

[5] R. Florence. “JSX: The Confusing Parts”. (2019), διεύθν.: https://reacttraining.com/blog/jsx-the-confusing-parts.

[6] S. Rajde. “Everything about state and props in React”. (2020), διεύθν.: https://letsreact.org/everything-about-state-and-props-in-react.

[7] V. Gatwiri. “React Component Lifecycle Methods – Explained with Examples”. (2023), διεύθν.: https://www.freecodecamp.org/news/react-component-lifecycle-methods/.

[8] S. Markbåge. “Introducing Hooks”. (2018), διεύθν.: https://reactjs.org/docs/hooks-intro.html.

[9] M. Stoiber. “React Community and Ecosystem”. (2018), διεύθν.: https://reactjs.org/community/support.html.

[10] R. Florence και M. Jackson. “Introducing Remix”. (2020), διεύθν.: https://remix.run/docs/en/main/discussion/introduction.

[11] G. Rauch. “Server-Side Rendering (SSR)”. (2019), διεύθν.: https://vercel.com/docs/frameworks/remix#server-side-rendering-ssr.

[12] M. Jackson. “Route Configuration”. (2020), διεύθν.: https://remix.run/docs/en/main/discussion/routes.

[13] J. Palmer. “Data Loading in Remix”. (2020), διεύθν.: https://formik.org/docs/overview.

[14] D. Abramov. “Building Great User Experiences with Concurrent Mode and Suspense”. (2019), διεύθν.: https://legacy.reactjs.org/blog/2019/11/06/building-great-user-experiences-with-concurrent-mode-and-suspense.html.

[15] L. Wroblewski. “An Event Apart: Why Progressive Enhancement Matters”. (2014), διεύθν.: https://www.lukew.com/ff/entry.asp?1875.

[16] A. Wathan. “Tailwind CSS: From Zero to Production”. (2017), διεύθν.: https://tailwindcss.com/.

[17] S. Schoger. “Build Projects Faster with Tailwind CSS”. (2018), διεύθν.: https://floatui.com/blog/build-projects-faster-with-tailwind-css.

[18] A. Wathan. “Customizing Tailwind CSS”. (2019), διεύθν.: https://tailwindcss.com/docs/configuration.

[19] J. Reinink. “Optimizing for Production with Tailwind CSS”. (2018), διεύθν.: https://tailwindcss.com/docs/optimizing-for-production.

[20] A. Wathan. “Utility-First CSS with Tailwind”. (2017), διεύθν.: https://tailwindcss.com/docs/utility-first.

[21] A. Wathan. “Responsive Design with Tailwind CSS”. (2018), διεύθν.: https://tailwindcss.com/docs/responsive-design.

[22] C. Sev. “Tailwind CSS: Community and Ecosystem”. (2020), διεύθν.: https://v1.tailwindcss.com/community.

[23] R. Dahl. “Node.js: Asynchronous I/O for JavaScript”. (2009), διεύθν.: https://nodejs.org/en/about/.

[24] B. Belder. “Understanding Event-Driven Architecture in Node.js”. (2014), διεύθν.: https://www.ibm.com/cloud/architecture/architectures/event-driven/event-driven-architecture.

[25] L. Voss. “The npm Ecosystem”. (2015), διεύθν.: https://www.npmjs.com/.

[26] G. Rauch. “Real-Time Web Applications with Node.js”. (2018), διεύθν.: https://socket.io/get-started/.

[27] P. Irish. “Why JavaScript Will Become The Dominant Programming Language Of The Enterprise”. (2012), διεύθν.: https://readwrite.com/why-javascript-will-become-the-dominant-programming-language-of-the-enterprise/.

[28] S. Newman. “Building Microservices”. (2015), διεύθν.: https://samnewman.io/books/building_microservices/.

[29] T. Cyren. “Node.js: Tools and Practices”. (2019), διεύθν.: https://github.com/nodejs/node.

[30] N. Burk. “Introducing Prisma: Modern ORM for Node.js”. (2018), διεύθν.: https://www.prisma.io/docs/concepts/overview/what-is-prisma.

[31] J. Piotrowski. “Type-Safe Database Access with Prisma”. (2019), διεύθν.: https://www.prisma.io/blog/announcing-prisma-2-zq1s745db8i5.

[32] N. Burk. “Prisma: Multi-Database Support”. (2020), διεύθν.: https://www.prisma.io/docs/concepts/database-connectors.

[33] T. Dooner. “Prisma Migrate: Database Schema Management”. (2020), διεύθν.: https://www.prisma.io/docs/concepts/components/prisma-migrate.

[34] N. Burk. “Getting Started with Prisma Client”. (2019), διεύθν.: https://www.prisma.io/docs/getting-started.

[35] J. Schickling. “Prisma Studio: GUI for Database Management”. (2019), διεύθν.: https://www.prisma.io/docs/concepts/components/prisma-studio.

[36] J. Piotrowski. “TypeScript ORM with zero-cost type-safety for your database”. (2020), διεύθν.: https://www.prisma.io/typescript.

[37] N. Burk. “Using GraphQL Nexus with a Database”. (2019), διεύθν.: https://www.prisma.io/blog/using-graphql-nexus-with-a-database-pmyl3660ncst.

[38] A. Hejlsberg. “Introducing TypeScript”. (2012), διεύθν.: https://www.typescriptlang.org/.

[39] D. Rosenwasser. “Type Safety in TypeScript”. (2015), διεύθν.: https://www.typescriptlang.org/docs/handbook/type-checking-javascript-files.html.

[40] A. Hejlsberg. “TypeScript for Large-Scale JavaScript Applications”. (2014), διεύθν.: https://www.typescriptlang.org/docs/handbook/migrating-from-javascript.html.

[41] R. Cavanaugh. “Collaborative Development with TypeScript”. (2018), διεύθν.: https://www.typescriptlang.org/docs/handbook/advanced-types.html.

[42] D. Vanderkam. “Integrating TypeScript in Existing Projects”. (2019), διεύθν.: https://www.typescriptlang.org/docs/handbook/declaration-files/introduction.html.

[43] A. Hejlsberg. “Advanced TypeScript Features”. (2017), διεύθν.: https://www.typescriptlang.org/docs/handbook/release-notes/typescript-2-0.html.

[44] E. Gamma. “TypeScript: Tooling and IDE Support”. (2018), διεύθν.: https://code.visualstudio.com/docs/languages/typescript.