Βάσεις Δεδομένων-Εξαμηνιαία Εργασία
Ακ. έτος 2022-2023

Project 33		
Όνομα	Γεώργιος	Νίκη
Επίθετο	Μπελίωτης	Σκορδάκη
Σχολή	ΗΜΜΥ	ΕΜΦΕ
Αριθμός Μητρώου	el19139	ge19034
Email	georgebel@hotmail.gr	ge19034@ntua.gr
Εξάμηνο	8ο	8ο

Περιεχόμενα

1 Προαπαιτούμενα της εφαρμογής

Για να τρέξετε την εφαρμογή, θα χρειαστείτε να έχετε εγκατεστημένα στο σύστημα σας:

- Μία βάση δεδομένων Mariadb,
- Έναν dbms client, εμείς επιλέξαμε το

Για το server- side της εφαρμογής, επιλέξαμε να χρησιμοποιήσουμε

1.1 Σύνδεσμος για git repo

Σύνδεσμο για το git repo της εφαρμογής μας:
https://github.com/nikiskordaki/project33_2023_el19139_ge19034

1.2Εγκατάσταση των πακέτων της python

Η εφαρμογή μας και τα πακέτα που χρειάζεται, τρέχουν στην python3. Τα πακέτα της Python που θα χρειαστείτε μπορείτε να τα αποκτήσετε μέσω pip:

> pip install flask flask_mysqldb

Αν δεν έχετε το pip διαθέσιμο στον υπολογιστή σας συμβουλευτείτε τις οδηγίες της διανομής

σας για να αποκτήσετε την έκδοση που είναι συμβατή με την python3.

2 Εγκατάσταση της εφαρμογής

Ανοίξτε το phpMyAdmin και κατεβαστέ από το github repo (https://github.com/nikiskordaki/project33_2023_el19139_ge19034) τα αρχεία DDL και DML. Σιγουρευτείτε ότι βρίσκεστε στην αρχική σελίδα του phpMyAdmin («home») και πως δεν υπάρχει άλλη βάση με το όνομα «project33».

2.1 Εισαγωγή του DDL αρχείου

Το αρχείο ddl θα δημιουργήσει μια νέα βάση με το όνομα «project33», και θα ορίσει όλους τους πίνακες (entities & relations) της βάσης, καθώς και τα views. Ακολουθώντας τα παρακάτω βήματα

μπορείτε να εισάγεται το εν λόγω αρχείο στην βάση:

1. Από την αρχική σελίδα του phpmyadmin επιλέξτε το κουμπί Import στο πάνω menu.

2. Στην σελίδα που μας ανοίγει, πρέπει να επιλέξετε για import κάποιο από τα διαθέσιμα στον υ-πολογιστή σας. Πατήστε «Choose Files» και επιλέξτε το ddl αρχείο μας, «DDL_project33_2023».

Οι υπόλοιπες επιλογές μπορούν να μείνουν ως έχουν.

3. Τέλος πατήστε το κουμπί «Go» στο κάτω δεξιά μέρος της σελίδας. Αν όλα πάνε καλά ευχάριστα μηνύματα επιτυχίας θα γεμίσουν την οθόνη σας και θα μπορείτε να δείτε την βάση μας στο αριστερό menu του phpmyadmin.

2.2 Εισαγωγή του DML αρχείου

Στην συνέχεια, θα πρέπει να εισάγετε δεδομένα στην βάση, μέσω του dml script. Επαναλάβετε τα ίδια βήματα που ακολουθήσατε στην προηγούμενη ενότητα για το ddl script, όμως αυτή την φορά επιλέξτε το αρχείο «DML_project33_2023».

2.3 Τρέχοντας τον web server

Στο github repo της ομάδας μας, θα βρείτε τον φάκελο «web server» μέσα στον οποίο υπάρχει το αρχείο app.py και μπορείτε να το τρέξετε ανοίγοντας ένα τερματικό στον εν λόγω φάκελο και πληκτρολογώντας την εντολή:

> flask run

Απαραίτητη προϋπόθεση για να συνδεθεί σωστά το flask με την βάση, είναι στις πρώτες γραμμές

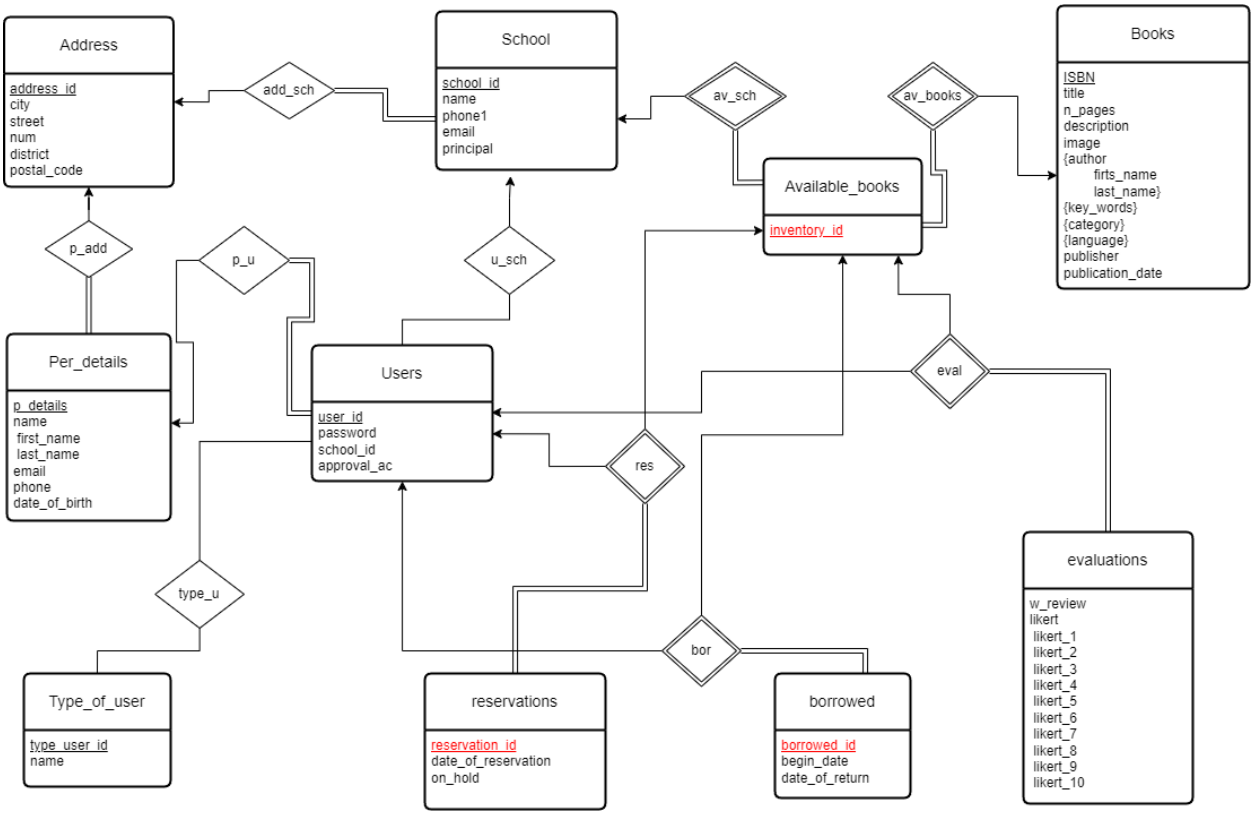
του αρχείου app.py να ορίσετε σωστά την διεύθυνση και την θύρα στην οποία τρέχει η βάση, το username, τον κωδικό και το όνομα της βάσης. Ενδεικτικά, με βάση όσα αναφέρθηκαν στις προηγούμενες ενότητες το σωστό configuration θα ήταν:

Σε ένα σύστημα που τρέχει windows και η Mariadb είναι εγκατεστημένη μέσω XAMPP, τότε το επιθυμητό configuration είναι:

3 Επισκόπηση της βάσης

3.1 Το διάγραμμα ER

Το ER diagram προκειμένου η βάση μας να είναι λειτουργική και να ικανοποιεί όλα τα ζητήματα, θα πρέπει σε conceptual επίπεδο να φτιαχτεί ένα διάγραμμα που να περιγράφει τις βασικές οντότητες της βάσης μας, τα attributes των οντοτήτων, και τις σχέσεις που περιγράφουν τις μεταξύ τους αλληλεπιδράσεις. Το διάγραμμα αυτό καλείται Entity Relationship diagram και για την βάση μας μπορείτε να το δείτε παρακάτω.



Τα attributes που βλέπεται κόκκινα και υπογραμμισμένα είναι discriminator για τα weak entities αυτά.

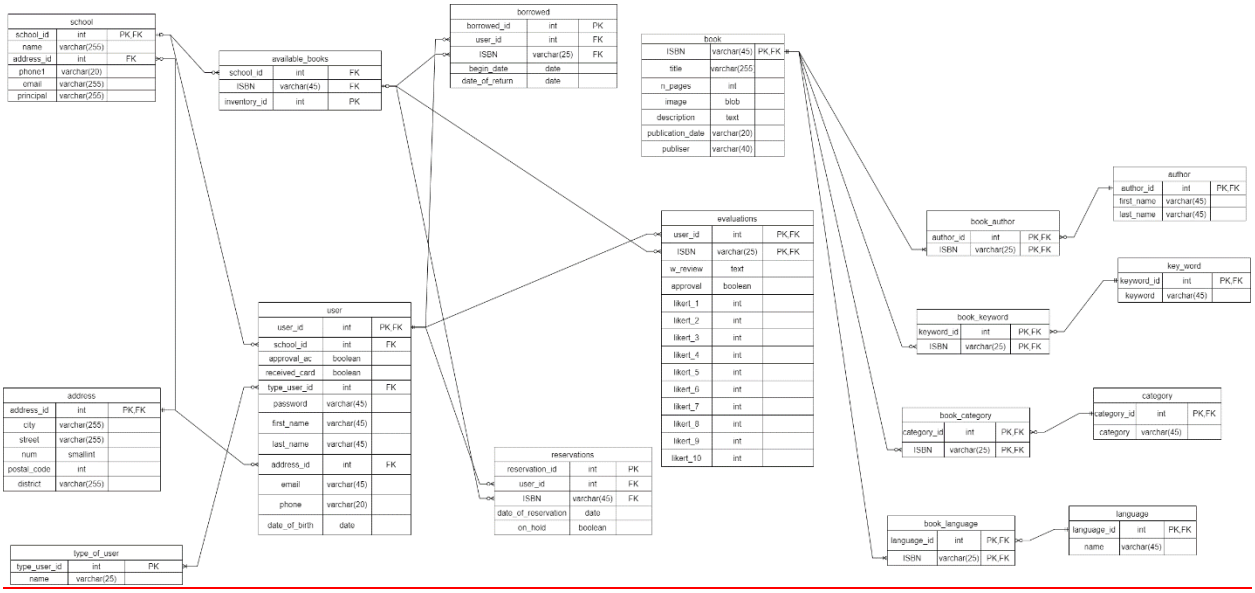
Τα attribute αναλύονται στο σχεσιακό σχήμα, τα περισσότερα έχουν τις τιμές που υποδεικνύει το όνομα τους στην φυσική γλώσσα. Θα θέλαμε επίσης να εξηγήσουμε τις σχέσεις που παρουσιάζονται παραπάνω.

- P_add & add_sch, μέσω αυτής της σχέσης ουσιαστικά αντιστοιχίζουμε τις διευθύνσεις στον αντίστοιχο χρήστη ή στο αντίστοιχο σχολείο.
- Type_u, μέσω αυτής της σχέσης αντιστοιχίζουμε κάθε χρήστη στον αντίστοιχο τύπο χρήστη. (Στο ER έχουμε θεωρήσει ότι ο υπεύθυνος χειριστής θα εισάγεται στην βάση και με αυτό του τον ρόλο αλλά και σαν εκπαιδευτικός)

- P_u, μέσω αυτής της σχέσης αντιστοιχίζουμε κάθε χρήστη με τα προσωπικά του στοιχεία.
- U-sch, αντιστοιχίζει κάθε χρήστη με το σχολείο στο οποίο ανήκει, με εξαίρεση τον κεντρικό διαχειριστή.
- Av_sch, αντιστοιχίζει την κάθε εισαγωγή βιβλίου ή επιστροφή με την αντίστοιχη σχολική μονάδα.
- Av_books, αντιστοιχίζει την κάθε εισαγωγή βιβλίου ή επιστροφή με τα στοιχεία του αντίστοιχου βιβλίου.
- Res, αντιστοιχίζει την κράτηση ενός βιβλίου με τον χρήστη, που την έκανε και το βιβλίο που περιέχει.
- Bor, αντιστοιχίζει τον δανεισμού ενός βιβλίου με τον χρήστη, που τον έκανε και το βιβλίο που περιέχει.
- Eval, αντιστοιχίζει την αξιολόγηση ενός βιβλίου με τον χρήστη, που την έκανε και το βιβλίο που αξιολόγησε.

3.2 Σχεσιακό Διάγραμμα

Στο σχεσιακό διάγραμμα μπορείτε να δείτε πως υλοποιήσαμε το ER στην βάση μας. Το εν λόγω διάγραμμα περιέχει τους πίνακες που ορίσαμε με τις αντίστοιχες στήλες τους, καθώς και τις σχέσεις μεταξύ των πινάκων.



Συνοπτική παρουσίαση των πινάκων που περιλαμβάνει το παραπάνω relation schema:

- School, στον πίνακα αυτόν αποθηκεύουμε τα στοιχεία που ζητούνται από την εκφώνηση για ένα σχολείο. Το attribute address_id ουσιαστικά συνδέοντας τον πίνακα school με τον πίνακα address, περιέχει έμμεσα την διεύθυνση του σχολείου και το attribute principal περιέχει το ονοματεπώνυμο του διευθυντή της σχολικής μονάδας.
- Address, στον πίνακα αυτόν αποθηκεύουμε τα απαραίτητα στοιχεία των διευθύνσεων των χρηστών και των σχολικών μονάδων.
- Type_of_user, στον πίνακα αυτόν αποθηκεύουμε τους τύπους των χρηστών που έχει η βάση. Στο attribute type_user_id με αριθμητικό τρόπο στο attribute name αποθηκεύουμε τον τίτλο('Administrator','Operator','Professor','Student'). Επιλέξαμε την δημιουργία αυτού του πίνακα διότι οι διαχειριστές ('Operator') είναι και εκπαιδευτικοί ('Professor')
- User, στον πίνακα αυτόν αποθηκεύουμε τα στοιχεία των χρηστών, το αν έχουν λάβει έγκριση για τον λογαριασμό (στο 'approval_ac') και το αν έχουν λάβει την κάρτα της βιβλιοθήκης (received_card).
- Book, στον πίνακα αυτόν αποθηκεύουμε τα στοιχεία που μας έχουν ζητηθεί για κάθε βιβλίο.
- Available_books, στον πίνακα αυτόν αποθηκεύουμε τα διαθέσιμα βιβλία που έχει κάποια σχολική μονάδα. Το attribute inventory_id ουσιαστικά δημιουργεί μια νέα πλειάδα για κάθε προσθήκη ενός βιβλίου στο σύστημα και για κάθε επιστροφή βιβλίου.
- Borrowed, στον πίνακα αυτόν αποθηκεύουμε κάθε δανεισμό που υλοποιείται. Στο begin_date αποθηκεύεται η ημερομηνία δανεισμού σε format yyyy-mm-dd και στο

date_of_return αποθηκεύεται η ημερομηνία επιστροφής σε format yyyy-mm-dd, το οποίο παραμένει NULL μέχρι να το επιστρέψει ο δανειζόμενος.

- Reservations, στον πίνακα αυτόν αποθηκεύουμε κάθε κράτηση βιβλίου, που πραγματοποιείται. Στο date_of_reservation αποθηκεύεται η ημερομηνία πραγματοποίησης του αιτήματος σε format yyyy-mm-dd και στο attribute on_hold, το οποίο είναι Boolean, αποθηκεύεται το αν έχει δώσει έγκριση ο διαχειριστής στο αίτημα και είναι σε αναμονή (on_hold=1) ή όχι (on_hold=0), μέχρι ο διαχειριστής να δώσει την έγκριση η τιμή του on_hold είναι NULL.
- Evaluations, στον πίνακα αυτόν αποθηκεύουμε της αξιολογήσεις που κάνει ένας χρήστης σε ένα βιβλίο. Στο w_review αποθηκεύουμε την γραπτή αξιολόγηση που μπορεί να υποβάλει κάποιος χρήστης, ενώ στα likert_ # αποθηκεύουμε την επιλογή του χρήστης στα παρακάτω ερωτήματα, οι likert_# παίρνουμε τιμές από το 1 στο 5.

  <u>Likert scale questions you can use to rate a book</u>

  1.Overall, how would you rate the book?
  - Strongly Dislike
  - Dislike
  - Neutral
  - Like
  - Strongly Like

  2.How engaging was the storyline?
  - Not engaging at all
  - Slightly engaging
  - Moderately engaging
  - Very engaging
  - Extremely engaging

  3.How well-developed were the characters?
  - Poorly developed
  - Somewhat developed
  - Moderately developed
  - Well-developed
  - Exceptionally well-developed

  4.How would you rate the pacing of the book?
  - Too slow
  - Somewhat slow
  - Just right
  - Somewhat fast
  - Too fast

  5.Did the book keep you interested throughout?
  - Not at all
  - Slightly
  - Moderately
  - Quite a bit
  - Definitely

  6.How well did the author convey the book's themes and messages?

Very poorly
Somewhat poorly
Moderately well
Very well
Exceptionally well

7.How satisfied were you with the ending of the book?
Very dissatisfied
Somewhat dissatisfied
Neutral
Somewhat satisfied
Very satisfied

8.How likely are you to recommend this book to others?
Very unlikely
Somewhat unlikely
Neutral
Somewhat likely
Very likely

9.How well-written was the book in terms of style and language?
Very poorly written
Somewhat poorly written
Moderately well-written
Very well-written
Exceptionally well-written

10.How emotionally impactful was the book?
Not impactful at all
Slightly impactful
Moderately impactful
Very impactful
Extremely impactful

- Author, στον πίνακα αυτόν αποθηκεύουμε τα στοιχείων των συγγραφέων.
- Key_word, στον πίνακα αυτόν αποθηκεύουμε τις λέξεις κλειδιά για τα βιβλία.
- Category, στον πίνακα αυτόν αποθηκεύουμε τις κατηγορίες βιβλίων.
- Language, στον πίνακα αυτόν αποθηκεύουμε τις πιθανές γλώσσες, στις οποίες μπορεί να είναι γραμμένο κάποιο βιβλίο.

Λόγω του ότι η Mariadb δεν υποστηρίζει multivalued attributed χρειάστηκε να δημιουργήσουμε επιπλέον πίνακες από αυτούς που παρουσιάσαμε στο ER διάγραμμα, αυτοί είναι οι πίνακες book_author, book_category, book_language και book_keyword και μας δίνουν την δυνατότητα σε ένα βιβλίο να αποθηκεύσουμε περισσότερους από έναν συγγραφείς, κατηγορίες, λέξεις κλειδιά και γλώσσες. Επιπλέον για τις ανάγκες υλοποίησης της βάσης συγχωνεύσαμε του πίνακες user-per_details.

3.3 Παραδοχές

Για την υλοποίηση της παραπάνω βάσης κάναμε τις εξής παραδοχές:

- Τα αιτήματα θα διαγράφονται στο τέλος της 7ης ημέρας, ώστε να μην μπορούν να κρατιόνται τα βιβλία παραπάνω από μια εβδομάδα.
- Οι επιστροφές θα είναι εκπρόθεσμες μετά το τέλος της 7ης ημέρας.
- Ένας χρήστης μπορεί να δανειστεί ή να κάνει κράτηση σε κάποιο βιβλίο παραπάνω από μια φορές.
- Ένας χρήστης μπορεί να πραγματοποιήσει όσο αιτήματα κράτησης επιθυμεί αρκεί να μην έχει καθυστερήσει κάποια επιστροφή, να μην έχει ξεπεράσει το όριο δανεισμού και να μην έχει ξεπεράσει το όριο των αποδεκτών κρατήσεων.
- Οι εκπαιδευτικοί που είναι διαχειριστές κάποιας σχολικής μονάδας εισάγονται στην βάση ως διαχειριστές (‘Operators’), κρατώντας φυσικά το δικαίωμα δανεισμού, κράτησης και αξιολόγησης κάποιου βιβλίου.
- Δεν μπορούμε να διαγράψουμε κάποιο βιβλίο.

3.4 Ευρετήρια

Από μόνη της η Mariadb όταν δημιουργούμε ένα νέο πίνακα, φτιάχνει indexes πάνω στο primary key και επίσης υπάρχουν ήδη ευρετήρια στα τυχόν foreign key τα οποία κάνουν reference το primary key κάποιου άλλου πίνακα. Μένει λοιπόν να φτιάξουμε τυχόν indexes για τα ερωτήματα του τρίτου σκέλους της εκφώνησης.

```
--INDEXES
--gia erwthma 3.1.1
create index date_bor on borrowed(begin_date);

--gia erwthma 3.1.2
create index cat on category(category);
create index first_name_u on user(first_name);
create index lastname_u on user(last_name);

--gia 3.1.3
create index cat_user on type_of_user(name);
create index birthD on user(date_of_birth);
--gia 3.1.4
create index author_first on author(first_name);
create index author_last on author(last_name);

--gia 3.2.1
create index title on book(title);
create index invent on available_books(inventory_id);
--gia 3.2.2
create index bor_user on borrowed(user_id);
create index ret_date on borrowed(date_of_return);
```

3.5 Όψεις και Procedures του σχεσιακού για το 3ο μέρος της εκφώνησης

```
--ERWTHMATA 3ou Merous ekfwnhsh
--gia erwthma 3.1.1
DELIMITER $$
CREATE PROCEDURE GetLoanCountBySchool(search_year INT, search_month INT)
BEGIN
  SELECT s.school_id, s.name, COUNT(*) AS total_loans
  FROM borrowed b
  JOIN user u ON b.user_id = u.user_id
  JOIN school s ON u.school_id = s.school_id
  WHERE YEAR(b.begin_date) = search_year
  AND MONTH(b.begin_date) = search_month
  GROUP BY s.school_id, s.name;
END$$
DELIMITER ;

--gia erwthma 3.1.2
create procedure book_cat(s varchar(45))
select a.first_name ,a.last_name from author a
inner join book_author b on b.author_id=a.author_id
inner join book_category bc on bc.ISBN=b.ISBN
inner join category c on bc.category_id=c.category_id
where c.category=s;

create procedure cat_prof(s varchar(45))
select u.first_name, u.last_name from user u
inner join type_of_user tu on tu.type_user_id=u.type_user_id
inner join borrowed bb on bb.user_id=u.user_id
inner join available_books ab on ab.ISBN=bb.ISBN
inner join book b on b.ISBN=ab.ISBN
inner join book_category bc on b.ISBN=bc.ISBN
inner join category c on bc.category_id=c.category_id
where c.category=s and tu.name='Professor';

call book_cat(s) ;
call cat_prof(s);

--gia erwthma 3.1.3
create view y_prof as
select u.first_name,u.last_name,count(bb.borrowed_id) from user u
inner join borrowed bb on bb.user_id=u.user_id
where DATEDIFF(CURDATE(), u.date_of_birth)<14600 AND (type_user_id=2 OR type_user_id=3)
group by u.user_id
order by count(bb.borrowed_id) desc;
```

```sql
--gia erwthma 3.1.4
create view no_bor_author as
select a.first_name,a.last_name from author a
where author_id not in (select a.author_id from author a
inner join book_author ba on ba.author_id=a.author_id
inner join book b on b.ISBN=ba.ISBN
inner join borrowed bb on bb.ISBN=b.ISBN
group by a.author_id
having count(borrowed_id)>0);

--gia erwthma 3.1.5
create view oper_bor as
 SELECT u.first_name, u.last_name, subquery.num_loans1
FROM (
  SELECT COUNT(bb.borrowed_id) AS num_loans1, u.school_id
   FROM borrowed bb
   INNER JOIN user u ON u.user_id = bb.user_id
   GROUP BY u.school_id
  HAVING COUNT(bb.borrowed_id) > 20
 ) AS subquery
INNER JOIN user u ON u.school_id = subquery.school_id
 where type_user_id=2;


--gia erwthma 3.1.6
create view top_category_pairs as
select c1.category as category_1, c2.category as category_2, count(*) as pair_count from category c1
inner join book_category bc1 on bc1.category_id=c1.category_id
inner join borrowed bb on bb.ISBN=bc1.ISBN
inner join book_category bc2 on bc2.ISBN=bc1.ISBN
inner join category c2 on bc2.category_id=c2.category_id
where c2.category_id<>c1.category_id and  c2.category_id<c1.category_id
group by category_1,category_2
order by count(*) desc
limit 3;


--gia erwthma 3.1.7
CREATE VIEW author_count AS
SELECT a.first_name, a.last_name, COUNT(ba.ISBN) AS book_count
FROM author a
INNER JOIN book_author ba ON ba.author_id = a.author_id
GROUP BY ba.author_id
HAVING
    COUNT(ba.ISBN) <= (
```

```
    SELECT MAX(cnt) - 5
    FROM (
      SELECT COUNT(ba2.ISBN) AS cnt
      FROM book_author ba2
      GROUP BY ba2.author_id
    ) subquery
);


--gia erwthma 3.2.1
DELIMITER //
CREATE PROCEDURE search_books(IN search_criteria VARCHAR(255), IN school_u INT)
BEGIN
  SELECT
    b.title,
    CONCAT(a.first_name, ' ', a.last_name) AS author,
    COUNT(*) AS copies
  FROM
    book b
    INNER JOIN available_books av ON av.ISBN = b.ISBN
    INNER JOIN book_author ba ON av.ISBN = ba.ISBN
    INNER JOIN author a ON ba.author_id = a.author_id
  WHERE
    av.school_id = school_u
    AND (
      b.title=search_criteria
      OR CONCAT(a.first_name, ' ', a.last_name) =search_criteria
      OR (
        SELECT COUNT(inventory_id)
        FROM available_books
        WHERE school_id = school_u AND ISBN = av.ISBN
        GROUP BY ISBN
      HAVING COUNT(inventory_id)=Cast(search_criteria as SIGNED INTEGER)
    ))
  GROUP BY
    b.title, a.author_id;
END //
DELIMITER ;

--idia me prin apla edw psaxnoume me kritirio thn kathgoria
DELIMITER //
CREATE PROCEDURE search_books_cat(IN search_criteria VARCHAR(255), IN school_u INT)
BEGIN
  SELECT
    b.title,
```

```
    CONCAT(a.first_name, ' ', a.last_name) AS author,
    COUNT(*) AS copies
  FROM
    book b
    INNER JOIN available_books av ON av.ISBN = b.ISBN
    INNER JOIN book_author ba ON av.ISBN = ba.ISBN
    INNER JOIN author a ON ba.author_id = a.author_id
    INNER JOIN book_category bc ON bc.ISBN = av.ISBN
    INNER JOIN category c ON bc.category_id = c.category_id
  WHERE
    av.school_id = school_u
    AND c.category=search_criteria
  GROUP BY
    b.title, a.author_id;
END //
DELIMITER ;


--gia erwthma 3.2.2
DELIMITER //
CREATE PROCEDURE search_bor_user(IN search_criteria VARCHAR(255))
BEGIN
  SELECT
    bb.user_id,
    u.first_name,
    u.last_name
  FROM
    borrowed bb
    INNER JOIN user u ON bb.user_id = u.user_id
  WHERE
    (
      SELECT COUNT(bbb.borrowed_id)
      FROM borrowed bbb
      WHERE bbb.user_id = bb.user_id
      GROUP BY bbb.user_id
      HAVING COUNT(bbb.borrowed_id) >= 1
    ) AND date_of_return is NULL
    AND DATEDIFF(CURDATE(), bb.begin_date) > 7
    AND (
      p.first_name = search_criteria
      OR p.last_name = search_criteria
      OR DATEDIFF(CURDATE(), bb.begin_date)-7 = CAST(search_criteria AS SIGNED INTEGER)
    );
end //
DELIMITER ;
```

```
create view bor_exp_all as
   SELECT
      bb.user_id,
      u.first_name,
      u.last_name
   FROM
      borrowed bb
      INNER JOIN user u ON bb.user_id = u.user_id
   WHERE
      (
         SELECT COUNT(bbb.borrowed_id)
         FROM borrowed bbb
         WHERE bbb.user_id = bb.user_id
         GROUP BY bbb.user_id
         HAVING COUNT(bbb.borrowed_id) >= 1
      )
      AND DATEDIFF(CURDATE(), bb.begin_date) > 7;



--gia erwthma 3.2.3

DELIMITER //

CREATE PROCEDURE calculate_average_ratings(IN search_criteria VARCHAR(255), IN sch int)
BEGIN
   SELECT u.user_id, c.category, AVG(e.likert_1) AS average_rating_q1, AVG(e.likert_2) AS
average_rating_q2, AVG(e.likert_3) AS average_rating_q3, AVG(e.likert_4) AS average_rating_q4,
AVG(e.likert_5) AS average_rating_q5, AVG(e.likert_6) AS average_rating_q6, AVG(e.likert_7) AS
average_rating_q7, AVG(e.likert_8) AS average_rating_q8, AVG(e.likert_9) AS average_rating_q9,
AVG(e.likert_10) AS average_rating_q10
   FROM user u
   INNER JOIN borrowed b ON u.user_id = b.user_id
   INNER JOIN book_category bc ON b.ISBN = bc.ISBN
   INNER JOIN category c ON bc.category_id = c.category_id
   INNER JOIN evaluations e ON b.ISBN = e.ISBN
   WHERE u.school_id=sch AND (u.user_id = search_criteria
     OR c.category = search_criteria)
   GROUP BY u.user_id, c.category;
END //

DELIMITER ;
```

```
--gia erwthma 3.3.1

create procedure available_books_u(in search_criteria varchar(255),sch int)
select av.ISBN, b.title,c.category,b.description,concat(a.first_name, ' ',a.last_name) as author from
available_books av
inner join book b on b.ISBN=av.ISBN
inner join book_category bc on av.ISBN=bc.ISBN
inner join category c on c.category_id=bc.category_id
inner join book_author ba on ba.ISBN=av.ISBN
inner join author a on a.author_id=ba.author_id
where av.school_id=sch AND ( b.title= search_criteria
or c.category = search_criteria
or concat(a.first_name,' ', a.last_name)=search_criteria )
group by av.ISBN;

--gia erwthma 3.3.2
create procedure user_bor_list(s int)
select b.ISBN,b.title from book b
inner join borrowed bb on bb.ISBN=b.ISBN
where bb.user_id=s;
```

3.5 Triggers

Δημιουργήσαμε κάποια trigger ώστε να εξασφαλίσουμε ότι η βάση θα λειτουργεί σύμφωνα με τους περιορισμούς, που μας υποδεικνύει η εκφώνηση.

```
--TRIGGERS

--trigger gia na exasfalisoume oti kapoios xrhsths den tha xeperasei to orio daneismou
DELIMITER $$
CREATE TRIGGER `check_borrow_limit_trigger`
BEFORE INSERT ON `borrowed`
FOR EACH ROW
BEGIN
  IF (
    (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 4 AND
    (
      SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND `date_of_return`
IS NULL
    ) =2
  ) OR (
    (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 3  AND
   ( SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND        `date_of_return`
IS NULL
    ) =1
  OR
  (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 2  AND
    (
      SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND `date_of_return`
IS NULL
    ) =1
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Borrowing limit exceeded for the user';
  END IF;
END $$
DELIMITER ;

--trigger gia na exasfalisoume oti den tha ginontai aithmata krathsews apo xrhstes ftasei to orio apodekton
krathsewn
DELIMITER $$
CREATE TRIGGER check_reservation_limit_trigger
BEFORE INSERT ON reservations
FOR EACH ROW
BEGIN
  IF (
    (SELECT type_user_id FROM user WHERE user_id = NEW.user_id) = 4 AND
    (
```

```
        SELECT COUNT(*) FROM reservations WHERE user_id = NEW.user_id AND on_hold=1
      ) >= 2)
    OR (
      (SELECT type_user_id FROM user WHERE user_id = NEW.user_id) = 2 AND
      ( (SELECT COUNT(*) FROM reservations WHERE user_id = NEW.user_id AND on_hold=1
      ) >= 1)
OR (SELECT `type_user_id` FROM user WHERE user_id = NEW.user_id) = 3 AND
        (
        SELECT COUNT(*) FROM reservations WHERE user_id = NEW.user_id AND on_hold=1) >= 1
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Reservation limit exceeded for the user';
  END IF;
END $$
DELIMITER ;

--trigger gia na exasfalisoume oti den tha ginontai aithmata krathsews apo xrhstes pou ekkremoun epistrofes
bibliwn
DELIMITER $$
CREATE TRIGGER `check_borrow_limit_trigger_res`
BEFORE INSERT ON reservations
FOR EACH ROW
BEGIN
  IF (
    (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 4 AND
    (
      SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND `date_of_return`
IS NULL
    ) =2
  ) OR (
    (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 3  AND
   ( SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND       `date_of_return`
IS NULL
    ) =1
  OR
  (SELECT `type_user_id` FROM `user` WHERE `user_id` = NEW.`user_id`) = 2  AND
    (
      SELECT COUNT(*) FROM `borrowed` WHERE `user_id` = NEW.`user_id` AND `date_of_return`
IS NULL
    ) =1
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Borrowing limit exceeded for the user';
  END IF;
END $$
DELIMITER ;
```

```
--trigger gia na exasfalisoume oti den tha ginei "daneismos" se kapoio biblio pou den exei diathesima antitypa
DELIMITER $$
CREATE TRIGGER check_borrow_book_limit_trigger
BEFORE INSERT ON borrowed
FOR EACH ROW
BEGIN
  IF (
    (SELECT COUNT(*) FROM available_books WHERE ISBN = NEW.ISBN) <=
    (SELECT COUNT(*) FROM borrowed WHERE user_id = NEW.user_id AND ISBN = NEW.ISBN
AND date_of_return IS NULL)
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'The book is not available for borrowing';
  END IF;
END$$
DELIMITER ;


--trigger gia na diathrhsw oti krathseis kai daneismoi ginontai apo xrhstes sta sxoleia ta opoia anhkoun
DELIMITER $$
CREATE TRIGGER check_user_school_bor
BEFORE INSERT ON borrowed
FOR EACH ROW
BEGIN
  IF ( (SELECT u.school_id FROM user u WHERE u.user_id = NEW.user_id) not in
    (SELECT ab.school_id FROM available_books ab WHERE ab.ISBN = NEW.ISBN)
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Wrong combination of user-school';
  END IF;
END$$
DELIMITER ;

DELIMITER $$
CREATE TRIGGER check_user_school_res
BEFORE INSERT ON reservations
FOR EACH ROW
BEGIN
  IF ( (SELECT u.school_id FROM user u WHERE u.user_id = NEW.user_id) not in
    (SELECT ab.school_id FROM available_books ab WHERE ab.ISBN = NEW.ISBN)
  ) THEN
    SIGNAL SQLSTATE '45000' SET MESSAGE_TEXT = 'Wrong combination of user-school';
  END IF;
END$$
DELIMITER ;
```

3.6 Επιπλέον Procedures

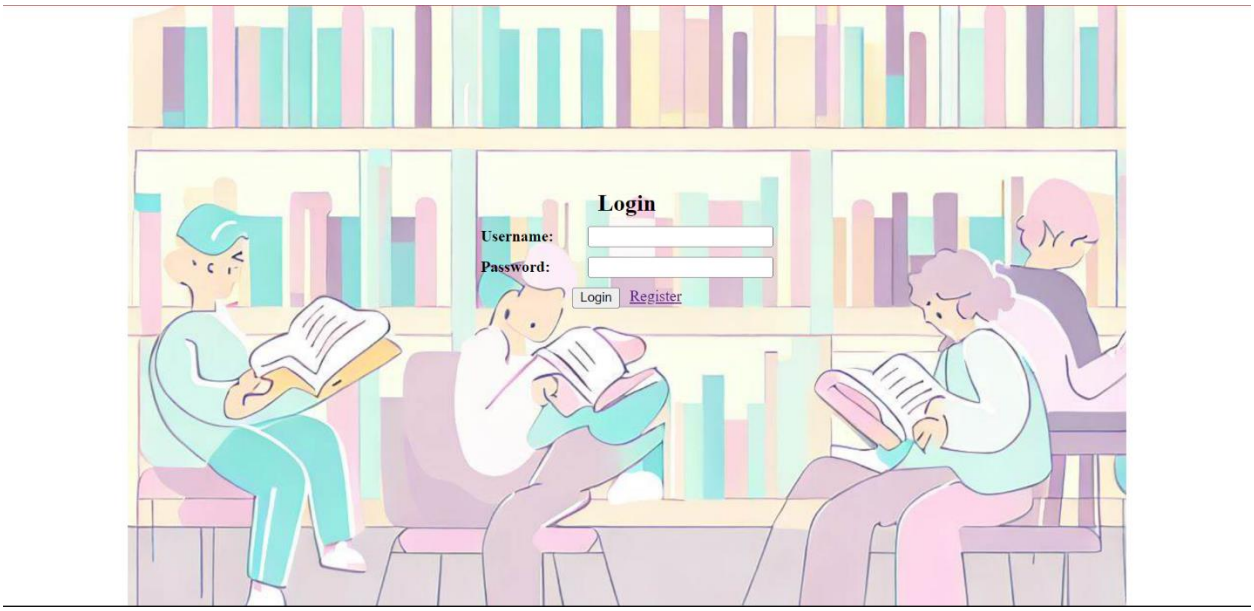
Δημιουργήσαμε κάποιες procedure ώστε να εξασφαλίσουμε ότι η βάση μας λειτουργεί σύμφωνα με τους περιορισμούς, υποδεικνύει η εκφώνηση.

```
--Synarthsh wste na lambanoun ameswsegkrish ta evaluations twn ekpaideutikwn
DELIMITER //
CREATE PROCEDURE prof_eval()
BEGIN
   UPDATE evaluations e
   SET approval = 1
   where e.user_id in (select e.user_id from evaluations e
   inner join user u on e.user_id=u.user_id
   where u.type_user_id=2 OR u.type_user_id=3);
END //
DELIMITER ;

--Synarthsh wste na diagrafontai oi krathseis pou exoun xeperasei thn 1 ebdomada
DELIMITER //
CREATE PROCEDURE oneweek()
BEGIN
   delete from reservations
   where DATEDIFF(CURDATE(), reservations.date_of_reservation)>7;
END //
DELIMITER ;

--Synarthsh wste na diagrafontai oi logariasmoi pou den exoun labei egkrish
DELIMITER //
CREATE PROCEDURE not_approved_u()
BEGIN
   delete from user
   where approval_ac=0;
END //
DELIMITER ;

--Synarthsh pou dinei ola ta antitypa enos bibliou pou yparxoun se kapoio apo ta sxoleia
create procedure inv_book_sch(s int,b int)
select count(ab.inventory_id) from available_books ab
where ab.ISBN=s and ab.school_id=b;
```

4 User manual

Εισαγωγή: Μετά την εγκατάσταση της εφαρμογής, όπως παρουσιάζεται στο git repo και στην αναφορά, εδώ παρουσιάζονται οι βασικές χρηστικότητες της εφαρμογής.

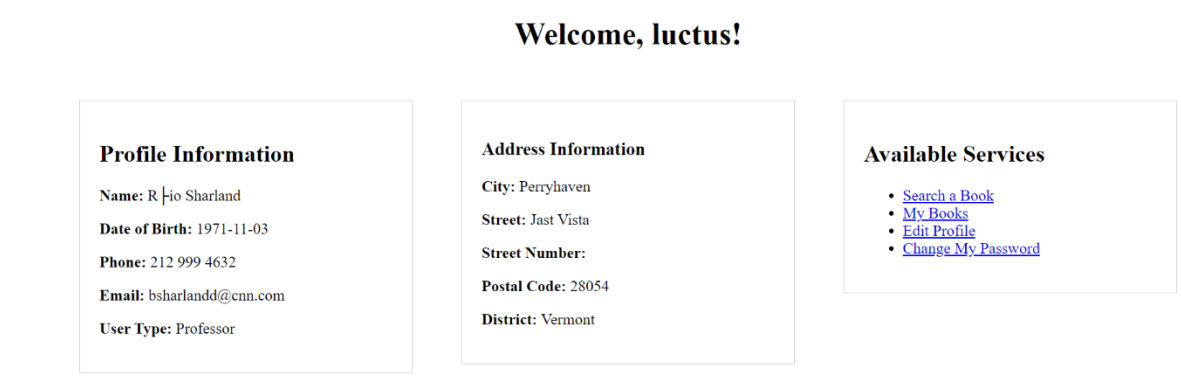
1. Μετά την εκκίνηση της εφαρμογή ο χρήστης ανακατευθύνεται στην αρχική σελίδα για να συνδεθεί με τα στοιχεία του:



2. Αν ο χρήστης δεν είναι εγγεγραμμένος στην εφαρμογή πατώντας τον σύνδεσμο Register ανακατευθύνεται στη σελίδα εγγραφής:

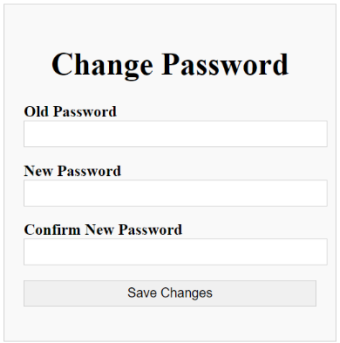
3. Μετά την συμπλήρωση των απαιτούμενων στοιχείων πατώντας το κουμπί Register ο χρήστης οδηγείτε πίσω στην αρχική σελίδα εφόσον η εγγραφή ήταν επιτυχής για να συνδεθεί.

4. Στην αρχική σελίδα Login αφού ο χρήστης συμπληρώσει το username και το password του και εφόσον αυτά είναι έγκυρα οδηγείτε στην βασική σελίδα της εφαρμογής:



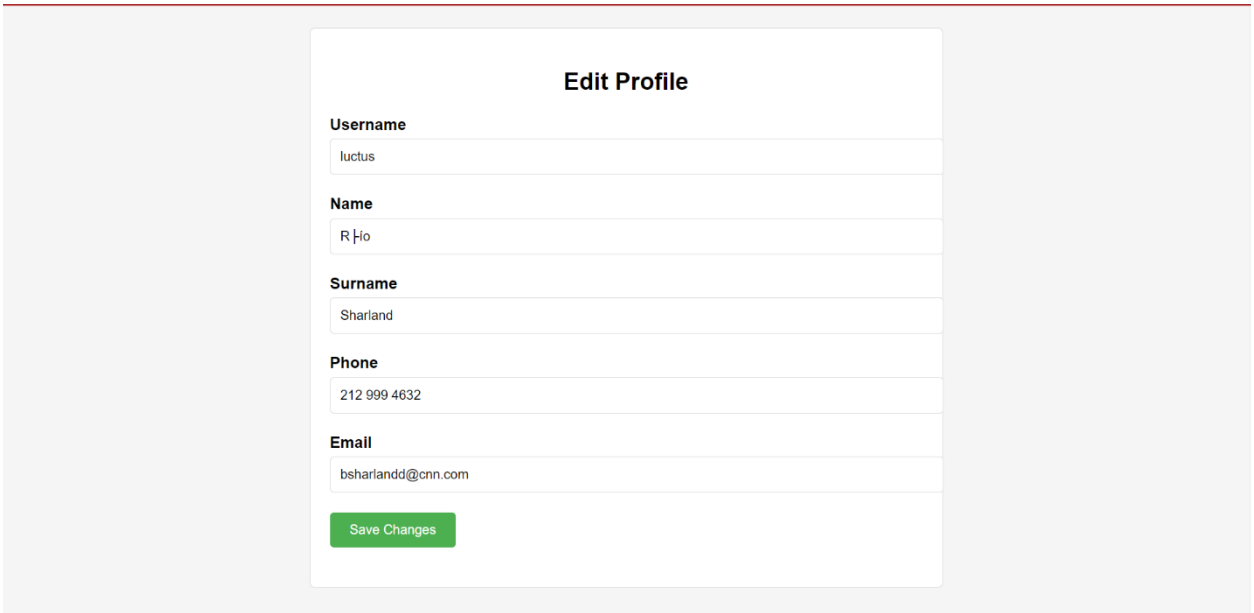
5. Στο βασικό παράθυρο της εφαρμογής παρουσιάζονται τα στοιχεία του χρήστη και του δίνονται οι διαθέσιμες επιλογές υπηρεσιών ανάλογα με τον τύπο του (μαθητής/καθηγητής/χειριστής/διαχειριστής) οι οποίες είναι:

α. Αλλαγή Κωδικού πατώντας το link Change Password



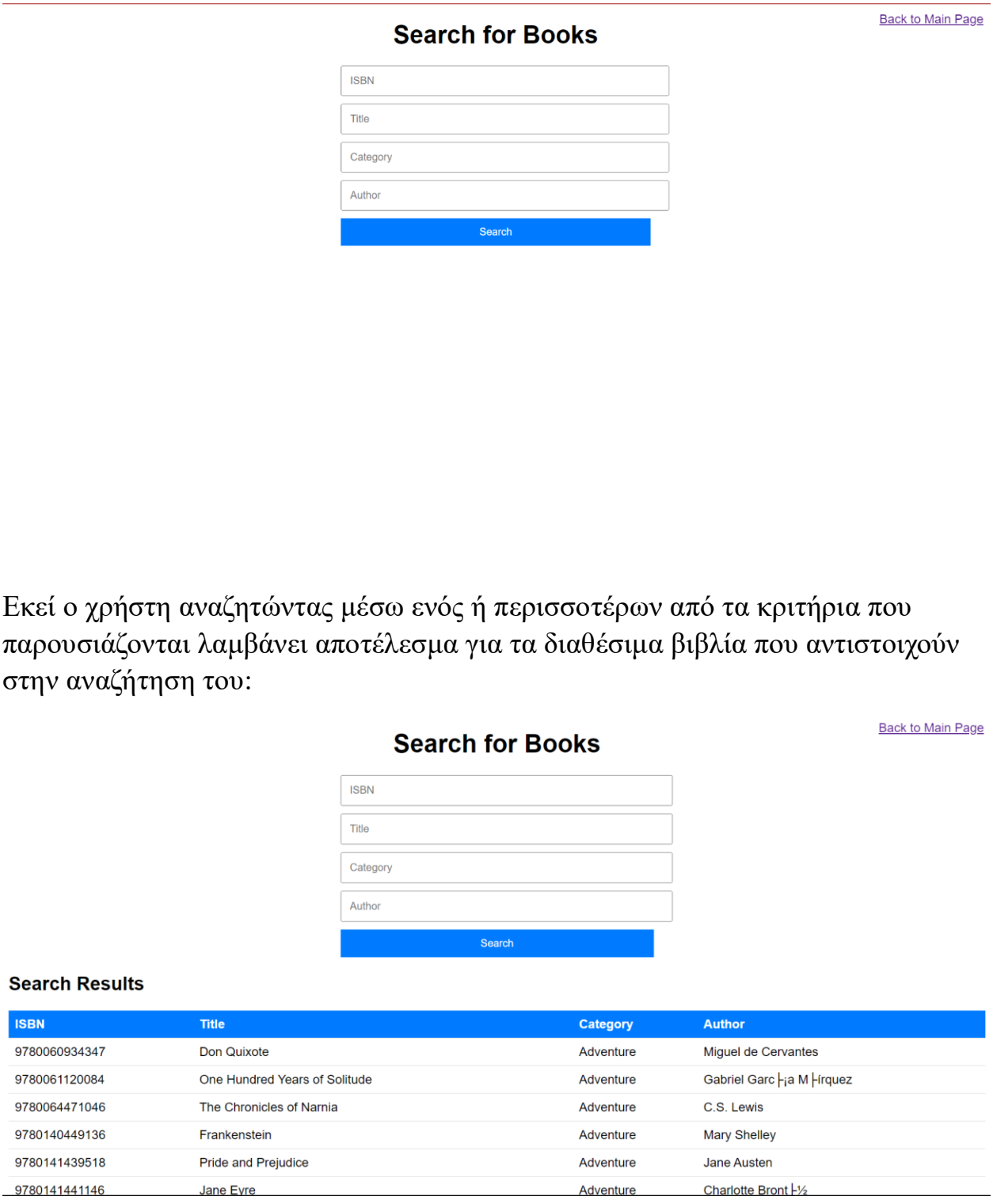
Αφού ο χρήστης συμπληρώσει τα στοιχεία και πατήσει save changes αν τα στοιχεία είναι ορθά οδηγείτε πίσω στη σελίδα login για να συνδεθεί με τον νέο κωδικό.

β. Επεξεργασία στοιχείων αν ο χρήστης είναι καθηγητής ή χειριστής:



Εδώ ο χρήστης όπως στην προηγούμενη σελίδα μπορεί να αλλάξει τα δεδομένα του και να αποθηκεύσει τις αλλαγές και στην συνέχεια ανακατευθύνεται στην κεντρική του σελίδα.

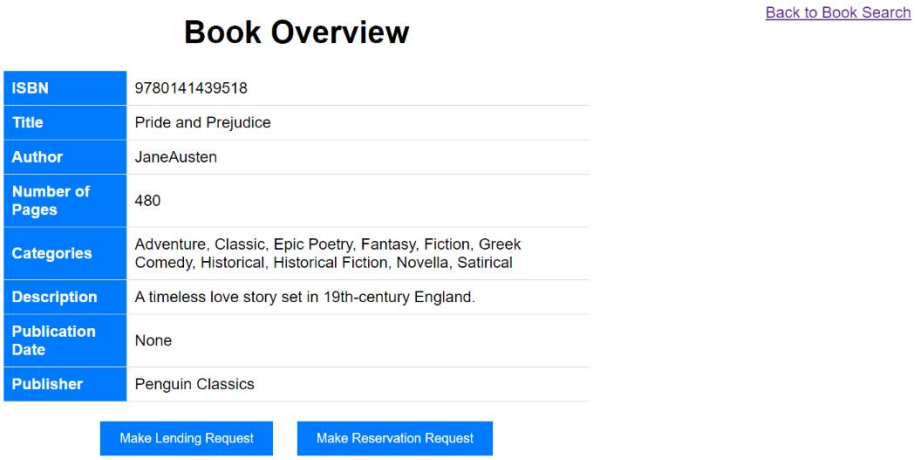
γ. Αναζήτηση βιβλίου πατώντας το link search a book που οδηγεί το χρήστη στην ακόλουθη σελίδα αναζήτησης:



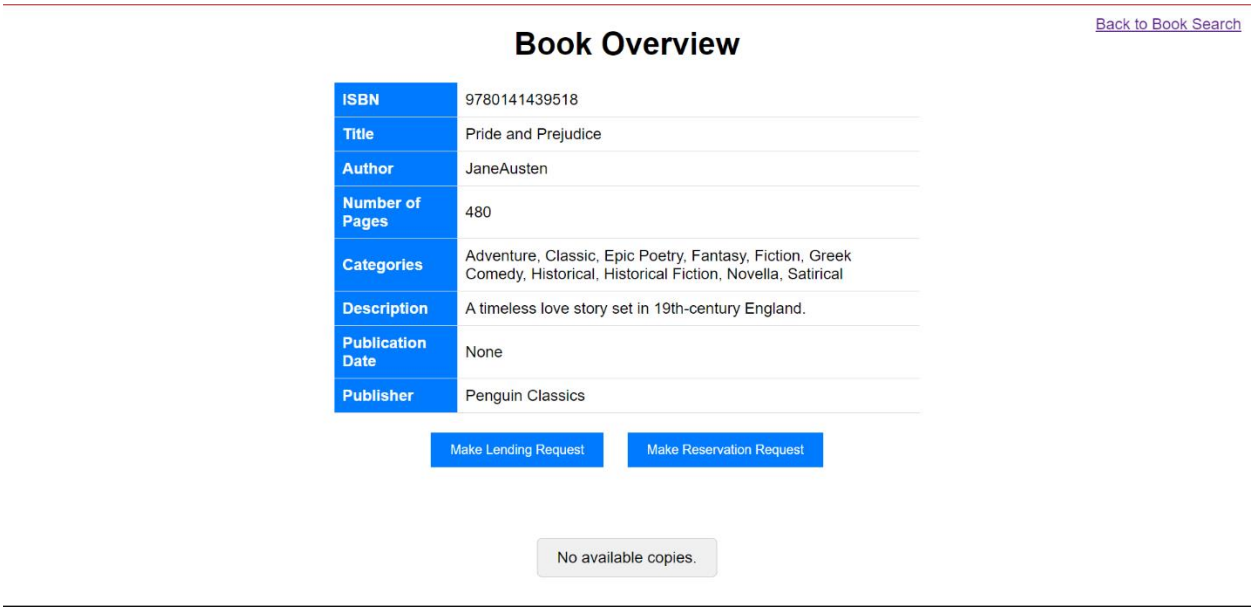
Εκεί ο χρήστη αναζητώντας μέσω ενός ή περισσοτέρων από τα κριτήρια που παρουσιάζονται λαμβάνει αποτέλεσμα για τα διαθέσιμα βιβλία που αντιστοιχούν στην αναζήτηση του:

Πατώντας διπλό click σε όποιο στοιχείο τον ενδιαφέρει ανακατευθύνεται στη σελίδα του βιβλίου, όπου παρουσιάζονται πιο αναλυτικά τα στοιχεία του βιβλίου

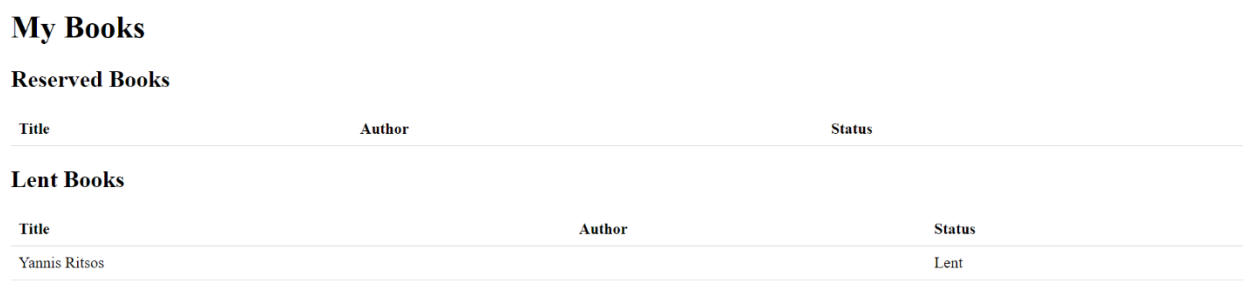
και δίνεται στο χρήστη η επιλογή να κάνει αίτημα δανεισμού ή κράτησης για το βιβλίο:



Ανάλογα με τη διαθεσιμότητα και τις τους τρέχοντες δανεισμούς/κρατήσεις του χρήστη προβάλλεται το αντίστοιχο μήνυμα:



δ. Προβολή τρεχόντων δανεισμών πατώντας την επιλογή My Books στην κεντρική σελίδα:



Εδώ παρουσιάζονται όλα τα βιβλία που έχει δανειστεί αυτή τη στιγμή ο χρήστης όπως και οι κρατήσεις του.