

Veritabanı Yönetimi Projesi

KONU: Film Gecesi Organizasyon ve Oy Kullanma Sistemi
(MovieNight)

AD:ALPARSLAN KOÇ

NUMARA:21011057

AD:OZAN DANIŞ

NUMARA:21011040

AD:ŞAHİN DOĞRUCA

NUMARA:22011049

DERSİN EĞİTMENİ:M.UTKU KALAY

Projenin Amacı

Film Etkinlik Yönetim Sistemi, kullanıcıların film odaklı etkinlikler düzenlemesini, filmleri değerlendirmesini ve önerilerde bulunmasını sağlayan bir platform sunmayı amaçlar. Bu sistem, kullanıcılar arasında sosyal etkileşimi artırmak ve filmlerle ilgili deneyimlerini paylaşmalarını teşvik eder.

Sistemin Temel Özellikleri

1. Kullanıcı Yönetimi

- Kullanıcılar sisteme kayıt olabilir ve giriş yapabilir.
- Kullanıcı bilgileri güvenli bir şekilde saklanır.
- Kullanıcılar kendi profillerini yönetebilir.

2. Film Yönetimi

- Kullanıcılar sisteme film ekleyebilir, güncelleyebilir veya silebilir.
- Her film, türü, vizyon tarihi ve puan bilgisiyle birlikte kaydedilir.
- Filmlerin ortalama puanı, kullanıcıların verdiği oylara göre dinamik olarak hesaplanır.

3. Etkinlik Yönetimi

- Kullanıcılar etkinlikler oluşturabilir ve diğer kullanıcıları davet edebilir.
- Her etkinlik için bir isim, tarih ve detaylar eklenebilir.
- Etkinlik katılımcıları listelenebilir ve yönetilebilir.

4. Davet Yönetimi

- Kullanıcılar, diğer kullanıcıları etkinliklere davet edebilir.
- Davetler kabul edilebilir, reddedilebilir veya beklemede bırakılabilir.

5. Film Oylama

- Kullanıcılar filmleri 1 ile 10 arasında puanlayabilir.
- Aynı kullanıcı bir filme sadece bir kez oy verebilir.
- Oy verme işlemleri, filmin ortalama puanını otomatik olarak günceller.

6. Film Önerisi

- Kullanıcılar, etkinlikler için film önerilerinde bulunabilir.
- Öneriler etkinlik ve kullanıcı bazında listelenebilir.

Teknolojik Alt Yapı

1. Arka Uç (Backend)

- Java kullanılarak sistemin tüm iş mantığı tasarlandı.
- DAO (Data Access Object) tasarım deseni kullanılarak veritabanı işlemleri soyutlandı.
- Verimli ve güvenli sorgular için PreparedStatement kullanıldı.

2. Veritabanı

- PostgreSQL kullanılarak ilişkisel bir veritabanı tasarlandı.
- Sistemde şu tablolar yer alır:
 - admin: Yöneticilerin bilgilerini saklar.
 - users: Kullanıcı bilgilerini saklar.
 - movies: Filmlerin bilgilerini saklar.
 - votes: Kullanıcıların verdiği oyları saklar.
 - events: Film etkinliklerini saklar.
 - invitation: Etkinlik davetlerini saklar.
 - suggestions: Etkinlikler için önerilen filmleri saklar.

3. Ön Yüz (Frontend)

- **Swing** kütüphanesi kullanılarak masaüstü uygulama tasarlandı.
- **IntelliJ IDEA** üzerinde geliştirildi.
- Kullanıcı dostu arayüz:
 - Film ekleme, oylama ve öneri işlemleri kolayca yapılabilir.
 - Etkinlik oluşturma ve davet yönetimi basit bir şekilde gerçekleştirilebilir.

Sistemin İşleyişı

1. Kayıt ve Giriş

- Yeni kullanıcılar kayıt formunu doldurarak sisteme üye olur.
- Daha önce kayıt olan kullanıcılar giriş yaparak işlemlerine devam edebilir.

2. Film Yönetimi

- Kullanıcılar yeni filmler ekleyebilir, mevcut filmleri güncelleyebilir veya silebilir.
- Eklenen filmler listelenir ve detaylı bir şekilde görüntülenebilir.

3. Etkinlik Yönetimi

- Kullanıcılar bir etkinlik oluşturur ve diğer kullanıcıları davet edebilir.
- Her etkinlik için film önerileri yapılabilir.

4. Oy Verme

- Kullanıcılar filmleri puanlayarak değerlendirme yapabilir.
- Verilen oylar, filmin ortalama puanını anında günceller.

5. Film Önerisi

- Kullanıcılar etkinlikler için film önerilerinde bulunabilir.
- Film önerileri etkinlik bazında listelenebilir.

Swing Kullanılarak Oluşturulan Arayüz

Arayüz Özellikleri

- **Ana Menü:**
 - Film ekleme, güncelleme ve silme işlemleri için seçenekler.
 - Etkinlik oluşturma ve davet yönetimi işlemleri.
- **Formlar:**
 - Kullanıcı kayıt ve giriş formu.
 - Film ekleme/güncelleme formu.
 - Oy verme ekranı.
- **Listeler:**
 - Tüm filmler, etkinlikler ve davetler liste görünümünde görüntülenebilir.
 - Kullanıcı dostu butonlar ve alanlarla işlemler basitleştirildi.

Projenin Avantajları

- **Genişletilebilirlik:** Modüler tasarımı sayesinde yeni özellikler kolayca eklenebilir.
- **Kullanıcı Dostu Arayüz:** Swing ile oluşturulan masaüstü uygulama, kullanıcıların rahatça işlem yapmasını sağlar.
- **Sosyal Etkileşim:** Kullanıcılar arasında etkinlikler ve öneriler yoluyla güçlü bir bağ kurulur.
- **Dinamik Güncellemeler:** Oy verme işlemleri gibi kritik noktalar, ilgili verileri anında günceller.

Projenin Gelecekteki Geliştirme Alanları

1. Web ve Mobil Uygulama Desteği

- Swing tabanlı masaüstü uygulamasına ek olarak, web ve mobil sürümler geliştirilebilir.

2. Yapay Zekâ Destekli Öneriler

- Kullanıcı beğenilerine göre öneri yapabilecek bir yapay zekâ sistemi entegre edilebilir.

3. Bildirim Sistemi

- E-posta veya anlık bildirimlerle kullanıcılar bilgilendirilebilir.

4. Analitik ve Raporlama

- Kullanıcıların etkinlik ve film oylama alışkanlıklarına dair istatistikler sunulabilir.

Sonuç

Film Etkinlik Yönetim Sistemi, kullanıcıların film odaklı sosyal bir ortamda bir araya gelmesini sağlayan, işlevsel ve genişletilebilir bir platformdur. Swing tabanlı masaüstü arayüzü ve veritabanı altyapısı ile bu sistem, kullanıcıların ihtiyaçlarına uygun bir deneyim sunar.

Veritabanı Tabloları

Invitations Tablosu

	invitation_id [PK] integer	event_id integer	user_id integer	status character varying (20)
1	1	1	6	PENDING
2	2	2	7	PENDING
3	3	1	8	PENDING
4	4	1	9	PENDING
5	5	1	2	PENDING
6	6	2	3	PENDING
7	7	2	4	PENDING
8	8	3	1	PENDING
9	9	4	1	PENDING
10	10	5	5	PENDING

Events Tablosu

	event_id [PK] integer	user_id integer	event_name character varying (100)	event_date date	created_at timestamp without time zone
1	1	1	Sinema Şöleni	2025-01-15	2025-01-10 19:53:28.792608
2	2	2	Film Rüzgarı	2025-01-16	2025-01-10 19:53:28.792608
3	3	3	Yıldızlı Gece	2025-01-17	2025-01-10 19:53:28.792608
4	4	4	Popcorn Partisi	2025-01-18	2025-01-10 19:53:28.792608
5	5	1	Perde Arkası Akşamı	2025-01-19	2025-01-10 19:53:28.792608
6	6	1	Film Tutkunları Buluşması	2025-01-20	2025-01-10 19:53:28.792608
7	7	2	Film ve Sohbet Gecesi	2025-01-21	2025-01-10 19:53:28.792608
8	8	2	Unutulmaz Filmler	2025-01-22	2025-01-10 19:53:28.792608
9	9	1	Son Perde	2025-01-23	2025-01-10 19:53:28.792608
10	10	1	Işıkların Altında	2025-01-24	2025-01-10 19:53:28.792608

Movies Tablosu

	movie_id [PK] integer	title character varying (100)	genre character varying (50)	release_date date	rating numeric (3,2)
1	1	Zor Ölüm 5	Aksiyon	2025-01-01	8.00
2	2	Piyanist	Drama	2025-01-02	7.00
3	3	Kolpaçino	Komedi	2025-01-03	6.00
4	4	Nefesini Tut	Gerilim	2025-01-04	0.00
5	5	Korku Seansı	Korku	2025-01-05	7.00
6	6	Kara Şövalye Yükseliyor	Bilim Kurgu	2025-01-06	7.80
7	7	Yukarı Bak	Animasyon	2025-01-07	6.90
8	8	Fetih 1453	Tarih	2025-01-08	7.10
9	9	Aşk Sarhoşu	Romance	2025-01-09	8.30
10	10	Hızlı ve Öfkeli 10	Adventure	2025-01-10	7.40






Suggestions Tablosu

	suggestion_id [PK] integer	event_id integer	user_id integer	movie_id integer	created_at timestamp without time zone
1	1	1	1	1	2025-01-10 19:53:28.792608
2	2	1	1	2	2025-01-10 19:53:28.792608
3	3	2	1	2	2025-01-10 19:53:28.792608
4	4	3	2	1	2025-01-10 19:53:28.792608
5	5	3	2	3	2025-01-10 19:53:28.792608
6	6	4	3	3	2025-01-10 19:53:28.792608
7	7	5	4	4	2025-01-10 19:53:28.792608
8	8	1	5	2	2025-01-10 19:53:28.792608
9	9	2	1	3	2025-01-10 19:53:28.792608
10	10	1	2	1	2025-01-10 19:53:28.792608





Users Tablosu

	user_id [PK] integer	username character varying (50)	email character varying (100)	password character varying (100)	created_at timestamp without time zone
1	1	alparslan	alparslan@gmail.com	alparslan123	2025-01-10 19:53:28.792608
2	2	ozan	ozan@gmail.com	ozan123	2025-01-10 19:53:28.792608
3	3	sahin	sahin@gmail.com	sahin123	2025-01-10 19:53:28.792608
4	4	ahmet	ahmet@gmail.com	ahmet123	2025-01-10 19:53:28.792608
5	5	mehmet	mehmet@gmail.com	mehmet123	2025-01-10 19:53:28.792608
6	6	ayse	ayse@gmail.com	ayse123	2025-01-10 19:53:28.792608
7	7	fatma	fatma@gmail.com	fatma123	2025-01-10 19:53:28.792608
8	8	hayriye	hayriye@gmail.com	hayriye123	2025-01-10 19:53:28.792608
9	9	ali	ali@gmail.com	ali123	2025-01-10 19:53:28.792608
10	10	recep	recep@gmail.com	recep123	2025-01-10 19:53:28.792608

Votes Tablosu

	vote_id [PK] integer 	movie_id integer 	user_id integer 	rating integer 	created_at timestamp without time zone 
1	1	1	3	8	2025-01-10 19:53:28.792608
2	2	2	4	7	2025-01-10 19:53:28.792608
3	3	3	5	6	2025-01-10 19:53:28.792608
4	4	1	6	9	2025-01-10 19:53:28.792608
5	5	1	7	8	2025-01-10 19:53:28.792608
6	6	1	8	7	2025-01-10 19:53:28.792608
7	7	2	9	6	2025-01-10 19:53:28.792608
8	8	2	10	7	2025-01-10 19:53:28.792608
9	9	2	1	8	2025-01-10 19:53:28.792608
10	10	5	1	7	2025-01-10 19:53:28.792608

Admin Tablosu

	admin_id [PK] integer 	name character varying (50) 	email character varying (100) 	password character varying (100) 
1	1	admin	admin@gmail.com	admin123

Kod Blokları

Primary Key ve Foreign Key

```
-- Suggestions table
CREATE TABLE if not exists suggestions (
    suggestion_id SERIAL PRIMARY KEY,
    event_id INT NOT NULL,
    user_id INT NOT NULL,
    movie_id INT NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT unique_suggestion UNIQUE (event_id, user_id, movie_id),
    FOREIGN KEY (event_id) REFERENCES events(event_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

```
-- Votes table
CREATE TABLE if not exists votes (
    vote_id SERIAL PRIMARY KEY,
    movie_id INT NOT NULL,
    user_id INT NOT NULL,
    rating INT NOT NULL CHECK (rating BETWEEN 1 AND 10),
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    CONSTRAINT unique_user_movie_vote UNIQUE (user_id, movie_id),
    FOREIGN KEY (movie_id) REFERENCES movies(movie_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

```
-- Movies table
CREATE TABLE if not exists movies (
    movie_id SERIAL PRIMARY KEY,
    title VARCHAR(100) NOT NULL,
    genre VARCHAR(50),
    release_date DATE,
    rating NUMERIC(3, 2) CHECK (rating >= 0 AND rating <= 10)
);
```

```
-- Invitations table
CREATE TABLE if not exists invitations (
    invitation_id SERIAL PRIMARY KEY,
    event_id INT NOT NULL,
    user_id INT NOT NULL,
    status VARCHAR(20) DEFAULT 'PENDING',
    constraint unique_invitation UNIQUE (event_id, user_id),
    FOREIGN KEY (event_id) REFERENCES events(event_id) ON DELETE CASCADE,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

```
-- Events table
CREATE TABLE if not exists events (
    event_id SERIAL PRIMARY KEY,
    user_id INT NOT NULL,
    event_name VARCHAR(100) NOT NULL,
    event_date DATE NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP,
    FOREIGN KEY (user_id) REFERENCES users(user_id) ON DELETE CASCADE
);
```

```
CREATE TABLE if not exists users (
    user_id INT PRIMARY KEY DEFAULT nextval('users_id_seq'),
    username VARCHAR(50) UNIQUE NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL,
    created_at TIMESTAMP DEFAULT CURRENT_TIMESTAMP
);
```

```
CREATE TABLE if not exists admin (
    admin_id INT PRIMARY KEY DEFAULT nextval('admin_id_seq'),
    name VARCHAR(50) NOT NULL,
    email VARCHAR(100) UNIQUE NOT NULL,
    password VARCHAR(100) NOT NULL
);
```

Insert Update Delete

```
public class AdminDAO { 2 usages

    public boolean addAdmin(Admin admin) { no usages
        if (isEmpty(admin.getName(), errorMessage: "Hata: Admin adı boş bırakılamaz.") ||
            isEmpty(admin.getEmail(), errorMessage: "Hata: Email adresi boş bırakılamaz.") ||
            !admin.getEmail().matches(regex: "[A-Za-z0-9+_.-]+@(.+)$")) {
            System.out.println("Hata: Geçersiz bir email adresi.");
            return false;
        }
        if (admin.getPassword() == null || admin.getPassword().length() < 6) {
            System.out.println("Hata: Şifre en az 6 karakter olmalı.");
            return false;
        }

        String sql = "INSERT INTO admin (name, email, password) VALUES (?, ?, ?)";
        try (Connection connection = DatabaseConfig.connect();
            PreparedStatement statement = connection.prepareStatement(sql)) {

            statement.setString(parameterIndex: 1, admin.getName());
            statement.setString(parameterIndex: 2, admin.getEmail());
            statement.setString(parameterIndex: 3, admin.getPassword());

            int rowsInserted = statement.executeUpdate();
            if (rowsInserted > 0) {
                System.out.println("Admin başarıyla eklendi!");
            }
        }
    }
}
```

```
public boolean deleteAdminByNameAndPassword(String name, String password) { no usages
    if (isEmpty(name, errorMessage: "Hata: Geçersiz admin adı.") ||
        isEmpty(password, errorMessage: "Hata: Geçersiz şifre.")) {
        return false;
    }

    String sql = "DELETE FROM admin WHERE name = ? AND password = ?";
    try (Connection connection = DatabaseConfig.connect();
        PreparedStatement statement = connection.prepareStatement(sql)) {

        statement.setString(parameterIndex: 1, name);
        statement.setString(parameterIndex: 2, password);

        int rowsDeleted = statement.executeUpdate();
        if (rowsDeleted > 0) {
            System.out.println("Admin başarıyla silindi!");
            return true;
        }
    }
}
```

```

String selectSQL = "SELECT * FROM admin WHERE name = ? AND password = ?";
String updateSQL = "UPDATE admin SET password = ? WHERE name = ? AND password = ?";

try (Connection connection = DatabaseConfig.connect();
    PreparedStatement selectStatement = connection.prepareStatement(selectSQL)) {

    selectStatement.setString( parameterIndex: 1, name);
    selectStatement.setString( parameterIndex: 2, currentPassword);

    ResultSet resultSet = selectStatement.executeQuery();

    if (resultSet.next()) {
        try (PreparedStatement updateStatement = connection.prepareStatement(updateSQL)) {
            updateStatement.setString( parameterIndex: 1, newPassword);
            updateStatement.setString( parameterIndex: 2, name);
            updateStatement.setString( parameterIndex: 3, currentPassword);

            int rowsUpdated = updateStatement.executeUpdate();
            if (rowsUpdated > 0) {
                System.out.println("Şifre başarıyla güncellendi!");
                return true;
            }
        }
    }
}

```

Arayüzden girilecek bir değere göre ekrana sonuçların listelendiği bir sorgu

```

public boolean addInvitation(int eventID, int userSessionId, String username) { // usage
    if (isEmptyOrNull(username, errorMessage: "Hata: Kullanıcı adı boş bırakılamaz.")) {
        return false;
    }

    // Kullanıcı ID'sini kullanıcı adı üzerinden alıyoruz
    int userID = usersDAO.getUserIDByUsername(username);
    if (userID == -1) {
        System.out.println("Hata: Davet gönderilecek kullanıcı bulunamadı.");
        return false;
    }

    if (eventID == -1) {
        System.out.println("Hata: Etkinlik bulunamadı.");
        return false;
    }

    try (Connection connection = DatabaseConfig.connect()) {

        String sql = "{? = CALL send_invitation_with_cursor(?,?,?,?)}";

        try (CallableStatement cstmt = connection.prepareCall(sql)) {
            cstmt.registerOutParameter( parameterIndex: 1, Types.VARCHAR);

            cstmt.setInt( parameterIndex: 2, eventID);
            cstmt.setInt( parameterIndex: 3, userSessionId);
            cstmt.setInt( parameterIndex: 4, userID);
            cstmt.setString( parameterIndex: 5, Invitation.Status.PENDING.toString());
        }
    }
}

```

View

```
CREATE OR REPLACE VIEW get_movies AS
  SELECT *
  FROM movies
  ORDER BY movie_id;

CREATE OR REPLACE VIEW get_users AS
  SELECT *
  FROM users
  ORDER BY user_id;
```

Sequence

```
CREATE SEQUENCE if not exists admin_id_seq
  START 1
  INCREMENT 1
  CACHE 1;
```

Union

```
CREATE OR REPLACE FUNCTION get_user_events(current_user_id INT)
    RETURNS TABLE(
        event_id INT,
        event_name VARCHAR,
        event_date DATE,
        user_id INT
    ) AS $$
BEGIN
    RETURN QUERY
        -- First part: events where the user is the organizer
        SELECT
            e.event_id,
            e.event_name,
            e.event_date,
            e.user_id
        FROM
            events e
        WHERE
            e.user_id = current_user_id

        UNION

        -- Second part: events where the user is invited and accepted the invitation
        SELECT
            e.event_id,
            e.event_name,
            e.event_date,
            e.user_id
        FROM
            invitations i
            JOIN
            events e ON e.event_id = i.event_id
        WHERE
```

Cursor

```
CREATE OR REPLACE FUNCTION send_invitation_with_cursor(  
    event_id_input INT,  
    current_user_id_input INT,  
    invited_user_id_input INT,  
    status_input TEXT  
)  
    RETURNS TEXT AS $$  
DECLARE  
    event_owner_id INT;  
    event_cursor CURSOR FOR  
        SELECT user_id  
        FROM events  
        WHERE event_id = event_id_input;  
BEGIN  
    -- Cursor'ı aç  
    OPEN event_cursor;  
  
    -- Cursor'dan değer al ve etkinlik sahibinin ID'sini kontrol et  
    FETCH event_cursor INTO event_owner_id;  
  
    -- Eğer etkinlik yoksa hata fırlat  
    IF NOT FOUND THEN  
        CLOSE event_cursor;  
        RETURN 'Etkinlik bulunamadı.';  
    END IF;
```


Trigger

```
CREATE OR REPLACE FUNCTION check_event_creator()
    RETURNS TRIGGER AS $$
BEGIN
    -- Eğer davetiye gönderilen kullanıcı etkinliği oluşturan kişi ise hata fırlat
    IF EXISTS (
        SELECT 1
        FROM events
        WHERE events.event_id = NEW.event_id AND events.user_id = NEW.user_id
    ) THEN
        RAISE EXCEPTION 'Event kurucusuna davetiye gönderilemez.';
    END IF;

    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

-- Trigger'ı oluştur
CREATE TRIGGER check_event_creator_trigger
    BEFORE INSERT ON invitations
    FOR EACH ROW
EXECUTE FUNCTION check_event_creator();
```

```
CREATE OR REPLACE FUNCTION update_movie_rating()
    RETURNS TRIGGER AS $$
BEGIN
    UPDATE movies
    SET rating = (
        SELECT AVG(rating)
        FROM votes
        WHERE movie_id = NEW.movie_id
    )
    WHERE movie_id = NEW.movie_id;
    RETURN NEW;
END;
$$ LANGUAGE plpgsql;

CREATE OR REPLACE TRIGGER trg_update_movie_rating
    AFTER INSERT OR UPDATE ON votes
    FOR EACH ROW
EXECUTE FUNCTION update_movie_rating();
```

Record

```
CREATE TYPE user_record_type AS (  
    user_id INT,  
    username VARCHAR,  
    email VARCHAR,  
    password VARCHAR  
);  
  
CREATE OR REPLACE FUNCTION get_user_by_credentials(user_name VARCHAR, user_password VARCHAR)  
    RETURNS SETOF user_record_type AS $$  
BEGIN  
    RETURN QUERY  
        SELECT  
            user_id,  
            username,  
            email,  
            password  
        FROM  
            users  
        WHERE  
            username=user_name AND password=user_password;  
  
END;  
$$ LANGUAGE plpgsql;
```

ER Diyagram

