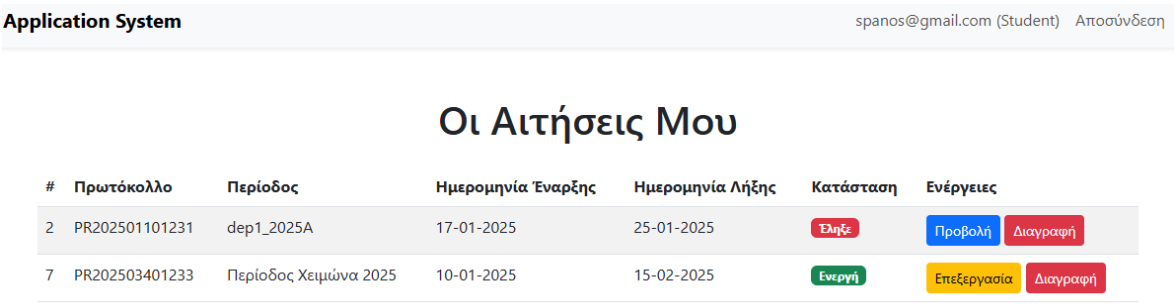
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΔΙΠΛΩΜΑΤΙΚΗ ΕΡΓΑΣΙΑ

«Διαδικτυακή εφαρμογή διαχείρισης αιτήσεων»



Φοιτητής

185277

ΑΝΤΩΝΙΟΣ ΣΠΑΝΟΣ

Επιβλέπων

Δρ. Κυριάκος Τσιακμάκης

Ιανουάριος 2025

Διαδικτυακή εφαρμογή διαχείρισης αιτήσεων

Κωδικός: 24302

Φοιτητής: Αντώνιος Σπανός

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 31-10-2024

Ημερομηνία περάτωσης Π.Ε. 26-01-2025

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.

Η έγκριση της πτυχιακής εργασίας από το Τμήμα Μηχανικών Πληροφορικής και Ηλεκτρονικών Συστημάτων του Διεθνούς Πανεπιστημίου της Ελλάδος, δεν υποδηλώνει απαραιτήτως και αποδοχή των απόψεων του συγγραφέα, εκ μέρους του Τμήματος.

Περίληψη

Η εργασία αυτή αφορά ένα ολοκληρωμένο σύστημα που διευκολύνει τη διαδικασία υποβολής, διαχείρισης και παρακολούθησης αιτήσεων. Οι χρήστες μπορούν να υποβάλουν αιτήσεις μέσω μιας φιλικής προς τον χρήστη διεπαφής, συμπληρώνοντας τα απαιτούμενα στοιχεία και ανεβάζοντας σχετικά έγγραφα, όπως ταυτότητα, πτυχία ή άλλα δικαιολογητικά. Η πλατφόρμα υποστηρίζει αυτοματοποιημένες ειδοποιήσεις, προηγμένα φίλτρα αναζήτησης και εργαλεία διαχείρισης, επιτρέποντας στους διαχειριστές να επεξεργάζονται αιτήσεις αποτελεσματικά. Παρέχει real-time ενημερώσεις για την κατάσταση των αιτήσεων, εξασφαλίζοντας διαφάνεια και ταχύτερους χρόνους απόκρισης. Με στόχο την απλοποίηση σύνθετων διαδικασιών, η εφαρμογή ενσωματώνει τεχνολογίες όπως το Laravel, τη MySQL και το Bootstrap για ταχύτητα και ευελιξία.

« Online Application Management System »

Abstract

This work concerns an integrated system that facilitates the process of submitting, managing and monitoring applications. Users can submit applications through a user-friendly interface, filling in the required information and uploading relevant documents, such as ID, degrees or other supporting documents. The platform supports automated notifications, advanced search filters and management tools, allowing administrators to process applications efficiently. It provides real-time updates on the status of applications, ensuring transparency and faster response times. Aiming to simplify complex processes, the application integrates technologies such as Laravel, MySQL and Bootstrap for speed and flexibility.

Ευχαριστίες

Να ευχαριστήσω τους γονείς μου και τους φίλους μου και τον επιβλέπων μου για τη συνεχή καθοδήγηση και συμβολή του.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Η διαχείριση αιτήσεων αποτελεί μια κρίσιμη πτυχή της καθημερινής λειτουργίας πολλών οργανισμών, εκπαιδευτικών ιδρυμάτων, επιχειρήσεων, και δημόσιων φορέων. Από τη συλλογή αιτήσεων μέχρι την αξιολόγηση και την απάντηση στους ενδιαφερόμενους, οι παραδοσιακές μέθοδοι διαχείρισης συχνά χαρακτηρίζονται από αργούς χρόνους απόκρισης, ανθρώπινα λάθη, και δυσκολίες στην παρακολούθηση των αιτήσεων. Σε αυτό το πλαίσιο, τα σύγχρονα πληροφοριακά συστήματα έχουν αναδειχθεί ως ουσιαστικά εργαλεία για τη βελτίωση της αποδοτικότητας και της διαφάνειας στη διαδικασία διαχείρισης αιτήσεων.

Το παρόν έργο αφορά την ανάπτυξη ενός ολοκληρωμένου συστήματος διαχείρισης αιτήσεων, σχεδιασμένου να ανταποκρίνεται στις ανάγκες ενός ευρέος φάσματος χρηστών, από φοιτητές που υποβάλλουν αιτήσεις σε εκπαιδευτικά ιδρύματα, έως καθηγητές και διαχειριστές που αξιολογούν και εγκρίνουν αιτήσεις. Το σύστημα προσφέρει ένα φιλικό προς τον χρήστη περιβάλλον, αυτοματοποιώντας παράλληλα κρίσιμες λειτουργίες, όπως η υποβολή, επεξεργασία, προβολή και αποθήκευση αιτήσεων.

Ο βασικός σκοπός του συστήματος είναι η βελτίωση της αποτελεσματικότητας και της διαφάνειας στη διαχείριση αιτήσεων. Το σύστημα σχεδιάστηκε για να αντικαταστήσει τις χειροκίνητες, αναποτελεσματικές διαδικασίες με μια ψηφιακή πλατφόρμα που προσφέρει:

- **Αυτοματοποίηση**: Εξάλειψη των επαναλαμβανόμενων και χρονοβόρων εργασιών.
- **Ακρίβεια**: Ελαχιστοποίηση των λαθών που σχετίζονται με την ανθρώπινη παρέμβαση.
- **Διαφάνεια**: Παροχή μιας καθαρής και οργανωμένης εικόνας των αιτήσεων στους χρήστες.
- **Προσβασιμότητα**: Ευκολία στη χρήση από διαφορετικές συσκευές και πλατφόρμες.

Οι στόχοι του έργου περιλαμβάνουν:

1. Τη δημιουργία μιας φιλικής προς τον χρήστη διεπαφής που επιτρέπει στους φοιτητές να υποβάλλουν αιτήσεις γρήγορα και εύκολα.
2. Τη δυνατότητα στους διδάσκοντες να διαχειρίζονται περιόδους και αιτήσεις με τρόπο οργανωμένο.
3. Την ενσωμάτωση σύγχρονων τεχνολογιών για την αποθήκευση και ασφάλεια των δεδομένων.

Η ανάγκη για την ανάπτυξη αυτού του συστήματος προέκυψε από τις προκλήσεις που αντιμετωπίζουν οι οργανισμοί στη διαχείριση αιτήσεων. Στα παραδοσιακά συστήματα, οι αιτήσεις καταχωρούνται

χειροκίνητα, καθιστώντας τη διαδικασία ευάλωτη σε ανθρώπινα λάθη, απώλεια δεδομένων και καθυστερήσεις. Επιπλέον, η έλλειψη κεντρικής αποθήκευσης των δεδομένων καθιστά δύσκολη την παρακολούθηση της προόδου των αιτήσεων και τη δημιουργία αναφορών.

Μια από τις βασικές προκλήσεις ήταν η ανάγκη δημιουργίας ενός συστήματος που να εξυπηρετεί διαφορετικούς ρόλους χρηστών, όπως φοιτητές, διδάσκοντες και διαχειριστές, με διαφορετικά επίπεδα πρόσβασης και δικαιώματα. Επιπλέον, έπρεπε να διασφαλιστεί η ασφάλεια των δεδομένων, καθώς η διαχείριση προσωπικών πληροφοριών απαιτεί αυστηρά μέτρα προστασίας.

Μια άλλη πρόκληση ήταν η εξασφάλιση της ευχρηστίας του συστήματος, ώστε να είναι κατανοητό και εύκολο στη χρήση από άτομα με περιορισμένες τεχνικές γνώσεις. Ταυτόχρονα, έπρεπε να παρέχεται η δυνατότητα στους διαχειριστές να προσαρμόζουν το σύστημα στις δικές τους ανάγκες, χωρίς την ανάγκη εξειδικευμένων δεξιοτήτων.

Το σύστημα διαχείρισης αιτήσεων που αναπτύξαμε περιλαμβάνει αρκετές λειτουργίες που το διαφοροποιούν από παραδοσιακά συστήματα:

- **Δυναμική Διαχείριση Περιόδων**: Οι διδάσκοντες μπορούν να δημιουργούν, να τροποποιούν και να διαγράφουν περιόδους, συνδέοντας αιτήσεις φοιτητών με συγκεκριμένες περιόδους.
- **Προσαρμοσμένα Δικαιώματα Πρόσβασης**: Κάθε χρήστης έχει πρόσβαση μόνο στις λειτουργίες που σχετίζονται με τον ρόλο του.
- **Αυτοματοποιημένη Αποθήκευση Αρχείων**: Τα αρχεία που υποβάλλουν οι φοιτητές αποθηκεύονται οργανωμένα και συνδέονται με τις αντίστοιχες αιτήσεις.
- **Προβολή Προθεσμιών**: Το σύστημα ενημερώνει τους χρήστες για τις προθεσμίες υποβολής ή επεξεργασίας αιτήσεων, περιορίζοντας τις ενέργειες εκτός του χρονικού πλαισίου.

Το σύστημα βασίζεται σε σύγχρονες τεχνολογίες, όπως το **Laravel** για την ανάπτυξη του backend, το **Bootstrap** για τη δημιουργία responsive διεπαφών χρήστη και το **MySQL** για την αποθήκευση δεδομένων. Ενσωματώθηκαν βέλτιστες πρακτικές ασφάλειας, όπως κρυπτογράφηση δεδομένων και έλεγχος πρόσβασης, για την προστασία των ευαίσθητων πληροφοριών.

Το σύστημα διαθέτει τρία βασικά επίπεδα χρηστών:

- **Φοιτητές**: Υποβάλλουν και παρακολουθούν τις αιτήσεις τους.
- **Διδάσκοντες**: Διαχειρίζονται περιόδους και βλέπουν τις αιτήσεις φοιτητών στις περιόδους που έχουν πρόσβαση.
- **Διαχειριστές**: Έχουν πλήρη έλεγχο του συστήματος, συμπεριλαμβανομένης της διαχείρισης χρηστών και περιόδων.

Το έργο σχεδιάστηκε ώστε να είναι επεκτάσιμο, με δυνατότητα προσθήκης νέων λειτουργιών και προσαρμογής στις ανάγκες του οργανισμού που το χρησιμοποιεί. Το τελικό αποτέλεσμα είναι ένα αποδοτικό, φιλικό προς τον χρήστη και ασφαλές σύστημα διαχείρισης αιτήσεων που ανταποκρίνεται στις απαιτήσεις της σύγχρονης εποχής.

Η εργασία ακολουθεί μια σαφή και δομημένη προσέγγιση, με στόχο να αναλύσει το σύστημα διαχείρισης αιτήσεων που αναπτύχθηκε και τις τεχνολογίες που χρησιμοποιήθηκαν. Αποτελείται από πέντε κύρια κεφάλαια, καθώς και συνοδευτικά μέρη, όπως την περίληψη, τις ευχαριστίες και τη βιβλιογραφία.

Η Περίληψη και το Abstract παρέχουν μια συνοπτική παρουσίαση του έργου, επισημαίνοντας τον σκοπό, τις κύριες λειτουργίες και τα αποτελέσματα της εργασίας.

Το Κεφάλαιο 1ο: Εισαγωγή θέτει τα θεμέλια της εργασίας, εισάγοντας τον αναγνώστη στο πρόβλημα που επιλύει το σύστημα, ενώ περιγράφει τη δομή της εργασίας.

Το Κεφάλαιο 2ο: Ανασκόπηση Παρόμοιων Συστημάτων παρουσιάζει παρόμοιες πλατφόρμες, όπως το *Submittable* και το *OpenConf*, και αναλύει τις δυνατότητές τους. Παράλληλα, εξετάζονται τοπικές και διεθνείς εφαρμογές που σχετίζονται με διαχείριση αιτήσεων, όπως η Ηλεκτρονική Αίτηση Erasmus+ και το Google Forms.

Το Κεφάλαιο 3ο: Εργαλεία και Τεχνολογίες Ανάπτυξης του Έργου επικεντρώνεται στα τεχνολογικά εργαλεία που χρησιμοποιήθηκαν, όπως το Laravel, η PHP και η MySQL/MariaDB. Αναλύονται η ιστορία, τα πλεονεκτήματα, και οι χρήσεις κάθε εργαλείου, καθώς και η συνεισφορά τους στην υλοποίηση του έργου.

Το Κεφάλαιο 4ο: Το Σύστημα Διαχείρισης Αιτήσεων εστιάζει στην υλοποίηση του συστήματος. Παρουσιάζονται ο κώδικας, οι βασικές λειτουργίες, και εικόνες που απεικονίζουν τις ιστοσελίδες της πλατφόρμας. Επιπλέον, περιλαμβάνεται αναλυτική περιγραφή της βάσης δεδομένων και των χαρακτηριστικών ασφαλείας.

Το Κεφάλαιο 5ο: Συμπεράσματα - Βελτιώσεις συνοψίζει τα αποτελέσματα της εργασίας και προτείνει τρόπους βελτίωσης του συστήματος. Εξετάζεται η δυνατότητα επέκτασης της πλατφόρμας και οι μελλοντικές εφαρμογές της.

Η Βιβλιογραφία παραθέτει τις πηγές που χρησιμοποιήθηκαν για τη συλλογή των δεδομένων και την ανάλυση. Στο Παράρτημα Α περιλαμβάνονται πρόσθετα στοιχεία που υποστηρίζουν την εργασία.

Κεφάλαιο 2ο: Ανασκόπηση παρόμοιων συστημάτων

Η διαχείριση αιτήσεων αποτελεί κρίσιμη λειτουργία για οργανισμούς, πανεπιστήμια, και ιδρύματα που επεξεργάζονται μεγάλο αριθμό υποβολών. Στο πλαίσιο αυτό, έχουν αναπτυχθεί διάφορα συστήματα που αυτοματοποιούν και βελτιώνουν τη διαδικασία υποβολής και αξιολόγησης αιτήσεων. Αρχικά παρουσιάζονται τρία υπάρχοντα συστήματα διαχείρισης αιτήσεων: Submittable, OpenConf, και ScholarOne Manuscripts [1-3]. Αυτά τα συστήματα παρέχουν χρήσιμες πληροφορίες και παραδείγματα που μπορούν να αξιοποιηθούν για τη βελτίωση του συστήματος που αναπτύσσεται.

2.1 Submittable

Το Submittable είναι ένα καινοτόμο εργαλείο διαχείρισης αιτήσεων που χρησιμοποιείται από οργανισμούς σε διάφορους τομείς, όπως μη κερδοσκοπικούς οργανισμούς, εκδοτικούς οίκους και εταιρείες. Το σύστημα αυτό παρέχει τη δυνατότητα δημιουργίας και παραμετροποίησης αιτήσεων, προσφέροντας στους χρήστες μια εύχρηστη και ευέλικτη πλατφόρμα.

Μια από τις κύριες λειτουργίες του Submittable είναι η διαχείριση αιτήσεων μέσω ενός ενοποιημένου περιβάλλοντος που υποστηρίζει πολλαπλά επίπεδα χρηστών, όπως αιτούντες, διαχειριστές και αξιολογητές. Οι διαχειριστές μπορούν να παρακολουθούν την πρόοδο των αιτήσεων σε πραγματικό χρόνο, ενώ οι αξιολογητές έχουν τη δυνατότητα να συνεργάζονται για την επιλογή των πιο κατάλληλων υποβολών. Το Submittable επιτρέπει την ενσωμάτωση σχολίων απευθείας στις αιτήσεις και τη χρήση προσαρμοσμένων φίλτρων για την κατηγοριοποίηση και αξιολόγηση υποβολών.

Ένα από τα πιο ισχυρά χαρακτηριστικά του Submittable είναι η υποστήριξη πολλαπλών φορμά αρχείων, όπως PDF, εικόνες και βίντεο, παρέχοντας μεγάλη ευελιξία στους αιτούντες. Επιπλέον, το σύστημα χρησιμοποιεί μηχανισμούς αυτοματοποίησης για την αποστολή ειδοποιήσεων και την παραγωγή αναφορών, διευκολύνοντας τη λήψη αποφάσεων από τους διαχειριστές.

Το Submittable επικεντρώνεται στη δημιουργία μιας συνεργατικής εμπειρίας που προάγει την αποδοτικότητα και τη διαφάνεια, καθιστώντας το ιδανικό εργαλείο για οργανισμούς που επιθυμούν να διαχειριστούν αιτήσεις χρηματοδότησης, υποτροφιών ή άλλων μορφών υποβολών.

2.2 OpenConf

Το OpenConf είναι ένα δημοφιλές σύστημα διαχείρισης αιτήσεων που χρησιμοποιείται κυρίως σε ακαδημαϊκά συνέδρια, επιστημονικά περιοδικά και εκδηλώσεις. Το σύστημα αυτό έχει σχεδιαστεί για

να καλύπτει τις ανάγκες οργανισμών που διαχειρίζονται μεγάλο αριθμό επιστημονικών εργασιών και αιτήσεων.

Το OpenConf υποστηρίζει τη διαδικασία υποβολής, αξιολόγησης και αποδοχής επιστημονικών εργασιών. Μέσω του συστήματος, οι διοργανωτές συνεδρίων μπορούν να διαχειρίζονται τις υποβολές, να αναθέτουν τις εργασίες σε κριτές και να παρακολουθούν την πρόοδο της αξιολόγησης. Το σύστημα διαθέτει ευέλικτα εργαλεία διαχείρισης, όπως δυνατότητα ορισμού προθεσμιών, παρακολούθηση της κατάστασης υποβολών και δημιουργία αναφορών για τη διαδικασία επιλογής.

Επιπλέον, το OpenConf παρέχει ισχυρές δυνατότητες ασφαλείας, διασφαλίζοντας την προστασία των δεδομένων των χρηστών. Η προσαρμοστικότητα του συστήματος το καθιστά ιδανικό για μικρές και μεγάλες διοργανώσεις, καθώς επιτρέπει τη διαχείριση διαφορετικών επιπέδων πρόσβασης και την προσαρμογή της διεπαφής χρήστη στις ανάγκες του οργανισμού. Η χρήση του OpenConf συμβάλλει στη μείωση της πολυπλοκότητας της διαχείρισης αιτήσεων, επιτρέποντας στους διοργανωτές να επικεντρωθούν στην ποιότητα του περιεχομένου.

2.3 ScholarOne Manuscripts

Το ScholarOne Manuscripts είναι ένα από τα πιο εξελιγμένα συστήματα διαχείρισης αιτήσεων που χρησιμοποιούνται στον χώρο των επιστημονικών εκδόσεων. Το σύστημα αυτό παρέχει ολοκληρωμένες λύσεις για τη διαχείριση επιστημονικών άρθρων, από τη φάση της υποβολής έως την τελική δημοσίευση.

Το ScholarOne είναι σχεδιασμένο για εκδότες επιστημονικών περιοδικών και υποστηρίζει διαδικασίες όπως η αξιολόγηση από κριτές, η επικοινωνία μεταξύ συγγραφέων και εκδοτών, και η παρακολούθηση της προόδου των υποβολών. Ένα από τα βασικά πλεονεκτήματα του συστήματος είναι η δυνατότητα διαχείρισης μεγάλου όγκου δεδομένων, καθώς και η αυτοματοποίηση διαδικασιών, όπως η ανάθεση κριτών και η αποστολή ειδοποιήσεων.

Η πλατφόρμα ScholarOne δίνει έμφαση στην ασφάλεια των δεδομένων και τη συμμόρφωση με τα πρότυπα διαχείρισης επιστημονικών πληροφοριών. Παράλληλα, προσφέρει αναλυτικά εργαλεία για τη δημιουργία αναφορών και στατιστικών στοιχείων, διευκολύνοντας τη λήψη αποφάσεων από τους εκδότες. Η χρήση του ScholarOne έχει συμβάλει σημαντικά στη βελτίωση της αποδοτικότητας και της διαφάνειας στον χώρο των επιστημονικών δημοσιεύσεων.

2.4 Google Forms

Το Google Forms είναι ένα από τα πιο δημοφιλή και προσβάσιμα εργαλεία για τη δημιουργία φορμών, συλλογή δεδομένων και οργάνωση διαδικασιών υποβολής αιτήσεων. Αναπτύχθηκε από την Google και αποτελεί μέρος της οικογένειας εργαλείων Google Workspace. Η απλότητα, η δωρεάν πρόσβαση και οι δυνατότητες εξαγωγής δεδομένων το καθιστούν ιδανική επιλογή για άτομα, ομάδες και οργανισμούς που αναζητούν μια γρήγορη και αποτελεσματική λύση.[4]

Το Google Forms προσφέρει μια σειρά από χαρακτηριστικά που το κάνουν ιδανικό για υποβολή και αξιολόγηση αιτήσεων:

Δωρεάν Χρήση: Δεν απαιτείται συνδρομή ή πληρωμή για βασική χρήση.

Εύκολη Δημιουργία Φορμών: Με έναν απλό και φιλικό προς τον χρήστη διαχειριστή, μπορείς να δημιουργήσεις φόρμες σε λίγα λεπτά.

Ποικιλία Ερωτήσεων: Υποστηρίζει πολλούς τύπους ερωτήσεων, όπως πολλαπλής επιλογής, κλίμακας, κειμένου, επιλογής ημερομηνίας κ.ά.

Προσαρμογή: Δυνατότητα προσθήκης εικόνων, βίντεο και ενσωμάτωσης λογότυπων για εταιρική ταυτότητα.

Αυτοματοποιημένη Συλλογή Δεδομένων: Οι απαντήσεις αποθηκεύονται αυτόματα σε Google Sheets για εύκολη ανάλυση και εξαγωγή.

Το Google Forms είναι ιδανικό για τη διαδικασία υποβολής αιτήσεων, καθώς επιτρέπει:

Δημιουργία Φορμών για Αιτήσεις: Μπορείς να δημιουργήσεις φόρμες για εγγραφές, αιτήσεις συμμετοχής, υποβολή προτάσεων κ.ά.

Πρόσβαση από Οπουδήποτε: Οι χρήστες μπορούν να υποβάλουν αιτήσεις από οποιαδήποτε συσκευή (υπολογιστή, κινητό, tablet) με σύνδεση στο διαδίκτυο.

Προσαρμοσμένα Πεδία: Προσθήκη υποχρεωτικών ή προαιρετικών πεδίων για συλλογή συγκεκριμένων πληροφοριών.

Αυτόματη Ενημέρωση: Οι υποβολές εμφανίζονται σε πραγματικό χρόνο, χωρίς την ανάγκη χειροκίνητης ενημέρωσης.

Μια από τις σημαντικότερες δυνατότητες του Google Forms είναι η εξαγωγή και ανάλυση των δεδομένων:

Αυτόματη Εξαγωγή σε Google Sheets: Οι απαντήσεις αποθηκεύονται σε ένα Google Spreadsheet, όπου μπορούν να ταξινομηθούν, να αναλυθούν και να εξαχθούν σε μορφές όπως Excel ή CSV.

Γραφήματα και Στατιστικά: Το Google Forms δημιουργεί αυτόματα γραφήματα και στατιστικά για μια γρήγορη επισκόπηση των αποτελεσμάτων.

Φιλτράρισμα και Ταξινόμηση: Μέσω του Google Sheets, μπορείς να φιλτράρεις και να ταξινομήσεις τις απαντήσεις για καλύτερη οργάνωση.

Το Google Forms προσφέρει βασικές δυνατότητες αξιολόγησης, ιδιαίτερα χρήσιμες για απλές διαδικασίες:

Αυτόματη Βαθμολόγηση: Για ερωτήσεις πολλαπλής επιλογής ή κλίμακας, μπορείς να ορίσεις σωστές απαντήσεις και να δημιουργήσεις ένα σύστημα αυτόματης βαθμολόγησης.

Προσωποποιημένη Ανατροφοδότηση: Μπορείς να προσθέσεις μηνύματα ανατροφοδότησης για τους χρήστες με βάση τις απαντήσεις τους.

Ειδοποιήσεις: Οι ειδοποιήσεις μέσω email επιτρέπουν την άμεση ενημέρωση για νέες υποβολές.

Πλεονεκτήματα

Χωρίς Κόστος: Δεν απαιτείται οικονομική επένδυση για βασική χρήση.

Απλότητα: Δεν απαιτείται τεχνική γνώση για τη δημιουργία και διαχείριση φορμών.

Ομαδική Συνεργασία: Πολλοί χρήστες μπορούν να συνεργαστούν στην ίδια φόρμα ή στα δεδομένα.

Ευελιξία: Μπορεί να χρησιμοποιηθεί για ποικίλες ανάγκες, από εκπαιδευτικές έως επαγγελματικές.

Περιορισμοί

Περιορισμένη Προσαρμογή: Δεν προσφέρει τόσο προηγμένες δυνατότητες προσαρμογής όσο άλλα εμπορικά εργαλεία.

Βασική Αξιολόγηση: Δεν υποστηρίζει πολύπλοκες διαδικασίες αξιολόγησης, όπως αυτές που προσφέρουν ειδικά συστήματα.

Εξάρτηση από Διαδίκτυο: Απαιτεί σύνδεση στο διαδίκτυο για τη δημιουργία και υποβολή φορμών.

Το Google Forms είναι μια ιδανική λύση για όσους αναζητούν μια γρήγορη, δωρεάν και απλή πλατφόρμα για υποβολή και αξιολόγηση αιτήσεων. Ενώ δεν προσφέρει προηγμένες δυνατότητες, η ευκολία χρήσης και η ενσωμάτωση με άλλα εργαλεία της Google (όπως το Google Sheets) το καθιστούν μια πολύ καλή επιλογή για βασικές ανάγκες. Για πιο σύνθετες διαδικασίες, μπορεί να απαιτηθεί η χρήση ειδικών πλατφορμών ή η προσαρμογή του Google Forms με πρόσθετα εργαλεία.

2.5 Ηλεκτρονική Υπηρεσία Υποβολής Αίτησης Συνταξιοδότησης και Εφάπαξ Παροχής

Ο e-ΕΦΚΑ προσφέρει μια προηγμένη πλατφόρμα για την υποβολή αιτήσεων συνταξιοδότησης και εφάπαξ παροχής. Μέσω αυτής, οι ασφαλισμένοι μπορούν να καταθέσουν τα αιτήματά τους ηλεκτρονικά, μειώνοντας τις επισκέψεις στα φυσικά καταστήματα και εξασφαλίζοντας άμεση επεξεργασία. Το σύστημα χρησιμοποιεί τον μηχανισμό «ΑΤΛΑΣ», ο οποίος συλλέγει αυτόματα δικαιολογητικά από διασυνδεδεμένες βάσεις δεδομένων, διευκολύνοντας τη διαδικασία απονομής. [5]

2.6 Εθνική Πύλη Αναπηρίας

Η Εθνική Πύλη Αναπηρίας επιτρέπει στους πολίτες να υποβάλλουν αιτήσεις για πιστοποίηση αναπηρίας μέσω του Κέντρου Πιστοποίησης Αναπηρίας (ΚΕΠΑ). [6] Οι ενδιαφερόμενοι αυθεντικοποιούνται μέσω της πλατφόρμας gov.gr και συμπληρώνουν ηλεκτρονικά την αίτηση χωρίς την ανάγκη φυσικής παρουσίας. Το σύστημα διασυνδέεται με άλλες υπηρεσίες, διασφαλίζοντας την αξιοπιστία των δεδομένων και μειώνοντας τη γραφειοκρατία για τα άτομα με αναπηρία.

2.7 Ηλεκτρονική Αίτηση Erasmus+ για Σπουδές

Το πρόγραμμα Erasmus+ επιτρέπει στους φοιτητές να υποβάλλουν ηλεκτρονικές αιτήσεις για σπουδές σε συνεργαζόμενα πανεπιστήμια του εξωτερικού. Το Αριστοτέλειο Πανεπιστήμιο Θεσσαλονίκης (ΑΠΘ) διαθέτει μια ειδική διαδικτυακή πλατφόρμα, όπου οι ενδιαφερόμενοι συμπληρώνουν την αίτηση εκδήλωσης ενδιαφέροντος, επισυνάπτουν δικαιολογητικά και ενημερώνονται για την πρόοδό της. Με αυτόν τον τρόπο, η διαδικασία γίνεται πιο διαφανής και εύχρηστη για τους φοιτητές.[7]

2.8 Ηλεκτρονική Αίτηση Erasmus+ για Πρακτική Άσκηση

Παρόμοια με τις αιτήσεις για σπουδές, το πρόγραμμα Erasmus+ για πρακτική άσκηση διαθέτει ηλεκτρονική διαδικασία υποβολής αιτήσεων. Οι φοιτητές που επιθυμούν να πραγματοποιήσουν πρακτική άσκηση σε ευρωπαϊκές εταιρείες ή οργανισμούς υποβάλλουν τα δικαιολογητικά τους μέσω της διαδικτυακής πλατφόρμας του πανεπιστημίου τους. Η ηλεκτρονική διαχείριση των αιτήσεων διευκολύνει την αξιολόγηση και επιλογή των υποψηφίων, επιταχύνοντας τη συνολική διαδικασία.[8]

Κεφάλαιο 3ο: Εργαλεία και Τεχνολογίες Ανάπτυξης του Έργου

Στη σύγχρονη εποχή, η τεχνολογία αποτελεί κομμάτι της καθημερινότητάς μας, επηρεάζοντας όλους τους τομείς της ζωής, από την εκπαίδευση και την εργασία έως την υγεία και την ψυχαγωγία. Η ανάπτυξη καινοτόμων συστημάτων βασισμένων σε τεχνολογίες αιχμής έχει καταστήσει δυνατή την αυτοματοποίηση και βελτίωση διαδικασιών που παλαιότερα απαιτούσαν σημαντική ανθρώπινη παρέμβαση. Ένα χαρακτηριστικό παράδειγμα αυτής της εξέλιξης είναι η δημιουργία συστημάτων διαχείρισης αιτήσεων, τα οποία απλοποιούν και επιταχύνουν τη διαχείριση δεδομένων παρέχοντας αποτελεσματικότητα και διαφάνεια.

Η ανάπτυξη του παρόντος έργου απαιτούσε τη χρήση προηγμένων τεχνολογιών και εργαλείων για την επίτευξη ενός λειτουργικού και φιλικού προς τον χρήστη συστήματος. Η επιλογή των κατάλληλων εργαλείων και τεχνολογιών δεν έγινε τυχαία αλλά βασίστηκε σε κριτήρια όπως η αξιοπιστία, η ασφάλεια, η επεκτασιμότητα και η ευκολία χρήσης. Οι τεχνολογίες που επιλέχθηκαν, όπως το Laravel για το backend, το Bootstrap για την ανάπτυξη responsive διεπαφών χρήστη, και το MySQL για τη διαχείριση δεδομένων, συνδυάζονται για να δημιουργήσουν μια ισχυρή και ευέλικτη πλατφόρμα.

Στο παρόν κεφάλαιο θα εξετάσουμε τα εργαλεία και τις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του συστήματος. Θα αναλύσουμε πώς αυτά συνέβαλαν στην ανάπτυξη του έργου, εξασφαλίζοντας την αποτελεσματικότητα, την ασφάλεια και τη χρηστικότητα της πλατφόρμας. Επιπλέον, θα περιγράψουμε πώς οι επιμέρους τεχνολογίες συνεργάζονται για να δημιουργήσουν μια ολοκληρωμένη λύση που εξυπηρετεί τις ανάγκες των χρηστών και διασφαλίζει την ομαλή λειτουργία του συστήματος.

3.1 Laravel

Το Laravel είναι ένα από τα πιο δημοφιλή και ευρέως χρησιμοποιούμενα πλαίσια ανάπτυξης εφαρμογών ιστού (web frameworks) που βασίζονται στη γλώσσα προγραμματισμού PHP. Δημιουργήθηκε από τον Taylor Otwell και παρουσιάστηκε για πρώτη φορά το 2011. Ο βασικός στόχος του Laravel ήταν να διευκολύνει την ανάπτυξη σύνθετων εφαρμογών, προσφέροντας στους προγραμματιστές ένα εργαλείο με κομψή σύνταξη, ισχυρές λειτουργικότητες και απλοποιημένη προσέγγιση. Ενώ η PHP είχε δεχτεί κριτική για την έλλειψη ενός συνεκτικού τρόπου ανάπτυξης εφαρμογών, το Laravel έφερε μια αισιοδοξία με την καθοδήγηση σε καλύτερες πρακτικές ανάπτυξης. [9-14]

Η αρχική έκδοση του Laravel (1.0) δημιουργήθηκε ως μια εναλλακτική λύση στο CodeIgniter, που εκείνη την εποχή δεν υποστήριζε χαρακτηριστικά όπως το σύστημα δρομολόγησης (routing) με δυνατότητα RESTful αρχιτεκτονικής. Από τότε, το Laravel εξελίχθηκε μέσα από αρκετές εκδόσεις,

προσθέτοντας σημαντικά χαρακτηριστικά όπως το Artisan CLI, το Eloquent ORM, το Blade templating engine και την υποστήριξη για middleware. Οι εκδόσεις από την 5.0 και μετά παρουσίασαν βελτιώσεις στην απόδοση και εισήγαγαν σημαντικά εργαλεία, όπως το Horizon (για διαχείριση ουρών εργασιών), το Nova (για διαχείριση backend) και το Vapor (για serverless ανάπτυξη).

Το Laravel χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών διαχείρισης περιεχομένου, ηλεκτρονικού εμπορίου, API, συστημάτων κρατήσεων και άλλων πολύπλοκων λύσεων. Ορισμένα από τα κύρια πλεονεκτήματά του περιλαμβάνουν:

1. **Κομψή Σύνταξη**: Ο προσεκτικά σχεδιασμένος κώδικας του Laravel διευκολύνει την ανάγνωση και συντήρηση. Το πλαίσιο επιτρέπει στους προγραμματιστές να δημιουργούν λειτουργικότητες γρήγορα χωρίς να θυσιάζουν τη δομή και την ποιότητα.
2. **Πλούσια Χαρακτηριστικά**: Το Laravel περιλαμβάνει ενσωματωμένη υποστήριξη για δρομολόγηση, έλεγχο ταυτότητας, διαχείριση συνεδριών, επαλήθευση δεδομένων και πολλά άλλα. Αυτά τα χαρακτηριστικά μειώνουν την ανάγκη για εξωτερικές βιβλιοθήκες και εργαλεία.
3. **Eloquent ORM**: Το Eloquent είναι ένα Object-Relational Mapping εργαλείο που παρέχει έναν απλό και εύκολο τρόπο διαχείρισης βάσεων δεδομένων. Οι προγραμματιστές μπορούν να γράφουν ερωτήματα χωρίς τη χρήση ακατέργαστου SQL.
4. **Blade Templating Engine**: Το Blade επιτρέπει τη χρήση απλής και αποτελεσματικής σύνταξης για την ανάπτυξη διεπαφών χρηστών, με την υποστήριξη δεδομένων δυναμικού περιεχομένου και επαναχρησιμοποιούμενων στοιχείων.
5. **Artisan CLI**: Αυτό το εργαλείο γραμμής εντολών επιτρέπει την αυτοματοποίηση συχνών εργασιών, όπως η δημιουργία μοντέλων, ο έλεγχος βάσεων δεδομένων και η ανάπτυξη εφαρμογών.
6. **Διεθνής Κοινότητα**: Με μια μεγάλη και ενεργή κοινότητα προγραμματιστών, το Laravel προσφέρει υποστήριξη μέσω φόρουμ, τεκμηρίωσης και online μαθημάτων.
7. **Υποστήριξη Microservices**: Μέσω της ευελιξίας του, το Laravel μπορεί να χρησιμοποιηθεί για την ανάπτυξη μικροϋπηρεσιών (microservices) που απαιτούν αποδοτικότητα και ταχύτητα.

Παρότι το Laravel προσφέρει πολλά πλεονεκτήματα, παρουσιάζει και ορισμένες αδυναμίες:

- **Καμπύλη Εκμάθησης**: Οι αρχάριοι προγραμματιστές μπορεί να δυσκολευτούν να κατανοήσουν όλες τις δυνατότητες του Laravel και τα χαρακτηριστικά του.

- **Απόδοση**: Αν και το Laravel είναι βελτιστοποιημένο για πολλές περιπτώσεις χρήσης, η απόδοση μπορεί να επηρεαστεί όταν χρησιμοποιείται για εφαρμογές που απαιτούν εξαιρετικά υψηλές ταχύτητες.
- **Μεγέθυνση Κώδικα**: Η χρήση πλούσιων χαρακτηριστικών μπορεί να οδηγήσει σε αύξηση του κώδικα, κάτι που καθιστά απαραίτητη τη σωστή δομή του έργου.

Το Laravel είναι ένα από τα πιο δημοφιλή PHP frameworks παγκοσμίως. Σύμφωνα με το GitHub, διαθέτει εκατομμύρια downloads και είναι το πιο αστεράτο PHP framework στην πλατφόρμα. Με βάση την πλατφόρμα Packagist, καταγράφονται πάνω από 100 εκατομμύρια λήψεις κάθε χρόνο, γεγονός που αποδεικνύει τη δημοτικότητά του.

Στο συγκεκριμένο έργο, το Laravel επιλέχθηκε λόγω της ευελιξίας και των δυνατοτήτων του. Η ενσωματωμένη υποστήριξη για έλεγχο ταυτότητας και εξουσιοδότησης διευκόλυνε την εφαρμογή ασφαλών μηχανισμών πρόσβασης. Επίσης, το σύστημα δρομολόγησης του Laravel επέτρεψε την απλή αλλά αποτελεσματική διαχείριση των αιτήσεων, ενώ το Eloquent ORM διευκόλυνε τη διαχείριση δεδομένων.

Ειδικότερα, το Laravel χρησιμοποιήθηκε για:

- **Διαχείριση Χρηστών**: Υλοποιήθηκε σύστημα αυθεντικοποίησης και εξουσιοδότησης για φοιτητές, εκπαιδευτικούς και διαχειριστές.
- **Διαχείριση Περιεχομένου**: Δημιουργήθηκαν λειτουργίες για την υποβολή, προβολή και επεξεργασία αιτήσεων.
- **Βάση Δεδομένων**: Το Eloquent ORM επέτρεψε τη δημιουργία ενός δυναμικού και αποτελεσματικού μοντέλου δεδομένων.

3.1.1 Τι είναι το Laravel;

Το Laravel είναι ένα PHP framework που διευκολύνει την ανάπτυξη εφαρμογών web, παρέχοντας εργαλεία και δομές για ταχύτερη και οργανωμένη εργασία. Χρησιμοποιεί την αρχιτεκτονική MVC (Model-View-Controller), προσφέροντας δυνατότητες όπως διαχείριση βάσεων δεδομένων, δρομολόγηση (routing), έλεγχο ταυτότητας (authentication), και templating.

Απλό Παράδειγμα Χρήσης Laravel

Ας υποθέσουμε ότι θέλουμε να δημιουργήσουμε μια απλή εφαρμογή "Διαχείρισης Βιβλίων" με τις βασικές λειτουργίες:

1. Προσθήκη βιβλίου.
2. Προβολή όλων των βιβλίων.

Βήμα 1: Δημιουργία Project Laravel

Εκτελούμε την εντολή:

composer create-project laravel/laravel BookManager

Βήμα 2: Δημιουργία Βάσης Δεδομένων

Δημιουργούμε έναν πίνακα books με τις στήλες:

- id (αυτόματη αύξηση),
- title (τίτλος βιβλίου),
- author (συγγραφέας),
- published_date (ημερομηνία έκδοσης).

```
CREATE TABLE books (
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255) NOT NULL,
    published_date DATE
);
```

Βήμα 3: Σύνδεση Laravel με τη Βάση Δεδομένων

Στο αρχείο .env, προσθέτουμε τις ρυθμίσεις:

```
DB_CONNECTION=mysql
DB_HOST=127.0.0.1
DB_PORT=3306
```

```
DB_DATABASE=book_manager
DB_USERNAME=root
DB_PASSWORD=your_password
```

Βήμα 4: Δημιουργία Μοντέλου

Το μοντέλο συνδέεται με τον πίνακα books:

```
php artisan make:model Book
```

Στο app/Models/Book.php:

```
namespace App\Models;
use Illuminate\Database\Eloquent\Factories\HasFactory;
use Illuminate\Database\Eloquent\Model;
class Book extends Model
{
    use HasFactory;

    // Ορισμός των στηλών που επιτρέπεται να εισάγουμε/ενημερώνουμε
    protected $fillable = ['title', 'author', 'published_date'];
}
```

Βήμα 5: Δημιουργία Controller

```
php artisan make:controller BookController
```

Στο app/Http/Controllers/BookController.php:

```
namespace App\Http\Controllers;
use App\Models\Book;
use Illuminate\Http\Request;

class BookController extends Controller
{
    // Προβολή όλων των βιβλίων
    public function index()
    {
        $books = Book::all();
```

```
        return view('books.index', ['books' => $books]);
    }

    // Προβολή φόρμας προσθήκης βιβλίου
    public function create()
    {
        return view('books.create');
    }

    // Αποθήκευση νέου βιβλίου
    public function store(Request $request)
    {
        $validated = $request->validate([
            'title' => 'required|string|max:255',
            'author' => 'required|string|max:255',
            'published_date' => 'required|date',
        ]);

        Book::create($validated);

        return redirect()->route('books.index')->with('success', 'Το βιβλίο προστέθηκε επιτυχώς!');
    }
}
```

Βήμα 6: Δημιουργία Routes

Στο αρχείο routes/web.php:

```
use App\Http\Controllers\BookController;

Route::get('/books', [BookController::class, 'index'])->name('books.index');

Route::get('/books/create', [BookController::class, 'create'])->name('books.create');

Route::post('/books', [BookController::class, 'store'])->name('books.store');
```

Βήμα 7: Δημιουργία Views

Στο φάκελο resources/views/books, δημιουργούμε δύο αρχεία:

1. index.blade.php:

```
<!DOCTYPE html>
<html lang="en">
```

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Λίστα Βιβλίων</title>
</head>
<body>
    <h1>Λίστα Βιβλίων</h1>
    <a href="{{ route('books.create') }}">Προσθήκη Νέου Βιβλίου</a>
    <ul>
        @foreach ($books as $book)
            <li>{{ $book->title }} - {{ $book->author }} ({{ $book->published_date }})</li>
        @endforeach
    </ul>
</body>
</html>
```

2. create.blade.php:

```
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Προσθήκη Βιβλίου</title>
</head>
<body>
    <h1>Προσθήκη Νέου Βιβλίου</h1>
    <form action="{{ route('books.store') }}" method="POST">
        @csrf
        <label for="title">Τίτλος:</label>
        <input type="text" id="title" name="title" required><br><br>
        <label for="author">Συγγραφέας:</label>
        <input type="text" id="author" name="author" required><br><br>
        <label for="published_date">Ημερομηνία Έκδοσης:</label>
        <input type="date" id="published_date" name="published_date" required><br><br>
        <button type="submit">Αποθήκευση</button>
    </form>
</body>
</html>
```

Πώς Λειτουργεί:

1. **Δημιουργία Βιβλίου**: Ο χρήστης επισκέπτεται το /books/create και εισάγει τα στοιχεία του βιβλίου. Το Laravel επαληθεύει τα δεδομένα και αποθηκεύει το βιβλίο στη βάση δεδομένων.
2. **Προβολή Βιβλίων**: Στη σελίδα /books, εμφανίζονται όλα τα βιβλία που υπάρχουν στη βάση δεδομένων.

3.2 PHP

Η PHP (Hypertext Preprocessor) είναι μια από τις πιο δημοφιλείς γλώσσες προγραμματισμού για την ανάπτυξη δυναμικών ιστοσελίδων και εφαρμογών web. Από τη δημιουργία της, έχει επηρεάσει σημαντικά τον τρόπο με τον οποίο σχεδιάζονται και λειτουργούν οι ιστοσελίδες, προσφέροντας ευελιξία και ισχυρά εργαλεία για την ανάπτυξη web εφαρμογών. [15-19]

Η PHP δημιουργήθηκε το 1994 από τον Rasmus Lerdorf ως μια σειρά από εργαλεία CGI (Common Gateway Interface) που αναπτύχθηκαν σε C. Αρχικά ονομάστηκε "Personal Home Page Tools", καθώς χρησιμοποιήθηκε από τον δημιουργό της για την παρακολούθηση των επισκεπτών στον προσωπικό του ιστότοπο. Σταδιακά, η PHP εξελίχθηκε σε μια ισχυρή γλώσσα προγραμματισμού και το 1995 κυκλοφόρησε η πρώτη επίσημη έκδοση.

Κατά τη διάρκεια των ετών, η PHP γνώρισε σημαντική ανάπτυξη. Με την κυκλοφορία της PHP 3.0 το 1997, η γλώσσα απέκτησε μια νέα μηχανή, την Zend Engine, και μετονομάστηκε σε "PHP: Hypertext Preprocessor". Από τότε, η PHP συνεχώς βελτιώνεται, με την πιο πρόσφατη έκδοση (PHP 8.0) να προσφέρει νέα χαρακτηριστικά, όπως τις Union Types, την JIT Compiler, και τις Attributes.

Χρήσεις και Εφαρμογές της PHP

Η PHP χρησιμοποιείται κυρίως για την ανάπτυξη εφαρμογών web. Είναι κατάλληλη για την κατασκευή ιστοσελίδων που απαιτούν διαδραστικότητα και δυναμικό περιεχόμενο. Χρησιμοποιείται σε διάφορες εφαρμογές, όπως:

Συστήματα Διαχείρισης Περιεχομένου (CMS): Πλατφόρμες όπως το WordPress, το Joomla και το Drupal είναι κατασκευασμένες με PHP, επιτρέποντας την εύκολη δημιουργία και διαχείριση περιεχομένου.

Ηλεκτρονικό Εμπόριο: Πλατφόρμες όπως το Magento και το OpenCart βασίζονται στην PHP, παρέχοντας λύσεις για τη δημιουργία ηλεκτρονικών καταστημάτων.

Πλατφόρμες Κοινωνικής Δικτύωσης: Δημοφιλή κοινωνικά δίκτυα, όπως το Facebook, ξεκίνησαν να κατασκευάζονται με PHP.

Web Portals και Εφαρμογές: Η PHP χρησιμοποιείται για την ανάπτυξη εταιρικών portals, συστημάτων κρατήσεων, και εκπαιδευτικών πλατφορμών.

Στατιστικά Χρήσης

Σύμφωνα με δεδομένα από το W3Techs, η PHP παραμένει μια από τις πιο δημοφιλείς γλώσσες για την ανάπτυξη ιστοσελίδων. Περίπου το 77.5% των ιστοσελίδων παγκοσμίως χρησιμοποιούν PHP ως server-side γλώσσα. Τα συστήματα CMS όπως το WordPress, που αποτελεί την πλατφόρμα περίπου του 40% όλων των ιστοσελίδων, βασίζονται αποκλειστικά στην PHP.

Επιπλέον, η κοινότητα της PHP είναι ιδιαίτερα ενεργή, με χιλιάδες διαθέσιμα πακέτα και βιβλιοθήκες μέσω του Composer, που είναι ο διαχειριστής εξαρτήσεων της PHP.

Πλεονεκτήματα της PHP

Ανοιχτού Κώδικα: Η PHP είναι δωρεάν για χρήση και προσφέρει πρόσβαση στον κώδικα της, επιτρέποντας τη συνεχή βελτίωση και την ανάπτυξη από την κοινότητα.

Ευκολία Μάθησης: Η PHP έχει μια εύκολη και κατανοητή σύνταξη, γεγονός που την καθιστά ιδανική για νέους προγραμματιστές.

Πολλαπλές Βιβλιοθήκες και Πλαίσια: Υπάρχουν πολλά frameworks (όπως το Laravel, το Symfony και το CodeIgniter) που βασίζονται στην PHP, διευκολύνοντας την ανάπτυξη εφαρμογών.

Ευελιξία: Μπορεί να χρησιμοποιηθεί σε διαφορετικά λειτουργικά συστήματα (Windows, Linux, macOS) και να ενσωματωθεί με διάφορες βάσεις δεδομένων.

Ενεργή Κοινότητα: Υπάρχει μεγάλη κοινότητα προγραμματιστών που παρέχει υποστήριξη και προσφέρει έτοιμες λύσεις μέσω φόρουμ και διαδικτυακών εργαλείων.

Μειονεκτήματα της PHP

Ασφάλεια: Αν και η PHP έχει εξελιχθεί, οι κακές πρακτικές προγραμματισμού μπορούν να οδηγήσουν σε ευπάθειες, όπως επιθέσεις SQL Injection ή Cross-Site Scripting (XSS).

Αργή Εκτέλεση: Σε σύγκριση με άλλες γλώσσες, όπως το Node.js ή το Python, η PHP μπορεί να είναι πιο αργή σε ορισμένες περιπτώσεις.

Ασυνεπής Σύνταξη: Λόγω της συνεχούς εξέλιξης, η PHP έχει αποκτήσει κάποιες ασυνέπειες στη σύνταξη, οι οποίες μπορεί να προκαλέσουν σύγχυση.

Γιατί Χρησιμοποιήθηκε σε Αυτό το Έργο

Η PHP επιλέχθηκε για την ανάπτυξη του συγκεκριμένου έργου λόγω της ευρείας χρήσης της για την ανάπτυξη εφαρμογών web. Το Laravel, ένα από τα δημοφιλέστερα frameworks της PHP, προσφέρει οργανωμένη ανάπτυξη μέσω της αρχιτεκτονικής MVC και ενσωματώνει ισχυρά χαρακτηριστικά, όπως η επαλήθευση χρηστών, η δημιουργία δρομολογίων, και η διαχείριση βάσεων δεδομένων. Ειδικά για το σύστημα διαχείρισης αιτήσεων, η PHP ήταν ιδανική, καθώς επέτρεψε τη σύνδεση με τη βάση δεδομένων MySQL και την υλοποίηση διαδραστικών διεπαφών χρήστη.

Η δυνατότητα χρήσης έτοιμων βιβλιοθηκών και εργαλείων μείωσε τον χρόνο ανάπτυξης, ενώ η μεγάλη κοινότητα της PHP πρόσφερε εύκολη πρόσβαση σε παραδείγματα και τεκμηρίωση για την υλοποίηση πολύπλοκων λειτουργιών.

3.3 MySQL Maria DB

Η MySQL και η MariaDB είναι δύο από τις πιο δημοφιλείς συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (Relational Database Management Systems - RDBMS) στον κόσμο. Αποτελούν τη ραχοκοκαλιά πολλών εφαρμογών web, επιτρέποντας την αποθήκευση, ανάκτηση και διαχείριση δεδομένων με αποτελεσματικό τρόπο. Και οι δύο λύσεις είναι ευρέως χρησιμοποιούμενες, ιδιαίτερα σε εφαρμογές που απαιτούν υψηλή απόδοση, αξιοπιστία και ευκολία χρήσης. [19-21]

Η MySQL δημιουργήθηκε το 1995 από την εταιρεία MySQL AB, που ιδρύθηκε από τον Michael Widenius, τον Allan Larsson και τον David Axmark. Το όνομά της προέρχεται από την κόρη του Widenius, την My. Η MySQL γρήγορα έγινε δημοφιλής, καθώς προσέφερε έναν γρήγορο και αποδοτικό τρόπο αποθήκευσης δεδομένων για web εφαρμογές. Το 2008, η MySQL εξαγοράστηκε από τη Sun Microsystems, η οποία αργότερα εξαγοράστηκε από την Oracle Corporation.

Η MariaDB αναπτύχθηκε το 2009 από τον Michael Widenius και την κοινότητα της MySQL ως εναλλακτική λύση ανοιχτού κώδικα μετά την εξαγορά της MySQL από την Oracle. Το όνομά της προέρχεται από τη δεύτερη κόρη του Widenius, τη Maria. Η MariaDB διατηρεί συμβατότητα με τη MySQL, επιτρέποντας την εύκολη μετάβαση από το ένα σύστημα στο άλλο.

Χρήσεις και Εφαρμογές

Οι βάσεις δεδομένων MySQL και MariaDB χρησιμοποιούνται σε πολλούς τομείς και εφαρμογές:

- Web εφαρμογές: Πολλές πλατφόρμες και CMS όπως το WordPress, το Joomla και το Drupal χρησιμοποιούν MySQL ή MariaDB για την αποθήκευση δεδομένων.

- Ηλεκτρονικό εμπόριο: Εφαρμογές όπως το Magento και το Shopify βασίζονται σε αυτές τις βάσεις δεδομένων για τη διαχείριση προϊόντων, πελατών και παραγγελιών.
- Πλατφόρμες κοινωνικής δικτύωσης: Πλατφόρμες όπως το Facebook (σε παλαιότερες εκδόσεις) χρησιμοποιούν MySQL για την αποθήκευση δεδομένων χρηστών και αναρτήσεων.
- Εταιρικές εφαρμογές: Συστήματα ERP, CRM και άλλα εργαλεία διαχείρισης χρησιμοποιούν αυτές τις βάσεις δεδομένων για την αποθήκευση και επεξεργασία επιχειρηματικών δεδομένων.

Στατιστικά Χρήσης

Η MySQL είναι μια από τις πιο δημοφιλείς βάσεις δεδομένων παγκοσμίως, με εκτεταμένη χρήση σε εφαρμογές web και πλατφόρμες CMS. Σύμφωνα με έρευνες του DB-Engines, η MySQL συγκαταλέγεται σταθερά στις κορυφαίες βάσεις δεδομένων, μαζί με συστήματα όπως το PostgreSQL και το Microsoft SQL Server.

Η MariaDB, παρότι πιο πρόσφατη, έχει αναπτυχθεί ραγδαία και υιοθετείται από πολλούς οργανισμούς, ιδιαίτερα λόγω του ανοιχτού κώδικά της και της κοινότητας που τη στηρίζει. Χρησιμοποιείται ευρέως από εταιρείες όπως η Wikipedia και η Red Hat.

Πλεονεκτήματα

MySQL:

- **Ευκολία χρήσης**: Είναι εύκολη στη ρύθμιση και στη διαχείριση.
- **Απόδοση**: Ιδανική για εφαρμογές που απαιτούν γρήγορη ανάκτηση δεδομένων.
- **Πολλαπλή Υποστήριξη**: Υποστηρίζει πολλές γλώσσες και πλατφόρμες.

MariaDB:

- **Ανοιχτός κώδικας**: Πλήρως ανοιχτού κώδικα χωρίς περιορισμούς.
- **Βελτιωμένες δυνατότητες**: Περιλαμβάνει νέες λειτουργίες, όπως καλύτερη υποστήριξη JSON και νέους μηχανισμούς αποθήκευσης δεδομένων.
- **Συμβατότητα**: Είναι πλήρως συμβατή με τη MySQL, επιτρέποντας την εύκολη μετάβαση.

Μειονεκτήματα

MySQL:

- **Άδεια χρήσης**: Κάποιες εκδόσεις υπόκεινται σε περιορισμούς λόγω της ιδιοκτησίας της Oracle.
- **Κοινότητα**: Υπάρχουν ανησυχίες για τη μελλοντική ανάπτυξη λόγω της ιδιοκτησίας της Oracle.

MariaDB:

- **Υποστήριξη**: Λιγότερο ευρεία υποστήριξη από εργαλεία τρίτων σε σχέση με τη MySQL.
- **Επικέντρωση**: Παρότι εξελίσσεται, δεν έχει την ίδια εμπορική προώθηση με τη MySQL.

Γιατί Χρησιμοποιήθηκαν στο Έργο

Η MySQL χρησιμοποιήθηκε στο συγκεκριμένο έργο για τη διαχείριση των δεδομένων των αιτήσεων, των χρηστών και των περιόδων. Επιλέχθηκε για την ευκολία ενσωμάτωσης με το framework Laravel και για την αξιοπιστία της στη διαχείριση μεγάλων όγκων δεδομένων. Επίσης, λόγω της ευρείας υποστήριξης από εργαλεία ανάπτυξης και διαχείρισης, η MySQL κατέστη η ιδανική επιλογή για τη βάση δεδομένων του έργου μας.

Η MariaDB θα μπορούσε επίσης να χρησιμοποιηθεί, καθώς προσφέρει βελτιώσεις απόδοσης και είναι πλήρως συμβατή με τη MySQL, δίνοντας στους προγραμματιστές περισσότερες επιλογές για επέκταση της εφαρμογής.

Κεφάλαιο 4ο: Το σύστημα διαχείρισης αιτήσεων

4.1 Περιγραφή

Το σύστημα διαχείρισης αιτήσεων που αναπτύξαμε αποτελεί ένα ολοκληρωμένο ψηφιακό εργαλείο που έχει σχεδιαστεί για να απλοποιήσει τη διαδικασία υποβολής, επεξεργασίας και παρακολούθησης αιτήσεων. Το σύστημα αυτό απευθύνεται σε εκπαιδευτικούς οργανισμούς, εταιρείες ή άλλους φορείς που χρειάζονται ένα σύστημα που να μπορεί να διαχειρίζεται μεγάλο όγκο αιτήσεων, διασφαλίζοντας ταυτόχρονα διαφάνεια, αξιοπιστία και αποτελεσματικότητα.

Η βασική λειτουργία του συστήματος είναι η διαχείριση αιτήσεων με τρόπο που να εξαλείφει τις χρονοβόρες διαδικασίες και τα λάθη που σχετίζονται με την παραδοσιακή (χειρόγραφη ή μέσω email) υποβολή αιτήσεων. Το σύστημα προσφέρει:

- Εύκολη και ασφαλή υποβολή αιτήσεων.
- Δυνατότητα αποθήκευσης και προβολής συνημμένων αρχείων.
- Έλεγχο και επεξεργασία αιτήσεων εντός προθεσμιών.
- Διαφάνεια στην παρακολούθηση της προόδου των αιτήσεων.

Το σύστημα προσφέρει τα παρακάτω κύρια χαρακτηριστικά:

1. **Διαχείριση Χρηστών**:
    - Κατηγορίες χρηστών: Φοιτητές, Εκπαιδευτικοί και Διαχειριστές.
    - Δικαιώματα πρόσβασης και λειτουργίες ανάλογα με τον ρόλο του χρήστη.
2. **Υποβολή Αιτήσεων**:
    - Υποβολή αιτήσεων από φοιτητές για συγκεκριμένες περιόδους.
    - Συνημμένα αρχεία με κατηγοριοποίηση (πτυχία, βιογραφικά, κ.λπ.).
3. **Επεξεργασία Αιτήσεων**:
    - Δυνατότητα επεξεργασίας αιτήσεων από τους χρήστες εντός προθεσμιών.
    - Αλλαγή ή διαγραφή συνημμένων αρχείων.
4. **Προβολή Αιτήσεων**:
    - Οι φοιτητές μπορούν να παρακολουθούν την κατάσταση των αιτήσεών τους.
    - Οι εκπαιδευτικοί μπορούν να βλέπουν αιτήσεις φοιτητών που ανήκουν στις περιόδους που έχουν πρόσβαση.

5. **Διαχείριση Περιόδων**:
    - Δημιουργία, επεξεργασία και διαχείριση περιόδων από τους διαχειριστές και εκπαιδευτικούς.
    - Σύνδεση αιτήσεων με συγκεκριμένες περιόδους.
6. **Ασφάλεια Δεδομένων**:
    - Χρήση κρυπτογραφημένων συνδέσεων για την προστασία των δεδομένων.
    - Δικαιώματα πρόσβασης για την αποτροπή μη εξουσιοδοτημένων ενεργειών.

Το σύστημα έχει σχεδιαστεί με έμφαση στη χρηστικότητα, ώστε οι τελικοί χρήστες να μπορούν να το χρησιμοποιούν χωρίς προηγούμενη τεχνική εξειδίκευση.

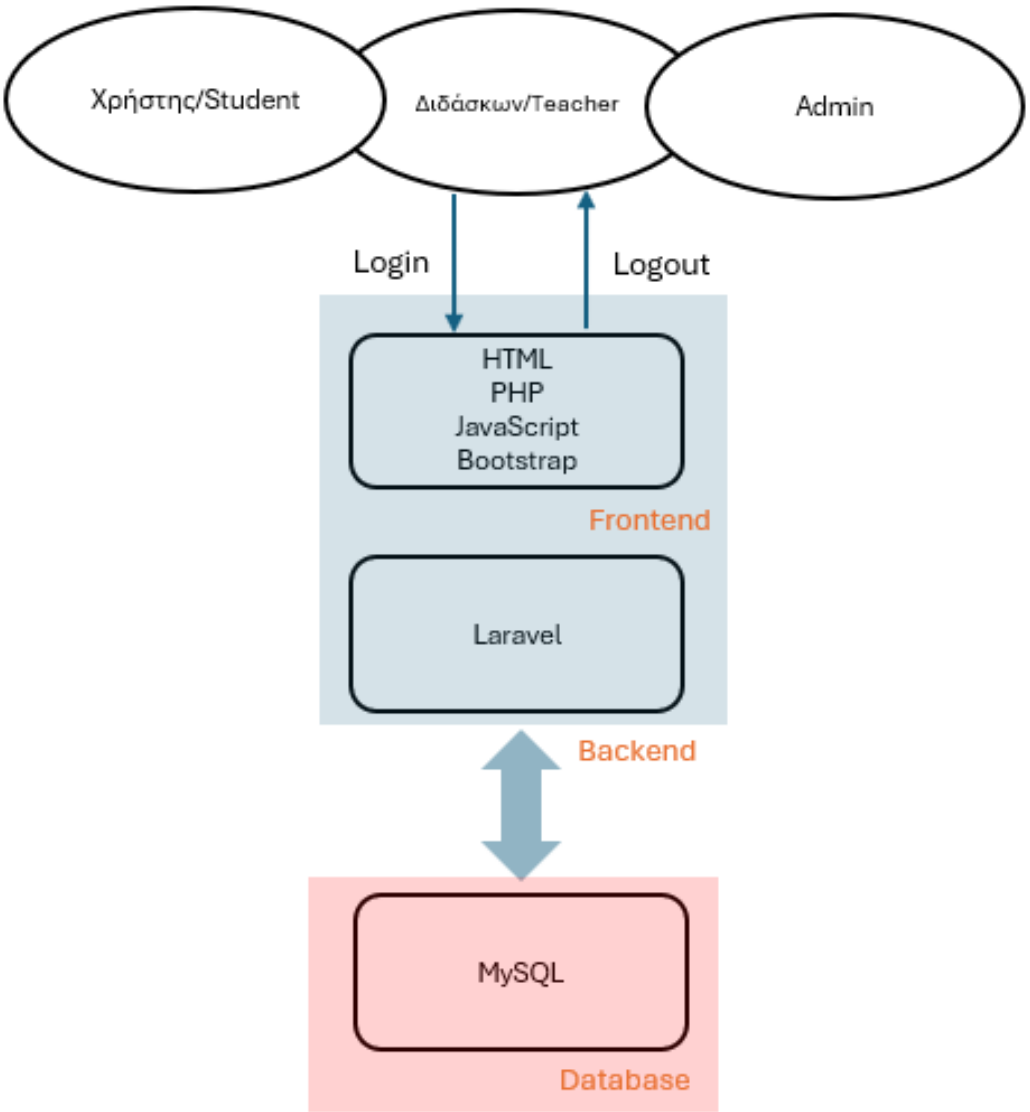
- **Φοιτητές**:
    - Υποβάλλουν αιτήσεις με συνημμένα αρχεία.
    - Επεξεργάζονται τις αιτήσεις τους εντός προθεσμιών.
- **Εκπαιδευτικοί**:
    - Δημιουργούν και επεξεργάζονται περιόδους.
    - Προβάλλουν αιτήσεις φοιτητών για συγκεκριμένες περιόδους.
- **Διαχειριστές**:
    - Έχουν πλήρη πρόσβαση σε όλες τις λειτουργίες του συστήματος.
    - Διαχειρίζονται χρήστες, αιτήσεις και περιόδους.

Τα οφέλη από τη χρήση του συστήματος είναι:

1. **Αυτοματοποίηση Διαδικασιών**:
    - Μείωση της χειροκίνητης διαχείρισης αιτήσεων.
    - Εξοικονόμηση χρόνου για φοιτητές και εκπαιδευτικούς.
2. **Ακρίβεια και Διαφάνεια**:
    - Κεντρική αποθήκευση δεδομένων για εύκολη παρακολούθηση.
    - Δικαιώματα πρόσβασης για αποφυγή λαθών.

3. **Προσαρμοστικότητα**:
    - Ευελιξία στην προσαρμογή για διαφορετικούς οργανισμούς.

Με την υλοποίηση αυτού του συστήματος, δίνεται η δυνατότητα για μια πλήρως ψηφιοποιημένη και αποδοτική διαδικασία διαχείρισης αιτήσεων, βελτιώνοντας την εμπειρία όλων των εμπλεκόμενων.



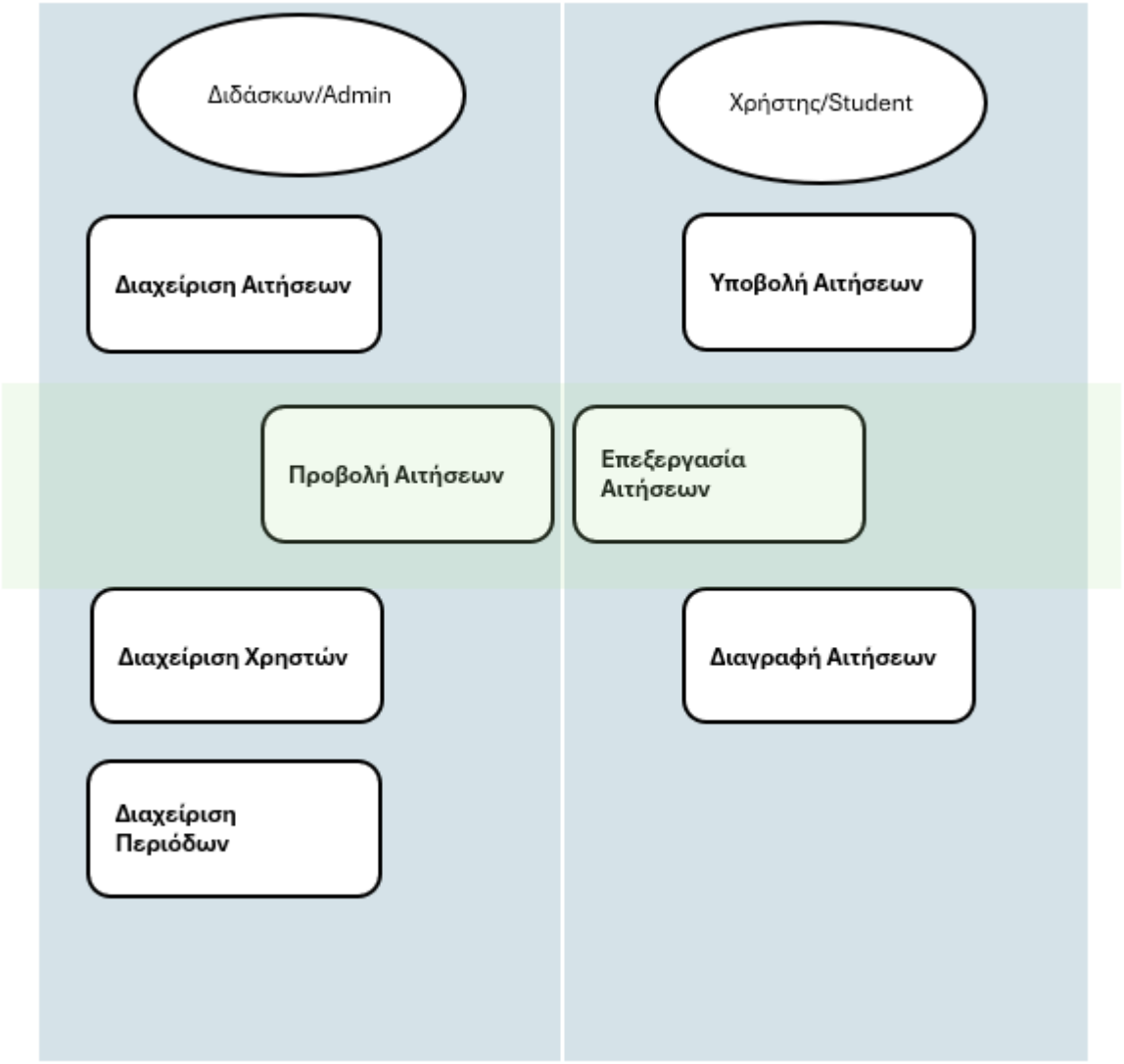
Εικόνα 4.1: Διάγραμμα Αρχιτεκτονικής Συστήματος: Περιγραφή

Το Διάγραμμα Αρχιτεκτονικής Συστήματος απεικονίζει τα κύρια στοιχεία του συστήματος και τη ροή επικοινωνίας μεταξύ αυτών. Περιλαμβάνει τα εξής:

1. **Επίπεδο Χρήστη (Frontend)**:
    - Περιλαμβάνει τους ρόλους: Φοιτητής, Εκπαιδευτικός και Διαχειριστής.
    - Οι χρήστες αλληλεπιδρούν με το σύστημα μέσω της διεπαφής χρήστη (HTML, JavaScript, PHP, Bootstrap).
2. **Επίπεδο Server (Backend)**:
    - Ο πυρήνας της εφαρμογής λειτουργεί με το framework Laravel.
    - Διαχειρίζεται τις αιτήσεις, την επεξεργασία δεδομένων, και τη σύνδεση με τη βάση δεδομένων.
3. **Επίπεδο Βάσης Δεδομένων**:
    - Χρησιμοποιείται MySQL για την αποθήκευση των δεδομένων χρηστών, αιτήσεων, περιόδων και συνημμένων αρχείων.

Η επικοινωνία πραγματοποιείται ως εξής:

- Οι χρήστες στέλνουν αιτήματα (HTTP Requests) προς τον server.
- Ο server χειρίζεται τις αιτήσεις και επικοινωνεί με τη βάση δεδομένων μέσω ερωτημάτων SQL.
- Τα αποτελέσματα επιστρέφονται στον χρήστη ως HTTP Responses.



Εικόνα 4.2: Διάγραμμα Ροής Δεδομένων: Περιγραφή

Το Διάγραμμα Ροής Δεδομένων (DFD) παρουσιάζει τη ροή των δεδομένων μεταξύ των βασικών μονάδων του συστήματος:

1. **Χρήστες (Φοιτητές, Εκπαιδευτικοί, Διαχειριστές)**:
    - Οι φοιτητές υποβάλλουν αιτήσεις με συνημμένα αρχεία.
    - Οι εκπαιδευτικοί έχουν πρόσβαση στις αιτήσεις συγκεκριμένων περιόδων.
    - Οι διαχειριστές διαχειρίζονται χρήστες, περιόδους και αιτήσεις.
2. **Διαχείριση Αιτήσεων**:
    - Η φόρμα αιτήσεων αποθηκεύει δεδομένα στη βάση και αντλεί πληροφορίες για την επεξεργασία και την προβολή.

3. **Διαχείριση Περιόδων**:
    - Οι εκπαιδευτικοί και οι διαχειριστές δημιουργούν και επεξεργάζονται περιόδους.
4. **Βάση Δεδομένων**:
    - Όλες οι πληροφορίες (χρήστες, αιτήσεις, αρχεία, περίοδοι) αποθηκεύονται στη βάση δεδομένων και διαχειρίζονται μέσω του server.

4.2 Περιγραφή του συστήματος και των λειτουργιών μέσω κώδικα

4.2.1 Χρήστης/Student

Θα περιγραφθούν οι βασικές λειτουργίες του Student Controller.

4.2.1.1 Μέθοδος index

```
public function index()
{
    // Ανάκτηση του user_id από το session
    $userId = Session::get('user_id');

    // Εύρεση των αιτήσεων του χρήστη και σύνδεση με τον πίνακα periods
    $applications = DB::table('applications')
        ->join('periods', 'applications.periodid', '=', 'periods.id') // Σύνδεση με
τον πίνακα periods για να ληφθούν οι ημερομηνίες περιόδου
        ->where('applications.userid', $userId) // Φιλτράρισμα των αιτήσεων που
ανήκουν στον τρέχοντα χρήστη
        ->select(
            'applications.*', // Όλα τα δεδομένα της αίτησης
            'periods.start_date', // Ημερομηνία έναρξης περιόδου
            'periods.end_date' // Ημερομηνία λήξης περιόδου
        )
        ->get();

    // Επιστροφή της προβολής που περιέχει τις αιτήσεις του χρήστη
    return view('student.viewapplication', ['applications' => $applications]);
}
```

Η μέθοδος `index` χρησιμοποιείται για την ανάκτηση και την εμφάνιση όλων των αιτήσεων που έχουν υποβληθεί από τον συνδεδεμένο χρήστη. Ενσωματώνει δεδομένα από τον πίνακα `applications` και τα συνδέει με τον πίνακα `periods` ώστε να περιλαμβάνονται οι ημερομηνίες έναρξης και λήξης της περιόδου κάθε αίτησης. Επιστρέφει την προβολή `student.viewapplication` με τα δεδομένα για παρουσίαση στον χρήστη.

4.2.1.2 Μέθοδος `create`

```
public function create()
{
    // Καταγραφή στο log ότι ξεκίνησε η διαδικασία δημιουργίας αίτησης
    Log::info('create');

    // Ανάκτηση του user_id από το session
    $userId = Session::get('user_id');

    // Εύρεση των περιόδων στις οποίες ο χρήστης έχει πρόσβαση
    $periods = DB::table('period_access')
        ->join('periods', 'period_access.periodid', '=', 'periods.id') // Σύνδεση
με τον πίνακα periods
        ->where('period_access.userid', $userId) // Φιλτράρισμα βάσει του χρήστη
        ->select('periods.*') // Επιστροφή όλων των πεδίων του πίνακα periods
        ->get();

    // Επιστροφή της προβολής για δημιουργία αίτησης με τις διαθέσιμες περιόδους
    return view('student.createapplication', ['periods' => $periods]);
}
```

Η μέθοδος `create` ετοιμάζει τα δεδομένα για τη δημιουργία μιας νέας αίτησης. Ελέγχει ποιες περιόδους είναι διαθέσιμες στον χρήστη μέσω του πίνακα `period_access` και επιστρέφει την προβολή `student.createapplication` μαζί με αυτές τις πληροφορίες. Καταγράφει τη διαδικασία στο log για παρακολούθηση.

4.2.1.3 Μέθοδος `store`

```
public function store(Request $request)
{
    // Επικύρωση των δεδομένων που υποβλήθηκαν από τη φόρμα
    $request->validate([
        'periodid' => 'required|integer',
        'firstname' => 'required|string|max:100',
        'lastname' => 'required|string|max:100',
        'parentname' => 'nullable|string|max:100',
        'idn' => 'nullable|string|max:10',
        'yearofbirth' => 'nullable|string|max:10',
        'city' => 'nullable|string|max:30',
        'tel' => 'nullable|string|max:20',
        'skill_maindegreetitle' => 'nullable|string|max:200',
        'skill_maindegreegrade' => 'required|numeric|min:0|max:10',
        'skill_occupation' => 'nullable|string|max:400',
        'skill_occupationmonths' => 'required|integer|min:0',
        'skill_agree' => 'required|boolean',
    ]);

    // Δημιουργία εγγραφής αίτησης στη βάση δεδομένων
    $applicationId = DB::table('applications')->insertGetId([
        'userid' => Session::get('user_id'), // Σύνδεση της αίτησης με τον τρέχοντα
χρήστη
        'periodid' => $request->periodid, // Η περίοδος της αίτησης
        'firstname' => $request->firstname,
        'lastname' => $request->lastname,
        'parentname' => $request->parentname,
        'idn' => $request->idn,
```

```
        'yearofbirth' => $request->yearofbirth,
        'city' => $request->city,
        'tel' => $request->tel,
        'skill_maindegreetitle' => $request->skill_maindegreetitle,
        'skill_maindegreegrade' => $request->skill_maindegreegrade,
        'skill_occupation' => $request->skill_occupation,
        'skill_occupationmonths' => $request->skill_occupationmonths,
        'skill_agree' => $request->skill_agree,
        'protocol_number' => strtoupper(uniqid('PR')), // Δημιουργία μοναδικού
αριθμού πρωτοκόλλου
        'created_at' => now(),
    ]);

    // Ανακατεύθυνση στη λίστα αιτήσεων με μήνυμα επιτυχίας
    return redirect()->route('student.applications.index')->with('success', 'Η
αίτηση δημιουργήθηκε επιτυχώς.');
}
```

Η μέθοδος `store` αποθηκεύει μια νέα αίτηση στη βάση δεδομένων. Πρώτα, επικυρώνει τα δεδομένα που παρέχονται από τη φόρμα. Στη συνέχεια, δημιουργεί την εγγραφή στον πίνακα `applications` με όλα τα δεδομένα της αίτησης. Δημιουργεί έναν μοναδικό αριθμό πρωτοκόλλου και ανακατευθύνει τον χρήστη στη λίστα αιτήσεων με μήνυμα επιτυχίας.

4.2.1.4 Μέθοδος `edit`

```
public function edit($id)
{
    // Ανάκτηση του user_id από το session
    $userId = Session::get('user_id');

    // Εύρεση της αίτησης βάσει του ID και του χρήστη
    $application = DB::table('applications')
        ->where('id', $id)
        ->where('userid', $userId)
        ->first();

    // Αν η αίτηση δεν βρεθεί, επιστροφή σφάλματος 404
    if (!$application) {
        abort(404, 'Η αίτηση δεν βρέθηκε.');
    }

    // Έλεγχος αν η αίτηση είναι επεξεργάσιμη βάσει της περιόδου
    $currentDate = now();
    $period = DB::table('periods')->where('id', $application->periodid)->first();
    $isEditable = $currentDate->between($period->start_date, $period->end_date);

    // Ανάκτηση των αρχείων που σχετίζονται με την αίτηση
    $files = DB::table('files')
        ->where('applicationid', $id)
        ->get();

    // Εύρεση των περιόδων στις οποίες έχει πρόσβαση ο χρήστης
    $periods = DB::table('period_access')
        ->join('periods', 'period_access.periodid', '=', 'periods.id')
        ->where('period_access.userid', $userId)
        ->select('periods.*')
        ->get();

    // Επιστροφή της προβολής επεξεργασίας με τα δεδομένα
    return view('student.editorviewapplication', [
        'application' => $application,
```

```
        'files' => $files,
        'periods' => $periods,
        'isEditable' => $isEditable,
    ]);
}
```

Η μέθοδος `edit` επιτρέπει την επεξεργασία μιας αίτησης. Ανακτά την αίτηση από τη βάση δεδομένων βάσει του ID και ελέγχει αν ανήκει στον συνδεδεμένο χρήστη. Ελέγχει αν η αίτηση είναι επεξεργάσιμη βάσει της περιόδου (αν η τρέχουσα ημερομηνία είναι εντός των ημερομηνιών της περιόδου). Φορτώνει τα σχετικά αρχεία και τις περιόδους στις οποίες έχει πρόσβαση ο χρήστης και επιστρέφει την προβολή `student.editorviewapplication`.

4.2.1.5 Μέθοδος `update`

```
public function update(Request $request, $id)
{
    // Επικύρωση των δεδομένων που υποβλήθηκαν από τη φόρμα
    $request->validate([
        'periodid' => 'required|integer',
        'firstname' => 'required|string|max:100',
        'lastname' => 'required|string|max:100',
        'parentname' => 'nullable|string|max:100',
        'idn' => 'nullable|string|max:10',
        'yearofbirth' => 'nullable|string|max:10',
        'city' => 'nullable|string|max:30',
        'tel' => 'nullable|string|max:20',
        'skill_maindegreetitle' => 'nullable|string|max:200',
        'skill_maindegreegrade' => 'required|numeric|min:0|max:10',
        'skill_occupation' => 'nullable|string|max:400',
        'skill_occupationmonths' => 'required|integer|min:0',
        'skill_agree' => 'required|boolean',
    ]);

    // Ενημέρωση της αίτησης στη βάση δεδομένων
    DB::table('applications')
        ->where('id', $id)
        ->where('userid', Session::get('user_id'))
        ->update([
            'periodid' => $request->periodid,
            'firstname' => $request->firstname,
            'lastname' => $request->lastname,
            'parentname' => $request->parentname,
            'idn' => $request->idn,
            'yearofbirth' => $request->yearofbirth,
            'city' => $request->city,
            'tel' => $request->tel,
            'skill_maindegreetitle' => $request->skill_maindegreetitle,
            'skill_maindegreegrade' => $request->skill_maindegreegrade,
            'skill_occupation' => $request->skill_occupation,
            'skill_occupationmonths' => $request->skill_occupationmonths,
            'skill_agree' => $request->skill_agree,
            'updated_at' => now(),
        ]);

    // Επιστροφή στην επεξεργασία της αίτησης με μήνυμα επιτυχίας
    return redirect()->route('student.application.edit', $id)
        ->with('success', 'Η αίτηση ενημερώθηκε επιτυχώς.');
```

```
}
```

Η μέθοδος `update` ενημερώνει τα δεδομένα μιας αίτησης. Πρώτα επικυρώνει τα δεδομένα που υποβλήθηκαν μέσω της φόρμας και στη συνέχεια ενημερώνει την αίτηση στη βάση δεδομένων. Τέλος, ανακατευθύνει τον χρήστη πίσω στη φόρμα επεξεργασίας με μήνυμα επιτυχίας.

4.2.1.6 Μέθοδος `show`

```
public function show($id)
{
    // Ανάκτηση του user_id και του user_role από το session
    $userId = Session::get('user_id');
    $userRole = Session::get('user_role'); // 0: student, 1: teacher, 2: admin

    // Εύρεση της αίτησης βάσει του ID
    $application = DB::table('applications')->where('id', $id)->first();

    // Επιστροφή σφάλματος 404 αν η αίτηση δεν βρεθεί
    if (!$application) {
        abort(404, 'Η αίτηση δεν βρέθηκε.');
    }

    // Έλεγχος πρόσβασης για τον καθηγητή
    if ($userRole == 1) {
        $hasAccess = DB::table('period_access')
            ->where('userid', $userId)
            ->where('periodid', $application->periodid)
            ->exists();

        if (!$hasAccess) {
            abort(403, 'Δεν έχετε πρόσβαση σε αυτή την αίτηση.');
        }
    }

    // Έλεγχος πρόσβασης για τον φοιτητή
    if ($userRole == 0 && $application->userid != $userId) {
        abort(403, 'Δεν έχετε πρόσβαση σε αυτή την αίτηση.');
    }

    // Επιστροφή της προβολής με τα δεδομένα
    return view('student.editorviewapplication', ['application' => $application]);
}
```

Η μέθοδος `show` εμφανίζει τα δεδομένα μιας αίτησης. Ελέγχει αν ο χρήστης έχει δικαίωμα πρόσβασης στην αίτηση και επιστρέφει την προβολή `student.editorviewapplication` για την εμφάνιση των δεδομένων.

4.2.1.7 Μέθοδος `destroy`

```
public function destroy($id)
{
    // Διαγραφή της αίτησης από τον πίνακα applications βάσει του ID
    DB::table('applications')->where('id', $id)->delete();

    // Ανακατεύθυνση στη λίστα αιτήσεων με μήνυμα επιτυχίας
    return redirect()->route('student.applications.index')
        ->with('success', 'Η αίτηση διαγράφηκε επιτυχώς.');
}
```

Η μέθοδος `destroy` διαγράφει μια αίτηση από τη βάση δεδομένων βάσει του ID της. Ανακατευθύνει τον χρήστη πίσω στη λίστα αιτήσεων με μήνυμα επιτυχούς διαγραφής.

4.2.2 Διδάσκων/Teacher

4.2.2.1 Μέθοδος `listPeriods`

```
public function listPeriods()
{
    // Καταγραφή στο log ότι ξεκίνησε η διαδικασία λήψης περιόδων
    Log::info('listPeriods');

    // Έλεγχος αν ο χρήστης είναι συνδεδεμένος
    if (!Session::has('user_id')) {
        return redirect()->route('login'); // Ανακατεύθυνση στη σελίδα σύνδεσης αν
δεν είναι συνδεδεμένος
    }

    // Ανάκτηση του teacherId από το session
    $teacherId = Session::get('user_id');

    // Εύρεση όλων των περιόδων στις οποίες έχει πρόσβαση ο διδάσκων
    $periods = DB::table('period_access')
        ->join('periods', 'period_access.periodid', '=', 'periods.id') // Σύνδεση
με τον πίνακα periods
        ->where('period_access.userid', $teacherId) // Φιλτράρισμα βάσει του χρήστη
        ->select('periods.*') // Επιστροφή όλων των πεδίων των περιόδων
        ->get();

    // Επιστροφή της προβολής με τις περιόδους
    return view('teacher.viewperiods', ['periods' => $periods]);
}
```

Η μέθοδος `listPeriods` ανακτά όλες τις περιόδους στις οποίες έχει πρόσβαση ο διδάσκων και επιστρέφει την προβολή `teacher.viewperiods`. Αν ο χρήστης δεν είναι συνδεδεμένος, ανακατευθύνεται στη σελίδα σύνδεσης.

4.2.2.2 Μέθοδος `viewStudents`

```
public function viewStudents($periodId)
{
    // Εύρεση όλων των φοιτητών που έχουν υποβάλει αιτήσεις για μια συγκεκριμένη
περίοδο
    $students = DB::table('applications')
        ->join('users', 'applications.userid', '=', 'users.id') // Σύνδεση με τον
πίνακα users
        ->where('applications.periodid', $periodId) // Φιλτράρισμα βάσει του ID
περιόδου
        ->select('users.id', 'users.firstname', 'users.lastname', 'users.email') //
Επιλογή πεδίων φοιτητών
        ->get();

    // Επιστροφή της προβολής με τη λίστα των φοιτητών
    return view('teacher.viewstudents', ['students' => $students]);
}
```

Η μέθοδος `viewStudents` επιστρέφει τη λίστα φοιτητών που έχουν υποβάλει αιτήσεις για μια συγκεκριμένη περίοδο. Φορτώνει δεδομένα από τους πίνακες `applications` και `users` και επιστρέφει την προβολή `teacher.viewstudents`.

4.2.2.3 Μέθοδος `viewApplication`

```
public function viewApplication($studentId)
{
    // Ανάκτηση της αίτησης του φοιτητή βάσει του ID του
    $application = DB::table('applications')->where('userid', $studentId)->first();

    // Επιστροφή της προβολής που περιέχει την αίτηση του φοιτητή
    return view('teacher.students.application', ['application' => $application]);
}
```

Η μέθοδος `viewApplication` επιστρέφει τα δεδομένα μιας αίτησης που ανήκει σε συγκεκριμένο φοιτητή. Επιστρέφει την προβολή `teacher.students.application` με την αίτηση.

4.2.2.4 Μέθοδος `createPeriod`

```
public function createPeriod()
{
    // Επιστροφή της προβολής για τη δημιουργία νέας περιόδου
    return view('teacher.createperiod');
}
```

Η μέθοδος `createPeriod` επιστρέφει τη φόρμα για τη δημιουργία μιας νέας περιόδου στην προβολή `teacher.createperiod`.

4.2.2.5 Μέθοδος `storePeriod`

```
public function storePeriod(Request $request)
{
    // Επικύρωση των δεδομένων που υποβλήθηκαν μέσω της φόρμας
    $request->validate([
        'periodtitle' => 'required|string|max:30',
        'start_date' => 'required|date',
        'end_date' => 'required|date|after_or_equal:start_date',
    ]);

    // Δημιουργία νέας εγγραφής στον πίνακα periods
    DB::table('periods')->insert([
        'periodtitle' => $request->periodtitle,
        'start_date' => $request->start_date,
        'end_date' => $request->end_date,
        'created_at' => now(),
        'updated_at' => now(),
    ]);

    // Ανακατεύθυνση στη λίστα περιόδων με μήνυμα επιτυχίας
    return redirect()->route('teacher.period.list')->with('success', 'Η περίοδος
δημιουργήθηκε επιτυχώς.');
}
```

Η μέθοδος `storePeriod` αποθηκεύει μια νέα περίοδο στη βάση δεδομένων. Επικυρώνει τα δεδομένα από τη φόρμα, εισάγει τη νέα περίοδο στον πίνακα `periods`, και ανακατευθύνει τον χρήστη στη λίστα περιόδων με μήνυμα επιτυχίας.

4.2.2.6 Μέθοδος `editPeriod`

```
public function editPeriod($id)
{
    // Ανάκτηση της περιόδου βάσει του ID
    $period = DB::table('periods')->where('id', $id)->first();

    // Αν η περίοδος δεν βρεθεί, επιστροφή σφάλματος 404
    if (!$period) {
        abort(404, 'Η περίοδος δεν βρέθηκε.');
    }

    // Επιστροφή της προβολής επεξεργασίας με τα δεδομένα της περιόδου
    return view('teacher.editperiod', ['period' => $period]);
}
```

Η μέθοδος `editPeriod` επιστρέφει τη φόρμα επεξεργασίας για μια συγκεκριμένη περίοδο. Ελέγχει αν η περίοδος υπάρχει και επιστρέφει την προβολή `teacher.editperiod`.

4.2.2.7 Μέθοδος `updatePeriod`

```
public function updatePeriod(Request $request, $id)
{
    // Επικύρωση των δεδομένων που υποβλήθηκαν
    $request->validate([
        'periodtitle' => 'required|string|max:30',
        'start_date' => 'required|date',
        'end_date' => 'required|date|after_or_equal:start_date',
    ]);

    // Ενημέρωση της περιόδου στη βάση δεδομένων
    DB::table('periods')
        ->where('id', $id)
        ->update([
            'periodtitle' => $request->periodtitle,
            'start_date' => $request->start_date,
            'end_date' => $request->end_date,
            'updated_at' => now(),
        ]);

    // Ανακατεύθυνση στη λίστα περιόδων με μήνυμα επιτυχίας
    return redirect()->route('teacher.period.list')->with('success', 'Η περίοδος
ενημερώθηκε επιτυχώς.');
}
```

Η μέθοδος `updatePeriod` ενημερώνει τα δεδομένα μιας περιόδου. Επικυρώνει τα δεδομένα της φόρμας, ενημερώνει τη βάση δεδομένων, και επιστρέφει τον χρήστη στη λίστα περιόδων με μήνυμα επιτυχίας.

4.2.2.8 Μέθοδος `listesApoPeriods`

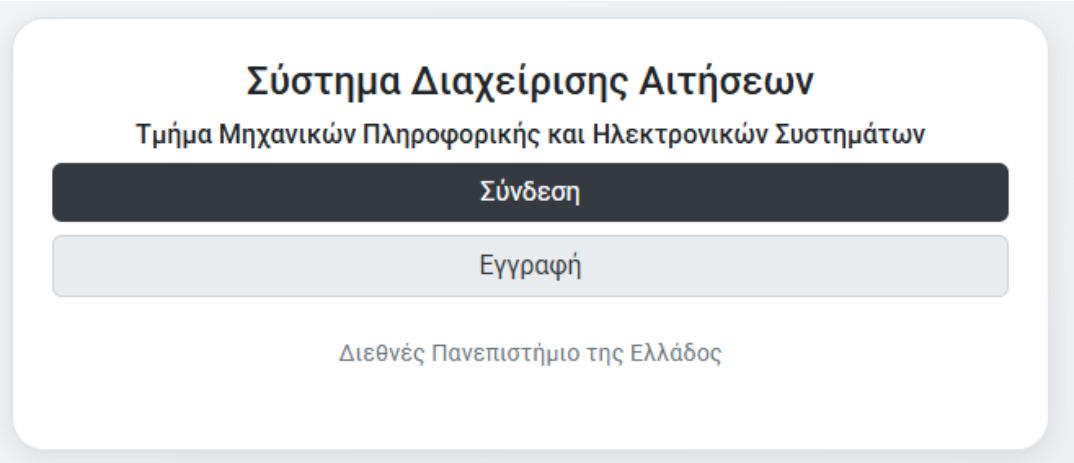
```
public function listesApoPeriods()
{
    // Ανάκτηση όλων των περιόδων από τη βάση δεδομένων
    $periods = DB::table('periods')->get();

    // Επιστροφή της προβολής με τη λίστα των περιόδων
    return view('teacher.listperiods', ['periods' => $periods]);
}
```

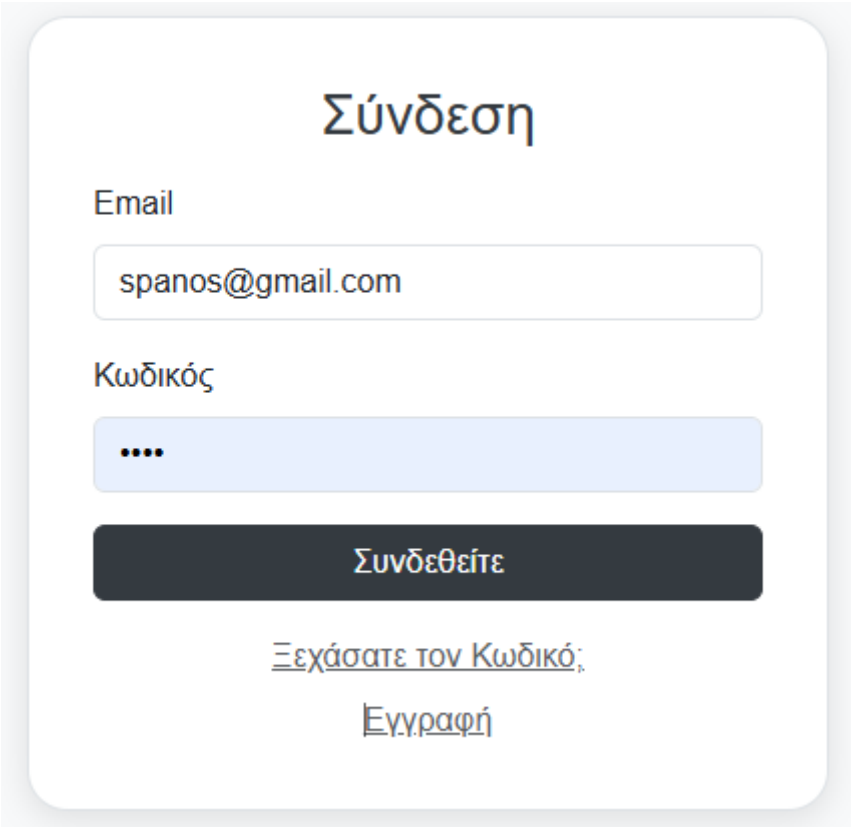
Η μέθοδος `listesApoPeriods` ανακτά όλες τις περιόδους από τη βάση δεδομένων και επιστρέφει την προβολή `teacher.listperiods` για να εμφανιστεί η λίστα.

4.3 Η ιστοσελίδα σε εικόνες

Στην Εικόνα 4.3 παρουσιάζεται η αρχική σελίδα (welcome page) του Συστήματος Διαχείρισης Αιτήσεων. Αυτή η σελίδα λειτουργεί ως σημείο εισόδου για όλους τους χρήστες, προσφέροντας δύο κύριες επιλογές: Σύνδεση (Login) και Εγγραφή (Register). Το περιβάλλον είναι μοντέρνο και φιλικό προς τον χρήστη, με σαφή διάταξη και κουμπιά δράσης για εύκολη πλοήγηση.



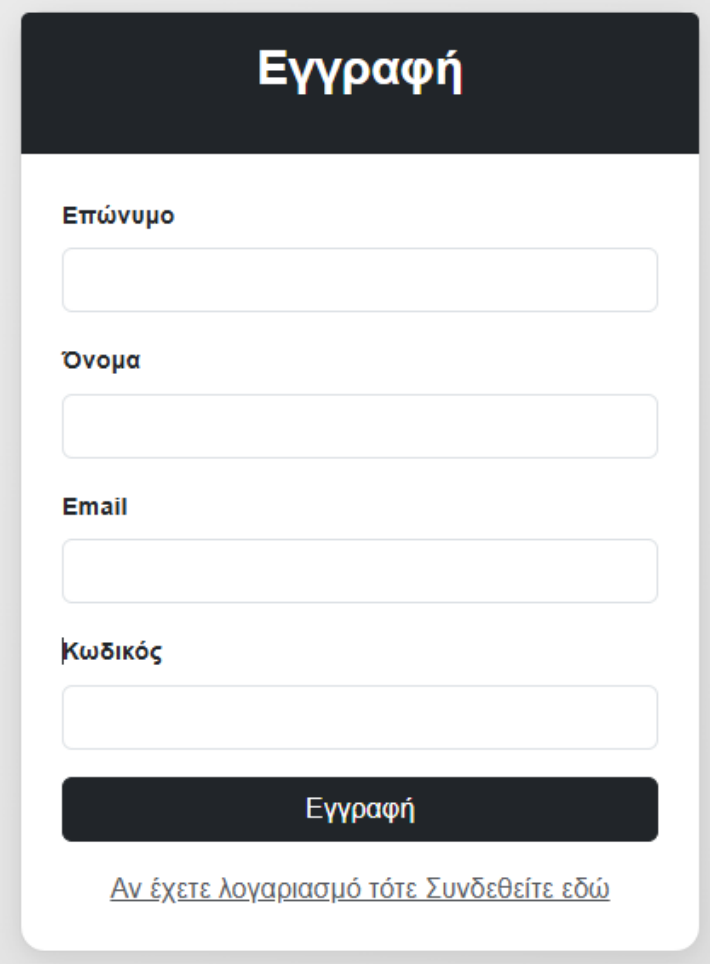
Εικόνα 4.3: Ιστοσελίδα welcome



Εικόνα 4.4: Ιστοσελίδα για τη σύνδεση του χρήστη

Στην Εικόνα 4.4 απεικονίζεται η σελίδα σύνδεσης (Login) του συστήματος. Οι χρήστες εισάγουν το email και τον κωδικό τους για να αποκτήσουν πρόσβαση στις λειτουργίες του συστήματος. Η διάταξη είναι απλή και καθαρή, διευκολύνοντας τη γρήγορη πλοήγηση.

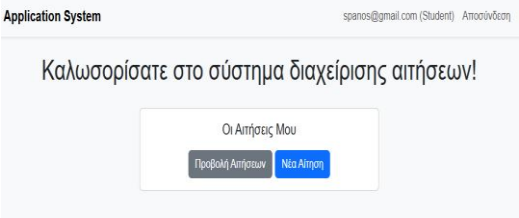
Στην Εικόνα 4.5 παρουσιάζεται η σελίδα εγγραφής νέου χρήστη. Η φόρμα περιλαμβάνει πεδία για όνομα, email και κωδικό. Το περιβάλλον είναι φιλικό και σαφές, ενθαρρύνοντας τον χρήστη να ολοκληρώσει την εγγραφή του εύκολα και γρήγορα.



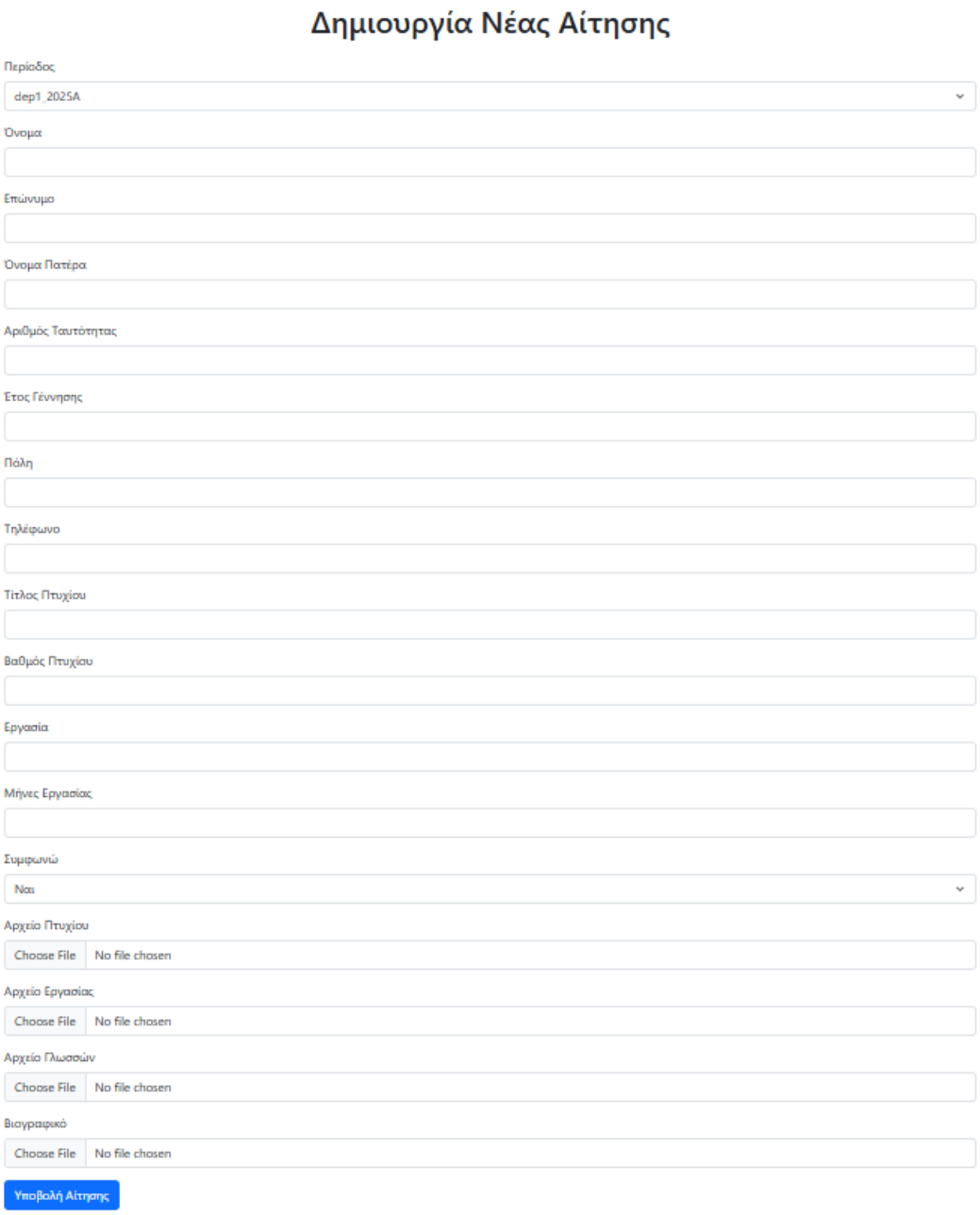
Εικόνα 4.5: Εγγραφή ενός χρήστη

Στην Εικόνα 4.6 φαίνεται το Dashboard του φοιτητή/χρήστη. Ο χρήστης μπορεί να δει τις υπάρχουσες αιτήσεις του και να δημιουργήσει νέες.

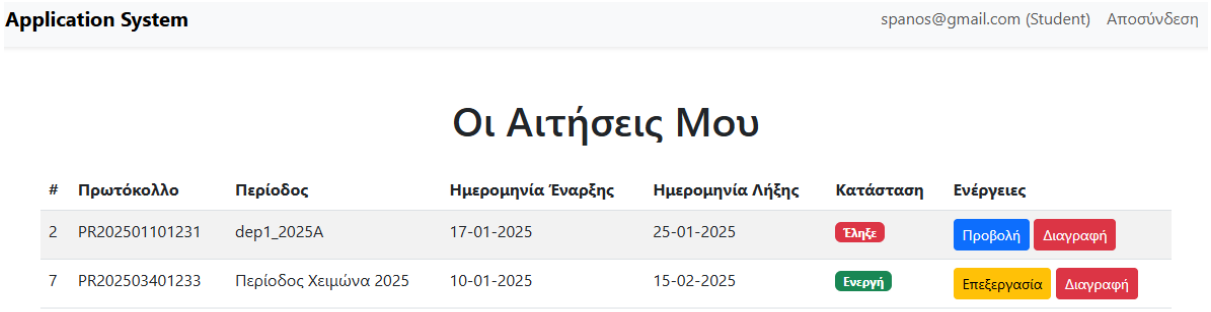
Στην Εικόνα 4.7 παρουσιάζεται η φόρμα δημιουργίας νέας αίτησης από τον χρήστη. Η φόρμα περιλαμβάνει πεδία για προσωπικά στοιχεία, δεξιότητες, και δυνατότητα προσθήκης συνημμένων αρχείων. Η διαδικασία είναι απλή και γρήγορη.



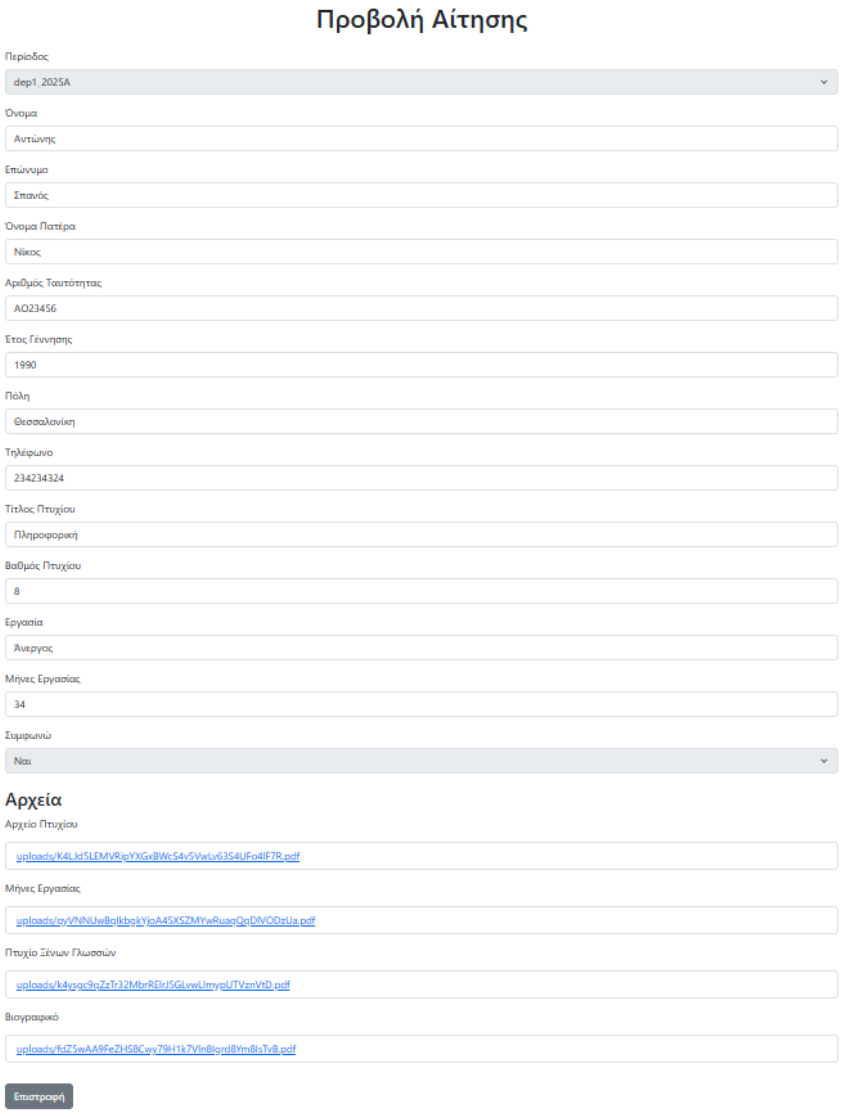
Εικόνα 4.6: Ιστοσελίδα Dashboard για τον χρήστη/student που θα κάνε αίτηση. Μπορεί να δει τις αιτήσεις του ή να δημιουργήσει νέα.



Εικόνα 4.7: Χρήστης – δημιουργία νέας αίτησης



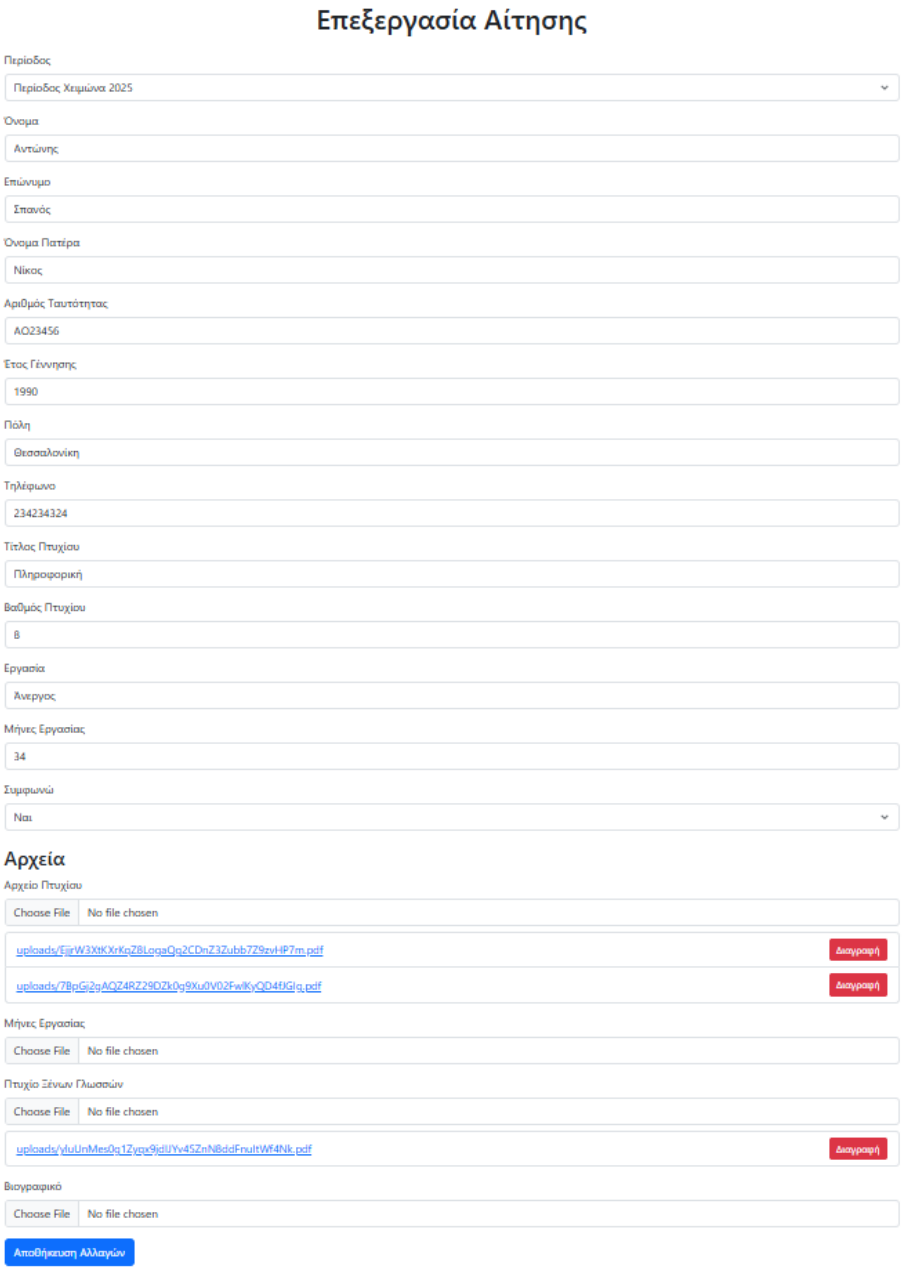
Εικόνα 4.8: Χρήστης – οι αιτήσεις του. Ανάλογα την προθεσμία μπορεί να την προβάλλει ή να συνεχίσει με την επεξεργασία της



Εικόνα 4.9: Χρήστης - Προβολή αίτησης όταν είναι εκτός προθεσμίας χωρίς δυνατότητα επεξεργασίας

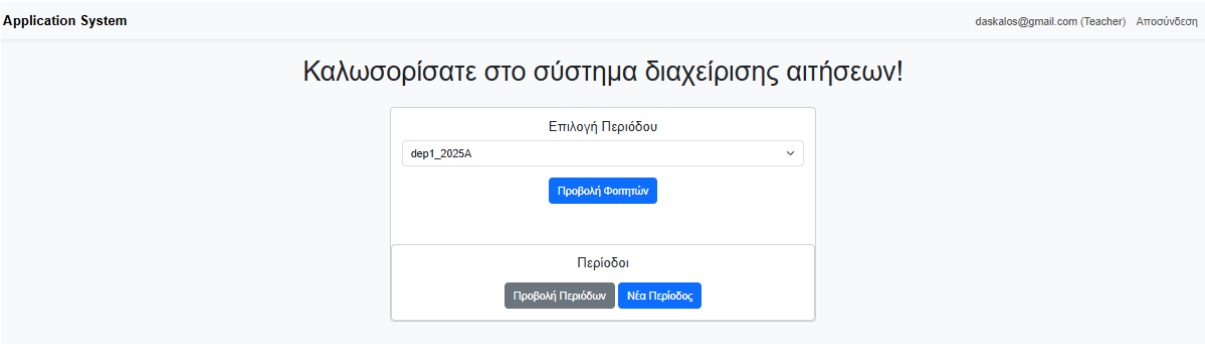
Στην Εικόνα 4.8 παρουσιάζεται η σελίδα όπου ο χρήστης βλέπει τις αιτήσεις του. Για κάθε αίτηση εμφανίζονται βασικές πληροφορίες, όπως το πρωτόκολλο και η περίοδος. Ανάλογα με την προθεσμία, μπορεί είτε να προβάλει είτε να επεξεργαστεί την αίτηση.

Στην Εικόνα 4.9 φαίνεται η προβολή μιας αίτησης όταν η προθεσμία έχει λήξει. Ο χρήστης μπορεί να δει όλα τα στοιχεία της αίτησης, αλλά η δυνατότητα επεξεργασίας είναι απενεργοποιημένη για λόγους διαφάνειας και ορθής διαχείρισης.



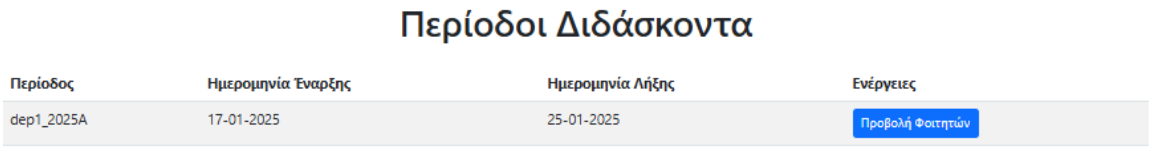
Εικόνα 4.10: Χρήστης - Επεξεργασία αίτησης όταν είναι εντός προθεσμίας

Στην Εικόνα 4.10 παρουσιάζεται η επεξεργασία μιας αίτησης που είναι εντός προθεσμίας. Ο χρήστης έχει πρόσβαση σε όλα τα πεδία και μπορεί να αλλάξει στοιχεία ή να προσθέσει/διαγράψει συνημμένα αρχεία. Η διαδικασία είναι εύχρηστη και ασφαλής.

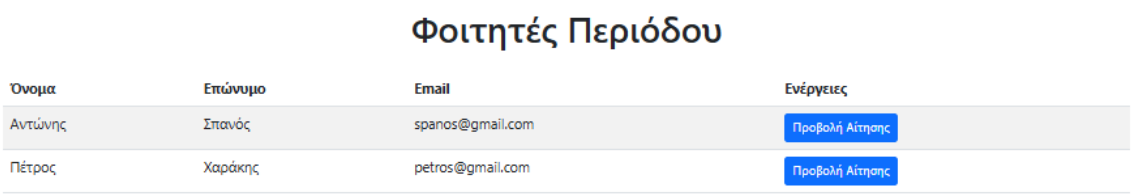


Εικόνα 4.11: Διαχειριστής/teacher – σελίδα υποδοχής dashboard – μπορεί να δει και να επεξεργαστεί τις περιόδους του ή να επιλέξει μια και να δει τις αιτήσεις των χρηστών/φοιτητών

Στην Εικόνα 4.11 φαίνεται το Dashboard του διαχειριστή/εκπαιδευτικού. Παρέχονται επιλογές για διαχείριση περιόδων και προβολή αιτήσεων χρηστών/φοιτητών. Το περιβάλλον είναι οργανωμένο για εύκολη πλοήγηση και γρήγορη διαχείριση.



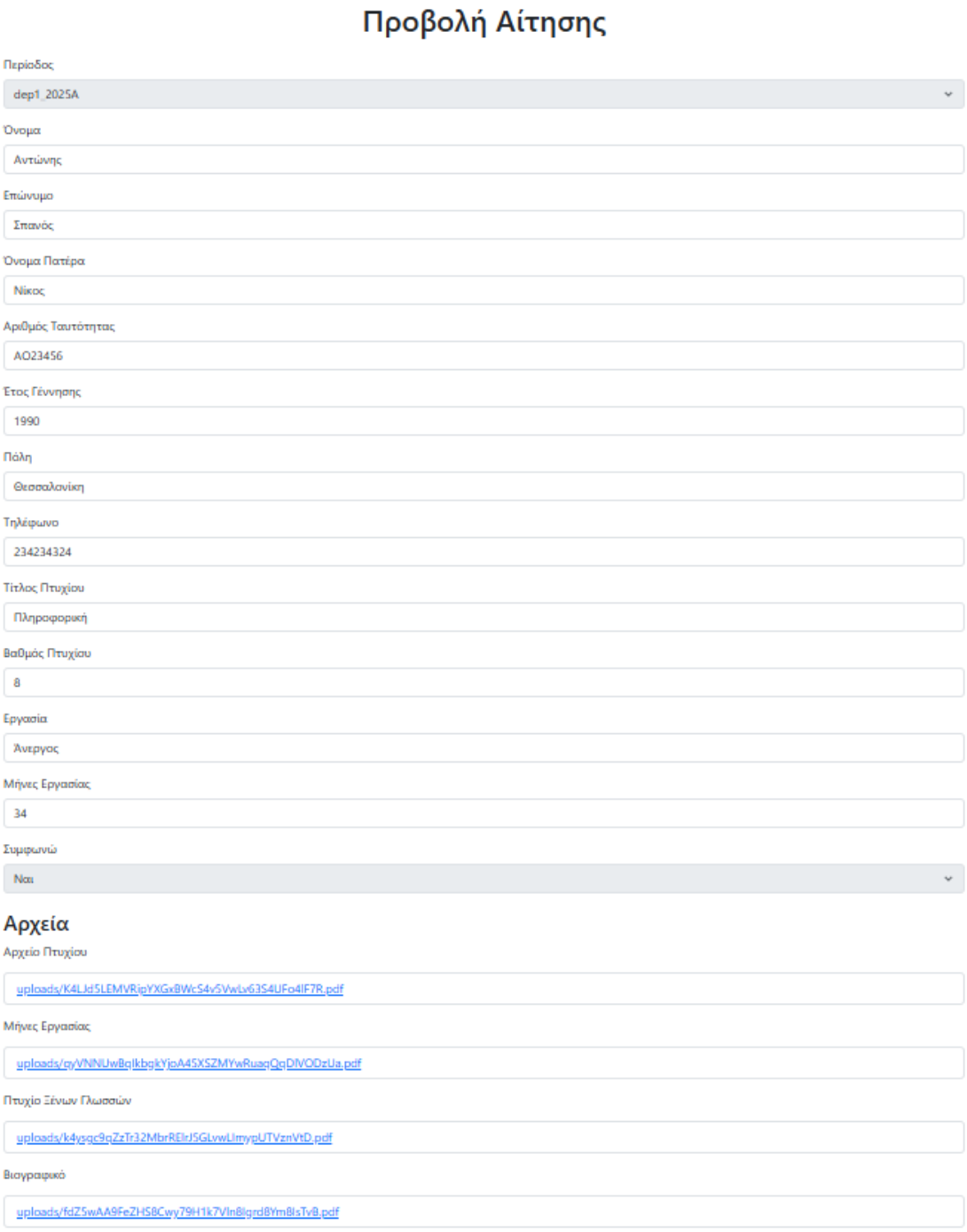
Εικόνα 4.12: Διαχειριστής/teacher – οι περίοδοι του



Εικόνα 4.13: Διαχειριστής/teacher – οι χρήστες/φοιτητές που έχουν κάνει αίτηση στην επιλεγμένη περίοδο

Στην Εικόνα 4.12 απεικονίζονται όλες οι περίοδοι στις οποίες έχει πρόσβαση ο διαχειριστής/εκπαιδευτικός. Παρέχονται βασικές πληροφορίες για κάθε περίοδο, όπως ο τίτλος και οι ημερομηνίες. Υπάρχει δυνατότητα επιλογής για προβολή ή επεξεργασία.

Στην Εικόνα 4.13 φαίνεται η λίστα των χρηστών/φοιτητών που έχουν υποβάλει αίτηση για την επιλεγμένη περίοδο. Για κάθε φοιτητή εμφανίζονται βασικά στοιχεία, όπως όνομα και email, με δυνατότητα προβολής της αίτησης.

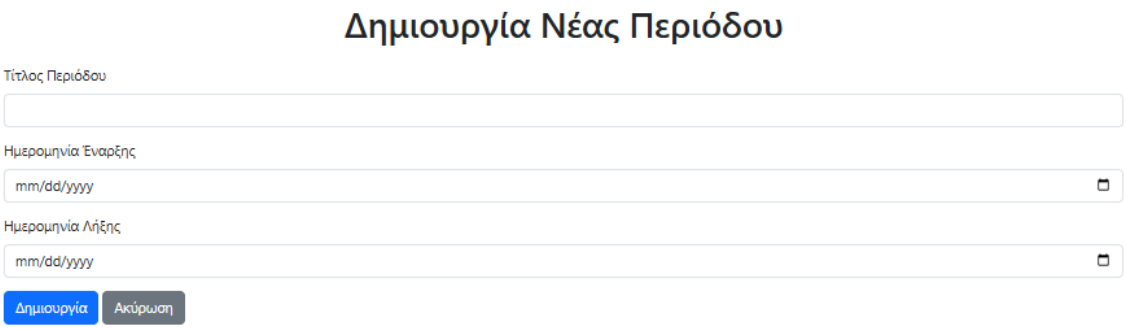


Εικόνα 4.14: Διαχειριστής/teacher – προβολή μια αίτησης ενός χρήστη/student

Στην Εικόνα 4.14 παρουσιάζεται η προβολή μιας αίτησης από τον διαχειριστή/εκπαιδευτικό. Ο διαχειριστής μπορεί να δει όλα τα στοιχεία της αίτησης, συμπεριλαμβανομένων των συνημμένων αρχείων, χωρίς τη δυνατότητα επεξεργασίας.

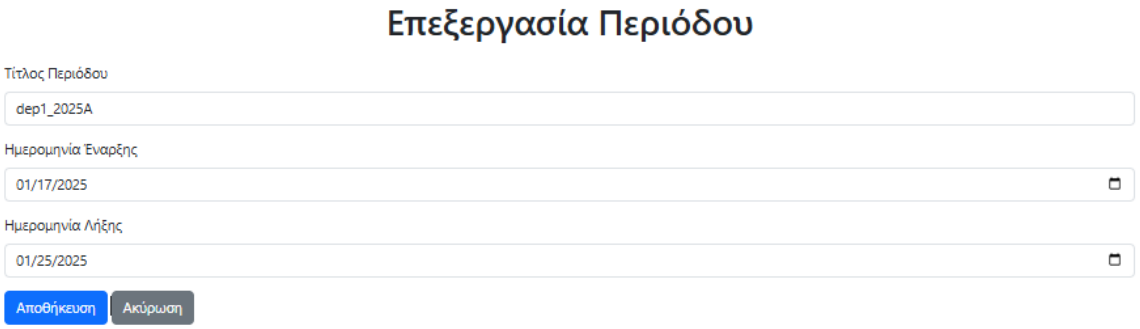
Εικόνα 4.15: Διαχειριστής/teacher – Λίστα με τις περιόδους του

Στην Εικόνα 4.15 απεικονίζεται η πλήρης λίστα περιόδων του διαχειριστή/εκπαιδευτικού. Η λίστα περιλαμβάνει όλες τις υπάρχουσες περιόδους με επιλογές προβολής ή επεξεργασίας ανάλογα με τα δικαιώματα πρόσβασης.



Εικόνα 4.16: Διαχειριστής/teacher – Δημιουργία νέας περιόδου

Στην Εικόνα 4.16 φαίνεται η φόρμα δημιουργίας νέας περιόδου. Ο διαχειριστής εισάγει τίτλο και ημερομηνίες έναρξης και λήξης. Η διαδικασία είναι απλή και διασφαλίζει τη σωστή καταχώριση της περιόδου.

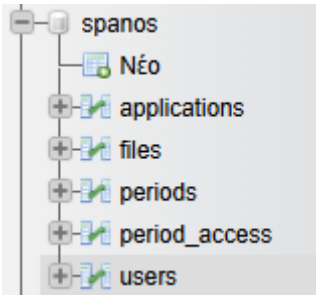


Εικόνα 4.17: Διαχειριστής/teacher – Επεξεργασία μιας περιόδου

Στην Εικόνα 4.17 παρουσιάζεται η επεξεργασία μιας υπάρχουσας περιόδου από τον διαχειριστή/εκπαιδευτικό. Ο διαχειριστής μπορεί να αλλάξει τον τίτλο ή τις ημερομηνίες της περιόδου, ενώ διατηρείται πλήρης καταγραφή των αλλαγών.

4.4 Η βάση δεδομένων MySQL

Η βάση δεδομένων του συστήματος διαχείρισης αιτήσεων έχει σχεδιαστεί για να υποστηρίζει όλες τις λειτουργίες που απαιτούνται για τη διαχείριση αιτήσεων, περιόδων, χρηστών, και συνημμένων αρχείων. Στόχος της είναι να παρέχει μια σταθερή και επεκτάσιμη υποδομή για την αποθήκευση, οργάνωση και ανάκτηση των δεδομένων, εξασφαλίζοντας την ακρίβεια και την ασφάλεια τους. Περιλαμβάνει πέντε κύριους πίνακες: users, applications, files, periods, και period_access. Κάθε πίνακας εξυπηρετεί συγκεκριμένο ρόλο στη λειτουργικότητα του συστήματος, διασφαλίζοντας την αρμονική διαχείριση των δεδομένων.



Εικόνα 4.18: Οι πίνακες

1. Πίνακας users

Ο πίνακας users αποθηκεύει πληροφορίες για τους χρήστες του συστήματος, οι οποίοι μπορεί να είναι φοιτητές, εκπαιδευτικοί ή διαχειριστές. Περιλαμβάνει πεδία όπως id, email, password, firstname, lastname, και kind, το οποίο καθορίζει τον ρόλο του χρήστη.

Λειτουργίες:

- Διαχείριση χρηστών (προσθήκη, ενημέρωση, διαγραφή).
- Αναγνώριση χρηστών για τη σύνδεση και τον καθορισμό δικαιωμάτων.
- Υποστήριξη πολλαπλών ρόλων (φοιτητής, εκπαιδευτικός, διαχειριστής).

2. Πίνακας applications

Ο πίνακας applications περιέχει δεδομένα για τις αιτήσεις που υποβάλλουν οι χρήστες. Κάθε αίτηση συνδέεται με έναν χρήστη (userid) και μια περίοδο (periodid). Περιλαμβάνει πληροφορίες όπως το όνομα, το επώνυμο, το πρωτόκολλο, στοιχεία δεξιοτήτων, και δεδομένα επικοινωνίας.

Λειτουργίες:

- Υποβολή αιτήσεων από φοιτητές.
- Σύνδεση αιτήσεων με συγκεκριμένες περιόδους.
- Προβολή και επεξεργασία αιτήσεων από φοιτητές ή διαχειριστές.
- Αποθήκευση βασικών πληροφοριών, όπως πρωτόκολλο και στοιχεία δεξιοτήτων.

3. Πίνακας files

Ο πίνακας files διαχειρίζεται τα συνημμένα αρχεία που συνδέονται με τις αιτήσεις. Κάθε εγγραφή περιλαμβάνει πληροφορίες όπως applicationid, userid, το πεδίο στο οποίο ανήκει το αρχείο (field), και το όνομα του αρχείου.

Λειτουργίες:

- Διαχείριση συνημμένων αρχείων (ανέβασμα, προβολή, διαγραφή).
- Σύνδεση αρχείων με αιτήσεις και χρήστες.
- Ασφαλής αποθήκευση και ανάκτηση αρχείων στον server.

4. Πίνακας periods

Ο πίνακας periods περιέχει δεδομένα για τις περιόδους κατά τις οποίες μπορούν να υποβάλλονται αιτήσεις. Περιλαμβάνει πεδία όπως periodtitle, start_date, και end_date.

Λειτουργίες:

- Διαχείριση περιόδων από διαχειριστές και εκπαιδευτικούς.
- Σύνδεση αιτήσεων με συγκεκριμένες περιόδους.
- Εξασφάλιση της υποβολής αιτήσεων εντός των προκαθορισμένων χρονικών ορίων.

5. Πίνακας period_access

Ο πίνακας period_access ελέγχει την πρόσβαση των χρηστών στις περιόδους. Κάθε εγγραφή περιλαμβάνει τα πεδία userid, periodid, και access_type, το οποίο καθορίζει αν ο χρήστης έχει δικαίωμα προβολής ή αίτησης.

Λειτουργίες:

- Ορισμός δικαιωμάτων πρόσβασης χρηστών σε περιόδους.
- Καθορισμός των χρηστών που μπορούν να διαχειριστούν ή να δουν αιτήσεις για συγκεκριμένες περιόδους.
- Εξασφάλιση της σωστής οργάνωσης και ασφάλειας των περιόδων.

Οι πίνακες συνεργάζονται αρμονικά για την υποστήριξη του συστήματος. Ο users λειτουργεί ως ο κεντρικός πίνακας για την ταυτοποίηση χρηστών. Ο applications και ο files συνεργάζονται για τη διαχείριση των αιτήσεων και των συνημμένων αρχείων. Ο periods εξασφαλίζει την οργάνωση σε χρονικές περιόδους, ενώ ο period_access ελέγχει την πρόσβαση των χρηστών σε αυτές. Μέσα από αυτή τη δομή, το σύστημα διασφαλίζει λειτουργικότητα, διαφάνεια και ασφάλεια.

4.5 Ασφάλεια στην πλατφόρμα

Η ασφάλεια αποτελεί έναν από τους πιο κρίσιμους παράγοντες στην ανάπτυξη και λειτουργία της πλατφόρμας διαχείρισης αιτήσεων. Με δεδομένο ότι το σύστημα επεξεργάζεται ευαίσθητα προσωπικά δεδομένα και αρχεία χρηστών, η εξασφάλιση της προστασίας των δεδομένων από μη εξουσιοδοτημένη πρόσβαση, αλλοίωση ή απώλεια είναι πρωταρχικής σημασίας.

4.5.1 Κρυπτογράφηση Δεδομένων

Η πλατφόρμα χρησιμοποιεί σύγχρονες τεχνικές κρυπτογράφησης για την προστασία των δεδομένων:

- **Κρυπτογράφηση Κωδικών Πρόσβασης**: Οι κωδικοί πρόσβασης των χρηστών αποθηκεύονται στη βάση δεδομένων σε μορφή κρυπτογραφημένου hash (π.χ., bcrypt). Αυτό καθιστά αδύνατη την αποκρυπτογράφηση τους, ακόμη και αν η βάση δεδομένων παραβιαστεί.

4.5.2 Διαχείριση Πρόσβασης

Η διαχείριση πρόσβασης υλοποιείται μέσω ενός συστήματος ρόλων και δικαιωμάτων:

- **Ρόλοι Χρηστών**: Οι χρήστες κατηγοριοποιούνται σε φοιτητές, εκπαιδευτικούς και διαχειριστές. Κάθε ρόλος έχει καθορισμένα δικαιώματα και περιορισμούς.
- **Περιορισμός Πρόσβασης**: Οι χρήστες έχουν πρόσβαση μόνο στα δεδομένα που σχετίζονται με τον ρόλο και την ταυτότητά τους. Για παράδειγμα, ένας φοιτητής μπορεί να δει και να επεξεργαστεί μόνο τις δικές του αιτήσεις, ενώ οι εκπαιδευτικοί μπορούν να έχουν πρόσβαση μόνο στις αιτήσεις των φοιτητών που ανήκουν στις δικές τους περιόδους.

4.5.3 Ασφάλεια Αρχείων

Τα αρχεία που ανεβαίνουν στην πλατφόρμα προστατεύονται με διάφορους μηχανισμούς:

- **Ασφαλής Αποθήκευση**: Τα αρχεία αποθηκεύονται σε ασφαλή διαδρομή στον διακομιστή και δεν είναι προσβάσιμα απευθείας από τον ιστό.
- **Επαλήθευση Αρχείων**: Κατά το ανέβασμα, τα αρχεία ελέγχονται για το μέγεθος και τον τύπο τους (π.χ., PDF, JPEG, PNG), μειώνοντας τον κίνδυνο κακόβουλων αρχείων.
- **Διαχείριση Δικαιωμάτων Πρόσβασης**: Μόνο ο χρήστης που ανέβασε το αρχείο ή οι αρμόδιοι διαχειριστές μπορούν να το προβάλουν ή να το κατεβάσουν.

4.5.4 Προστασία από Επιθέσεις

Το σύστημα έχει σχεδιαστεί για να αντέχει σε διάφορους τύπους επιθέσεων:

- **Προστασία από SQL Injection**: Όλες οι ερωτήσεις προς τη βάση δεδομένων χρησιμοποιούν παραμετροποιημένα ερωτήματα, αποτρέποντας την εκτέλεση κακόβουλου κώδικα.
- **Προστασία από XSS (Cross-Site Scripting)**: Οι εισαγόμενες τιμές από τους χρήστες ελέγχονται και καθαρίζονται, διασφαλίζοντας ότι δεν εισάγεται κακόβουλος κώδικας στις σελίδες.

- **Περιορισμός Αιτημάτων**: Με τη χρήση μηχανισμών όπως το rate limiting, αποτρέπεται η κατάχρηση του συστήματος από bot ή κακόβουλους χρήστες.

Κεφάλαιο 5ο: Συμπεράσματα - βελτιώσεις

Το σύστημα διαχείρισης αιτήσεων που αναπτύχθηκε παρουσιάστηκε ως μια ολοκληρωμένη λύση για την αυτοματοποίηση της διαδικασίας υποβολής, διαχείρισης και παρακολούθησης αιτήσεων. Η υλοποίησή του έφερε στο φως τις δυνατότητες της τεχνολογίας να βελτιώσει την εμπειρία των χρηστών, να εξοικονομήσει χρόνο και να αυξήσει την αποδοτικότητα σε εκπαιδευτικά και διοικητικά περιβάλλοντα. Τα κύρια χαρακτηριστικά του περιλάμβαναν την αυτοματοποίηση της διαδικασίας διαχείρισης αιτήσεων, την ενίσχυση της ασφάλειας για την προστασία των προσωπικών δεδομένων και ένα φιλικό προς τον χρήστη περιβάλλον που εξασφαλίζει την ευκολία πρόσβασης για όλους.

Η αυτοματοποίηση που επιτεύχθηκε μείωσε σημαντικά την ανάγκη για χειροκίνητη παρέμβαση, διευκολύνοντας τόσο τους χρήστες όσο και τους διαχειριστές. Επιπλέον, η υιοθέτηση προχωρημένων μέτρων ασφαλείας ενίσχυσε την εμπιστοσύνη των χρηστών, ενώ το διαφανές περιβάλλον τους παρείχε πλήρη έλεγχο και παρακολούθηση της πορείας των αιτήσεών τους. Ωστόσο, η ανάπτυξη της πλατφόρμας ανέδειξε και ορισμένες προκλήσεις, οι οποίες προσφέρουν ευκαιρίες για μελλοντική βελτίωση.

Μια πρώτη κατεύθυνση για βελτίωση αφορά την απόδοση της πλατφόρμας. Η εφαρμογή τεχνικών caching θα μπορούσε να μειώσει τους χρόνους απόκρισης, ειδικά κατά την περίοδο έντονης δραστηριότητας, όταν μεγάλος αριθμός χρηστών υποβάλλει ή επεξεργάζεται αιτήσεις ταυτόχρονα. Παράλληλα, η προσθήκη λειτουργιών αναλυτικών στοιχείων, όπως αναφορές και γραφήματα, θα ενίσχυε τη δυνατότητα των διαχειριστών να αξιολογούν τα δεδομένα αιτήσεων και να λαμβάνουν τεκμηριωμένες αποφάσεις.

Ένας άλλος τομέας που θα μπορούσε να βελτιωθεί είναι η προσαρμοστικότητα της πλατφόρμας. Η δυνατότητα εξατομίκευσης της λειτουργικότητας και της εμφάνισης ανάλογα με τις ανάγκες διαφορετικών οργανισμών ή τμημάτων θα αύξανε τη χρηστικότητά της. Παράλληλα, η υποστήριξη για πολλαπλές γλώσσες θα έκανε το σύστημα πιο προσιτό σε ευρύτερες ομάδες χρηστών, διευκολύνοντας τη χρήση του σε πολυπολιτισμικά περιβάλλοντα.

Επιπλέον, η ενσωμάτωση τεχνολογιών τεχνητής νοημοσύνης θα μπορούσε να προσφέρει σημαντικά πλεονεκτήματα, όπως η αυτόματη ανίχνευση λαθών ή ελλείψεων στις αιτήσεις, καθιστώντας τη διαδικασία αξιολόγησης πιο αποτελεσματική και λιγότερο επιρρεπή σε ανθρώπινα λάθη. Παρόλο που η πλατφόρμα εφαρμόζει ισχυρά μέτρα ασφάλειας, είναι σημαντικό να υπάρχει συνεχής επανεξέταση και αναβάθμιση των πρωτοκόλλων για την αντιμετώπιση νέων απειλών. Αυτό θα διασφαλίσει τη διατήρηση ενός υψηλού επιπέδου προστασίας για τα δεδομένα των χρηστών.

Τέλος, αν και η πλατφόρμα είναι σχεδιασμένη ώστε να είναι προσβάσιμη από κινητές συσκευές, η ανάπτυξη μιας ειδικής εφαρμογής για κινητά θα ενίσχυε την εμπειρία χρήσης, παρέχοντας πιο άμεση πρόσβαση και προσαρμοσμένες λειτουργίες. Με αυτές τις βελτιώσεις, η πλατφόρμα θα μπορούσε να

καταστεί ακόμα πιο ισχυρή και ευέλικτη, ανταποκρινόμενη καλύτερα στις αυξανόμενες απαιτήσεις και τις ανάγκες των χρηστών.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.submittable.com/

[2] https://www.open-conf.gr/

[3] https://www.silverchair.com/scholarone/

[4] https://docs.google.com/forms/u/0/

[5] https://www.efka.gov.gr/el/elektronikes-yperesies/syntaxioychoi/aitisi

[6] https://www.efka.gov.gr/el/menoy/sychnes-eroteseis/paroches-kai-ygeia/kepa/dikaioychoi-parochon-kepa/aitese-kepa

[7] https://eurep.auth.gr/el/students/international/studies/online-application

[8] https://eurep.auth.gr/el/students/traineeship/application_procedure

[9] https://en.wikipedia.org/wiki/Laravel

[10] https://laravel.com/docs/11.x

[11] https://www.glorywebs.com/blog/laravel-usage-statistics

[12] https://www.fastfwd.com/why-choose-laravel-for-your-next-web-project/

[13] https://scaleupally.io/blog/benefits-of-laravel-framework/

[14] https://dolphinwebsolution.com/blog/advantages-and-disadvantages-of-laravel-development/

[15] https://en.wikipedia.org/wiki/PHP

[16] https://codeinstitute.net/global/blog/what-is-php-programming/

[17] https://firstsiteguide.com/php-stats/

[18] https://ellow.io/advantages-and-disadvantages-of-php/

[19] https://db-engines.com/en/system/MySQL

[20] https://www.tessell.com/blogs/mysql-concepts-benefits-and-use-cases

[21] https://blueclawdb.com/mysql/advantages-disadvantages-mysql/

ΠΑΡΑΡΤΗΜΑ Α

Στο παράρτημα αυτό αναφέρονται τα βασικά κομμάτια του κώδικα που χρησιμοποιήθηκε.

Routes

```
<?php

use Illuminate\Support\Facades\Route;
use App\Http\Controllers\UserController;
use App\Http\Controllers\StudentController;
use App\Http\Controllers\TeacherController;
use App\Http\Controllers\AdminController;

Route::get('/', [UserController::class, 'welcome'])->name('welcome');
Route::get('/login', [UserController::class, 'showLoginForm'])->name('login');
Route::post('/login', [UserController::class, 'login']);
Route::post('/logout', [UserController::class, 'logout'])->name('logout');
Route::get('/register', [UserController::class, 'showRegisterForm'])->name('register');
Route::post('/register', [UserController::class, 'register']);
Route::get('/forgot-password', [UserController::class, 'showForgotPasswordForm'])-
>name('password.request');
Route::post('/forgot-password', [UserController::class, 'sendResetLinkEmail'])-
>name('password.email');

// Dashboard
Route::get('/dashboard', [UserController::class, 'dashboard'])->name('dashboard');
Route::post('/logout', [UserController::class, 'logout'])->name('logout');

// Student Routes
Route::get('/application/{id}', [StudentController::class, 'show'])->name('student.application.show');
Route::get('/ggg/aaa', [StudentController::class, 'aaa'])->name('student.application.create');
Route::post('/applications', [StudentController::class, 'store'])->name('student.application.store');
Route::get('/applications', [StudentController::class, 'index'])->name('student.applications.index');
Route::get('/applications/{id}/edit', [StudentController::class, 'edit'])-
>name('student.application.edit');
Route::put('/applications/{id}', [StudentController::class, 'update'])-
>name('student.application.update');
Route::delete('/files/{id}', [StudentController::class, 'deleteFile'])->name('student.file.delete');
Route::delete('/applications/{id}', [StudentController::class, 'destroy'])-
>name('student.application.destroy');

// Teacher Routes
Route::get('/teacher/periods', [TeacherController::class, 'listPeriods'])-
>name('teacher.period.view');
Route::get('/teacher/periods/{id}/students', [TeacherController::class, 'viewStudents'])-
>name('teacher.period.students');
Route::get('/teacher/students/{id}/applications', [TeacherController::class, 'viewApplication'])-
>name('teacher.student.application');
```

```
Route::get('/teacher/periods/create', [TeacherController::class, 'createPeriod'])-
>name('teacher.period.create');
Route::post('/teacher/periods', [TeacherController::class, 'storePeriod'])-
>name('teacher.period.store');
Route::get('/teacher/periods/{id}/edit', [TeacherController::class, 'editPeriod'])-
>name('teacher.period.edit');
Route::put('/teacher/periods/{id}', [TeacherController::class, 'updatePeriod'])-
>name('teacher.period.update');

Route::get('/teacher/periodslist', [TeacherController::class, 'listesApoPeriods'])-
>name('teacher.period.list');

// Admin Routes
Route::get('/admin/periods', [AdminController::class, 'listPeriods'])->name('admin.periods');
Route::get('/admin/periods/{id}/students', [AdminController::class, 'viewStudents'])-
>name('admin.period.students');
Route::get('/admin/students/{id}/applications', [AdminController::class, 'viewApplication'])-
>name('admin.student.application');
```

StudentController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Log;

class StudentController extends Controller
{
    public function index()
    {
        $userId = Session::get('user_id');
        $applications = DB::table('applications')
            ->join('periods', 'applications.periodid', '=', 'periods.id') // Join με periods
            ->where('applications.userid', $userId)
            ->select(
                'applications.*',
                'periods.periodtitle',
                'periods.start_date',
                'periods.end_date'
```

```
        )
        ->get();

    return view('student.viewapplication', ['applications' => $applications]);
}



public function aaa()
{
    Log::info('create');

    $userId = Session::get('user_id');
    $periods = DB::table('period_access')
        ->join('periods', 'period_access.periodid', '=', 'periods.id')
        ->where('period_access.userid', $userId)
        ->select('periods.*')
        ->get();

    return view('student.createapplication', ['periods' => $periods]);
}

public function store(Request $request)
{
    $request->validate([
        'periodid' => 'required|integer',
        'firstname' => 'required|string|max:100',
        'lastname' => 'required|string|max:100',
        'parentname' => 'nullable|string|max:100',
        'idn' => 'nullable|string|max:10',
        'yearofbirth' => 'nullable|string|max:10',
        'city' => 'nullable|string|max:30',
        'tel' => 'nullable|string|max:20',
        'skill_maindegreetitle' => 'nullable|string|max:200',
        'skill_maindegreegrade' => 'required|numeric|min:0|max:10',
        'skill_occupation' => 'nullable|string|max:400',
        'skill_occupationmonths' => 'required|integer|min:0',
        'skill_agree' => 'required|boolean',
        'file_maindegree' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        'file_occupationmonths' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        'file_langdegree' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
```

```
            'file_cv' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        ]);

        // Δημιουργία αίτησης
        $applicationId = DB::table('applications')->insertGetId([
            'userid' => Session::get('user_id'),
            'periodid' => $request->periodid,
            'firstname' => $request->firstname,
            'lastname' => $request->lastname,
            'parentname' => $request->parentname,
            'idn' => $request->idn,
            'yearofbirth' => $request->yearofbirth,
            'city' => $request->city,
            'tel' => $request->tel,
            'skill_maindegreetitle' => $request->skill_maindegreetitle,
            'skill_maindegreegrade' => $request->skill_maindegreegrade,
            'skill_occupation' => $request->skill_occupation,
            'skill_occupationmonths' => $request->skill_occupationmonths,
            'skill_agree' => $request->skill_agree,
            'protocol_number' => strtoupper(uniqid('PR')), // Μοναδικός αριθμός πρωτοκόλλου
            'created_at' => now(),
        ]);

        // Αποθήκευση αρχείων
        // $fileData = [];
          // foreach (['file_maindegree', 'file_occupationmonths', 'file_langdegree', 'file_cv'] as
$fileField) {
        //     if ($request->hasFile($fileField)) {
        //         $fileName = $request->file($fileField)->store('uploads', 'public');
        //         $fileData[$fileField] = $fileName;
        //     }
        // }

        // if (!empty($fileData)) {
        //     $fileData['userid'] = Session::get('user_id');
        //     $fileData['applicationid'] = $applicationId;
        //     DB::table('files')->insert($fileData);
        // }

            // Αποθήκευση αρχείων
```

```
        foreach (['file_maindegree', 'file_occupationmonths', 'file_langdegree', 'file_cv'] as $field)
{
            if ($request->hasFile($field)) {
                foreach ($request->file($field) as $file) {
                    $fileName = $file->store('uploads', 'public');
                    DB::table('files')->insert([
                        'userid' => Session::get('user_id'),
                        'applicationid' => $applicationId,
                        'field' => $field,
                        'filename' => $fileName,
                        'created_at' => now(),
                    ]);
                }
            }
        }

        return redirect()->route('student.applications.index')->with('success', 'Η αίτηση δημιουργήθηκε
επιτυχώς.');
    }

    public function deleteFile($id)
    {
        $file = DB::table('files')->where('id', $id)->first();

        if (!$file || $file->userid != Session::get('user_id')) {
            return response()->json(['error' => 'Μη εξουσιοδοτημένη πρόσβαση.'], 403);
        }

        // Διαγραφή του αρχείου από τον δίσκο
        Storage::disk('public')->delete($file->filename);

        // Διαγραφή του αρχείου από τη βάση δεδομένων
        DB::table('files')->where('id', $id)->delete();

        return response()->json(['success' => 'Το αρχείο διαγράφηκε επιτυχώς.']);
    }




    public function edit($id)
    {
```

```
        $userId = Session::get('user_id');

        // Φόρτωση αίτησης
        $application = DB::table('applications')
            ->where('id', $id)
            ->where('userid', $userId)
            ->first();

        if (!$application) {
            abort(404, 'Η αίτηση δεν βρέθηκε.');
        }

        // Έλεγχος προθεσμίας
        $currentDate = now();
        $period = DB::table('periods')->where('id', $application->periodid)->first();
        $isEditable = $currentDate->between($period->start_date, $period->end_date);

        // Φόρτωση αρχείων
        $files = DB::table('files')
            ->where('applicationid', $id)
            ->get();

        // Φόρτωση περιόδων
        $periods = DB::table('period_access')
            ->join('periods', 'period_access.periodid', '=', 'periods.id')
            ->where('period_access.userid', $userId)
            ->select('periods.*')
            ->get();

        return view('student.editorviewapplication', [
            'application' => $application,
            'files' => $files,
            'periods' => $periods,
            'isEditable' => $isEditable,
        ]);
    }

    //1:file_maindegree, 2:file_occupationmonths, 3:file_langdegree, 4:file_cv

public function update(Request $request, $id)
```

```
{
    $request->validate([
        'periodid' => 'required|integer',
        'firstname' => 'required|string|max:100',
        'lastname' => 'required|string|max:100',
        'parentname' => 'nullable|string|max:100',
        'idn' => 'nullable|string|max:10',
        'yearofbirth' => 'nullable|string|max:10',
        'city' => 'nullable|string|max:30',
        'tel' => 'nullable|string|max:20',
        'skill_maindegreetitle' => 'nullable|string|max:200',
        'skill_maindegreegrade' => 'required|numeric|min:0|max:10',
        'skill_occupation' => 'nullable|string|max:400',
        'skill_occupationmonths' => 'required|integer|min:0',
        'skill_agree' => 'required|boolean',
        'file_maindegree' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        'file_occupationmonths' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        'file_langdegree' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
        'file_cv' => 'nullable|file|mimes:pdf,jpg,png|max:2048',
    ]);

    // Ενημέρωση αίτησης
    DB::table('applications')
        ->where('id', $id)
        ->where('userid', Session::get('user_id'))
        ->update([
            'periodid' => $request->periodid,
            'firstname' => $request->firstname,
            'lastname' => $request->lastname,
            'parentname' => $request->parentname,
            'idn' => $request->idn,
            'yearofbirth' => $request->yearofbirth,
            'city' => $request->city,
            'tel' => $request->tel,
            'skill_maindegreetitle' => $request->skill_maindegreetitle,
            'skill_maindegreegrade' => $request->skill_maindegreegrade,
            'skill_occupation' => $request->skill_occupation,
            'skill_occupationmonths' => $request->skill_occupationmonths,
            'skill_agree' => $request->skill_agree,
            'updated_at' => now(),
```

```
        ]);

    // Αποθήκευση νέων αρχείων
    $fields = [
        'file_maindegree' => 1,
        'file_occupationmonths' => 2,
        'file_langdegree' => 3,
        'file_cv' => 4,
    ];

    foreach ($fields as $fieldName => $fieldValue) {
        if ($request->hasFile($fieldName)) {
            $file = $request->file($fieldName);
            $fileName = $file->store('uploads', 'public');

            DB::table('files')->insert([
                'userid' => Session::get('user_id'),
                'applicationid' => $id,
                'field' => $fieldValue,
                'filename' => $fileName,
                'created_at' => now(),
            ]);
        }
    }

    return redirect()->route('student.application.edit', $id)
        ->with('success', 'Η αίτηση ενημερώθηκε επιτυχώς.');
}

public function show($id)
{
    $userId = Session::get('user_id');
    $userRole = Session::get('user_role'); // 0: student, 1: teacher, 2: admin

    // Φόρτωση της αίτησης
    $application = DB::table('applications')->where('id', $id)->first();

    if (!$application) {
        abort(404, 'Η αίτηση δεν βρέθηκε.');
    }
```

```
    // Αν είναι teacher, έλεγχος αν η αίτηση ανήκει σε επιτρεπόμενη περίοδο
    if ($userRole == 1) { // Teacher
        $hasAccess = DB::table('period_access')
            ->where('userid', $userId)
            ->where('periodid', $application->periodid)
            ->exists();

        if (!$hasAccess) {
            abort(403, 'Δεν έχετε πρόσβαση σε αυτή την αίτηση.');
        }
    }

    // Αν είναι student, έλεγχος αν η αίτηση ανήκει στον χρήστη
    if ($userRole == 0 && $application->userid != $userId) {
        abort(403, 'Δεν έχετε πρόσβαση σε αυτή την αίτηση.');
    }

    // Φόρτωση περιόδου της αίτησης
    $period = DB::table('periods')->where('id', $application->periodid)->first();

    // Φόρτωση αρχείων
    $files = DB::table('files')
        ->where('applicationid', $id)
        ->get();

    // Η αίτηση είναι μόνο για προβολή
    $isEditable = false;

    return view('student.editorviewapplication', [
        'application' => $application,
        'files' => $files,
        'periods' => [$period], // Εμφάνιση μόνο της περιόδου της αίτησης
        'isEditable' => $isEditable,
    ]);
}

    public function destroy($id)
    {
        DB::table('applications')->where('id', $id)->delete();
```

```
        return redirect()->route('student.applications.index');
    }
}
```

TeacherController

```
<?php

namespace App\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Support\Facades\DB;
use Illuminate\Support\Facades\Session;
use Illuminate\Support\Facades\Log;

class TeacherController extends Controller
{
    public function listPeriods()
    {
        Log::info('listPeriods');

        if (!Session::has('user_id')) {
            return redirect()->route('login');
        }

        $teacherId = Session::get('user_id');
        $periods = DB::table('period_access')
            ->join('periods', 'period_access.periodid', '=', 'periods.id')
            ->where('period_access.userid', $teacherId)
            ->select('periods.*')
            ->get();

        return view('teacher.viewperiods', ['periods' => $periods]);
    }

    public function viewStudents($periodId)
    {
        $students = DB::table('applications')
            ->join('users', 'applications.userid', '=', 'users.id')
            ->where('applications.periodid', $periodId)
```

```
            ->select('users.id', 'users.firstname', 'users.lastname', 'users.email')
            ->get();

        return view('teacher.viewstudents', ['students' => $students]);
    }



    public function viewApplication($studentId)
    {
        $application = DB::table('applications')->where('userid', $studentId)->first();
        return view('teacher.students.application', ['application' => $application]);
    }

    public function createPeriod()
    {
        return view('teacher.createperiod');
    }

    public function storePeriod(Request $request)
    {
        $request->validate([
            'periodtitle' => 'required|string|max:30',
            'start_date' => 'required|date',
            'end_date' => 'required|date|after_or_equal:start_date',
        ]);

        DB::table('periods')->insert([
            'periodtitle' => $request->periodtitle,
            'start_date' => $request->start_date,
            'end_date' => $request->end_date,
            'created_at' => now(),
            'updated_at' => now(),
        ]);

          return redirect()->route('teacher.period.list')->with('success', 'Η περίοδος δημιουργήθηκε επιτυχώς.');
    }

    public function editPeriod($id)
    {
        $period = DB::table('periods')->where('id', $id)->first();
```

```
        if (!$period) {
            abort(404, 'Η περίοδος δεν βρέθηκε.');
        }

        return view('teacher.editperiod', ['period' => $period]);
    }

    public function updatePeriod(Request $request, $id)
    {
        $request->validate([
            'periodtitle' => 'required|string|max:30',
            'start_date' => 'required|date',
            'end_date' => 'required|date|after_or_equal:start_date',
        ]);

        DB::table('periods')
            ->where('id', $id)
            ->update([
                'periodtitle' => $request->periodtitle,
                'start_date' => $request->start_date,
                'end_date' => $request->end_date,
                'updated_at' => now(),
            ]);

           return redirect()->route('teacher.period.list')->with('success', 'Η περίοδος ενημερώθηκε επιτυχώς.');
    }

    public function listesApoPeriods()
{
    $periods = DB::table('periods')->get();

    return view('teacher.listperiods', ['periods' => $periods]);
}

}
```