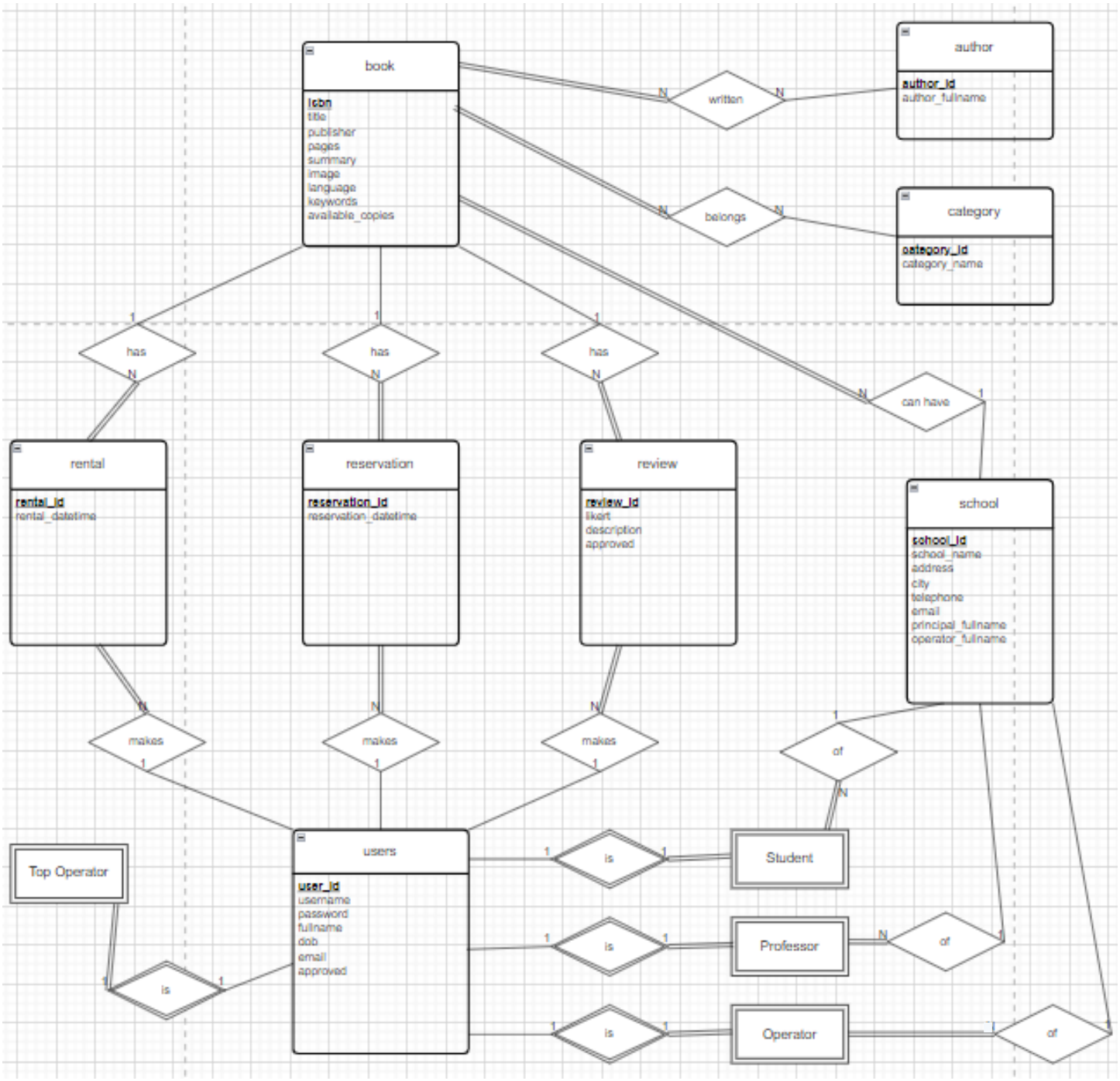
Βάσεις Δεδομένων 6ο εξάμηνο ΣΗΜΜΥ ΕΜΠ

Εξαμηνιαία Εργασία (2022-23)

Φοιτητές: Ανδρέας Καλαβάς (03119709)
Χρίστος Μάης (03120608)

1. Entity – Relationship Diagram

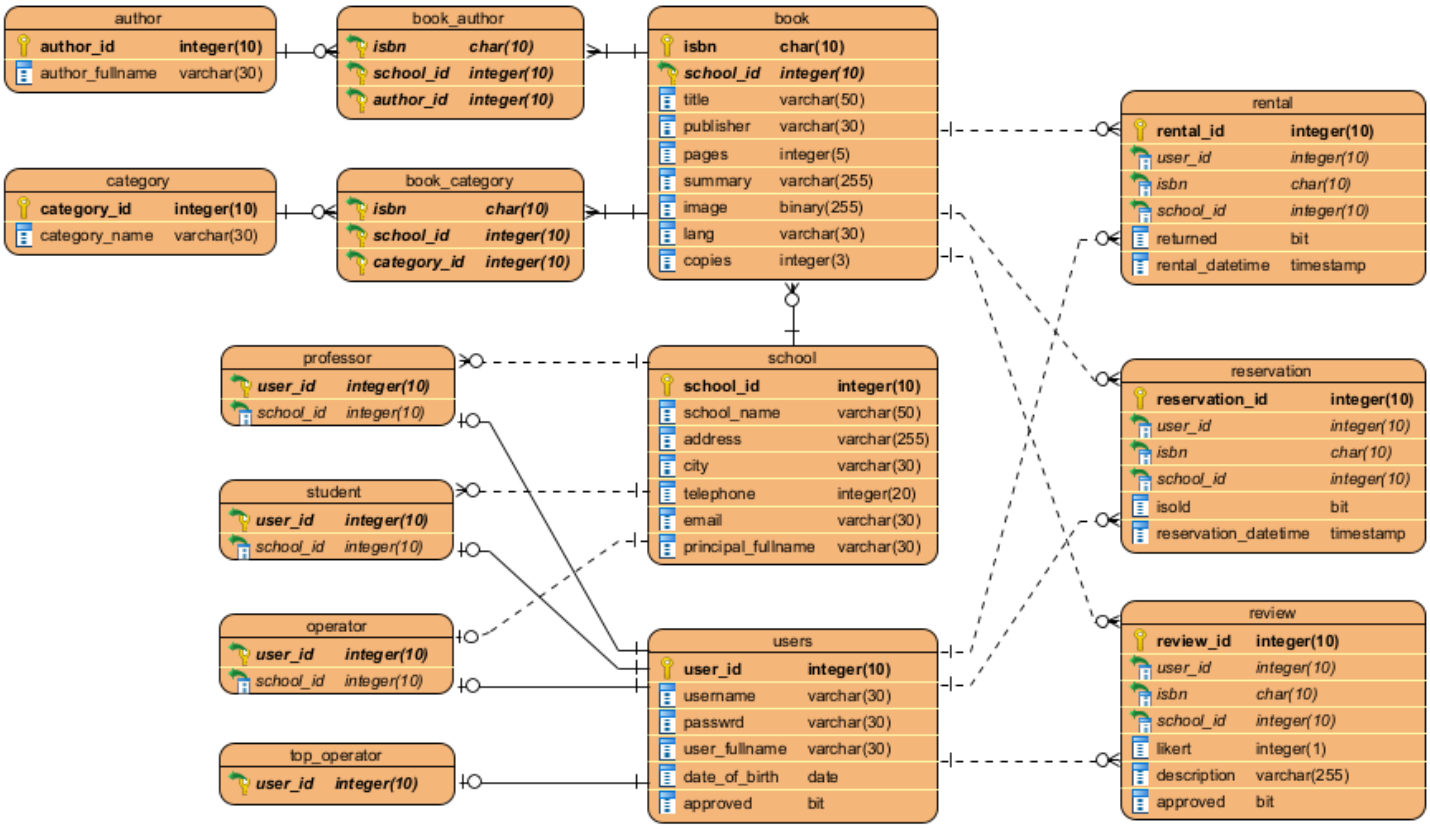
Παρακάτω φαίνεται το διάγραμμα οντοτήτων – συσχετίσεων της βάσης που αναπτύξαμε.



Αποτελείται από 12 οντότητες, και 16 συσχετίσεις. Οι αριθμοί δηλώνουν το αν η συσχέτιση είναι 1 προς 1 (1-1), 1 προς πολλά (1-N), ή πολλά προς πολλά (N-N), και η διπλή γραμμή δηλώνει ολική συμμετοχή των καταχωρήσεων ενός πίνακα. Επίσης οι οντότητες Top Operator, Operator, Professor και Student είναι αδύναμες, και έχουν ως αναγνωρίζουσα οντότητα την Users. Παρακάτω εξηγούμε το διάγραμμα.

- Τα βιβλία έχουν τουλάχιστον 1 συγγραφέα, και κάθε συγγραφέας, έγραψε από 0 μέχρι πολλά βιβλία.
- Το ίδιο ισχύει και για τις κατηγορίες.
- Ένα βιβλίο μπορεί να τύχει πολλών δανεισμών (ή και καθόλου), αλλά ένας δανεισμός αντιστοιχεί πάντα σε 1 βιβλίο.
- Επίσης ένας δανεισμός γίνεται από αυστηρά 1 χρήστη, και ένας χρήστης μπορεί να κάνει από 0 μέχρι πολλούς δανεισμούς.
- Τα 2 παραπάνω σημεία ισχύουν και για τις οντότητες κρατήσεις και αξιολογήσεις.
- Κάθε βιβλίο ανήκει αυστηρά σε 1 σχολείο, ενώ ένα σχολείο μπορεί να έχει πολλά βιβλία.
- Ένας Top Operator αντιστοιχεί αυστηρά σε ένα user (και εξαρτάται από αυτον).
- Αυτό ισχύει και για τις οντότητες Student, Professor και Operator.
- Τέλος, κάθε Student, Professor και Operator ανήκει αυστηρά σε ένα σχολείο, και κάθε σχολείο μπορεί να έχει από 0 μέχρι πολλούς Students και Professors, αλλά μέχρι 1 Operator.

2. Relational Diagram and Database



Από το διάγραμμα οντοτήτων και συσχετίσεων πιο πάνω, το σχεσιακό μοντέλο που προκύπτει είναι το εξής: Κάθε οντότητα του ER, αντιστοιχεί σε έναν πίνακα στο σχεσιακό. Επιπλέον, οι συσχετίσεις μεταξύ βιβλίων και συγγραφέων, και βιβλίων και κατηγοριών χρειάζονται και αυτές πίνακα στο σχεσιακό για να κρατήσουν την σχέση πολλά προς πολλά. Οι σχέσεις 1-προς-1, 1-προς-πολλά και πολλά-προς-πολλά απεικονίζονται πάλι στο μοντέλο. Επίσης στο σχεσιακό διάγραμμα φαίνονται ξεκάθαρα τα πρωτεύοντα και τα ξένα κλειδιά, τα οποία και θα αναλυθούν στη συνέχεια.

1. DDL and DML scripts

Το DDL script δημιουργεί τη βάση και τους πίνακές της, μαζί με τα ευρετήρια και τους διάφορους περιορισμούς. Ο κώδικας για κάθε πίνακα φαίνεται παρακάτω:

```
USE libraries;

CREATE TABLE school(
    school_id integer(10) NOT NULL auto_increment,
    school_name varchar(50) NOT NULL,
    address varchar(255) NOT NULL,
    city varchar(30) NOT NULL,
    telephone integer(20) NOT NULL,
    email varchar(30) NOT NULL,
    principal_fullname varchar(30) NOT NULL,
    PRIMARY KEY (school_id),
    CHECK (email NOT LIKE '% %' AND telephone>0)
);
CREATE INDEX school_index on school(school_id);
```

```
CREATE TABLE book(
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    title varchar(50) NOT NULL,
    publisher varchar(30) NOT NULL,
    pages integer(5) NOT NULL,
    summary varchar(255) NOT NULL,
    image binary(255),
    lang varchar(30) NOT NULL,
    copies integer(3) NOT NULL,
        CONSTRAINT book_of_school FOREIGN KEY (school_id)
        REFERENCES school(school_id) ON DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (isbn,school_id),
    CHECK (pages>0 AND copies>=0 AND CHAR_LENGTH(isbn)=10
    AND (SUBSTRING(isbn,1,1) LIKE '0' OR SUBSTRING(isbn,1,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,2,1) LIKE '0' OR SUBSTRING(isbn,2,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,3,1) LIKE '0' OR SUBSTRING(isbn,3,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,4,1) LIKE '0' OR SUBSTRING(isbn,4,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,5,1) LIKE '0' OR SUBSTRING(isbn,5,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,6,1) LIKE '0' OR SUBSTRING(isbn,6,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,7,1) LIKE '0' OR SUBSTRING(isbn,7,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,8,1) LIKE '0' OR SUBSTRING(isbn,8,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,9,1) LIKE '0' OR SUBSTRING(isbn,9,1) LIKE '1' OR SU
    AND (SUBSTRING(isbn,10,1) LIKE '0' OR SUBSTRING(isbn,10,1) LIKE '1' OR
);
CREATE INDEX book_index on book(isbn,school_id);
```

```
CREATE TABLE author(
    author_id integer(10) NOT NULL auto_increment,
    author_fullname varchar(30) NOT NULL UNIQUE,
    PRIMARY KEY (author_id)
);
CREATE INDEX author_index on author(author_id);
```

```
CREATE TABLE book_author(
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    author_id integer(10) NOT NULL,
        CONSTRAINT book_written_by FOREIGN KEY (isbn, school_id)
        REFERENCES book(isbn,school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT author_wrote FOREIGN KEY (author_id)
        REFERENCES author(author_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (isbn,school_id,author_id)
);
CREATE INDEX book_author_index on book_author(author_id);
```

```
CREATE TABLE category(
    category_id integer(10) NOT NULL auto_increment,
    category_name varchar(30) NOT NULL UNIQUE,
    PRIMARY KEY (category_id)
);
CREATE INDEX category_index on category(category_id);
```

```
CREATE TABLE book_category(
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    category_id integer(10) NOT NULL,
        CONSTRAINT book_written FOREIGN KEY (isbn, school_id)
        REFERENCES book(isbn,school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT category_of_book FOREIGN KEY (category_id)
        REFERENCES category(category_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (isbn,school_id,category_id)
);
CREATE INDEX book_category_index on book_category(category_id);
```

```
CREATE TABLE users(
    user_id integer(10) NOT NULL auto_increment,
    username varchar(30) NOT NULL UNIQUE,
    passwrd varchar(30) NOT NULL,
    user_fullname varchar(30) NOT NULL,
    date_of_birth date NOT NULL,
    approved boolean NOT NULL,
    PRIMARY KEY (user_id),
    CHECK (date_of_birth< date '2015-01-01')
);
CREATE INDEX users_index on users(user_id);
```

```
CREATE TABLE top_operator(
    user_id integer(10) NOT NULL,
        CONSTRAINT is_topop FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (user_id)
);
```

```sql
CREATE TABLE operator(
    user_id integer(10) NOT NULL,
    school_id integer(10) NOT NULL,
        CONSTRAINT is_op FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT op_of_school FOREIGN KEY (school_id)
        REFERENCES school(school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (user_id)
);
CREATE INDEX operator_index on operator(user_id);
```

```sql
CREATE TABLE professor(
    user_id integer(10) NOT NULL,
    school_id integer(10) NOT NULL,
        CONSTRAINT is_prof FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT prof_of_school FOREIGN KEY (school_id)
        REFERENCES school(school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (user_id)
);
CREATE INDEX professor_index on professor(user_id);
```

```sql
CREATE TABLE student(
    user_id integer(10) NOT NULL,
    school_id integer(10) NOT NULL,
        CONSTRAINT is_student FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT student_of_school FOREIGN KEY (school_id)
        REFERENCES school(school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (user_id)
);
CREATE INDEX student_index on student(user_id);
```

```sql
CREATE TABLE rental(
    rental_id integer(10) NOT NULL auto_increment,
    user_id integer(10) NOT NULL,
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    returned boolean NOT NULL,
    rental_datetime timestamp NOT NULL,
        CONSTRAINT person_who_rent FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT book_rent FOREIGN KEY (isbn,school_id)
        REFERENCES book(isbn,school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (rental_id)
);
CREATE INDEX rental_index on rental(rental_id);
```

```sql
CREATE TABLE reservation(
    reservation_id integer(10) NOT NULL auto_increment,
    user_id integer(10) NOT NULL,
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    isold boolean NOT NULL,
    reservation_datetime timestamp NOT NULL,
        CONSTRAINT person_who_reserved FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT book_resrved FOREIGN KEY (isbn,school_id)
        REFERENCES book(isbn,school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (reservation_id)
);
CREATE INDEX reservation_index on reservation(reservation_id);
```

```
CREATE TABLE review(
    review_id integer(10) NOT NULL auto_increment,
    user_id integer(10) NOT NULL,
    isbn char(10) NOT NULL,
    school_id integer(10) NOT NULL,
    likert integer(1) NOT NULL,
    description varchar(255),
    approved bit NOT NULL,
        CONSTRAINT person_who_reviewed FOREIGN KEY (user_id)
        REFERENCES users(user_id) ON  DELETE CASCADE ON UPDATE CASCADE,
        CONSTRAINT book_reviewed FOREIGN KEY (isbn,school_id)
        REFERENCES book(isbn,school_id) ON  DELETE CASCADE ON UPDATE CASCADE,
    PRIMARY KEY (review_id),
    CHECK (likert>=1 AND likert<=5)
);
CREATE INDEX review_index on review(review_id);
```

Τα κλειδιά, τα ευρετήρια και οι διάφοροι περιορισμοί φαίνονται στον κωδικα, και αναλύονται παρακάτω. Για DML script, τα μόνα δεδομένα που εισάγουμε στη βάση σε αυτό το αρχικό στάδιο είναι αυτά του Top Operator. Επίσης δημιουργήσαμε το script dbdata.sql, το οποίο εισάγει στη βάση αρκετά δεδομένα (πιο αναλυτικά παρακάτω). Τα δεδομένα του Top Operator εισάγονται εκτελώντας τα παρακάτω queries.

```
insert into users (username,passwrd,user_fullname,date_of_birth,approved)
values ('topoperator','password','TOP OPERATOR',DATE '2000-01-01',true);
insert into top_operator values ((select user_id from users where username = 'topoperator'));
```

2. Περιορισμοί

➢ Περιορισμοί Ακεραιότητας

- Για τη δήλωση Σχολικής μονάδας πρέπει να συμπληρώνονται τα πεδία:
  Όνομα σχολείου, διεύθυνση, πόλη, τηλέφωνο, email και το όνομα του διευθυντή. Αυτό ισχύει και όταν ο κεντρικός διαχειριστής τροποποιήσει τα στοιχεία ενός σχολείου.
- Για τη δήλωση του Προφίλ ενός χρήστη πρέπει να συμπληρώνονται τα πεδία:
  Επιλογή χρήστη(μαθητής/καθηγητής/χειριστής), Επιλογή σχολείου, Ονοματεπώνυμο, Ημ.Γέννησης, Όνομα χρήστη(username), κωδικός(password). Αυτό ισχύει και όταν ο χρήστης τροποποιήσει τα στοιχεία του.
- Για την δήλωση ενός βιβλίου στο σύστημα πρέπει να συμπληρώνονται τα πεδία:
  ISBN, Τίτλος βιβλίου, Συγγραφέας/είς, Κατηγορία/ες, Εκδότης, Σελίδες, Περίληψη, Γλώσσα, Διαθέσιμα αντίτυπα. Αυτό ισχύει και όταν ο χειριστής τροποποιήσει τα στοιχεία ενός βιβλίου.
- Για την καταχώρηση αξιολόγησης ενός βιβλίου πρέπει να συμπληρώνονται τα πεδία:
  Βαθμολογία βιβλίου, Περιγραφή.

➢ Πρωτεύοντα Κλειδιά

- Ο πίνακας **school** έχει ως πρωτεύον κλειδί το **school_id**.
- Ο πίνακας **users** έχει ως πρωτεύον κλειδί το **user_id**.
- Ο πίνακας **top_operator** έχει ως πρωτεύον κλειδί το **user_id**.
- Ο πίνακας **operator** έχει ως πρωτεύον κλειδί το **user_id**.
- Ο πίνακας **professor** έχει ως πρωτεύον κλειδί το **user_id**.
- Ο πίνακας **student** έχει ως πρωτεύον κλειδί το **user_id**.
- Ο πίνακας **book** έχει ως πρωτεύον κλειδί τον συνδυασμό πεδίων **isbn** και **school_id**.
- Ο πίνακας **author** έχει ως πρωτεύον κλειδί το **author_id**.
- Ο πίνακας **book_author** έχει ως πρωτεύον κλειδί τον συναδυασμό **author_id**, **isbn** και **school_id**.
- Ο πίνακας **category** έχει ως πρωτεύον κλειδί το **category_id**.
- Ο πίνακας **book_category** έχει ως πρωτεύον κλειδί τον συναδυασμό **category_id**, **isbn** και **school_id**.
- Ο πίνακας **rental** έχει ως πρωτεύον κλειδί το **rental_id**.
- Ο πίνακας **reservation** έχει ως πρωτεύον κλειδί το **reservation_id**.
- Ο πίνακας **review** έχει ως πρωτεύον κλειδί το **review_id**.

- **Ξένα Κλειδιά**

  - Ο πίνακας **top_operator** έχει ξένο κλειδί το **user_id** του πίνακα users.
  - Ο πίνακας **operator** έχει ξένα κλειδιά το **user_id** του πίνακα users και το **school_id** του πίνακα school.
  - Ο πίνακας **professor** έχει ξένα κλειδιά το **user_id** του πίνακα users και το **school_id** του πίνακα school.
  - Ο πίνακας **student** έχει ξένα κλειδιά το **user_id** του πίνακα users και το **school_id** του πίνακα school.
  - Ο πίνακας **book** έχει ξένο κλειδί το **school_id** του πίνακα school.
  - Ο πίνακας **book_author** έχει ξένα κλειδιά το **author_id** από τον πίνακα author, το **isbn** και το **school_id** του πίνακα book.
  - Ο πίνακας **book_category** έχει ξένα κλειδιά το **category_id** από τον πίνακα category, το **isbn** και το **school_id** του πίνακα book.
  - Ο πίνακας **rental** έχει ξένα κλειδιά το **user_id** από του πίνακα users, το **isbn** και το **school_id** του book.
  - Ο πίνακας **reservation** έχει ξένα κλειδιά το **user_id** από τον πίνακα users, το **isbn** και το **school_id** του book.
  - Ο πίνακας **review** έχει ξένα κλειδιά το **user_id** από τον πίνακα users, το **isbn** και το **school_id** του book.

- **Αναφορική Ακεραιότητα**
  - Όταν διαγράφεται ή τροποποιείται μια Σχολική μονάδα, όλες οι σχετικές εγγραφές εντός του σχολείου θα πρέπει επίσης να διαγραφούν ή να ενημερωθούν αντίστοιχα (χρήστες, βιβλία).
  - Όταν διαγράφεται ή τροποποιείται ένα βιβλίο, όλες οι σχετικές εγγραφές που αφορούν το βιβλίο θα πρέπει να διαγράφονται ή να ενημερώνονται ανάλογα.
- **Ακεραιότητα Πεδίου Τιμών:** Οι τύποι δεδομένων των χαρακτηριστικών πρέπει να ταιριάζουν με τις αντίστοιχες τιμές τους, και να συμφωνούν και με τα ζητούμενα της εκφώνησης. Πιο συγκεκριμένα:
  - Το isbn πρέπει να είναι 10ψήφιος αριθμός, ή 9ψήφιος αριθμός ακολουθόμενος από τον χαρακτήρα X.
  - Η βαθμολογία ενός βιβλίου θα πρέπει να είναι αριθμός από το 1 μέχρι το 5.
  - Το email δεν πρέπει να περιέχει κενό.
  - Οι ημερομηνίες πρέπει να είναι λογικές (πχ ημερομηνίες γέννησης πριν το 2015).
- **Περιορισμοί Καθορισμένοι από Χρήστη**
  - Το username ενός χρήστη πρέπει να είναι μοναδικό (για να μπορεί να γίνεται σωστά η επαλήθευση).
  - Μοναδικό επίσης πρέπει να είναι το όνομα κάθε σχολείου, η κατηγορία, και το όνομα συγγραφέα (για λόγους σαφήνειας).
  - Κάθε μαθητής μπορεί να δανειστεί 2 βιβλία τη φορά (σε περίοδο μιας βδομάδας), ενώ οι καθηγητές δικαιούνται μόνο 1.
  - Το ίδιο ισχύει και για τις κρατήσεις.
  - Επιπλέον για να γίνει κράτηση πρέπει να μην υπάρχει δανεισμός από τον ίδιο χρήστη.
  - Οι κρατήσεις διαγράφονται μετά από 1 βδομάδα. Όταν ένας δανεισμός καταχωρηθεί, θα πρέπει να υπάρχει μηχανισμός ενημέρωσης των διαθέσιμων αντιτύπων του αντίστοιχου βιβλίου.

3. Εισαγωγή πληροφοριών

Το dbdata.sql script εισάγει στη βάση δεδομένα. Συγκεκριμένα εισάγει 135 βιβλία, 52 κατηγορίες, 22 συγγραφείς, 4 σχολικές μονάδες με τους χειριστές τους , 21 μαθητές, 15 καθηγητές, 600 αξιολογήσεις, 901 δανεισμούς και 35 κρατήσεις.

4. Indexes

Όπως φαίνεται και από το DDL script, τα περισσότερα ευρετήρια είναι και τα πρωτεύοντα κλειδιά (τα οποία είναι της μορφής (xxx_id). Επιλέξαμε αυτά τα ευρετήρια επειδή είναι αυτά τα πεδία που κληρονομούνται ως ξένα κλειδιά σε άλλους πίνακας, και ο κύριος τρόπος να κάνουμε join πίνακες στα queries της εφαρμογής. Οι μόνοι πίνακες με ευρετήριο διαφορετικό από το πρωτεύον κλειδί είναι ο book_author και ο book_category. Αυτοί έχουν ως ευρετήρια το πεδίο author_id και category_id αντίστοιχα. Αυτό γίνεται γιατί, όπως και στην περίπτωση που το ευρετήριο είναι ίδιο με το πρωτεύον κλειδί, τα πεδία αυτά χρησιμοποιούνται περισσότερο στην ένωση πινάκων, ιδιαίτερα του category και του author.

3. Ανάπτυξη Εφαρμογής και Ερωτήματα (Queries)

Η εφαρμογή αναπτύχθηκε σε περιβάλλον Node.js (v18.13.0), και χρησιμοποιήθηκε ο MySQL Workbench server. Έχει αρχιτεκτονική REST, όπου το backend υλοποιήθηκε με το framework Express, και το frontend με το framework React. Εκτός από τα ζητούμενα use case της ιστοσελίδας, παρακάτω αναλύονται τα ζητούμενα ερωτήματα.

1. Ερωτήματα Διαχειριστή

1. Παρουσίαση λίστας με συνολικό αριθμό δανεισμών ανά σχολείο (Κριτήρια αναζήτησης: έτος, μήνας)

Θέλουμε να μετρήσουμε τους δανεισμούς κάθε σχολείου, οπότε κάνουμε COUNT τους δανεισμούς, και GROUP BY το σχολείο. Επίσης χρησιμοποιούμε LEFT JOIN, για να μην χαθούν τα σχολεία τα οποία δεν έχουν κρατήσεις ακόμη. Επιπλέον, γίνεται χρήση της συνάρτησης CAST, για να μην χρειάζονται παραλλαγές του ερωτήματος, σε περίπτωση όπου κάποιο κριτήριο μείνει κενό.

```
let year = "%";
if (req.params.year !== "none") year = req.params.year + year;
let month = "%";
if (req.params.month !== "none") month = req.params.month + month;
if (month[0] == "0") month = month[1];

const ans_list = await conn.query(
  `SELECT school.school_name, COUNT(rental.rental_id) AS num_of_rents
  FROM school
  LEFT JOIN rental ON school.school_id = rental.school_id
  AND CAST(YEAR(rental.rental_datetime) AS CHAR(4)) LIKE ?
  AND CAST(MONTH(rental.rental_datetime) AS CHAR(4)) LIKE ?
  GROUP BY school.school_id`,
  [year, month, year, month]
);
```

2. Για δεδομένη κατηγορία βιβλίων, ποιοι συγγραφείς ανήκουν σε αυτήν και ποιοι εκπαιδευτικοί έχουν δανειστεί βιβλία αυτής της κατηγορίας το τελευταίο έτος; Εδώ κάνουμε UNION 2 παρόμοια queries όπου το πρώτο μας δίνει τους συγγραφείς, και το άλλο τους καθηγητές.

```
const ans_list = await conn.query(
  `SELECT author.author_fullname AS author, 0 AS professor
  FROM category
  INNER JOIN book_category ON category.category_id = book_category.category_id
  INNER JOIN book ON book_category.school_id = book.school_id AND book_category.isbn = book.isbn
  INNER JOIN book_author ON book.isbn = book_author.isbn AND book.school_id = book_author.school_id
  INNER JOIN author ON book_author.author_id = author.author_id
  WHERE category.category_id = ?

  UNION

  SELECT 0 AS author, users.user_fullname AS professor
  FROM category
  INNER JOIN book_category ON category.category_id = book_category.category_id
  INNER JOIN book ON book_category.school_id = book.school_id AND book_category.isbn = book.isbn
  INNER JOIN rental ON book.isbn = rental.isbn AND book.school_id = rental.school_id
  INNER JOIN users ON rental.user_id = users.user_id
  INNER JOIN professor ON users.user_id = professor.user_id
  WHERE category.category_id = ? AND TIMESTAMPDIFF(YEAR, rental.rental_datetime, NOW()) < 1`,
  [req.params.catID, req.params.catID]
);
```

3. Βρείτε τους νέους εκπαιδευτικούς (ηλικία < 40 ετών) που έχουν δανειστεί τα περισσότερα βιβλία και των αριθμό των βιβλίων. Εδώ εκτελούμε το ερώτημα χρησιμοποιώντας παρόμοιους τρόπους με πιο πάνω, και κάνουμε επίσης ORDDER BY τον αριθμό των βιβλίων, για να πάρουμε τα αποτελέσματα σε φθίνουσα σειρά.

```
const ans_list = await conn.query(
  `SELECT users.user_fullname AS fullname, COUNT(rental.rental_id) AS num_of_books
  FROM rental
  INNER JOIN users ON rental.user_id = users.user_id
  INNER JOIN professor ON users.user_id = professor.user_id
  WHERE TIMESTAMPDIFF(YEAR, users.date_of_birth, NOW()) < 40
  GROUP BY users.user_id
  ORDER BY num_of_books DESC`
);
```

4. Βρείτε τους συγγραφείς των οποίων κανένα βιβλίο δεν έχει τύχει δανεισμού. Εδώ χρησιμοποιείται το HAVING μετά το GROUP BY για να φιλτράρει τους τους δανεισμούς που έγιναν στα βιβλία κάθε συγγραφέα. Στην περίπτωση αυτή κρατάει μόνο όσους συγγραφείς δεν έτυχε να δανειστούν βιβλίο τους.

```
const ans_list = await conn.query(
  `SELECT DISTINCT author.author_fullname AS author
  FROM author
  INNER JOIN book_author ON author.author_id = book_author.author_id
  INNER JOIN book ON book_author.isbn = book.isbn AND book_author.school_id = book.school_id
  LEFT JOIN rental ON book.isbn = rental.isbn AND book.school_id = rental.school_id
  GROUP BY author.author_fullname
  HAVING COUNT(rental.rental_id) = 0;`
);
```

5. Ποιοι χειριστές έχουν δανείσει τον ίδιο αριθμό βιβλίων το τελευταίο έτος με περισσότερους από 20 δανεισμούς; Ας αναλύσουμε κομμάτι κομμάτι αυτό το ερώτημα. Το υποερώτημα Q1 επιστρέφει τον αριθμό των δανεισμών κάθε σχολείου. Στη συνέχεια τα αποτελέσματα φιλτράρονται με το HAVING, όπου μένουν μόνο τα σχολεία που έχουν ίδιο αριθμό δανεισμών με τουλάχιστον ακόμη ένα άλλο σχολείο. Το υποερώτημα Q2 επιστρέφει τον αριθμό δανεισμών κάθε χειριστή (που ισούται με τον αριθμό δανεισμών του αντίστοιχου σχολείου). Τελικά, ολόκληρο το ερώτημα επιστρέφει τους χειριστές και τον αριθμό δανεισμών που έκαναν, εφόσον αυτός ο αριθμός ανήκει στα αποτελέσματα του φιλτραρισμένου ερωτήματος Q1.

```
const ans_list = await conn.query(
  `SELECT operator, num_of_rents
  FROM (
      SELECT users.user_fullname AS operator, COUNT(rental.rental_id) AS num_of_rents
      FROM rental
      INNER JOIN operator ON rental.school_id = operator.school_id
      INNER JOIN users ON operator.user_id = users.user_id
      WHERE TIMESTAMPDIFF(YEAR, rental.rental_datetime, NOW()) < 1
      GROUP BY operator.user_id
      ORDER BY num_of_rents DESC
  ) AS Q2
  WHERE Q2.num_of_rents IN (
      SELECT num_of_rents
      FROM (
          SELECT rental.school_id AS school, COUNT(rental.rental_id) AS num_of_rents
          FROM rental
          WHERE TIMESTAMPDIFF(YEAR, rental.rental_datetime, NOW()) < 1
          GROUP BY school
          HAVING num_of_rents > ?
          ORDER BY num_of_rents DESC
      ) AS Q1
      GROUP BY Q1.num_of_rents
      HAVING COUNT(Q1.school) > ?
  )
  ORDER BY num_of_rents DESC`,
  [min_no_of_rents, min_no_of_ops]
);
```

6. Ανάμεσα σε ζεύγη κατηγοριών που είναι κοινά στα βιβλία, βρείτε τα 3 κορυφαία (top-3) ζεύγη που εμφανίστηκαν σε δανεισμούς. Εδώ κάνουμε JOIN τον πίνακα category με τον εαυτό του, για να πάρουμε όλα τα πιθανά ζεύγη (χρησιμοποιούμε τον έλεγχο c1.category_id < c2.category_id για να παρουμε μια φορά κάθε ζεύγος). Στη συνέχεια ελέγχουμε ότι το βιβλίο ανήκει και στις 2 κατηγορίες, και μετά ταξινομούμε τα αποτελέσματα ανάλογα με τον αριθμό δανεισμών όλων των βιβλίων που ανήκουν στις 2 κατηγορίες. Τέλος περιορίζουμε τον αριθμό των εγγραφών σε τρεις.

```
const ans_list = await conn.query(
  `SELECT c1.category_name as cat1, c2.category_name as cat2, COUNT(rental.rental_id) as num_of_rents
  FROM category AS c1
  JOIN category AS c2 ON c1.category_id < c2.category_id
  JOIN book_category AS b1 ON c1.category_id = b1.category_id
  JOIN book_category AS b2 ON c2.category_id = b2.category_id
  JOIN rental ON rental.school_id = b1.school_id AND rental.isbn = b1.isbn
  WHERE b1.isbn = b2.isbn AND b1.school_id = b2.school_id
  GROUP BY cat1, cat2
  ORDER BY num_of_rents DESC
  LIMIT 3`
);
```

7. Βρείτε όλους τους συγγραφείς που έχουν γράψει τουλάχιστον 5 βιβλία λιγότερα από τον συγγραφέα με τα περισσότερα βιβλία. Αρχικά βρίσκουμε όλους τους συγγραφείς και τον αριθμό βιβλίων που έγραψαν, και με το HAVING φιλτράρουμε έτσι ώστε να μας μείνουν αυτοί που έχουν γράψει τουλάχιστον 5 λιγότερα από αυτόν που έγραψε τα περισσότερα (τον οποίο βίσκουμε με ένα υποερώτημα, ταξινομώντας τα αποτελέσματα σε φθίνουσα σειρά, και περιορίζοντας το στη μία εγγραφή).

```
const ans_list = await conn.query(
  `SELECT a.author_fullname, COUNT(*) as num_of_books
   FROM author a
   JOIN book_author ba ON a.author_id = ba.author_id
   GROUP BY a.author_id, a.author_fullname
   HAVING num_of_books < (
       SELECT COUNT(*)
       FROM book_author
       GROUP BY author_id
       ORDER BY COUNT(*) DESC
       LIMIT 1
   ) -4
   ORDER BY num_of_books DESC`
);
```

2. Ερωτήματα Χειριστή

1. Παρουσίαση όλων των βιβλίων κατά Τίτλο, Συγγραφέα (Κριτήρια αναζήτησης: τίτλος/ κατηγορία/ συγγραφέας/ αντίτυπα). Υλοποιείται με πολύ παρόμοιο τρόπο όσον αφορά τα κριτήρια αναζήτησης με το ερώτημα 1 του διαχειριστή, όπου σε περίπτωση που κάποιο μείνει κενό δεν λαμβάνεται υπόψη. (εικόνα δεξιά)

```
let title = "%";
if (req.params.title !== "none") title = req.params.title + title;
let category = "%";
if (req.params.category !== "none")
  category = req.params.category + category;
let author = "%";
if (req.params.author !== "none") author = req.params.author + author;
let copies = "%";
if (req.params.copies !== "none") copies = req.params.copies + copies;
const ans_list = await conn.query(
  `SELECT b.title, a.author_fullname
      FROM book AS b
      JOIN book_author AS ba ON b.isbn = ba.isbn AND b.school_id = ba.school_id
      JOIN author AS a ON ba.author_id = a.author_id
      JOIN book_category AS bc ON b.isbn = bc.isbn AND b.school_id = bc.school_id
      JOIN category AS c ON bc.category_id = c.category_id
      WHERE
      (b.title LIKE ?)
      AND (c.category_name LIKE ?)
      AND (a.author_fullname LIKE ?)
      AND (b.copies LIKE ?)
      AND b.school_id = ?
      ORDER BY b.title`,
  [title, category, author, copies, req.params.schlid]
);
```

2. Εύρεση όλων των δανειζόμενων που έχουν στην κατοχή τους τουλάχιστον ένα βιβλίο και έχουν καθυστερήσει την επιστροφή του (Κριτήρια αναζήτησης: Όνομα, Επώνυμο, Ημέρες Καθυστέρησης). Πάλι υλοποιείται με παρόμοιους τρόπους όπως παραπάνω. Πρέπει να προσέξουμε επίσης ότι στη βάση αποθηκεύουμε το όνομα και το επίθετο μαζί, ως fullname, έτσι ο έλεγχος αυτός πρέπει να γίνει χρησιμοποιώντας χαρακτήρες %. (εικόνα κάτω)

```
let name = "%";
if (req.params.name !== "none") name = req.params.name + name;
if (req.params.surname !== "none")
  name = name + " " + req.params.surname + "%";
let ddays = 0;
if (req.params.delay_days !== "none")
  ddays = Number(req.params.delay_days);
ddays += 7;
const ans_list = await conn.query(
  `SELECT COUNT(rental.rental_id) AS num_books,users.user_fullname, DATEDIFF(CURDATE(),
   MAX(rental.rental_datetime))-7 AS days_delayed
   FROM rental
   JOIN users ON rental.user_id = users.user_id
   WHERE rental.school_id = ? AND users.user_fullname LIKE ? AND rental.returned = false
   AND users.user_id = rental.user_id AND TIMESTAMPDIFF(DAY, rental.rental_datetime, NOW()) > ?
   GROUP BY users.user_id
   ORDER BY users.user_fullname`,
  [req.params.schlid, name, ddays]
);
```

3. Μέσος Όρος Αξιολογήσεων ανά δανειζόμενο και κατηγορία (Κριτήρια αναζήτησης: χρήστης/ κατηγορία). Το ερώτημα επιστρέφει τον μέσο όρο αξιολογήσεων για όλους τους χρήστες και για όλες τις κατηγορίες. Μπορούν πάλι να προστεθούν ως κριτήρια ο χρήστης (username) και η κατηγορία (category_name), τα οποία ελέγχονται με τη βοήθεια των χαρακτήρων %. Επίσης γίνεται φθίνουσα ταξινόμηση των αποτελεσμάτων.

```
let user = "%";
if (req.params.user !== "none") user = req.params.user + user;
let category = "%";
if (req.params.category !== "none")
  category = req.params.category + category;

const ans_list = await conn.query(
  `SELECT users.username, category.category_name AS category, AVG(review.likert) AS avguser_rating
      FROM users
      JOIN review ON users.user_id = review.user_id
      JOIN book_category ON review.isbn = book_category.isbn AND review.school_id = book_category.school_id
      JOIN category ON book_category.category_id=category.category_id
      WHERE users.username LIKE ? AND category.category_name LIKE ? AND review.school_id = ?
      GROUP BY users.username, category.category_name
      ORDER BY avguser_rating DESC`,
  [user, category, req.params.schlid]
);
```

3. Ερωτήματα Χρήστη

1. Όλα τα βιβλία που έχουν καταχωριστεί (Κριτήρια αναζήτησης: τίτλος/ κατηγορία/ συγγραφέας), δυνατότητα επιλογής βιβλίου και δημιουργία αιτήματος κράτησης. Πολύ παρόμοιο με το ερώτημα 3.2.1.

```
`SELECT b.title, b.isbn , a.author_fullname AS author, c.category_name AS category
    FROM book AS b
    JOIN book_author AS ba ON b.isbn = ba.isbn AND b.school_id = ba.school_id
    JOIN author AS a ON ba.author_id = a.author_id
    JOIN book_category AS bc ON b.isbn = bc.isbn AND b.school_id = bc.school_id
    JOIN category AS c ON bc.category_id = c.category_id
    WHERE
    (b.title LIKE ?)
    AND (c.category_name LIKE ?)
    AND (a.author_fullname LIKE ?)
    AND (b.school_id = ?)
    ORDER BY b.title`,
```

2. Λίστα όλων των βιβλίων που έχει δανειστεί ο συγκεκριμένος χρήστης. Δεν υπάρχει κάτι ιδαίτερο για να σχολιαστεί

```
const ans_list = await conn.query(
  `SELECT DISTINCT book.title
      FROM book
      JOIN rental ON book.isbn = rental.isbn AND book.school_id = rental.school_id
      JOIN users ON rental.user_id = users.user_id
      WHERE users.user_id LIKE ?`,
  [req.params.userID]
);
```

4. User Manual

Η εφαρμογή μας είναι φιλική προς τον χρήστη, ο οποίος δεν χρειάζεται να έχει κάποια εξειδικευμένη γνώση για να την χρησιμοποιήσει. Οι σελίδες της εφαρμογής είναι οι ακόλουθες:

1. Log In: Ο χρήστης μπορεί να επαληθεύσει τα στοιχεία του, και να συνδεθεί με τον λογαριασμό του.
2. Sign Up: Ο χρήστης μπορεί να κάνει αίτηση εγγραφής σαν χειριστής, καθηγητής ή μαθητής και για συγκεκριμένο σχολείο
3. Profile: Ο χρήστης μπορεί να δει τα στοιχεία του, και να επιλέξει να τα τροποποιήσει, ή να αποσυνδεθεί. Αν είναι ο διαχειριστής έχει και τη δυνατότητα να κάνει back up και restore ολόκληρη τη βάση.
4. Edit Profile: Ο χρήστης μπορεί να αλλάξει τα στοιχεία του (εκτός του username του). Αν είναι μαθητής τότε μπορεί να αλλάξει μόνο τον κωδικό του.
5. School Units: Ο διαχειριστής μπορεί να δει τα υπάρχοντα σχολεία, να επιλέξει κάποιο συγκεκριμένο για να δει περαιτέρω πληροφορίες, ή να προσθέσει νέο.
6. Add School Unit: Ο διαχειριστής μπορεί να καταχωρίσει στοιχεία νέου σχολείου.
7. School: Φαίνονται τα στοιχεία του σχολείου, και υπάρχει επιλογή για διαγραφή ή τροποποίησή του.
8. Edit School: Ο διαχειριστής μπορεί να τροποποιήσει τα στοιχεία του σχολείου.
9. Operators: Ο διαχειριστής μπορεί να δει τους χειριστές, χωρισμένους σε εγκεκριμένους και μη (αυτοί που έκαναν απλά αίτηση).
10. Operator: Ο διαχειριστής μπορεί να δει τα στοιχεία ενός χειριστή, να εγκρίνει την αίτησή του (ή να απενεργοποιήσει τον λογαριασμό του), όπως και να τον διαγράψει εντελώς.
11. **Queries:** Η σελίδα όπου ο κάθε χρήστης μπορεί να δει τα ερωτήματά του.
12. Books: Οι χρήστες (εκτός του διαχειριστή), μπορούν να δουν τα βιβλία του σχολείου τους, και να επιλέξουν κάποιο συγκεκριμένο για περισσότερες πληροφορίες. Αν είναι χειριστές τότε έχουν τη δυνατότητα να προσθέσουν νέο.
13. Add Book: Οι χειριστές καταχωρούν τα στοιχεία νέου βιβλίου. Μπορούν να προσθέσουν συγγραφείς και κατηγορίες, ή να επιλέξουν από τις προϋπάρχουσες στη βάση.
14. Book: Φαίνονται τα στοιχεία ενός βιβλίου, και υπαρχει δυνατότητα προβολής των αξιολογήσεών του. Για χειριστές υπάρχει η δυνατότητα πρόσθεσης αντιτύπου, τροποποίησης, διαγραφής αλλά και επιλογή για δανεισμό σε κάποιο χρήστη, ενώ για τους μαθητές και τους εκπαιδευτικούς υπάρχει επιπλέον μόνο η δυνατότητα κράτησης.
15. Reviews of Book: Οι χρήστες (εκτός του διαχειριστή) μπορούν να δουν τις αξιολογήσεις ενός βιβλίου (αν είναι μαθητές ή εκπαιδευτικοί έχουν τη δυνατότητα να δημιουργήσουν νέα).
16. Make Review: Οι καθηγητές και οι μαθητές μπορουν να κάνουν αξιολόγηση του βιβλίου.
17. Edit Book: Οι χειριστές μπορούν να τροποποιήσουν τα στοιχεία ενός βιβλίου.
18. Rent Book: Οι χειριστές μπορούν να καταχωρίσουν δανεισμό και χωρίς κράτηση.
19. Users: Οι χειριστές μπορούν να δουν τους χρήστες του σχολείου (εγκεκριμένους και μη).
20. User: Οι χειριστές βλέπουν τα στοιχεία του χρήστη. Μπορούν να εγκρίνουν την αίτησή του, να απενεργοποιήσουν τον λογαριασμό του, να τον διαγράψουν, ή να τυπώσουν την καρτέλα του.
21. Reservations: Οι κρατήσεις του σχολείου, ή του χρήστη, με δυνατότητα επιλογής.
22. Reservation: Στοιχεία συγκεκριμένης κράτησης, με δυνατότητα διαγραφής ή δανεισμού (για χειριστές).
23. Rentals: Οι δανεισμοί του σχολείου, ή του χρήστη (χωρισμένοι σε αυτούς που καθυστερήσαν, αυτούς που είναι σε ισχύ, και οι παλιοί).
24. Rental: Στοιχεία ενός δανεισμού, με δυνατότητα επιστροφής για χειριστές.
25. Reviews: Ο χειριστής βλέπει τις αξιολογήσεις των μαθητών που δεν έχουν ακόμη εγκριθεί.
26. Review: Συγκεκριμένη αξιολόγηση μαθητή, με δυνατότητα διαγραφής ή έγκρισης.

Για να τρέξει κανείς την εφαρμογή πρέπει να ακολουθήσει τα βήματα που αναφέρονται και στο αρχείο README.md του git repository:

Required Tools

The project runs with Node.js and MySQL Database server

Before Starting the App

From directory `/databases` you need to run the script `schema.sql` to initialize the database. The script creates the tables, and inserts the data of the Top Operator in table 'users':

```
user_id: 1
username: topoperator
password: password
user_fullname: TOP OPERATOR
date_of_birth: 2000-01-01
approved: true
```

This data can be edited upon the first log in of the top operator.

Required Dependencies

To install all required dependencies (and build the project), run:
`npm run installAll`

Start

To launch the application run:
`npm start`

Sample Data

If you want, you can run the script `dbdata.sql` found in `/database` directory, to fill the database with sample data.

5. **Git Repository:** https://github.com/ntua-el19709/ntua_databases_project