MVC патерн

Model-View-Controller – це архітектурний шаблон проектування, який передбачає поділ системи на три частини: модель даних, інтерфейс та контролер, який з'єднує дві попередні частини. Даний патерн застосовується для полегшення роботи над кодом, оскільки всі три частини чітко відокремлені і можуть розроблятися окремо. Таким чином зміни в інтерфейсі користувача мінімально впливають на модель і, відповідно, навпаки, зміни в моделі, мінімально впливають на інтерфейс.

В моїй системі даний патерн представлений таким чином:

- Model. В даній частині у мене розміщені моделі даних (entity), взаємодія з базою даних (dao, DBManager), а також компаратори, які використовуються для порівняння моделей.
- View. В даній частині знаходяться всі jsp-сторінки, які відображаються у користувача. Також, тут же знаходяться стилі до цих сторінок та дескриптор розгортання веб-додатку (web.xml).
- Controller. В даній частині знаходиться головна частина, яка керує всім проектом. В моєму проекті, контролерами виступають спеціальні класи – сервлети, які приймають дані з jsp-сторінок, обробляють їх, та відсилають відповідь. Також тут розміщені спеціальні класи – фільтри, які обмежують доступ до тих чи інших сторінок.

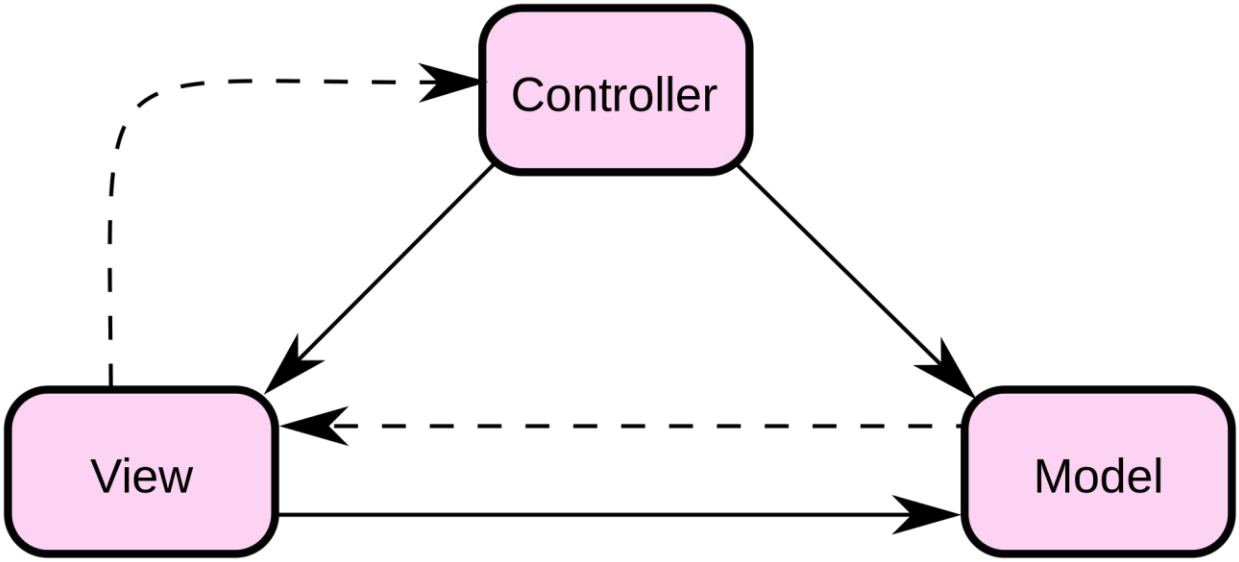

Рис. 1. MVC Pattern

Servlet

Servlet – це класи-контролери, які керують проектом. Кожний такий клас відповідає за обробку однієї сторінки, тому їх кількість збігається з jsp-сторінками (рис. 1). При виконанні коду цього класу створюються всі необхідні об’єкти для jsp сторінки. Після чого вони передаються на неї і відображаються у користувача. При виконанні користувачем певної дії, наприклад натиснення кнопки реєстрації, йде запит на сервер – до сервлета, де цей запит і обробляється.

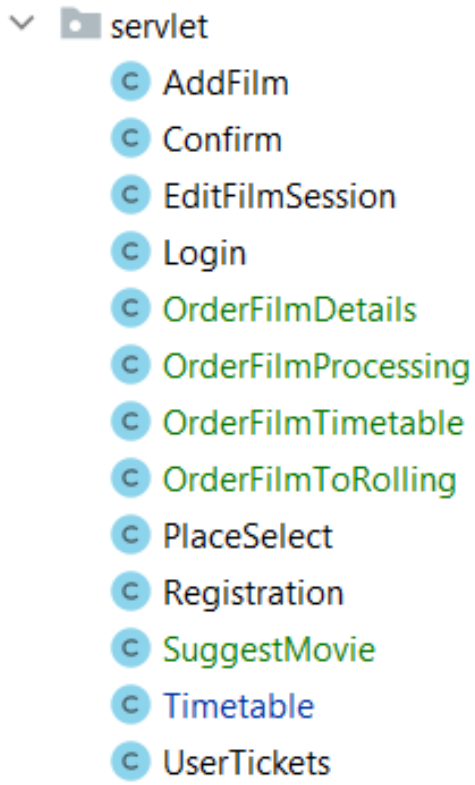

Рис. 1. Servlet класи системи

Приклад сервлету для сторіки логіну. При зверненні на сторінку сервлет обробляє запит методом doGet (рис. 2), який в даному випадку просто перенаправляє користувача на jsp сторінку з логіном.

```
@Override
protected void doGet(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    RequestDispatcher requestDispatcher = req.getRequestDispatcher( s: "WEB-INF/cinema/login.jsp");
    requestDispatcher.forward(req, resp);
}
```

Рис. 2. Метод doGet сервлета

У випадку натиснення кнопки логіну, запит обробляється іншим методом – doPost (рис. 3). Який метод буде обробляти запит – вказується в

формі на jsp сторінці за допомогою атрибута method. В даному прикладі (рис. 3) метод doPost звертається до бази даних і намагається знайти там користувача з введеним логіном. Якщо ж такого користувача не знайдено, то користувачу буде надіслано повідомлення, що логін не вірний. Якщо ж такий користувач існує, то йде перевірка правильності вводу пароля. При успішному логіненню користувач потрапляє на головну сторінку кінотеатру.

```
@Override
protected void doPost(HttpServletRequest req, HttpServletResponse resp) throws ServletException, IOException {
    String login = req.getParameter( s: "login");
    User user;
    String errorMessage = null;
    String active;

    try {
        UserDAO userDAO = new UserDAO();
        user = userDAO.getUser(login);
        try {
            if (CryptPassword.check(req.getParameter( s: "password"), user.getPassword())) {
                active = "active";
                req.getServletContext().setAttribute( s: "active", active);
                HttpSession userSession = req.getSession();
                userDAO.getUserDetails(user);
                userSession.setAttribute( s: "user", user);
                resp.sendRedirect( s: "/cinema");
                return;
            } else {
                errorMessage = "Incorrect password";
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
    } catch (SQLException e) {
        errorMessage = "Incorrect login";
    }
    req.setAttribute( s: "errorMessage", errorMessage);
    RequestDispatcher requestDispatcher = req.getRequestDispatcher( s: "/WEB-INF/cinema/login.jsp");
    requestDispatcher.forward(req, resp);
}
```

Рис. 3. Метод doPost сервлета

Web module

У моїй системі веб-модуль має назву view, оскільки він являє собою view частину патерну MVC. В даній частині знаходяться всі jsp-сторінки, які відображаються у користувача. Також, тут же знаходяться стилі до цих сторінок та дескриптор розгортання веб-додатку (web.xml).

Jsp (Java Server Page) – являє собою html сторінку з можливістю динамічно змінювати вміст сторінки, що робить ці сторінки дуже зручними. Також велики плюсом використання jsp сторінок є бібліотека jstl, яка дозволяє використовувати такі елементи як <c:if> для перевірки певної умови або <c:forEach> для відображення одразу списку об’єктів. Приклад сторінки для відображення квитків, придбаних користувачем:

```
<c:forEach var="ticket" items="${tickets}">
    <form class="ticket mt-5 mb-4 mx-auto bg-white p-4">
        <p class="h3 text-center text-black mb-4">Ticket</p>
        <ul>
            <li>
                <p class="h4">${ticket.filmName}</p>
            </li>
            <li>
                <p class="h5">${ticket.genre}</p>
            </li>
            <li>
                <p class="h5">Duration: ${ticket.duration} min</p>
            </li>
            <br>
            <li>
                <p class="h5">${ticket.date} ${ticket.time}</p>
            </li>
            <li>
                <p class="h5">Place: ${ticket.place}</p>
            </li>
            <br>
            <li>
                <p class="h5">Price: $${ticket.price}</p>
            </li>
        </ul>
    </form>
</c:forEach>
```

Рис. 4. Приклад jsp сторінки

Перелік jsp сторінок, які були використані в системі:

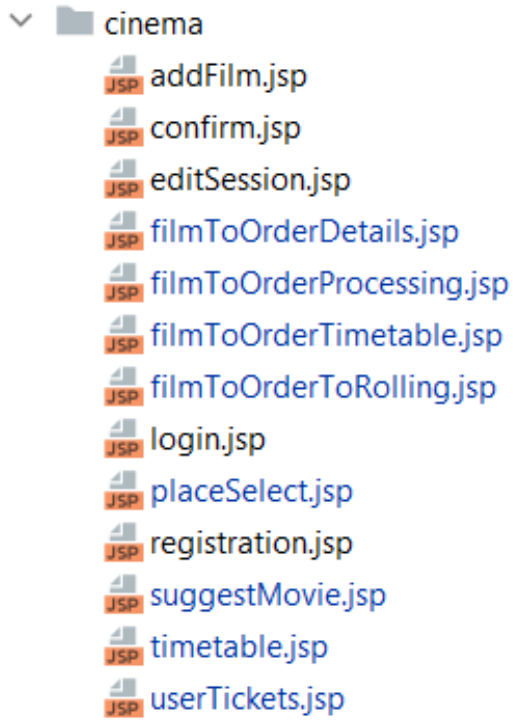

Рис. 5. Jsp сторінки системи

Web.xml

Даний конфігураційний файл являє собою дескриптор розгортання веб-додатку. Даний файл являє собою файл xml формату, який дуже часто використовується для конфігурації додатків. Головною превагою таких файлів є те, що розробник сам може створити розмітку, яка потрібна йому, таким чином він не обмежений ситаксичними правилами мови програмування.

Тут описуються та мапуються (задаються шляхи) всі сервлети системи (рис. 6), фільтри (рис. 7), а також, можна задавати інші параметри. Наприклад час, який буде тривати сесія користувача.

Web.xml створюється автоматично при додаванні до проекту веб-модуля.

```
<servlet>
    <servlet-name>Timetable</servlet-name>
    <servlet-class>com.cinema.controller.servlet.Timetable</servlet-class>
</servlet>

<servlet-mapping>
    <servlet-name>Timetable</servlet-name>
    <url-pattern>/cinema</url-pattern>
</servlet-mapping>
```

Рис. 6. Приклад мапування сервлета

```
<filter>
    <filter-name>UserFilter</filter-name>
    <filter-class>com.cinema.controller.filter.UserFilter</filter-class>
</filter>

<filter-mapping>
    <filter-name>UserFilter</filter-name>
    <servlet-name>Confirm</servlet-name>
    <servlet-name>UserTickets</servlet-name>
    <servlet-name>SuggestMovie</servlet-name>
</filter-mapping>
```

Рис. 7. Приклад фільтру та сторінки, до яких він застосовується

Pom.xml

Даний конфігураційний файл створюється автоматично при створенні додатку з структурою Maven. Pom.xml містить таку інформацію про проект як groupId, artifactId, version (рис. 8).
Також цей файл відповідає за додавання залежностей в проект (рис. 9). При додаванні будь-якої залежності в цей файл, вона буде автоматично скачана, встановлена, налаштована та готова до використання, що є дуже зручним.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<project xmlns="http://maven.apache.org/POM/4.0.0"
         xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
         xsi:schemaLocation="http://maven.apache.org/POM/4.0.0 http://maven.apache.org/xsd/maven-4.0.0.xsd">
    <modelVersion>4.0.0</modelVersion>

    <groupId>com.cinema</groupId>
    <artifactId>Cinema</artifactId>
    <version>1.0-SNAPSHOT</version>

    <packaging>war</packaging>
```

Рис. 8. Приклад pom.xml

```xml
<dependencies>

    <dependency>
        <groupId>mysql</groupId>
        <artifactId>mysql-connector-java</artifactId>
        <version>8.0.22</version>
    </dependency>

    <dependency>
        <groupId>commons-dbcp</groupId>
        <artifactId>commons-dbcp</artifactId>
        <version>1.4</version>
    </dependency>

    <dependency>
        <groupId>jstl</groupId>
        <artifactId>jstl</artifactId>
        <version>1.2</version>
    </dependency>

</dependencies>
```

Рис. 9. Додавання залежностей в проект

База даних

Першим кроком, який необхідно виконати після створення алгоритму – це розробка бази даних. Вона буде відображати структуру даних всієї системи. Для створення бази даних було використано MySQL Workbench, який містить зручні засоби для побудови таблиць та зв’язків між ними.

Були використані таки типи зв’язків між таблицями:

- One to one
- One to many
- Many to many

Зв’язок one to one створюється, коли у кожного об’єкту в одній таблиці може бути лише один зв’язок з іншою таблицею. Прикладом в моїй системі такий зв’язок є між таблицями account та account_details. Це означає, що в одного користувача може бути лише один запис в таблиці з його деталями.

Зв’язок one to many створюється, коли у об’єкта з однієї таблиці можуть бути багато зв’язків з іншою таблицею, але не навпаки. Прикладом такого зв’язку є таблиці account та role, які слугують для розподілення ролей користувачів. У користувача може бути тільки один зв’язок з таблицею role. А ось у таблиці role може бути багато зв'язків з таблицею account.

При створенні зв’язку many to many необхідно створити ще одну проміжну таблицю, яка буде зберігати id з обох таблиць, а також при необхідності інші параметри. Такий зв’язок дає змогу елементам з однієї таблиці зберігати зв’язки одразу з декількома об’єктами з іншої таблиці. Прикладом в моїй системі є зв’язок між таблицями account та film_to_order.

Автоматично створилася проміжна таблиця (user_vote), яка зберігає id на обидва об’єкти.

Акаунт користувача складається з 3 таблиць:

- Account (містить головну інформацію)
- Account_details (містить додаткову інформацію)
- Role (визначає роль користувача admin або user)

Фільм на прокат зберігає дані про фільми, які знаходяться в прокаті. Також було вирішено жанри винести в окрему таблицю, щоб уникнути повторення однієї і тієї ж інформації в записах фільмів.

- Film (інформація про фільм)
- Genre (зберігає жанри фільмів)

Для фільму на замовлення я вирішив створити окрему таблицю від фільму в прокаті, оскільки у фільму на замовлення є поля, які не будуть використовуватись для фільму на прокат, а також є необхідність у створенні додаткової таблиці з статусами, які будуть визначати статус пропонованого користувачем фільму, а також, є необхідність у створення додаткового зв’язку з таблицею account. Все це призведе до надлишкової інформації в таблиці з фільмом на прокат, тому створення окремої таблиці для фільму на замовлення є необхідним. Таблиці для фільму на замовлення:

- Film_to_order (містить всю інформацію про фільм на замовлення)
- Film_status (містить можливі статуси, в яких може бути фільм на замовлення: suggestion, voting, approved, rejected)
- User_vote (містить набір голосів користувача та фільмів, за які користувач проголосував)

Сеанси фільмів (сесії) – тут зберігаються сеанси кожного фільму. Також була створена додаткова таблиця зі статусами, оскільки сесія може бути у трьох станах: Available, No places, Canceled.

- Session (містить інформацію про сесії)
- Status (містить можливі статуси сесії)

Інформація про місця в залі зберігається також у двох таблицях. Одна містить інформацію про знаходження місця в залі. А інша містить інформацію про тип цього місця, та його вартість.

- Place (зберігає інфомацію про місце)
- Type (визначає тип місця (standard, lux, super lux) та його вартість)

Далі була створена проміжна таблиця між place та session, яка буде визначати конкретне місце на конкретній сесії, а також це місце може бути забронюваним користувачем. Також в цю таблицю було додане додаткове поле з часом бронювання місця. Це зроблено для того, щоб у випадку, якщо користувач покинув сторінку підтвердження вибору місця, то з збігом певного часу, заброньоване місце автоматично звільнилося.

- Session_has_place (містить інформацію про конкретне місце на певному сеансі)

Також була створена окрема таблиця для зберігання придбаних користувачем квитків. Квиток містить інформацію про фільм, сеанс, місце та ціну.

- Ticket (зберігає придбані користувачем квитки)

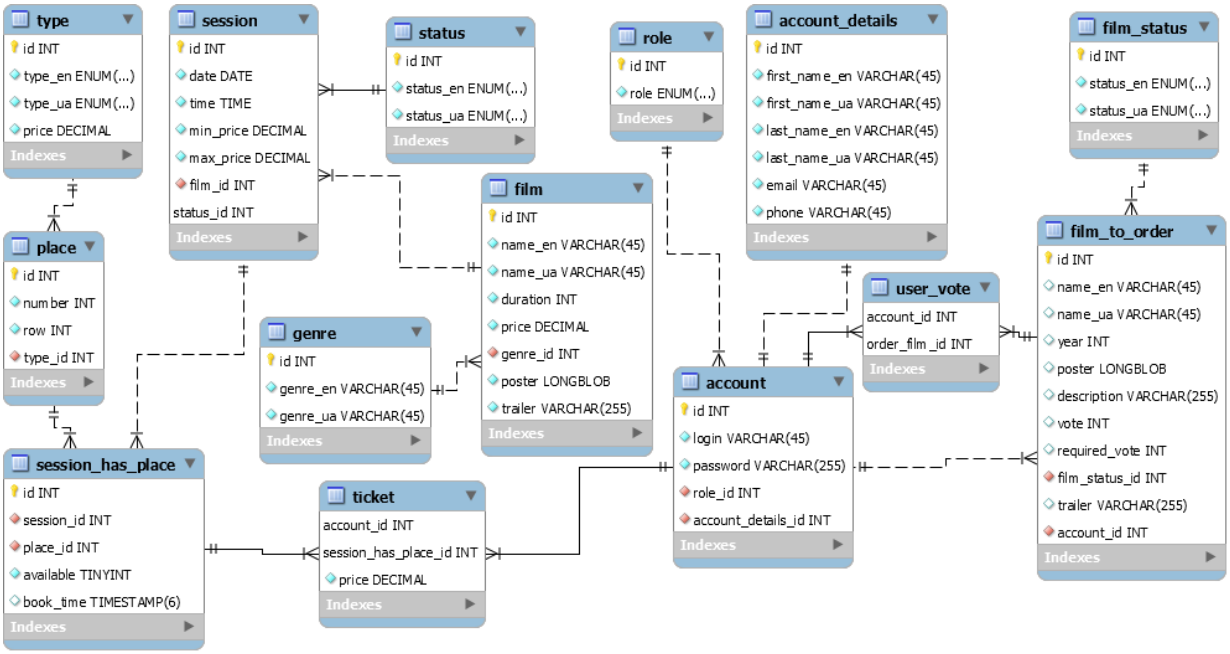

Рис. 10. Вигляд бази даних