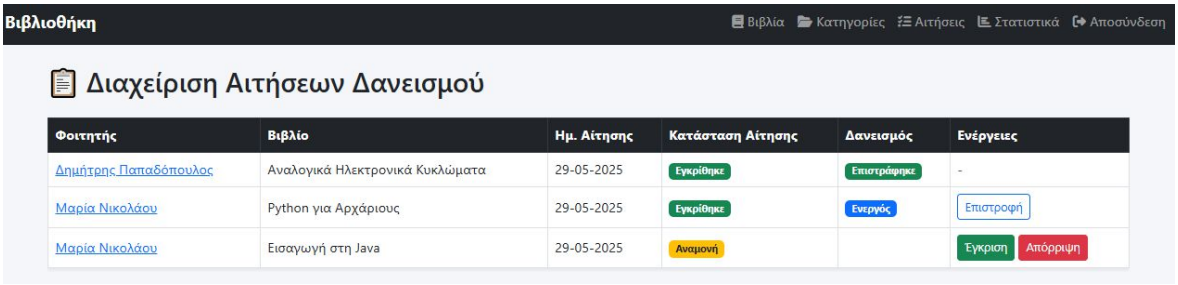
ΔΙΕΘΝΕΣ ΠΑΝΕΠΙΣΤΗΜΙΟ ΤΗΣ ΕΛΛΑΔΟΣ

ΣΧΟΛΗ ΜΗΧΑΝΙΚΩΝ

ΤΜΗΜΑ ΜΗΧΑΝΙΚΩΝ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΗΛΕΚΤΡΟΝΙΚΩΝ ΣΥΣΤΗΜΑΤΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ

«Ανάπτυξη διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης»



Φοιτητής

ΓΕΩΡΓΙΟΣ ΜΕΝΕΚΟΣ - 515084

Επιβλέπων

Κυριάκος Τσιακμάκης

Επ. Καθηγητής

Ιούνιος 2025

Ανάπτυξη διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης

Κωδικός: 25154

Φοιτητής: Μενέκος Γεώργιος

Εισηγητής: Δρ Κυριάκος Τσιακμάκης

Ημερομηνία ανάληψης Π.Ε. 08-03-2025

Ημερομηνία περάτωσης Π.Ε. 28-05-2025

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 περιγράφει την ανάπτυξη μιας διαδικτυακής πλατφόρμας για τη διαχείριση και τον δανεισμό βιβλίων, σχεδιασμένη για να εξυπηρετεί τις ανάγκες ενός εκπαιδευτικού τμήματος. Οι φοιτητές μπορούν να δουν τα διαθέσιμα βιβλία, να κάνουν αιτήσεις δανεισμού και να παρακολουθούν την πορεία τους. Ο διαχειριστής έχει πρόσβαση σε λειτουργίες έγκρισης, επιστροφής, στατιστικών και διαχείρισης καταλόγου. Η πλατφόρμα είναι απλή στη χρήση, φιλική προς τον χρήστη και υποστηρίζεται από σύστημα ειδοποιήσεων. Το έργο καλύπτει τις βασικές ανάγκες μιας μικρής βιβλιοθήκης με δυνατότητα μελλοντικής επέκτασης.

« Development of an online library management and lending platform»

Abstract

This paper describes the development of an online platform for book management and lending, designed to serve the needs of an educational department. Students can view available books, make loan requests and monitor their progress. The administrator has access to approval, return, statistics and catalog management functions. The platform is simple to use, user-friendly and supported by a notification system. The project covers the basic needs of a small library with the possibility of future expansion.

Ευχαριστίες

Θέλω να ευχαριστήσω τους γονείς μου, φίλους μου και τον επιβλέποντα για την υποστήριξη του σε θέματα κώδικα και καθοδήγησης.

Περιεχόμενα

Κατάλογος Σχημάτων

Κεφάλαιο 1ο: Εισαγωγή

1.1 Εισαγωγή

Ένας από τους βασικούς τομείς που έχει εξελιχθεί ραγδαία είναι η διαχείριση των βιβλιοθηκών με τις λεγόμενες ψηφιακές βιβλιοθήκες να αποτελούν πλέον βασικό εργαλείο για κάθε εκπαιδευτικό ή ερευνητικό ίδρυμα. Οι ψηφιακές βιβλιοθήκες δεν είναι απλώς αποθηκευτικοί χώροι αρχείων. Είναι συστήματα που επιτρέπουν την αναζήτηση, την πρόσβαση, τη διαχείριση και την παρακολούθηση των διαθέσιμων βιβλίων ή άλλων υλικών από οποιονδήποτε έχει δικαίωμα πρόσβασης.

Σε πολλές περιπτώσεις ιδιαίτερα σε τμήματα που διαθέτουν φυσικά αντίτυπα βιβλίων (π.χ. για μαθήματα ή εσωτερικές σημειώσεις), τίθεται το εξής ερώτημα, αν μπορεί να γίνει η καταγραφή και ο έλεγχος του δανεισμού αυτών των βιβλίων με απλό σύγχρονο αποδοτικό τρόπο. Ένα τετράδιο ή ένα πρόχειρο excel δεν αρκεί όταν θέλουμε διαφάνεια, ακρίβεια, στατιστικά και σύστημα πολλών χρηστών. Σε αυτές τις περιπτώσεις είναι απαραίτητο να υπάρχει ένα διαδικτυακό σύστημα δανεισμού όπου οι φοιτητές μπορούν να βλέπουν ποια βιβλία είναι διαθέσιμα και να κάνουν αίτηση και οι διαχειριστές να εγκρίνουν, να διαχειρίζονται τις επιστροφές.

Οι πλατφόρμες τέτοιου τύπου δεν είναι απλώς χρήσιμες αλλά είναι πλέον αναγκαίες για κάθε οργανωμένο ακαδημαϊκό χώρο. Έχουν ευκολία πρόσβασης, μείωση λαθών και δυνατότητα στατιστικής επεξεργασίας των δεδομένων. Οι φοιτητές εξυπηρετούνται ταχύτερα και οι διαχειριστές δεν χρειάζεται να διαχειρίζονται χαρτιά ή να ψάχνουν με το χέρι ποιος έχει τι.

Στα πλαίσια αυτής της εργασία, υλοποιήσαμε μια πλήρη διαδικτυακή εφαρμογή διαχείρισης και δανεισμού βιβλιοθήκης η οποία προσαρμόστηκε στις ανάγκες ενός ακαδημαϊκού τμήματος. Το σύστημα υποστηρίζει βασικούς ρόλους, όπως φοιτητές και διαχειριστές (admins). Οι φοιτητές μπορούν να περιηγηθούν στον κατάλογο των βιβλίων να κάνουν αιτήματα δανεισμού, να παρακολουθούν την πορεία των αιτήσεών τους και να τους ενημερώνονται μέσω ειδοποιήσεων. Οι διαχειριστές μπορούν να εγκρίνουν ή να απορρίπτουν αιτήσεις, να καταχωρούν νέους τίτλους, να διαχειρίζονται τις κατηγορίες βιβλίων, να επιστρέφουν βιβλία και να εξάγουν στατιστικά.

Η εφαρμογή αναπτύχθηκε με Python και Flask για τον server-side κώδικα και χρησιμοποιήθηκε MySQL για την αποθήκευση των δεδομένων. Το περιβάλλον χρήστη βασίστηκε σε Bootstrap, εξασφαλίζοντας έτσι λειτουργικότητα και ευχρηστία τόσο σε υπολογιστές όσο και σε κινητές συσκευές.

Στόχος της εργασίας είναι να παρουσιάσουμε αναλυτικά όλα τα στάδια της υλοποίησης και να δείξουμε τη χρησιμότητα μιας τέτοιας πλατφόρμας και να δείξουμε πώς μπορεί ένα απλό αλλά καλά οργανωμένο σύστημα να λύσει ένα πρόβλημα διαχείρισης φυσικών βιβλίων μέσα σε ένα τμήμα.

1.2 Η δομή εργασίας

Αρχικά παρουσιάζεται μια εισαγωγή στο θέμα με αναφορά στη σημασία των ψηφιακών βιβλιοθηκών και στην ανάγκη ύπαρξης διαδικτυακών πλατφορμών για τον δανεισμό φυσικών βιβλίων στο πλαίσιο ενός Τμήματος. Το επόμενο κεφάλαιο είναι αφιερωμένο στις τεχνολογίες που χρησιμοποιήθηκαν για την υλοποίηση του έργου περιλαμβάνοντας την Python, το Flask, τη MySQL και το Bootstrap με HTML για το frontend.

Στη συνέχεια ακολουθεί το κεφάλαιο, στο οποίο αναλύονται παρόμοια συστήματα, όπως το Koha, το Evergreen, το OPALS και το OpenBiblio. Παρουσιάζονται τα χαρακτηριστικά τους και εξετάζεται το δικό μας σύστημα σε σχέση με αυτά, με έμφαση στη χρησιμότητά του για εκπαιδευτικούς σκοπούς. Στο πέμπτο κεφάλαιο γίνεται αναλυτική παρουσίαση της ίδιας της πλατφόρμας. Περιλαμβάνεται η περιγραφή της βάσης δεδομένων, η αρχιτεκτονική της εφαρμογής, αναλυτικός σχολιασμός του βασικού κώδικα και παρουσίαση των κύριων λειτουργιών της πλατφόρμας. Επιπλέον περιγράφονται τα μέτρα που έχουν ληφθεί για την ασφάλεια του συστήματος. Τέλος στο έκτο κεφάλαιο καταγράφονται τα συμπεράσματα που προέκυψαν από την ανάπτυξη της εφαρμογής καθώς και προτάσεις για μελλοντικές βελτιώσεις και επεκτάσεις του συστήματος. Η εργασία ολοκληρώνεται με τη βιβλιογραφία που αξιοποιήθηκε κατά την εκπόνησή της και το παράρτημα που περιλαμβάνει ενδεικτικό κώδικα του συστήματος.

Κεφάλαιο 2ο: Τι χρησιμοποιήθηκε για την υλοποίηση της εργασίας

2.1 Python

Η Python είναι μια γλώσσα προγραμματισμού υψηλού επιπέδου, δυναμική, διαδραστική και ερμηνευόμενη, η οποία δημιουργήθηκε από τον Ολλανδό προγραμματιστή Guido van Rossum στις αρχές της δεκαετίας του 1990. Το πρώτο της release έγινε το 1991 και από τότε έχει ακολουθήσει μια συνεχώς ανοδική πορεία, εξελισσόμενη μέσα από μια κοινότητα ανοικτού κώδικα που παραμένει ιδιαίτερα ενεργή μέχρι και σήμερα.

Από την αρχή ο στόχος του van Rossum ήταν να δημιουργήσει μια γλώσσα που να είναι απλή στη χρήση, να έχει καθαρή σύνταξη και να επιτρέπει στους προγραμματιστές να εκφράζουν τις ιδέες τους με λιγότερες γραμμές κώδικα σε σχέση με άλλες γλώσσες της εποχής, όπως η C ή η Java [1-3]

Με την πάροδο του χρόνου, η Python απέκτησε ευρεία αποδοχή σε διάφορους τομείς της επιστήμης των υπολογιστών. Σήμερα χρησιμοποιείται σε πανεπιστήμια, επιχειρήσεις, startups, ερευνητικά κέντρα και κυβερνητικούς φορείς για εφαρμογές που εκτείνονται από την επιστήμη δεδομένων και την τεχνητή νοημοσύνη, μέχρι την ανάπτυξη εφαρμογών web και τον αυτοματισμό διαδικασιών.

Η μεγάλη επιτυχία της γλώσσας οφείλεται σε πολλούς παράγοντες: στη φιλοσοφία σχεδιασμού της, στην αναγνωσιμότητά της, στην τεράστια συλλογή βιβλιοθηκών που την πλαισιώνουν και στο γεγονός ότι είναι ανοικτού κώδικα και πλήρως δωρεάν για χρήση. Η Python έχει πλέον δύο κύριες εκδόσεις (2.x και 3.x), με τη Python 3 να είναι η επίσημα υποστηριζόμενη από το 2020 και μετά με σαφή προσανατολισμό προς την ασφάλεια, την απόδοση και τον καθαρότερο σχεδιασμό.

Η γλώσσα υποστηρίζεται από τον οργανισμό Python Software Foundation, ο οποίος αναπτύσσει την κύρια υλοποίηση του διερμηνέα και συντονίζει την εξέλιξη του οικοσυστήματος. Η κοινότητα της Python έχει συμβάλει στη δημιουργία χιλιάδων πακέτων και εργαλείων που επεκτείνουν τη λειτουργικότητά της σε όλους σχεδόν τους τομείς της σύγχρονης πληροφορικής.

Η γλώσσα Python, λοιπόν, δεν είναι απλώς ένα τεχνικό εργαλείο. Είναι ένα ισχυρό περιβάλλον ανάπτυξης, ιδανικό για εκπαιδευτικούς σκοπούς, πρωτότυπα έργα αλλά και για εφαρμογές παραγωγής μεγάλης κλίμακας. Η απλότητά της δεν μειώνει τις δυνατότητές της – αντίθετα, τις ενισχύει, κάνοντάς την προσβάσιμη και ισχυρή ταυτόχρονα.

Η Python είναι σχεδιασμένη με οδηγό την απλότητα, την αναγνωσιμότητα και την ευκολία στη χρήση. Αυτά τα χαρακτηριστικά την καθιστούν ιδιαίτερα δημοφιλή, τόσο σε αρχάριους όσο και σε επαγγελματίες προγραμματιστές. Παρακάτω περιγράφονται οι βασικότερες ιδιότητες που διακρίνουν τη γλώσσα Python και την καθιστούν ένα ισχυρό εργαλείο ανάπτυξης λογισμικού.

Ένα από τα πιο χαρακτηριστικά στοιχεία της Python είναι η καθαρή και απλή σύνταξή της. Αντί να χρησιμοποιεί άγκιστρα ({}) για τα μπλοκ εντολών όπως άλλες γλώσσες, η Python χρησιμοποιεί εσοχές (indentation), κάτι που όχι μόνο καθιστά τον κώδικα πιο ευανάγνωστο, αλλά και επιβάλλει καλές πρακτικές οργάνωσης του κώδικα από νωρίς. Η απλότητα αυτή είναι αποτέλεσμα του σχεδιαστικού μότο της γλώσσας: *“There should be one — and preferably only one — obvious way to do it.”*

Η Python είναι επίσης δυναμικά τυποποιημένη (dynamically typed). Αυτό σημαίνει ότι οι μεταβλητές δεν χρειάζονται ρητό καθορισμό τύπου κατά τη δήλωσή τους. Ο διερμηνέας καθορίζει αυτόματα τον τύπο βάσει της τιμής που του αποδίδεται. Αυτό επιταχύνει τη διαδικασία ανάπτυξης και κάνει τη γλώσσα πιο ευέλικτη.

Μια άλλη σημαντική ιδιότητα είναι ότι η Python είναι ερμηνευόμενη γλώσσα (interpreted). Δεν χρειάζεται μεταγλώττιση (compilation) για να εκτελεστεί ο κώδικας, κάτι που επιτρέπει ταχύτερο testing και debugging. Αυτό διευκολύνει ιδιαίτερα την ανάπτυξη εφαρμογών και την εκπαίδευση.

Η Python υποστηρίζει πολλαπλά παραδείγματα προγραμματισμού, όπως:

- Δομημένος προγραμματισμός (procedural)
- Αντικειμενοστραφής προγραμματισμός (object-oriented)
- Λειτουργικός προγραμματισμός (functional)

Αυτή η ευελιξία επιτρέπει στους προγραμματιστές να χρησιμοποιούν την προσέγγιση που ταιριάζει καλύτερα στις ανάγκες της εφαρμογής τους.

Επιπλέον, η Python διακρίνεται για την τεράστια βιβλιοθήκη εργαλείων (standard & third-party libraries). Υπάρχουν διαθέσιμες βιβλιοθήκες για επεξεργασία κειμένου, μαθηματικά, γραφικά, μηχανική μάθηση, web development, βάσεις δεδομένων, αισθητήρες IoT, και σχεδόν κάθε άλλο τομέα της πληροφορικής. Έτσι, η Python προσφέρει έτοιμες λύσεις για κοινά προβλήματα, μειώνοντας τον χρόνο ανάπτυξης.

Επίσης, η Python είναι επεκτάσιμη και ενσωματώσιμη. Μπορεί να ενσωματωθεί σε άλλα προγράμματα γραμμένα σε γλώσσες όπως η C ή η C++, και το αντίστροφο: μπορεί να καλεί βιβλιοθήκες άλλων γλωσσών μέσω bindings.

Τέλος, είναι πολλαπλής πλατφόρμας (cross-platform). Μπορεί να εκτελεστεί σε Windows, macOS, Linux και άλλες πλατφόρμες χωρίς αλλαγή στον κώδικα, γεγονός που την καθιστά κατάλληλη για ανάπτυξη εφαρμογών που πρέπει να λειτουργούν σε διαφορετικά περιβάλλοντα.

Η συνδυασμένη παρουσία όλων αυτών των χαρακτηριστικών έχει καθιερώσει την Python ως μία από τις πιο σημαντικές γλώσσες προγραμματισμού της εποχής μας. Είναι εξίσου κατάλληλη για γρήγορο πρωτότυπο (prototype) όσο και για μεγάλης κλίμακας συστήματα παραγωγής.

Η Python έχει κερδίσει την εμπιστοσύνη προγραμματιστών, πανεπιστημίων και εταιρειών χάρη στα πολλαπλά της πλεονεκτήματα. 4-5]

Ένα από τα κυριότερα πλεονεκτήματα της Python είναι η ευκολία στη μάθηση. Η σύνταξή της είναι τόσο απλή και καθαρή που μοιάζει με φυσική γλώσσα. Αυτό καθιστά την Python ιδανική για φοιτητές, εκπαιδευτικά ιδρύματα αλλά και για όσους ξεκινούν τον προγραμματισμό από την αρχή. Χωρίς περίπλοκες δηλώσεις, χωρίς αγκύλες και τελείες, ο μαθητής μπορεί να επικεντρωθεί στη λογική του προγράμματος.

Ακόμη η Python διαθέτει τεράστια ποικιλία έτοιμων βιβλιοθηκών και πακέτων. Αυτό σημαίνει ότι πολλές σύνθετες λειτουργίες μπορούν να υλοποιηθούν με λίγες γραμμές κώδικα, χάρη σε βιβλιοθήκες όπως requests, pandas, matplotlib, flask, django, numpy και πολλές άλλες. Αυτό το χαρακτηριστικό καθιστά τη γλώσσα εξαιρετικά παραγωγική και κατάλληλη για γρήγορη ανάπτυξη εφαρμογών (rapid development).

Η ευκολία συντήρησης και τροποποίησης του κώδικα είναι ακόμη ένα πλεονέκτημα και ιδιαίτερα σημαντικό για ομάδες ανάπτυξης ή έργα που συνεχώς εξελίσσονται. Ο αναγνώσιμος και δομημένος κώδικας μειώνει τα λάθη και διευκολύνει τη συνεργασία μεταξύ προγραμματιστών.

Η Python είναι επίσης πλατφορμικά ανεξάρτητη. Ένα πρόγραμμα μπορεί να γραφεί μία φορά και να τρέξει χωρίς αλλαγές σε Windows, Linux και macOS. Παράλληλα, ενδείκνυται και για scripting, αυτοματοποίηση διαδικασιών και εργασία με αρχεία ή βάσεις δεδομένων.

Βέβαια υπάρχουν και μειονεκτήματα που πρέπει να δούμε. Πρώτο και βασικότερο είναι ότι λόγω της ερμηνευόμενης φύσης της η απόδοσή της σε χρόνους εκτέλεσης είναι χαμηλότερη σε σύγκριση με compiled γλώσσες όπως η C++ ή η Java. Για εφαρμογές που απαιτούν ταχύτητα (όπως real-time συστήματα ή παιχνίδια υψηλής απόδοσης) η Python μπορεί να μην είναι η ιδανική επιλογή.

Ένα δεύτερο μειονέκτημα είναι η υψηλή κατανάλωση μνήμης. Σε περιβάλλοντα με αυστηρούς πόρους, όπως embedded συστήματα ή κινητές συσκευές, η Python μπορεί να είναι πιο απαιτητική από άλλες γλώσσες.

Η Python δεν είναι η καλύτερη επιλογή για mobile εφαρμογές καθώς η υποστήριξη για Android και iOS είναι περιορισμένη σε σύγκριση με άλλες τεχνολογίες όπως Kotlin, Swift ή React Native.

Το γεγονός ότι οι μεταβλητές δεν έχουν σταθερό τύπο μπορεί να οδηγήσει σε δυσδιάκριτα σφάλματα κατά την εκτέλεση, τα οποία δε θα εντοπιστούν κατά τη συγγραφή του κώδικα κάτι που ενδέχεται να αυξήσει το debugging time σε μεγάλα έργα.

Παρά τα μειονεκτήματα αυτά η Python συνεχίζει να αποτελεί μία από τις κορυφαίες επιλογές προγραμματισμού ειδικά σε έργα που δίνουν έμφαση στην ταχύτητα ανάπτυξης.

Η Python είναι μία από τις πιο ευρέως χρησιμοποιούμενες γλώσσες προγραμματισμού στον κόσμο όχι μόνο στην εκπαίδευση αλλά και σε πραγματικές εφαρμογές παραγωγής. Ο συνδυασμός απλότητας, και ισχυρών βιβλιοθηκών την έχει κάνει εργαλείο επιλογής για μια τεράστια γκάμα έργων από αυτοματισμούς γραφείου μέχρι τεχνητή νοημοσύνη και ανάπτυξη διαδικτυακών εφαρμογών.

Ένας από τους βασικότερους τομείς εφαρμογής της Python είναι η επιστήμη δεδομένων και η τεχνητή νοημοσύνη. Με βιβλιοθήκες όπως pandas, numpy, matplotlib, scikit-learn και tensorflow, η Python χρησιμοποιείται ευρέως για την ανάλυση δεδομένων, τη στατιστική επεξεργασία, την εξόρυξη γνώσης και την ανάπτυξη αλγορίθμων μηχανικής μάθησης. Πολλές εταιρείες τεχνολογίας, οικονομικών και υγειονομικής περίθαλψης στηρίζουν τις υπηρεσίες τους σε εργαλεία που έχουν αναπτυχθεί με Python.

Σημαντική είναι και η χρήση της στην ανάπτυξη web εφαρμογών. Frameworks όπως το Flask και το Django επιτρέπουν τη γρήγορη δημιουργία πλήρως λειτουργικών ιστοσελίδων και διαδικτυακών συστημάτων. Αυτά τα εργαλεία προσφέρουν δομή, ασφάλεια και ευκολία σύνδεσης με βάσεις δεδομένων και υποστηρίζουν καλές πρακτικές σχεδιασμού RESTful APIs και πολυεπίπεδης αρχιτεκτονικής.

Η Python χρησιμοποιείται ευρέως για automations και scripting, τόσο σε μικρές καθημερινές εργασίες όσο και σε διαδικασίες DevOps και system administration. Για παράδειγμα, μπορεί να χρησιμοποιηθεί για την αυτοματοποίηση δημιουργίας αναφορών, επεξεργασία αρχείων, σύνδεση με APIs, ή ακόμη και για τη διαχείριση cloud υποδομών μέσω εργαλείων όπως το Ansible.

Στον τομέα της ρομποτικής και των embedded συστημάτων η Python χρησιμοποιείται συχνά για την επικοινωνία με συσκευές όπως Raspberry Pi, Arduino (μέσω bridge), ESP32, καθώς και για τον έλεγχο αισθητήρων και ενεργοποιητών. Αν και σε επίπεδο real-time υλοποίησης δεν υπερτερεί η Python συχνά αξιοποιείται στο επίπεδο του ελέγχου, της διεπαφής ή της επικοινωνίας μεταξύ υποσυστημάτων.

Στο εκπαιδευτικό περιβάλλον, η Python έχει πλέον καθιερωθεί ως η γλώσσα εκκίνησης στην πλειοψηφία των προγραμμάτων σπουδών πληροφορικής παγκοσμίως. Ο συνδυασμός εύκολης σύνταξης, σαφούς δομής και μεγάλης τεκμηρίωσης την καθιστά ιδανική για μάθηση προγραμματιστικών εννοιών.

Χρήση της Python στο παρόν έργο

Στο πλαίσιο αυτής της εργασίας, επιλέχθηκε η Python για την υλοποίηση της διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλίων, με σκοπό την κάλυψη των αναγκών ενός Τμήματος που διαθέτει βιβλία σε φυσική μορφή και επιθυμεί να οργανώσει τη διαδικασία με τρόπο ψηφιακό.

Η χρήση της Python επέτρεψε την ταχύτατη ανάπτυξη ενός backend με το framework Flask, τη σύνδεση με MySQL βάση δεδομένων και την υλοποίηση λειτουργιών όπως η διαχείριση χρηστών, η καταχώρηση βιβλίων, οι αιτήσεις δανεισμού, οι ειδοποιήσεις και η προβολή στατιστικών. Η γλώσσα αποδείχθηκε ιδιαίτερα κατάλληλη για τον συγκεκριμένο τύπο εφαρμογής και η απλότητά της διευκόλυνε τη συνεργασία, την τεκμηρίωση αλλά και την ενσωμάτωση με άλλα εργαλεία του έργου, όπως Bootstrap για το frontend και DataTables για την παρουσίαση δεδομένων.

Η επιλογή της Python δεν έγινε μόνο για λόγους εκπαιδευτικούς αλλά και για πρακτικούς. Αποτελεί μια σταθερή επιλογή που εξασφαλίζει συντήρηση, επεκτασιμότητα και υποστήριξη στο μέλλον.

2.2 Flask

Το Flask είναι ένα δημοφιλές framework ανάπτυξης web εφαρμογών με τη γλώσσα προγραμματισμού Python. Δημιουργήθηκε με στόχο να είναι απλό, ελαφρύ και ευέλικτο για να μπορεί να χρησιμοποιηθεί τόσο από αρχάριους όσο και από προχωρημένους προγραμματιστές για την υλοποίηση διαδικτυακών εφαρμογών. Στην εργασία μας το Flask αποτελεί τη βασική υποδομή στην οποία βασίζεται όλη η λογική της πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης. [6-10]

Το κύριο χαρακτηριστικό του Flask είναι η φιλοσοφία "microframework". Αυτό σημαίνει ότι περιλαμβάνει μόνο τα απολύτως απαραίτητα εργαλεία για τη λειτουργία μιας web εφαρμογής: routing, request handling, rendering templates, διαχείριση session και cookies. Ο προγραμματιστής είναι ελεύθερος να επιλέξει ποια επιπλέον βιβλιοθήκη ή λειτουργικότητα θα προσθέσει, ανάλογα με τις ανάγκες της εφαρμογής. Έτσι, αποφεύγεται η πολυπλοκότητα που μπορεί να έχουν πιο “βαριά” frameworks.

Το Flask βασίζεται στην βιβλιοθήκη Werkzeug για το HTTP handling και στο Jinja2 για την παραγωγή HTML μέσω templates. Αυτά τα δύο εργαλεία είναι από τα πιο ισχυρά στο οικοσύστημα της Python και παρέχουν στον Flask σταθερότητα και ευελιξία. Επιπλέον, υποστηρίζει πλήρως JSON parsing, form processing και λειτουργίες authentication με χρήση εξωτερικών βιβλιοθηκών (π.χ. Flask-Login, Flask-WTF).

Σε αντίθεση με άλλα frameworks που επιβάλλουν συγκεκριμένη δομή φακέλων και αρχείων το Flask επιτρέπει στον προγραμματιστή να οργανώσει την εφαρμογή όπως εκείνος θεωρεί κατάλληλο. Αυτό είναι ιδιαίτερα χρήσιμο για εκπαιδευτικά έργα ή για συστήματα που αναπτύσσονται σταδιακά, όπως η πλατφόρμα μας.

Μια βασική Flask εφαρμογή μπορεί να αποτελείται από ένα μόνο αρχείο (π.χ. app.py) και λίγους φακέλους για templates (templates/) και στατικά αρχεία (static/). Από εκεί και πέρα, μπορεί να επεκταθεί με modular routing, blueprints, custom middleware και σύνδεση με βάσεις δεδομένων όπως MySQL, PostgreSQL ή SQLite.

Το Flask επιλέχθηκε για την ανάπτυξη της παρούσας πλατφόρμας για πολλούς πρακτικούς λόγους. Στην αρχή η καμπύλη μάθησης είναι ομαλή κάτι που επιτρέπει σε έναν φοιτητή ή αρχάριο

προγραμματιστή να κατανοήσει εύκολα πώς “ρολάρει” μια web εφαρμογή. Ακόμη, η απλότητα του routing (@app.route(...)) και της λογικής κάθε view function επιτρέπει στον χρήστη να έχει πλήρη έλεγχο της ροής δεδομένων και της επεξεργασίας κάθε αιτήματος.

Η δυνατότητα άμεσης σύνδεσης με MySQL και η εύκολη χρήση sessions και flash μηνυμάτων διευκόλυναν την υλοποίηση κρίσιμων λειτουργιών όπως: σύνδεση/αποσύνδεση χρηστών, προστασία routes ανά ρόλο, δυναμική διαχείριση αιτήσεων δανεισμού και αποστολή ειδοποιήσεων. Όλα αυτά χωρίς να χρειαστεί η χρήση βαρύγδουπων εργαλείων ή ORM συστημάτων, αφού έγινε χρήση απλών SQL queries.

Η ίδια η πλατφόρμα μπορεί εύκολα να επεκταθεί με API endpoints (RESTful), authentication tokens, προσθήκη email συστήματος και πολλές ακόμα δυνατότητες. Παράλληλα επειδή είναι τόσο διαδεδομένο το Flask υπάρχει πληθώρα παραδειγμάτων, tutorials και υποστήριξης στο διαδίκτυο που βοηθά την ομάδα ανάπτυξης ή τους φοιτητές που θα συνεχίσουν το έργο.

Ας υποθέσουμε ότι θέλουμε να δημιουργήσουμε μια σελίδα όπου ο χρήστης μπορεί να δει τις πληροφορίες ενός βιβλίου, με βάση το id του στη βάση δεδομένων.

Ο παρακάτω απλός κώδικας Flask δείχνει πώς μπορούμε να υλοποιήσουμε αυτή τη λειτουργία:

```
@app.route('/book/<int:book_id>')
def view_book(book_id):
    # Σύνδεση με τη βάση δεδομένων
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Εκτέλεση SQL εντολής για εύρεση βιβλίου με το συγκεκριμένο ID
    cursor.execute("""
        SELECT b.*, c.name AS category
        FROM books b
        JOIN categories c ON b.category_id = c.id
        WHERE b.id = %s
    """, (book_id,))

    # Ανάκτηση του βιβλίου ως dictionary
    book = cursor.fetchone()

    # Κλείσιμο σύνδεσης
    cursor.close()
    conn.close()

    # Αν δεν βρέθηκε βιβλίο, εμφάνιση μηνύματος και επιστροφή στην αρχική
    if not book:
        flash('Το βιβλίο δεν βρέθηκε.', 'danger')
```

```
    return redirect(url_for('welcome'))


# Αν βρέθηκε, εμφάνιση του template με τα δεδομένα του βιβλίου
return render_template('view_book.html', book=book)
```

Αυτό το απλό παράδειγμα δείχνει πόσο καθαρά και με σαφήνεια μπορεί κανείς να υλοποιήσει λειτουργικό backend με τη βοήθεια του Flask. Ο κώδικας είναι άμεσα κατανοητός ακόμα και από κάποιον που ξεκινά τώρα με την Python.

2.3 MySQL

Η MySQL είναι ένα από τα πιο διαδεδομένα συστήματα διαχείρισης σχεσιακών βάσεων δεδομένων (RDBMS) στον κόσμο. Αναπτύχθηκε αρχικά από τη σουηδική εταιρεία MySQL AB και πλέον διατηρείται από την Oracle. Η MySQL χρησιμοποιεί την γλώσσα SQL (Structured Query Language) για την εκτέλεση εντολών, όπως η εισαγωγή, ανάκτηση, τροποποίηση και διαγραφή δεδομένων. [11-14]

Στην πλατφόρμα που αναπτύχθηκε, η MySQL αναλαμβάνει τον κρίσιμο ρόλο της αποθήκευσης και οργάνωσης όλων των δεδομένων του συστήματος. Από τα στοιχεία των χρηστών και των βιβλίων μέχρι τις αιτήσεις δανεισμού και τις ειδοποιήσεις όλα αποθηκεύονται με δομημένο και αξιόπιστο τρόπο σε πίνακες της βάσης.

Η επιλογή της MySQL έγινε για πολλούς λόγους: είναι ελαφριά, υποστηρίζεται από πολλές γλώσσες προγραμματισμού (και ιδιαίτερα από την Python μέσω του mysql-connector), είναι ανοιχτού κώδικα και προσφέρει σταθερότητα και υψηλή απόδοση ακόμα και σε μεγάλα έργα.

Η σχεδίαση της βάσης έγινε με στόχο την κάλυψη όλων των λειτουργικών απαιτήσεων του συστήματος. Οι πιο σημαντικοί πίνακες περιλαμβάνουν:

- users: Αποθηκεύει πληροφορίες για τους χρήστες, όπως ονοματεπώνυμο, email, ρόλος (φοιτητής ή admin).
- books: Περιέχει όλα τα δεδομένα των βιβλίων (τίτλος, συγγραφέας, κατηγορία, πλήθος διαθέσιμων και συνολικών αντιτύπων).
- categories: Βοηθητικός πίνακας για τις κατηγορίες των βιβλίων.
- loan_requests: Κρατάει τα αιτήματα δανεισμού που υποβάλλουν οι φοιτητές, σε κατάσταση "pending", "approved" ή "rejected".
- loans: Καταγράφει τους ενεργούς ή παρελθόντες δανεισμούς που έχουν εγκριθεί και ολοκληρωθεί.
- notifications: Πίνακας που αποθηκεύει ειδοποιήσεις για αλλαγές κατάστασης (έγκριση, απόρριψη, επιστροφή).

Ένα παράδειγμα δημιουργίας του πίνακα των βιβλίων μπορεί να είναι το εξής:

```
CREATE TABLE books (
```

```
    id INT AUTO_INCREMENT PRIMARY KEY,
    title VARCHAR(255) NOT NULL,
    author VARCHAR(255),
    isbn VARCHAR(50),
    description TEXT,
    image_url TEXT,
    category_id INT,
    total_copies INT DEFAULT 1,
    available_copies INT DEFAULT 1
);
```

Η σύνδεση της εφαρμογής Flask με τη βάση γίνεται μέσω του mysql.connector:

```
import mysql.connector


def get_db_connection():
    return mysql.connector.connect(
        host='localhost',
        database='menekos',
        user='root',
        password=''
    )
```

Αυτό το απλό interface επιτρέπει στον προγραμματιστή να συνδεθεί και να διαχειρίζεται όλα τα δεδομένα της πλατφόρμας με ασφαλή τρόπο.

Πλεονεκτήματα της MySQL στην Εφαρμογή

1. Ταχύτητα και Απόδοση: Η MySQL είναι γνωστή για την υψηλή απόδοση, ακόμα και σε μεγάλα σύνολα δεδομένων.
2. Ασφάλεια: Υποστηρίζει έλεγχο πρόσβασης και ασφαλή επικοινωνία.
3. Συμβατότητα: Λειτουργεί άψογα με το Flask και το Python οικοσύστημα.
4. Ευκολία Χρήσης: Με απλές εντολές SQL μπορούμε να διαχειριστούμε τα δεδομένα του συστήματος με ακρίβεια.

2.4 Bootstrap - HTML

Για να μπορεί ένας χρήστης να αλληλεπιδράσει με μια διαδικτυακή εφαρμογή είναι απαραίτητη η ύπαρξη μιας κατανοητής, λειτουργικής και αισθητικά ευχάριστης διεπαφής. Στην πλατφόρμα

διαχείρισης και δανεισμού βιβλιοθήκης που αναπτύχθηκε kai η διεπαφή υλοποιήθηκε με HTML για τη δομή και το Bootstrap για το design, τη διάταξη και την προσαρμοστικότητα της σελίδας.[15-16]

Η HTML (HyperText Markup Language) αποτελεί τη βασική γλώσσα για τη δημιουργία των ιστοσελίδων. Με τη χρήση ετικετών (, <table>, <form>, <a> κ.λπ.),ορίζεται η δομή κάθε σελίδας του συστήματος: πίνακες βιβλίων, φόρμες σύνδεσης, διαχειριστικές καρτέλες, ειδοποιήσεις, κουμπιά και άλλα.

Σε κάθε template του συστήματος χρησιμοποιείται το Jinja2 engine του Flask το οποίο ενσωματώνει δυναμικό περιεχόμενο (όπως τα δεδομένα βιβλίων ή αιτήσεων) μέσα στο HTML με τη μορφή {{ μεταβλητή }} ή {% for item in items %}.

Παράδειγμα HTML μέσα σε Flask template:

```
<table class="table table-striped">
  <thead><tr><th>Τίτλος</th><th>Συγγραφέας</th></tr></thead>
  <tbody>
    {% for book in books %}
      <tr>
        <td>{{ book.title }}</td>
        <td>{{ book.author }}</td>
      </tr>
    {% endfor %}
  </tbody>
</table>
```

Το Bootstrap είναι ένα δημοφιλές CSS framework που επιτρέπει τη δημιουργία σύγχρονων και πλήρως responsive διεπαφών. Παρέχει έτοιμα στοιχεία όπως buttons, cards, grid σύστημα, φόρμες και ειδοποιήσεις (alerts) που μπορούν εύκολα να ενσωματωθούν χωρίς να χρειάζεται να γράψει ο προγραμματιστής CSS από την αρχή.

Για παράδειγμα, ένα alert μηνύματος επιτυχίας μπορεί να εμφανιστεί έτσι:

```
<div class="alert alert-success alert-dismissible fade show" role="alert">
  Το αίτημα καταχωρήθηκε με επιτυχία!
  <button type="button" class="btn-close" data-bs-dismiss="alert" aria-label="Close"></button>
</div>
```

Χάρη στο grid system του Bootstrap (container, row, col-md-*) η εφαρμογή προσαρμόζεται σε κάθε ανάλυση οθόνης: desktop, tablet, κινητό. Αυτό είναι ιδιαίτερα σημαντικό σε ένα σύστημα όπως το δικό μας που μπορεί να χρησιμοποιείται τόσο από σταθερούς υπολογιστές σε αίθουσες όσο και από κινητές συσκευές.

Παράδειγμα responsive διάταξης:

```
<div class="container">
  <div class="row">
```

```
    <div class="col-md-8">
      <h3>Τίτλος βιβλίου</h3>
    </div>
    <div class="col-md-4">
      <button class="btn btn-primary">Αίτηση Δανεισμού</button>
    </div>
  </div>
</div>
```

Όλα τα HTML αρχεία βρίσκονται στον φάκελο templates/ και επεκτείνουν το base.html, που περιέχει την κοινή δομή (π.χ. navbar, footer). Χάρη στο Jinja2 γίνεται δυνατή η εισαγωγή δυναμικού περιεχομένου, όπως λίστες βιβλίων, ονόματα χρηστών, μηνύματα ή στατιστικά.

```
{% extends 'base.html' %}
{% block title %}Πίνακας Διαχειριστή{% endblock %}
{% block content %}
  <h2>Καλώς ήρθατε, {{ session.user_name }}</h2>
{% endblock %}
```

Η χρήση HTML και Bootstrap επέτρεψε:

- την ταχεία κατασκευή διεπαφών χωρίς εξειδικευμένες γνώσεις γραφικών,
- την ευκολία στην πλοήγηση και την καλή εμπειρία χρήστη (UX),
- την προσαρμοστικότητα σε κάθε είδους συσκευή.

Το αποτέλεσμα είναι μια ευχάριστη, καθαρή και πρακτική πλατφόρμα για χρήση από φοιτητές και διαχειριστές.

Η σωστή παρουσίαση των δεδομένων σε ένα πληροφοριακό σύστημα είναι εξίσου σημαντική με τη διαχείριση τους. Για τον σκοπό αυτό, στην πλατφόρμα δανεισμού βιβλιοθήκης υιοθετήθηκε η βιβλιοθήκη Chart.js, ένα σύγχρονο και ευέλικτο εργαλείο για τη δημιουργία διαδραστικών γραφημάτων με τη βοήθεια JavaScript και HTML5. [17]

Το Chart.js είναι μια open-source βιβλιοθήκη JavaScript που επιτρέπει την απεικόνιση δεδομένων με εντυπωσιακά και responsive γραφήματα, όπως:

- Γραμμικά (line)
- Ραβδογράμματα (bar)
- Κυκλικά (pie, doughnut)
- Πολικά (radar, polarArea)
- Bubble και scatter charts

Η χρήση του είναι εξαιρετικά απλή και ενσωματώνεται απευθείας σε HTML σελίδες με πολύ λίγες γραμμές κώδικα.

Η πλατφόρμα περιλαμβάνει πίνακα διαχειριστή (Admin Panel) όπου ο admin μπορεί να δει στατιστικά όπως:

- Πόσες αιτήσεις έγιναν ανά μήνα
- Ποια βιβλία δανείζονται περισσότερο
- Πόσα βιβλία υπάρχουν ανά κατηγορία

Η χρήση του Chart.js δίνει στον διαχειριστή τη δυνατότητα να κατανοεί τη λειτουργία της βιβλιοθήκης οπτικά και γρήγορα, χωρίς να χρειάζεται να αναλύσει πίνακες δεδομένων.

Κεφάλαιο 3ο: Παρόμοια συστήματα διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης

3.1 Koha

Το Koha είναι ένα από τα πιο διαδεδομένα και ώριμα λογισμικά ανοικτού κώδικα για τη διαχείριση βιβλιοθηκών παγκοσμίως. [18] Αναπτύχθηκε για πρώτη φορά στη Νέα Ζηλανδία το 1999 από την εταιρεία Katipo Communications για τη Horowhenua Library Trust και έκτοτε έχει εξελιχθεί μέσω μιας ενεργής παγκόσμιας κοινότητας προγραμματιστών και χρηστών.

Το Koha καλύπτει πλήρως τις λειτουργικές ανάγκες μιας σύγχρονης βιβλιοθήκης, προσφέροντας πλήρες σύστημα ολοκληρωμένης διαχείρισης βιβλιοθήκης (Integrated Library System – ILS). Περιλαμβάνει λειτουργικότητες όπως:

- Καταλογογράφηση (cataloging): επιτρέπει την εισαγωγή βιβλίων και εγγραφών με μεταδεδομένα σε μορφή MARC21, προσφέροντας διεθνή συμβατότητα.
- Δανεισμοί (circulation): διαχειρίζεται την κίνηση των βιβλίων (δανεισμούς, επιστροφές, κρατήσεις, ανανεώσεις).
- Διαχείριση Χρηστών (patron management): κρατά πλήρες αρχείο χρηστών με δυνατότητες ειδοποιήσεων για καθυστερημένες επιστροφές ή ανανεώσεις.
- Αναζήτηση (OPAC – Online Public Access Catalog): φιλική διεπαφή για τους χρήστες ώστε να αναζητούν βιβλία και υλικό μέσω διαφόρων φίλτρων.
- Αναφορές (reports): παρέχει δυνατότητα εξαγωγής στατιστικών και προσαρμοσμένων αναφορών για τη διοίκηση της βιβλιοθήκης.

Ένα από τα ισχυρά σημεία του Koha είναι ότι υποστηρίζει πολλαπλές γλώσσες και διαφορετικές τοπικές παραμετροποιήσεις, γεγονός που το καθιστά ιδανικό για δημόσιες, δημοτικές, σχολικές και πανεπιστημιακές βιβλιοθήκες ανά τον κόσμο.

Το περιβάλλον διαχείρισης είναι πλήρως προσβάσιμο μέσω web browser, κάτι που επιτρέπει την απομακρυσμένη διαχείριση χωρίς την ανάγκη τοπικών εγκαταστάσεων σε κάθε σταθμό εργασίας. Επιπλέον, είναι συμβατό με RFID, barcode scanners και άλλες τεχνολογίες αυτοματοποίησης.

Η κοινότητα του Koha είναι εξαιρετικά ενεργή, με τακτικές ενημερώσεις και εκδόσεις και το λογισμικό διαθέτει εκτενή τεκμηρίωση και φόρουμ υποστήριξης. Επιχειρήσεις και οργανισμοί παρέχουν επίσης εμπορική υποστήριξη και hosting υπηρεσίες για βιβλιοθήκες που δεν διαθέτουν τεχνική υποδομή.

Το Koha είναι ιδανικό για οργανισμούς που επιθυμούν να αξιοποιήσουν ένα πλήρως παραμετροποιήσιμο, επεκτάσιμο και δωρεάν σύστημα διαχείρισης βιβλιοθήκης, χωρίς να εξαρτώνται από κλειστά και εμπορικά συστήματα.

3.2 Evergreen

Το Evergreen είναι ένα ανοικτού κώδικα ολοκληρωμένο σύστημα διαχείρισης βιβλιοθήκης (ILS) το οποίο αναπτύχθηκε αρχικά για τις ανάγκες του συνεταιρισμού δημοσίων βιβλιοθηκών της πολιτείας της Georgia (Georgia Public Library Service) στις Ηνωμένες Πολιτείες. Η πρώτη του έκδοση κυκλοφόρησε το 2006 και έκτοτε έχει εξελιχθεί σε ένα από τα πιο αξιόπιστα και ευέλικτα λογισμικά για βιβλιοθήκες μεγάλης κλίμακας [19]

Το Evergreen έχει σχεδιαστεί για να υποστηρίζει μεγάλα δίκτυα βιβλιοθηκών με εκατοντάδες ή και χιλιάδες καταχωρήσεις και χρήστες. Βασίζεται σε τεχνολογίες ανοικτού κώδικα όπως PostgreSQL και Perl, και έχει δομηθεί έτσι ώστε να είναι επεκτάσιμο, παραμετροποιήσιμο και ανθεκτικό.

Βασικά χαρακτηριστικά του περιλαμβάνουν:

- Καταλογογράφηση: Υποστήριξη για MARC και Z39.50 πρωτόκολλα, με εργαλεία για συγχώνευση και διόρθωση εγγραφών.
- Διαχείριση κυκλοφορίας: Επιτρέπει την πλήρη διαχείριση δανεισμών, επιστροφών, ανανεώσεων, κρατήσεων και ποινών (π.χ. για καθυστέρηση).
- Διαχείριση μελών: Υποστηρίζει δημιουργία και παρακολούθηση εγγραφών χρηστών, εκδόσεις καρτών μέλους, ρυθμίσεις προνομίων ανά κατηγορία χρήστη κ.ά.
- Διεπαφή OPAC: Προσφέρει ένα φιλικό προς τον χρήστη Online Public Access Catalog για την εύκολη αναζήτηση και περιήγηση σε καταχωρήσεις.
- Διαχείριση παραρτημάτων: Ιδανικό για περιβάλλοντα με πολλαπλές βιβλιοθήκες (π.χ. δίκτυα σχολείων ή δημοτικά παραρτήματα), με δυνατότητες κοινής χρήσης πόρων και διαχείρισης τοπικών πολιτικών.
- Ασφάλεια και παρακολούθηση: Περιλαμβάνει λειτουργίες audit, διαχείριση ρόλων και εξουσιοδοτήσεων, αλλά και παρακολούθηση συμβάντων.

Το Evergreen είναι ιδιαίτερα γνωστό για τη σταθερότητα και την κλιμάκωσή του, καθώς έχει σχεδιαστεί να υποστηρίζει τεράστιες βάσεις δεδομένων με υψηλή απόδοση. Παράλληλα η κοινότητα ανάπτυξής του είναι ενεργή και ενθαρρύνει τη συνεισφορά από φορείς και άτομα.

Ένα ακόμα πλεονέκτημά του είναι η δυνατότητα ενσωμάτωσης με τρίτα συστήματα, όπως web portals, συστήματα RFID, ERM (electronic resource management) και άλλες λύσεις μέσω API.

Το Evergreen προτείνεται συχνά για μεγάλα δίκτυα βιβλιοθηκών, δημόσιες ή πανεπιστημιακές, που επιθυμούν σταθερότητα, επεκτασιμότητα και έλεγχο στον πηγαίο κώδικα και τη δομή της πλατφόρμας τους.

3.3 OPALS

Το OPALS είναι ένα πλήρως εξοπλισμένο, ανοικτού κώδικα σύστημα διαχείρισης βιβλιοθηκών που αναπτύχθηκε αρχικά για τις ανάγκες σχολικών βιβλιοθηκών, αλλά σύντομα υιοθετήθηκε και από άλλα ιδρύματα όπως θρησκευτικές βιβλιοθήκες, δημόσιες και ιδιωτικές συλλογές ακόμη και ειδικές βιβλιοθήκες (special libraries). Το OPALS είναι διαθέσιμο ως web-based λύση και παρέχεται τόσο ως αυτο-φιλοξενούμενο (self-hosted) όσο και ως φιλοξενούμενη υπηρεσία (cloud-hosted) [20]

Βασίζεται σε τεχνολογίες ανοικτού κώδικα και στοχεύει να προσφέρει μια ολοκληρωμένη λύση με ελάχιστο κόστος για μικρού και μεσαίου μεγέθους βιβλιοθήκες που δεν διαθέτουν μεγάλο τεχνικό προσωπικό.

Κύρια χαρακτηριστικά του OPALS:

- Κατάλογος και αναζήτηση (OPAC): Διαθέτει σύγχρονο και εύχρηστο περιβάλλον για την αναζήτηση τίτλων, με υποστήριξη MARC, πολυγλωσσικών μεταδεδομένων και φιλτραρίσματος.
- Διαχείριση αποθεμάτων: Παρέχει λειτουργίες για προσθήκη νέων τίτλων, τροποποιήσεις, διαγραφή, barcode tracking και οργάνωση σε κατηγορίες.
- Κυκλοφορία: Διαχειρίζεται δανεισμούς, επιστροφές, ανανεώσεις και κρατήσεις χρηστών, με ρυθμίσεις για ποινές και υπενθυμίσεις.
- Διαχείριση χρηστών: Επιτρέπει καταχώρηση μελών, προβολή ιστορικού, προσθήκη προνομίων ανά τύπο χρήστη και ενσωμάτωση με συστήματα SSO (single sign-on) σε εκπαιδευτικά περιβάλλοντα.
- Διαχείριση περιοδικών και πολυμέσων: Εκτός από βιβλία, υποστηρίζει και αρχεία ήχου/βίντεο, περιοδικά, εφημερίδες και άλλους τύπους υλικού.
- Αναφορές και στατιστικά: Περιλαμβάνει έτοιμες αναφορές και δυνατότητα δημιουργίας εξατομικευμένων στατιστικών για τη χρήση της βιβλιοθήκης, τη δραστηριότητα των χρηστών και τις κινήσεις τίτλων.
- Υποστήριξη για εκπαίδευση: Διαθέτει εργαλεία για βιβλιοθήκες σε σχολεία, όπως προτάσεις για projects, εκπαιδευτικά tags και ενσωμάτωση με e-learning περιβάλλοντα.

Όπως φαίνεται, το OPALS επικεντρώνεται στην προσβασιμότητα, την ευκολία χρήσης και τη λειτουργικότητα. Είναι ιδιαίτερα φιλικό για μη τεχνικούς χρήστες, ενώ η διεπαφή του είναι σχεδιασμένη ώστε να μπορούν να την διαχειρίζονται εύκολα και μαθητές ή δάσκαλοι.

Ένα ακόμα πλεονέκτημα είναι η ενεργή κοινότητα και οι οργανισμοί που προσφέρουν υποστήριξη, είτε εμπορικά είτε εθελοντικά, κάνοντάς το ιδανική επιλογή για εκπαιδευτικά ιδρύματα που θέλουν μια σταθερή και δοκιμασμένη λύση χωρίς να δεσμεύονται σε εμπορικές άδειες.

3.4 OpenBiblio

Το OpenBiblio είναι ένα ελαφρύ, ανοικτού κώδικα σύστημα διαχείρισης βιβλιοθήκης που δημιουργήθηκε με σκοπό να εξυπηρετήσει μικρού και μεσαίου μεγέθους βιβλιοθήκες όπως σχολικές, κοινοτικές ή εξειδικευμένες συλλογές. Βασισμένο σε PHP και MySQL, το OpenBiblio είναι γνωστό για την απλότητα στην εγκατάσταση, τη χαμηλή απαίτηση σε πόρους [21]

Το σύστημα διαθέτει βασικές λειτουργίες διαχείρισης καταλόγου, κυκλοφορίας και χρηστών. Μέσω της ενότητας καταλόγου, ο διαχειριστής μπορεί να καταχωρεί και να ενημερώνει βιβλία με όλα τα απαραίτητα πεδία, όπως τίτλος, συγγραφέας, ISBN, θέμα και εκδότης. Ακόμη έχει τη δυνατότητα αναζήτησης με λέξεις-κλειδιά ή συνδυασμούς κριτηρίων. Η κυκλοφορία βιβλίων πραγματοποιείται μέσω απλών φορμών που υποστηρίζουν δανεισμό, επιστροφή και ανανέωση υλικού. Το σύστημα καταγράφει το ιστορικό κινήσεων και υποστηρίζει περιορισμένη διαχείριση ποινών και καθυστερήσεων.

Το OpenBiblio επιτρέπει επίσης τη διαχείριση χρηστών και μελών με καταγραφή στοιχείων και δυνατότητα ελέγχου των δανεισμών ανά χρήστη. Η διεπαφή χρήστη είναι λιτή και καθαρή, διευκολύνοντας την εκμάθηση του συστήματος, ενώ η πρόσβαση στις βασικές λειτουργίες είναι άμεση.

Ένα σημαντικό πλεονέκτημα του OpenBiblio είναι η ευκολία προσαρμογής του από τον διαχειριστή ή έναν προγραμματιστή. Παρόλο που δεν υποστηρίζει εξελιγμένες λειτουργίες όπως στατιστικά χρήσης, διαχείριση περιοδικών ή ενσωμάτωση με εξωτερικές πλατφόρμες και αποτελεί μια εξαιρετική λύση για περιπτώσεις όπου ζητείται ένα βασικό αλλά λειτουργικό σύστημα. Είναι ιδανικό για σχολικές βιβλιοθήκες ή συλλογές που ξεκινούν τώρα την οργάνωση και μηχανογράφηση του υλικού τους.

3.5 Το δικό μας σύστημα – Πλεονεκτήματα και Εκπαιδευτική Χρήση σε σχέση με άλλα

Τα υπάρχοντα συστήματα διαχείρισης βιβλιοθηκών, όπως το Koha, το Evergreen, το OpenBiblio και το PMB, αποτελούν καλές λύσεις που έχουν αναπτυχθεί εδώ και χρόνια και έχουν πλούσια χαρακτηριστικά, επεκτασιμότητα και υποστήριξη μεγάλων οργανισμών. Ωστόσο αυτά τα συστήματα

χαρακτηρίζονται από σύνθετη αρχιτεκτονική και συχνά απαιτούν γνώση τεχνολογιών όπως Apache, Perl, MARC21, LDAP, και RESTful APIs. Αν και είναι πανίσχυρα δεν είναι απαραίτητα εύχρηστα ή κατάλληλα για μικρότερες εφαρμογές ή για εκπαιδευτική αξιοποίηση μέσα σε ένα τμήμα.

Ακριβώς αυτό το κενό έρχεται να καλύψει η πλατφόρμα που αναπτύξαμε στο πλαίσιο αυτής της εργασίας. Πρόκειται για ένα ανοιχτό και πλήρως κατανοητό σύστημα βασισμένο σε τεχνολογίες που διδάσκονται στα περισσότερα προγράμματα σπουδών Πληροφορικής όπως η Python (Flask framework), η MySQL και η HTML/Bootstrap για το frontend. Το σύστημα είναι σχεδιασμένο να λειτουργεί με απλό τρόπο και ταυτόχρονα υποστηρίζει όλες τις βασικές ανάγκες μιας ακαδημαϊκής ή σχολικής βιβλιοθήκης: καταχώρηση βιβλίων, διαχείριση κατηγοριών, αιτήσεις δανεισμού, διαχείριση χρηστών, πίνακες ελέγχου για admin και φοιτητές, ειδοποιήσεις και εξαγωγή στατιστικών.

Ένα από τα μεγαλύτερα πλεονεκτήματα του συστήματος είναι πως μπορεί να χρησιμοποιηθεί ως εργαλείο εκπαίδευσης. Ο φοιτητής ή ο εκπαιδευόμενος μπορεί να διαβάσει, να καταλάβει και να επεξεργαστεί τον κώδικα. Μπορεί να προσθέσει νέες λειτουργίες, να τροποποιήσει τις υπάρχουσες, να ενσωματώσει νέα εργαλεία ή APIs, να αλλάξει τον τρόπο εμφάνισης των δεδομένων ή να δημιουργήσει νέες οθόνες και flows. Αυτό είναι αδύνατο ή εξαιρετικά δύσκολο σε ένα σύστημα όπως το Koha ή το Evergreen, λόγω της πολυπλοκότητάς τους και της απαίτησης για τεχνική γνώση.

Ακόμη η προσαρμοστικότητα του συστήματος το κάνει ιδανικό για ένα συγκεκριμένο τμήμα ή μονάδα. Αντί να χρησιμοποιείται ένα γενικό εργαλείο που δεν γνωρίζει τη λειτουργία της βιβλιοθήκης μας το παρόν σύστημα έχει δημιουργηθεί ειδικά για τις ανάγκες ενός χώρου που διαθέτει περιορισμένα βιβλία και απαιτεί γρήγορη, εύκολη και λειτουργική διαχείριση. Περιλαμβάνει λειτουργίες όπως ειδοποιήσεις, dashboard ανά ρόλο, ακύρωση αιτήσεων, στατιστικά και μπορεί εύκολα να επεκταθεί με ειδοποιήσεις μέσω email, barcode σάρωση ή ακόμη και τεχνητή νοημοσύνη για προτάσεις βιβλίων.

Κεφάλαιο 4ο: Το σύστημα διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης

4.1 Μια εισαγωγή

Η παρούσα εργασία περιγράφει την υλοποίηση ενός ολοκληρωμένου συστήματος διαχείρισης βιβλιοθήκης για τις ανάγκες ενός Τμήματος Ανώτατου Εκπαιδευτικού Ιδρύματος. Το σύστημα έχει σχεδιαστεί ώστε να υποστηρίζει τη λειτουργία μιας φυσικής αίθουσας δανεισμού και μας δίνει ταυτόχρονα ένα εύχρηστο ψηφιακό περιβάλλον για φοιτητές και διαχειριστές.

Η ανάγκη για ένα τέτοιο σύστημα προκύπτει από τη δυσκολία παρακολούθησης και διαχείρισης φυσικών βιβλίων χωρίς την υποστήριξη κατάλληλου λογισμικού. Πρακτικά απαιτείται μια πλατφόρμα μέσω της οποίας οι φοιτητές μπορούν να βλέπουν τα διαθέσιμα βιβλία, να κάνουν αιτήματα δανεισμού και να ενημερώνονται για επιστροφές ή τυχόν εκκρεμότητες. Αντίστοιχα οι διαχειριστές (βιβλιοθηκάριοι ή μέλη ΔΕΠ) χρειάζονται ένα απλό interface για να καταχωρούν νέα βιβλία, να εγκρίνουν ή να απορρίπτουν αιτήσεις και να έχουν πλήρη εικόνα για το ιστορικό χρήσης του κάθε φοιτητή.

Το σύστημα αναπτύχθηκε με χρήση της Python και του framework Flask, το οποίο επιλέχθηκε για την ευελιξία, τη γρήγορη ανάπτυξη και την απλότητα υλοποίησης RESTful routes. Η βάση δεδομένων είναι υλοποιημένη με MySQL, με έμφαση στην αποθήκευση των σχέσεων μεταξύ χρηστών, βιβλίων, αιτημάτων και δανεισμών. Δεν χρησιμοποιούνται ORM frameworks (όπως SQLAlchemy) αλλά γίνεται άμεση χρήση SQL queries μέσω της βιβλιοθήκης mysql.connector για μεγαλύτερο έλεγχο και απλότητα.

Το frontend της εφαρμογής είναι υλοποιημένο με Bootstrap, προσφέροντας responsive σχεδιασμό ώστε να είναι πλήρως λειτουργικό τόσο σε επιτραπέζιους υπολογιστές όσο και σε φορητές συσκευές. Έχουν ενσωματωθεί επίσης βασικά στοιχεία UI/UX, όπως ειδοποιήσεις (notifications), σύστημα επιβεβαιώσεων (π.χ. κατά τη διαγραφή), καθώς και ταξινόμηση δεδομένων μέσω DataTables.

Η λογική του συστήματος βασίζεται σε ρόλους: φοιτητές και διαχειριστές. Οι φοιτητές μπορούν να κάνουν login και να αιτηθούν τον δανεισμό ενός βιβλίου, να ακυρώσουν το αίτημα αν δεν έχει εγκριθεί ακόμη και να δουν το ιστορικό τους. Οι διαχειριστές έχουν πρόσβαση σε όλες τις λειτουργίες, όπως:

- Διαχείριση καταλόγου βιβλίων και κατηγοριών.
- Προβολή και διαχείριση αιτημάτων.
- Έγκριση/απόρριψη αιτημάτων.

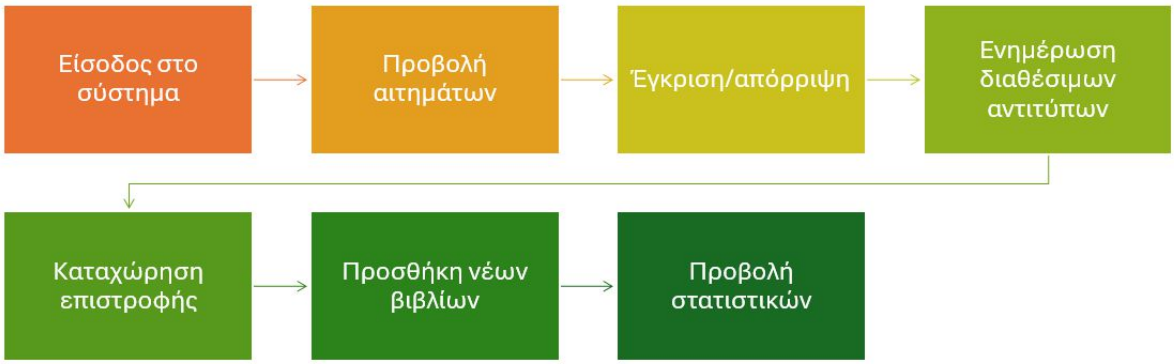
- Καταχώρηση επιστροφών και αυτόματη ενημέρωση διαθεσιμότητας αντιτύπων.
- Προβολή ιστορικού ανά φοιτητή.
- Στατιστικά χρήσης και δυνατότητα εξαγωγής σε CSV.

Στην υλοποίηση έχει υλποποιηθεί η διασύνδεση των αιτημάτων με τους πραγματικούς δανεισμούς μέσω μοναδικών IDs, ώστε να καταγράφεται ποιος χρήστης, πότε και ποιο βιβλίο έλαβε ή επέστρεψε. Εχει προβλεφθεί η χρήση ειδοποιήσεων ανά χρήστη για σημαντικές ενέργειες όπως η έγκριση ή η επιστροφή ενός βιβλίου.

Το σύστημα έχει ως στόχο να απλοποιήσει τη διαδικασία διαχείρισης της βιβλιοθήκης, να μειώσει την ανάγκη για φυσική γραφειοκρατία και να προσφέρει μια λειτουργική λύση.

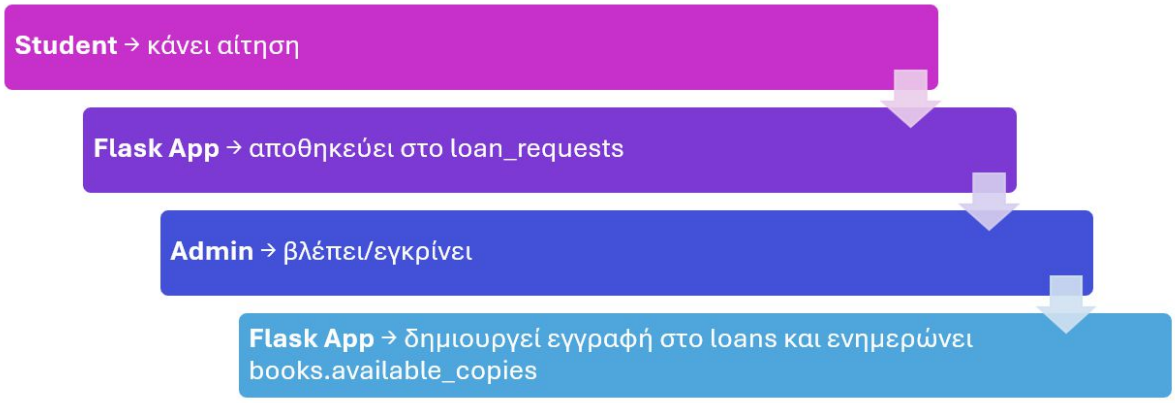
Εικόνα 4.1: Διάγραμμα Ροής Χρήστη

Το συγκεκριμένο διάγραμμα 3.1 απεικονίζει την αλληλουχία ενεργειών που εκτελεί ένας τυπικός χρήστης του συστήματος (φοιτητής) προκειμένου να αιτηθεί και να διαχειριστεί τον δανεισμό ενός βιβλίου. Η ροή ξεκινά από την αναζήτηση και προβολή των διαθέσιμων βιβλίων συνεχίζεται με την υποβολή αίτησης δανεισμού και καταλήγει στην ενημέρωσή του για την κατάσταση της αίτησής του. Εφόσον η αίτηση εγκριθεί, ο χρήστης μπορεί να επιστρέψει το βιβλίο, διαδικασία που καταχωρείται από τον διαχειριστή. Το διάγραμμα απεικονίζει καθαρά τη λογική ροή των λειτουργιών και τις ενδιάμεσες καταστάσεις.



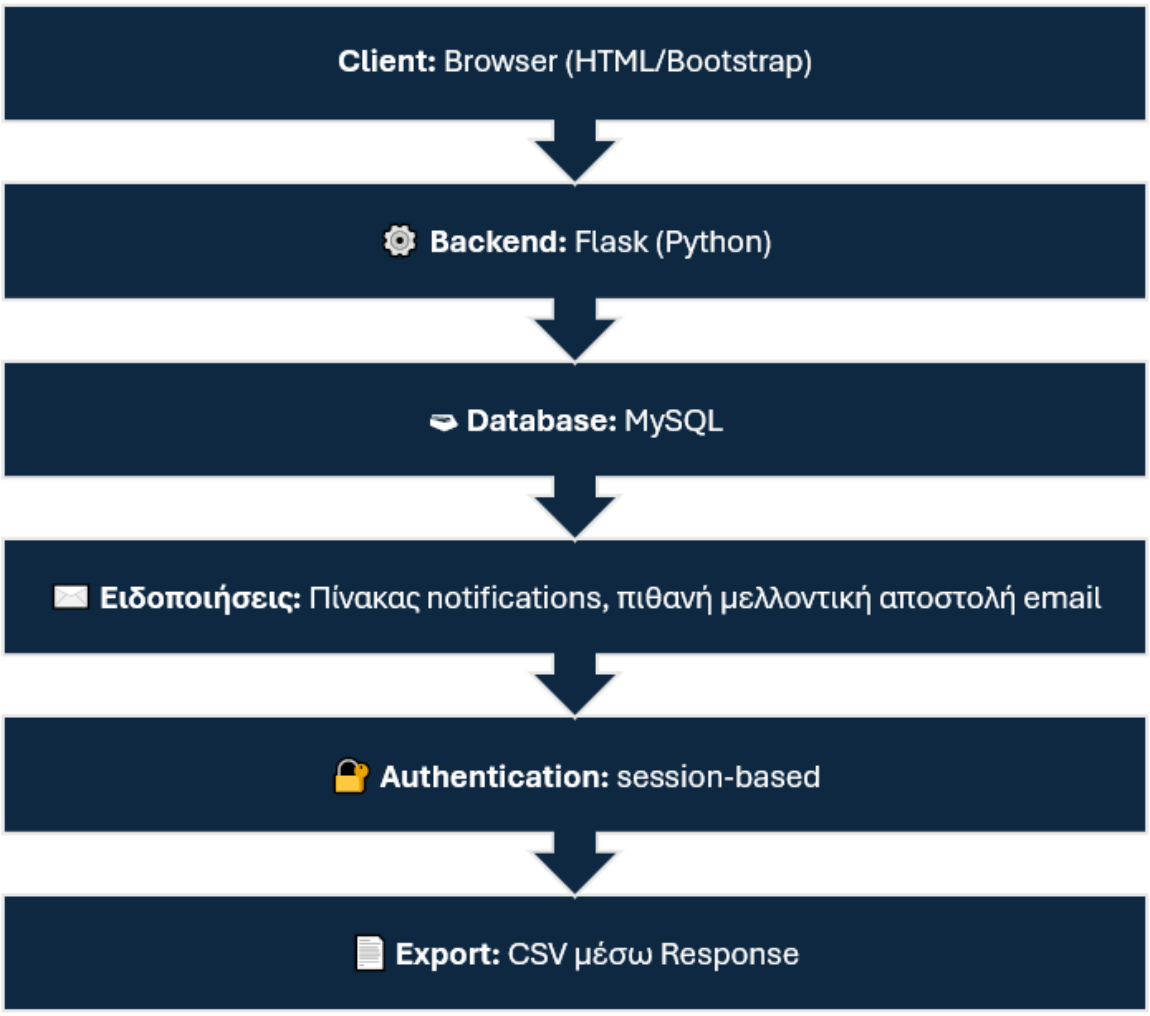
Εικόνα 4.2: Διάγραμμα Δραστηριότητας Διαχειριστή

Αυτό το διάγραμμα 3.2 αποτυπώνει τα βήματα που ακολουθεί ο διαχειριστής (admin) του συστήματος για να διαχειριστεί τις αιτήσεις δανεισμού και τη βιβλιοθήκη συνολικά. Περιλαμβάνει τις διαδικασίες έγκρισης ή απόρριψης αιτήσεων, καταχώρησης επιστροφών, προσθήκης ή διαγραφής βιβλίων, δημιουργίας κατηγοριών και προβολής στατιστικών χρήσης. Το διάγραμμα παρέχει συνολική εικόνα των υπευθυνοτήτων του ρόλου του διαχειριστή και υπογραμμίζει τη σημασία του στην ομαλή λειτουργία της πλατφόρμας.



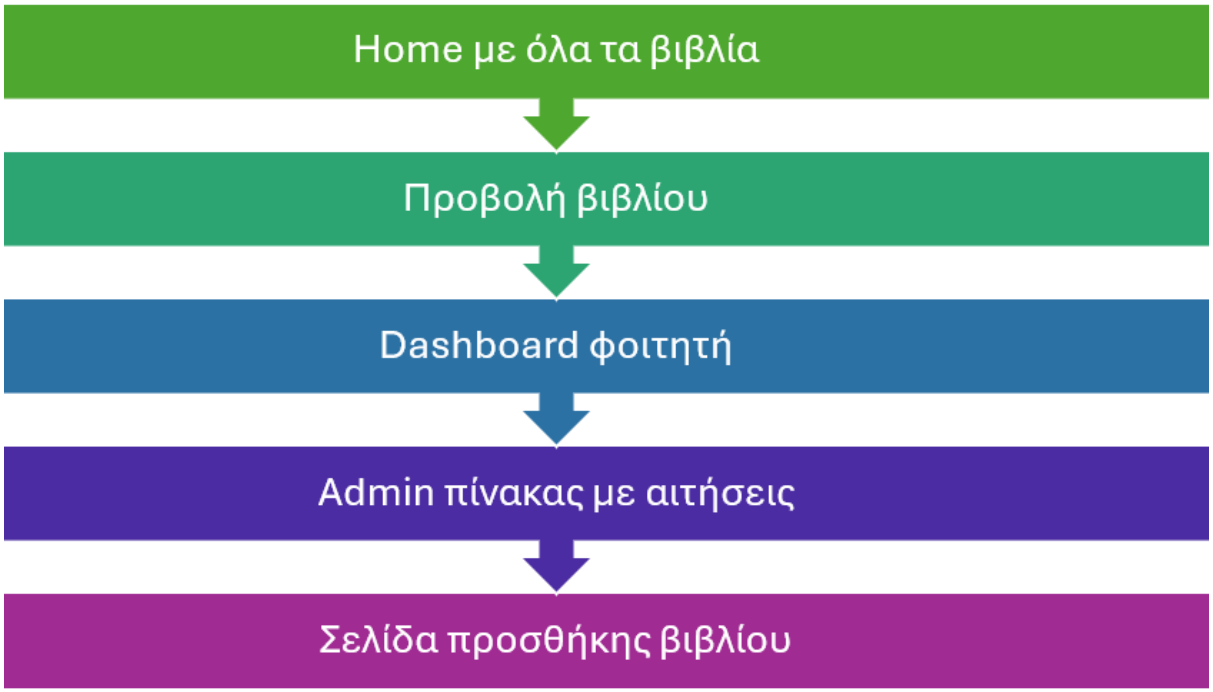
Εικόνα 4.3: Διάγραμμα Ακολουθίας

Το διάγραμμα ακολουθίας (sequence diagram) απεικονίζει τον χρονικά καθορισμένο τρόπο με τον οποίο τα επιμέρους υποσυστήματα και οι ρόλοι του συστήματος αλληλεπιδρούν για την υλοποίηση της διαδικασίας αιτήματος δανεισμού. Περιγράφει την επικοινωνία μεταξύ του φοιτητή, της web διεπαφής, του backend server (Flask) και της βάσης δεδομένων. Η διαδικασία ξεκινά με την υποβολή της αίτησης, συνεχίζεται με την αποθήκευσή της, την έγκριση από τον διαχειριστή, την δημιουργία εγγραφής δανεισμού και την ειδοποίηση του χρήστη. Το διάγραμμα επιτρέπει την κατανόηση της εσωτερικής ροής των δεδομένων και της επιχειρησιακής λογικής.



Εικόνα 4.4: Διάγραμμα Συστημάτων

Το διάγραμμα αυτό παρουσιάζει την αρχιτεκτονική του συστήματος, αποτυπώνοντας τη ροή δεδομένων και τις τεχνολογίες που εμπλέκονται. Η αρχιτεκτονική ακολουθεί το μοντέλο client-server, με το frontend (HTML/Bootstrap) να επικοινωνεί με τον Flask server μέσω HTTP, ενώ η βάση δεδομένων MySQL υποστηρίζει την αποθήκευση όλων των σχετικών πληροφοριών. Οι ειδοποιήσεις καταχωρούνται στη βάση, ενώ υπάρχει δυνατότητα εξαγωγής δεδομένων σε αρχεία CSV. Η σαφής δομή του διαγράμματος διευκολύνει την τεκμηρίωση της τεχνολογικής στοίβας και της οργάνωσης του συστήματος.



Εικόνα 4.5: Διάγραμμα Διεπαφής (UI Wireframe)

Το διάγραμμα διεπαφής παρουσιάζει τα βασικά γραφικά μέρη του συστήματος όπως τα αντιλαμβάνεται ο χρήστης. Περιλαμβάνει την αρχική σελίδα με την παρουσίαση όλων των βιβλίων, την προβολή λεπτομερειών για κάθε βιβλίο, τη σελίδα υποβολής αίτησης, το dashboard του φοιτητή με τις ενεργές και παρελθούσες αιτήσεις και το διαχειριστικό πάνελ με τις ενέργειες έγκρισης, προσθήκης και στατιστικών. Το UI wireframe καθιστά δυνατή τη γρήγορη κατανόηση της διάταξης και της λειτουργικότητας του συστήματος από την πλευρά του τελικού χρήστη.

4.2 Εξήγηση της βάσης

Η βάση δεδομένων αποτελεί τον πυρήνα της διαδικτυακής πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης. Όλα τα δεδομένα —χρήστες, βιβλία, αιτήσεις, δανεισμοί, κατηγορίες, στατιστικά— αποθηκεύονται και οργανώνονται με τρόπο που επιτρέπει εύκολη ανάκτηση, τροποποίηση και συσχέτιση μεταξύ τους. Παρακάτω παρουσιάζεται αναλυτικά η δομή της βάσης, τα βασικά σχήματα που χρησιμοποιούνται και η λογική σχεδίασής της.

Η βάση δεδομένων υλοποιήθηκε με το MySQL/MariaDB, λόγω της αξιοπιστίας της, της ευρείας υποστήριξης από την κοινότητα και της δυνατότητας εύκολης διασύνδεσης με εφαρμογές Flask μέσω Python. Τα δεδομένα είναι αποθηκευμένα με χαρακτήρες UTF-8 ώστε να υποστηρίζονται πλήρως τα ελληνικά.

Περιγραφή Πινάκων

Πίνακας users

Αποθηκεύει τις πληροφορίες των χρηστών του συστήματος. Υπάρχουν δύο ρόλοι χρηστών:

- student (φοιτητές)
- admin (διαχειριστής)

Κύρια πεδία:

- id: πρωτεύον κλειδί
- name, email, password
- role: καθορίζει τα δικαιώματα
- created_at: ημερομηνία εγγραφής

Πίνακας books

Περιέχει όλα τα βιβλία της βιβλιοθήκης.

Πεδία:

- title, author, isbn, description
- category_id: συσχετισμός με τον πίνακα categories
- total_copies, available_copies: υποστήριξη πολλαπλών αντιτύπων
- image_url: σύνδεσμος για την εικόνα του εξωφύλλου

Πίνακας categories

Κατηγοριοποιεί τα βιβλία με βάση το γνωστικό τους αντικείμενο. Παρέχει δυνατότητα φιλτραρίσματος.

Πίνακας loan_requests

Είναι το πρώτο βήμα της διαδικασίας δανεισμού. Όταν ένας φοιτητής κάνει αίτηση για βιβλίο, δημιουργείται εγγραφή σε αυτόν τον πίνακα με status = 'pending'.

Πεδία:

- user_id, book_id

- status: pending, approved, rejected
- admin_comment: πιθανή παρατήρηση

Πίνακας loans

Περιέχει μόνο τις αιτήσεις που έχουν εγκριθεί. Ο δανεισμός συνδέεται με αίτηση μέσω request_id.

Πεδία:

- loan_date, return_date, actual_return
- status: borrowed, returned, late

Πίνακας notifications

Καταγράφει ειδοποιήσεις για γεγονότα όπως:

- έγκριση/απόρριψη αίτησης
- επιστροφή βιβλίου
- καθυστερημένος δανεισμός

Οι ειδοποιήσεις εμφανίζονται στο dashboard του φοιτητή.

Πίνακας stats

Παρακολουθεί στατιστικά δανεισμού ανά βιβλίο. Μπορεί να χρησιμοποιηθεί για προβολή των πιο δημοφιλών τίτλων.

Το σύστημα έχει τις εξής βασικές σχέσεις:

- Ένας user μπορεί να έχει πολλές loan_requests και loans.
- Κάθε book ανήκει σε μια category.
- Κάθε loan συνδέεται με ένα loan_request μέσω request_id.
- Οι notifications είναι συσχετισμένες με χρήστες μέσω user_id.

Η λογική της βάσης εξασφαλίζει ότι:

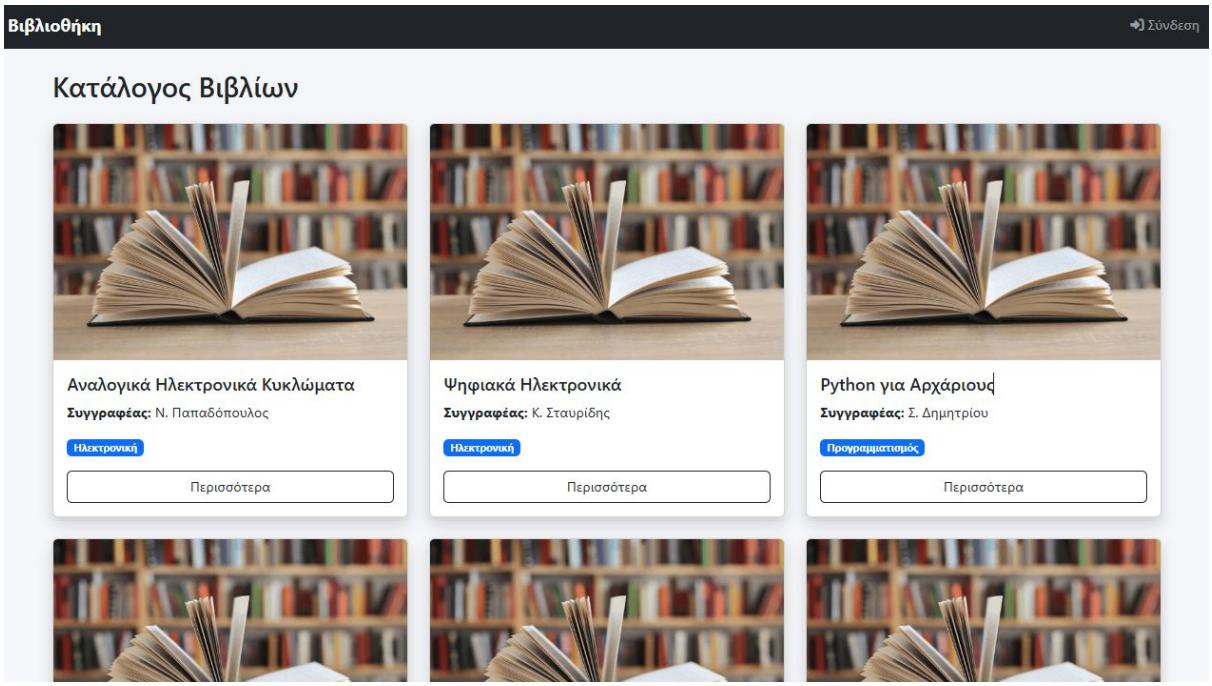
- Δεν επιτρέπεται διπλή αίτηση για το ίδιο βιβλίο από τον ίδιο φοιτητή εάν υπάρχει ενεργή.

- Η διαχείριση αντιτύπων γίνεται αυτόματα κατά την έγκριση/επιστροφή.
- Οι αιτήσεις δεν μετατρέπονται άμεσα σε δανεισμούς —αυτό γίνεται μόνο με έγκριση διαχειριστή.

Η παρούσα σχεδίαση είναι απλή αλλά επεκτάσιμη:

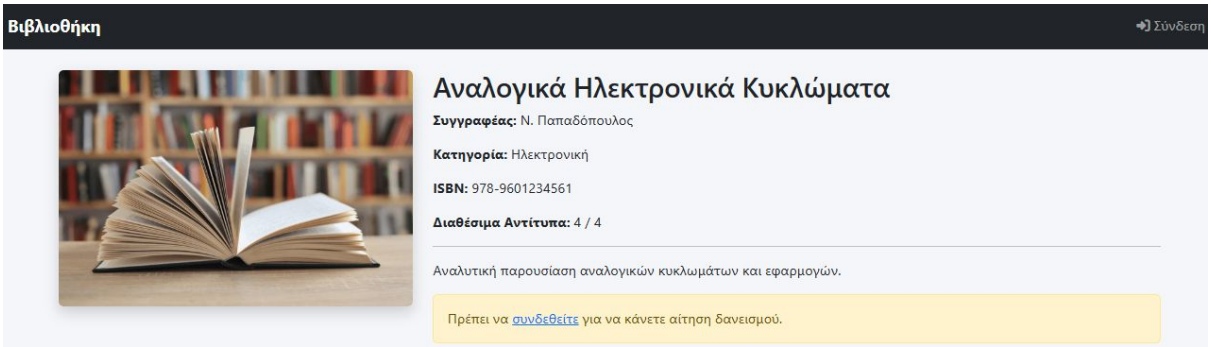
- Μπορεί να προστεθεί υποστήριξη για καθυστερημένα πρόστιμα.
- Μπορεί να ενσωματωθεί σύστημα ειδοποίησης με email ή SMS.
- Οι δανεισμοί και τα στατιστικά μπορούν να προβληθούν με οπτικοποιήσεις (π.χ. Chart.js).

4.3 Η Πλατφόρμα



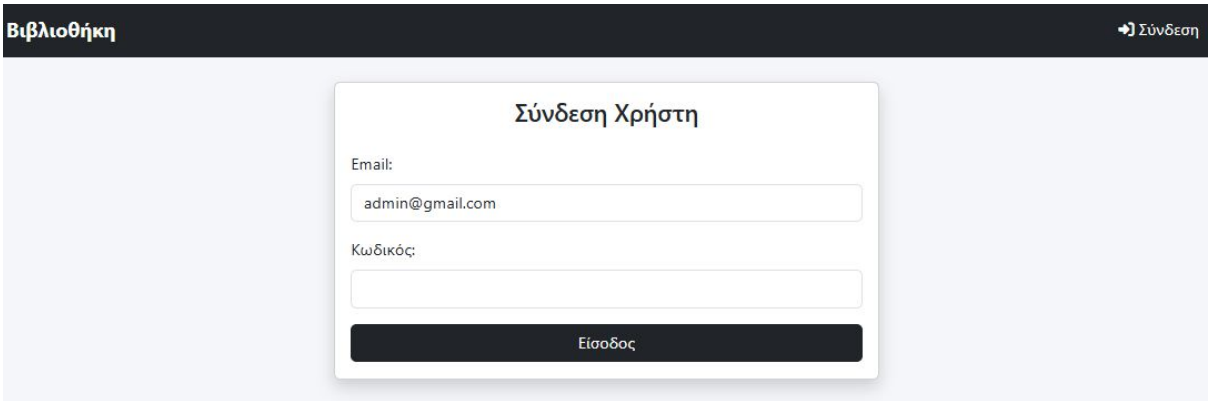
Εικόνα 4.6: Αρχική Dashboard του Admin

Στην Εικόνα 3.6 παρουσιάζεται η αρχική σελίδα του διαχειριστή μετά από επιτυχή σύνδεση στο σύστημα. Μέσω του μενού πλοήγησης στο επάνω μέρος, ο χρήστης μπορεί να μεταβεί σε επιμέρους περιοχές διαχείρισης όπως "Βιβλία", "Κατηγορίες", "Αιτήσεις" και "Στατιστικά". Το σύστημα παρέχει άμεση πρόσβαση στις βασικές λειτουργίες με τρόπο λειτουργικό ώστε ο διαχειριστής να εκτελεί γρήγορα τις βασικές ενέργειες χωρίς περιττές μετακινήσεις.



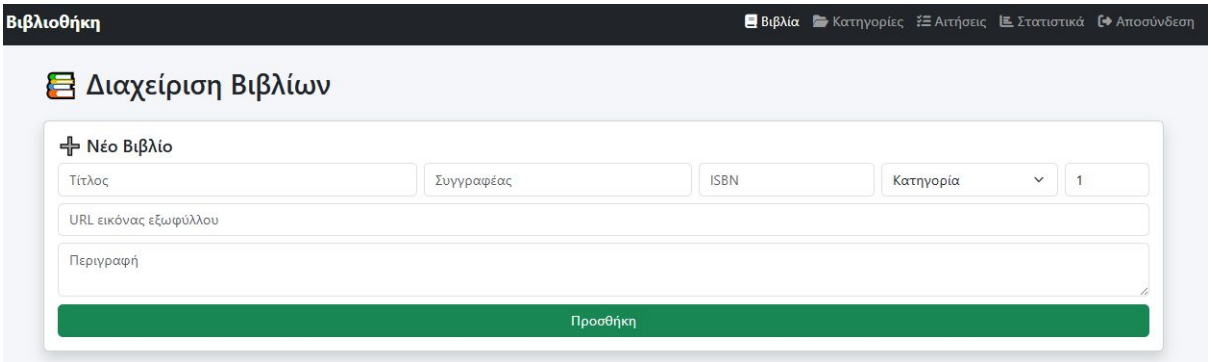
Εικόνα 4.7: Προβολή βιβλίου για έναν που συνδέθηκε

Όπως φαίνεται στην Εικόνα 3.7, όταν ένας χρήστης επιλέξει ένα βιβλίο, εμφανίζεται αναλυτική προβολή με τίτλο, συγγραφέα, κατηγορία, ISBN, περιγραφή και πλήθος διαθέσιμων αντιτύπων. Αν ο χρήστης δεν είναι συνδεδεμένος του εμφανίζεται ενημερωτικό μήνυμα με παραπομπή για σύνδεση για να μπορεί να προχωρήσει σε αίτηση δανεισμού. Το περιβάλλον είναι καθαρό, φιλικό από τον τελικό χρήστη.



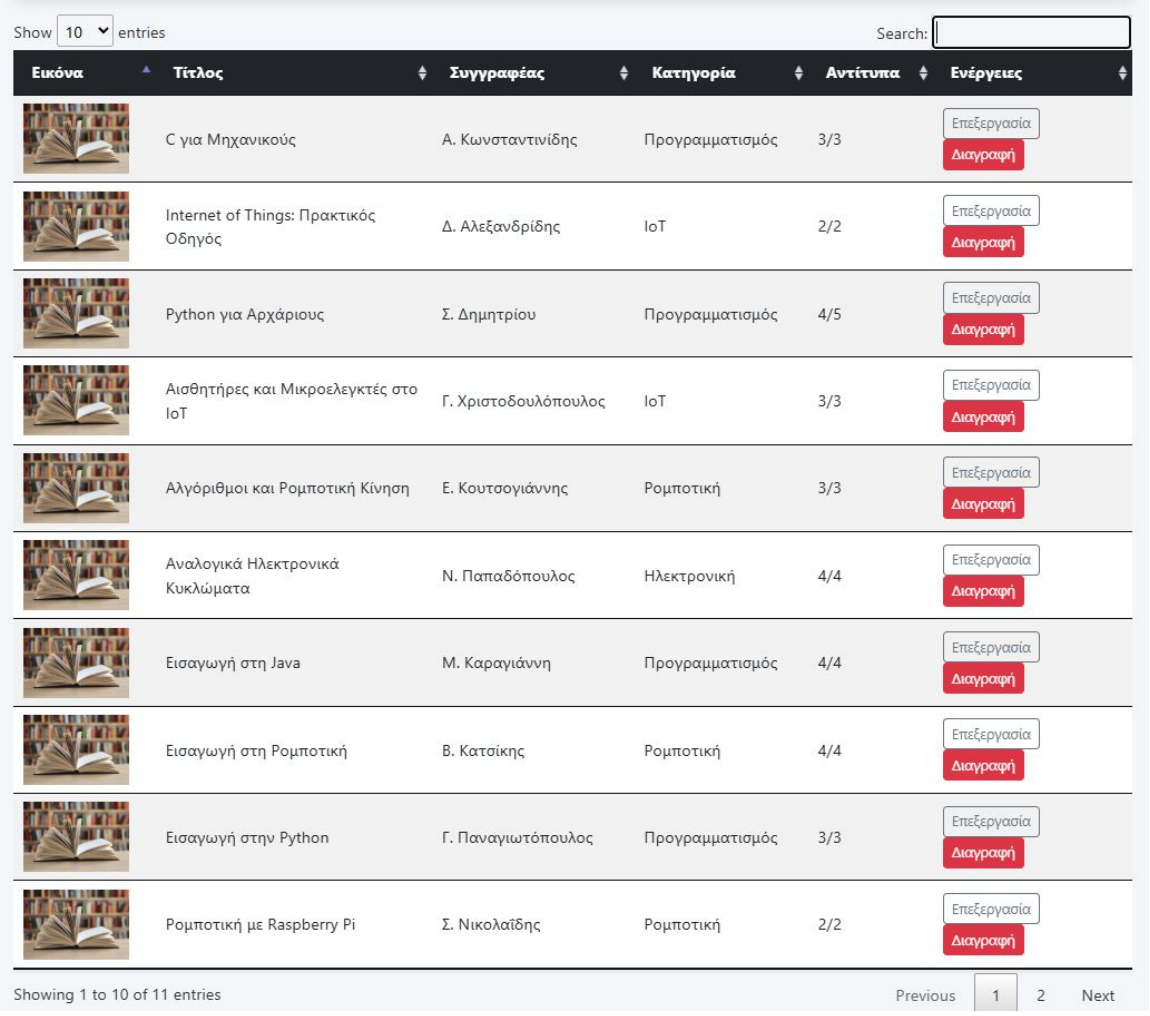
Εικόνα 4.8: Σελίδα σύνδεσης

Στην Εικόνα 3.8 απεικονίζεται η σελίδα σύνδεσης χρήστη, μέσω της οποίας γίνεται η ταυτοποίηση είτε φοιτητών είτε διαχειριστών. Η φόρμα είναι απλή και λειτουργική, με απαιτούμενα πεδία για email και κωδικό πρόσβασης. Μετά τη σύνδεση, ο χρήστης οδηγείται στο αντίστοιχο dashboard ανάλογα με τον ρόλο του.



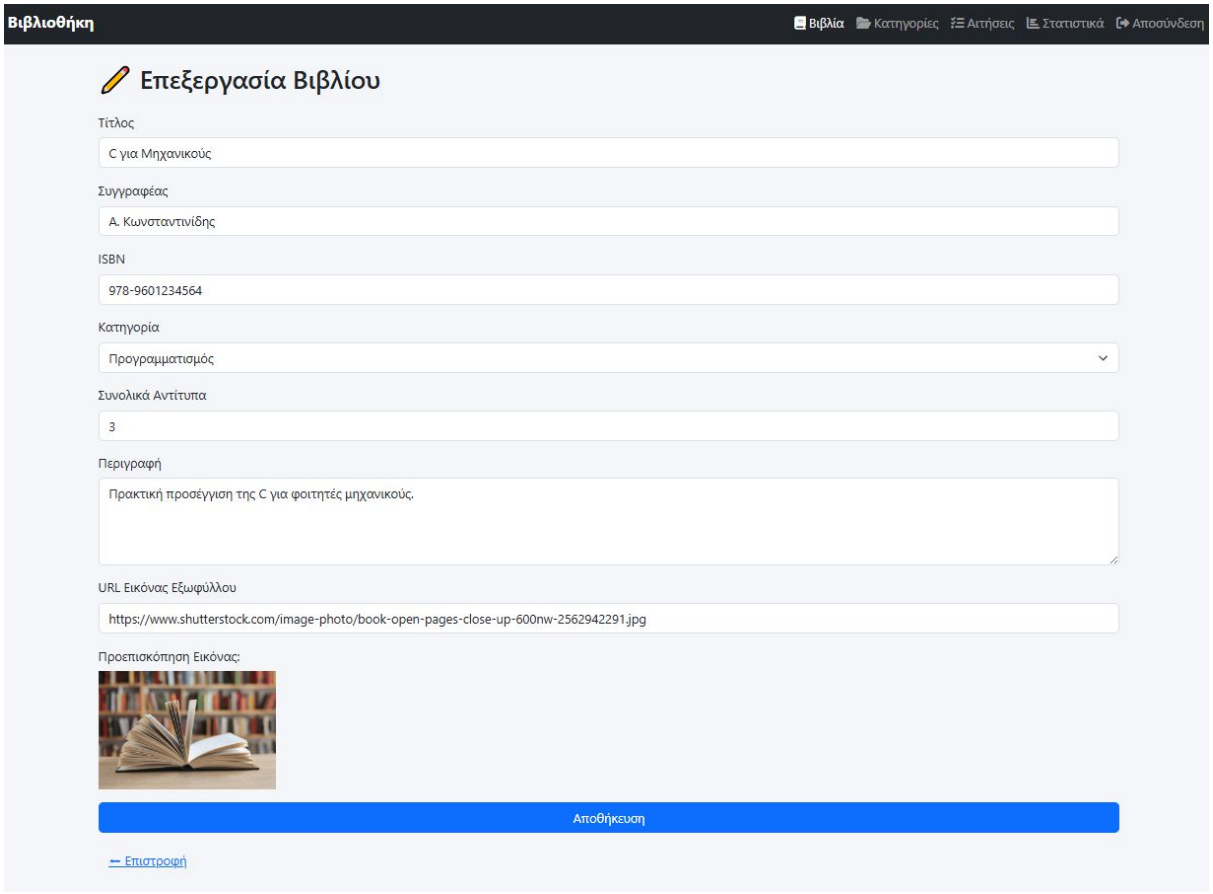
Εικόνα 4.9: Προσθήκη νέου βιβλίου στη Διαχείριση βιβλίων – admin

Η Εικόνα 3.9 παρουσιάζει τη φόρμα προσθήκης νέου βιβλίου η οποία είναι διαθέσιμη αποκλειστικά στους διαχειριστές. Περιλαμβάνει όλα τα απαραίτητα πεδία, όπως τίτλος, συγγραφέας, ISBN, κατηγορία, περιγραφή, πλήθος αντιτύπων και σύνδεσμος για εικόνα εξωφύλλου. Η σχεδίαση διευκολύνει την άμεση καταχώρηση βιβλίων με ελάχιστα κλικ.



Εικόνα 4.10: Διαχείριση βιβλίων – admin

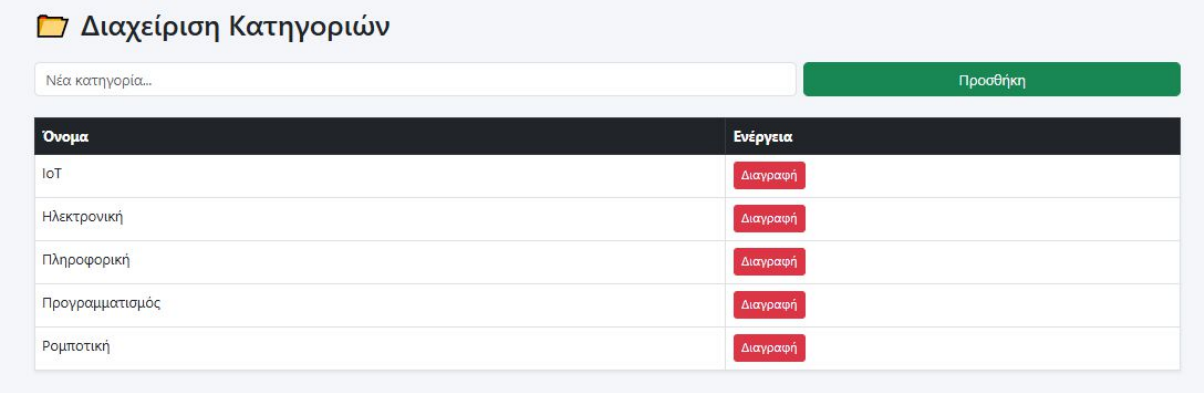
Όπως απεικονίζεται στην Εικόνα 3.10, η σελίδα διαχείρισης βιβλίων περιλαμβάνει πίνακα με όλα τα καταχωρημένα βιβλία του συστήματος. Κάθε γραμμή περιλαμβάνει τις βασικές πληροφορίες και τα διαθέσιμα αντίτυπα, ενώ υπάρχουν ενέργειες για "Επεξεργασία" και "Διαγραφή". Η χρήση της βιβλιοθήκης DataTables προσφέρει δυνατότητα αναζήτησης, ταξινόμησης και σελιδοποίησης, καθιστώντας τη διαχείριση γρήγορη και αποτελεσματική ακόμη και με μεγάλο αριθμό εγγραφών.



Εικόνα 4.11: Επεξεργασία βιβλίου – admin

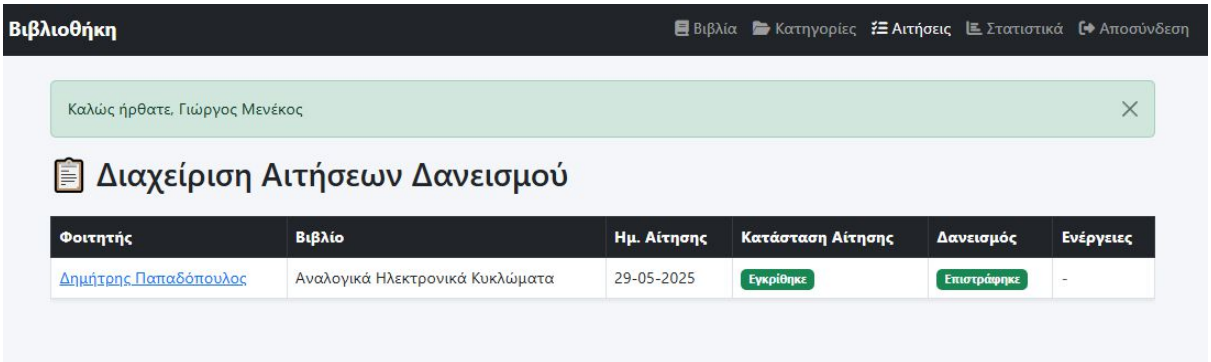
Στην Εικόνα 3.11 παρουσιάζεται η φόρμα επεξεργασίας ενός βιβλίου. Όλα τα στοιχεία μπορούν να τροποποιηθούν, συμπεριλαμβανομένου του εξωφύλλου και του πλήθους αντιτύπων. Η δυνατότητα άμεσης αλλαγής χωρίς επιπλέον βήματα ενισχύει τη χρηστικότητα για τον διαχειριστή. Στο κάτω μέρος εμφανίζεται προεπισκόπηση της εικόνας που έχει εισαχθεί.

Όπως φαίνεται στην Εικόνα 3.12, το σύστημα επιτρέπει στον διαχειριστή να προσθέσει ή να διαγράψει κατηγορίες βιβλίων. Οι κατηγορίες χρησιμοποιούνται για την ταξινόμηση των βιβλίων και εμφανίζονται σε όλες τις σχετικές φόρμες. Η λειτουργία αυτή απλοποιεί τη θεματική οργάνωση του υλικού της βιβλιοθήκης.

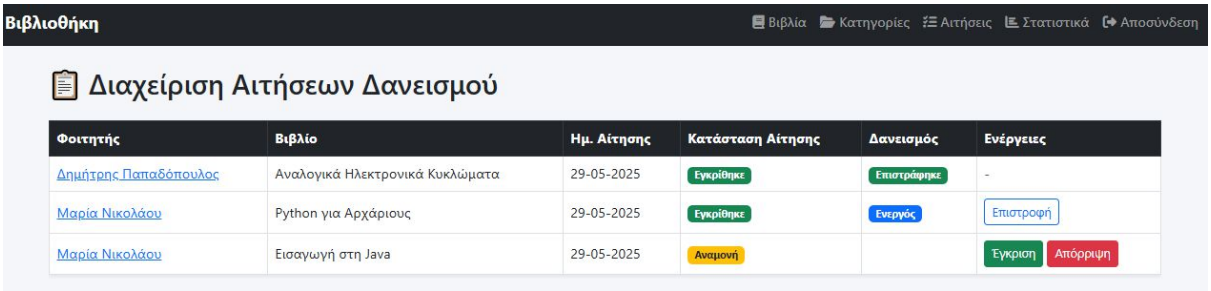


Εικόνα 4.12: Διαχείριση Κατηγοριών – admin

Στην Εικόνα 3.13 παρουσιάζεται ο πίνακας παρακολούθησης των αιτήσεων που έχουν καταχωρηθεί από φοιτητές. Εμφανίζονται πληροφορίες όπως ο φοιτητής, ο τίτλος του βιβλίου, η ημερομηνία αίτησης και οι καταστάσεις (αιτήματος και δανεισμού). Εφόσον η αίτηση έχει εγκριθεί και το βιβλίο έχει επιστραφεί, αυτό αποτυπώνεται καθαρά, προσφέροντας διαφάνεια και ιστορικότητα στη διαδικασία.



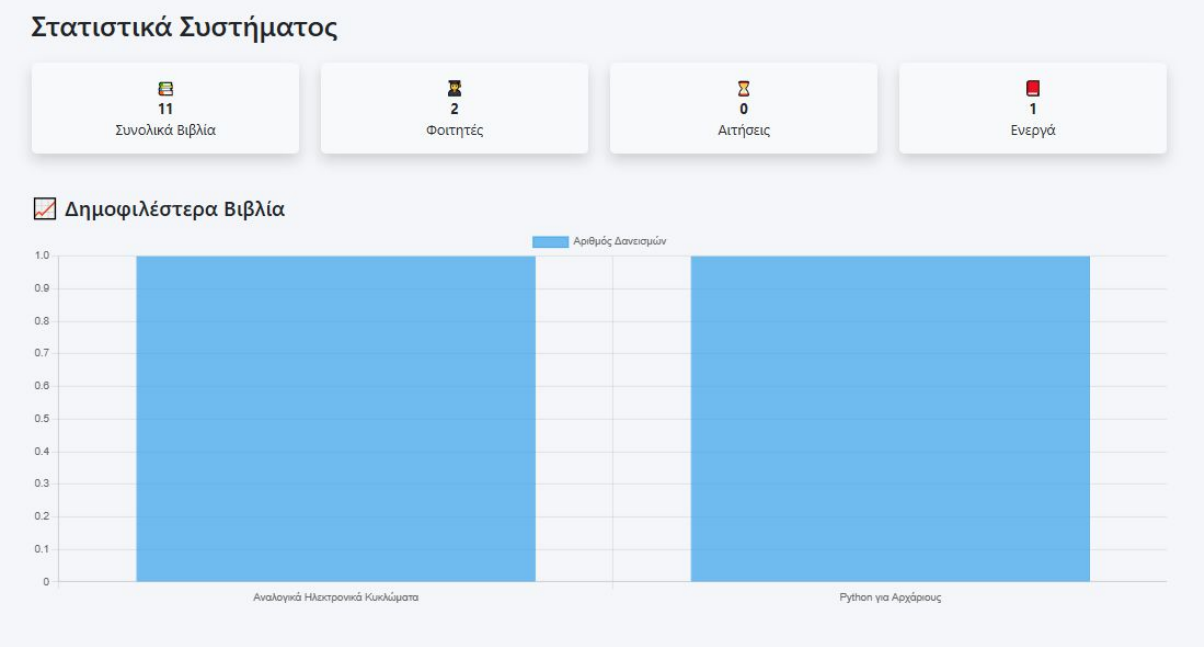
Εικόνα 4.13: Διαχείριση αιτήσεων δανεισμού 1 – admin



Εικόνα 4.14: Προσθήκη νέου βιβλίου στη Διαχείριση βιβλίων 2 – admin

Όπως φαίνεται στην Εικόνα 3.14, επεκτείνεται η προβολή αιτήσεων με παραδείγματα διαφορετικών σταδίων: αναμονή, έγκριση, δανεισμός και επιστροφή. Εμφανίζονται δυναμικά κουμπιά που

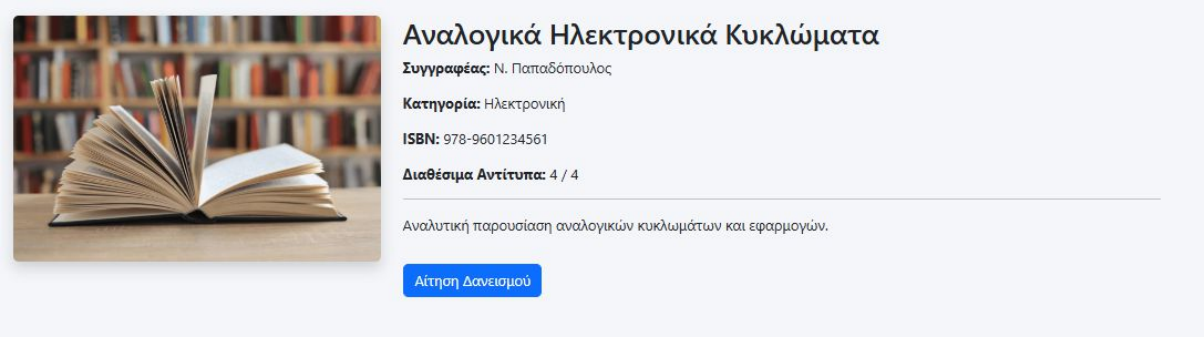
επιτρέπουν στον διαχειριστή να εγκρίνει, να απορρίψει ή να καταχωρήσει επιστροφή βιβλίου. Το interface είναι λειτουργικό και επιτρέπει άμεσες ενέργειες χωρίς περιττή πλοήγηση.



Εικόνα 4.15: Στατιστικά συστήματος – admin

Η Εικόνα 3.15 απεικονίζει τη σελίδα στατιστικών, η οποία συνοψίζει τη δραστηριότητα του συστήματος. Περιλαμβάνει αριθμό συνολικών βιβλίων, φοιτητών, αιτήσεων και ενεργών δανεισμών. Επιπλέον, παρουσιάζεται γράφημα με τα πιο δημοφιλή βιβλία βάσει του πλήθους δανεισμών. Τα στατιστικά βοηθούν τον διαχειριστή να αξιολογήσει την απόδοση της βιβλιοθήκης.

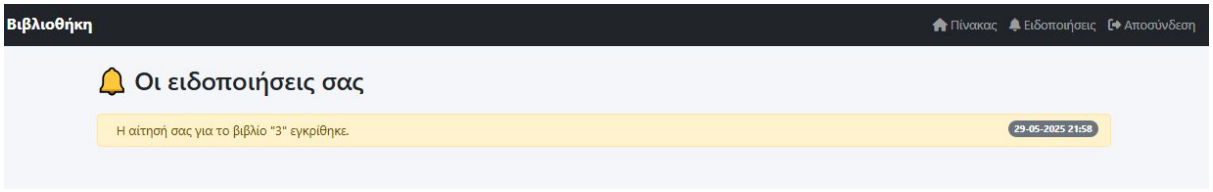
Student



Εικόνα 4.16: Προβολή βιβλίου – δυνατότητα αίτησης δανεισμού – student

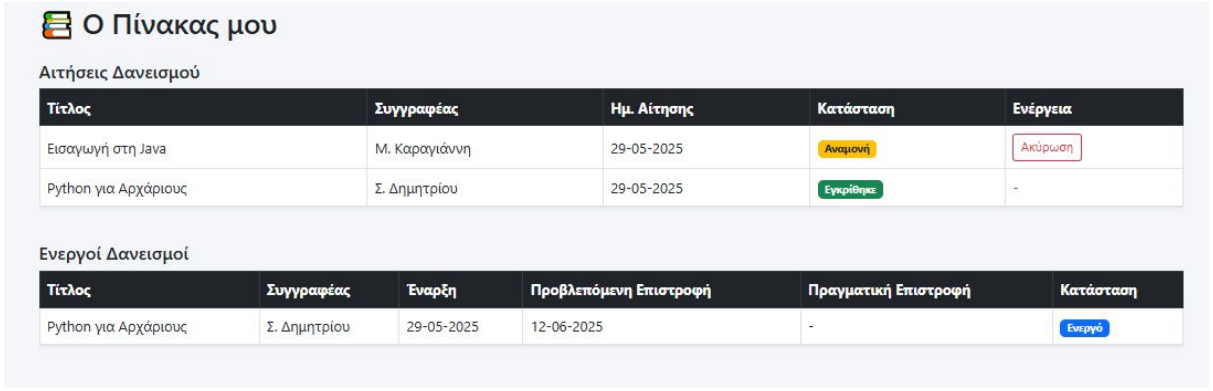
Στην Εικόνα 3.16 φαίνεται η σελίδα προβολής βιβλίου όπως την βλέπει ένας φοιτητής που έχει ήδη συνδεθεί. Εκτός από τις βασικές πληροφορίες, εμφανίζεται και κουμπί "Αίτηση Δανεισμού", το οποίο

επιτρέπει τη δημιουργία σχετικού αιτήματος. Αν δεν υπάρχει διαθεσιμότητα ή υπάρχει ήδη αίτηση, εμφανίζεται το κατάλληλο μήνυμα.



Εικόνα 4.17: Ειδοποιήσεις – student

Όπως παρουσιάζεται στην Εικόνα 3.17, η πλατφόρμα υποστηρίζει σύστημα ειδοποιήσεων για κάθε χρήστη. Ο φοιτητής βλέπει ειδοποιήσεις σχετικά με την πορεία των αιτήσεών του, την έγκριση ή την απόρριψη, καθώς και επιβεβαίωση επιστροφών. Οι ειδοποιήσεις αποθηκεύονται σε βάση και μαρκάρονται ως αναγνωσμένες με την επίσκεψη στη σελίδα.



Εικόνα 4.18: Πίνακας με τα βιβλία για τον student

Στην Εικόνα 3.18 απεικονίζεται η προσωπική σελίδα του φοιτητή με όλες του τις αιτήσεις και τους δανεισμούς. Για κάθε βιβλίο εμφανίζεται η κατάσταση, οι ημερομηνίες και οι διαθέσιμες ενέργειες, όπως ακύρωση αίτησης. Ο φοιτητής έχει πλήρη εικόνα της αλληλεπίδρασής του με το σύστημα.

4.4 Περιγραφή συστήματος μέσω του κώδικα

Η εφαρμογή βασίζεται στη γλώσσα Python και στο μικρό αλλά ισχυρό framework Flask. Το αρχείο app.py είναι ο πυρήνας της λειτουργικότητας και περιλαμβάνει όλες τις βασικές διαδρομές (routes) και την επιχειρησιακή λογική.

```
# Εισαγωγή των βασικών αντικειμένων της Flask για τη δημιουργία της εφαρμογής και των views
from flask import Flask, render_template, request, redirect, url_for, session, flash
# Εισαγωγή του connector για σύνδεση με MySQL βάση δεδομένων
import mysql.connector
# Εισαγωγή της custom συνάρτησης που δημιουργεί σύνδεση με τη βάση (από αρχείο db_config.py)
from db_config import get_db_connection
# Εισαγωγή της βιβλιοθήκης csv για εξαγωγή δεδομένων σε μορφή CSV
import csv
# Εισαγωγή του StringIO για προσωρινή αποθήκευση αρχείων σε μνήμη (αντί για φυσικό δίσκο)
from io import StringIO
```

Εισάγονται οι βασικές βιβλιοθήκες για Flask και σύνδεση με τη βάση MySQL. Η get_db_connection() προέρχεται από εξωτερικό αρχείο και χρησιμοποιείται για ασφαλή διαχείριση της σύνδεσης.

Το παρακάτω τμήμα του κώδικα χειρίζεται τη διαδικασία σύνδεσης ενός χρήστη στην πλατφόρμα. Ανάλογα με τον ρόλο του (διαχειριστής ή φοιτητής), οδηγείται στο αντίστοιχο dashboard. Σε περίπτωση λανθασμένων στοιχείων, εμφανίζεται μήνυμα σφάλματος.

```
# Route για τη σελίδα σύνδεσης (login)
@app.route('/login', methods=['GET', 'POST'])
def login():
    # Αν η μέθοδος είναι POST (δηλαδή έγινε υποβολή φόρμας)
    if request.method == 'POST':
        # Λήψη του email και του password που καταχωρήθηκαν από τη φόρμα
        email = request.form['email']
        password = request.form['password']

        # Δημιουργία σύνδεσης με τη βάση δεδομένων
        conn = get_db_connection()
        # Ορισμός cursor με επιστροφή αποτελεσμάτων ως λεξικά
        cursor = conn.cursor(dictionary=True)

        # Εκτέλεση SQL ερωτήματος για αναζήτηση χρήστη με το συγκεκριμένο email και password
        cursor.execute("SELECT * FROM users WHERE email = %s AND password = %s", (email, password))
        # Λήψη του αποτελέσματος (αν υπάρχει κάποιος χρήστης)
        user = cursor.fetchone()

        # Κλείσιμο cursor και σύνδεσης
```

```
        cursor.close()
        conn.close()

        # Αν βρέθηκε χρήστης με αυτά τα στοιχεία
        if user:
            # Αποθήκευση των στοιχείων του χρήστη στο session (για να γνωρίζουμε ποιος είναι συνδεδεμένος)
            session['user_id'] = user['id']
            session['user_name'] = user['name']
            session['user_role'] = user['role']

            # Μήνυμα επιτυχούς σύνδεσης
            flash('Καλώς ήρθατε, ' + user['name'], 'success')

            # Ανακατεύθυνση είτε στο admin dashboard αν είναι διαχειριστής είτε στο dashboard του φοιτητή
            return redirect(url_for('admin_dashboard' if user['role'] == 'admin' else 'dashboard'))
        else:
            # Αν δεν βρέθηκε χρήστης με αυτά τα στοιχεία, εμφάνιση μηνύματος σφάλματος
            flash('Λάθος email ή κωδικός.', 'danger')
            return redirect(url_for('login'))

    # Αν είναι GET (απλή επίσκεψη της σελίδας), προβολή της φόρμας σύνδεσης
    return render_template('login.html')
```

Το παρακάτω τμήμα του κώδικα διαχειρίζεται τη διαδικασία αποσύνδεσης του χρήστη. Καθαρίζει τα δεδομένα του session και επιστρέφει τον χρήστη στην αρχική σελίδα (welcome), εμφανίζοντας και σχετικό μήνυμα.

```
# Route για αποσύνδεση χρήστη (logout)
@app.route('/logout')
def logout():
    # Καθαρισμός όλων των δεδομένων που είναι αποθηκευμένα στο session
    session.clear()

    # Εμφάνιση μηνύματος ότι ο χρήστης αποσυνδέθηκε
    flash('Αποσυνδεθήκατε.', 'info')

    # Ανακατεύθυνση στην αρχική σελίδα (welcome)
    return redirect(url_for('welcome'))
```

Το παρακάτω τμήμα του κώδικα επιτρέπει σε έναν χρήστη (φοιτητή ή επισκέπτη) να δει τις πληροφορίες ενός συγκεκριμένου βιβλίου. Εμφανίζονται τίτλος, συγγραφέας, κατηγορία, ISBN, περιγραφή και αριθμός διαθέσιμων αντιτύπων.

```
# Route για προβολή λεπτομερειών ενός βιβλίου
@app.route('/book/<int:book_id>')
def view_book(book_id):
    # Δημιουργία σύνδεσης με τη βάση δεδομένων
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Εκτέλεση ερωτήματος για να φέρει το βιβλίο με το συγκεκριμένο ID
    cursor.execute("""
        SELECT b.*, c.name AS category
        FROM books b
        JOIN categories c ON b.category_id = c.id
        WHERE b.id = %s
    """, (book_id,))

    # Αποθήκευση του αποτελέσματος
    book = cursor.fetchone()

    # Κλείσιμο σύνδεσης με τη βάση
    cursor.close()
    conn.close()

    # Αν δεν βρέθηκε το βιβλίο, εμφάνιση μηνύματος και επιστροφή στην αρχική σελίδα
    if not book:
        flash('Το βιβλίο δεν βρέθηκε.', 'danger')
        return redirect(url_for('welcome'))

    # Απόδοση του template view_book.html με τα δεδομένα του βιβλίου
    return render_template('view_book.html', book=book)
```

Αυτό το route εμφανίζει τις ειδοποιήσεις ενός φοιτητή που σχετίζονται με αιτήσεις, εγκρίσεις ή επιστροφές βιβλίων.

```
# Route για εμφάνιση των ειδοποιήσεων του φοιτητή
@app.route('/notifications')
def notifications():
    # Αν ο χρήστης δεν είναι φοιτητής, δεν επιτρέπεται η πρόσβαση
    if session.get('user_role') != 'student':
        flash('Απαγορεύεται η πρόσβαση.', 'danger')
        return redirect(url_for('login'))

    # Ανάκτηση του ID του συνδεδεμένου χρήστη
    user_id = session['user_id']

    # Σύνδεση με βάση
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Ανάκτηση των ειδοποιήσεων του χρήστη (ταξινομημένες από τις πιο πρόσφατες)
    cursor.execute("""
        SELECT * FROM notifications
        WHERE user_id = %s
        ORDER BY created_at DESC
    """, (user_id,))
    notifications = cursor.fetchall()

    cursor.close()
    conn.close()

    # Απόδοση του template notifications.html με τις ειδοποιήσεις
    return render_template('notifications.html', notifications=notifications)
```

Ο φοιτητής μπορεί να ακυρώσει μια αίτηση δανεισμού εφόσον βρίσκεται σε κατάσταση "αναμονής".

```
# Route για ακύρωση αίτησης δανεισμού από φοιτητή
@app.route('/cancel-request/<int:request_id>')
def cancel_request(request_id):
    # Έλεγχος αν είναι φοιτητής
    if session.get('user_role') != 'student':
        flash('Απαγορεύεται.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor()

    # Ενημέρωση της κατάστασης της αίτησης σε "ακυρώθηκε" μόνο αν είναι σε αναμονή
    cursor.execute("""
        UPDATE loan_requests
        SET status = 'cancelled'
        WHERE id = %s AND user_id = %s AND status = 'pending'
    """, (request_id, user_id))

    if cursor.rowcount > 0:
        flash('Η αίτηση ακυρώθηκε.', 'info')
    else:
        flash('Δεν ήταν δυνατή η ακύρωση.', 'warning')

    conn.commit()
    cursor.close()
    conn.close()

    return redirect(url_for('dashboard'))
```

Ο διαχειριστής μπορεί να εξάγει στατιστικά για τις αιτήσεις και τους δανεισμούς σε αρχείο .csv.

```
# Route για εξαγωγή στατιστικών σε CSV
```

```
@app.route('/admin/export')
def export_csv():
    # Έλεγχος πρόσβασης admin
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    # Σύνδεση με βάση
    conn = get_db_connection()
    cursor = conn.cursor()

    # Ανάκτηση όλων των δανεισμών με πληροφορίες για το χρήστη και το βιβλίο
    cursor.execute("""
        SELECT u.name, b.title, l.loan_date, l.return_date, l.status
        FROM loans l
        JOIN users u ON l.user_id = u.id
        JOIN books b ON l.book_id = b.id
    """)
    rows = cursor.fetchall()

    # Δημιουργία CSV string με κεφαλίδες
    si = StringIO()
    writer = csv.writer(si)
    writer.writerow(['Όνομα Χρήστη', 'Βιβλίο', 'Ημ. Δανεισμού', 'Ημ. Επιστροφής', 'Κατάσταση'])

    # Προσθήκη των γραμμών στο αρχείο
    for row in rows:
        writer.writerow(row)

    cursor.close()
    conn.close()

    # Δημιουργία και επιστροφή του αρχείου ως απάντηση (για κατέβασμα)
```

```
    output = si.getvalue()
    return Response(output, mimetype='text/csv', headers={"Content-Disposition": "attachment;filename=stats.csv"})
```

Ο διαχειριστής βλέπει μια σύνοψη με αριθμό βιβλίων, ενεργών αιτήσεων και φοιτητών.

```
# Admin Dashboard – Σύνοψη
@app.route('/admin/dashboard')
def admin_dashboard():
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Μετρήσεις: σύνολο βιβλίων
    cursor.execute("SELECT COUNT(*) FROM books")
    total_books = cursor.fetchone()[0]

    # Μετρήσεις: ενεργές αιτήσεις
    cursor.execute("SELECT COUNT(*) FROM loan_requests WHERE status = 'pending'")
    pending_requests = cursor.fetchone()[0]

    # Μετρήσεις: εγγεγραμμένοι φοιτητές
    cursor.execute("SELECT COUNT(*) FROM users WHERE role = 'student'")
    total_students = cursor.fetchone()[0]

    cursor.close()
    conn.close()

    return render_template('admin_dashboard.html',
                           total_books=total_books,
                           pending_requests=pending_requests,
                           total_students=total_students)
```

4.5 Ασφάλεια

Η ασφάλεια αποτελεί κρίσιμο παράγοντα για κάθε διαδικτυακή πλατφόρμα ειδικά όταν αυτή διαχειρίζεται προσωπικά δεδομένα χρηστών και κρίσιμες λειτουργίες όπως αιτήσεις και ιστορικά δανεισμών. Το σύστημα διαχείρισης και δανεισμού βιβλιοθήκης που αναπτύχθηκε για τις ανάγκες του Τμήματος βασίστηκε σε βασικές αρχές ασφάλειας εφαρμογών web και έχει προστασία από κακόβουλες ενέργειες διασφαλίζοντας τα δεδομένα και αποτρέποντας μη εξουσιοδοτημένη πρόσβαση.

Έλεγχος Πρόσβασης (Role-Based Access Control)

Το σύστημα εφαρμόζει μηχανισμό ελέγχου πρόσβασης με βάση ρόλους (RBAC). Οι χρήστες διακρίνονται σε δύο βασικές κατηγορίες:

- **Φοιτητές (student):** έχουν πρόσβαση μόνο στα δικά τους δεδομένα, όπως προβολή βιβλίων, αποστολή αιτήσεων και παρακολούθηση της κατάστασής τους.
- **Διαχειριστές (admin):** έχουν πλήρη πρόσβαση στο σύστημα, όπως διαχείριση βιβλίων, κατηγοριών, έγκριση/απόρριψη αιτήσεων και εξαγωγή στατιστικών.

Σε κάθε route του Flask, γίνεται έλεγχος του session['user_role'] ώστε να αποτρέπεται η πρόσβαση μη εξουσιοδοτημένων χρηστών. Σε περίπτωση απόπειρας, ο χρήστης ανακατευθύνεται στη σελίδα σύνδεσης ή λαμβάνει αντίστοιχο μήνυμα.

Επαλήθευση Ταυτότητας Χρηστών

Η πρόσβαση στη λειτουργικότητα της εφαρμογής απαιτεί έγκυρη σύνδεση με email και κωδικό πρόσβασης. Το σύστημα συγκρίνει τα διαπιστευτήρια των χρηστών με τα δεδομένα της βάσης.

Προστασία από SQL Injection

Η σύνδεση με τη βάση δεδομένων υλοποιείται με χρήση cursor.execute(...) με παραμέτρους, π.χ.:

cursor.execute("SELECT * FROM users WHERE email = %s AND password = %s", (email, password))

Αυτή η τεχνική αποτρέπει SQL injection επιθέσεις, καθώς η βιβλιοθήκη MySQLdb ή mysql-connector διαχειρίζεται σωστά τις παραμέτρους, απομονώνοντας τα δεδομένα του χρήστη από το query.

Διαχείριση Συνεδρίας (Session Security)

Η εφαρμογή χρησιμοποιεί το Flask session για τη διατήρηση της σύνδεσης. Το session['user_id'], session['user_name'] και session['user_role'] ορίζονται κατά την επιτυχή σύνδεση και διαγράφονται πλήρως κατά το logout με session.clear().

Επιπλέον:

- Ο secret_key του Flask ορίζεται για την προστασία του session cookie από τροποποιήσεις.
- Θα μπορούσε να ενισχυθεί με ενεργοποίηση secure cookies (μόνο μέσω HTTPS) και timeout συνεδρίας.

Εξουσιοδότηση Ενεργειών

Πέρα από τον έλεγχο ρόλων, πολλές λειτουργίες στο backend ελέγχουν την ταυτότητα του χρήστη. Για παράδειγμα, στην ακύρωση αίτησης:

WHERE id = %s AND user_id = %s

Έτσι, ακόμα και αν κάποιος αλλάξει το URL, δεν μπορεί να ακυρώσει αίτηση άλλου χρήστη.

Ενημέρωση Χρηστών με Ειδοποιήσεις

Η αποστολή ειδοποιήσεων (π.χ. έγκριση/απόρριψη/επιστροφή) υλοποιείται με αποθήκευση στη βάση και εμφάνιση σε ειδική σελίδα.

Κεφάλαιο 5ο: Συμπεράσματα - βελτιώσεις

Η υλοποίηση της πλατφόρμας διαχείρισης και δανεισμού βιβλιοθήκης είναι μια λειτουργική λύση για την υποστήριξη ενός τμήματος στην οργάνωση των βιβλίων του. Το σύστημα δίνει τη δυνατότητα σε φοιτητές να περιηγούνται στα διαθέσιμα βιβλία, να καταθέτουν αιτήσεις δανεισμού και να παρακολουθούν την πορεία τους. Παράλληλα ο διαχειριστής έχει στη διάθεσή του ένα περιβάλλον πλήρους εποπτείας των λειτουργιών και μέσω του οποίου μπορεί να εγκρίνει ή να απορρίπτει αιτήσεις, να προσθέτει και να επεξεργάζεται βιβλία και κατηγορίες αλλά και να παρακολουθεί τη συνολική στατιστική εικόνα του συστήματος. Το γεγονός ότι υπάρχει διαχωρισμός ρόλων ενισχύει την ασφάλεια και αποτρέπει την πρόσβαση σε ευαίσθητα δεδομένα από μη εξουσιοδοτημένους χρήστες. Ο σχεδιασμός βασίστηκε σε καθαρό backend, πλήρως οργανωμένο ανά λειτουργία, και σε frontend που αξιοποιεί Bootstrap για να προσφέρει ένα καθαρό και φιλικό περιβάλλον εργασίας. Να σημειωθεί ότι έχει χρησιμοποιηθεί ai τεχνολογία για τη διόρθωση συντακτικού κάποιων κειμένων στην εργασία.

Υπάρχουν κάποια περιθώρια βελτίωσης και ένα από αυτά είναι η αποθήκευση των κωδικών πρόσβασης θα πρέπει να μεταβεί σε κρυπτογραφημένη μορφή χρησιμοποιώντας ασφαλείς αλγορίθμους όπως ο bcrypt, ώστε να προστατεύονται οι λογαριασμοί σε περίπτωση παραβίασης της βάσης δεδομένων. Επίσης η αναζήτηση και το φιλτράρισμα βιβλίων θα προσέφεραν στον φοιτητή καλύτερη πλοήγηση ειδικά σε περιπτώσεις μεγάλου καταλόγου. Ένα άλλο σημείο που μπορεί να ενισχύσει τη χρηστικότητα είναι η δυνατότητα ειδοποιήσεων μέσω email οι οποίες θα ενημερώνουν τον χρήστη για αλλαγές στην κατάσταση της αίτησής του ή για την επιστροφή βιβλίου. Στην παρούσα φάση μόνο ο διαχειριστής έχει δικαίωμα καταχώρησης επιστροφής, κάτι που θα μπορούσε να συμπληρωθεί με μια επιλογή «αίτησης επιστροφής» από την πλευρά του φοιτητή. Μια ακόμη δυνατότητα είναι η υποστήριξη ηλεκτρονικών αρχείων (όπως PDF) ώστε η πλατφόρμα να μπορεί να επεκταθεί και ως ψηφιακή βιβλιοθήκη.

Αξίζει να αναφερθεί ότι θα μπορούσε να προστεθεί σε κάθε φοιτητή προσωπική στατιστική αναφορά ώστε να βλέπει πόσα βιβλία έχει δανειστεί, πότε και ποια είδη προτιμά περισσότερο. Επίσης η δυνατότητα δημιουργίας αναφορών καθυστερήσεων θα βοηθούσε τον διαχειριστή να παρακολουθεί ποιες επιστροφές είναι εκτός χρονικού πλαισίου και να λαμβάνει κάποια μέτρα. Σε επίπεδο ασφάλειας το σύστημα θα μπορούσε να ενισχυθεί περαιτέρω με την ενεργοποίηση λήξης συνεδριών, τη χρήση HTTPS και την προστασία από CSRF επιθέσεις στις φόρμες.

Η πλατφόρμα που αναπτύχθηκε αποτελεί μια πρώτη ισχυρή βάση για τη διαχείριση της βιβλιοθήκης ενός εκπαιδευτικού τμήματος. Καλύπτει τις βασικές ανάγκες διαχείρισης φυσικών αντιτύπων με πλήρη ροή αιτήσεων, έγκρισης, δανεισμού και επιστροφής. Με σταδιακές βελτιώσεις και επεκτάσεις μπορεί να εξελιχθεί σε ένα ολοκληρωμένο σύστημα διαχείρισης φυσικών και ψηφιακών βιβλίων, προσφέροντας μια ποιοτική εμπειρία τόσο για τον φοιτητή όσο και για τον διαχειριστή.

ΒΙΒΛΙΟΓΡΑΦΙΑ

[1] https://www.python.org/

[2] https://en.wikipedia.org/wiki/Python_(programming_language)

[3] https://www.w3schools.com/python/

[4] https://www.geeksforgeeks.org/python-language-advantages-applications/

[5] https://realpython.com/

[6] https://flask.palletsprojects.com/en/stable/

[7] https://en.wikipedia.org/wiki/Flask_(web_framework)

[8] https://pypi.org/project/Flask/

[9] https://www.geeksforgeeks.org/flask-tutorial/

[10] https://dev.to/detimo/python-flask-pros-and-cons-1mlo

[11] https://www.mysqltutorial.org/

[12] https://www.w3schools.com/MySQL/default.asp

[13] https://www.remoteplatz.ch/en/blog/advantages-and-disadvantages-of-mysql

[14] https://www.w3schools.com/python/python_mysql_getstarted.asp

[15] https://getbootstrap.com/

[16] https://www.w3schools.com/bootstrap/bootstrap_get_started.asp

[17] https://www.chartjs.org/

[18] https://koha-community.org/

[19] https://en.wikipedia.org/wiki/Evergreen_(software)

[20] https://opalsinfo.net/

[21] https://en.wikipedia.org/wiki/OpenBiblio

ΠΑΡΑΡΤΗΜΑ του ΚΩΔΙΚΑ

App.py

```
from flask import Flask, render_template, request, redirect, url_for, session, flash
import mysql.connector
from db_config import get_db_connection
import csv
from io import StringIO
from flask import Response
from datetime import datetime, timedelta, date

app = Flask(__name__)
app.secret_key = 'your_secret_key'

@app.route('/')
def welcome():
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT b.id, b.title, b.author, b.image_url, c.name AS category
        FROM books b
        JOIN categories c ON b.category_id = c.id
    """)
    books = cursor.fetchall()
    cursor.close()
    conn.close()
    return render_template('welcome.html', books=books)

@app.route('/login', methods=['GET', 'POST'])
def login():
    if request.method == 'POST':
        email = request.form['email']
        password = request.form['password']

        conn = get_db_connection()
        cursor = conn.cursor(dictionary=True)
        cursor.execute("SELECT * FROM users WHERE email = %s AND password = %s", (email, password))
        user = cursor.fetchone()
        cursor.close()
        conn.close()

        if user:
            session['user_id'] = user['id']
            session['user_name'] = user['name']
            session['user_role'] = user['role']
            flash('Καλώς ήρθατε, ' + user['name'], 'success')
            return redirect(url_for('admin_dashboard' if user['role'] == 'admin' else 'dashboard'))
        else:
            flash('Λάθος email ή κωδικός.', 'danger')
            return redirect(url_for('login'))
    return render_template('login.html')

@app.route('/logout')
def logout():
    session.clear()
    flash('Αποσυνδεθήκατε.', 'info')
    return redirect(url_for('welcome'))

@app.route('/book/<int:book_id>')
def view_book(book_id):
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
        SELECT b.*, c.name AS category
        FROM books b
        JOIN categories c ON b.category_id = c.id
        WHERE b.id = %s
    """, (book_id,))
    book = cursor.fetchone()
    cursor.close()
    conn.close()

    if not book:
        flash('Το βιβλίο δεν βρέθηκε.', 'danger')
        return redirect(url_for('welcome'))
```

```
    return render_template('view_book.html', book=book)

from datetime import datetime, timedelta
from flask import abort

@app.route('/request-loan/<int:book_id>')
def request_loan(book_id):
    if not session.get('user_id') or session.get('user_role') != 'student':
        flash('Η πρόσβαση επιτρέπεται μόνο σε φοιτητές.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT * FROM books WHERE id = %s", (book_id,))
    book = cursor.fetchone()
    if not book:
        flash('Το βιβλίο δεν βρέθηκε.', 'danger')
        return redirect(url_for('welcome'))

    # Έλεγχος αν υπάρχει ήδη pending αίτηση
    cursor.execute("""
        SELECT * FROM loan_requests
        WHERE user_id = %s AND book_id = %s AND status = 'pending'
    """, (user_id, book_id))
    existing = cursor.fetchone()
    if existing:
        flash('Έχετε ήδη αιτηθεί το συγκεκριμένο βιβλίο.', 'warning')
        return redirect(url_for('view_book', book_id=book_id))

    cursor.execute("""
        INSERT INTO loan_requests (user_id, book_id)
        VALUES (%s, %s)
    """, (user_id, book_id))
    conn.commit()
    cursor.close()
    conn.close()

    flash('Η αίτηση καταχωρήθηκε και αναμένει έγκριση.', 'success')
    return redirect(url_for('view_book', book_id=book_id))

@app.route('/cancel-request/<int:request_id>')
def cancel_request(request_id):
    if not session.get('user_id') or session.get('user_role') != 'student':
        flash('Δεν επιτρέπεται.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        DELETE FROM loan_requests
        WHERE id = %s AND user_id = %s AND status = 'pending'
    """, (request_id, user_id))
    conn.commit()
    cursor.close()
    conn.close()

    flash('Η αίτηση ακυρώθηκε.', 'info')
    return redirect(url_for('dashboard'))

@app.route('/dashboard')
def dashboard():
    if not session.get('user_id') or session.get('user_role') != 'student':
        flash('Η πρόσβαση επιτρέπεται μόνο σε φοιτητές.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Αιτήσεις
    cursor.execute("""
        SELECT r.*, b.title, b.author
        FROM loan_requests r
        JOIN books b ON r.book_id = b.id
        WHERE r.user_id = %s
        ORDER BY r.request_date DESC
    """, (user_id,))
    requests = cursor.fetchall()

    # Εγκεκριμένοι δανεισμοί
    cursor.execute("""
        SELECT l.*, b.title, b.author
```

```
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.user_id = %s
        ORDER BY l.loan_date DESC
    """, (user_id,))
    loans = cursor.fetchall()

    cursor.close()
    conn.close()

    return render_template('dashboard.html', requests=requests, loans=loans)

@app.route('/cancel-loan/<int:loan_id>')
def cancel_loan(loan_id):
    if not session.get('user_id') or session.get('user_role') != 'student':
        flash('Δεν επιτρέπεται.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("DELETE FROM loans WHERE id = %s AND user_id = %s AND status = 'pending'", (loan_id, user_id))
    conn.commit()
    cursor.close()
    conn.close()

    flash('Η αίτηση ακυρώθηκε.', 'info')
    return redirect(url_for('dashboard'))

@app.route('/admin/user/<int:user_id>/loans')
def admin_user_loans(user_id):
    if session.get('user_role') != 'admin':
        flash('Απαγορεύεται.', 'danger')
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT name FROM users WHERE id = %s", (user_id,))
    user = cursor.fetchone()
    if not user:
        flash('Ο φοιτητής δεν βρέθηκε.', 'warning')
        cursor.close()
        conn.close()
        return redirect(url_for('admin_requests'))

    cursor.execute("""
        SELECT l.*, b.title, b.author
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.user_id = %s
        ORDER BY l.loan_date DESC
    """, (user_id,))
    loans = cursor.fetchall()

    cursor.close()
    conn.close()
    return render_template('admin_user_loans.html', user=user, loans=loans)

@app.route('/admin/requests')
def admin_requests():
    if session.get('user_role') != 'admin':
        flash('Απαγορεύεται η πρόσβαση.', 'danger')
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("""
    SELECT r.*, u.name AS student_name, b.title AS book_title,
        b.available_copies, b.total_copies,
        l.id AS loan_id, l.status AS loan_status
    FROM loan_requests r
    JOIN users u ON r.user_id = u.id
    JOIN books b ON r.book_id = b.id
    LEFT JOIN loans l ON l.request_id = r.id
    ORDER BY r.request_date DESC
    """)
    requests = cursor.fetchall()

    cursor.close()
    conn.close()

    # for r in requests:
```

```
    #     print(f"req_id={r['id']} loan_id={r['loan_id']} loan_status={r['loan_status']}")

    return render_template('admin_requests.html', requests=requests)

@app.route('/admin')
def admin_dashboard():
    if session.get('user_role') != 'admin':
        flash('Απαγορεύεται η πρόσβαση.', 'danger')
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)
    cursor.execute("""
    SELECT r.*, u.name AS student_name, b.title AS book_title,
        b.available_copies, b.total_copies,
        l.id AS loan_id, l.status AS loan_status
    FROM loan_requests r
    JOIN users u ON r.user_id = u.id
    JOIN books b ON r.book_id = b.id
    LEFT JOIN loans l ON l.request_id = r.id
    ORDER BY r.request_date DESC

    """)
    requests = cursor.fetchall()
    cursor.close()
    conn.close()
    return render_template('admin_requests.html', requests=requests)

@app.route('/approve-request/<int:request_id>')
def approve_request(request_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Φέρνουμε την αίτηση
    cursor.execute("SELECT * FROM loan_requests WHERE id = %s AND status = 'pending'", (request_id,))
    req = cursor.fetchone()

    if not req:
        flash('Η αίτηση δεν βρέθηκε ή έχει ήδη επεξεργαστεί.', 'warning')
        return redirect(url_for('admin_requests'))

    loan_date = datetime.now().date()
    return_date = loan_date + timedelta(days=14)

    # Δημιουργία δανεισμού και ενημέρωση αίτησης
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO loans (user_id, book_id, loan_date, return_date, status, request_id)
        VALUES (%s, %s, %s, %s, 'borrowed', %s)
    """, (req['user_id'], req['book_id'], loan_date, return_date, request_id))

    cursor.execute("UPDATE books SET available_copies = available_copies - 1 WHERE id = %s", (req['book_id'],))
    cursor.execute("UPDATE loan_requests SET status = 'approved' WHERE id = %s", (request_id,))

    # Ειδοποίηση
    cursor.execute("""
        INSERT INTO notifications (user_id, message)
        VALUES (%s, %s)
    """, (req['user_id'], f'Η αίτησή σας για το βιβλίο "{req["book_id"]}" εγκρίθηκε.'))

    conn.commit()
    conn.close()

    flash('Η αίτηση εγκρίθηκε και δημιουργήθηκε δανεισμός.', 'success')
    return redirect(url_for('admin_requests'))

@app.route('/admin/return/<int:loan_id>')
def admin_return_book(loan_id):
    if session.get('user_role') != 'admin':
        flash('Απαγορεύεται η πρόσβαση.', 'danger')
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Ενημέρωση δανεισμού
    cursor.execute("""
        UPDATE loans
        SET status = 'returned', actual_return = %s
        WHERE id = %s AND status = 'borrowed'
    """, (date.today(), loan_id))
```

```
    if cursor.rowcount == 0:
        flash('Ο δανεισμός δεν είναι ενεργός ή δεν βρέθηκε.', 'warning')
        cursor.close()
        conn.close()
        return redirect(url_for('admin_requests'))

    # Ενημέρωση αντίτυπων
    cursor.execute("""
        UPDATE books
        SET available_copies = available_copies + 1
        WHERE id = (SELECT book_id FROM loans WHERE id = %s)
    """, (loan_id,))

    # Εισαγωγή ειδοποίησης
    cursor.execute("""
        INSERT INTO notifications (user_id, message)
        SELECT user_id, CONCAT('Η επιστροφή του βιβλίου "', b.title, '" καταχωρήθηκε.')
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.id = %s
    """, (loan_id,))

    conn.commit()
    cursor.close()
    conn.close()

    flash('Η επιστροφή καταχωρήθηκε επιτυχώς.', 'success')
    return redirect(url_for('admin_requests'))

@app.route('/reject-request/<int:request_id>')
def reject_request(request_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Παίρνουμε την αίτηση για ειδοποίηση
    cursor.execute("""
        SELECT r.user_id, b.title
        FROM loan_requests r
        JOIN books b ON r.book_id = b.id
        WHERE r.id = %s
    """, (request_id,))
    data = cursor.fetchone()

    # Ενημερώνουμε την αίτηση
    cursor.execute("UPDATE loan_requests SET status = 'rejected' WHERE id = %s", (request_id,))

    # Εισαγωγή ειδοποίησης
    cursor.execute("""
        INSERT INTO notifications (user_id, message)
        VALUES (%s, %s)
    """, (data['user_id'], f'Η αίτησή σας για το βιβλίο "{data["title"]}" απορρίφθηκε.'))

    conn.commit()
    conn.close()
    flash('Η αίτηση απορρίφθηκε.', 'info')
    return redirect(url_for('admin_dashboard'))

@app.route('/approve-loan/<int:loan_id>')
def approve_loan(loan_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Μείωση διαθέσιμων αντιτύπων
    cursor.execute("""
        UPDATE books b
        JOIN loans l ON b.id = l.book_id
        SET b.available_copies = b.available_copies - 1
        WHERE l.id = %s AND b.available_copies > 0
    """, (loan_id,))

    # Εγκρίνουμε μόνο αν υπάρχουν αντίτυπα
    if cursor.rowcount == 0:
        flash('Δεν υπάρχουν διαθέσιμα αντίτυπα.', 'warning')
    else:
        cursor.execute("UPDATE loans SET status = 'borrowed' WHERE id = %s", (loan_id,))
        flash('Η αίτηση εγκρίθηκε.', 'success')

        # Μετά την αλλαγή κατάστασης
        cursor.execute("""
```

```
            INSERT INTO notifications (user_id, message)
            SELECT user_id, CONCAT('Η αίτησή σας για το βιβλίο "', b.title, '" εγκρίθηκε.')
            FROM loans l
            JOIN books b ON l.book_id = b.id
            WHERE l.id = %s
        """, (loan_id,))

    conn.commit()
    cursor.close()
    conn.close()
    return redirect(url_for('admin_dashboard'))

@app.route('/reject-loan/<int:loan_id>')
def reject_loan(loan_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("UPDATE loans SET status = 'rejected' WHERE id = %s", (loan_id,))
    conn.commit()
    cursor.close()
    conn.close()

    cursor.execute("""
        INSERT INTO notifications (user_id, message)
        SELECT user_id, CONCAT('Η αίτησή σας για το βιβλίο "', b.title, '" απορρίφθηκε.')
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.id = %s
    """, (loan_id,))

    flash('Η αίτηση απορρίφθηκε.', 'info')
    return redirect(url_for('admin_dashboard'))

@app.route('/admin/stats')
def admin_stats():
    if not session.get('user_id') or session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT COUNT(*) AS total_books FROM books")
    total_books = cursor.fetchone()['total_books']

    cursor.execute("SELECT COUNT(*) AS total_students FROM users WHERE role = 'student'")
    total_students = cursor.fetchone()['total_students']

    cursor.execute("SELECT COUNT(*) AS pending_loans FROM loans WHERE status = 'pending'")
    pending_loans = cursor.fetchone()['pending_loans']

    cursor.execute("SELECT COUNT(*) AS active_loans FROM loans WHERE status = 'borrowed'")
    active_loans = cursor.fetchone()['active_loans']

    cursor.execute("""
        SELECT b.title, COUNT(*) AS times
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.status IN ('borrowed', 'returned')
        GROUP BY l.book_id
        ORDER BY times DESC
        LIMIT 5
    """)
    top_books = cursor.fetchall()

    cursor.close()
    conn.close()

    return render_template('admin_stats.html', total_books=total_books, total_students=total_students,
                           pending_loans=pending_loans, active_loans=active_loans, top_books=top_books)

@app.route('/myloans')
def my_loans():
    if not session.get('user_id') or session.get('user_role') != 'student':
        flash('Η πρόσβαση επιτρέπεται μόνο σε φοιτητές.', 'danger')
        return redirect(url_for('login'))

    user_id = session['user_id']
    status_filter = request.args.get('status')  # π.χ. borrowed, returned, pending

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    query = """
```

```
        SELECT l.*, b.title, b.author
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.user_id = %s
    """
    params = [user_id]

    if status_filter:
        query += " AND l.status = %s"
        params.append(status_filter)

    query += " ORDER BY l.loan_date DESC"

    cursor.execute(query, params)
    loans = cursor.fetchall()

    cursor.close()
    conn.close()

    return render_template('my_loans.html', loans=loans, status_filter=status_filter)

# @app.route('/admin/user/<int:user_id>')
# def admin_user_loans(user_id):
#     if not session.get('user_id') or session.get('user_role') != 'admin':
#         return redirect(url_for('login'))

#     conn = get_db_connection()
#     cursor = conn.cursor(dictionary=True)

#     cursor.execute("SELECT name FROM users WHERE id = %s", (user_id,))
#     user = cursor.fetchone()

#     if not user:
#         flash('Ο φοιτητής δεν βρέθηκε.', 'danger')
#         return redirect(url_for('admin_dashboard'))

#     cursor.execute("""
#         SELECT l.*, b.title AS book_title, b.author
#         FROM loans l
#         JOIN books b ON l.book_id = b.id
#         WHERE l.user_id = %s
#         ORDER BY l.loan_date DESC
#     """, (user_id,))
#     loans = cursor.fetchall()

#     cursor.close()
#     conn.close()

#     return render_template('admin_user_loans.html', loans=loans, student=user)

@app.route('/admin/user/<int:user_id>/export')
def export_user_loans_csv(user_id):
    if not session.get('user_id') or session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("SELECT name FROM users WHERE id = %s", (user_id,))
    user = cursor.fetchone()
    if not user:
        flash('Ο φοιτητής δεν βρέθηκε.', 'danger')
        return redirect(url_for('admin_dashboard'))

    cursor.execute("""
        SELECT b.title, b.author, l.loan_date, l.return_date, l.actual_return, l.status
        FROM loans l
        JOIN books b ON l.book_id = b.id
        WHERE l.user_id = %s
        ORDER BY l.loan_date DESC
    """, (user_id,))
    loans = cursor.fetchall()
    cursor.close()
    conn.close()

    si = StringIO()
    writer = csv.writer(si)
    writer.writerow(['Τίτλος', 'Συγγραφέας', 'Ημ. Αίτησης', 'Ημ. Επιστροφής', 'Πραγματική Επιστροφή', 'Κατάσταση'])
    for loan in loans:
        writer.writerow([
            loan['title'],
            loan['author'],
            loan['loan_date'].strftime('%d-%m-%Y'),
            loan['return_date'].strftime('%d-%m-%Y'),
            loan['actual_return'].strftime('%d-%m-%Y') if loan['actual_return'] else '',
```

```
            loan['status']
        ])

    output = si.getvalue()
    return Response(
        output,
        mimetype='text/csv',
        headers={'Content-Disposition': f'attachment; filename=loans_{user["name"]}.csv'}
    )

@app.route('/return-loan/<int:loan_id>')
def return_loan(loan_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Ενημέρωση κατάστασης στον πίνακα loans
    cursor.execute("""
        UPDATE loans
        SET status = 'returned', actual_return = %s
        WHERE id = %s AND status = 'borrowed'
    """, (date.today(), loan_id))

    if cursor.rowcount == 0:
        flash('Ο δανεισμός δεν είναι ενεργός ή δεν βρέθηκε.', 'warning')
    else:
        # Ενημέρωση διαθέσιμων αντιτύπων
        cursor.execute("""
            UPDATE books
            SET available_copies = available_copies + 1
            WHERE id = (SELECT book_id FROM loans WHERE id = %s)
        """, (loan_id,))

        # Προσθήκη ειδοποίησης
        cursor.execute("""
            INSERT INTO notifications (user_id, message)
            SELECT user_id, CONCAT('Η επιστροφή του βιβλίου "', b.title, '" καταχωρήθηκε.')
            FROM loans l
            JOIN books b ON l.book_id = b.id
            WHERE l.id = %s
        """, (loan_id,))

        flash('Η επιστροφή καταχωρήθηκε.', 'success')

    conn.commit()
    cursor.close()
    conn.close()
    return redirect(url_for('admin_requests'))

@app.route('/admin/books')
def admin_books():
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    cursor.execute("""
        SELECT b.*, c.name AS category_name
        FROM books b
        JOIN categories c ON b.category_id = c.id
        ORDER BY b.title ASC
    """)
    books = cursor.fetchall()

    cursor.execute("SELECT * FROM categories ORDER BY name")
    categories = cursor.fetchall()

    cursor.close()
    conn.close()

    return render_template('admin_books.html', books=books, categories=categories)

@app.route('/admin/books/add', methods=['POST'])
def admin_books_add():
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    title = request.form['title']
    author = request.form['author']
    isbn = request.form['isbn']
    category_id = request.form['category_id']
    copies = int(request.form['copies'])
```

```
    description = request.form['description']

    conn = get_db_connection()
    cursor = conn.cursor()
    cursor.execute("""
        INSERT INTO books (title, author, isbn, category_id, total_copies, available_copies, description)
        VALUES (%s, %s, %s, %s, %s, %s, %s)
    """, (title, author, isbn, category_id, copies, copies, description))

    conn.commit()
    conn.close()

    flash('Το βιβλίο προστέθηκε.', 'success')
    return redirect(url_for('admin_books'))

@app.route('/admin/books/delete/<int:book_id>')
def admin_books_delete(book_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Πρώτα έλεγχος για ενεργούς δανεισμούς
     cursor.execute("SELECT COUNT(*) FROM loans WHERE book_id = %s AND status IN ('pending', 'borrowed', 'late')",
(book_id,))
    if cursor.fetchone()[0] > 0:
        conn.close()
        flash('Το βιβλίο δεν μπορεί να διαγραφεί γιατί υπάρχουν ενεργοί δανεισμοί.', 'danger')
        return redirect(url_for('admin_books'))

    cursor.execute("DELETE FROM books WHERE id = %s", (book_id,))
    conn.commit()
    conn.close()
    flash('Το βιβλίο διαγράφηκε.', 'info')
    return redirect(url_for('admin_books'))

@app.route('/admin/books/edit/<int:book_id>', methods=['GET', 'POST'])
def admin_books_edit(book_id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    if request.method == 'POST':
        title = request.form['title']
        author = request.form['author']
        isbn = request.form['isbn']
        category_id = request.form['category_id']
        description = request.form['description']
        total_copies = int(request.form['total_copies'])

        # Ενημέρωση συνολικών αντίτυπων – ΔΕΝ αλλάζουμε διαθέσιμα για να μη χαθούν δανεισμοί
        cursor.execute("""
            UPDATE books
            SET title = %s, author = %s, isbn = %s, category_id = %s,
                description = %s, total_copies = %s
            WHERE id = %s
        """, (title, author, isbn, category_id, description, total_copies, book_id))

        conn.commit()
        flash('Το βιβλίο ενημερώθηκε.', 'success')
        conn.close()
        return redirect(url_for('admin_books'))

    # GET – φέρνουμε τα δεδομένα
    cursor.execute("SELECT * FROM books WHERE id = %s", (book_id,))
    book = cursor.fetchone()

    cursor.execute("SELECT * FROM categories ORDER BY name")
    categories = cursor.fetchall()

    cursor.close()
    conn.close()

    if not book:
        flash('Το βιβλίο δεν βρέθηκε.', 'danger')
        return redirect(url_for('admin_books'))

    return render_template('admin_books_edit.html', book=book, categories=categories)

@app.route('/admin/categories', methods=['GET', 'POST'])
def admin_categories():
    if session.get('user_role') != 'admin':
```

```
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    if request.method == 'POST':
        name = request.form['name'].strip()
        if name:
            try:
                cursor.execute("INSERT INTO categories (name) VALUES (%s)", (name,))
                conn.commit()
                flash('Η κατηγορία προστέθηκε.', 'success')
            except:
                flash('Η κατηγορία υπάρχει ήδη.', 'warning')

    cursor.execute("SELECT * FROM categories ORDER BY name")
    categories = cursor.fetchall()

    cursor.close()
    conn.close()
    return render_template('admin_categories.html', categories=categories)

@app.route('/admin/categories/delete/<int:id>')
def admin_delete_category(id):
    if session.get('user_role') != 'admin':
        return redirect(url_for('login'))

    conn = get_db_connection()
    cursor = conn.cursor()

    # Έλεγχος αν χρησιμοποιείται
    cursor.execute("SELECT COUNT(*) FROM books WHERE category_id = %s", (id,))
    if cursor.fetchone()[0] > 0:
        flash('Η κατηγορία δεν μπορεί να διαγραφεί. Υπάρχουν βιβλία που την χρησιμοποιούν.', 'danger')
        conn.close()
        return redirect(url_for('admin_categories'))

    cursor.execute("DELETE FROM categories WHERE id = %s", (id,))
    conn.commit()
    conn.close()
    flash('Η κατηγορία διαγράφηκε.', 'info')
    return redirect(url_for('admin_categories'))

@app.route('/notifications')
def notifications():
    if not session.get('user_id'):
        flash('Πρέπει να συνδεθείτε.', 'warning')
        return redirect(url_for('login'))

    user_id = session['user_id']
    conn = get_db_connection()
    cursor = conn.cursor(dictionary=True)

    # Παίρνουμε όλες τις ειδοποιήσεις
    cursor.execute("""
        SELECT id, message, created_at, seen
        FROM notifications
        WHERE user_id = %s
        ORDER BY created_at DESC
    """, (user_id,))
    notes = cursor.fetchall()

    # Μαρκάρουμε ως διαβασμένες
    cursor.execute("UPDATE notifications SET seen = 1 WHERE user_id = %s", (user_id,))
    conn.commit()
    cursor.close()
    conn.close()

    return render_template('notifications.html', notes=notes)

@app.context_processor
def inject_unseen_count():
    if 'user_id' in session:
        conn = get_db_connection()
        cursor = conn.cursor()
        cursor.execute("SELECT COUNT(*) FROM notifications WHERE user_id = %s AND seen = 0", (session['user_id'],))
        count = cursor.fetchone()[0]
        cursor.close()
        conn.close()
        return {'unseen_notifications': count}
    return {}

if __name__ == '__main__':
    app.run(debug=True)
```