ΠΑΝΕΠΙΣΤΗΜΙΟ
ΙΩΑΝΝΙΝΩΝ

ΣΧΟΛΗ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ
ΤΗΜΑ ΠΛΗΡΟΦΟΡΙΚΗΣ ΚΑΙ ΤΗΛΕΠΙΚΟΙΝΩΝΙΩΝ

ΠΤΥΧΙΑΚΗ ΕΡΓΑΣΙΑ:

ΣΧΕΔΙΑΣΜΟΣ ΚΑΙ ΑΝΑΠΤΥΞΗ ΔΙΑΔΥΚΤΙΑΚΗΣ ΕΦΑΡΜΟΓΗΣ ΔΙΑΧΕΙΡΙΣΗΣ ΒΙΒΛΙΩΝ ΚΑΙ ΚΟΙΝΩΝΙΚΟΥ ΔΙΚΤΥΟΥ

Ονοματεπώνυμο: Νίκη Χάρλα
Αριθμός Μητρώου: 15744
Επιβλέπων Καθηγητής: Ιωάννης Τσούλος

Άρτα Σεπτέμβριος, 2023

BUILDING WEB APPLICATION FOR BOOK MANAGEMENT AND SOCIAL NETWORKING

Δήλωση μη λογοκλοπής

Δηλώνω υπεύθυνα και γνωρίζοντας τις κυρώσεις του Ν. 2121/1993 περί Πνευματικής Ιδιοκτησίας, ότι η παρούσα πτυχιακή εργασία είναι εξ ολοκλήρου αποτέλεσμα δικής μου ερευνητικής εργασίας, δεν αποτελεί προϊόν αντιγραφής ούτε προέρχεται από ανάθεση σε τρίτους. Όλες οι πηγές που χρησιμοποιήθηκαν (κάθε είδους, μορφής και προέλευσης) για την συγγραφή της εργασίας περιλαμβάνονται στη βιβλιογραφία.

Νίκη Χάρλα

Υπογραφή

ΕΥΧΑΡΙΣΤΙΕΣ

Θα ήθελα να ευχαριστήσω τον επιβλέπων καθηγητή μου κ. Ιωάννη Τσούλο για όλη του την υποστήριξη κατά την διάρκεια εκπόνησης της πτυχιακής μου εργασίας.

ΠΕΡΙΛΗΨΗ

Η εν λόγω πτυχιακή εργασία έχει ως στόχο το σχεδιασμό και την δημιουργία μιας διαδικτυακής εφαρμογής η οποία θα χρησιμεύει ως προσωπικό αποθετήριο βιβλίων και ως μέσο κοινωνικής δικτύωσης, με στόχο την ανταλλαγή αξιολογήσεων για βιβλία.
Αναλύεται η διαδικασία σχεδιασμού και ανάπτυξης της διαδικτυακής εφαρμογής, τα εργαλεία που χρησιμοποιήθηκαν όπως και οι λόγοι για τους οποίους επιλέχτηκαν, οι απαιτήσεις τις εφαρμογής όπως και κάποιοι μέθοδοι υλοποίησης τους. Τέλος παρουσιάζεται ο τρόπος λειτουργίας της εφαρμογής.

ABSTRACT

This thesis aims to design and create a web application that will serve as a personal repository of books and as a social networking tool, aiming at the exchange of book reviews.
The process of designing and developing the web application, the tools used as well as the reasons why they were chosen, the requirements of the application as well as some methods of implementation are analysed. Finally, the mode of operation of the application is presented.

Περιεχόμενα

Πίνακας εικόνων

ΚΕΦΑΛΑΙΟ 1 ΤΕΧΝΟΛΟΓΙΕΣ ΠΟΥ ΧΡΗΣΙΜΟΠΟΙΗΘΗΚΑΝ

1.1 Εισαγωγή στη Spring Boot

Στον κόσμο της ανάπτυξης εφαρμογών που βασίζονται σε Java, η Spring Boot κατέχει μια θέση ως μια τεχνολογία που έχει μεταμορφώσει τον τρόπο με τον οποίο οι προγραμματιστές δημιουργούν ένα ισχυρό και διαχειρίσιμο λογισμικό. Σε αυτό το κεφάλαιο, θα εμβαθύνουμε στην προέλευση, τις βασικές αρχές και στο σημαντικό αντίκτυπο που έχει η Spring Boot στο πλαίσιο της ανάπτυξης εφαρμογών διαδικτύου.

1.1.1 Η προέλευση της Spring Boot

Η γέννηση της Spring Boot, η οποία αποτελεί μέρος του Spring Framework., προήλθε από την ανάγκη να αντιμετωπιστούν οι πολυπλοκότητες και οι προκλήσεις που αντιμετωπίζουν οι προγραμματιστές της Java κατά τη δημιουργία εφαρμογών επιχειρηματικού επιπέδου. Εμφανίστηκε ως απάντηση στους περιορισμούς της Java EE (Enterprise Edition), παρέχοντας εκείνη την εποχή μια ευέλικτη εναλλακτική λύση. Το πλαίσιο δημιουργήθηκε αρχικά από τον Rod Johnson και διατέθηκε ως έργο ανοικτού κώδικα το 2003. Με την πάροδο του χρόνου έχει κερδίσει δημοτικότητα και εξελίχθηκε σε ένα ολοκληρωμένο οικοσύστημα. Ένα σημαντικό πλεονέκτημα της είναι ότι σε αντίθεση με την προκάτοχο της την Spring η Spring Boot προσφέρει ποιο εύκολη παραμετροποίηση, όπως θα αναλυθεί και παρακάτω.

1.1.2 Η σημασία της Spring Boot

Είναι ιδιαίτερη η σημασία της Spring Boot στη σύγχρονη ανάπτυξη εφαρμογών καθώς προσφέρει πολλά και επιτακτικά πλεονεκτήματα, μερικά από τα οποία θα αναφερθούν παρακάτω.

1. Απλοποιημένη παραμετροποίηση: Η Spring Boot χρησιμοποιεί τον κανόνα "convention over configuration" μειώνοντας την ανάγκη για εκτεταμένα αρχεία παραμετροποίησης (configuration files). Παρέχει προεπιλογές επιτρέποντας στους προγραμματιστές να ξεκινήσουν γρηγορότερα την ανάπτυξη της εφαρμογής τους.
2. Ενσωματωμένοι διακομιστές: Η Spring Boot περιλαμβάνει ενσωματωμένους διακομιστές ιστού (όπως Tomcat, Jetty και Undertow) καταργώντας την ανάγκη εγκατάστασης ενός εξωτερικού διακομιστή, το οποίο απλοποιεί την εγκατάσταση του λογισμικού.
3. Αυτόματη διαμόρφωση: Η λειτουργία αυτόματης διαμόρφωσης της Spring Boot αναλύει το σύνολο το σύνολο των φακέλων που βρίσκονται σε μεταγλωττισμένες κλάσεις (classpath) και ρυθμίζει αυτόματα τα στοιχεία της εφαρμογής με βάση τις υπάρχουσες εξαρτήσεις. Αυτή η έξυπνη ρύθμιση μειώνει δραστικά την εμφάνιση επαναλαμβανόμενου κώδικα και ενισχύει την παραγωγικότητα.
4. Spring Boot Actuator: Η Spring Boot είναι εξοπλισμένη με λειτουργίες production-ready (έτοιμες για παραγωγή), όπως και ενσωματωμένων health-checks, μετρήσεων και

security configurations. Αυτά τα χαρακτηριστικά επιταχύνουν την ανάπτυξη εφαρμογών που είναι τόσο ισχυρές όσο και ασφαλείς.

5. Ενσωμάτωση οικοσυστήματος : Επιτρέπει στους προγραμματιστές να αξιοποιήσουν τα υπάρχοντα εργαλεία και βιβλιοθήκες από άλλες τεχνολογίες λογισμικού, επωφελούμενοι παράλληλα από την εξορθολογισμένη προσέγγιση της Spring Boot.
6. Γρήγορη ανάπτυξη λογισμικού: Οι προγραμματιστές επωφελούνται από τα εργαλεία ανάπτυξης λογισμικού και τις δυνατότητες ζωντανής επαναφόρτισης της Spring Boot, τα οποία διευκολύνουν τις γρήγορες επαναλήψεις και την ανατροφοδότηση σε πραγματικό χρόνο κατά τη διάρκεια της διαδικασίας ανάπτυξης. [7]

1.1.3 Inversion of Control (IoC) και Dependency Injection (DI) στη Spring Boot

Είναι ιδιαίτερη σημαντική η δημιουργία ευέλικτων, συντηρήσιμων και ελέγξιμων εφαρμογών. Η Spring Boot το επιτυγχάνει αυτό χρησιμοποιώντας έννοιες όπως IoC και DI.

1.1.3.1 Εισαγωγή στο Inversion of Control (IoC) και Dependency Injection (DI)

Στον πυρήνα του, το IoC αντιπροσωπεύει μια θεμελιώδη αλλαγή στον τρόπο με τον οποίο γίνεται η διαχείριση στα διάφορα στοιχεία μιας εφαρμογής. Παραδοσιακά, σε ένα στενά συνδεδεμένο σύστημα (tightly coupled), τα συστατικά συχνά ενσαρκώνουν και διαχειρίζονται τις εξαρτήσεις τους απευθείας. Με το IoC, ο έλεγχος αυτός αντιστρέφεται: τα συστατικά δεν είναι πλέον υπεύθυνα για τη δημιουργία και τη διαχείριση των εξαρτήσεών τους. Αντ' αυτού, μια εξωτερική οντότητα, που αναφέρεται ως IoC container, αναλαμβάνει αυτό τον ρόλο. Στο πλαίσιο της Spring Boot, το IoC container διαχειρίζεται τον κύκλο ζωής των αντικειμένων και των εξαρτήσεών τους. Επιπρόσθετα επιβλέπει τη δημιουργία, την καλωδίωση και την καταστροφή των beans (αντικείμενα που διαχειρίζεται η Spring) εντός της εφαρμογής.
Το Dependency Injection χρησιμοποιείται για την υλοποίση του IoC. Είναι η διαδικασία μέσω της οποίας το IoC «»δοχείο»» (container) προμηθεύει ένα συστατικό με τις απαιτούμενες εξαρτήσεις του. Αντί τα αντικείμενα να δημιουργούν τις εξαρτήσεις τους, τις λαμβάνουν από το «δοχείο». Αυτό συνήθως επιτυγχάνεται μέσω της έγχυσης κατασκευαστή (constructor injection), της έγχυσης καθοριστών (setter injection) ή της έγχυσης μεθόδων (method injection).

Τα παραπάνω συντελούν στο να αποσυνδέουν (decouples) τα συστατικά θέτοντας τα χαλαρά συνδεδεμένα (loosely coupled). Πλέον τα αντικείμενα δεν βασίζονται σε συγκεκριμένες υλοποιήσεις, αλλά αντ' αυτού βασίζονται σε αφαιρέσεις (διεπαφές ή αφηρημένες κλάσεις). Έτσι προωθείτε η τμηματοποίση του κώδικα το οποίο επιτρέπει την ευκολότερη συντήρηση, κατανόσηση του και τη δυνατότητα ελέγχου του (testability) καθώς το DI καθιστά εύκολη την αντικατάσταση των πραγματικών εξαρτήσεων με εικονικά αντικείμενα κατά τη διάρκεια των ελέγχων. Αυτό επιτρέπει τον ολοκληρωμένο και αξιόπιστο τμηματικό έλεγχο (unit testing) μεμονωμένων στοιχείων.

1.1.3.2 Υλοποίηση του Inversion of Control (IoC) και Dependency Injection (DI) στη Spring Boot

Η Spring Boot αξιοποιεί το IoC «δοχείο» container του Spring Framework για να επιτύχει το IoC και το DI. Όταν επισημαίνεται μια κλάση με @Component, @Service, @Repository, ή άλλες με κάποια άλλη επισήμανση, η Spring Boot την αναγνωρίζει ως Bean το οποίο θα πρέπει να διαχειρίζεται από το «δοχείο» container. Μέσω constructor injection ή άλλων μηχανισμών έγχυσης, όπως αναφέρθηκε και στο υπό κεφάλαιο 2.1.3.1, η Spring Boot εγχέει εξαρτήσεις, καθιστώντας τις διαθέσιμες για χρήση από τα διάφορα συστατικά. Ακόμα προσφέρει διάφορους τρόπους για τη διαμόρφωση και την προσαρμογή της συμπεριφοράς ενός IoC container. Μπορούν να χρησιμοποιηθούν οι κλάσεις @Configuration, @Bean μέθοδοι και τα αρχεία ιδιοτήτων (property files) για να ρυθμιστούν λεπτομερώς οι λειτουργίες του container. Αυτή η ευελιξία διασφαλίζει ότι η Spring Boot προσαρμόζεται στις συγκεκριμένες απαιτήσεις της κάθε εφαρμογής. Το IoC και το Dependency Injection είναι θεμελιώδεις έννοιες του Spring Boot που στηρίζουν τις αρχές του σχεδιασμού του. Αγκαλιάζοντας αυτές τις έννοιες, το Spring Boot προωθεί τη δημιουργία τμηματοποιημένων, ελέγξιμων και συντηρήσιμων εφαρμογών. [2]

1.1.4 Ενσωματώνοντας στη Spring Boot, Java Persistence Api (JPA) και Hibernate.

1.1.4.1 Εισαγωγή στο Java Persistence Api (JPA) και Hibernate.

Το JPA είναι μια τυποποιημένη προδιαγραφή της Java που απλοποιεί την πρόσβαση σε βάσεις δεδομένων προσφέροντας μια δίοδο επικοινωνίας ανάμεσα σε μια Spring Boot εφαρμογή και σε μια σχεσιακή βάση δεδομένων. Το Hiberate είναι ένα framework και ένα ORM (Object Relational Mapping) εργαλείο το οποίο υλοποιεί τις JPA λειτουργίες. Αυτά τα δυο σε συνδυασμό προσφέρουν ποικίλα πλεονεκτήματα στην διαχείριση δεδομένων. Αφαιρούν την πολυπλοκότητα των σχεσιακών βάσεων δεδομένων καθώς επιτρέπουν στους προγραμματιστές να αλληλοεπιδρούν με αντικείμενα Java αντί για SQL queries, Με τον τρόπο αυτό απλοποιείται η αλληλεπίδραση του προγραμματιστή με τις βάσεις δεδομένων και μειώνεται η πολυπλοκότητα. Επιπλέον, το ORM παρέχει έναν βαθμό ανεξαρτησίας από τη βάση δεδομένων, επιτρέποντας στις εφαρμογές να εναλλάσσονται μεταξύ συστημάτων βάσεων δεδομένων με ελάχιστες τροποποιήσεις στον κώδικα, μειώνοντας έτσι τον εγκλωβισμό στον προμηθευτή (vendor lock-in). Το ORM μεταφέρει το παράδειγμα του αντικειμενοστραφούς προγραμματισμού στις λειτουργίες της βάσης δεδομένων. Αυτή η μεταφορά απλοποιεί την ανάπτυξη και διατηρεί την αντικειμενοστραφή ακεραιότητα της εφαρμογής. Ακόμα ενισχύει καθαρότερο και πιο συντηρήσιμο κώδικα συγκεντρώνοντας την λογική πρόσβασης δεδομένων σε ένα σημείο με αποτέλεσμα να δημιουργείται ξεκάθαρος διαχωρισμός μεταξύ της επιχειρησιακής λογικής και της πρόσβασης στα δεδομένα.

1.1.4.2 Παράδειγμα χρήσης του JPA.

Στην εφαρμογή η οποία υλοποιήθηκε στα πλαίσια αυτής της εργασίας χρησιμοποιήθηκε η τεχνολογία JPA. Αυτό επιτεύχθηκε κάνοντας τα παρακάτω.

Προστέθηκαν οι απαραίτητες εξαρτήσεις στο pom.xml αρχείο

```
<dependency>
   <groupId>org.springframework.boot</groupId>
   <artifactId>spring-boot-starter-data-jpa</artifactId>
</dependency>
```

Η Spring Boot παρέχει ένα ευρύ φάσμα από configuration properties οι οποίες επιτρέπουν την προσαρμογή των JPA και Hibernate με βάση τις ανάγκες του εκάστοτε προγράμματος. Παρακάτω με βάση τις ανάγκες της εφαρμογής που υλοποιήθηκε **προστέθηκαν οι παρακάτω παραμετροποιήσεις στο αρχείο application.properties**.

```
org.hibernate.dialect = PostgreSQLDialect
spring.datasource.driver-class-name=org.postgresql.Driver
spring.jpa.hibernate.ddl-auto = none
spring.datasource.url=jdbc:postgresql://localhost:5432/literature_social_
network
spring.datasource.username=postgres
spring.datasource.password=o
spring.jpa.show-sql=true
spring.jpa.properties.hibernate.format_sql=true
```

Το “org.hibernate.dialect” χρησιμοποιείται για να οριστεί η κατάλληλη διάλεκτος για την Βάση δεδομένων. Αυτό εξασφαλίζει ότι το Hibernate θα παράγει εντολές SQL βελτιστοποιημένες και συμβατές με τη συγκεκριμένη βάση δεδομένων με αυτή που χρησιμοποιείται στην εκάστοτε εφαρμογή. Η “PostgreSQLDialect” δηλώνει ότι χρησιμοποιείται PostgreSQL.

Το “spring.datasource.driver-class-name” χρησιμοποιείται για να καθοριστεί το πλήρες όνομα της κλάσης του JDBC (Java Database Connectivity) το οποίο πρέπει να χρησιμοποιηθεί για τη σύνδεση σε μια βάση δεδομένων.

Το “spring.jpa.hibernate.ddl-auto” χρησιμοποιείται για να ρυθμίσει τον τρόπο με τον οποίο το Hibernate, το πλαίσιο της αντικειμενοσχεσιακής απεικόνισης (ORM), θα πρέπει να χειρίζεται τη δημιουργία και τον συγχρονισμό του σχήματος της βάσης δεδομένων. Κατά την αρχική δημιουργία του σχήματος της βάσης δεδομένων η τιμή της παραμέτρου ήταν “create-drop” με σκοπό κάθε φορά που θα τρέχει το πρόγραμμα το Hibernate να διαγράφει και να δημιουργεί το σχήμα της βάσης από την αρχή. Αφού η διαδικασία δημιουργίας της βάσης δεδομένων ολοκληρώθηκε η τιμή της παραμέτρου ενημερώθηκε σε “none” ώστε να μην τροποποιηθεί το σχήμα της βάσης ξανά.

Το spring.datasource.url χρησιμοποιείται για τον καθορισμό της διεύθυνσης URL που θα πρέπει να χρησιμοποιεί το Hibernate ώστε να δημιουργήσει μια σύνδεση με την βάση δεδομένων.

Τα «spring.datasource.username» και «spring.datasource.password» χρησιμοποιούνται για την αυθεντικοποίηση με την βάση δεδομένων.

Τα «spring.jpa.show-sql» και « spring.jpa.properties.hibernate.format_sql» χρησιμοποιούνται για την απασφαλμάτωση του προγράμματος καθώς εμφανίζουν στην έξοδο καταγραφής τις εντολές SQL.
Σε ένα μη τοπικό περιβάλλον (production) αποφεύγονται να χρησιμοποιούνται τα στοιχεία αυθεντικοποίησης και να εκθέτονται οι εντολές SQL σε αρχεία καταγραφής, για λόγους ασφαλείας.

Έπειτα δημιουργήθηκαν οι κλάσεις οντότητα (entinty classes) οι οποίες αντιπροσωπεύουν τους πίνακες της βάσεις δεδομένων. Ο τρόπος με τον οποίο δημιουργούνται είναι η επισήμανση της κλάσης με το «@Entity». Ακόμα μπορούν να οριστούν οι σχέσεις μεταξύ των πινάκων χρησιμοποιώντας επισημάνσεις όπως «@ManyToMany» και «@ManyToOne». Στο παρακάτω παράδειγμα εμφανίζεται η κλάση Friendship η οποία προσομοιάζει έναν πίνακα της βάσης δεδομένων ο οποίος αντιπροσωπεύει τη «φιλία» μεταξύ των διαφόρων χρηστών της εφαρμογής

```
@Getter
@Setter
@Entity(name = "Friendship")
public class Friendship {
   @Id
   @GeneratedValue(strategy = GenerationType.IDENTITY)
   private Long friendshipID;

   @ManyToOne
   @JoinColumn(name = "appUserID")
   private AppUser appUserID;

   @ManyToOne
   @JoinColumn(name = "friendID")
   private AppUser friendID;

   @Enumerated(EnumType.STRING)
   private FriendshipStatus FriendshipStatus;

   private Date timestamp;

}
```

αποθηκεύοντας το id ενός χρήστη με το id ενός άλλου. Οι χρήστες αυτοί είτε είναι φίλοι είτε έχει σταλθεί αίτημα φιλίας από τον ένα χρήστη στον άλλον. Όπως παρατηρείται γίνεται χρήση και της επισήμανσης «@ManyToOne» το οποία υποδηλώνει ότι πολλά instances της επισυναπτόμενης Entity κλάσης μπορούν να συσχετιστούν με ένα άλλο instance της κλάσης AppUser. Η επισημάνσεις «@JoinColumn(name = "appUserID")» και «@JoinColumn(name = "friendID")» χρησιμοποιούνται για να καθοριστεί ότι υπάρχει μια στήλη με όνομα "appUserID" και μια με όνομα «friendID» στον πίνακα της βάσης δεδομένων που χρησιμεύουν ως ξένα κλειδιά για την αναπαράσταση της σχέσης. Αυτές οι στήλες χρησιμοποιούνται και για τη δημιουργία της συσχέτισης μεταξύ των δύο οντοτήτων. Η Spring Boot απλοποιεί την πρόσβαση

στη βάση δεδομένων παρέχοντας διεπαφές JpaRepository. Αυτές οι διεπαφές, όταν κληρονομηθούν, προσφέρουν ενσωματωμένες λειτουργίες CRUD (Create, Read, Update, Delete) για τις οντότητες JPA. Στο παρακάτω παράδειγμα έχει υλοποιηθεί η διεπαφή FriendshipRepository η οποία κληρονομεί την διεπαφή JpaRepository. Η FriendshipRepository φέρει τη επισήμανση @Repository. Αυτή η επισήμανση δεν συσχετίζεται άμεσα με την JPA αλλά χρησιμοποιείται στην Spring για να υποδείξει ότι μια συγκεκριμένη κλάση είναι συστατικό αποθετηρίου (repository). Ένα αποθετήριο, στο πλαίσιο του Spring Data JPA, είναι ένας εξειδικευμένος τύπος Spring Bean που είναι υπεύθυνο για την πρόσβαση στη βάση δεδομένων και την επεξεργασία δεδομένων. Συχνά αλληλοεπιδρά με οντότητες JPA για την εκτέλεση λειτουργιών CRUD όπως φανερώνεται και στο συγκεκριμένο παράδειγμα. Έτσι παρατηρούμε ότι μέσω της διεπαφής FriendshipRepository καθίσταται ιδιαίτερα απλή η αποθήκευση δεδομένων χρησιμοποιώντας μεθόδους όπως η save(). Ακόμα προσφέρονται και αρκετές άλλες μέθοδοι για διαγραφή ή επεξεργασία δεδομένων όπως είναι η delete() και η findById() αντίστοιχα.

```
@Repository
public interface FriendshipRepository extends JpaRepository<Friendship, Long> {

    Friendship findFriendshipByAppUserIDAndAndFriendID(AppUser loggeInUser, AppUser
targetUser);

    @Query("SELECT f.appUserID, f.friendID FROM Friendship f WHERE
(f.appUserID.appUserID = :userId OR f.friendID.appUserID = :userId) AND f.FriendshipStatus =
'ACCEPTED'")
    Map<AppUser, AppUser> findFriendsByUserId(@Param("userId") Long userId);

    @Query("SELECT f.appUserID FROM Friendship f WHERE f.friendID.appUserID = :userId
AND f.FriendshipStatus = 'PENDING'")
    List<AppUser> findPendingFriendRequestsByUserId(@Param("userId") Long userId);

    @Query("SELECT f.friendID FROM Friendship f WHERE f.appUserID.appUserID = :userId
AND f.FriendshipStatus = 'REJECTED'")
    List<AppUser> findRejectedFriendRequestsByUserId(@Param("userId") Long userId);

    void deleteFriendshipByAppUserIDAndAndFriendID(AppUser sourceUser, AppUser
targetUser);

}
```

1.2 Εισαγωγή στη τεχνολογία Vaadin

Το Vaadin είναι ένα framework εφαρμογών ιστού βασισμένο σε Java και έχει κερδίσει την δημοτικότητα για τη ιδιαίτερη του προσέγγιση στη δημιουργία σύγχρονων διεπαφών διαδικτύου. Προσφέρει μια εναλλακτική λύση στην παραδοσιακή ανάπτυξη εφαρμογών διαδικτύου, στις οποίες κυριαρχούν προκλήσεις στην αλληλεπίδραση πελάτη-διακομιστή, στον ευέλικτο σχεδιασμό και στην διατήρηση της ποιότητας του κώδικα.

1.2.1 Η προέλευση της τεχνολογίας Vaadin

Το 2000, το Vaadin (παλαιότερα γνωστό ως IT Mill Toolkit) ξεκίνησε επίσημα από τους Joonas Lehtinen και Aleksi Kallio, τους συνιδρυτές της Vaadin Ltd. Η εργαλειοθήκη αποσκοπούσε στην παροχή μιας πιο ολοκληρωμένης λύσης για την ανάπτυξη εφαρμογών διαδικτύου επεκτείνοντας τις δυνατότητες του GWT. Οι προγραμματιστές μπορούσαν πλέον να δημιουργούν εφαρμογές διαδικτύου εξ ολοκλήρου σε Java, αξιοποιώντας εμμέσως τη δύναμη του GWT. Η εισαγωγή των server-driven UI components σηματοδότησε μια σημαντική απομάκρυνση από την παραδοσιακή ανάπτυξη εφαρμογών διαδικτύου η οποία ήταν προσανατολισμένη στην JavaScript.

1.2.2 Απλοποιώντας την ανάπτυξη διεπαφής χρήστη με την αρχιτεκτονική του Vaadin

Στο κέντρο της ανάπτυξης διεπαφής χρήστη (UI) βρίσκεται η έννοια των συστατικών (components). Στο Vaadin, ένα συστατικό είναι ένα αυτοτελές δομικό στοιχείο που ενσωματώνει ένα συγκεκριμένο κομμάτι λειτουργικότητας ή στοιχείο διεπαφής χρήστη. Τα συστατικά μπορεί να είναι τόσο απλά όσο ένα κουμπί ή τόσο σύνθετα όσο ένα πλέγμα δεδομένων. Αυτά τα συστατικά μπορούν να οργανωθούν σε μια ιεραρχία, με τα συστατικά-γονείς να περιέχουν συστατικά-παιδιά, δημιουργώντας μια δομημένη αναπαράσταση της διεπαφής χρήστη. Τα συστατικά στο Vaadin μπορούν να είναι επαναχρησιμοποιήσιμα. Οι προγραμματιστές μπορούν να δημιουργήσουν προσαρμοσμένα συστατικά για την αναπαράσταση οποιουδήποτε τμήματος του UI και να τα χρησιμοποιήσουν σε πολλά Views μέσα σε μια εφαρμογή. Αυτή η δυνατότητα επαναχρησιμοποίησης όχι μόνο απλοποιεί την ανάπτυξη, αλλά εξασφαλίζει επίσης την ομοιομορφία στην διεπαφή χρήστη. Τα συστατικά του Vaadin είναι καθοδηγούμενα από συμβάντα. Αυτό σημαίνει ότι μπορούν να δημιουργούν συμβάντα (π.χ. κλικ κουμπιών, αλλαγές τιμών) και να ανταποκρίνονται σε συμβάντα που προκαλούνται από τους χρήστες ή τον διακομιστή. Ο χειρισμός συμβάντων είναι ένα κρίσιμο κομμάτι της δημιουργίας διαδραστικών εφαρμογών ιστού και το μοντέλο συστατικών του Vaadin απλοποιεί αυτή τη διαδικασία.
Με την αφαίρεση των χαμηλού επιπέδου HTML, CSS και JavaScript, το Vaadin απαλλάσσει τους προγραμματιστές από την πολυπλοκότητα της ανάπτυξης ιστοσελίδων. Αντ' αυτού, μπορούν να επικεντρωθούν στη σχεδίαση στοιχείων UI και στον ορισμό της συμπεριφοράς τους χρησιμοποιώντας τη Java, μια γλώσσα με την οποία είναι ήδη εξοικειωμένοι. Αυτή η αλλαγή προοπτικής επιταχύνει σημαντικά τον κύκλο ανάπτυξης. Το Vaadin ενθαρρύνει την τμηματοποιημένη ανάπτυξη μέσω της αρχιτεκτονικής του που βασίζεται σε συστατικά. Αυτή η τμηματοποίηση προωθεί τον καθαρό διαχωρισμό του κώδικα, διευκολύνοντας τη διαχείριση, την ενημέρωση και την επέκταση της εφαρμογής με την πάροδο του χρόνου. Οι

προγραμματιστές μπορούν να διασφαλίσουν μια ομοιόμορφη εμφάνιση και αίσθηση σε ολόκληρη την εφαρμογή με την επαναχρησιμοποίηση τυποποιημένων στοιχείων. Παρόλα αυτά δίνεται η δυνατότητα στον προγραμματιστή να προσαρμόσει την μορφοποίηση των διαφόρων συστατικών και κατά συνέπεια της διεπαφής χρήστη χρησιμοποιώντας εάν επιθυμεί με HTML, CSS η και JavaScript.

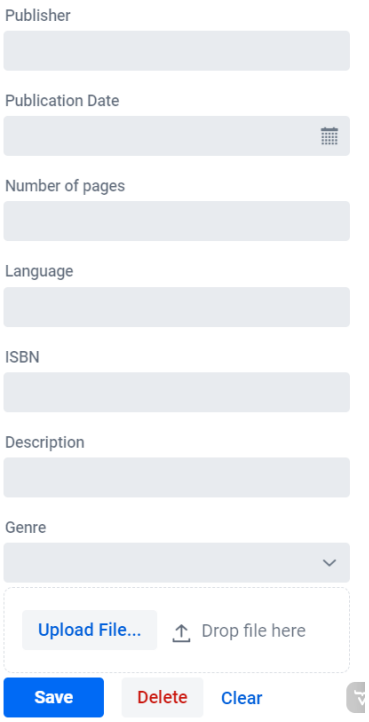
1.2.3 Επικοινωνία πελάτη διακομιστή

Το Vaadin διαχειρίζεται την επικοινωνία πελάτη – διακομιστή απρόσκοπτα, επιτρέποντας στους προγραμματιστές να επικεντρωθούν στην κατασκευή της διεπαφής χρήστη και της λογικής της εφαρμογής. Οι διαδικτυακές εφαρμογές Vaadin ξεκινούν από την πλευρά του διακομιστή. Όταν ένας χρήστης αποκτά πρόσβαση σε μια εφαρμογή Vaadin στο πρόγραμμα περιήγησης ιστού του, η αρχική απόδοση της σελίδας γίνεται στον διακομιστή. Ο διακομιστής παράγει κώδικα HTML, JavaScript και CSS που αντιπροσωπεύουν την αρχική κατάσταση της διεπαφής χρήστη. Μόλις η αρχική σελίδα HTML σταλεί στο πρόγραμμα περιήγησης του πελάτη, αναλαμβάνει η βιβλιοθήκη JavaScript της Vaadin στην πλευρά του πελάτη. Αυτή η βιβλιοθήκη JavaScript αρχικοποιεί τα στοιχεία πλευράς πελάτη, τα οποία αντικατοπτρίζουν τα στοιχεία πλευράς διακομιστή. Δημιουργεί επίσης ένα WebSocket ή ένα παρόμοιο κανάλι αμφίδρομης επικοινωνίας με τον διακομιστή. Όταν ένας χρήστης αλληλεπιδρά με την εφαρμογή (π.χ. πατάει ένα κουμπί, εισάγει δεδομένα σε ένα πεδίο κειμένου), οι αλληλεπιδράσεις αυτές καταγράφονται στην πλευρά του πελάτη. Η βιβλιοθήκη JavaScript στην πλευρά του πελάτη ανιχνεύει αυτά τα συμβάντα και στέλνει μηνύματα στον διακομιστή μέσω του καθιερωμένου καναλιού επικοινωνίας. Στην πλευρά του διακομιστή, το Vaadin επεξεργάζεται αυτά τα εισερχόμενα μηνύματα. Ερμηνεύει τις ενέργειες του χρήστη και ενημερώνει ανάλογα την κατάσταση στην πλευρά του διακομιστή. Εδώ βρίσκεται η επιχειρησιακή λογική της εφαρμογής. Ο διακομιστής μπορεί να εκτελεί ερωτήματα στη βάση δεδομένων, να εκτελεί υπολογισμούς ή να λαμβάνει αποφάσεις με βάση τις ενέργειες του χρήστη. Μετά την επεξεργασία των μηνυμάτων από τον πελάτη, το Vaadin διασφαλίζει ότι η κατάσταση στην πλευρά του διακομιστή ταιριάζει με την κατάσταση στην πλευρά του πελάτη. Τυχόν αλλαγές στην κατάσταση συγχρονίζονται μεταξύ του διακομιστή και του πελάτη για να διατηρηθεί η συνέπεια. Εάν οι ενέργειες του χρήστη έχουν ως αποτέλεσμα αλλαγές στη διεπαφή χρήστη (π.χ. εμφάνιση ή απόκρυψη στοιχείων, ενημέρωση δεδομένων), το Vaadin ενημερώνει τα συστατικά από την πλευρά του διακομιστή και παράγει νέα HTML, JavaScript και CSS για την αναπαράσταση αυτών των αλλαγών. Αυτές οι ενημερώσεις αποστέλλονται στη συνέχεια στον πελάτη. Στην πλευρά του πελάτη, η βιβλιοθήκη JavaScript του Vaadin λαμβάνει και εφαρμόζει αυτές τις ενημερώσεις στα στοιχεία της πλευράς του πελάτη. Αυτή η διαδικασία διασφαλίζει ότι η διεπαφή χρήστη παραμένει συγχρονισμένη με την κατάσταση στην πλευρά του διακομιστή. Το Vaadin χρησιμοποιεί ασύγχρονη επικοινωνία για να ελαχιστοποιήσει την ανάγκη για επαναφορτώσεις ολόκληρης της σελίδας. Ενημερώνονται μόνο τα τμήματα της σελίδας που έχουν αλλάξει, με αποτέλεσμα την ομαλότερη εμπειρία του χρήστη. Η αρχιτεκτονική του Vaadin που βασίζεται σε συμβάντα επιτρέπει στους προγραμματιστές να ορίζουν ακροατές για διάφορα συμβάντα (π.χ. κλικ κουμπιών, αλλαγές τιμών). Όταν συμβαίνουν τέτοια συμβάντα στην πλευρά του πελάτη, ενεργοποιούνται οι αντίστοιχοι ακροατές στην πλευρά του διακομιστή, επιτρέποντας στους προγραμματιστές να ανταποκρίνονται στις ενέργειες του χρήστη. Παρακάτω θα αναφερθεί ένα παράδειγμα χρήσης ακροατών. Το Vaadin φροντίζει και για τις πτυχές ασφάλειας που

σχετίζονται με την επικοινωνία πελάτη-διακομιστή, όπως η προστασία από επιθέσεις πλαστογράφησης αιτήσεων cross-site (CSRF) και η διασφάλιση ότι μόνο εξουσιοδοτημένοι χρήστες μπορούν να έχουν πρόσβαση σε ορισμένα τμήματα της εφαρμογής

1.2.4 Βασικές έννοιες της τεχνολογίας Vaadin

Όπως αναφέρθηκε και παραπάνω υπάρχουν τα συστατικά που αποτελούν τη διεπαφή χρήστη μιας εφαρμογής Vaadin. Αντιπροσωπεύουν διαδραστικά και οπτικά συστατικά, όπως κουμπιά, πεδία κειμένου, πίνακες και διαγράμματα. Το Vaadin προσφέρει μια εκτεταμένη βιβλιοθήκη προ-κατασκευασμένων στοιχείων διεπαφής χρήστη που μπορούν να προσαρμοστούν ώστε να ανταποκρίνονται στις ανάγκες της εκάστοτε εφαρμογής. Επιπλέον, οι προγραμματιστές μπορούν να δημιουργήσουν προσαρμοσμένα στοιχεία διεπαφής χρήστη για να ενθυλακώσουν συγκεκριμένες λειτουργίες, προωθώντας την επαναχρησιμοποίηση και την τμηματικοποίηση. Μερικά παραδείγματα τέτοιων οπτικών συστατικών είναι τα TextFields, τα buttons, τα comboboxes κ.α. Ένα από τα μέρη στα οποία έχουν χρησιμοποιηθεί αυτά τα συστατικά είναι η φόρμα [Εικόνα 1] η οποία υπάρχει στην σελίδα του Administrator (AdministratorView) και υλοποιείται στην κλάση BookForm που είναι παρακάτω.



Εικόνα 1: Φόρμα βιβλίου

```
public class BookForm extends FormLayout {
    Binder<Book> binder = new BeanValidationBinder<>(Book.class);
    TextField title = new TextField("Title");
    TextField wrote = new TextField("Authors");
    TextField publisher = new TextField("Publisher");
    DatePicker publicationDate = new DatePicker("Publication Date");
    IntegerField pages = new IntegerField("Number of pages");
    TextField language = new TextField("Language");
    TextField isbn = new TextField("ISBN");
    TextField description = new TextField("Description");
    ComboBox<Genre> genre = new ComboBox<>("Genre");
    Upload upload;

    Button save = new Button("Save");
    Button delete = new Button("Delete");
    Button clear = new Button("Clear");
    private Book book;
    private AuthorService authorService;
    private AppUserService appUserService;
    private AppUserBookService appUserbookService;
    byte[] image;
    …}
```

Ένα σημαντικό χαρακτηριστικό είναι τα Layouts τα οποία παίζουν καθοριστικό ρόλο στην οργάνωση και διάταξη των συστατικών του περιβάλλοντος εργασίας σε μια ιστοσελίδα. Το Vaadin παρέχει μια ποικιλία διαχειριστών διάταξης, συμπεριλαμβανομένων των VerticalLayout, HorizontalLayout και GridLayout, για τον έλεγχο της τοποθέτησης και της ευθυγράμμισης των συστατικών. Αυτοί οι διαχειριστές διάταξης απλοποιούν τη δημιουργία ευέλικτων και καλά δομημένων διεπαφών χρήστη.
Στον κώδικα του προαναφερθέντος παραδείγματος βλέπουμε ότι η κλάση BookForm κληρονομεί το FormLayout. Αυτή η διάταξη είναι ιδανική για την οργάνωση των στοιχείων της φόρμας σε κάθετη διάταξη, η οποία είναι μια κοινή προσέγγιση για τις φόρμες. Προσθέτει αυτόματα απόσταση μεταξύ των στοιχείων της φόρμας, εξασφαλίζοντας τον οπτικό διαχωρισμό τους. Ακόμα έχει σχεδιαστεί για να λειτουργεί καλά με τις αρχές του ευέλικτου σχεδιασμού εφαρμογών διαδικτύου (responsive web design). Τέλος προσαρμόζεται σε διαφορετικά μεγέθη οθόνης, διασφαλίζοντας ότι η φόρμα παραμένει ευανάγνωστη και προσβάσιμη σε διάφορες συσκευές, από υπολογιστές έως και κινητά.
Το Vaadin υιοθετεί μια αρχιτεκτονική με γνώμονα τα συστατικά, όπου οι αλληλεπιδράσεις του χρήστη και οι ενέργειες των προκαλούν συμβάντα. Ο χειρισμός συμβάντων επιτρέπει στους προγραμματιστές να ανταποκρίνονται σε αυτά τα συμβάντα, επιτρέποντας δυναμικές και ευέλικτες διεπαφές χρήστη. Τα συμβάντα μπορεί να κυμαίνονται από κλικ κουμπιών και αλλαγές επιλογής μέχρι προσαρμοσμένες αλληλεπιδράσεις. Ο χειρισμός συμβάντων από την πλευρά του διακομιστή του Vaadin εξαλείφει την ανάγκη για εκτεταμένο κώδικα από την πλευρά του πελάτη (client-side), απλοποιώντας τη λογική της εφαρμογής.

```
private Upload uploadImage(Book book) {
    MemoryBuffer buffer = new MemoryBuffer();
    final byte[][] img = {new byte[1]};
   upload = new Upload(buffer);
    upload.setAcceptedFileTypes("image/jpeg", "image/jpg", "image/png", "image/gif");

    upload.addSucceededListener(event -> {
        InputStream fileData = buffer.getInputStream();
        try {
            img[0] = IOUtils.toByteArray(fileData);
            if (book != null) {
                book.setImg(img[0]);
            } else {
                 image = img[0];
            }
        } catch (IOException e) {
            e.printStackTrace();
        }
    });
    return upload;
```

Συνεχίζοντας την ανάλυση της κλάσης BookForm συναντάμε την μέθοδο uploadImage() η οποία χρησιμοποιείται για τη δημιουργία και τη ρύθμιση παραμέτρων ενός συστατικού Vaadin

Upload για τη μεταφόρτωση εικόνων. Στο συγκεκριμένο παράδειγμα συναντάμε το συμβάν upload.addSucceededListener(event -> { ... }); Σε αυτό το σημείο προστίθεται ένας ακροατής στο συστατικό Upload για τον χειρισμό του συμβάντος, όταν η μεταφόρτωση ενός αρχείου είναι επιτυχής. Μέσα στην έκφραση λάμδα, εκτελούνται κάποιες ενέργειες με σκοπό την επιτυχή αποθήκευση της εικόνας στην βάση δεδομένων.

Η δέσμευση δεδομένων (data binding) είναι ένα ισχυρό χαρακτηριστικό του Vaadin που διευκολύνει το συγχρονισμό δεδομένων μεταξύ των συστατικών της διεπαφής χρήστη και των μοντέλων δεδομένων backend. Απλοποιεί τη διαδικασία ενημέρωσης των στοιχείων διεπαφής χρήστη με αλλαγές δεδομένων και αντίστροφα. Με τη δέσμευση στοιχείων διεπαφής χρήστη απευθείας σε πηγές δεδομένων, οι προγραμματιστές μπορούν να διασφαλίσουν τη συνέπεια μεταξύ της διεπαφής χρήστη και των υποκείμενων δεδομένων.

Στην παρακάτω μέθοδο saveBook() της κλάσης BookForm βλέπουμε την χρήση της δέσμευσης δεδομένων με την βοήθεια ενός BeanValidationBinder το οποίο χρησιμοποιείται για τη σύνδεση πεδίων φόρμας με ιδιότητες ενός αντικειμένου δεδομένων Book. Η μέθοδος withConverter καθορίζει ότι πρέπει να χρησιμοποιηθεί ένας StringToAuthorSetConverter για τη μετατροπή της τιμής του πεδίου στον κατάλληλο τύπο δεδομένων (ένα σύνολο συγγραφέων σε αυτή την περίπτωση). Στη συνέχεια, η μέθοδος bind συσχετίζει το πεδίο wrote με τις μεθόδους getWrote και setWrote της κλάσης Book, επιτρέποντας τον αυτόματο συγχρονισμό μεταξύ του πεδίου και του αντικειμένου Book. Η γραμμή binder.bindInstanceFields(this); λέει στο binder να δεσμεύσει αυτόματα τα πεδία της τρέχουσας κλάσης (τα πεδία της φόρμας) με τις αντίστοιχες ιδιότητες του αντικειμένου Book. Αυτή είναι μια μέθοδος ευκολίας για την αυτόματη ρύθμιση της δέσμευσης δεδομένων για όλα τα πεδία με τα ίδια ονόματα. Τέλος η γραμμή binder.readBean(book); συμπληρώνει τα πεδία της φόρμας με δεδομένα από το παρεχόμενο αντικείμενο book. Ουσιαστικά αρχικοποιεί τη φόρμα με τις τιμές από το αντικείμενο book, επιτρέποντας στους χρήστες να βλέπουν ή να επεξεργάζονται τις πληροφορίες του βιβλίου στα πεδία της.

```
public void setBook(Book book) {
    this.book = book;
    StringToAuthorSetConverter converter =new StringToAuthorSetConverter(authorService);
    binder.forField(wrote)
            .withConverter(converter)
            .bind(Book::getWrote, Book::setWrote);
    if (book != null && image != null) {
        book.setImg(image);
        image = null;
    }
    binder.bindInstanceFields(this);
    binder.readBean(book);
}
```

Ο Navigator του Vaadin είναι ένα κρίσιμο συστατικό για την κατασκευή εφαρμογών μίας σελίδας (SPA) με πλοήγηση στην πλευρά του πελάτη. Επιτρέπει στους προγραμματιστές να ορίζουν πολλαπλές προβολές ή σελίδες μέσα σε μια εφαρμογή και να διαχειρίζονται απρόσκοπτα την πλοήγησή τους. Το Navigator απλοποιεί τη δημιουργία σύνθετων ροών πλοήγησης, βελτιώνοντας την εμπειρία του χρήστη. Στην διαδικτυακή εφαρμογή η οποία υλοποιήθηκε στα πλαίσια αυτής της εργασίας όπως φαίνεται στον παρακάτω κώδικα χρησιμοποιήθηκε το συστατικό NavBar, ως Navigator το οποίο ενδείκνυται και εφαρμογές με μικρό αριθμό σελίδων.

```
public MainLayout(@Autowired AuthenticationService authenticationService) {
    this.authenticationService = authenticationService;
    addHeaderContent();
    H1 title = new H1("Literature Network");
    title.getStyle().set("font-size", "var(--lumo-font-size-l)")
            .set("left", "var(--lumo-space-l)").set("margin", "0")
            .set("position", "absolute");
    Tabs tabs = getTabs();
    AvatarMenuBar avatarMenuBar = new AvatarMenuBar(authenticationService);
    addToNavbar(tabs, avatarMenuBar);
    authenticationService.createAdminUser();
}
```

Ένα από τα καθοριστικά χαρακτηριστικά του Vaadin είναι το μοντέλο επεξεργασίας από την πλευρά του διακομιστή. Στις εφαρμογές Vaadin, το μεγαλύτερο μέρος της λογικής της εφαρμογής και της διαχείρισης της κατάστασης πραγματοποιείται στο διακομιστή. Αυτή η προσέγγιση απλοποιεί την ανάπτυξη, διασφαλίζει την ασφάλεια και διευκολύνει τη δημιουργία ανταποκρινόμενων εφαρμογών διαδικτύου. Η επεξεργασία από την πλευρά του διακομιστή επιτρέπει επίσης την αποτελεσματική επικοινωνία πελάτη-διακομιστή, μειώνοντας την ανάγκη για χειροκίνητη κωδικοποίηση AJAX.

1.3 Επιλογή τεχνολογιών για την δημιουργία της διαδικτυακής εφαρμογής

Κατά τον σχεδιασμό της διαδικτυακής εφαρμογής αντιμετωπίστηκαν κρίσιμες αποφάσεις σχετικά με την επιλογή των τεχνολογιών. Αυτές οι επιλογές θα διαμόρφωναν τα θεμέλια της εφαρμογής, επηρεάζοντας την απόδοση, την επεκτασιμότητα και την εμπειρία ανάπτυξης. Παρακάτω θα αναλυθούν οι λόγοι που επιλέχτηκαν αυτές οι συγκεκριμένες τεχνολογίες, λαμβάνοντας υπόψη τα βασικά χαρακτηριστικά και τις απαιτήσεις της εφαρμογής.

1.3.1 Επιλογή Βάσης Δεδομένων

Δεδομένης της φύσης της εφαρμογής, όπου οι χρήστες μπορούν να προσθέτουν βιβλία σε διάφορες λίστες, να διαβάζουν και να υποβάλλουν κριτικές και να διατηρούν προσωπικούς λογαριασμούς, ένα σχεσιακό σύστημα διαχείρισης βάσεων δεδομένων (RDBMS) ήταν η φυσική επιλογή. Η PostgreSQL, γνωστή για την ευρωστία και την επεκτασιμότητά της,

ευθυγραμμίζεται με της απαιτήσεις της εφαρμογής. Η συμμόρφωσή της με το ACID εγγυάται την ακεραιότητα των δεδομένων, ενώ η υποστήριξή της για προηγμένους τύπους δεδομένων επιτρέπει την ευελιξία στη διαχείριση πολύπλοκων δομών δεδομένων, της λίστες χρηστών και κριτικές. Ακόμα ένα σημαντικό πλεονέκτημα της PostgresSQL είναι το γεγονός ότι είναι ένα open-source RDBMS το οποίο δεν σημαίνει μόνο ότι εξαλείφεται το κόστος αδειοδότησης αλλά παρέχει της διαφάνεια και ευελιξία, καθιστώντας την επιλογή προσιτή.

Για την ενίσχυση της διαχείρισης και τη επεκτασιμότητα της βάσης δεδομένων χρησιμοποιήθηκε το Docker ως container. Τα docker containers ενθυλακώνουν το περιβάλλον της βάσης δεδομένων PostgreSQL, συμπεριλαμβανομένων των εξαρτήσεων και των ρυθμίσεών της, σε απομονωμένες μονάδες. Τα Docker containers είναι ελαφριά και αποδοτικά, καταναλώνοντας λιγότερους πόρους συστήματος σε σύγκριση με της παραδοσιακές εικονικές μηχανές. Αυτή η αποδοτικότητα εξασφαλίζει τη βέλτιστη αξιοποίηση των πόρων του διακομιστή της, ενισχύοντας τη συνολική απόδοση της εφαρμογής. Επιπρόσθετα παρέχουν απομόνωση, διασφαλίζοντας ότι η βάση δεδομένων PostgreSQL και οι εξαρτήσεις της δεν επηρεάζουν άλλα στοιχεία της εφαρμογής ή του συστήματός της. Αυτή η απομόνωση προάγει σταθερότητα και την ασφάλεια. Συνδυάζοντας τη δύναμη και την αξιοπιστία της PostgreSQL με την ευελιξία και τη διαχειρισιμότητα του Docker, δημιουργείτε μια σταθερή βάση για τη διαχείριση της βάσης δεδομένων.

1.3.2 Επιλογή τεχνολογιών backend και frontend

Η Spring Boot επιτρέπει στους προγραμματιστές να ρυθμίζουν γρήγορα το backend, ελαχιστοποιώντας την επανάληψη ίδιου κώδικα και τη διαμόρφωση. Δεδομένης της πολυπλοκότητας της εφαρμογής, αυτή η εξορθολογισμένη προσέγγιση επιτρέπει την συγκέντρωση περισσότερο στην υλοποίηση χαρακτηριστικών και λιγότερο στη ρύθμιση της υποδομής.

Η εφαρμογή απαιτούσε ένα ποικίλο σύνολο στοιχείων διεπαφών χρήστη για εργασίες όπως η εμφάνιση λεπτομερειών βιβλίων, η διαχείριση λιστών χρηστών και η διευκόλυνση των αλληλεπιδράσεων των χρηστών. Το Vaadin παρέχει μια πλούσια βιβλιοθήκη από συστατικά UI τα οποία μπορούν να καλύψουν σε ένα βαθμό τις ανάγκες της εφαρμογής. Επίσης παρέχει απρόσκοπτη επικοινωνία μεταξύ πελάτη και διακομιστή το οποίο είναι απαραίτητο σε μια εφαρμογή όπου οι χρήστες αλληλοεπιδρούν με διάφορα χαρακτηριστικά, η ευέλικτη επικοινωνία μεταξύ του πελάτη και του διακομιστή είναι ζωτικής σημασίας. Η απρόσκοπτη επικοινωνία πελάτη-διακομιστή του Vaadin διασφαλίζει ότι οι αλληλεπιδράσεις των χρηστών καταγράφονται, επεξεργάζονται και αντικατοπτρίζονται σε πραγματικό χρόνο, βελτιώνοντας την εμπειρία του χρήστη. Συνδυάζοντας τη Spring Boot για το backend και το Vaadin για το frontend, δημιουργείται μια τεχνολογική στοίβα που δεν ανταποκρίνεται μόνο στις τρέχουσες απαιτήσεις αλλά καθιστά δυνατή και την μελλοντική ανάπτυξη της εφαρμογής.

ΚΕΦΑΛΑΙΟ 2 ΑΡΧΙΚΟΣ ΣΧΕΔΙΑΣΜΟΣ

Κάθε επιτυχημένη διαδικτυακή εφαρμογή ξεκινά με μια δομημένη διαδικασία σχεδιασμού, έναν οδικό χάρτη που καθοδηγεί τις αποφάσεις και τις ενέργειες. Έτσι το πρώτο στάδιο της

δημιουργίας της διαδικτυακής εφαρμογής ήταν η καταγραφή των λειτουργικών και μη λειτουργικών απαιτήσεων και της δημιουργίας του σχήματος της βάσης δεδομένων.

2.1 Λειτουργικές απαιτήσεις λογισμικού

Οι λειτουργικές απαιτήσεις είναι συγκεκριμένες και λεπτομερείς περιγραφές των χαρακτηριστικών, των δυνατοτήτων και των αλληλεπιδράσεων που πρέπει να έχει ένα σύστημα ή μια εφαρμογή λογισμικού για να ικανοποιεί τις ανάγκες των χρηστών του. Οι απαιτήσεις αυτές καθορίζουν τι πρέπει να κάνει το σύστημα και πώς πρέπει να συμπεριφέρεται σε απόκριση σε ενέργειες ή εισόδους του χρήστη [4]. Παρακάτω παρατίθενται οι λειτουργικές απαιτήσεις της εφαρμογής η οποία δημιουργήθηκε στα πλαίσια αυτής της εργασίας. Από τις παρακάτω λειτουργικές απαιτήσεις δεν ολοκληρώθηκαν οι δυο τελευταίες απαιτήσεις (Ειδοποιήσεις και Ανατροφοδότηση) οι οποίες θα μπορούσαν να ολοκληρωθούν σε μια μελλοντική έκδοση της εφαρμογής.

Αυθεντικοποίηση και εξουσιοδότηση χρήστη	Εγγραφή χρήστη	Οι χρήστες θα πρέπει να μπορούν να δημιουργούν λογαριασμό παρέχοντας βασικές πληροφορίες, όπως όνομα χρήστη, μια μοναδική διεύθυνση ηλεκτρονικού ταχυδρομείου και κωδικό πρόσβασης.
Αυθεντικοποίηση και εξουσιοδότηση χρήστη	Σύνδεση χρήστη	Οι εγγεγραμμένοι χρήστες θα πρέπει να μπορούν να συνδέονται με ασφάλεια χρησιμοποιώντας το όνομα χρήστη και τον προσωπικό τους κωδικό.
Αυθεντικοποίηση και εξουσιοδότηση χρήστη	Αποσύνδεση χρήστη	Οι συνδεδεμένοι χρήστες θα πρέπει να μπορούν να αποσυνδέονται.
Διαχείριση βιβλίων	Δικαιώματα Διαχειριστή	Οι διαχειριστές θα πρέπει να έχουν όλες τις λειτουργίες των απλών χρηστών και επιπρόσθετα να μπορούν να δημιουργούν, να βλέπουν, να διαγράφουν, και να ενημερώνουν βιβλία (λειτουργίες CRUD).
Διαχείριση βιβλίων	Λίστες βιβλίων	Οι χρήστες θα πρέπει να μπορούν να βλέπουν και να διαχειρίζονται τις λίστες των βιβλίων τους, όπως "Προς ανάγνωση (TBR)", "Διαβάζω αυτή τη στιγμή" (Currently Reading) και "Διαβάζω" (Read). Θα μπορούν να προσθέτουν και να αφαιρούν βιβλία από αυτές τις λίστες.

Διαχείριση βιβλίων	Αναζήτηση βιβλίων	Οι χρήστες θα πρέπει να μπορούν να αναζητούν βιβλία με βάση τον τίτλο τους.
Διαχείριση βιβλίων	Λεπτομέρειες βιβλίων	Οι χρήστες θα πρέπει να μπορούν να δουν λεπτομερείς πληροφορίες για κάθε βιβλίο, όπως είναι ο/οι συγγραφέας/εις του, η περιγραφή του, ο εκδοτικός οίκος και άλλων σχετικών πληροφοριών
Διαχείριση βιβλίων	Αξιολογήσεις βιβλίων	Οι χρήστες θα μπορούν να γράφουν αξιολογήσεις για βιβλία που έχουν διαβάσει.
Αλληλεπίδραση φίλων	Αποστολή αιτημάτων φιλίας	Οι χρήστες θα πρέπει να μπορούν να αποστέλλουν αιτήματα φιλίας σε άλλους χρήστες της πλατφόρμα με τους οποίους δεν είναι ήδη φίλοι.
Αλληλεπίδραση φίλων	Αποδοχή αιτημάτων φιλίας	Οι χρήστες θα πρέπει να μπορούν να βλέπουν και να αποδέχονται αιτήματα φιλίας από άλλους χρήστες της πλατφόρμα.
Αλληλεπίδραση φίλων	Απόρριψη αιτημάτων φιλίας	Οι χρήστες θα πρέπει να μπορούν να βλέπουν και να απορρίπτουν αιτήματα φιλίας σε άλλους χρήστες της πλατφόρμα.
Αλληλεπίδραση φίλων	Διάβασμα αξιολογήσεων	Οι χρήστες θα πρέπει να μπορούν να βλέπουν τις αξιολογήσεις βιβλίων των φίλων τους.
Διαχείριση προφίλ χρήστη	Προφίλ χρήστη	Κάθε χρήστης θα πρέπει να έχει μια σελίδα προφίλ που να εμφανίζει τις βασικές πληροφορίες του.
Διαχείριση προφίλ χρήστη	Επεξεργασία προφίλ	Οι χρήστες θα πρέπει να μπορούν να επεξεργάζονται τις πληροφορίες του προφίλ τους.
Διαχείριση προφίλ χρήστη	Αλλαγή κωδικού πρόσβασης	Οι χρήστες θα πρέπει να έχουν τη δυνατότητα να αλλάζουν τον κωδικό πρόσβασης του λογαριασμού τους.
Ειδοποιήσεις		Οι χρήστες θα πρέπει να έχουν την δυνατότητα να λαμβάνουν ειδοποιήσεις όταν κάποιος φίλος τους κάνει μια αξιολόγηση σε ένα βιβλίο ή όταν λαμβάνουν ένα αίτμα φιλίας
Ανατροφοδότηση		Οι χρήστες θα πρέπει να έχουν την δυνατότητα να γράφουν σχόλια για πιθανές αστοχίες που εντοπίζουν στο

		πρόγραμμα. Αυτά τα σχόλια θα πρέπει να εμφανίζονται στους διαχειριστές.

2.2 Μη λειτουργικές απαιτήσεις λογισμικού

Οι μη λειτουργικές απαιτήσεις καθορίζουν τα χαρακτηριστικά, τους περιορισμούς και τις ιδιότητες που πρέπει να παρουσιάζει ένα σύστημα ή μια εφαρμογή λογισμικού. Οι απαιτήσεις αυτές επικεντρώνονται στη συνολική απόδοση, τη χρηστικότητα, την ασφάλεια, την αξιοπιστία και άλλες πτυχές που επηρεάζουν τη συμπεριφορά του συστήματος και την εμπειρία του χρήστη. Από τις παρακάτω αρχικές μη λειτουργικές απαιτήσεις αυτές που δεν συναντήθηκαν ολοκληρωτικά στην εφαρμογή η οποία υλοποιήθηκε στα πλαίσια αυτής της εργασίας ήταν αυτή της ασφάλειας δεδομένων. Σε μια μελλοντική έκδοση της εφαρμογής θα μπορούσε ο κωδικός του χρήστη να αποθηκεύεται κρυπτογραφημένος όταν ο χρήστης εγγράφεται και να κρυπτογραφείται και όταν ο χρήστης προσπαθεί να αυθεντικοποιηθεί στην εφαρμογή και όταν επιθυμεί να αλλάξει κωδικό (σε αυτή την περίπτωση ο χρήστης πρέπει να εισάγει το παλιό του κωδικό). Για την κρυπτογραφηθεί ο κωδικός του χρήστη θα μπορούσε να χρησιμοποιηθεί η μέθοδος **BCryptPasswordEncoder**().

```
@Bean public PasswordEncoder encoder() {
return new BCryptPasswordEncoder();
}
```

Το Java Bean PasswordEncoder θα μπορούσε να εισαχθεί μετέπειτα στην κλάση AuthenticationService [6].

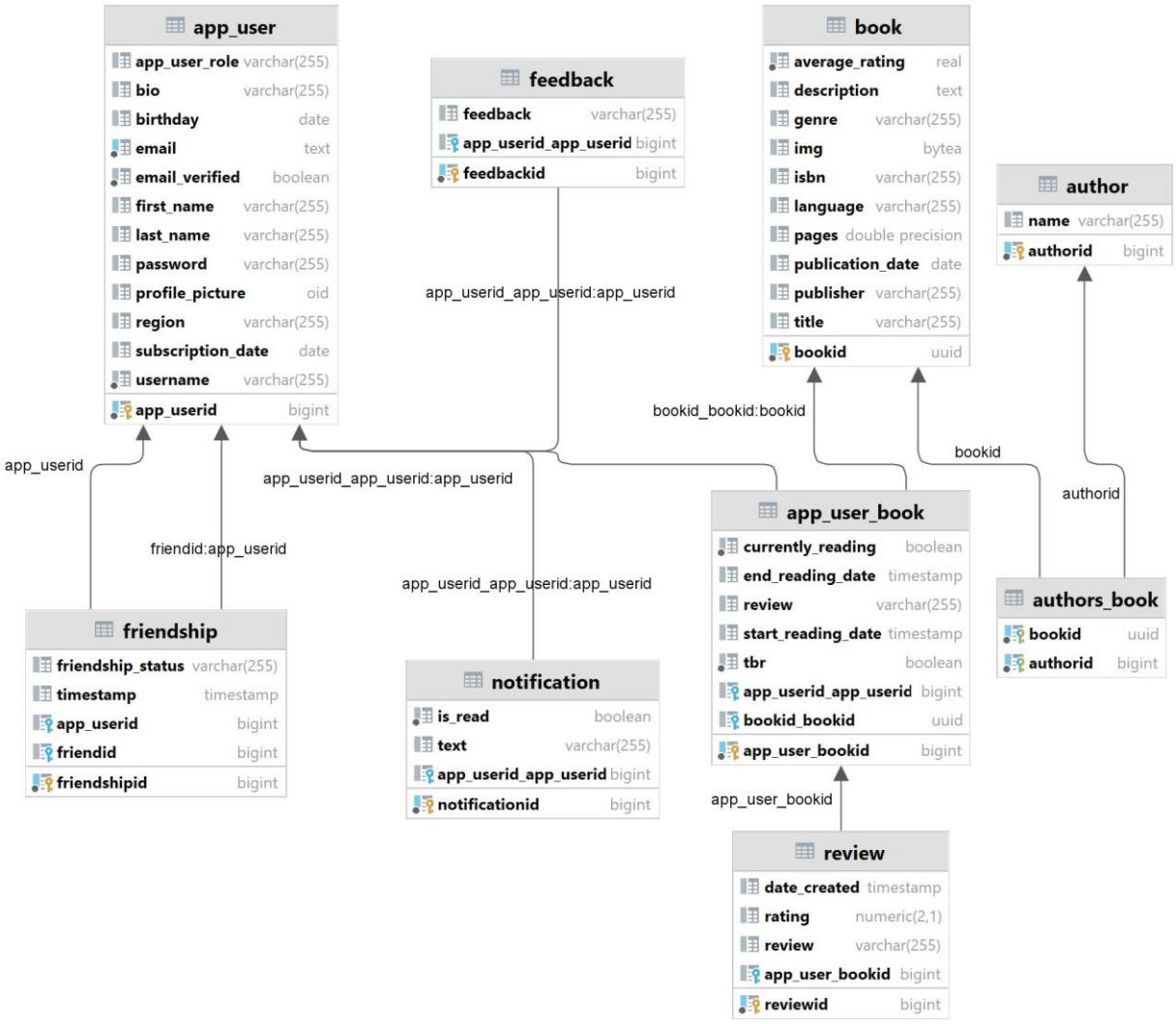
Επιδόσεις	Ανταπόκριση	Η εφαρμογή πρέπει να παρέχει μια ευέλικτη διεπαφή χρήστη, διασφαλίζοντας ότι οι σελίδες φορτώνονται γρήγορα και οι αλληλεπιδράσεις των χρηστών είναι ομαλές
Ασφάλεια	Ασφάλεια δεδομένων	Τα δεδομένα των χρηστών, συμπεριλαμβανομένων των κωδικών πρόσβασης και των ευαίσθητων πληροφοριών, πρέπει να

		αποθηκεύονται και να διαβιβάζονται με ασφάλεια χρησιμοποιώντας τεχνικές κρυπτογράφησης και κατακερματισμού για την προστασία από παραβιάσεις δεδομένων.
	Αυθεντικοποίηση	Η διαδικασία ελέγχου ταυτότητας πρέπει να είναι ισχυρή και ασφαλής, αποτρέποντας τη μη εξουσιοδοτημένη πρόσβαση σε λογαριασμούς χρηστών.
	Εξουσιοδότηση	Πρέπει να εφαρμοστεί έλεγχος πρόσβασης βάσει ρόλων (RBAC), ώστε να διασφαλίζεται ότι οι χρήστες μπορούν να έχουν πρόσβαση σε πόρους και να εκτελούν ενέργειες μόνο για τις οποίες είναι εξουσιοδοτημένοι βάσει του ρόλου τους (διαχειριστής ή χρήστης).
Ανοχή σφαλμάτων		με κατάλληλο χειρισμό σφαλμάτων και μηχανισμούς εναλλαγής αποτυχίας για την αποφυγή απώλειας δεδομένων ή διακοπής της υπηρεσίας
Ευχρηστία		Η διεπαφή χρήστη θα πρέπει να είναι διαισθητική, φιλική προς τον χρήστη και προσιτή σε χρήστες με διαφορετικό τεχνικό υπόβαθρο.

2.3 Σχήμα Βάσης δεδομένων

Το σχήμα της βάσης δεδομένων είναι η δομή που αντιπροσωπεύει όλη την βάση δεδομένων [11]. Με βάση τις παραπάνω απαιτήσεις και όπως αναφέρθηκε και στο προηγούμενο κεφάλαιο επιλέχθηκε να υλοποιηθεί μια σχεσιακή βάση για τις ανάγκες της εφαρμογής. Στη εικόνα εμφανίζονται οι πίνακες της βάσης δεδομένων όπως και οι μεταξύ τους σχέσεις. Με το σύμβολο

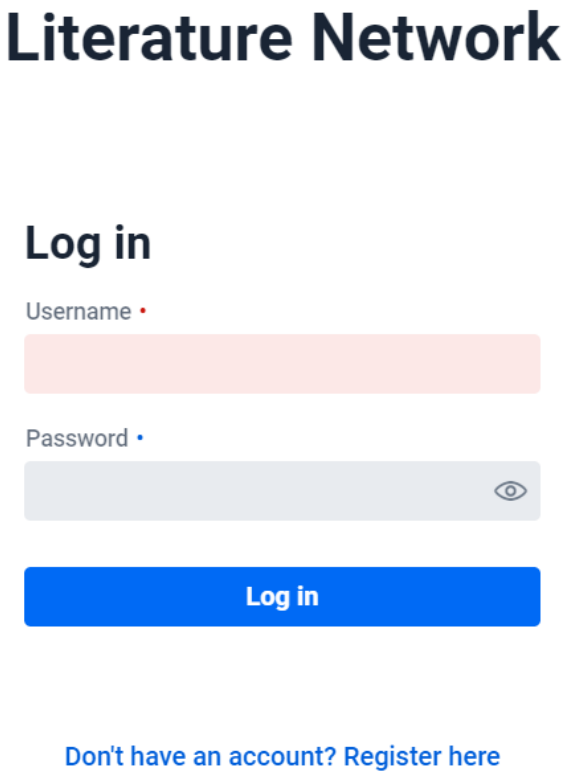
του χρυσού κλειδιού σημειώνονται τα πρωτεύοντα κλειδιά του πίνακα και με το σύμβολο μπλε κλειδιού σημειώνονται τα ξένα κλειδιά του πίνακα. Τα βέλη δείχνουν με ποια πεδία κλειδιά συσχετίζονται οι πίνακες μεταξύ τους.



ΚΕΦΑΛΑΙΟ 3 ΛΕΙΤΟΥΡΓΙΕΣ ΤΗΣ ΔΙΑΔΥΚΤΙΑΚΗΣ ΕΦΑΡΜΟΓΗΣ

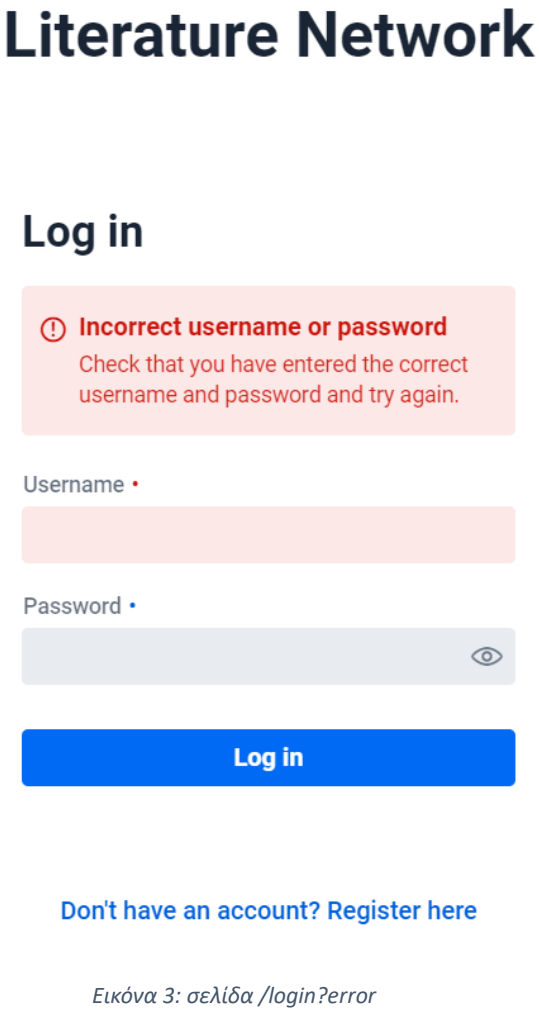
3.1 Αυθεντικοποίση και εγγραφή χρήστη

Η πρώτη σελίδα την οποία βλέπει ο χρήστης είναι η σελίδα αυθεντικοποίησης (/login). Σε αυτή την σελίδα όπως φαίνεται και στην [Εικόνα 2] ο χρήστης μπορεί να εισάγει τα στοιχεία αυθεντικοποίησης (όνομα χρήστη, κωδικός) και πατώντας το κουμπί 'Login' να ανακατευθυνθεί στην σελίδα «/» (My Books).



Εικόνα 2: Σελίδα login

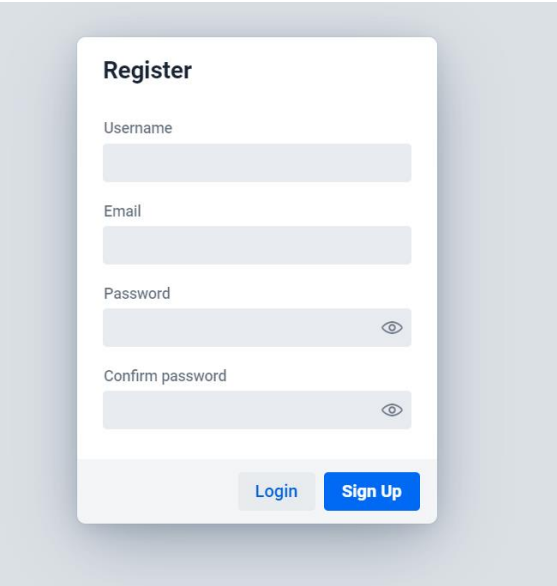
Σε περίπτωση που τα στοιχεία χρήστη είναι εσφαλμένα εμφανίζεται το κατάλληλο μήνυμα [Εικόνα 3].



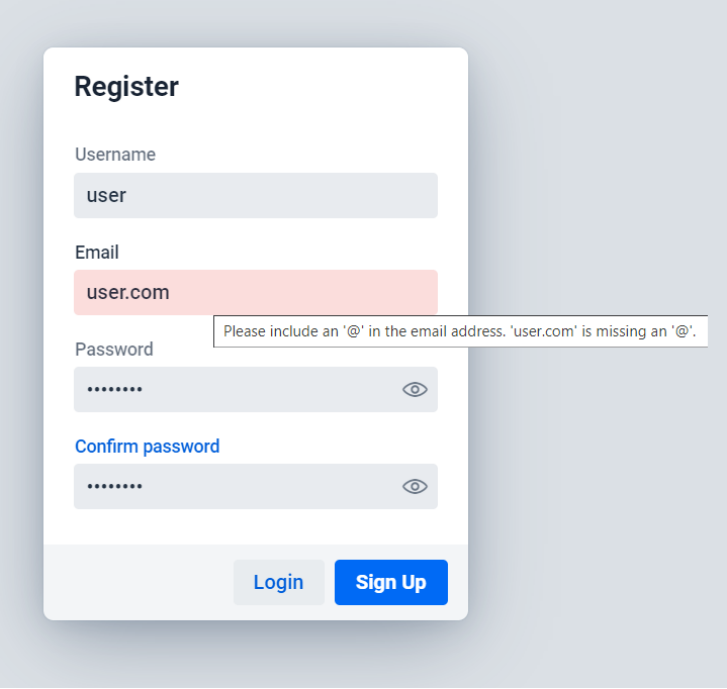
Εικόνα 3: σελίδα /login?error

Ο μη εγγεγραμμένος χρήστης μπορεί να ανακατευθυνθεί στην σελίδα /register οπού θα μπορεί να δημιουργήσει έναν λογαριασμό [Εικόνα 4]. Όπως φαίνεται και στην [Εικόνα 5] το email το οποίο θα χρησιμοποιήσει ο χρήστης θα πρέπει να είναι της μορφής «user@foo.com». Σε αντίθετη περίπτωση το χρώμα του πεδίου email γίνεται κόκκινο και πηγαίνοντας τον δείκτη του ποντικιού πάνω στο πεδίο ο χρήστης βλέπει το κατάλληλο μήνυμα. Σε περίπτωση που ο χρήστης αγνοήσει το κόκκινο πεδίο και πατήσει το κουμπί 'Sign Up' εμφανίζεται το μήνυμα «enter a valid email address». Αντίστοιχοι έλεγχοι και μηνύματα υπάρχουν για τις περιπτώσεις που ο χρήστης αφήσει κενά τα πεδία όνομα χρήστη ή email ή κωδικό πρόσβασης. Ακόμα

υπάρχει έλεγχος και το αντίστοιχο μήνυμα για το αν τα παιδία «password» και «confirm password» έχουν τις ίδιες τιμές. Εφόσον ο χρήστης έχει εγγραφεί επιτυχώς ανακατευθύνεται στην σελίδα /login από όπου μπορεί να συνδεθεί στην εφαρμογή.



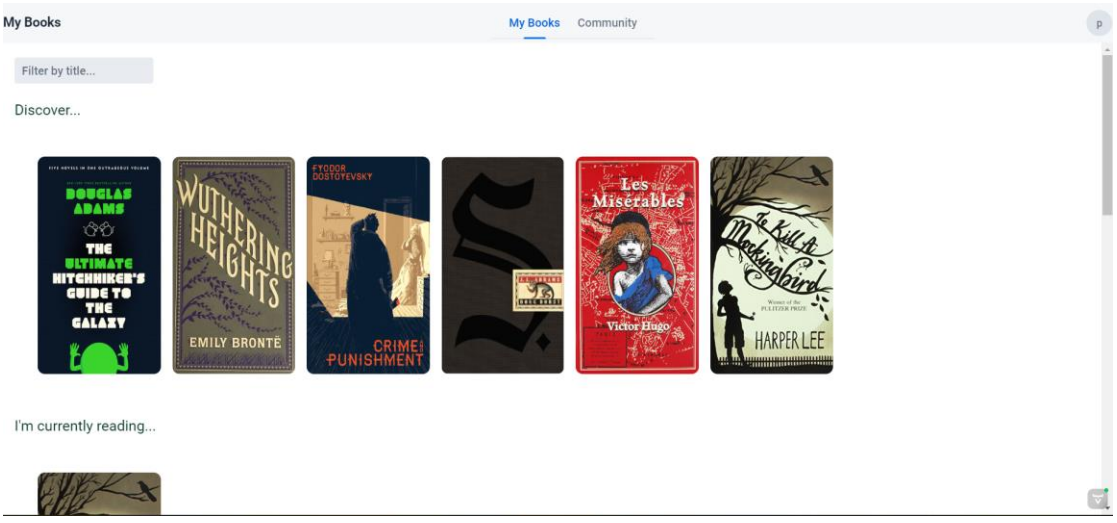
Εικόνα 4: σελίδα /register



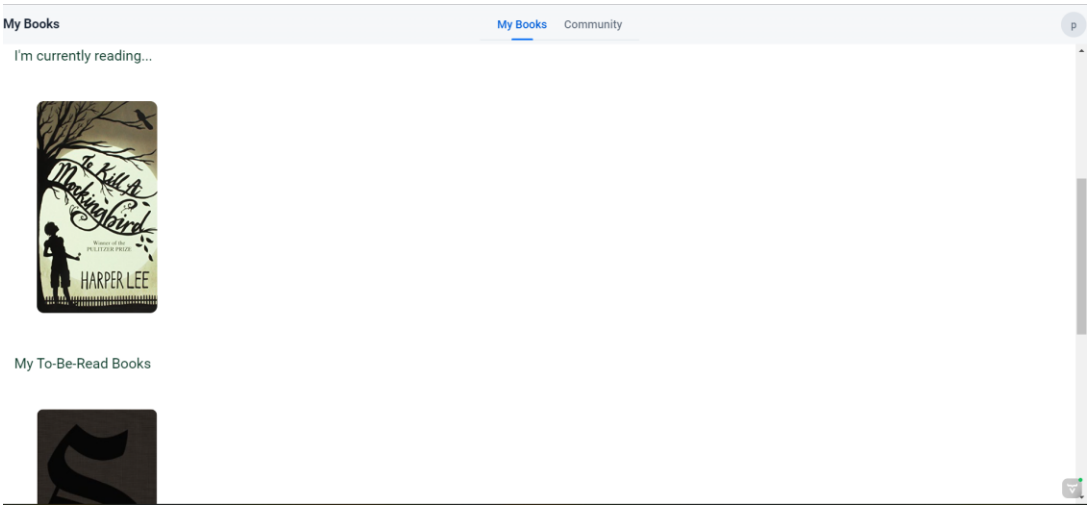
Εικόνα 5: σελίδα /register με εισαγωγή εσφαλμένης μορφής email

3.2 Σελίδα My Books

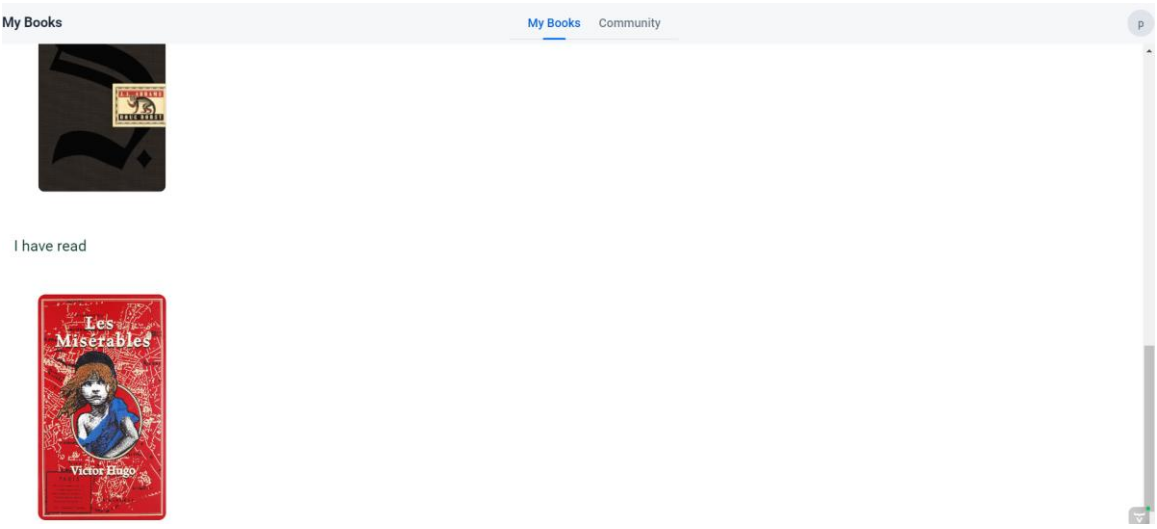
Αφού συνδεθεί ο χρήστης επιτυχώς στην διαδικτυακή εφαρμογή ανακατευθύνεται στην σελίδα /MyBooks [Εικόνα 6]. Στην σελίδα /MyBooks ο χρήστης μπορεί να δει όλα τα διαθέσιμα βιβλία της εφαρμογής στην οριζόντια λίστα «Discover…». Παρακάτω στην οριζόντια λίστα «I'm currently reading…» φαίνονται τα βιβλία που ο χρήστης διαβάζει αυτή την στιγμή και έχει προσθέσει ως «currently reading». Αντίστοιχα στις οριζόντιες λίστες «My To-Be-Read books» και «Read» [Εικόνα 7] και [Εικόνα 8] εμφανίζονται τα βιβλία τα οποία ο χρήστης θα ήθελε να διαβάσει και τα έχει προσθέσει ως «TBR» και τα βιβλία που έχει διαβάσει και τα έχει προσθέσει ως «Read» αντίστοιχα. Ακόμα ο χρήστης μπορεί να ψάξει κάποιο βιβλίο με βάση τον τίτλο του πληκτρολογώντας στο πεδίο αναζήτησης που βρίσκεται στην πάνω αριστερή γωνία της οθόνης [Εικόνα 9].



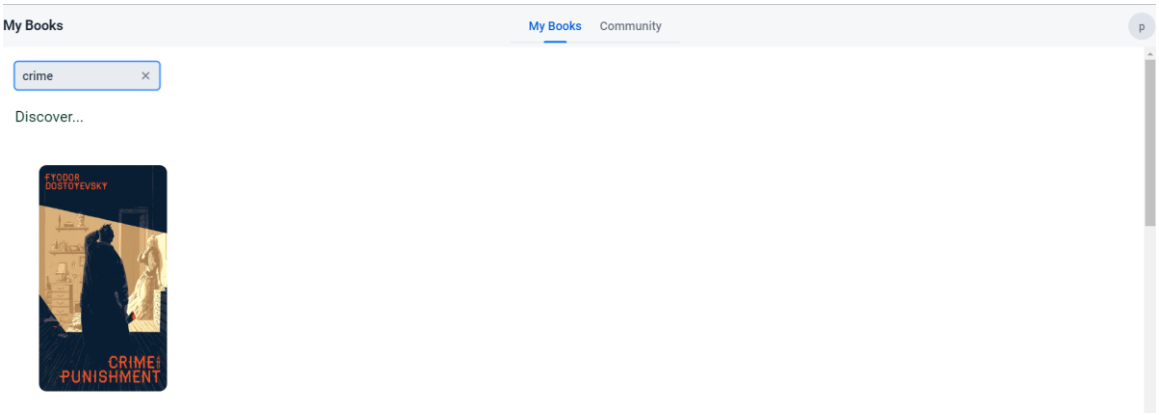
Εικόνα 6: σελίδα / .Λίστα Discover...



Εικόνα 7: σελίδα / .Λίστα I'm currently reading... και My To-Be-Read Books



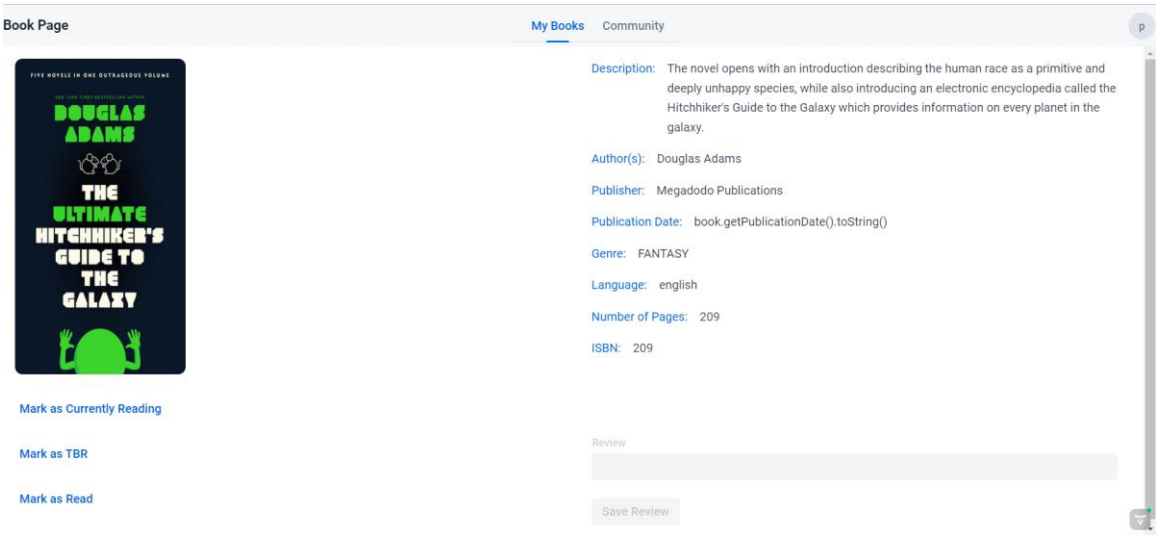
Εικόνα 8: σελίδα / .Λίστα I have read



Εικόνα 9: σελίδα / .Αναζήτηση βιβλίων

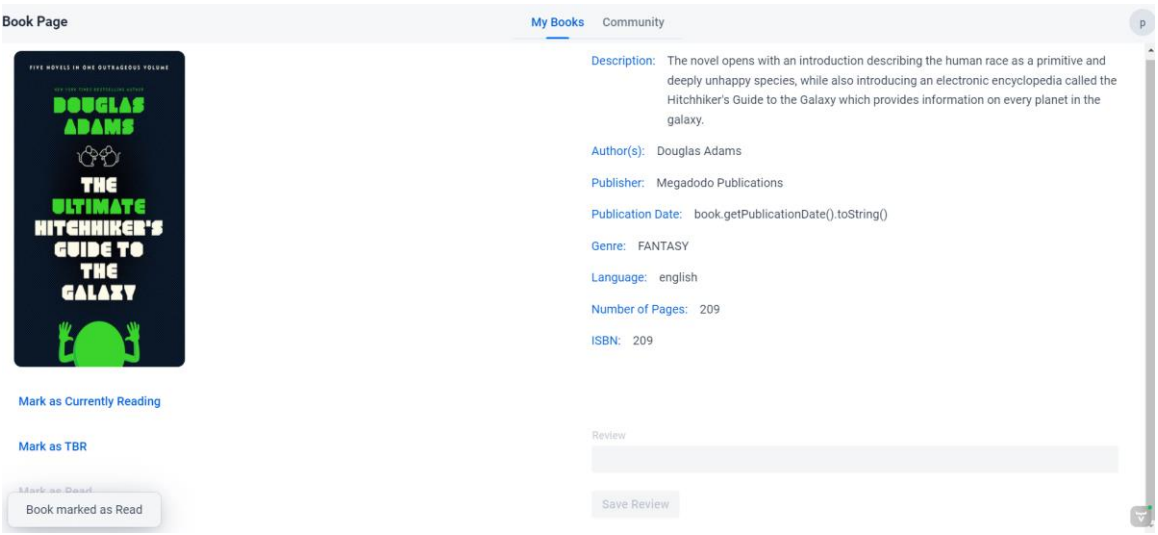
3.3 Σελίδα Βιβλίου

Κάνοντας κλικ σε μια εικόνα βιβλίου από την σελίδα My Books ο χρήστης μεταφέρεται στην σελίδα /book?bookId={bookID} από όπου μπορεί να δει περισσότερες πληροφορίες για αυτό, να το προσθέσει σε κάποια λίστα ή να γράψει κάποια κριτική για αυτό, εφόσον το έχει διαβάσει , δηλαδή το βιβλίο είναι στην λίστα «Read» του χρήστη [Εικόνα 10]. Στην[Εικόνα 10] φαίνεται ότι ο χρήστης δεν έχει προσθέσει το βιβλίο σε καμία λίστα.

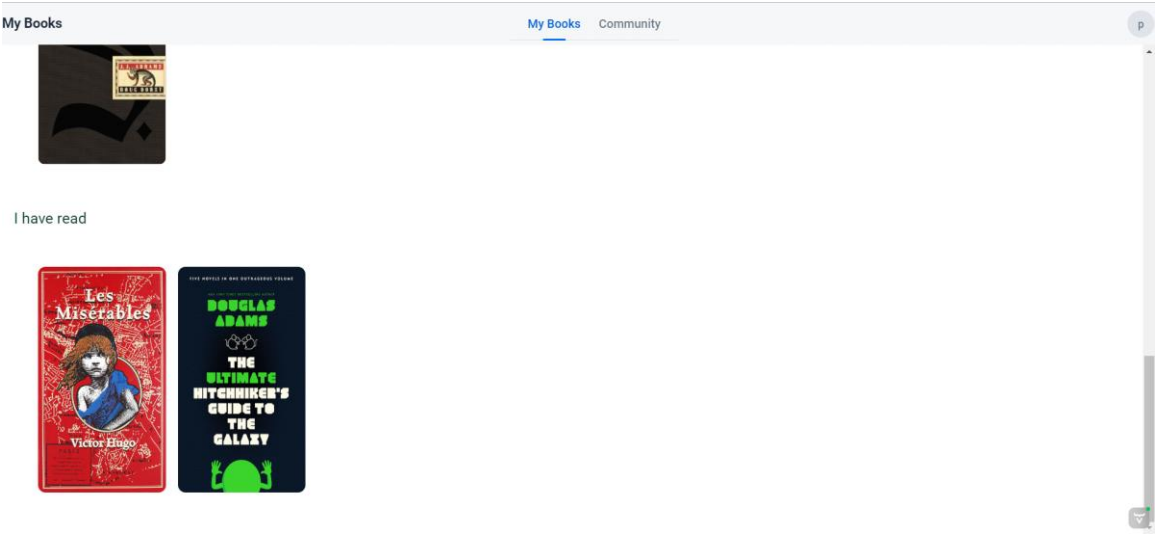


Εικόνα 10: σελίδα /book?bookId={bookID}

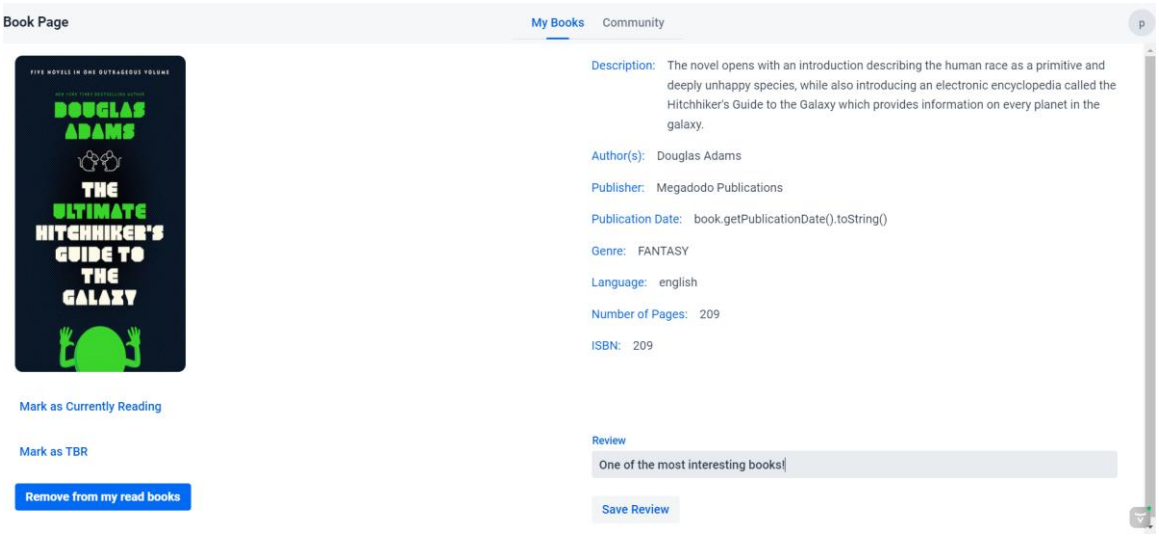
Ο χρήστης μπορεί να προσθέσει το βιβλίο σε κάποια λίστα επιλέγοντας ένα από τα «Mark as Currently Reading», «Mark as TBR» ή «Mark as Read» τα οποία βρίσκονται κάτω από την εικόνα του βιβλίου. Στην [Εικόνα 11] φαίνεται ότι ο χρήστης έχει πατήσει το κουμπί «Mark as Read» και έχει προσθέσει το βιβλίο «The hitchhicker's guide to the galaxy» στη λίστα «I have read» [Εικόνα 12]. Ακόμα στην [Εικόνα 11] βλέπουμε ότι ο χρήστης πλέων μπορεί να αφαιρέσει το βιβλίο από την λίστα «I have read» και μπορεί να γράψει μια κριτική για αυτό [Εικόνα 13]



Εικόνα 11: σελίδα /book?bookId={bookId}. Book marked as read



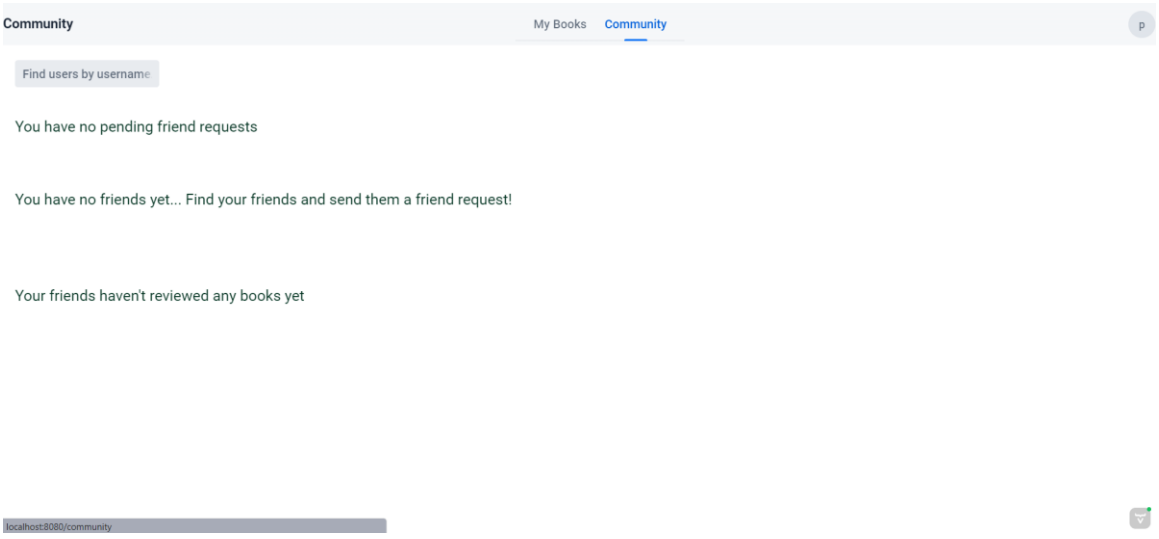
Εικόνα 12: σελίδα /book?bookId={bookID}. Βιβλίο προστέθηκε στην λίστα I have read



Εικόνα 13: σελίδα book?bookId={bookID}. Ενεργοποιημένο πεδίο review

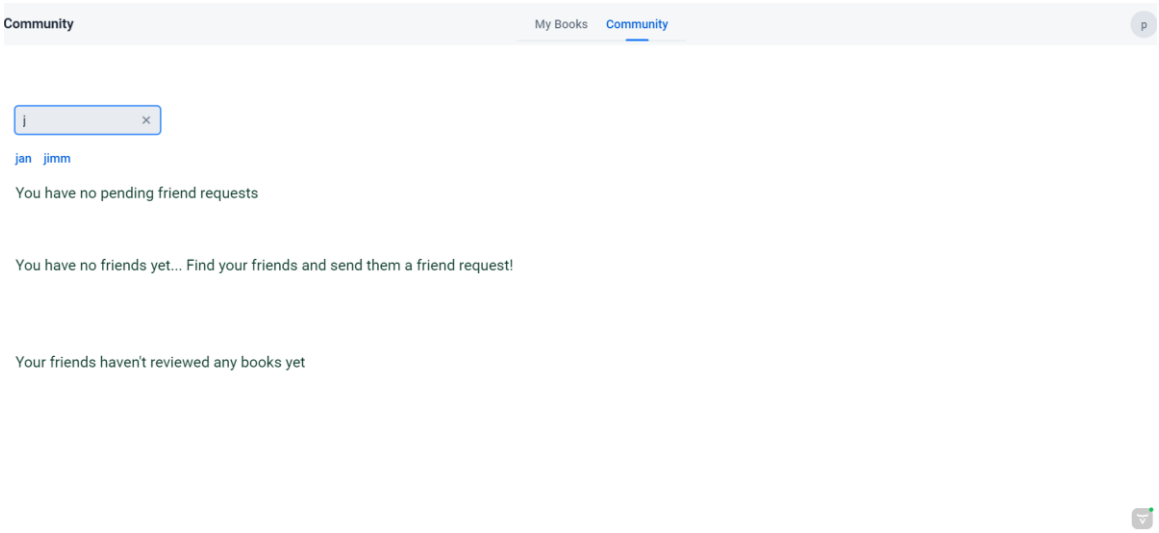
3.4 Σελίδα Community

Οι χρήστες μπορούν να στείλουν, να αποδεχτούν και να απορρίψουν αιτήματα φιλίας από την σελίδα Community. Ακόμα από εκεί μπορούν να δουν τις κριτικές για βιβλία που έχουν γράψει οι φίλοι τους. Αρχικά εφόσον ο χρήστης χρησιμοποιεί την εφαρμογή για πρώτη φορά η σελίδα Community μοιάζει με αυτή της [Εικόνα 14].



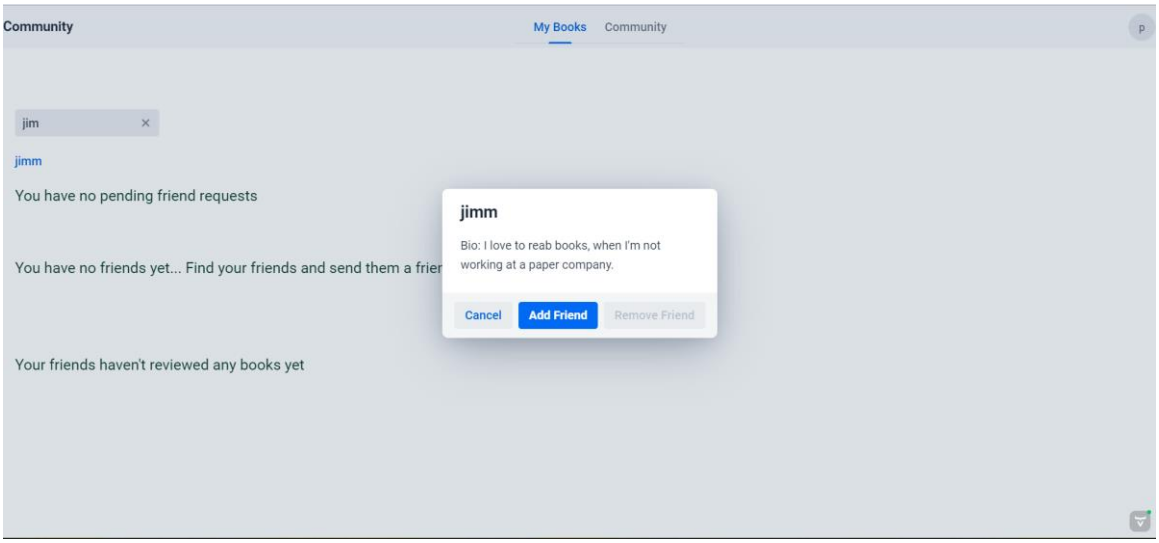
Εικόνα 14: σελίδα /community χωρίς δεδομένα

Ο συνδεδεμένος χρήστης μπορεί να αναζητήσει φίλους του που χρησιμοποιούν την ίδια εφαρμογή μέσω του πεδίου αναζήτησης χρηστών χρησιμοποιώντας το όνομα χρήστη τους [Εικόνα 15].



Εικόνα 15: σελίδα /community. Αναζήτηση χρηστών

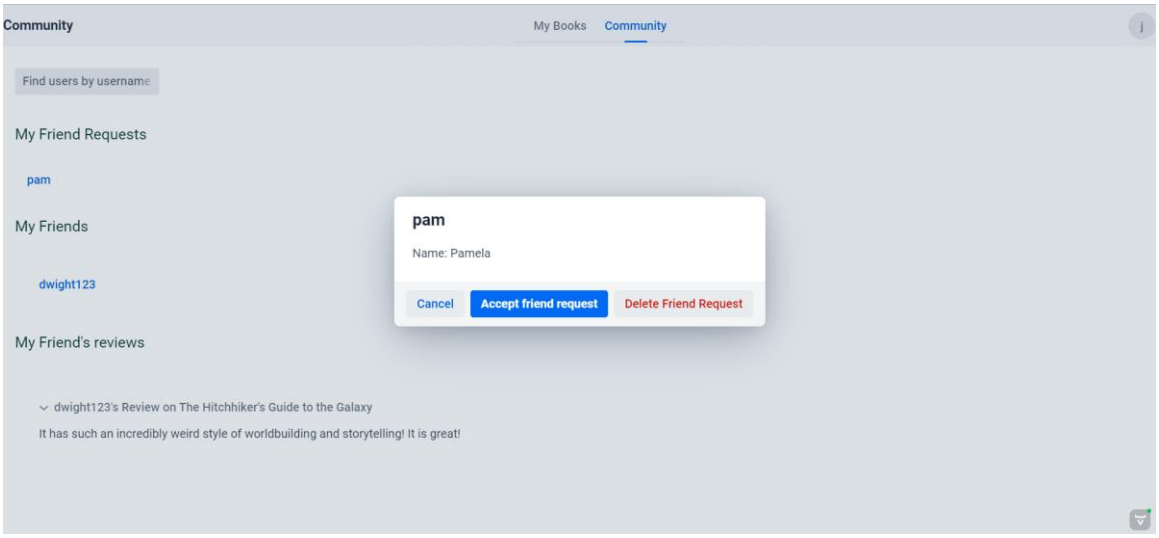
Πατώντας πάνω στο όνομα του χρήστη εμφανίζεται ένα παράθυρο από το οποίο ο χρήστης μπορεί να δει μια περιγραφή του χρήστη και να του στείλει αίτημα φιλίας [Εικόνα 16]. Από το ίδιο παράθυρο ο χρήστης μπορεί να αφαιρέσει κάποιον χρήστη από τους φίλους του.



Εικόνα 16: σελίδα /community. Αίτημα φιλίας.

Στην [Εικόνα 16] ο συνδεδεμένος χρήστης είναι ο χρήστης Jimm (όπως φαίνεται και από το εικονίδιο του (avatar) πάνω δεξιά). Ο χρήστης jimm έχει ένα αίτημα φιλίας από τον χρήστη pam. Εάν επιλέξει τον χρήστη pam θα εμφανιστεί ένα παράθυρο από το οποίο μπορεί να απορρίψει ή να δεχτεί το αίτημα φιλίας του χρήστη pam. Ακόμα στην ίδια εικόνα φαίνεται ο

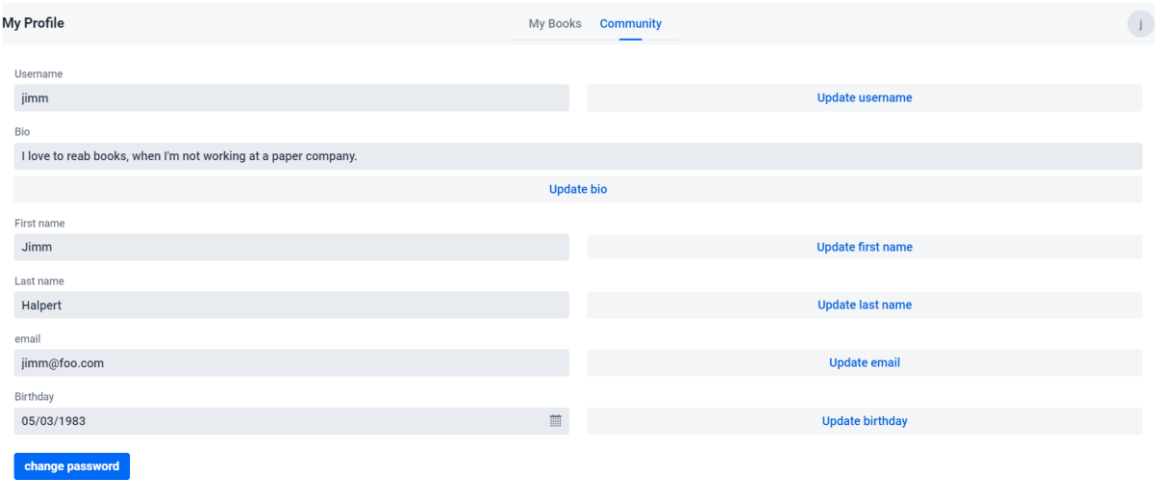
χρήστης jimm να έχει φίλο τον χρήστη dwight123 ο οποίος έχει κάνει μια αξιολόγηση για ένα βιβλίο.



Εικόνα 17: σελίδα /community. Αποδοχη/ Απόριψη αιτήματος φιλίας

3.5 Σελίδα Προσωπικών στοιχείων χρήστη

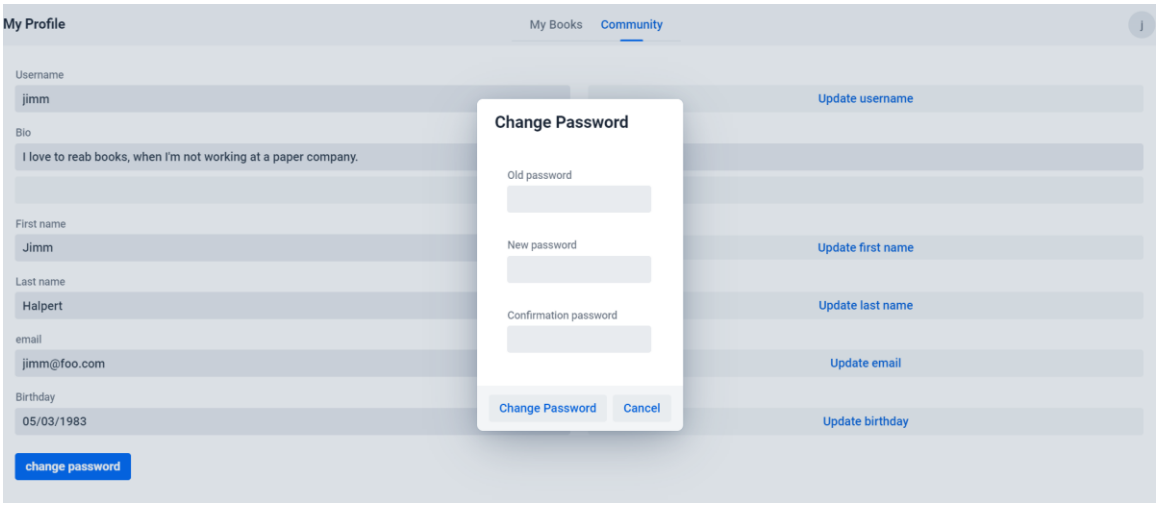
Πατώντας στο εικονίδιο του χρήστη που βρίσκεται στην πάνω δεξιά γωνία ο χρήστης μπορεί είτε να επιλέξει να ενημερώσει τα προσωπικά του στοιχεία είτε να αποσυνδεθεί από την εφαρμογή. Εάν ο χρήστης επιλέξει την επιλογή «Manage Account» τότε μεταφέρεται στην σελίδα της [Εικόνα 18]. Από αυτή την σελίδα ο χρήστης μπορεί να δει και να ενημερώσει τα προσωπικά του στοιχεία.



Εικόνα 18: σελίδα /profile

Πατώντας το κουμπί «change password» εμφανίζεται ένα παράθυρο από το οποίο ο χρήστης μπορεί να αλλάξει τον κωδικό πρόσβασης του [Εικόνα 19]. Στην φόρμα που εμφανίζεται στην [Εικόνα 19]

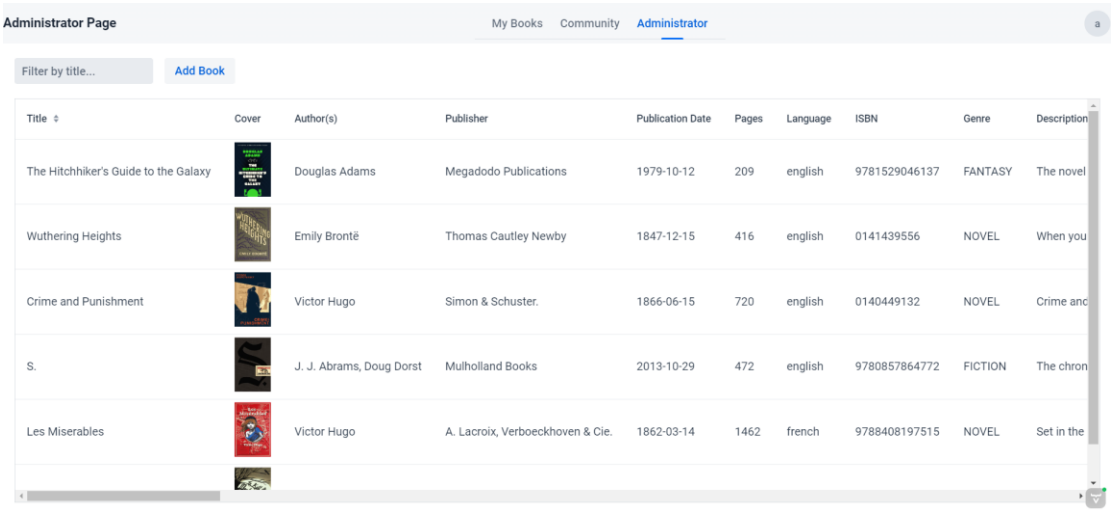
υπάρχουν οι απαραίτητοι έλεγχοι ώστε να επιβεβαιωθεί ότι ο παλιός κωδικός που θα εισάγει ο χρήστης είναι σωστός και για να επιβεβαιωθεί ότι οι κωδικοί «new password» και «confirmation password» είναι ίδιοι.



Εικόνα 19: σελίδα /profile. Αλλαγή κωδικού πρόσβασης

3.6 Σελίδα διαχειριστή

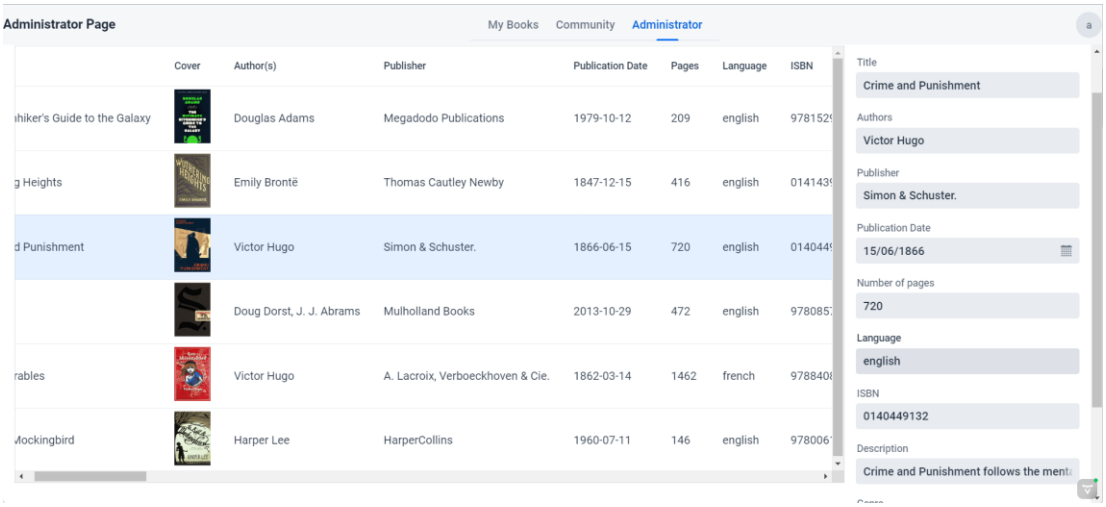
Όταν συνδέεται ένας διαχειριστής (administrator) στην διαδικτυακή εφαρμογή έχει πρόσβαση σε όλες τις σελίδες τις διαδικτυακής εφαρμογής και στην σελίδα του διαχειριστή [Εικόνα] από την οποία μπορεί να διαχειριστή όλα τα διαθέσιμα βιβλία. Από αυτή την σελίδα μπορεί ακόμα να αναζητήσει βιβλία με βάση τον τίτλο τους.



Εικόνα 20: σελίδα /admin

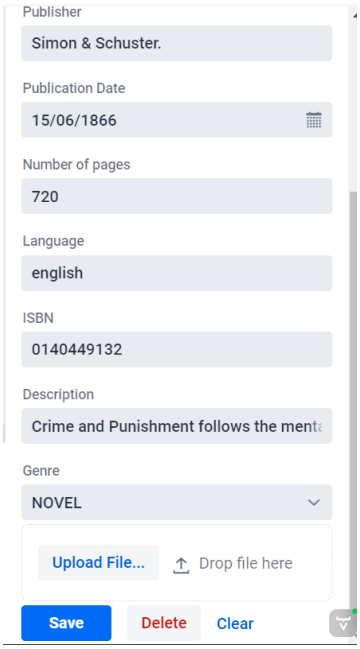
Πατώντας ο διαχειριστής το κουμπί «Add Book» εμφανίζεται μια φόρμα στην δεξιά πλευρά της σελίδας. Σε αυτή τη φόρμα μπορεί ο διαχειριστής να συμπληρώσει τα στοιχεία του βιβλίου, να ανεβάσει την εικόνα του εξωφύλλου του βιβλίου, από το πεδίο «Upload» και πατώντας το «Save» να αποθηκεύσει το βιβλίο. Από εκείνη την στιγμή το βιβλίο αυτό θα είναι διαθέσιμο για να το προσθέσουν

οι χρήστες σε κάποια από την λίστα τους. Η ίδια φόρμα εμφανίζεται και στην περίπτωση που ο διαχειριστής επιλέξει ένα βιβλίο. Σε αυτή την περίπτωση τα πεδία της φόρμας είναι προ συμπληρωμένα με τα στοιχεία του βιβλίου που επέλεξε ο διαχειριστής [Εικόνα 20] και [Εικόνα 21]. Αλλάζοντας ο διαχειριστής κάποιο δεδομένο του βιβλίου και πατώντας το κουμπί «Save» μπορεί να ενημερώσει τα στοιχεία του βιβλίου που επέλεξε.



Εικόνα 21: σελίδα /admin. Επεξεργασία βιβλίου

Πατώντας το κουμπί Delete ο διαχειριστής μπορεί να διαγράψει ένα βιβλίο εφόσον κανένας χρήστης δεν το έχει προσθέσει σε κάποια προσωπική του λίστα.



Εικόνα 22: σελίδα /admin Επεξεργασία βιβλίο

ΚΕΦΑΛΑΙΟ 4 ΣΥΜΠΕΡΑΣΜΑΤΑ ΚΑΙ ΠΡΟΤΑΣΕΙΣ

Στην παρούσα πτυχιακή σχεδιάστηκε και υλοποιήθηκε μια διαδικτυακή εφαρμογή η οποία δίνει την δυνατότητα στους χρήστες όχι μόνο να οργανώνουν την προσωπική τους βιβλιοθήκη αλλά και να αλληλοεπιδρούν μεταξύ τους ανταλλάσσοντας απόψεις για τα βιβλία τα οποία διάβασαν. Η ενσωμάτωση της Spring Boot και του Vaadin όχι μόνο μόνο διευκόλυνε τη δημιουργία μιας ευέλικτης και δυναμικής εφαρμογής, αλλά και κατέδειξε την ικανότητα αυτών των τεχνολογιών σε συνδυασμό. Παρόλα αυτά είναι σημαντικό να τονιστούν και οι ιδέες για την εξέλιξη και την περαιτέρω βελτίωση τη διαδικτυακής εφαρμογής. Αρχικά θα ενισχυθεί η ασφάλεια του συστήματος κατά την αυθεντικοποίηση των χρηστών όπως αναφέρθηκε και αναλύθηκε παραπάνω στην πτυχιακή εργασία. Έπειτα θα προστεθεί η δυνατότητα στους χρήστες να λαμβάνουν ειδοποιήσεις όταν δέχονται κάποιο αίτημα φιλίας ή όταν κάποιος χρήστης κάνει μια καινούρια κριτική βιβλίου. Τις ειδοποιήσεις αυτές θα μπορούν να της βλέπουν συγκεντρωτικά σε ένα σημείο το οποίο θα συντελέσει στην καλύτερη οργάνωση και ενημέρωση του χρήστη. Επιπρόσθετα θα δοθεί η δυνατότητα ανατροφοδότησης μοιράζοντας οι χρήστες πιθανούς προβληματισμούς σχετικά με την εφαρμογή με τον διαχειριστή. Αυτό θα συντελέσει στην όσο τον δυνατόν καλύτερη εμπειρία του χρήστη.

Βιβλιογραφία

[1] baeldung (2017). *Introduction to Vaadin | Baeldung*. [online] www.baeldung.com. Available at: https://www.baeldung.com/vaadin.

[2] Crusoveanu, L. (2016). *Inversion of Control and Dependency Injection with Spring*. [online] Baeldung. Available at: https://www.baeldung.com/inversion-control-and-dependency-injection-in-spring.

[3] Duarte, A. (2021). The World of Vaadin. *Apress eBooks*. doi:https://doi.org/10.1007/978-1-4842-7179-7_1.

[4] Gorbachenko, P. (2021). *Functional vs Non-Functional Requirements [Updated 2021]*. [online] enkonix.com. Available at: https://enkonix.com/blog/functional-requirements-vs-non-functional/.

[5] Luan, X. (2021). *IMPLEMENTATION AND ANALYSIS OF SOFTWARE DEVELOPMENT IN SPRING BOOT*. [online] Available at: https://scholarworks.calstate.edu/downloads/zg64ts132.

[6] Paraschiv, E. (2015). *Password Encoding with Spring | Baeldung*. [online] www.baeldung.com. Available at: https://www.baeldung.com/spring-security-registration-password-encoding-bcrypt.

[7] Ray, D. (n.d.). Spring Boot Features. *www.academia.edu*. [online] Available at: https://www.academia.edu/31605796/Spring_Boot_Features.

[8] Sawant, P. (2012). Spring Framework: A Companion to JavaEE. *www.academia.edu*. [online] Available at: https://www.academia.edu/83325479/Spring_Framework_A_Companion_to_JavaEE.

[9] Tutorialspoint.com. (2019). *DBMS - Data Schemas - Tutorialspoint*. [online] Available at: https://www.tutorialspoint.com/dbms/dbms_data_schemas.htm.

[10] Vaadin Team (2019). *Book of Vaadin*. Independently published.

[11] www.ibm.com. (n.d.). *What is a database schema? | IBM*. [online] Available at: https://www.ibm.com/topics/database-schema#:~:text=A%20database%20schema%20defines%20how.