

Bilbo Εφαρμογή Διαχείρισης Βιβλιοθηκών

Το παρόν έγγραφο είναι εργασία για το μάθημα Προγραμματισμός Διαδικτύου 8^ο εξάμηνο του τμήματος Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών του Πανεπιστημίου Πατρών.

Κωνσταντίνος Κοτορένης

Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Πανεπιστήμιο Πατρών, up1053750@upnet.gr

Νικόλαος Φιλιππάτος

Ηλεκτρολόγων Μηχανικών και Τεχνολογίας Υπολογιστών Πανεπιστήμιο Πατρών, filippatos.nikolaos@upnet.gr

Το Bilbo είναι μια εφαρμογή διαχείρισης βιβλιοθηκών. Σε αυτή τη πλατφόρμα οι βιβλιοθήκες μπορούν να διαθέσουν τα βιβλία τους προς ενοικίαση και οι χρήστες της εφαρμογής μπορούν να αναζητήσουν βιβλία, να δουν την διαθεσιμότητά τους, σε ποιες βιβλιοθήκες είναι διαθέσιμα και τέλος να πραγματοποιήσουν δανεισμούς των βιβλίων αυτών. Το Bilbo είναι εμπνευσμένο από το skrouz.gr, μία ελληνική ιστοσελίδα εύρεσης καταστημάτων και προϊόντων, καθώς και πραγματοποίησης online αγορών. Στην εργασία αυτή χρησιμοποιήθηκαν οι εξής γλώσσες προγραμματισμού: HTML, CSS, JavaScript, SQLite και Python. Επίσης χρησιμοποιήθηκαν τα frameworks: Handlebars, Express.js. Χρησιμοποιήθηκε το runtime: Node.js. Για τις υπηρεσίες server χρησιμοποιήθηκε το Heroku. Τέλος για UI/UX Design χρησιμοποιήθηκε το Figma.

CCS CONCEPTS • Software and its engineering • Software creation and management • Designing software • Software design engineering

1 ΕΙΣΑΓΩΓΗ

Η δημιουργία μιας εφαρμογής διαχείρισης βιβλιοθηκών ήταν μία πρόκληση γιατί στην αρχή δεν είχαμε δημιουργικές ιδέες πάνω στο θέμα. Για να μπορέσουμε να προσπεράσουμε αυτό το εμπόδιο κάναμε μία χειρά από συναντήσεις όπου καταγράφαμε ιδέες πάνω στο αντικείμενο. Ενδιάμεσα σε αυτές τις συναντήσεις ο καθένας μας έκανε έρευνα να βρούμε τι ήδη υπάρχει στην Ελληνική αγορά σχετικά με το αντικείμενο. Τελικά καταλήξαμε να κάνουμε μια ιστοσελίδα, το Bilbo, η οποία είναι μία μηχανή αναζήτησης βιβλίων διαθέσιμα προς δανεισμό από βιβλιοθήκες. Η ιστοσελίδα αυτή είναι εμπνευσμένη από το skrouz.gr και ο στόχος της είναι να διευκολύνει την εύρεση βιβλίων διαθέσιμα προς δανεισμό από τοπικές βιβλιοθήκες στον τόπο στον οποίο βρίσκετε ο χρήστης.

1.1 Έμπνευση

Η διαδικασία που ακολουθήσαμε για να σχεδιάσουμε την εφαρμογή μας ήταν η ακόλουθη. Αρχικά κάναμε έρευνα να δούμε τι εφαρμογές υπάρχουν ήδη στο διαδίκτυο που έχουν παρόμοιο σκοπό με την εφαρμογή που είχαμε να σχεδιάσουμε. Οι πλατφόρμες που βρήκαμε μαζί με τα μειονεκτήματά που εντοπίσαμε είναι οι παρακάτω:

1. opac.seab.gr

Ο χρήστης για να βρει ένα βιβλίο που θέλει πρέπει να ψάξει μία μία τις βιβλιοθήκες που είναι διαθέσιμες. Δεν δείχνει όλες τις βιβλιοθήκες που ένα βιβλίο είναι διαθέσιμο.

2. www.lib.uoa.gr

Η σελίδα έχει πολλά διαφορετικά περιβάλλοντα χρήστη και δεν είναι άμεσα κατανοητό που μπορεί ο χρήστης να βρει ένα βιβλίο που έχει εμφανιστεί στην αναζήτηση.

3. hzeephyr.lib.uoc.gr

Η σελίδα έχει το ίδιο πρόβλημα με τη πρώτη, πρέπει να ξέρει ο χρήστης τη βιβλιοθήκη στην οποία πρέπει να ψάξει το βιβλίο. Επίσης ο χρήστης πρέπει να γράψει ακριβώς το όνομα του βιβλίου για να το εντοπίσει.

4. search.lib.auth.gr

Η σελίδα των βιβλιοθηκών του ΑΠΘ είχε πολλά χαρακτηριστικά τα οποία θεωρήσαμε απαραίτητα για την εφαρμογή μας. Το κύριο μειονέκτημα αυτής της σελίδας είναι ότι στον χρήστη δεν προτείνονται βιβλία και για τις τοποθεσίες των βιβλιοθηκών δεν υπάρχει κάποιος χάρτης.

Στη συνέχεια προσπαθήσαμε να δούμε τον σκοπό αυτής της εφαρμογής πιο σφαιρικά και να ερευνήσουμε πως άλλοι κλάδοι δανεισμών-αγορών χειρίζονται πολλά καταστήματα και προϊόντα. Η πλατφόρμα που εστιάσαμε κυρίως είναι skrutz.gr, μία ελληνική πλατφόρμα διαχείρισης καταστημάτων και προϊόντων. Τα χαρακτηριστικά που μας κίνησαν το ενδιαφέρον ήταν η προβολή συστάσεων για προϊόντα, η αναζήτηση που έχει, τα φίλτρα και το προφίλ χρήστη για τη παρακολούθηση των παραγγελιών. Μία άλλη πηγή έμπνευσής μας ήταν η προβολή των βιβλιοθηκών σε έναν χάρτη όπως είναι στο eestec.net.

Αφού είδαμε τα προτερήματα και τα μειονεκτήματα των παραπάνω σελίδων δημιουργήσαμε μία λίστα με πιθανές δυνατότητες που θα μπορούσε να έχει η δική μας σελίδα και έπειτα αποφασίσαμε ποιες από αυτές τις δυνατότητες θα ήταν εφικτό να υλοποιήσουμε στα πλαίσια του μαθήματος.

Τέλος εμπνευσμένοι από το skrutz.gr θέλαμε η μάρκα της ιστοσελίδας μας να συμπεριλαμβάνει έναν χαρακτήρα ο οποίος ήταν βιβλιοφάγος. Έτσι καταλήξαμε στον Bilbo, ένας βιβλιοφάγος χαρακτήρας από την τριλογία του Άρχοντα των Δαχτυλιδιών

1.2 Σχεδιασμός

Για τον σχεδιασμό της ιστοσελίδας μας χρησιμοποιήθηκε το Figma ([link](#)). Στη διαδικασία του σχεδιασμού αρχικά δημιουργήθηκαν wireframes και στη συνέχεια έγιναν τα mockups των σελίδων. Οι σελίδες που σχεδιάστηκαν είναι οι ακόλουθες:

1. Home Page
2. Search
3. Library Info
4. Book Info
5. Library Profile (Admin)
6. User Profile
7. About us

Για την δημιουργία της μάρκας όπως και στο skrutz.gr χρησιμοποιήσαμε το όνομα ενός διάσημου βιβλιοφάγου χαρακτήρα, του Bilbo. Στο λογότυπο της ιστοσελίδας μπορεί κανείς να δει μια φιγούρα του Bilbo να τρέχει πάνω σε ένα ανοιχτό βιβλίο. Το χρώμα που χρησιμοποιήσαμε είναι αποχρώσεις του αμέθυστου.

Για τον σχεδιασμό της εφαρμογής εστιάσαμε στο πως θα λύσουμε τα προβλήματα που βρήκαμε στις ιστοσελίδες που παρουσιάστηκαν στο κεφάλαιο 1.1. Πιο συγκεκριμένα ο σχεδιασμός της σελίδας μας τοποθετεί την αναζήτηση να είναι

το κυρίαρχο στοιχείο όταν ο χρήστης μπαίνει στη σελίδα μας. Ο χρήστης μπορεί επίσης να δει προτάσεις για βιβλία και βιβλιοθήκες. Στην ιδανική έκδοση της εφαρμογής μας ο χρήστης μπορεί να έχει προσωποποιημένες προτάσεις για βιβλία και βιβλιοθήκες που είναι κοντά του. Ο χρήστης δεν χρειάζεται να ξέρει από ποια βιβλιοθήκη θέλει ένα βιβλίο, μπορεί πρώτα να ψάξει για ένα βιβλίο και μετά να επιλέξει από ποια βιβλιοθήκη θα το δανειστεί. Ο χρήστης αφού κάνει μία αναζήτηση μπορεί να επιλέξει φίλτρα για να περιορίσει το αποτέλεσμα της αναζήτησης. Αφού ο χρήστης επιλέξει το βιβλίο που θέλει μπορεί να δει σε ποιες βιβλιοθήκες είναι διαθέσιμο το βιβλίο που έχει επιλέξει και μπορεί να κάνει κράτηση για να το δανειστεί με ένα κουμπί που βρίσκεται δίπλα από τη βιβλιοθήκη. Επίσης ο χρήστης μπορεί να δει όλες τις βιβλιοθήκες να προβάλλονται σε έναν χάρτη για να είναι πιο εύκολα διακριτές από τον χρήστη.

Στις υπόλοιπες σελίδες δεν σχεδιάσαμε για να βελτιώσουμε τις ιστοσελίδες της έρευνας αλλά με βάση κάποιες δικές μας ιδέες που θεωρήσαμε ενδιαφέρουσες. Στο προφίλ του χρήστη, ο χρήστης μπορεί να δει τη κάρτα δανεισμού του η οποία θα λειτουργεί σε κάθε βιβλιοθήκη καθώς και την κατάσταση του δανεισμού του• ότι έκανε τη κράτηση, έκανε το δανεισμό, τότε πρέπει να επιστρέψει το βιβλίο και αν έχει ξεπεράσει το όριο δανεισμού. Στη σελίδα της βιβλιοθήκης ο χρήστης μπορεί να δει πληροφορίες για τη βιβλιοθήκη, έναν χάρτη ο οποίος δείχνει που βρίσκεται η βιβλιοθήκη, τις ώρες που είναι ανοιχτή και κάποια προτεινόμενα βιβλία. Η βιβλιοθήκη έχει τη δυνατότητα να επεξεργαστεί αυτές τις πληροφορίες και να προσθέσει και να αφαιρέσει βιβλία.

1.3 Παραδοχές

Στο επόμενο στάδιο της υλοποίησης αποφασίσαμε ποιες δυνατότητες από αυτές που σχεδιάσαμε θα υλοποιήσουμε στα πλαίσια του μαθήματος. Στην τελική μας υλοποίηση δημιουργούμε την σελίδα από τη μεριά ενός χρήστη που δεν έχει λογαριασμό και ενός χρήστη που έχει. Δεν υλοποιήσαμε τη διαχείριση των βιβλίων από τη μεριά της βιβλιοθήκης καθώς και τη διαχείριση των δανεισμών.

Τα σημεία που είναι λειτουργικά είναι η αναζήτηση βιβλίων με κομμάτια από τον τίτλο ενός βιβλίου και η προσθήκη φίλτρων για περιορισμό των αποτελεσμάτων. Ο χρήστης μπορεί να δει πλήρως τις σελίδες των βιβλίων, των βιβλιοθηκών και του προφίλ χρήστη. Πιο συγκεκριμένα στο προφίλ χρήστη φαίνεται η κατάσταση δανεισμού του και στη σελίδα των βιβλίων μπορεί να πραγματοποιήσει μία κράτηση. Ο χρήστης μπορεί να δει προτεινόμενα βιβλία και βιβλιοθήκες. Τέλος ένας νέος χρήστης έχει τη δυνατότητα να εγγραφεί στη σελίδα μας

Τα σημεία που δεν υλοποιήσαμε είναι το Admin Panel της κάθε βιβλιοθήκης. Αυτό σημαίνει ότι ένας χρήστης όταν κάνει έναν δανεισμό δεν μπορεί να προχωρήσει στον δανεισμό του ή την επιστροφή εκτός και αν αυτή η πληροφορία έχει γραφτεί στη βάση δεδομένων από πριν. Οι προτάσεις βιβλίων και βιβλιοθηκών είναι απλά τα πρώτα αποτελέσματα που βρίσκει το πρόγραμμα στη βάση και δεν υπάρχει κάποιος αλγόριθμος προτεινόμενων εφόσον αυτό ξεφεύγει από τα πλαίσια του μαθήματος. Τέλος ο χρήστης δεν έχει δυνατότητα να αλλάξει τις πληροφορίες του.

1.4 Μεθοδολογία

Για την υλοποίηση αυτής της εργασίας είχαμε μία καθιερωμένη συνάντηση κάθε βδομάδα. Σε αυτή τη συνάντηση συζητούσαμε τι είχε γίνει μέχρι τότε, καθορίζαμε ποια ήταν τα επόμενα μας βήματα και τα βάζαμε σε ένα χρονικό πλαίσιο. Επίσης σε κάθε συνάντηση γινόταν αξιολόγηση των λειτουργιών που είχαν υλοποιηθεί και κάναμε ανάλογες αλλαγές όπου χρειαζόταν. Σε αυτές τις συναντήσεις συχνά δουλεύαμε με τη τεχνική του pair programming και μετά χωρίζαμε εργασίες για να υλοποιήσει ο καθένας στον χρόνο που είχε μέσα στην εβδομάδα. Παρακάτω μπορείτε να δείτε έναν γενικό καταμερισμό των εργασιών. Ο καταμερισμός αυτός δείχνει κυρίως ποιος ήταν υπεύθυνος για κάθε κομμάτι βέβαια και οι δύο συμβάλουμε στα κομμάτια του άλλου.

1.5 Κύριες Ενέργειες

- Σχεδιασμός της ιστοσελίδας στο Figma (Κοτορένης)
- Σχεδίαση Βάσης Δεδομένων (Κοτορένης, Φιλιππάτος)
- Προγραμματισμός ιστοσελίδων:
 - User profile, Book info, About us (Κοτορένης)
 - Homepage, Search Page, Library Info (Φιλιππάτος)
- Δημιουργία χαρτών (Φιλιππάτος)
- Mobile view (Κοτορένης)
- Δημιουργία φίλτρων (Φιλιππάτος)
- Κατασκευή Βάσης Δεδομένων (Φιλιππάτος)
- Αλληλεπίδραση με την Βάση Δεδομένων (Κοτορένης, Φιλιππάτος)
- Server set up & deployment to Heroku (Φιλιππάτος)

1.6 Χρονοδιάγραμμα

11/3 - 17/3	Μικρόκοσμος, επιλογή αναγκαίων σελίδων
18/3 - 24/3	Δημιουργία των wireframes και αρχική μερική υλοποίηση σε HTML και CSS
25/3 - 31/3	Ολοκλήρωση του τελικού σχεδιασμού της εφαρμογής, μετατροπή του σε html-css-javascript κώδικα
3/4	Ενδιάμεση παρουσίαση
4/4 - 21/4	Ετοιμασία του server με την χρήση node express , μετατροπή των HTML αρχείων σε handlebars
22/4-29/4	Δημιουργία βάσης δεδομένων
30/4-6/5	Δημιουργία συναρτήσεων και query για αλληλεπίδραση με την βάση
7/5 - 13/5	Έτοιμες οι περισσότερες σελίδες , εξερεύνηση των επίλογων για website hosting, ολοκλήρωση φίλτρων, αναζήτησης και mobile view
14/5 - 20/5	Διαμόρφωση της ιστοσελίδας για deployment στο Heroku, ολοκλήρωση δανεισμών
21/5 - 26/5	Ολοκλήρωση του κώδικα, συγγραφή της αναφοράς

2 ΥΛΟΠΟΙΗΣΗ

Για την υλοποίηση της εργασίας μας χρησιμοποιήσαμε τις ακόλουθες τεχνολογίες: Node.JS , Express-Handlebars, CSS, JavaScript, HTML και για τη βάση δεδομένων την SQLite.

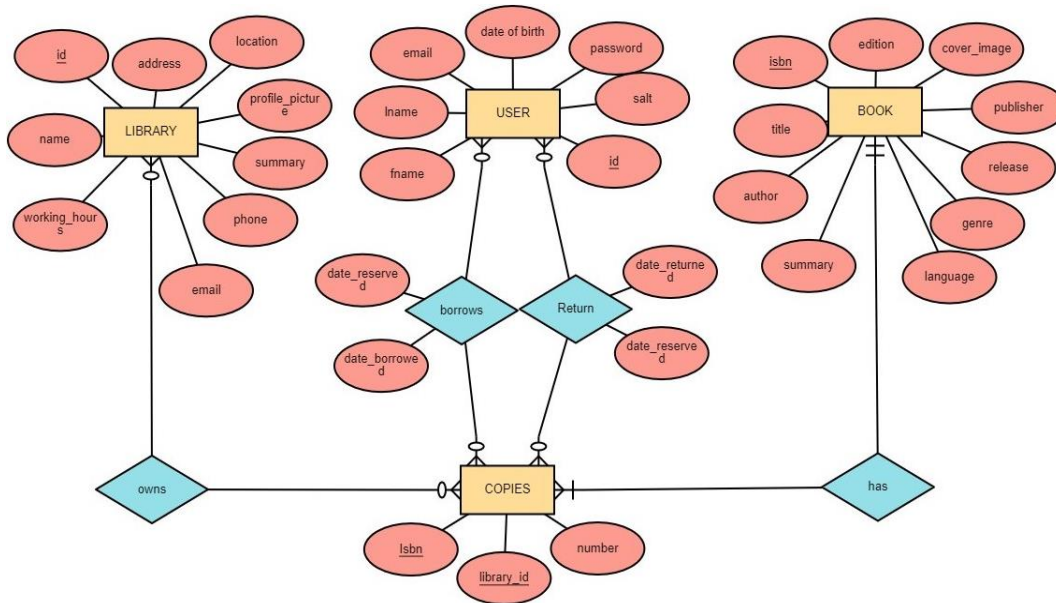
Σε αυτή την ενότητα θα αναλύσουμε τις διαδικασίες που χρησιμοποιήσαμε για την πρακτική υλοποίηση της εργασίας

2.1 Σχεδιασμός της Βάσης Δεδομένων

Για τον σχεδιασμό της Βάσης Δεδομένων της ιστοσελίδας έγινε ανάλυση του σχεδίου που είχε γίνει στο Figma και έγινε ομαδοποίηση και κατηγοριοποίηση των δεδομένων που ήταν απαραίτητα για την υλοποίηση της ιστοσελίδας. Στη συνέχεια σχεδιάστηκε το Σχεσιακό Μοντέλο Οντοτήτων (ERD) καθώς και Σχήμα της βάσης δεδομένων.

2.1.1 Σχεσιακό Μοντέλο Οντοτήτων (ERD).

Στην εικόνα 2.1 μπορεί κανείς να διακρίνει το Σχεσιακό Μοντέλο Οντοτήτων. Αυτό το σχέδιο αφορά μόνο την τελική υλοποίηση της ιστοσελίδας μετά τις παραδοχές που αναφέρονται στην ενότητα 1.3.

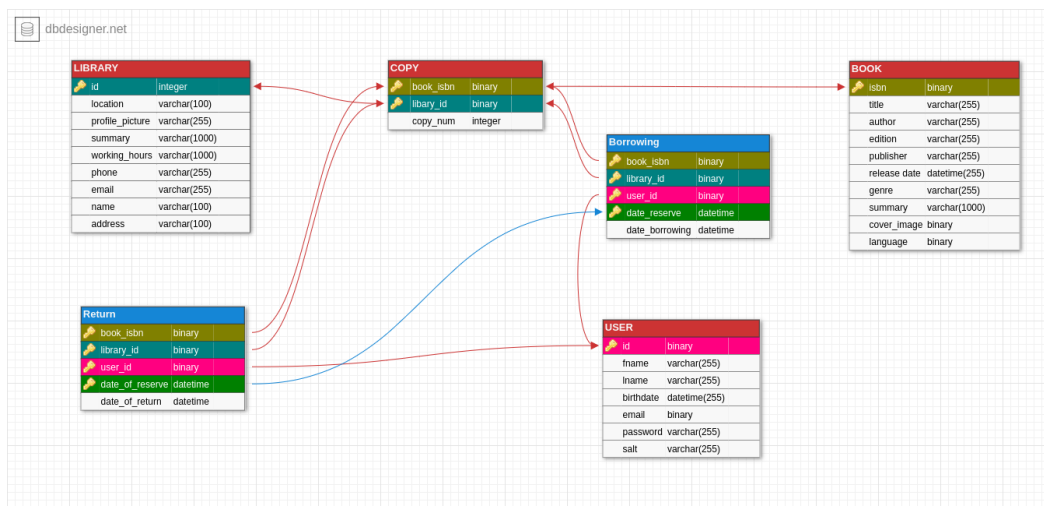


Εικόνα 2.1: Σχεσιακό Μοντέλο Οντοτήτων Βάσης Δεδομένων της ιστοσελίδας Bilbo

Στο σχεσιακό μοντέλο μπορεί κανείς να διακρίνει τέσσερις οντότητες: Users, Library, Books και Copies. Ένας χρήστης (User) μπορεί να δανειστεί ή να επιστρέψει ένα αντίγραφο (Copy) από ένα βιβλίο (Book) που βρίσκεται σε μία βιβλιοθήκη (Library).

2.1.2 Σχήμα της Βάσης Δεδομένων.

Μετά τον σχεδιασμό του Σχεσιακού Μοντέλου προχωρήσαμε στη δημιουργία του σχήματος της Βάσης Δεδομένων της ιστοσελίδας (εικόνα 2.2). Οι σχέσεις borrows και returns έχουν γίνει πίνακες όπως και όλες οι οντότητες.



Εικόνα 2.2: Σχήμα Βάσης Δεδομένων της ιστοσελίδας Bilbo στο DBdesigner

2.1.3 Κατασκευή της Βάσης Δεδομένων.

Για τη κατασκευή της Βάσης Δεδομένων δουλέψαμε με την τεχνολογία SQLite. Για την δημιουργία και το γέμισμα της βάσης, αξιοποιήθηκε ένα πρόγραμμα γραμμένο σε Python (model/database_managing.py) .

Πιο συγκεκριμένα, από το DBdesigner κάναμε εξαγωγή της βάσης σε ένα αρχείο SQL, επεξεργαστήκαμε τα δεδομένα του και το αποθηκεύσαμε ως «model/dbdesigner.sql». Το αρχείο «database_managing.py» ανοίγει το αρχείο «dbdesigner.sql» και εκτελεί όλες τις απαραίτητες εντολές για την δημιουργία της Βάσης Δεδομένων.

2.1.4 Δεδομένα της Βάσης Δεδομένων.

Για τη Βάση Δεδομένων χρειάζομασταν 3 ειδών δεδομένα• για τους χρήστες, τις βιβλιοθήκες, και τα βιβλία:

- Για τους χρήστες δημιουργήσαμε εμείς τα δεδομένα.
- Για τις βιβλιοθήκες κάναμε αναζήτηση στο ίντερνετ και καταχωρήσαμε 4 βιβλιοθήκες: την Εθνική βιβλιοθήκη της Αθήνας, την δημοτική βιβλιοθήκη της Πάτρας, την βιβλιοθήκη του Πανεπιστημίου Πατρών και την δημοτική βιβλιοθήκη της Ναύπακτου.
- Για τα βιβλία αξιοποιήσαμε ένα API της Google, στο οποίο βάζοντας σαν πληροφορία έναν τίτλο ή ISBN επιστρέφει τις πληροφορίες για αυτό το βιβλίο, και κοντινά αποτελέσματα σε μορφή JSON files.
<https://www.googleapis.com/books/v1/volumes?q=title:>

2.1.5 Φόρτωση των δεδομένων της Βάσης Δεδομένων.

Το πρόγραμμα «database_managing.py» κοιτάει άμα υπάρχει η βάση δεδομένων και άμα ο φάκελος «bookdata» δεν είναι άδειος .

Εάν ο φάκελος «bookdata» είναι άδειος, κατεβάζει όλες τις πληροφορίες από την λίστα των τίτλων (titles.py) από το <https://www.googleapis.com/books/v1/volumes?q=title:> και τις αποθηκεύει σε JSON αρχεία. Οι τίτλοι έχουν επιλεχθεί από μια αναζήτηση για τους 100 πιο διάσημους τίτλους, και αποτελούν ένα δείγμα.

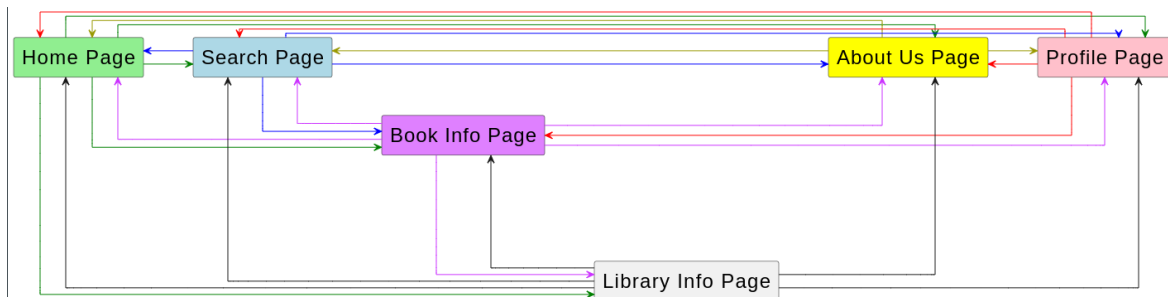
Στη συνέχεια κοιτάζει άμα υπάρχει η Βάση Δεδομένων «bilboData.sqlite», καθαρίζει τα δεδομένα που έχει μέσα, και ξεκινάει πάλι την διαδικασία γεμίσματος.

Η διαδικασία γεμίσματος της Βάσης Δεδομένων φορτώνει τα στοιχεία των βιβλίων από τα αρχεία που βρίσκονται μέσα στον φάκελο «bookdata» σε μια λίστα λεξικών. Έπειτα θα κοιτάζει τα αρχεία libraries.py για τις πληροφορίες των βιβλιοθηκών, και γεμίζει τον πίνακα «Copies» με τυχαία δεδομένα, αξιοποιώντας την βιβλιοθήκη «itertools» και «random» της Python. Τέλος γεμίζει τους πίνακες «borrowing» και «return» από τα αντίστοιχα αρχεία (borrowing.py, return.py).

2.2 Διαδρομές

Η εφαρμογή μας έχει 6 σελίδες στις οποίες μπορεί να πάει ένας χρήστης

- **Homepage**
Είναι η αρχική σελίδα της εφαρμογής, προβάλλει μια λίστα με βιβλία που είναι πιο δημοφιλή (προς το παρόν η λίστα είναι στατική και δείχνει απλώς 12 βιβλία από την βάση) καθώς και κάποιες από τις βιβλιοθήκες.
- **Search Page**
Αφού ο χρήστης κάνει κάποια αναζήτηση τίτλου, μπορεί να εφαρμόσει φίλτρα και να δει τα αποτελέσματα της αναζήτησης του.
- **Profile Page (εφόσον είναι συνδεδεμένος)**
Εδώ ο χρήστης μπορεί να δει τις πληροφορίες του, και τα βιβλία που έχει κρατήσει, δανειστεί και χρωστάει να επιστρέψει, καθώς και το ιστορικό του.
- **About us Page**
Δίνουμε πληροφορίες για το ποια είναι η ιστοσελίδα και οι δημιουργοί της.
- **Book Info page**
Προβάλλει πληροφορίες για το κάθε βιβλίο, καθώς και σε ποιες βιβλιοθήκες μπορεί κάποιος να το δανειστεί
- **Library Page**
Προβάλλονται πληροφορίες για την κάθε βιβλιοθήκη, και ορισμένα βιβλία που έχει η βιβλιοθήκη.

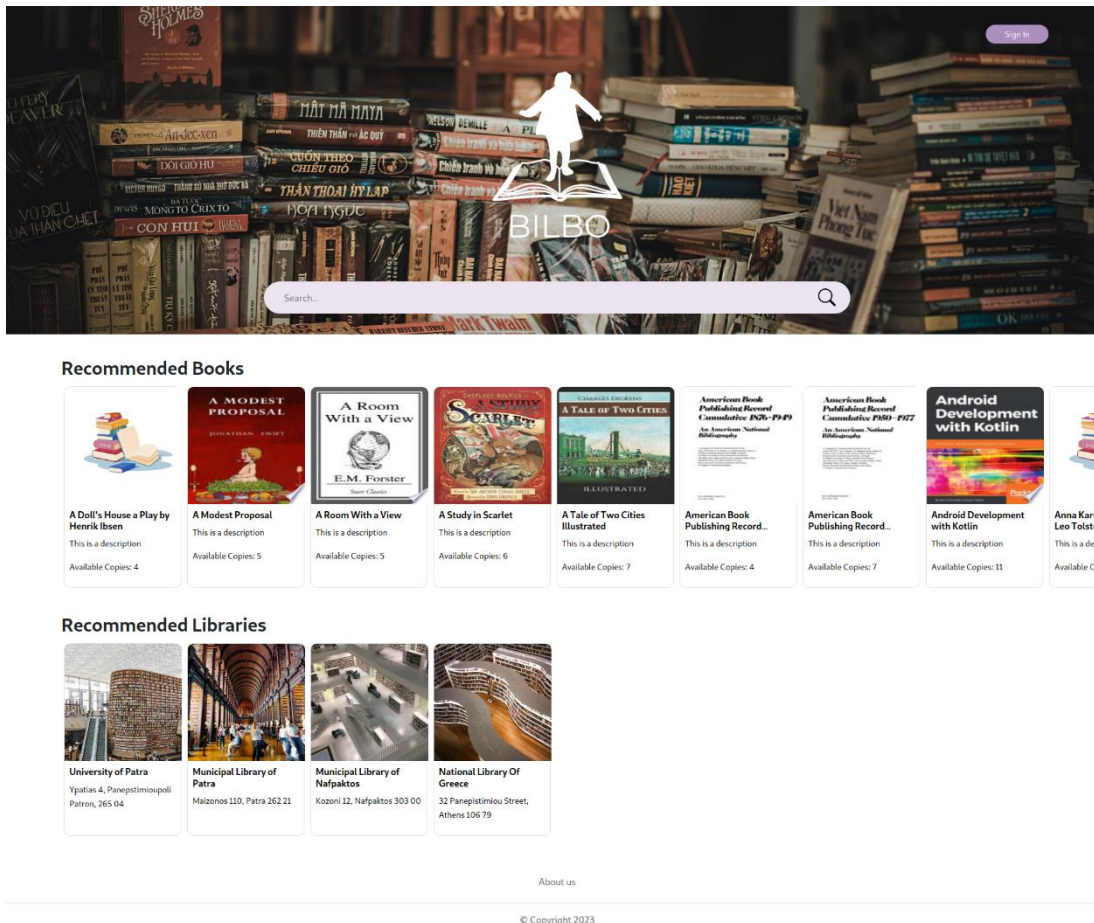


Εικόνα 2.3: Οι σελίδες της ιστοσελίδας Bilbo και οι διαδρομές που μπορεί να ακολουθήσει ένας χρήστης

2.3 Χρήση της εφαρμογής

Σε αυτό το σημείο θα παρουσιάσουμε πιο αναλυτικά τις χρήσεις της ιστοσελίδας που υλοποιήθηκαν.

2.3.1 Homepage

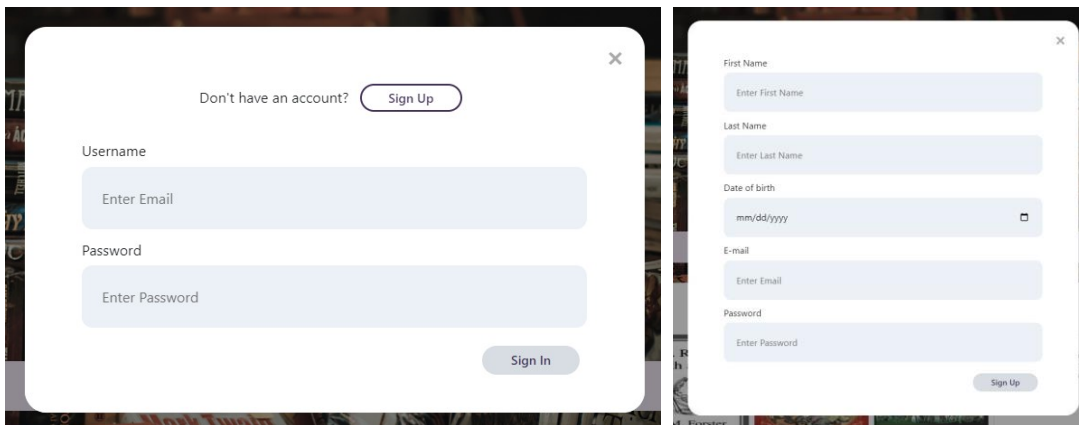


Εικόνα 2.4: Αρχική σελίδα της ιστοσελίδας Bilbo

Από την αρχική σελίδα (εικόνα 2.4) ο χρήστης μπορεί να δει ορισμένα προτεινόμενα βιβλία και βιβλιοθήκες (προς το παρόν η λίστα που εμφανίζεται είναι στατική). Σε κάθε σελίδα ο χρήστης έχει την δυνατότητα να συνδεθεί ή να εγγραφεί και να αναζητήσει βιβλία. Πιο συγκεκριμένα στην αρχική σελίδα η αναζήτηση παρουσιάζεται σαν τη κύρια και πιο σημαντική ενέργεια που μπορεί να κάνει ο χρήστης βάζοντας την περιοχή αναζήτησης όσο πιο κεντρικά γίνεται.

2.3.2 Αναδυόμενα Παράθυρα

Ο χρήστης μπορεί να συνδεθεί στον λογαριασμό του ή να εγγραφεί στην ιστοσελίδα μέσα από αναδυόμενα παράθυρα στα οποία έχει πρόσβαση από κάθε σελίδα. Οι κωδικοί του χρήστη αποθηκεύονται κρυπτογραφημένοι στην βάση, με την βιβλιοθήκη bcrypt. Ταυτόχρονα έχουμε πάρει μέτρα για την αποφυγή SQL injection.



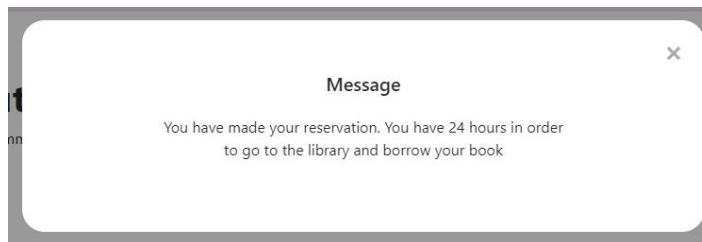
Εικόνα 2.5 – 2.6: Αναδυόμενο παράθυρο σύνδεσης στο λογαριασμό χρήστη/ εγγραφής χρήστη στην ιστοσελίδα

Στην περίπτωση που ο χρήστης δεν υπάρχει ή έβαλε λάθος κωδικό, εμφανίζεται το κατάλληλο αναδυόμενο μήνυμα λάθους έτσι ώστε ο χρήστης να ξέρει ότι έκανε λάθος τον κωδικό του ή το e-mail του. Εφόσον συνδεθεί σωστά, μπορεί να δει το προφίλ του.



Εικόνα 2.7 – 2.8: Αναδυόμενο παράθυρο μηνύματος ο χρήστης δεν βρέθηκε, ο κωδικός είναι λάθος

Τελευταίο αναδυόμενο μήνυμα είναι όταν ο χρήστης κάνει μία κράτηση όπου ενημερώνεται ότι η κράτηση έγινε επιτυχώς και ο χρήστης έχει 24 ώρες να παραλάβει το βιβλίο που έκλεισε.



Εικόνα 2.9: Αναδυόμενο παράθυρο μηνύματος ο χρήστης έκανε επιτυχώς τη κράτηση του και πρέπει να πάει να παραλάβει το βιβλίο από τη βιβλιοθήκη

2.3.3 User profile

BILBO Search...

Konstantinos Kotorenis [Sign Out](#)

ID: 2
E-mail: konstantinos.kotorenis@gmail.com
Date of birth: 30/2/1960

Borrowings

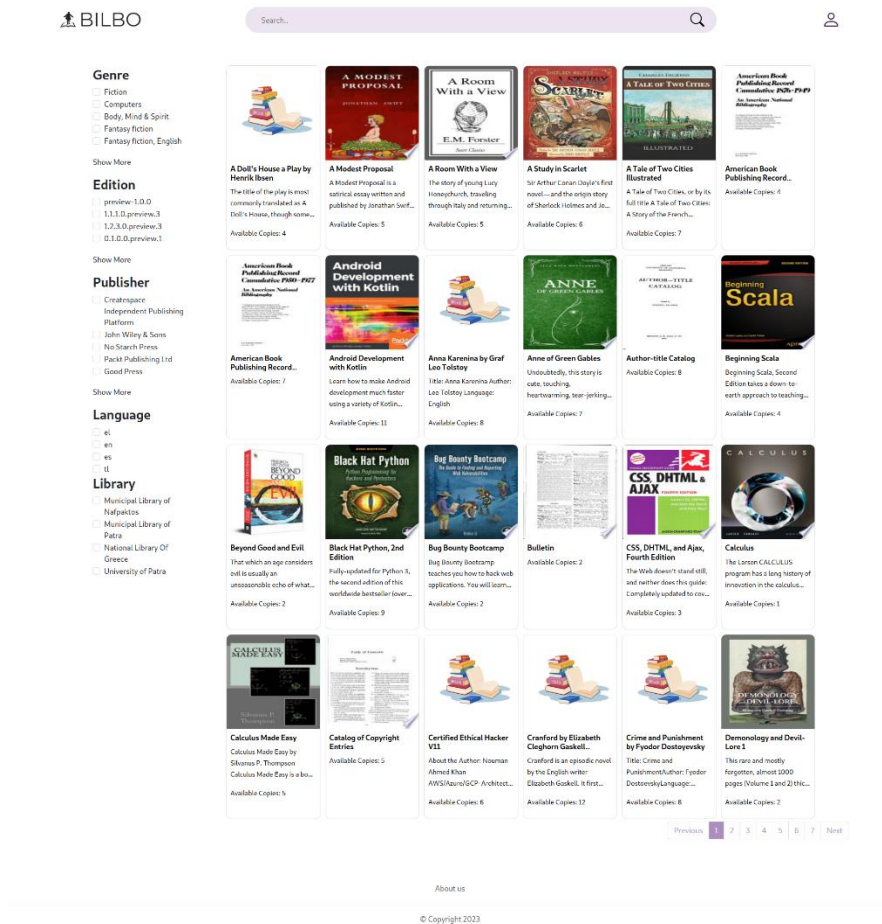
Name	ISBN	Library	Status	Start Date	End Date	Overdue
Name: Mastering Kali Linux for Advanced Penetration Testing	ISBN: 9781787128170	Library: Municipal Library of Patra	Borrowed	2023-05-15	2023-05-16	2023-05-18
Name: Η Αλίκη στη χώρα των θαυμάτων	ISBN: 1006912185	Library: University of Patra	Borrowed	2023-05-11	2023-05-12	Overdue 1 days
Name: A Modest Proposal	ISBN: 9783736800762	Library: University of Patra	Borrowed	2023-05-02	2023-05-03	Returned Late 2023-05-21
Name: Tale Adventures of Huckleberry Finn	ISBN: 1973321879	Library: University of Patra	Borrowed	2023-05-01	2023-05-02	Overdue 11 days
Name: A Tale of Two Cities Illustrated	ISBN: 9798589534207	Library: University of Patra	Borrowed	2023-04-02	2023-04-03	Returned Late 2023-05-20

[About us](#)
© Copyright 2023

Εικόνα 2.10: Προφίλ χρήστη

Σε αυτή τη σελίδα ο χρήστης μπορεί να δει τις πληροφορίες του. Ο χρήστης μπορεί επίσης να κάνει sign out. Τέλος εμφανίζεται και μια λίστα με την κατάσταση των βιβλίων που έχει δανειστεί ο χρήστης• φαίνεται αν έχει γίνει κράτηση ενός βιβλίου, αν ο χρήστης πήγε και δανείστηκε ένα βιβλίο, τότε πρέπει να το επιστρέψει, αν το επέστρεψε και αν έχει ξεπεράσει τη περίοδο επιστροφής.

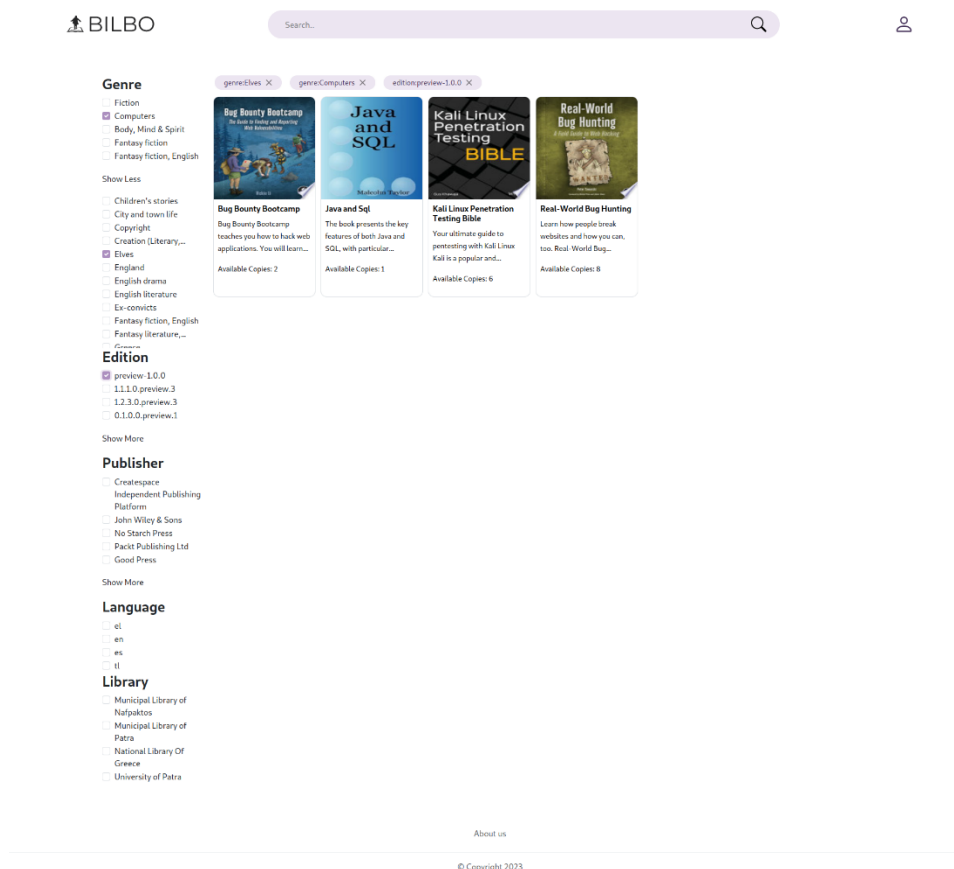
2.3.4 Search page



Εικόνα 2.11: Σελίδα αναζήτησης

Στην σελίδα search εμφανίζονται τα αποτελέσματα της αναζήτησης. Σε κάθε σελίδα εμφανίζονται 24 αποτελέσματα και υπάρχει δυνατότητα ο χρήστης να προχωρήσει και σε επόμενες σελίδες αποτελεσμάτων. Η κάθε σελίδα φορτώνει τα βιβλία από τη βάση μόνο όταν επιλεγεί, προκειμένου να είναι πιο γρήγορη η ανταπόκριση.

Εάν ο χρήστης δεν αναζητήσει κάποια συγκεκριμένη πρόταση ή λέξη, εμφανίζονται όλα τα βιβλία. Έπειτα μπορεί να εφαρμόσει φίλτρα πάνω στα αποτελέσματα.




Εικόνα 2.12: Σελίδα αναζήτησης αφού έχουν επιλεγεί φίλτρα

Τα επιλεγμένα φίλτρα εμφανίζονται πάνω από τα βιβλία σαν φυσαλίδες και διαγράφονται πατώντας πάνω τους. Τα φίλτρα επιλέγονται δυναμικά από τα δεδομένα που υπάρχουν μέσα στην βάση. Στην λίστα εμφανίζονται τα φίλτρα που έχουν τα περισσότερα αποτελέσματα, και στο *show more* υπάρχουν όλα τα υπόλοιπα για την διευκόλυνση του χρήστη.

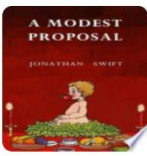
Πατώντας πάνω στο βιβλίο ο χρήστης κατευθύνεται στην σελίδα book info που περιέχει πληροφορίες για το βιβλίο και σε ποιες βιβλιοθήκες είναι διαθέσιμο.

2.3.5 Book info page

Στη σελίδα του βιβλίου ο χρήστης μπορεί να διαβάσει μία περίληψη του βιβλίου καθώς και να δει χαρακτηριστικά του όπως ο συγγραφέας, το ISBN, την έκδοση κ.α. Ακριβώς από κάτω στον χρήστη παρουσιάζεται ένας χάρτης με τις βιβλιοθήκες στις οποίες είναι διαθέσιμο το βιβλίο.



Sign in




A Modest Proposal

A Modest Proposal is a satirical essay written and published by Jonathan Swift. Swift suggests that the impoverished Irish might ease their economic troubles by selling their children as food for rich gentlemen and ladies. This satirical hyperbole mocks heartless attitudes towards the poor, as well as Irish policy in general. In English writing, the phrase "a modest proposal" is now conventionally an allusion to this style of straight-faced satire. Swift goes to great lengths to support his argument, including a list of possible preparation styles for the children, and calculations showing the financial benefits of his suggestion. He uses methods of argument throughout his essay which lampoon the then-influential William Petty and the social engineering popular among followers of Francis Bacon. These lampoons include appealing to the authority of "a very knowing American of my acquaintance in London" and "the famous Psalmist, a native of the island Formosa." This essay is widely held to be one of the greatest examples of sustained irony in the history of the English language. Much of its shock value derives from the fact that the first portion of the essay describes the plight of starving beggars in Ireland, so that the reader is unprepared for the surprise of Swift's solution when he states, "A young healthy child well nursed, is, at a year old, a most delicious nourishing and wholesome food, whether stewed, roasted, baked, or boiled; and I make no doubt that it will equally serve in a fricassee, or a ragout." Readers unacquainted with its reputation as a satirical work often do not immediately realize that Swift was not seriously proposing cannibalism and infanticide. The satirical element of the pamphlet is often only understood after the reader notes the allusions made by Swift to the attitudes of landlords, such as the following: "I grant this food may be somewhat dear, and therefore very proper for Landlords, who as they have already devoured most of the Parents, seem to have the best Title to the Children." Swift extends the conceit to get in a few jibes at England's mistreatment of Ireland, noting that "For this kind of commodity will not bear exportation, and flesh being of too tender a consistence, to admit a long continuance in salt, although perhaps I could name a country, which would be glad to eat up our whole nation without it."

Author	Publisher	Edition
Jonathan Swift	BookRix	1.20.21.0 preview.3
Genre	Release	Language
Fiction	2018-10-16	en
ISBN	9783736800762	

Reservation points




Library	Address	Copies
University of Patra	Ypatias 4, Panepistimioupoli Patron, 265 04	5


About us

© Copyright 2023

Εικόνα 2.13: Σελίδα βιβλίου χωρίς να έχει συνδεθεί ο χρήστης με μία βιβλιοθήκη να παρέχει το βιβλίο



Sign in

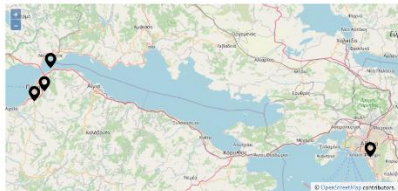


The History of the Decline and Fall of the Roman Empire

No summary

Author	Publisher	Edition
Edward Gibbon		0.5.1.0 preview.0
Genre	Release	Language
Byzantine Empire	1870	en
ISBN	PSU/0000027559437	

Reservation points



Library	Address	Copies
University of Patra	Ypatias 4, Panepistimioupoli Patron, 265 04	2
Municipal Library of Patra	Malazonas 110, Patra 262 21	3
Municipal Library of Naupaktos	Kezeri 12, Naupaktos 303 00	3
National Library Of Greece	32 Panepistimiou Street, Athens 106 79	1

About us

© Copyright 2023

Εικόνα 2.14: Σελίδα βιβλίου χωρίς να έχει συνδεθεί ο χρήστης με τέσσερις βιβλιοθήκες να παρέχουν το βιβλίο

Πιο συγκεκριμένα ο χάρτης δείχνει σε ποιες τοποθεσίες υπάρχουν αντίτυπα. Περνώντας το ποντίκι πάνω από τα εικονίδια επισημαίνεται η βιβλιοθήκη στην δεξιά λίστα. Επίσης ο χάρτης αλλάζει τη μεγέθυνσή του δυναμικά ανάλογα με το πόσες βιβλιοθήκες υπάρχουν.

Τέλος όταν ο χρήστης είναι συνδεδεμένος, δίνεται η δυνατότητα να κάνει κράτηση στο βιβλίο από μία από τις διαθέσιμες βιβλιοθήκες.

BILBO Search. [User Icon]

The History of the Decline and Fall of the Roman Empire
No summary

Author: Edward Gibbon
Publisher: [blank]
Edition: 0.5.1.0.preview.0
Genre: Byzantine Empire
Release: 1870
Language: en
ISBN: PSU:000027559437

Reservation points

Library	Address	Copies	Reserve
University of Patra	Υπάτιος 4, Πανεπιστιμιουπολι Patron, 265 04	2	Reserve W
Municipal Library of Patra	Μαζωπος 110, Patra 262 21	3	Reserve W
Municipal Library of Nafpaktes	Κοζονι 12, Nafpaktes 303 00	3	Reserve W
National Library Of Greece	32 Panepistimiou Street, Athens 106 79	1	Reserve W

About us
© Copyright 2023

Εικόνα 2.15: Σελίδα βιβλίου αφού έχει συνδεθεί ο χρήστης με δυνατότητα κράτησης και τέσσερις βιβλιοθήκες να παρέχουν το βιβλίο

2.3.6 Library info page

BILBO Search. [User Icon]

University of Patra
Υπάτιος 4, Πανεπιστιμιουπολι Patron, 265 04

Location & Working Hours

Address: Υπάτιος 4, Πανεπιστιμιουπολι Patron, 265 04
Contact: Τηλεφωνο: 2621038894, Email: info@upatras.gr
Working Hours: Monday: 8:30 AM-7 PM, Tuesday: 8:30 AM-7 PM, Wednesday: 8:30 AM-7 PM, Thursday: 8:30 AM-7 PM, Friday: closed, Saturday: closed, Sunday: closed

Recommended Books

A Modest Proposal, A Modest Proposal, A Study in Scarlet, A Tale of Two Cities Illustrated, The Adventures of Huckleberry Finn, Η Αλίκη στο γαλάζιο πλοίο, The National Union Catalog, Free 1954., Anna Karenina by Graf Leo Tolstoy, Black Hat by [blank]

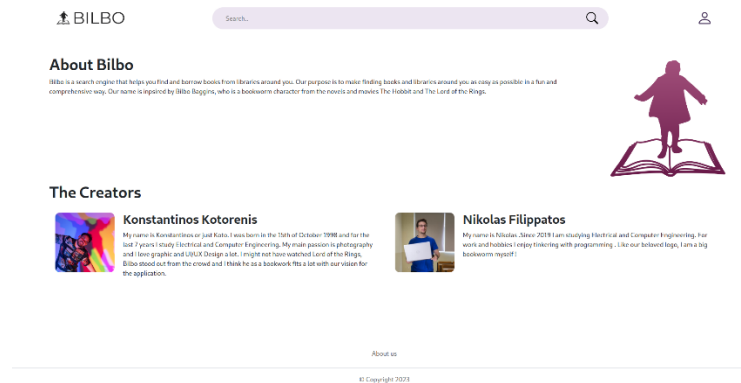
About us
© Copyright 2023

Εικόνα 2.16: Σελίδα πληροφοριών της βιβλιοθήκης

Για την κάθε βιβλιοθήκη προβάλλονται πληροφορίες όπως η διεύθυνση της, οι ώρες λειτουργίας της και κάποια προτεινόμενα βιβλία από την βιβλιοθήκη (προς το παρόν η λίστα είναι στατική).

2.3.7 About us page

Τέλος υπάρχει και το about us page, στο οποίο προβάλλουμε πληροφορίες για την ιστοσελίδα και τους δημιουργούς της.



Εικόνα 2.17: Σελίδα «σχετικά με εμάς»

3 COMPUTER CODE

Δομή MVC (Model View Controller) .

Η εφαρμογή μας αποτελείται από τους φακέλους :

- **Controllers**
Συναρτήσεις ελέγχου.
- **Model**
Στον φάκελο model υπάρχει η βάση, και τα αρχεία για την δημιουργία της
- **Public**
Εικόνες, CSS, JavaScript
- **Routes**
Τα αρχεία των διαδρομών που επιτρέπουμε στον χρήστη να έχει πρόσβαση
- **Views**
Τα αρχεία handlebars που αποτελούν την ιστοσελίδα μας

3.1 Εκκίνηση της εφαρμογής

Υπάρχουν 3 τρόποι να τρέξει τοπικά η εφαρμογή .

- i. **Node** :
npm run startWebsite (node index.js)
- ii. **Nodemon**
npm start (nodemon index.js)

iii. Docker

```
docker build -t bilbo
```

```
docker run -d -p 8080:8080 bilbo
```

Μπορούμε να δούμε την εφαρμογή στην διεύθυνση localhost:8080/

Εναλλακτικά η εφαρμογή φιλοξενείται στην υπηρεσία Heroku: bilbolibrary.herokuapp.com

3.2 Επικοινωνία με την βάση

Στο αρχείο *controllers/database.js* υπάρχουν οι συναρτήσεις που επικοινωνούν με την βάση.

Σε κάθε συνάρτηση χτίζουμε το query με τις παραμέτρους που περνάμε. Αξιοποιήσαμε την SQLite σαν βάση, και τις βιβλιοθήκες better-sqlite3

ALGORITHM 1: getBookInfo

```
/**
 * Returns information about books . Every option other than callback is optional , if no option is given it will
return all the books without copies
 * @param {*} isbn specify the isbn (optional)
 * @param {*} title specify the title (optional)
 * @param {*} numOf true or null, if we want to focus more on the number of results back
 * @param {*} copies true or null , if we want to also get the number of copies that exist
 * @param {*} filters searches with filters
 * @param {*} limit limits the results
 * @param {*} offset starts returning from a specific result
 * @param {*} callback
 */
getBookInfo: function (isbn, title, numOf, copies, filters, limit, offset, callback) {
  let stmt, books, query;

  query = `SELECT * FROM BOOK`

  if (numOf) {

    query = `SELECT COUNT(distinct isbn) as count_result,title,library.name as library,genre ,
language,book.summary as summary,publisher, edition, author from BOOK join COPIES on isbn=book_isbn join
LIBRARY on library_id=LIBRARY.id`

  }

  if (copies && !numOf) {
```

```

    query = `SELECT isbn,title,author,edition,publisher,release, genre , language, book.summary as summary,
BOOK.photo as photo,SUM(copy_num) as copy_num, library.name as library FROM BOOK join COPIES on
isbn=book_isbn join LIBRARY on library_id=LIBRARY.id`
  }
  if (isbn || title) { query += ` WHERE` }

  if (isbn) { query += ` isbn=?` }

  if (isbn && title) { query += ` or` }

  if (title) {
    let matchingPhrases;
    try {
      matchingPhrases = getRegex(title);
    } catch (err) {
      callback(err, null);
    }

    query += ` title in (${matchingPhrases.map(() => '?').join(',')}`
    title = matchingPhrases;
  }

  if (filters && Object.keys(filters) !== 0) {
    filters = JSON.parse(filters);

    for (let key in filters) {
      if (filters[key].length) {
        if (!title && !isbn && key !== Object.keys(filters)[0]) {
          query += ` and`
        } else if (!title && !isbn && key === Object.keys(filters)[0]) {
          query += ` WHERE`
        }
        else if (title || isbn) { query += ` and` }
        let list = filters[key].map(word => `${word}`).join(',')
        query += ` ${key} in (${list})`
      }
    }
  }

  if (copies) { query += ` GROUP BY isbn` }

```

```
query += ` ORDER BY title ASC`

if (limit) { query += ` LIMIT ?` }

if (offset) { query += ` OFFSET ?` }

stmt = betterDb.prepare(query);

try {
  if (isbn && title && limit && offset) {
    books = stmt.all(isbn, title, limit, offset)
  } else if (isbn && title && limit) {
    books = stmt.all(isbn, title, limit)
  } else if (isbn && limit && offset) {
    books = stmt.all(isbn, limit, offset)
  } else if (title && limit && offset) {
    books = stmt.all(title, limit, offset)
  } else if (isbn && title) {
    books = stmt.all(isbn, title)
  } else if (isbn && limit) {
    books = stmt.all(isbn, limit)
  } else if (title && limit) {
    books = stmt.all(title, limit)
  } else if (limit && offset) {
    books = stmt.all(limit, offset)
  } else if (title) {
    books = stmt.all(title)

  } else if (limit) {
    books = stmt.all(limit)

  } else if (isbn) {
    books = stmt.all(isbn)
  } else {
    books = stmt.all();
  }
} catch (err) {
  callback(err, null);
}
callback(null, books);
},
```

Η συνάρτηση «getBookInfo» παίρνει διάφορες τιμές σαν παραμέτρους για να μας επιστρέψει πληροφορίες που επιθυμούμε. Όλες οι παράμετροι είναι προεραϊτικοί. Εάν είναι όλα null επιστρέφει όλα τα βιβλία που είναι στην βάση. Για να χρησιμοποιήσουμε το αποτέλεσμα περνάμε μέσα στην συνάρτηση callback είτε το error είτε το αποτέλεσμα.

ALGORITHM 2: Router for book info

```
router.get('/',
  (req, res, next) => {
    // this is a static book list displaying 12 books
    database.getBookInfo(isbn=null,title=null,numOf=null,copies=true,filters=null,limit=12,offset=null, function
(err, books) {
      if (err) {
        console.log(err)
        res.status(500).send('Internal Server Error')
      } else {
        res.locals.booklist = books;
        next();
      }
    })
  },
  (req, res) => {
    res.render('homepage', {
      style: 'index.css',
      title: 'Home',
      signedIn: req.session.signedIn,
    });
  });
});
```

Εδώ βλέπουμε μια χρήση της. Σαν συνάρτηση callback, περνάμε την ανώνυμη συνάρτηση function (err,books) . Εάν η αίτηση στην βάση είναι επιτυχής, θα αποθηκεύτουν τα βιβλία στην μεταβλητή books, και θα τα στείλουμε στην handlebars για rendering στην μεταβλητή booklist.

Άλλη μια σημαντική συνάρτηση είναι η getBorrowingState .

ALGORITHM 3: getAllBorrowingState

```
getAllBorrowingState: function (userId, callback) {
  const stmt = betterDb.prepare("Select BOOK.title AS book_title, BOOK.photo, B.book_isbn, L.name AS
library_name, B.library_id, B.user_id, DATE(B.date_reserved) AS date_reserved, DATE(B.date_borrowed) AS
date_borrowed, DATE(R.date_returned) AS date_returned, DATE(B.date_borrowed, '+15 days') AS date_return,
```

```

IIF(ROUND(JULIANDAY(R.date_returned) - JULIANDAY(DATE(B.date_borrowed, '+15 days'))),
ROUND(JULIANDAY(R.date_returned) - JULIANDAY(DATE(B.date_borrowed, '+15 days'))),
ROUND(JULIANDAY(DATE('now')) - JULIANDAY(DATE(B.date_borrowed, '+15 days')))) AS difference,
IIF(IIF(ROUND(JULIANDAY(R.date_returned) - JULIANDAY(DATE(B.date_borrowed, '+15 days'))),
ROUND(JULIANDAY(R.date_returned) - JULIANDAY(DATE(B.date_borrowed, '+15 days'))),
ROUND(JULIANDAY(DATE('now')) - JULIANDAY(DATE(B.date_borrowed, '+15 days'))))<=0, 0, 1) AS overdue
from LIBRARY AS L JOIN(BOOK JOIN(Borrowing AS B LEFT OUTER JOIN Return AS R ON B.user_id=R.user_id
AND B.book_isbn=R.book_isbn AND B.library_id=R.library_id) ON BOOK.isbn=B.book_isbn) ON
L.id=B.library_id where B.user_id=? ORDER BY B.date_reserved DESC")

let borrowingStates;
try {
    borrowingStates = stmt.all(userId)
    callback(null, borrowingStates)
}
catch (err) {
    callback(err, null)
}
},

```

Μέσω ενός SQL query επιστρέφει όλα τα βιβλία που έχει δανειστεί ο χρήστης και σε ποια βιβλιοθήκη έχει γίνει ο κάθε δανεισμός καθώς και την κατάσταση δανεισμού τους.

3.3 Αναζήτηση

Το request για την αναζήτηση πραγματοποιείται στο clientside javascript. (οι συναρτήσεις βρίσκονται στα αρχεία search.js, show_pages.js, route_search.js)

Γίνεται πρώτα από όλα ένα fetch για να δούμε τον αριθμό των αποτελεσμάτων με βάση την αναζήτηση και τα φίλτρα του χρήστη, με βάση τον οποίο δημιουργούμε σελίδες με μέγιστο αριθμό βιβλίων 24.

ALGORITHM 4: Search.js

```

/**
 * Returns the number of results we would get with the title and filters that the user searched
 * @returns
 */
async function fetchNumOfResults() {

    let link;

    link = '/fetchNumOfResults'

```

```

return await fetch(link, {
  method: "POST",
  credentials: "same-origin",
  headers: {
    "Content-Type": "application/json",
  },
  redirect: "follow",
  referrerPolicy: "no-referrer",
  body: JSON.stringify({ "filters": window.gFilters, "title": window.searchBarValue }),
}).then((res) => {
  return res.json();
}).then((data) => {
  return data[0].count_result;
}).catch(error => {
  console.log(error);
});
}

```

Έπειτα δημιουργείται το page navigation, με κουμπία Previous, ξεχωριστά κουμπία που απαριθμούν κάθε σελίδα και Next, και καλείται να φορτωθεί η πρώτη σελίδα.

ALGORITHM 5.1: Show_pages.js

```

/**
 * Initializes the pages for the result
 * Fetches the number of results, creates the navigation if the number of pages exceeds 1 and calls forth the first page
to be displayed
 */
async function page_initilazation() {

  let numOfResults, numOfPages;

  // removes the previous results
  document.getElementById('page-selector').innerHTML = "

  // for the number of page created
  numOfResults = await showResult();

  window.booksPerPage = 24;

  numOfPages = Math.ceil(numOfResults / window.booksPerPage);

```

```
if (numOfPages > 1) {  
  pageNavigationCreation(numOfPages);  
}
```

```
// shows first page of results  
showPage(1);
```

```
}
```

Η κάθε σελίδα κάνει αίτηση για μεγίστο αριθμό 24 βιβλία, και offset 24* αριθμο_σελίδας

ALGORITHM 5.2: Show_pages.js

```
/**  
 * Displays the results of the search  
 * @param number Is the page number called forth to be displayed  
 */  
function showPage(number) {  
  
  // placeAllBooksBytitle is in the file search.js  
  placeAllBooksByTitle(limit = window.booksPerPage, offset = (number - 1) * window.booksPerPage);  
}
```

Η αίτηση αυτή διαχειρίζεται στον server στην διεύθυνση /fetch_filters

ALGORITHM 6: Route_search.js

```
router.post('/fetch_filters', (req, res) => {  
  
  let filters = JSON.stringify(req.body.filters);  
  
  database.getBookInfo(isbn = null, title = req.body.title, numOf = false, copies = true, filters = filters, limit =  
req.body.limit, offset = req.body.offset, function (err, books) {  
    if (err) {  
      console.log(err)  
      res.status(500).send('Internal Server Error')  
    } else {  
      res.send(books);  
    }  
  })  
})  
})
```

Κάθε φορά που ο χρήστης κάνει κάποια αναζήτηση, προσθέτει φίλτρα ή πηγαίνει σε άλλη σελίδα, φορτώνεται το περιεχόμενο που πρέπει να βλέπει.

Η ίδια η αναζήτηση των βιβλίων πραγματοποιείται με regex. Ελέγχει από όλους τους τίτλους των βιβλίων, εάν η αναζήτηση του χρήστη υπάρχει αυτούσια η φράση, εάν υπάρχει κάποια από τις λέξεις της φράσης είτε τέλος κάποιο partial από τις λέξεις.

ALGORITHM 7: Database.js

```
function getRegex(title) {

  let rows;
  try {
    rows = betterDb.prepare('Select title from BOOK').all()
  } catch (err) {
    throw err;
  }
  const commonWords = [
    'the', 'of', 'and', 'a', 'to', 'in', 'is', 'you', 'that', 'it', 'he', 'was', 'for', 'on', 'are', 'as', 'with', 'his', 'they', 'I']

  title = title.trim();
  const words = title.split(" ");

  const partialWords = words.filter(word => !commonWords.includes(word));

  const partialPattern = partialWords.map(word => `(?:\\b\\B){escapeRegExp(word)}\\w*(?:\\b\\B)`).join('|');

  const exactPattern = `\\b${escapeRegExp(title)}\\b`;

  const pattern = `(?:${partialPattern})|${exactPattern}`;

  const matchingPhrases = rows.filter(row => new RegExp(pattern, 'i').test(row.title)).map(row => row.title);

  return matchingPhrases;
}
```

3.4 Ο κώδικας

Κανείς μπορεί να βρει ολόκληρο τον κώδικα της εργασίας στο παρακάτω σύνδεσμο του GitHub
<https://github.com/nikolasfil/Bilbo>