Πανεπιστήμιο Θεσσαλίας
Τμήμα Ηλεκτρολόγων Μηχανικών και Μηχανικών Υπολογιστών

Προεραιτική εργασία ΣΒΔ**1**

Ονοματεπώνυμο: Ραφτόπουλος Εμμανουήλ

Μάθημα: Συστήματα Βάσεων Δεδομένων Ι – ΑΕΜ: *03735*
Ημερομηνία: *19/01/2025*

Contents

1. Θέμα της Βάσης

Ως εφαρμογή της βάσης επέλεξα ένα μον τέλο Βιντεοκλάμπ με όνομα Eight Spots, δηλαδή μια αλυσίδα καταστημάτων η οποία θα έχει διάφορα καταστήματα, εγγεγραμένους χρήστες και πολλές διαθέσιμες ταινίες τις οποίες θα μπορούν να αγοράσουν, να σχολιάσουν, ακόμα και να δηλώσουν αν έχουν καταφέρει ή όχι να τις δούνε οι χρήστες/πελάτες της αλυσίδας. Η υλοποίηση της βάσης έγινε με τη χρήση PostgreSQL, ενώ η λειτουργία της έγινε με τη σύνδεση της με ένα web application γραμμένο σε NodeJS και συγκεκριμένα με το framework ExpressJS. Όλες οι λειτουργίες που περιγράφονται από το ERD και τα Schemes είναι πλήρως λειτουργικές και θα μπορείτε να τις δοκιμάσετε μέσα από την ιστοσελίδα. Τέλος, η σύνδεση μεταξύ της βάσης δεδομένων και του web app επιτεύχθηκε με την χρήση docker και για να τρέξετε τοπικά την εφαρμογή αρκεί να έχετε εγκατεστημένο μόνο το docker και να εκτελέσετε το shell script build_docker.sh

Figure 1: Logo της αλυσίδας

2. Διάγραμμα Οντοτήτων-Συσχετίσεων

Οι οντότητες αλλά και οι συσχετίσεις που προκύπτουν από τις λειτουργίες της εφαρμογής, καθώς και επιπλέον χαρακτηριστικά της βάσης που δεν φαίνονται στην περιγραφή αλλά είναι καίρια για το web app καλύπτονται πλήρως από το παρακάτω διάγραμμα, το οποίο σχεδιάστηκε με τη βοήθεια του ERD Tools το οποίο χρησιμοποιήσαμε εκτενώς στο εργαστήριο του μαθήματος.

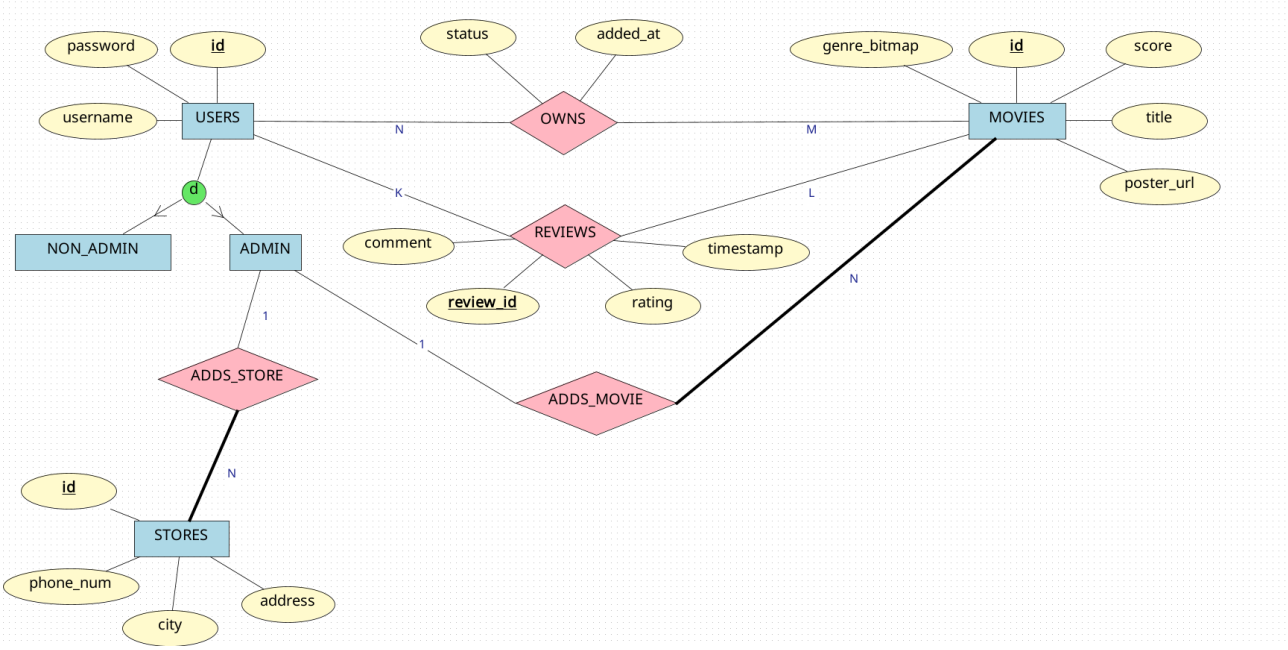

Figure 2: ERD

3. Σχεσιακό Σχήμα

Από το παραπάνω διάγραμμα μπορούμε σχετικά γρήγορα να δημιουργήσουμε και το σχεσιακό σχήμα:

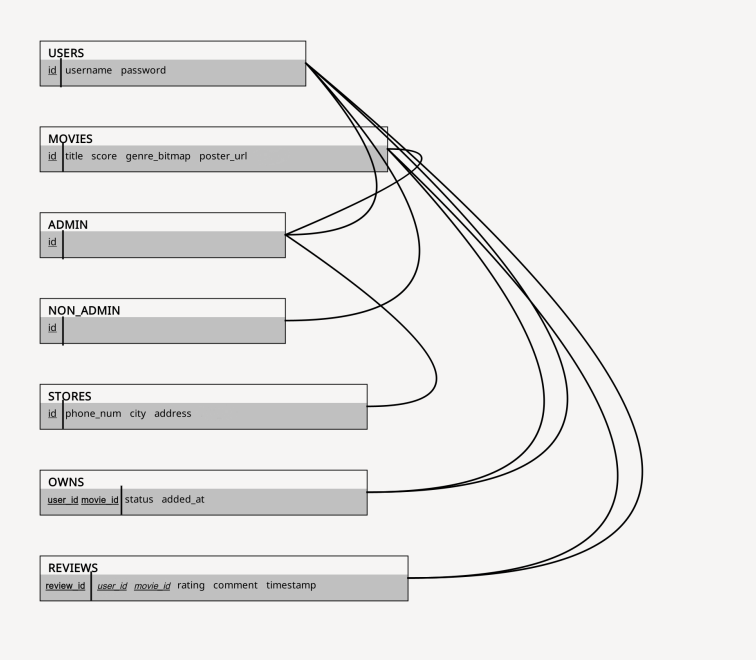

Figure 3: Relational Schema

Τα προτεύοντα κλειδία είναι αριστερά της μαύρης γραμμής με σκούρο χρώμα και υπογράμμιση, ενώ τα ξένα είναι με ιταλική γραμματοσειρά και υπογράμμιση. Παρατηρούμε, επίσης ότι στον πίνακα OWNS έχουμε ένα σύνθετο προτεύον κλειδί.
Περιορισμοί ακεραιότητας της βάσεις είναι οι εξής:

- Δεν μπορεί σε καμία στήλης, καμίας γραμμής να υπάρχει NULL εγγραφή
- Το score κάθε ταινίας πρέπει να είναι πραγματικός αριθμός x : $0 \leq x \leq 10$
- Το rating κάθε ταινίας στα review πρέπει να είναι ακέραιος αριθμός x : $0 \leq x \leq 5$
- Το genre_bitmap κάθε ταινίας θα πρέπει να είναι > 0, να ανήκει δηλαδή μία τουλάχιστον κατηγορία ταινιών
- Το poster_url θα πρέπει να αντιστοιχεί σε τοπικό μονοπάτι του server όπου περιέχει κάποια εικόνα

4. Συναρτησιακές Εξαρτήσεις και Κανονικοποίηση

Παρακάτω αναλύονται οι συναρτησιακές εξαρτήσεις για κάθε πίνακα της βάσης δεδομένων και εξετάζεται η κανονική μορφή στην οποία βρίσκονται:

4.1. Πίνακας **users.** Συναρτησιακές εξαρτήσεις:

- id → username, password

Ο πίνακας είναι σε BCNF καθώς το id είναι το primary key και καθορίζει μοναδικά όλα τα υπόλοιπα γνωρίσματα.

4.2. Πίνακας **movies.** Συναρτησιακές εξαρτήσεις:

- id → title, score, price, genre_bitmap, poster_url

Ο πίνακας είναι σε BCNF καθώς το id είναι το primary key και καθορίζει μοναδικά όλα τα υπόλοιπα γνωρίσματα.

4.3. Πίνακας **stores.** Συναρτησιακές εξαρτήσεις:

- id → phone_num, city, address

Ο πίνακας είναι σε BCNF καθώς το id είναι το primary key και καθορίζει μοναδικά όλα τα υπόλοιπα γνωρίσματα.

4.4. Πίνακας **user_library.** Συναρτησιακές εξαρτήσεις:

- (user_id, movie_id) → status, added_at

Ο πίνακας είναι σε BCNF καθώς έχει σύνθετο primary key (user_id, movie_id) που καθορίζει τα υπόλοιπα γνωρίσματα.

4.5. Πίνακας **reviews.** Συναρτησιακές εξαρτήσεις:

- review_id → user_id, movie_id, rating, comment, timestamp
- (user_id, movie_id) → review_id, rating, comment, timestamp

Ο πίνακας είναι σε BCNF καθώς το review_id είναι το primary key και καθορίζει μοναδικά όλα τα υπόλοιπα γνωρίσματα.

4.6. Συμπέρασμα**.** Η βάση δεδομένων είναι ήδη κανονικοποιημένη σε BCNF (Boyce-Codd Normal Form), η οποία είναι ισχυρότερη από την 3NF, για τους εξής λόγους:

1. Κάθε πίνακας έχει ένα ξεκάθαρο primary key (είτε απλό είτε σύνθετο)
2. Όλες οι συναρτησιακές εξαρτήσεις προέρχονται από το primary key
3. Δεν υπάρχουν μεταβατικές εξαρτήσεις
4. Δεν υπάρχουν μερικές εξαρτήσεις (όλα τα non-key attributes εξαρτώνται πλήρως από το primary key)
5. Οι σχέσεις μεταξύ των πινάκων υλοποιούνται με foreign keys
6. Δεν υπάρχει πλεονασμός δεδομένων

Η χρήση των views (user_movie_library και movie_reviews) επίσης δεν επηρεάζει την κανονικοποίηση καθώς είναι απλά εικονικοί πίνακες που συνδυάζουν δεδομένα από τους ήδη κανονικοποιημένους πίνακες.

5. Όψεις και Ευρετήρια

Για να κάνουμε πιο αποδοτική τη βάση κατά την αναζήτηση περιεχομένων έγινε χρήση εικονικών όψεων και ευρετηρίων. Συγκεκριμένα, όπου υπήρχε ανάγκη για JOIN 2 πινάκων φτιάξαμε μία όψη η οποία θα περιέχει μόνο τα γνωρίσματα που χρειαζόμαστε για την επιστροφή. Δυό τέτοιες όψεις στην βάση μας είναι οι εξής:

```
CREATE VIEW user_movie_library AS
SELECT ul.user_id, m.*, ul.status
FROM movies m
JOIN user_library ul ON m.id = ul.movie_id;

CREATE VIEW movie_reviews AS
SELECT r.movie_id, r.review_id, r.rating, r.comment, r.timestamp, u.username
FROM reviews r
JOIN users u ON r.user_id = u.id;
```

Η πρώτη όψη αφορά την βιβλιοθήκη ταινιών του κάθε χρήστη, ενώ η δεύτερη αφορά τις κριτικές κάθε ταινίας από τους χρήστες της εφαρμογής. Όσον αφορά τα ευρετήρια παρατηρήσαμε ότι 2 κλειδιά είναι αυτά που χρησιμοποιούνται πολύ συχνά από την εφαρμογή μας και για αυτό αποφασίσαμε ότι αξίζει για την καλύτερη απόδοσή της να τα εισάγουμε ως indexes:

```
CREATE INDEX idx_user_library_user_movie ON user_library ( user_id , movie_id ) ;
CREATE INDEX idx_reviews_movie_id ON reviews ( movie_id ) ;
```

6. Δημιουργία Βάσης

```sql
CREATE SEQUENCE users_id_seq;
CREATE SEQUENCE movies_id_seq;
CREATE SEQUENCE reviews_id_seq;
CREATE SEQUENCE stores_id_seq;

CREATE TABLE users (
    id INT,
    username TEXT,
    password TEXT,
    PRIMARY KEY(id)
);
CREATE TABLE movies (
    id INT,
    title TEXT,
    score REAL,
    price REAL,
    genre_bitmap INT DEFAULT 0,
    poster_url TEXT,
    PRIMARY KEY(id)
);
CREATE TABLE stores (
    id INT,
    phone_num TEXT,
    city TEXT,
    address TEXT,
    PRIMARY KEY(id)
);

CREATE TABLE user_library (
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    status BOOLEAN, -- 'watched', 'unwatched'
    added_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (user_id, movie_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (movie_id) REFERENCES movies(id)
);
CREATE TABLE reviews (
    review_id INT,
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    rating INT,
    comment TEXT,
    timestamp TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    PRIMARY KEY (review_id),
    FOREIGN KEY (user_id) REFERENCES users(id),
    FOREIGN KEY (movie_id) REFERENCES movies(id)
);

```

```
-- Create indexes
CREATE INDEX idx_user_library_user_movie ON user_library (user_id, movie_id);
CREATE INDEX idx_reviews_movie_id ON reviews (movie_id);

-- Create views
CREATE VIEW user_movie_library AS
SELECT ul.user_id, m.*, ul.status
FROM movies m
JOIN user_library ul ON m.id = ul.movie_id;

CREATE VIEW movie_reviews AS
SELECT r.movie_id, r.review_id, r.rating, r.comment, r.timestamp, u.username
FROM reviews r
JOIN users u ON r.user_id = u.id;
```

Για την κατασκευή της βάσης σε PostgreSQL, χρησιμοποιούμε τα προηγούμενα διαγράμματα και συμπεράσματα με συμπέρασμα η βάση να είναι απλή και κατανοητή. Το μόνο κομμάτι που βλέπουμε για πρώτη φορά είναι τα SEQUENCE τα οποία πρακτικά χρησιμοποιούνται για να μην καταχωρούμε χειροκίνητα κάθε φορά το ID κάθε εγγραφής, αλλά να γίνεται αυτόματα η παραγωγή του.

7. Ερωτήματα/**Queries** της **SQL** από την εφαρμογή

Τα ερωτήματα που χρησιμοποιεί η εφαρμογή για να εκπληρώσει όλες τις λειτουργίες που προαναφέρθηκαν είναι τα εξής:

```
1) SELECT * FROM users WHERE username = $1
2) SELECT * FROM movies
        WHERE (genre_bitmap & $1) > 0
        ORDER BY score DESC
        LIMIT $2
3) SELECT * FROM stores
4) INSERT INTO movies (id, title, score, price, genre_bitmap, poster_url) VALUES (
   nextval(\'movies_id_seq\'), $1, $2, $3, $4, $5)
5) INSERT INTO stores (id, phone_num, city, address) VALUES (nextval(\'stores_id_seq
   \'), $1, $2, $3)
6) SELECT * FROM user_movie_library WHERE user_id = $1
7) SELECT status FROM user_library WHERE user_id = $1 AND movie_id = $2
8) UPDATE user_library SET status = $1 WHERE user_id = $2 AND movie_id = $3
9) SELECT * FROM movies ORDER BY id ASC
10) SELECT * FROM movies WHERE id = $1
11) SELECT * FROM movie_reviews WHERE movie_id = $1 ORDER BY timestamp DESC
12) INSERT INTO user_library (user_id, movie_id, status) VALUES ($1, $2, $3)
13) INSERT INTO reviews (review_id ,user_id, movie_id, rating, comment) VALUES (
   nextval(\'reviews_id_seq\'), $1, $2, $3, $4)
14) INSERT INTO users (id, username, password) VALUES (nextval(\'users_id_seq\'), $1
   , $2)
15) UPDATE users SET username = $1 WHERE id = $2
```

Ας ερευνήσουμε σύντομα τι κάνει το κάθε ένα από αυτά:

1. Επιλέγει όλες τις στήλες από τον πίνακα **users** όπου το **username** είναι ίδιο με το πρώτο **arguement** που του δίνουμε
2. Επιλέγει όλες τις στήλες από τον πίνακα **movies** όπου η **boolean** πράξη **&** του **bit** που έχουμε ορίσει για την εκάστοτε κατηγορία ταινιών με το **bitmap** της ταινίας είναι > 0, έπειτα τις κάνει ταξινόμιση με φθείνουσα σειρά ανάλογα το **score** και κρατάει τις πρώτες N, όπου N το δεύτερο **arguement**
3. Επιλέγει όλες τις στήλες από τον πίνακα **stores**
4. Προσθέτει καινούργια εγγραφή στον πίνακα **movies** με χρήση **nextval(seq)** για να μπει αυτόματα το **id**
5. Προσθέτει καινούργια εγγραφή στον πίνακα **stores** με χρήση **nextval(seq)** για να μπει αυτόματα το **id**
6. Χρησιμοποιεί την όψη που φτιάξαμε νωρίτερα και επιστρέφει όλα τα στοιχεία της ταινίας, αλλά και το **watched status** της από τον συνδεδεμένο χρήστη
7. Επιλέγει το **status** από τον πίνακα **user_library** για συγκεκριμένο χρήστη (πρώτο όρισμα) και ταινία (δεύτερο όρισμα)
8. Ανανεώνει το **status** στον πίνακα **user_library** με βάση το πρώτο όρισμα για συγκεκριμένο χρήστη (δεύτερο όρισμα) και ταινία (τρίτο όρισμα)
9. Επιλέγει όλες τις στήλες από τον πίνακα **movies** και τις ταξινομεί με φθείνουσα σειρά ανάλογα με το **id** τους

10. Επιλέγει όλες τις στήλες από τον πίνακα **movies** με ίδιο **id** όπως και το πρώτο όρισμα
11. Χρησιμποιεί την όψη που φτιάξαμε νωρίτερα και επιστρέφει 4 στήλες του πίνακα **reviews** (**review_id, rating, comment, timestamp**) καθώς και τη στήλη **username** του πίνακα **users** και έπειτα τις ταξινομεί με φθείνουσα σειρά με βάση το **timestamp**
12. Προσθέτει καινούργια εγγραφή στον πίνακα **user_library**
13. Προσθέτει καινούργια εγγραφή στον πίνακα **reviews** με χρήση **nextval(seq)** για να μπει αυτόματα το **id**
14. Προσθέτει καινούργια εγγραφή στον πίνακα **users** με χρήση **nextval(seq)** για να μπει αυτόματα το **id**
15. Ανανεώνει το **username** στον πίνακα **users** για τον συνδεδεμένο χρήστη ανάλογα με το πρώτο όρισμα

8. Τοπολογία αρχείων

- `app/`: Περιέχει τον κώδικα της εφαρμογής και τα scripts κατασκευής.
  - `build_docker.sh`: Script για την κατασκευή και εκτέλεση των Docker containers.
  - `docker-compose.yml`: Αρχείο ρύθμισης για το Docker Compose.
  - `Dockerfile`: Οδηγίες για την κατασκευή του Docker image.
  - `.env`: Μερικές περιβαλλοντικές μεταβλητές που πρέπει να παραμείνουν μυστικές.
  - `package[-lock].json`: Απαραίτητο αρχείο yaml για το nodejs και τις εξαρτήσεις του.
  - `src/`: Περιέχει τον πηγαίο κώδικα της εφαρμογής.
    - `index.js`: Κύριο σημείο εισόδου της εφαρμογής.
    - `assets/`: Κατάλογος για στατικά αρχεία (π.χ., εικόνες).
    - `styles/`: Κατάλογος για CSS styles.
    - `uploads/posters/`: Κατάλογος για τα uploads των αφισών των ταινιών.
  - `initdb/`: Περιέχει τον κώδικα SQL για την αρχικοποίηση της βάσης δεδομένων.
    - `db.sql`: SQL script για τη δημιουργία του σχήματος της βάσης δεδομένων.
    - `initdb.sql`: SQL script για την εισαγωγή αρχικών δεδομένων στη βάση.
- `report.pdf`: Η αναφορά του project.
- `schema.png`: Αρχείο εικόνας που αναπαριστά το σχήμα της βάσης δεδομένων.
- `db.erdt`: Το ERD που δημιουργήθηκε με χρήση του ERD TOOLS.
- `README.md`: Ένα README που εξηγεί την χρήση της εφαρμογής.

9. Οδηγός Χρήσης

Αυτό το έγγραφο παρέχει έναν αναλυτικό οδηγό για την εγκατάσταση και χρήση της εφαρμογής που περιλαμβάνεται στο αρχείο ZIP. Η εφαρμογή είναι κατασκευασμένη με χρήση Node.js και PostgreSQL, και είναι containerized με χρήση Docker.

9.1. Προαπαιτούμενα**.** Πριν ξεκινήσετε, βεβαιωθείτε ότι έχετε εγκαταστήσει τα ακόλουθα στον υπολογιστή σας:

- **Docker**: Κατεβάστε και εγκαταστήστε την τελευταία έκδοση του Docker από `https://www.docker.com/get-started`.
- **Docker-Compose (**προαιρετικό αφού η τελευταία έκδοση **docker** έχει δικό της **compose)**: Αυτό συμπεριλαμβάνεται στις εγκαταστάσεις του Docker Desktop.

9.2. Οδηγίες Εγκατάστασης**.**

1. Λήψη του Αρχείου **ZIP**: Κατεβάστε το αρχείο ZIP που περιέχει την εφαρμογή και αποσυμπιέστε το στην επιθυμητή τοποθεσία.
2. Δομή Καταλόγων: Ο φάκελος που θα προκύψει θα πρέπει να περιέχει την ακόλουθη δομή:
   - `app/`: Περιέχει τον κώδικα της εφαρμογής και τα scripts κατασκευής.
     - `build_docker.sh`: Script για την κατασκευή και εκτέλεση των Docker containers.
     - `docker-compose.yml`: Αρχείο ρύθμισης για το Docker Compose.
     - `Dockerfile`: Οδηγίες για την κατασκευή του Docker image.
     - `.env`: Μερικές περιβαλλοντικές μεταβλητές που πρέπει να παραμείνουν μυστικές.
     - `package[-lock].json`: Απαραίτητο αρχείο yaml για το nodejs και τις εξαρτήσεις του.
     - `src/`: Περιέχει τον πηγαίο κώδικα της εφαρμογής.
       - `index.js`: Κύριο σημείο εισόδου της εφαρμογής.
       - `assets/`: Κατάλογος για στατικά αρχεία (π.χ., εικόνες).
       - `styles/`: Κατάλογος για CSS styles.
       - `uploads/posters/`: Κατάλογος για τα uploads των αφισών των ταινιών.
     - `initdb/`: Περιέχει τον κώδικα SQL για την αρχικοποίηση της βάσης δεδομένων.
       - `db.sql`: SQL script για τη δημιουργία του σχήματος της βάσης δεδομένων.
       - `initdb.sql`: SQL script για την εισαγωγή αρχικών δεδομένων στη βάση.
   - `report.pdf`: Η αναφορά του project.
   - `schema.png`: Αρχείο εικόνας που αναπαριστά το σχήμα της βάσης δεδομένων.
   - `db.erdt`: Το ERD που δημιουργήθηκε με χρήση του ERD TOOLS.
   - `README.md`: Ένα README που εξηγεί την χρήση της εφαρμογής.
3. Μετάβαση στον Κατάλογο του **Project**: Ανοίξτε ένα τερματικό (Command Prompt, PowerShell, ή Terminal) και μεταβείτε στον κατάλογο του project:

   ```
   cd hw1/app
   ```

9.3. Εκτέλεση της Εφαρμογής. Για να εκτελέσετε την εφαρμογή, ακολουθήστε τα παρακάτω βήματα:

1. Κατασκευή και Εκκίνηση των **Docker Containers**: Εκτελέστε το script κατασκευής για να δημιουργήσετε τα Docker images και να ξεκινήσετε τα containers:

   ```
   ./build_docker.sh
   ```

   Αυτό το script θα κατασκευάσει τα Docker images και θα ξεκινήσει τα containers της PostgreSQL και της web εφαρμογής.

2. Τερματισμός της Εφαρμογής: Για να σταματήσετε την εφαρμογή, πατήστε 'Ctrl+C' στο τερματικό όπου εκτελείται το script. Αυτό θα τερματίσει ομαλά τα Docker containers.

9.4. Πρόσβαση στην Εφαρμογή. Μόλις η εφαρμογή εκκινηθεί μετά από 1 λεπτό περίπου για να γίνουν οι αρχικοποιήσεις της βάσης, μπορείτε να την ανοίξετε στον web browser σας:

- Ανοίξτε τον web browser σας και πηγαίνετε στο `http://localhost:3000`.
- Μπορείτε να συνδεθείτε είτε ως admin με κωδικό admin είτε ως απλός user με κωδικό user

9.5. Αρχικοποίηση Βάσης Δεδομένων. Η εφαρμογή χρησιμοποιεί μια βάση δεδομένων PostgreSQL. Τα scripts αρχικοποίησης περιλαμβάνονται στη ρύθμιση Docker, και η βάση δεδομένων θα δημιουργηθεί αυτόματα με τους απαραίτητους πίνακες και αρχικά δεδομένα όταν εκτελέσετε την εφαρμογή για πρώτη φορά.

9.6. Ασφαλής Τερματισμός Εφαρμογής. Αφού έχετε χρησιμοποιήσει την εφαρμογή και θέλετε να κλείσετε τα docker αρκεί μόνο να πάτε πάλι στον φάκελο /app και να εκτελέσετε την εντολή:

```
docker compose down
```

ή αν θέλετε να διαγράψετε τα δεδομένα που προσθέσατε/αλλάξατε στη βάση

```
docker compose down -v
```

9.7. Αντιμετώπιση Προβλημάτων. Αν αντιμετωπίσετε προβλήματα, εξετάστε τα ακόλουθα:

- Βεβαιωθείτε ότι το Docker εκτελείται στον υπολογιστή σας στην τελευταία έκδοση.
- Ελέγξτε την έξοδο του τερματικού για τυχόν μηνύματα σφάλματος κατά τη διάρκεια της κατασκευής ή εκκίνησης.
- Επιβεβαιώστε ότι το αρχείο '.env' περιέχει τις σωστές περιβαλλοντικές μεταβλητές για τη ρύθμισή σας.
- Δοκιμάστε να σταματήσετε το compose και να επανεκτελέσετε το script κατασκευής ενώ βρίσκεστε στον κατάλογο /app χρησιμοποιώντας:

  ```
  docker compose down -v
  ./build_docker.sh
  ```

- Συμβουλευτείτε την αναφορά του project (`report.pdf`) για πρόσθετες πληροφορίες και συμβουλές αντιμετώπισης προβλημάτων.

9.8. Συμπέρασμα. Αυτός ο οδηγός παρέχει τα απαραίτητα βήματα για την εγκατάσταση και εκτέλεση της εφαρμογής. Αν έχετε περαιτέρω ερωτήσεις ή χρειάζεστε βοήθεια, μη διστάσετε να επικοινωνήσετε.

10. Μελλοντικές Προσθήκες

Το web app που υλοποίησα έχει σχεδόν όλες τις λειτουργίες που χρειάζεται ένας χρήστης σε μία εφαρμογή για να αγοράζει ταινίες. Παρ' όλα αυτά θα μπορούσε να περιέχει αρκετά ακόμα features για να συμβαδίζει με τις σύγχρονες τεχνολογίες. Κάποια από αυτά είναι:

- Εμφάνιση των δημοφιλέστερων ταινιών
- Προτάσεις ταινιών στους χρήστες βασισμένες σε ένα απλό σχετικά αλγόριθμο με την χρήση του genre_bitmap της κάθε ταινίας που έχει αγοράσει/παρακολουθήσει
- Έλεγχος υπολοίπου του χρήστη για να μπορεί να αγοράζει μόνο ταινίες εντός budget
- Υλοποίηση για τον admin να μπορεί να βάζει εύκολα προσφορές στις ταινίες
- Διαθεσιμότητα ταινιών στα φυσικά καταστήματα. Θα απαιτούσε επιπλέον N:M σχέση μεταξύ καταστήματος και ταινιών
- Σύστημα φίλων, να μπορεί δηλαδή ένας χρήστης να γίνεται φίλος με άλλους και να βλέπει τις λίστα ταινιών τους
- Καταγραφή στατιστικών πώλησης ταινιών για τον admin
- Πολυγλωσσική υποστήριξη

11. Βιβλιογραφία

References

[1] PostgreSQL. *PostgreSQL: The World's Most Advanced Open Source Relational Database*. Available at: [PostgreSQL Official Website] and [PostgreSQL Documentation].

[2] Docker. *Docker: Empowering App Development*. Available at: [Docker Official Website] and [Docker Get Started Tutorial].

[3] Express.js. *Express - Node.js web application framework*. Available at: [Express.js Official Website] and [Express.js Installation Guide].

[4] Node.js. *Node.js: JavaScript runtime built on Chrome's V8 JavaScript engine*. Available at: [Node.js Official Website] and [Node.js Tutorials].