

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

## **ПОЯСНЮВАЛЬНА ЗАПИСКА**

до кваліфікаційної роботи освітнього ступеня «бакалавр»  
за спеціальністю 121 «Інженерія програмного забезпечення»  
(освітня програма «Інженерія програмного забезпечення»)

на тему:

**«Онлайн-сервіс замовлення квитків та послуг  
“Кінотеатр”»**

Виконав студент групи ІПЗ-19-2  
СИМОН Олексій Вікторович

Керівник роботи:  
ЛОБАНЧИКОВА Надія Миколаївна

Рецензент:  
МОРОЗОВ Андрій Васильович

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ  
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»  
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ  
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

**«ЗАТВЕРДЖУЮ»**

В.о. зав. кафедри інженерії  
програмного забезпечення

Андрій МОРОЗОВ

«18» лютого 2023 р.

**ЗАВДАННЯ**  
**на кваліфікаційну роботу**

Здобувач вищої освіти: СИМОН Олексій Вікторович

Керівник роботи: ЛОБАНЧИКОВА Надія Миколаївна

Тема роботи: «Онлайн-сервіс замовлення квитків та послуг "Кінотеатр"»,  
затверджена Наказом закладу вищої освіти від «14» лютого 2023 р., №70с.

Вихідні дані для роботи: Методи та засоби розробки веб-сайту за визначеною предметною областю. Об'єктом дослідження є застосування технологій для реалізації онлайн-сервісу замовлення квитків. Перелік сервісів для реалізації, розклад сеансів, вартість квитків, кількість місць у залі, перелік додаткових опцій замовлення, технології реалізації, вимоги до сумісності програмного продукту.

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів, що їх стосуються:

| Розділ | Консультант      | Завдання видав | Завдання прийняв |
|--------|------------------|----------------|------------------|
| 1      | Лобанчикова Н.М. | 16.02.2023р.   | 16.02.2023р.     |
| 2      | Лобанчикова Н.М. | 16.02.2023р.   | 16.02.2023р.     |
| 3      | Лобанчикова Н.М. | 16.02.2023р.   | 16.02.2023р.     |

### Календарний план

| № з/п | Назва етапів випускної кваліфікаційної роботи | Термін виконання етапів роботи | Примітка |
|-------|-----------------------------------------------|--------------------------------|----------|
| 1     | Постановка задачі                             | 01.03.2023 - 25.03.2023        | Виконано |
| 2     | Пошук та аналіз додатків для реалізації       | 26.03.2023 - 29.03.2023        | Виконано |
| 3     | Формування технічного завдання                | 30.03.2023 - 03.04.2023        | Виконано |
| 4     | Опрацювання літературних джерел               | 04.04.2023 - 07.04.2023        | Виконано |
| 5     | Проектування структури                        | 08.04.2023 - 14.04.2023        | Виконано |
| 6     | Написання програмного коду                    | 15.04.2023 - 11.05.2023        | Виконано |
| 7     | Налагодження                                  | 12.05.2023 - 24.05.2023        | Виконано |
| 8     | Написання пояснювальної записки               | 25.05.2023 - 10.06.2023        | Виконано |
| 9     | Попередній захист                             | 26.04.2023                     | Виконано |

## РЕФЕРАТ

Випускна робота бакалавра складається з програмного комплексу онлайн-сервісу замовлення квитків та пояснювальної записки.

Пояснювальна записка до випускної роботи містить 69 сторінок, 30 ілюстрацій, 14 таблиць, 4 додатки та 19 літературних джерел.

Метою роботи є розробка онлайн-сервісу замовлення квитків, який надає користувачам зручний та простий спосіб придбати квитки на кіносеанси.

В роботі було визначено основні завдання на розробку системи, проаналізовано аналоги розробленої системи. Обґрунтовано використання архітектури додатку.

Проведено обґрунтування актуальності теми, аналіз методів та алгоритмів, а також досліджено можливості обраних технологій та їх актуальність. Наведено сценарії роботи програмного комплексу, загальна структура програмного комплексу, опис алгоритмів взаємодії окремих модулів системи, опис об'єктної структури системи. Також наводяться результати тестування системи.

Ключові слова: аналіз даних, модель класифікації, кінотеатр, замовлення квитків, API, PHP, WEB-ДОДАТОК, САЙТ, LARAVEL, СУБД, SPA, REACT JS.

|           |      |                 |        |      |                                                           |  |  |                                            |      |         |    |
|-----------|------|-----------------|--------|------|-----------------------------------------------------------|--|--|--------------------------------------------|------|---------|----|
|           |      |                 |        |      | ІПЗ.КР.Б – 121 – 23 – ІЗ                                  |  |  |                                            |      |         |    |
|           |      |                 |        |      |                                                           |  |  |                                            |      |         |    |
| Змн.      | Арк. | № докум.        | Підпис | Дата |                                                           |  |  |                                            |      |         |    |
| Розроб.   |      | О.В. Симон      |        |      | Онлайн-сервіс замовлення<br>квитків та послуг “Кінотеатр” |  |  | Літ.                                       | Арк. | Аркушів |    |
| Керівник  |      | Н.М.Лобанчикова |        |      |                                                           |  |  |                                            |      | 4       | 69 |
|           |      |                 |        |      |                                                           |  |  |                                            |      |         |    |
| Рецензент |      | А.В. Морозов    |        |      |                                                           |  |  | Житомирська політехніка,<br>група ІПЗ-19-2 |      |         |    |
| Зав. каф. |      | А.В. Морозов    |        |      |                                                           |  |  |                                            |      |         |    |
|           |      |                 |        |      | Пояснювальна записка                                      |  |  |                                            |      |         |    |

## ABSTRACT

The bachelor's thesis consists of a software complex of an online tickets ordering service and an explanatory note.

The explanatory note to the bachelor's thesis contains 69 pages, 30 illustrations, 14 tables, 4 appendices and 19 literary sources.

The aim of the work is to develop an online tickets ordering service that provides users with a convenient and easy way to get cinema tickets.

In the work, the main tasks for the development of the system were defined, and analogues of the developed system were analyzed. The application architecture usage is established.

The justification of the relevance of the topic, the analysis of methods and algorithms were investigated, as well as the possibilities of the selected technologies and their relevance. Such scenarios are given: the operation of the software complex, the general structure of the software complex, the description of the interaction algorithms of individual system modules, the description of the object structure of the system. A system for testing results is also given.

Keywords: data analysis, classification model, cinema, tickets ordering, API, PHP, WEB-APP, SITE, LARAVEL, DBMS, SPA, REACT JS.

|  |   |  |  |  |                           |   |
|--|---|--|--|--|---------------------------|---|
|  |   |  |  |  | ИПЗ.КР.Б – 121 – 23 – ИПЗ | . |
|  |   |  |  |  |                           |   |
|  | . |  |  |  |                           | 5 |

## ЗМІСТ

|                                                                                                                                                             |    |
|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----|
| ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....                                                                                                                              | 7  |
| ВСТУП .....                                                                                                                                                 | 8  |
| РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ<br>ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ОНЛАЙН-СЕРВІСУ ЗАМОВЛЕННЯ<br>КВИТКІВ ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР” ..... | 10 |
| 1.1 Постановка задачі .....                                                                                                                                 | 10 |
| 1.2 Аналіз аналогів програмного продукту .....                                                                                                              | 11 |
| 1.3 Вибір архітектури веб-орієнтованої системи до обраної теми .....                                                                                        | 13 |
| 1.4 Інструментальні засоби реалізації програмного продукту та вимоги до<br>апаратних засобів .....                                                          | 15 |
| Висновки до першого розділу.....                                                                                                                            | 16 |
| РОЗДІЛ 2 ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ЗАМОВЛЕННЯ КВИТКІВ<br>ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР” .....                                                           | 18 |
| 2.1 Визначення варіантів використання та об’єктно-орієнтованої структури<br>системи.....                                                                    | 18 |
| 2.2 Розробка структури бази даних системи .....                                                                                                             | 25 |
| 2.3 Проектування та реалізація алгоритмів роботи системи.....                                                                                               | 34 |
| 2.4 Реалізація онлайн-сервісу замовлення квитків та послуг “Кінотеатр” .....                                                                                | 39 |
| Висновки до другого розділу.....                                                                                                                            | 43 |
| РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-СЕРВІСУ<br>ЗАМОВЛЕННЯ КВИТКІВ ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР” .....                                          | 44 |
| 3.1 Порядок встановлення та налаштування параметрів системи .....                                                                                           | 44 |
| 3.2 Структура інтерфейсу та порядок взаємодії з онлайн-сервісом .....                                                                                       | 45 |
| 3.3 Тестування роботи програмного продукту .....                                                                                                            | 52 |
| Висновки до третього розділу .....                                                                                                                          | 53 |
| ВИСНОВКИ.....                                                                                                                                               | 55 |
| СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....                                                                                                                             | 57 |
| ДОДАТКИ.....                                                                                                                                                | 59 |

## ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

API – Application Programming Interface

REST – Representational state transfer

SPA – Single-Page Application

URI – Uniform Resource Identifier

БД – База даних

ІС – Інформаційна система

СУБД – Система управління базами даних

|  |   |  |  |  |                          |   |
|--|---|--|--|--|--------------------------|---|
|  |   |  |  |  | ІПЗ.КР.Б – 121 – 23 – ПЗ | . |
|  |   |  |  |  |                          |   |
|  | . |  |  |  |                          | 7 |

## ВСТУП

**Актуальність теми.** Кінотеатри є популярними місцями для розваг та культурного відпочинку. З плином часу зростає кількість фільмів, які виходять щомісяця, а також зростає кількість кінозалів. В цьому контексті ефективна інформаційна система та онлайн-сервіс замовлення квитків стають важливими для забезпечення зручності та задоволення потреб глядачів. Веб-орієнтована система замовлення квитків онлайн дозволяє глядачам замовляти квитки на фільми зручним для них способом, без необхідності особистого візиту до каси кінотеатру. Це забезпечує більшу доступність кінотеатральних послуг для широкої аудиторії та покращує користувацький досвід. Дозволяє автоматизувати процеси продажу квитків, управління розкладом сеансів, облік вільних місць та інші аспекти роботи кінотеатру. Це сприяє підвищенню ефективності, зменшенню людського впливу на процеси та запобіганню помилок. Наявність сучасної та функціональної системи може стати конкурентною перевагою для кінотеатру на ринку. Користувачі шукають зручність та швидкість обслуговування, і наявність такого сервісу може привернути більше клієнтів і підвищити рівень задоволення їх потреб.

Отже, розробка інформаційної системи, алгоритмів роботи та програмного комплексу Web-орієнтованої системи онлайн-сервісу замовлення квитків та послуг "Кінотеатр" має велику актуальність в контексті розвитку кінотеатральної індустрії, забезпечення зручності та доступності для глядачів, автоматизації процесів та підвищення конкурентоспроможності.

Метою дослідження є розробка алгоритмів роботи та реалізація програмного комплексу Web-орієнтованої системи онлайн-сервісу замовлення квитків та послуг "Кінотеатр".

Досягнення поставленої мети передбачає розв'язання таких завдань:

- проведення аналізу задач проблемних тем майбутніх клієнтів та напрямків їх оптимального вирішення;
- визначення архітектури та узагальненої структури системи;

- розробка інформаційного та математичного забезпечення системи;
- перевірки та тестування працездатності розробленої технології.

Об’єктом дослідження є процес побудови моделі для покращення та полегшення обслуговування та замовлення квитків.

Предметом дослідження є використання Web-орієнтованих технологій для розробки онлайн-сервісу замовлення квитків та послуг “Кінотеатр”.

Методи дослідження. В кваліфікаційній роботі було використано такі методи: спостереження та інтерв'ювання; візуального програмування та математичного моделювання; веб-програмування.

Практичне значення роботи полягає у проектуванні та програмній реалізації онлайн-сервісу замовлення квитків та додаткових послуг “Кінотеатр”. Розроблена інформаційна технологія дозволяє підвищити якість надання послуг кінотеатрами, сприяє підвищенню ефективності, зменшенню людського впливу на процеси та запобіганню помилок. Проведене тестування розробленого програмного продукту довело його працездатність та готовність до експлуатації.

# РОЗДІЛ 1 АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ ОНЛАЙН-СЕРВІСУ ЗАМОВЛЕННЯ КВИТКІВ ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР”

## 1.1 Постановка задачі

Постановка задачі полягає в конкретному визначенні завдань і цілей проекту. У випадку онлайн-сервісу замовлення квитків та послуг "Кінотеатр", постановка задачі включає наступні етапи:

1. Розробка інтерфейсу користувача: Створення зручного та привабливого інтерфейсу, що дозволяє користувачам легко знаходити інформацію про фільми, сеанси та здійснювати замовлення квитків.
2. Розробка функціональності замовлення квитків: Реалізація можливості вибору фільму, сеансу, вибору місць та оплати квитків онлайн.
3. Розробка адміністративної панелі: Розробка інтерфейсу та функціоналу для адміністратора, що дозволяє керувати інформацією про фільми, сеанси, кінозали та переклади. Включає можливості додавання та редагування, а також керування доступністю сеансів.
4. Інтеграція з платіжними системами: Забезпечення можливості безпечної оплати квитків онлайн.
5. Розробка системи управління користувачами: Реалізація системи реєстрації та авторизації користувачів, забезпечення безпеки та конфіденційності персональних даних користувачів.
6. Тестування та налагодження: Проведення тестування системи для перевірки її функціональності, надійності та продуктивності. Виправлення помилок та вдосконалення системи на основі результатів тестування.

Постановка задачі детально описує обсяг робіт та визначає основні цілі, які необхідно досягти під час розробки онлайн-сервісу замовлення квитків та послуг "Кінотеатр".

## 1.2 Аналіз аналогів програмного продукту

Аналіз аналогів програмного продукту на прикладі кінотеатрів "Multiplex" та "Planeta Kino" може надати цінну інформацію про існуючі рішення у сфері онлайн-сервісів замовлення квитків для кінотеатрів. Давайте розглянемо деякі особливості цих кінотеатрів та їхніх веб-орієнтованих систем.

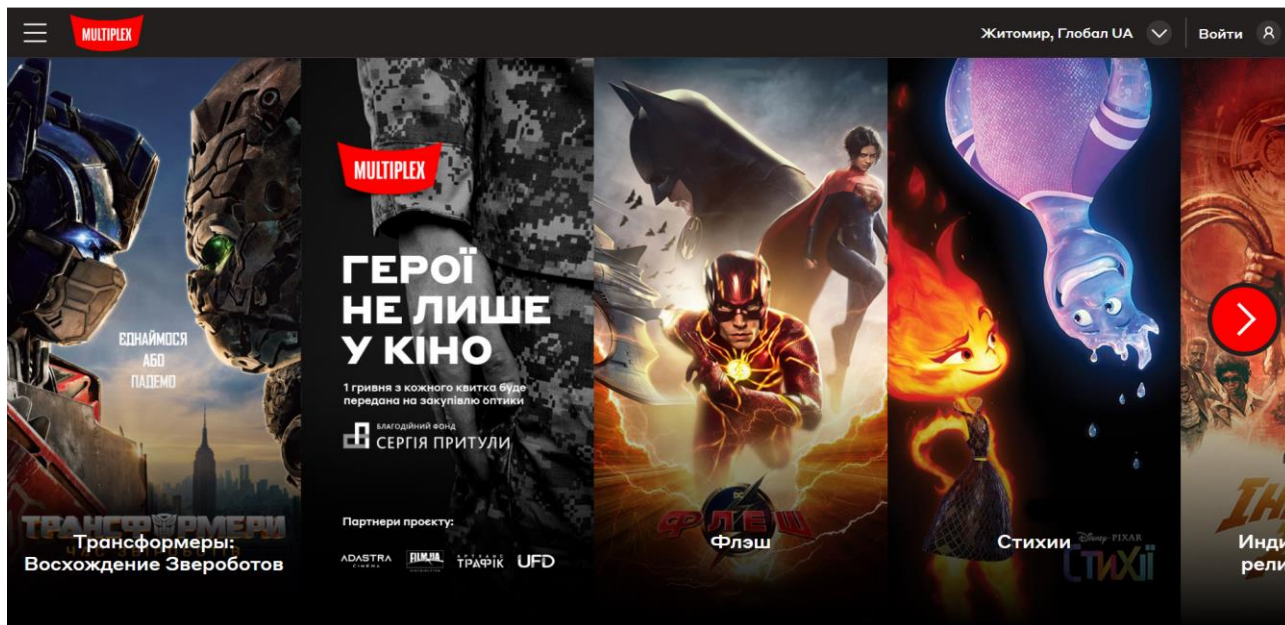


Рисунок 1.1 – Приклад онлайн сервісу “Multiplex”

"Multiplex" є одним з найбільших сервісів надання перегляду контенту в Україні. У них є веб-сайт, який дозволяють користувачам замовляти квитки онлайн (рис.1.1).

Основні особливості їхньої системи включають:

1. Зручний веб-інтерфейс: Веб-сайт "Multiplex" мають простий і зрозумілий інтерфейс, де користувачі можуть швидко знайти потрібний фільм, переглянути розклад сеансів та замовити квитки.
2. Різноманітні послуги: Крім замовлення квитків, система "Multiplex" також пропонує додаткові послуги, такі як вибір місць у залі, можливість обрати залу тощо.

Потенційні мінуси систем замовлення квитків онлайн у кінотеатрах "Multiplex":

1. Не завжди існує детальна інформація про наявність місць у залі під час замовлення квитків, що може створювати незручності для користувачів, які бажають вибрати місце.
2. Іноді можуть виникати проблеми з онлайн-платежами, такі як технічні помилки під час транзакцій.

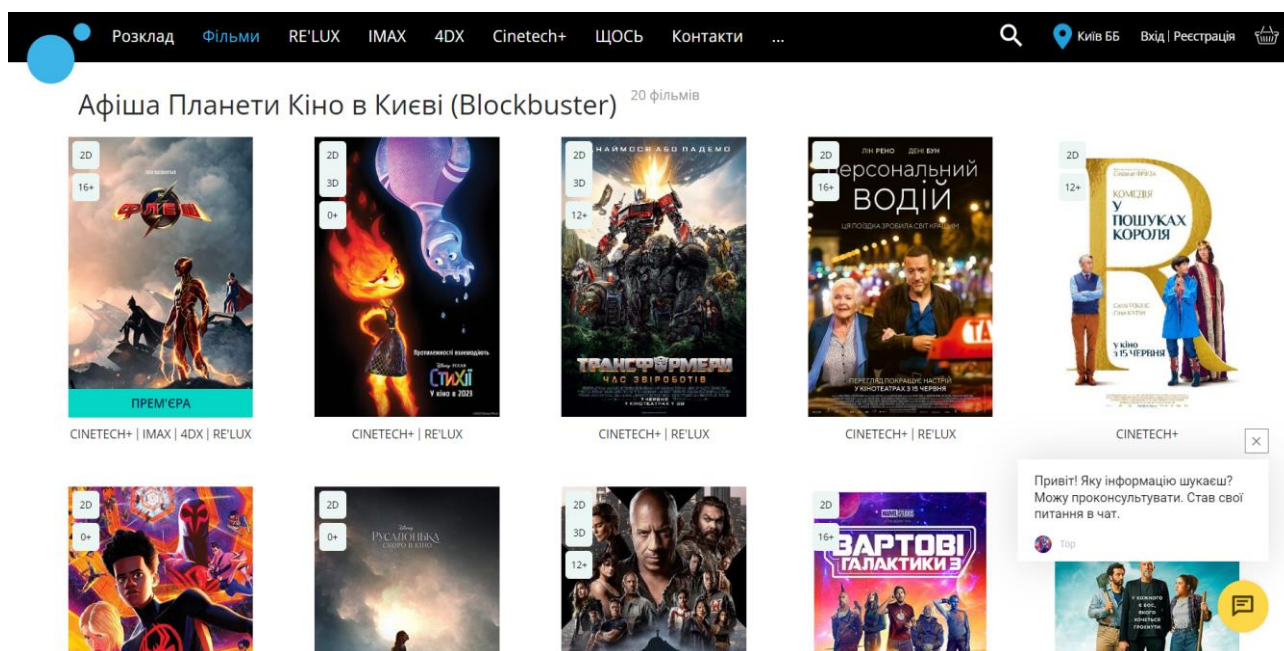


Рисунок 1.2 – Приклад онлайн сервісу “Planeta Kino”

"Planeta Kino", (рис.1.2), також є популярним кінотеатральним оператором в Україні, який надає можливість замовляти квитки онлайн через свій веб-сайт. Деякі особливості системи "Planeta Kino" включають:

1. Персоналізований підхід: В системі "Planeta Kino" користувачам пропонується персоналізований підхід, де вони можуть створювати облікові записи, зберігати свої вподобання та історію замовлень.
2. Онлайн-оплата та бронювання: Користувачі можуть зручно оплачувати квитки онлайн через платіжні системи.

Потенційні мінуси систем замовлення квитків онлайн у кінотеатрах "Planeta Kino":

1. Деякі користувачі можуть відзначати, що інтерфейс веб-сайту "Planeta Kino" може бути не дуже зручним або недостатньо інтуїтивним, що може викликати певну складність у процесі замовлення квитків.
2. У деяких випадках, інформація про розклад сеансів або доступність квитків може оновлюватися з певною затримкою, що може призвести до ситуацій, коли користувачі не можуть отримати актуальну інформацію під час замовлення.

Загалом, обидва кінотеатри "Multiplex" та "Planeta Kino" пропонують зручні та функціональні системи замовлення квитків онлайн, де користувачі можуть швидко знайти і замовити квитки на бажаний фільм, обирати місця, користуватися додатковими послугами та отримувати персоналізований підхід. Аналіз цих аналогів може допомогти врахувати їхні сильні сторони та недоліки при розробці веб-орієнтованої системи замовлення квитків для нашого проекту "Кінотеатр".

### **1.3 Вибір архітектури веб-орієнтованої системи до обраної теми**

Вибір архітектури веб-орієнтованої системи для обраної теми "Онлайн-сервіс замовлення квитків та послуг "Кінотеатр"" є важливим етапом проекту. Продукт має бути високоякісною, зрозумілою, доступною, легкою для розуміння та використання.

Мета архітектури – забезпечити структурованість, надійність, розширюваність та керованість програмного рішення. Вона дозволяє розробникам розуміти взаємозв'язок між компонентами та вирішувати складні проблеми, пов'язані з розробкою та підтримкою програмного забезпечення.

Порівняння та загальний вигляд SPA [1] та Клієнт-серверної архітектури представлено на рисунку 1.3. Архітектура Single-Page Application (SPA) може бути відмінним варіантом для веб-орієнтованої системи замовлення квитків для кінотеатру. Вона надає швидкий та зручний інтерфейс для користувачів, забезпечуючи плавну взаємодію без перезавантаження сторінок.

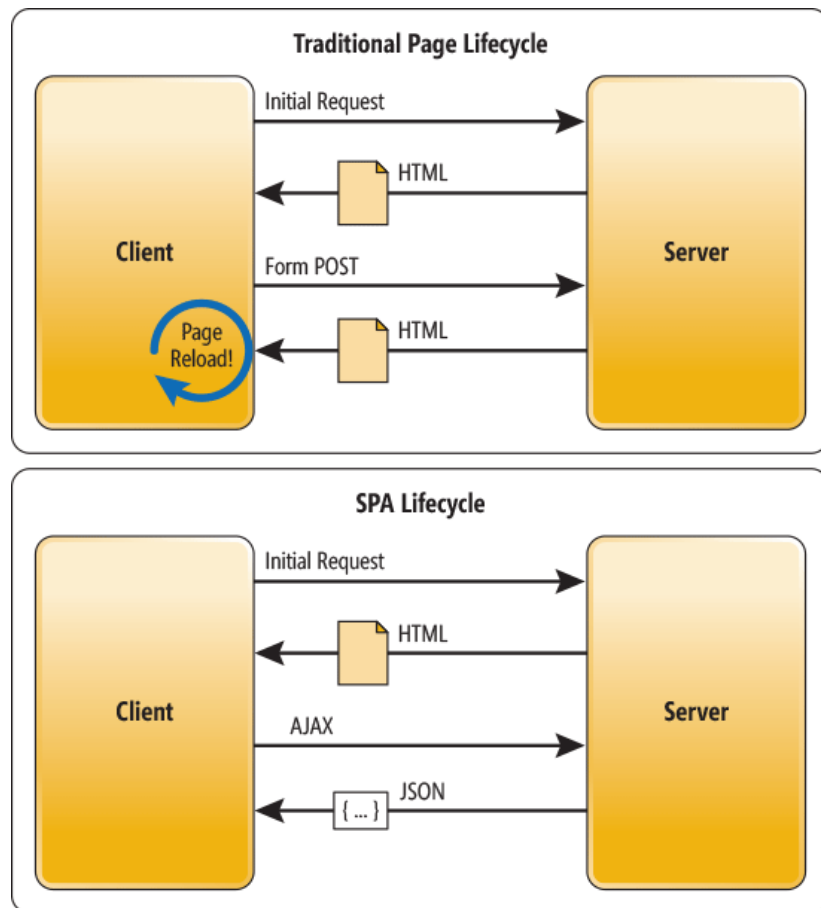


Рисунок 1.3 – Порівняння та загальний вигляд SPA та Клієнт-серверної архітектури

Основні компоненти такої архітектури включають:

- Front-end (клієнтська частина): Це додаток, що виконується на боці користувача. Використовуючи HTML, CSS та JavaScript, SPA завантажується один раз при початковому завантаженні сторінки. Він реагує на дії користувача та взаємодіє з сервером, відображаючи змінену інформацію без повного перезавантаження.
- Back-end (серверна частина): Це компонент, що відповідає за обробку запитів від клієнта. Він отримує запити, виконує необхідні операції, такі як обробка замовлень, оплата, робота з базою даних тощо, і повертає відповіді у форматі JSON [2] або XML.
- API [3] (Application Programming Interface): Це набір функцій та протоколів, які дозволяють взаємодіяти між клієнтом та сервером. SPA використовує API для взаємодії з сервером, надсилаючи запити та отримуючи відповіді.

- Дані: SPA може залежати від бази даних, яка зберігає інформацію про фільми, розклад сеансів, наявність квитків та іншу важливу інформацію. Дані можуть бути доступні через API та використовуватися для відображення інформації для користувачів.

Переваги архітектури SPA включають:

- Більш плавну та швидку взаємодію, оскільки сторінки не перезавантажуються повністю.
- Зручну навігацію та інтерфейс, оскільки користувачі можуть швидко переміщатися між різними розділами та функціями.
- Можливість створювати багатофункціональні додатки без необхідності перезавантаження сторінки.
- Високу масштабованість та зручність для розробки, оскільки розподілення логіки між клієнтом та сервером дозволяє незалежно розробляти фронтенд та бекенд.

#### **1.4 Інструментальні засоби реалізації програмного продукту та вимоги до апаратних засобів**

Інструментальні засоби реалізації програмного продукту включають набір програмних інструментів та технологій, які використовуються для розробки, тестування та впровадження системи.

Для написання системи було обрано редактор PhpStorm, так як компанія JetBrains для студентів надає студентську підписку.

Мови програмування: Для розробки веб-орієнтованої системи були використані мови програмування, такі як JavaScript та PHP. JavaScript є основною мовою для реалізації фронтенду системи, тоді як PHP використовується для реалізації бекенду.

Фреймворки: Фреймворк, такий як React, використовується для розробки клієнтської частини системи – фронтенду. На бекенді був використаний фреймворк – Laravel.

Бази даних: Для зберігання інформації про фільми, сеанси, кінозали та інші дані, була використана реляційна база даних – MySQL.

OpenServer – це легке та зручне у використанні серверне середовище, спеціально розроблене для роботи з PHP, MySQL та Apache. Він включає в собі всі необхідні компоненти для запуску веб-додатків, такі як веб-сервер, база даних та інтерпретатор PHP.

Системи керування версіями: Системи керування версіями, такі як Git, можуть використовуватись для контролю версій коду та спільної роботи над проектом.

Системні вимоги:

1. Вимоги до середовища виконання:

Система повинна задовольняти зазначені вимоги на комп'ютері в наступній мінімальній комплектації:

- підтримка браузера Google Chrome версії 45+ або;
- підтримка браузера Mozilla Firefox версії 38+ або;
- підтримка браузера Microsoft Edge версії 12+ або;
- підтримка браузера Opera версії 30+ або;
- підтримка браузера Safari версії 9+;

2. Вимоги до серверного обладнання:

- обсяг оперативної пам'яті – 8 GB;
- обсяг жорсткого диску від 250 GB;
- операційна система – Linux;
- процесор з частотою від 3GHz;
- node.js версії 18 і вище;

### Висновки до першого розділу

1. Обґрунтовано актуальність теми роботи, яка полягає в тому, що розробка інформаційної системи для онлайн-сервісу замовлення квитків має відповідати потребам сучасних користувачів, які шукають зручність та

швидкість у процесі замовлення. Веб-орієнтована система допоможе полегшити процес замовлення квитків для користувачів та покращити роботу кінотеатру.

2. Проведено аналіз аналогів програмного продукту на прикладі кінотеатрів Multiplex та Planetakino, які вже мають онлайн-сервіси замовлення квитків. Multiplex надає широкий вибір фільмів, але має обмежену наявність додаткових послуг та може бути складним у виборі місць. Planetakino має обмежений вибір фільмів та менш розвинену функціональність. Це дає можливість для розробки конкурентоспроможного продукту з більш широким функціоналом та більшим вибором фільмів.

3. Проведено аналіз та обґрунтування вибору архітектури веб-орієнтованої системи до обраної теми. Архітектура SPA (Single-Page Application) була обрана як оптимальний варіант, оскільки вона забезпечує швидкий та зручний інтерфейс для користувачів, уникнення перезавантаження сторінок та легку масштабованість.

## РОЗДІЛ 2 ПРОЕКТУВАННЯ ОНЛАЙН-СЕРВІСУ ЗАМОВЛЕННЯ КВИТКІВ ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР”

### 2.1 Визначення варіантів використання та об’єктно-орієнтованої структури системи

Розглянемо високорівневі вимоги, якими повинна володіти система, щоб мати можливість безперервно виконувати покладені на неї функції, тим самим відповідно задовольнити всі зазначені стандарти, специфікації та інші формальні документи. Зазначені умови, яких повинен дотримуватись користувач для роботи з даною інформаційною системою та вказані основні ділові переваги даної ІС. Приведені можливості системи, обмеження та показники якості.

Бізнес-вимоги:

1. Основні цілі: покращення обслуговування клієнтів та легкий доступ до інформації про проведення сеансів фільмів.
2. Можливості: перегляд сеансів, придбання квитків, оплата квитків.
3. Представлення проекту: проект буде реалізовано у вигляді веб-додатку, що містить зручний інтерфейс та необхідну інформацію за даною тематикою.

Вимоги користувачів:

- Зовнішні користувачі – авторизований користувач
  1. Можливість перегляду доступних сеансів;
  2. Можливість переглядати обраний сеанс та обрати доступні місця;
  3. Можливість купити квитки;
  4. Можливість перегляду квитків, які вже купувались;
  5. Можливість перегляду трейлера;
- Внутрішній користувач – адміністратор
  1. Редагування БД;

### Характеристика об'єкту комп'ютеризації:

Користувач буде мати можливість переглянути сеанси, доступні фільми та трейлери до них, які показують в кінотеатрі, обрати доступне місце та купити квитки. Адміністратор через веб-інтерфейс буде мати можливість редагувати, додавати дані, користуватися всіма функціями системи.

### Функціональні вимоги:

1. Підключення сторонніх API для реалізації оплати.
2. Підключення пакетів для отримання готових модулів.
3. Створення модуля для замовлення квитків.

### Нефункціональні вимоги:

#### 1. Сприйняття

- для звичайних користувачів на навчання роботи з інформаційною системою необхідно витратити 2-3 години, а для досвідчених цей час скорочується до 1 години.
- Інтерфейс має бути легким у використанні для користувача та не вимагати додаткового навчання.
- Система повинна надавати відповідь на звичайні запити протягом не більше 5 секунд, а на складніші запити – протягом 10 секунд.

#### 2. Продуктивність

- Система повинна забезпечувати одночасну роботу принаймні 50 користувачів, які взаємодіють зі спільною базою даних.

### Системні вимоги:

#### 1. Вимоги до середовища виконання:

Система повинна задовольняти зазначені вимоги на комп'ютері в наступній мінімальній комплектації:

- підтримка браузера Google Chrome версії 45+ або
- підтримка браузера Mozilla Firefox версії 38+ або
- підтримка браузера Microsoft Edge версії 12+ або
- підтримка браузера Opera версії 30+ або
- підтримка браузера Safari версії 9+

## 2. Вимоги до СУБД та доступу до даних:

У ядрі системи повинна бути представлена СУБД реляційного доступу;

Бізнес правила:

Система повинна відповідати всім вимогам та стандартам, які використовуються при розробці сучасних веб-застосунків на базі архітектури SPA (Single-page application).

Антивимоги:

- система не використовує недійсну інформацію або ту, що може завдати шкоди психіці користувачів.
- система не допускає використання інформації, що суперечить нормам моралі та законодавству.

Бізнес-логіка:

Система пропонує послуги, а саме купівлю квитків для полегшення обслуговування клієнтів.

Підсумовуючи, побудуємо діаграму варіантів використання (Use Case Diagram) [4]. На рисунку 2.1 зображено побудовану діаграму варіантів використання.

Неавторизований користувач має можливість зареєструватись на сайті, або перед цим переглянути доступний контент, інформацію на сайті про проведення сеансу та ознайомитись з детальною інформацією.

Авторизований користувач має можливість увійти до особистого кабінету, де зможе переглянути вже раніше придбані квитки та інформацію про себе. Є можливість обрати сеанс та місце для перегляду, після чого оплатити квитки.

Адміністратор потрапляє до адміністративної панелі, де зможе редагувати інформацію про фільм, додавати сеанси, кінозали та переклади інформації (система передбачає можливість перемикання мови та виведення контенту для користувача на Англійській та Українській мовах).

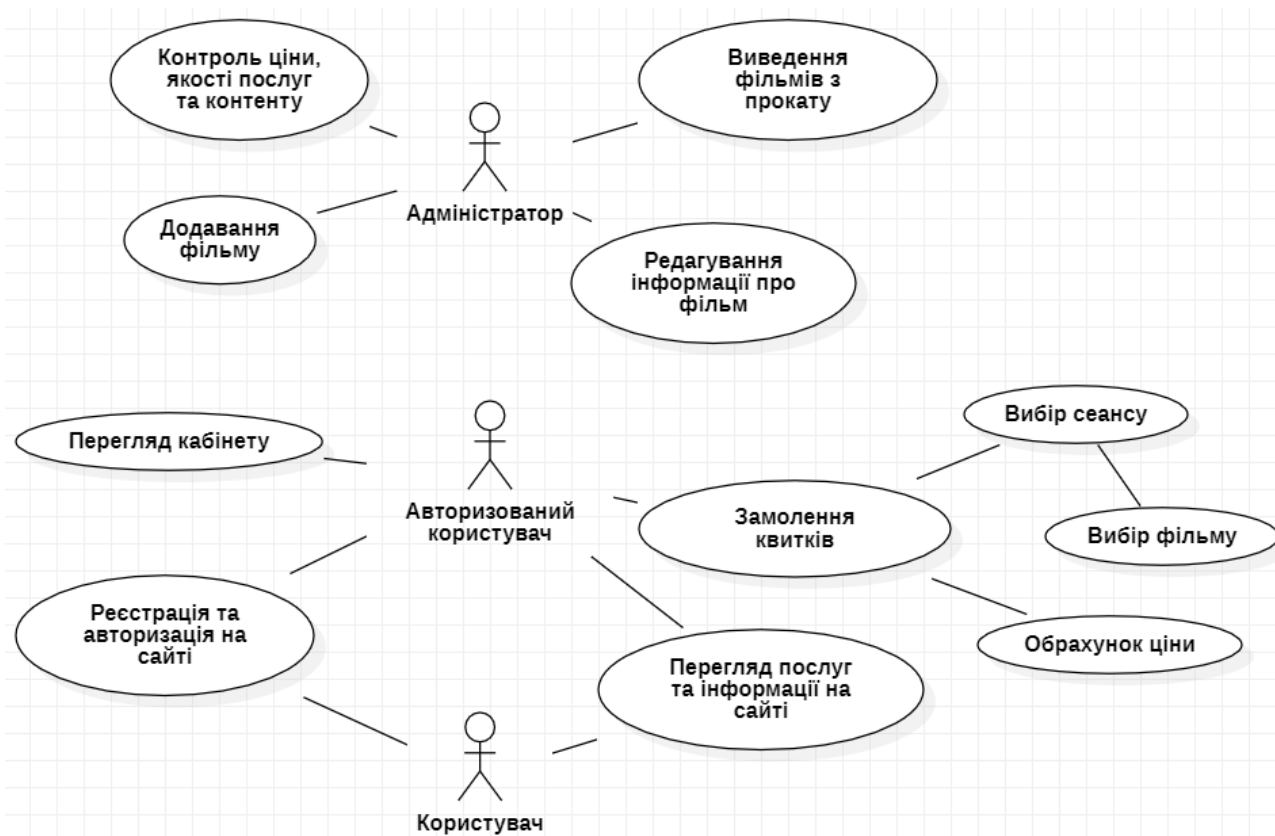


Рисунок 2.1 – Діаграма варіантів використання

Об'єктно-орієнтовану модель системи прийнято розглядати у вигляді діаграми класів (Class diagram) [5]. Для побудови даної діаграми спочатку потрібно розібратися, що вона собою представляє.

Діаграма класів – це тип діаграми статичної структури, який описує структуру системи, показуючи класи системи, їхні атрибути, операції (або методи) і зв'язки між об'єктами. На рисунку 2.2 зображено побудовану діаграму класів.

Нижче наведено опис класів, а також їх атрибутів та методів:

- Клас Users містить інформацію про зареєстрованого користувача та реалізовує методи, що відповідають за визначення адміністратора та роботу з паролями.
- Клас Orders передбачає створення ордеру для клієнта при оплаті квитка.
- Клас Tickets передбачає створення квитка для клієнта виходячи з попередньої інформації про фільм та залу.

- Клас Cinema Hall відповідає за зали в кінотеатрі, їх кількість та кількість місць для сидіння.

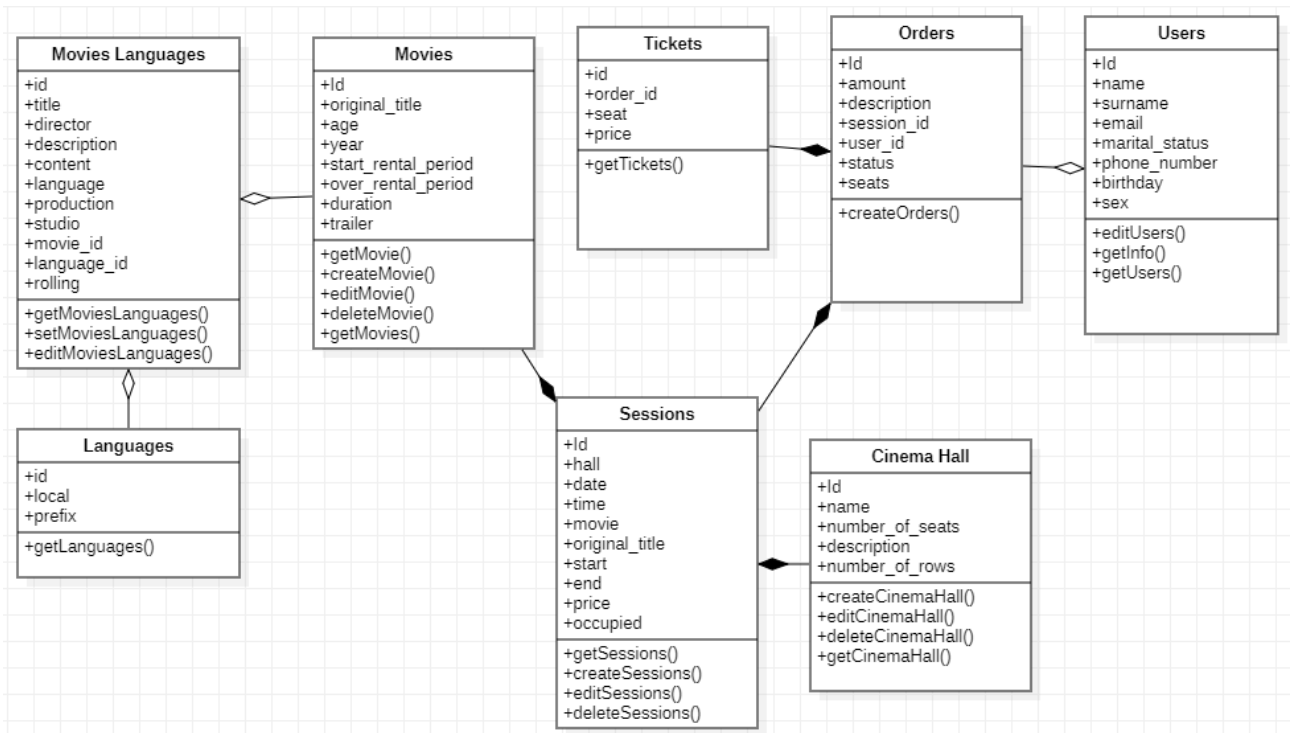


Рисунок 2.2 – Діаграма класів

- Клас Sessions відповідає за сеанси, а саме час проведення, який фільм буде проходити в даний час , місця які вже є зарезервованими та в якому залі.
- Клас Movies відповідає за фільми які показують в даний час в кінотеатрі.
- Клас Movies Languages відповідає за переклад фільму відповідно його id та локалізації користувача.
- Клас Languages відповідає за локалізацію сайту.

За допомогою патернів проектування [6], технології змінюються надзвичайно швидко, вони використовуються для спрощення та розв'язання проблем. Використання цих патернів значно полегшує процес роботи з технологіями. Тому вже на етапі моделювання майбутньої системи детально продумується, які патерни будуть використовуватись.

Тому було вирішено застосувати патерн "Репозиторій" [7], що сприяє покращенню продуктивності та обробки даних для користувача.

“Repository” – патерн, який розділяє рівні джерела даних і логіки програми. Це класи або компоненти, які інкапсулюють логіку, необхідну для доступу до джерел даних. Вони централізують спільну функціональність доступу до даних, забезпечуючи кращу зручність обслуговування та відокремлюють інфраструктуру або технологію, що використовується для доступу до бази даних, від рівня моделі предметної області. Часто використовується із патерном “Unit Of Work”. На рисунку 2.3 зображено патерн “Repository”.

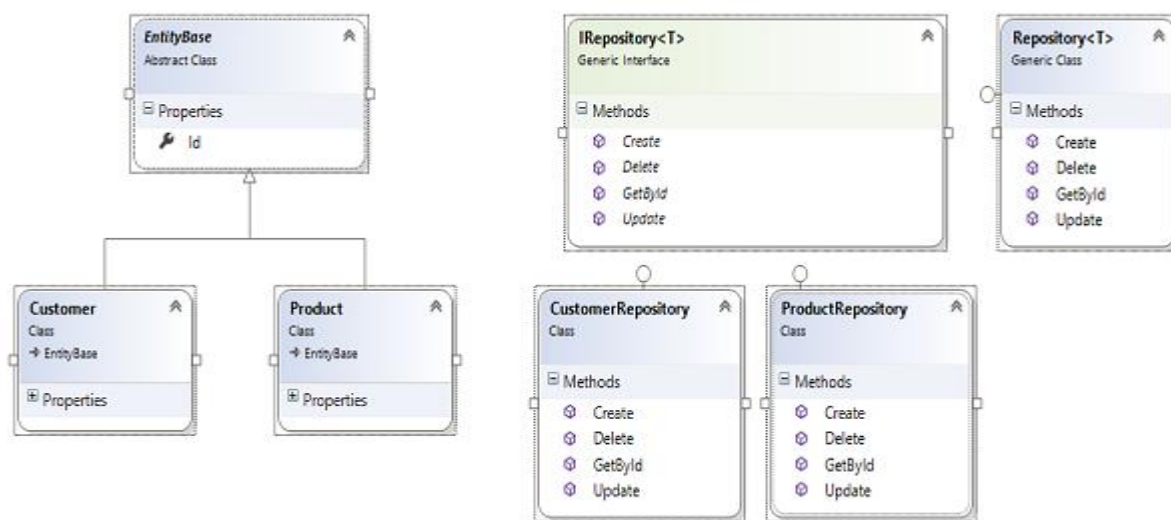


Рисунок 2.3 – Реалізація патерну “Repository”

Крім того, для розробки системи був використаний патерн "Unit Of Work" [8]. Однак, немає потреби його реалізовувати, оскільки більшість фреймворків вже надають цей патерн вбудованим засобом. Запропонований патерн “Unit Of Work” пропонує відстежувати всі зміни об'єктів і вносити їх у вигляді однієї транзакції.

Unit of Work (одиниця роботи) – це патерн проектування, який використовується для керування транзакціями та змінами даних в контексті реляційних баз даних. Він дозволяє групувати кілька операцій збереження даних в одну транзакцію, що дозволяє забезпечити консистентність та атомарність змін.

У контексті патерну Unit of Work, "одиниця роботи" визначає границі однієї транзакції, в якій відбуваються зміни даних. Вона включає в себе оператори додавання, видалення та оновлення даних. При використанні цього

патерна, зміни в об'єктах зберігаються в буфері та виконуються разом при збереженні одиниці роботи (транзакції) до бази даних.

Цей патерн дозволяє підтримувати цілісність та консистентність даних, спрощує управління транзакціями та покращує продуктивність шляхом зменшення кількості взаємодій з базою даних. Багато сучасних фреймворків і бібліотек надають вбудовану підтримку для патерну Unit of Work, що дозволяє легко і ефективно використовувати його у розробці програмного забезпечення.

На рисунку 2.4 зображено вигляд патерну “Unit Of Work”.

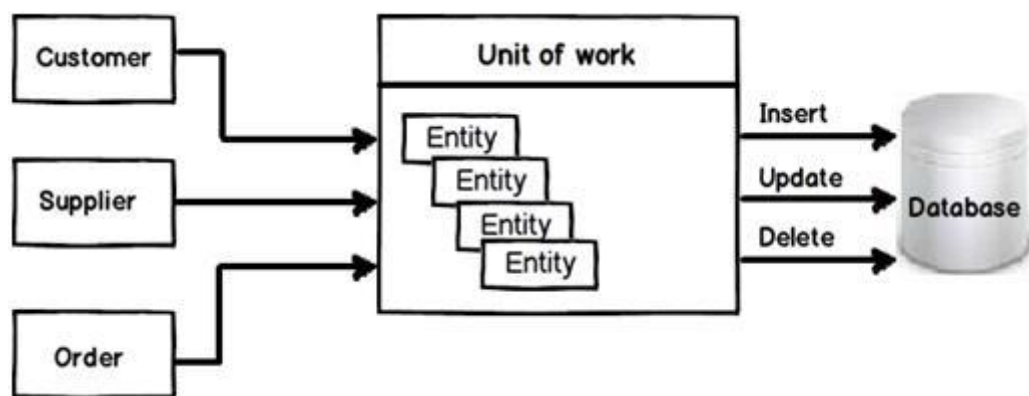


Рисунок 2.4 – Вигляд патерну “Unit Of Work”

React [9] може бути використаний для реалізації функціонального програмування у клієнтській частині веб-додатку. Функціональне програмування – це парадигма програмування, яка ставить акцент на використанні функцій як основних будівельних блоків програми.

У React функціональне програмування може бути досягнуто за допомогою функціональних компонентів. Функціональні компоненти в React є простими JavaScript функціями, які приймають вхідні дані (props) і повертають відображення (UI) на основі цих даних.

Крім того, React пропонує вбудовані хуки, такі як useState та useEffect, які дозволяють використовувати стан та ефекти в функціональних компонентах, зберігаючи при цьому імутабельність даних та ізолюваність ефектів.

Таким чином, React добре підходить для реалізації функціонального програмування у клієнтській частині, надаючи зручні засоби для створення функціональних компонентів і сприяючи розробці чистого та модульного коду.

## 2.2 Розробка структури бази даних системи

Для розробки БД було використано MySQL [10]. MySQL – це відкрите програмне забезпечення для керування базами даних, що базується на моделі реляційної бази даних. Він є одним з найпопулярніших та широко використовуваних реляційних СУБД (систем управління базами даних). MySQL надає багато функцій та можливостей для зберігання, організації та маніпулювання даними.

MySQL є популярним вибором для багатьох типів проектів, включаючи веб-додатки, системи керування вмістом, електронну комерцію та багато іншого. Його простота використання, продуктивність та широкі можливості роблять його потужним інструментом для роботи з базами даних.

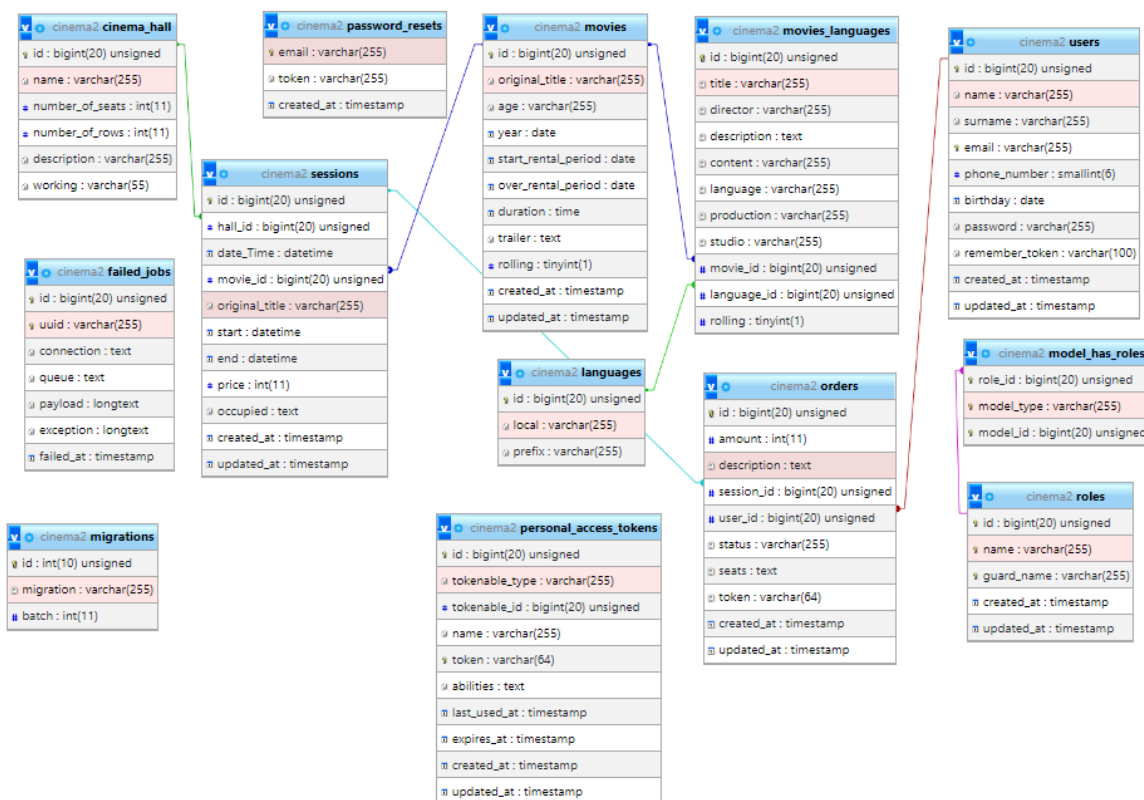


Рисунок 2.3 – Діаграма «сутність-зв'язок» (логічний рівень) системи

Для збереження даних про ордери, була створена таблиця “orders”, табл.2.1. Вона містить інформацію про ордер з інформація про оплату.

Таблиця 2.1

Структура таблиці “orders”

| Назва       | Тип даних   | РК | FK | Опис поля                                          |
|-------------|-------------|----|----|----------------------------------------------------|
| id          | int         | +  | –  | Id ордеру                                          |
| amount      | int (11)    | –  | –  | Ціна за квитки (максимальна кількість символів 11) |
| description | text        | –  | –  | Назначення платежу                                 |
| session_id  | int         | –  | +  | Id сеансу                                          |
| user_id     | int         | –  | +  | Id користувача                                     |
| status      | text        | –  | –  | Інформація про статус оплати                       |
| seats       | text        | –  | –  | Придбані місця                                     |
| token       | varchar(64) | –  | –  | Токен оплати (максимальна кількість символів 64)   |
| created_at  | timestamp   | –  | –  | Час створення                                      |
| updated_at  | timestamp   | –  | –  | Час оновлення                                      |

Для збереження даних про користувача, була створена таблиця “users”, табл.2.2. Вона містить інформацію про користувача та данні для входу до особистого кабінету.

Таблиця 2.2

Структура таблиці “users”

| Назва | Тип даних    | РК | FK | Опис поля                                              |
|-------|--------------|----|----|--------------------------------------------------------|
| id    | int          | +  | –  | Id користувача                                         |
| name  | varchar(255) | –  | –  | Ім'я користувача (максимальна кількість символів 255)  |
| email | varchar(255) | –  | –  | Email користувача (максимальна кількість символів 255) |

Продовження таблиці 2.2

| Назва          | Тип даних    | PK | FK | Опис поля                                                            |
|----------------|--------------|----|----|----------------------------------------------------------------------|
| surname        | varchar(255) | —  | —  | Прізвище користувача<br>(максимальна кількість символів 255)         |
| phone_number   | int          | —  | —  | Номер телефону користувача                                           |
| birthday       | date         | —  | —  | Дата народження користувача                                          |
| password       | varchar(255) | —  | —  | Зашифрований пароль користувача (максимальна кількість символів 255) |
| remember_token | varchar(100) | —  | —  | Токен для запам'ятовування користувача                               |
| created_at     | timestamp    | —  | —  | Час створення                                                        |
| updated_at     | timestamp    | —  | —  | Час оновлення                                                        |

“sessions” – містить інформацію про всі сеанси в кінотеатрі, табл.2.3.

Таблиця 2.3

Структура таблиці “sessions”

| Назва     | Тип даних | PK | FK | Опис поля                                     |
|-----------|-----------|----|----|-----------------------------------------------|
| id        | int       | +  | —  | Id сеансу                                     |
| hall_id   | int(11)   | —  | +  | Id ордеру (максимальна кількість символів 11) |
| date_Time | datetime  | —  | —  | Дата початку сеансу                           |
| movie_id  | int       | —  | +  | Id фільму                                     |
| start     | datetime  | —  | —  | Початок сеансу                                |
| end       | datetime  | —  | —  | Кінець сеансу                                 |
| price     | int(11)   | —  | —  | Ціна за одне місце                            |
| occupied  | text      | —  | —  | Вже заброньовані місця іншими користувачами   |

Продовження таблиці 2.3

| Назва      | Тип даних | PK | FK | Опис поля     |
|------------|-----------|----|----|---------------|
| created_at | timestamp | —  | —  | Час створення |
| updated_at | timestamp | —  | —  | Час оновлення |

“movies\_languages” – містить інформацію про всі переклади фільмів в кінотеатрі відповідно до локалізації, табл.2.4.

Таблиця 2.4

Структура таблиці “movies\_languages”

| Назва       | Тип даних    | PK | FK | Опис поля                                                  |
|-------------|--------------|----|----|------------------------------------------------------------|
| id          | int          | +  | —  | Id перекладу                                               |
| title       | varchar(255) | —  | —  | Назва (максимальна кількість символів 255)                 |
| director    | varchar(255) | —  | —  | Режисер (максимальна кількість символів 255)               |
| description | text         | —  | —  | Опис                                                       |
| content     | varchar(255) | —  | —  | Зображення фільму (максимальна кількість символів 255)     |
| language    | varchar(255) | —  | —  | Мова перекладу фільму (максимальна кількість символів 255) |
| production  | varchar(255) | —  | —  | Продакшн (максимальна кількість символів 255)              |
| studio      | varchar(255) | —  | —  | Студія (максимальна кількість символів 255)                |
| movie_id    | int          | —  | +  | Id фільму                                                  |
| language_id | int          | —  | +  | Id локалізації фільму                                      |
| rolling     | tinyint(1)   | —  | —  | Відображення для користувача прокату фільму                |

“movies” – містить інформацію про всі фільми в кінотеатрі, табл.2.5.

Таблиця 2.5

Структура таблиці “movies”

| Назва               | Тип даних    | PK | FK | Опис поля                                                       |
|---------------------|--------------|----|----|-----------------------------------------------------------------|
| id                  | int          | +  | –  | Id сеансу                                                       |
| original_title      | varchar(255) | –  | –  | Оригінальна назва<br>(максимальна кількість символів 255)       |
| age                 | varchar(255) | –  | –  | Вікові обмеження до фільму (максимальна кількість символів 255) |
| year                | date         | –  | –  | Рік випуску                                                     |
| start_rental_period | date         | –  | –  | Початок продажу квитків                                         |
| over_rental_period  | date         | –  | –  | Кінець продажу квитків                                          |
| duration            | text         | –  | –  | Час скільки йде фільм                                           |
| trailer             | text         | –  | –  | Посилання на трейлер фільму                                     |
| rolling             | tinyint(1)   | –  | –  | Відображення для користувача прокату фільму                     |
| created_at          | timestamp    | –  | –  | Час створення                                                   |
| updated_at          | timestamp    | –  | –  | Час оновлення                                                   |

“languages” – містить інформацію про всі локалізації (en, ua), табл.2.6.

Таблиця 2.6

Структура таблиці “languages”

| Назва | Тип даних | PK | FK | Опис поля    |
|-------|-----------|----|----|--------------|
| id    | int       | +  | –  | Id перекладу |

Продовження таблиці 2.6

| Назва  | Тип даних    | РК | FK | Опис поля                                           |
|--------|--------------|----|----|-----------------------------------------------------|
| local  | varchar(255) | –  | –  | Назва локалі (максимальна кількість символів 255)   |
| prefix | varchar(255) | –  | –  | Префікс локалі (максимальна кількість символів 255) |

“cinema\_hall” – таблиця, що містить інформацію про всі зали, які є в кінотеатрі, табл.2.7.

Таблиця 2.7

Структура таблиці “cinema\_hall”

| Назва           | Тип даних    | РК | FK | Опис поля                                                  |
|-----------------|--------------|----|----|------------------------------------------------------------|
| id              | int          | +  | –  | Id залу                                                    |
| name            | varchar(255) | –  | –  | Назва залу (максимальна кількість символів 255)            |
| number_of_seats | int(11)      | –  | –  | Кількість мість в залі (максимальна кількість символів 11) |
| number_of_rows  | int(11)      | –  | –  | Кількість рядів в залі (максимальна кількість символів 11) |
| description     | varchar(255) | –  | –  | Опис залу (максимальна кількість символів 255)             |
| working         | varchar(55)  | –  | –  | Формат показу (максимальна кількість символів 55)          |

“password\_resets” – таблиця, що містить інформацію для скидання паролю, табл.2.8.

Таблиця 2.8

Структура таблиці “password\_resets”

| Назва      | Тип даних    | РК | FK | Опис поля                                                         |
|------------|--------------|----|----|-------------------------------------------------------------------|
| email      | int          | +  | –  | Ідентифікатор пошти                                               |
| token      | varchar(255) | –  | –  | Токен для скидання паролю<br>(максимальна кількість символів 255) |
| created_at | timestamp    | –  | –  | Час створення                                                     |

“migrations” – таблиця, що містить інформацію про міграції, табл.2.9.

Таблиця 2.9

Структура таблиці “migrations”

| Назва     | Тип даних    | РК | FK | Опис поля                                           |
|-----------|--------------|----|----|-----------------------------------------------------|
| id        | int          | +  | –  | Id міграції                                         |
| migration | varchar(255) | –  | –  | Назва міграції (максимальна кількість символів 255) |
| batch     | int(255)     | –  | –  | Час створення                                       |

“personal\_access\_tokens” – таблиця, що містить персональний токен доступу користувача, табл.2.10.

Таблиця 2.10

Структура таблиці “personal\_access\_tokens”

| Назва          | Тип даних    | РК | FK | Опис поля                                                 |
|----------------|--------------|----|----|-----------------------------------------------------------|
| id             | bigint(20)   | +  | –  | Ідентифікатор (максимальна кількість символів 20)         |
| tokenable_type | varchar(255) | –  | –  | Тип моделі токена<br>(максимальна кількість символів 255) |

Продовження таблиці 2.10

| Назва        | Тип даних    | РК | FK | Опис поля                                               |
|--------------|--------------|----|----|---------------------------------------------------------|
| tokenable_id | bigint(20)   | —  | +  | Id Моделі токена<br>(максимальна кількість символів 20) |
| name         | varchar(255) | —  | —  | Назва токена                                            |
| token        | varchar(64)  | —  | —  | Токен                                                   |
| abilities    | text         | —  | —  | Можливості                                              |
| last_used_at | timestamp    | —  | —  | Час останнього використання токена                      |
| expires_at   | timestamp    | —  | —  | Час закінчення токена                                   |
| created_at   | timestamp    | —  | —  | Час створення                                           |
| updated_at   | timestamp    | —  | —  | Час оновлення                                           |

“roles” – таблиця, що містить інформацію про ролі, табл.2.11.

Таблиця 2.11

Структура таблиці “roles”

| Назва      | Тип даних    | РК | FK | Опис поля                                          |
|------------|--------------|----|----|----------------------------------------------------|
| id         | bigint(20)   | +  | —  | Ідентифікатор (максимальна кількість символів 20)  |
| name       | varchar(255) | —  | —  | Назва ролі (максимальна кількість символів 255)    |
| guard_name | varchar(255) | —  | —  | Час створення (максимальна кількість символів 255) |
| created_at | timestamp    | —  | —  | Час створення                                      |
| updated_at | timestamp    | —  | —  | Час оновлення                                      |

“failed\_jobs” – таблиця, що містить інформацію про неуспішні завдання, табл.2.12.

Таблиця 2.12

Структура таблиці “failed\_jobs”

| Назва      | Тип даних    | РК | FK | Опис поля                                           |
|------------|--------------|----|----|-----------------------------------------------------|
| id         | int          | +  | —  | Id міграції                                         |
| uuid       | varchar(255) | —  | —  | Назва міграції (максимальна кількість символів 255) |
| connection | text         | —  | —  | Час створення                                       |
| queue      | text         | —  | —  | Черга                                               |
| payload    | longtext     | —  | —  | Навантаження                                        |
| exception  | longtext     | —  | —  | Виключення                                          |
| failed_at  | timestamp    | —  | —  | Час виникнення помилки                              |

“model\_has\_roles” – таблиця, що використовується для встановлення зв’язку між моделями користувачів та ролями, табл.2.13.

Таблиця 2.13

Структура таблиці “model\_has\_roles”

| Назва      | Тип даних    | РК | FK | Опис поля                                                   |
|------------|--------------|----|----|-------------------------------------------------------------|
| role_id    | bigint(20)   | —  | +  | Id ролі (максимальна кількість символів 20)                 |
| model_type | varchar(255) | —  | —  | Тип моделі користувача (максимальна кількість символів 255) |
| model_id   | bigint(20)   | —  | —  | Id моделі (максимальна кількість символів 255)              |

Переходимо до розробки та реалізації алгоритмів роботи системи замовлення квитків відповідно до запропонованих рішень, зокрема алгоритмів пошуку фільмів та сеансів, оброблення замовлень квитків та платежів, керування резерваціями та скасуваннями, забезпечення безпеки.

## 2.3 Проектування та реалізація алгоритмів роботи системи

Проектування та реалізація алгоритмів роботи системи замовлення квитків включає наступні аспекти:

1. Алгоритм пошуку фільмів та сеансів. Система повинна мати ефективний алгоритм, який дозволяє користувачам швидко знаходити доступні фільми та сеанси. Цей алгоритм може базуватися на параметрах пошуку, як тижневий проміжок пошуку сеансів.

2. Алгоритм обробки замовлень квитків. При отриманні замовлення на квитки система повинна мати алгоритм, який перевіряє наявність вільних місць у залі та резервує їх для користувача. Цей алгоритм також повинен враховувати можливість одночасного замовлення квитків різними користувачами.

3. Алгоритм обробки платежів. При здійсненні оплати за квитки система повинна мати безпечний алгоритм, який забезпечує захист фінансових даних користувачів та підтверджує успішну оплату. Цей алгоритм може використовувати платіжні шлюзи.

4. Алгоритм керування резерваціями та скасуваннями. Система повинна мати алгоритм, який керує резервуванням квитків.

5. Алгоритм забезпечення безпеки. Система повинна мати алгоритми, що забезпечують захист особистих даних користувачів, безпеку транзакцій та запобігання шахрайству. Це може включати шифрування даних та автентифікацію користувачів.

Оскільки обраною архітектурою нашої системи є SPA, тому для повного функціонування системи, ми будемо використовувати RESTful API.

RESTful API [3] (Application Programming Interface) є інтерфейсом, який використовує принципи архітектури REST для створення веб-сервісів, що надають доступ до ресурсів та їх управління.

Основні характеристики RESTful API включають:

– Використання HTTP методів: RESTful API використовує HTTP методи, такі як GET, POST, PUT та DELETE, для взаємодії з ресурсами. Наприклад,

GET використовується для отримання ресурсу, POST – для створення нового ресурсу, PUT – для оновлення ресурсу, DELETE – для його видалення.

- Використання URI для ідентифікації ресурсів: Кожен ресурс має свій унікальний URI (Uniform Resource Identifier), який ідентифікує його.

- Використання стандартних форматів даних: RESTful API може використовувати різні формати даних для представлення ресурсів, такі як JSON (JavaScript Object Notation) або XML (eXtensible Markup Language). JSON є найпоширенішим форматом для передачі даних у RESTful API.

- Незалежність від стану: RESTful API повинен бути безстандартним, що означає, що кожен запит до сервера має містити всю необхідну інформацію для обробки запиту. Сервер не зберігає жодних даних про попередні запити клієнта.

- Кешування: RESTful API може підтримувати кешування для збереження результатів запитів та покращення продуктивності. Клієнти можуть кешувати отримані відповіді і використовувати їх для майбутніх запитів.

На рисунку 2.4 представлено загальний алгоритм роботи системи.

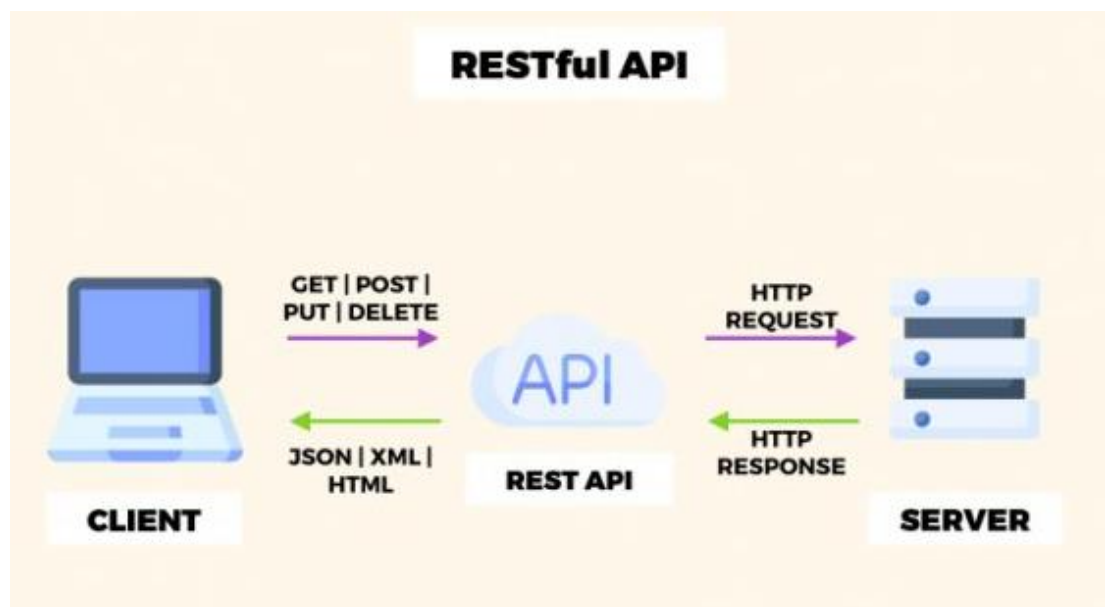


Рисунок 2.4 – Загальний алгоритм роботи системи

Під час згадки алгоритму роботи та можливих станів програмної системи відразу приходить на думку побудова діаграми діяльності (activity diagram).

Діаграма діяльності [11] є важливою діаграмою поведінки для опису динамічних аспектів системи. Діаграма діяльності – це, вдосконалена версія

блок-схеми, яка моделює потік від однієї діяльності до іншої. На рисунку 2.5 зображено діаграму діяльності.

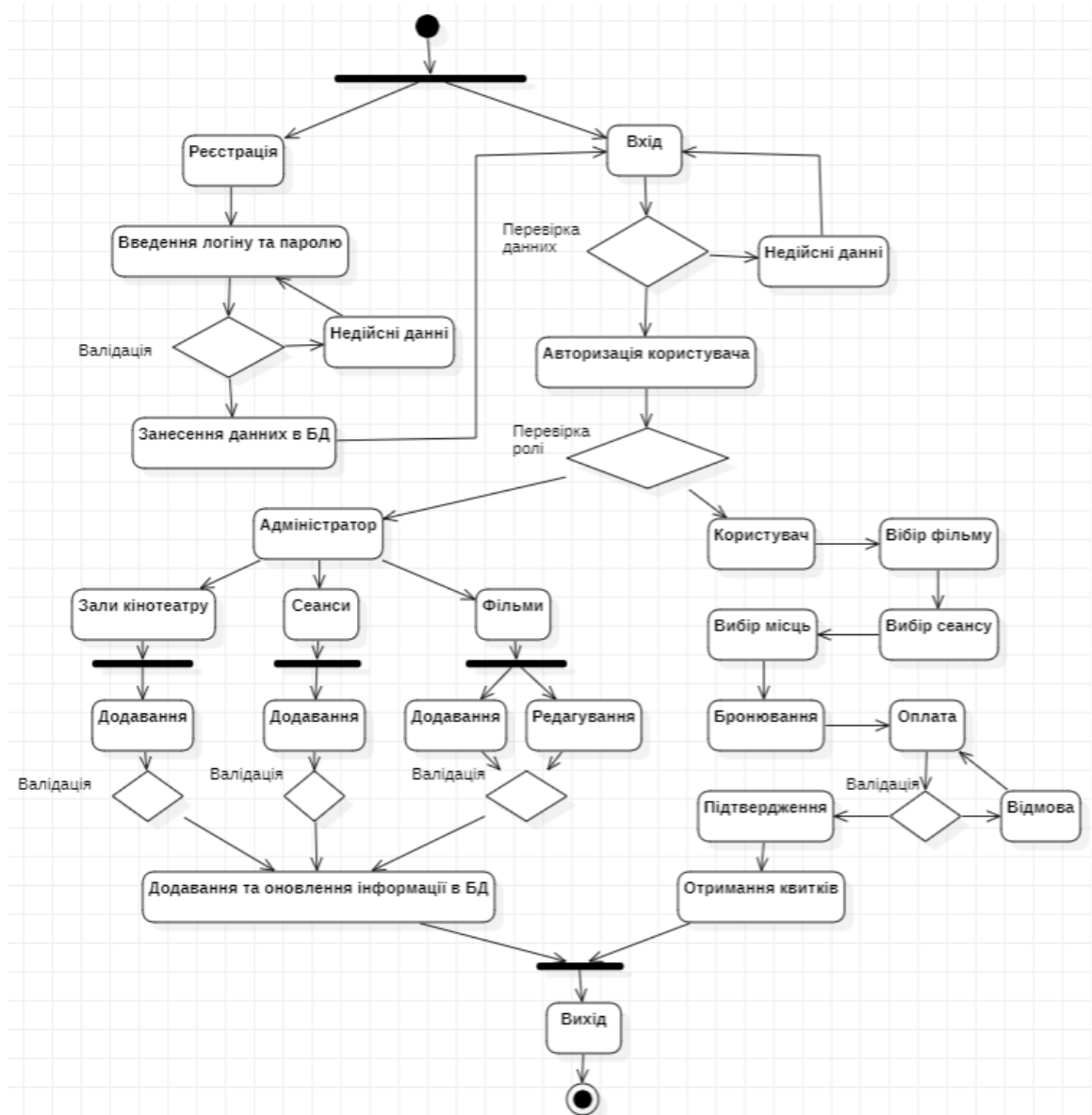


Рисунок 2.5 – Діаграма діяльності

Опис функціоналу (користувача):

1. Користувач (неавторизований) має можливість переглянути доступні фільми та сеанси.
2. Авторизація (Так – перехід до головної сторінки, ні – перехід до сторінки авторизації з повідомленням про помилку).
3. Перегляд доступних фільмів та інформації про них.
4. Замовлення квитків.

– Вибір фільму

- Вибір сеансу
- Вибір місць в залі
- Оплата.

5. Можливість переглянути квитки, які були замовлені раніше.

6. Кінець роботи.

Опис функціоналу (адміністратора):

1. Адміністратор має можливість додати фільм.
2. Відредагувати вже існуючі фільми.
3. Додати залу.
4. Додати сеанс перегляду вже до існуючої зали та фільму.
5. Кінець роботи.

На рисунку 2.6 зображено діаграму компонентів [12].

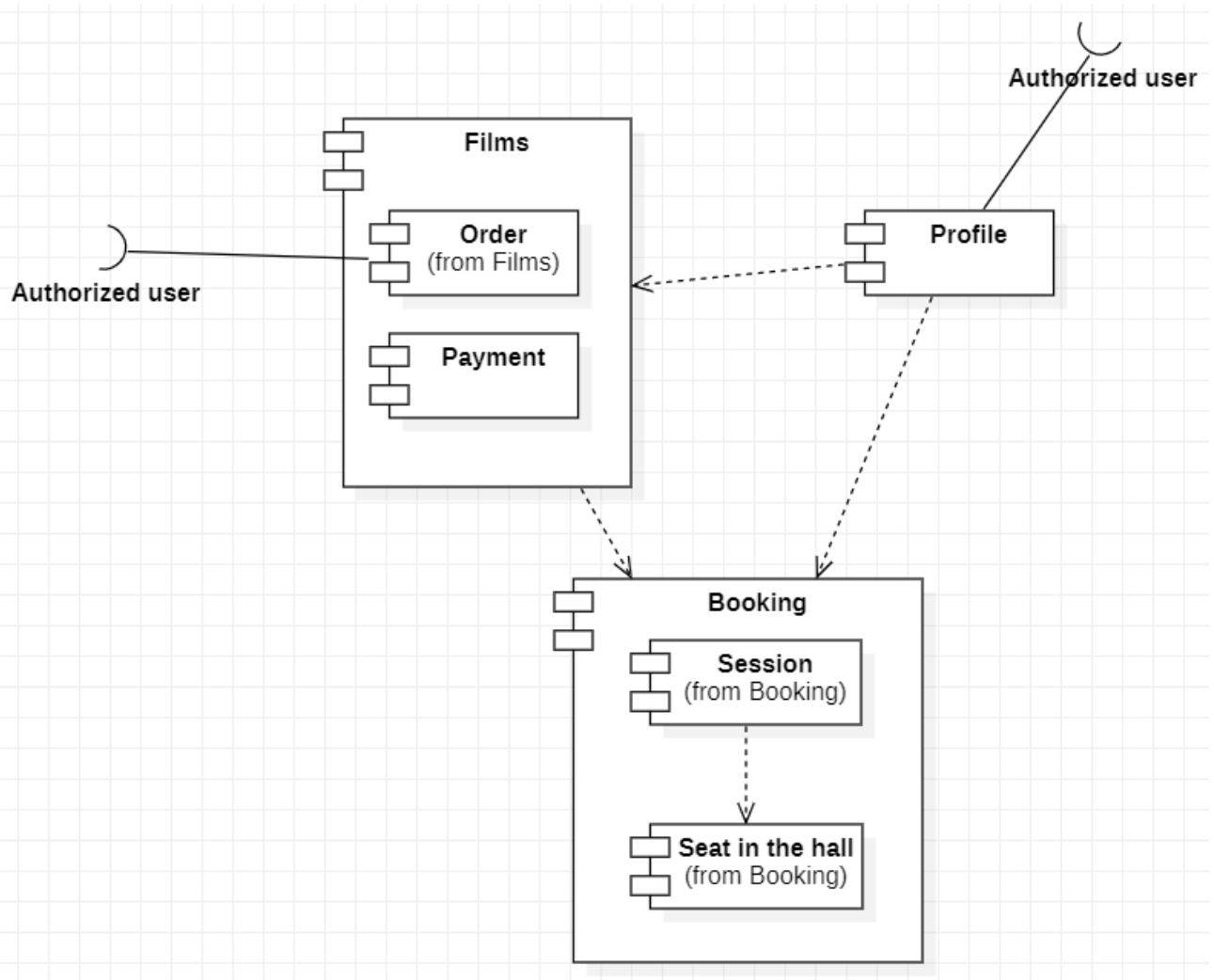


Рисунок 2.6 – Діаграма компонентів

## Аналіз діаграми:

1. Користувач має бути авторизованим для використання повного функціоналу системи.
2. Система складається з одного компонента та його підкомпонента.
3. Підкомпонент Booking відповідає за здійснення оплати для отримання квитків на обраний сеанс.
4. Підсистема Films відповідає за обраний фільм та його інформацію.
5. Компонент Profile Відповідає за профіль користувача та відображає всі квитки з особистою інформацією клієнта.

Розглянемо алгоритм оплати liqpay. Для більшої наочності було побудовано діаграму. На рисунку 2.7 зображено діаграму послідовності [13] liqpay.

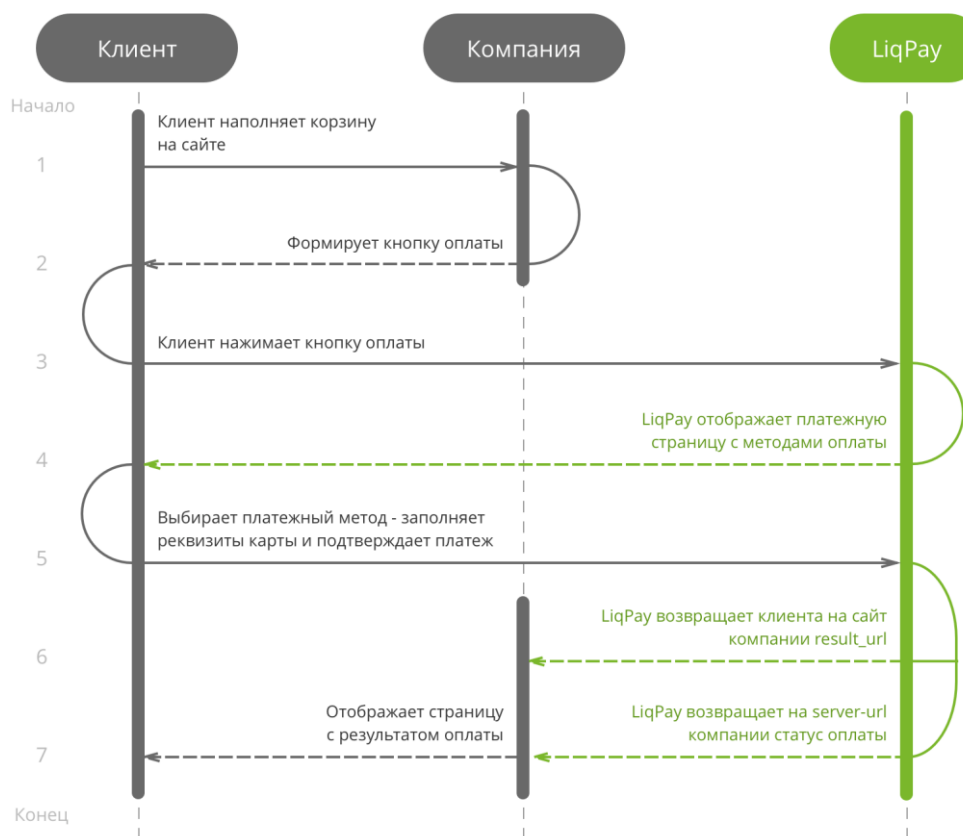


Рисунок 2.7 – Діаграма послідовності liqpay

Після того, як користувач (авторизований) обрав фільм та сеанс, він має можливість обрати місця в залі та придбати квитки. Після вибору місць, користувач переходить до процесу оплати. Після успішної оплати, статус його замовлення оновлюється в базі даних, підтверджуючи його придбання. Після

завершення оплати, користувач отримує можливість завантажити pdf-файл квитка зі свого особистого кабінету на веб-сайті. Також є можливість переглянути квитки, які вже були придбані користувачем раніше. Функція перегляду квитків дозволяє користувачам мати повний контроль над своїми попередніми покупками, забезпечуючи зручний спосіб перегляду інформації про придбані квитки. Цей квиток містить важливу інформацію, таку як назву фільму, дату та час сеансу, номери обраних місць та інші деталі, які допомагають користувачу зручно здійснити вхід у кінозал на вказаний сеанс.

## 2.4 Реалізація онлайн-сервісу замовлення квитків та послуг “Кінотеатр”

Для замовлення квитків, користувач повинен бути зареєстрованим. Тому розглянемо реалізацію реєстрації. Метод обробляє запит на реєстрацію нового користувача. Він створює новий об'єкт користувача в базі даних з введеними даними, хешує пароль, створює токен доступу для автентифікації, призначає роль "user" та повертає успішну відповідь з токеном і даними користувача.

На рисунку 2.8 зображено метод register, що відповідає за реєстрацію нового користувача.

```
public function Register(RegisterRequest $request)
{
    try {
        $user = User::create([
            'name' => $request->name,
            'surname' => $request->surname,
            'email' => $request->email,
            'password' => Hash::make($request->password),
        ]);
        $token = $user->createToken('app')->plainTextToken;
        $user->assignRole('user');

        return $this->success('Registration Successfull', $token, $user);
    } catch (\Throwable $exception) {
        return $this->error($exception->getMessage());
    }
    return response()->json([
        'message' => 'Invalid Email Or Password'
    ], 401);
}
```

Рисунок 2.8 – Метод register, що відповідає за реєстрацію користувача класу(AuthRepository)

Наступний метод обробляє запит на скидання пароля користувача. Він перевіряє, чи існує користувач з вказаною електронною поштою, генерує випадковий токен для скидання пароля, зберігає цей токен разом з електронною поштою користувача в базі даних, та відправляє листа на вказану електронну пошту. Користувач має можливість перейти за вказаним посиланням та перейти на сторінку скидання паролі. Після відправлення форми відбувається оновлення паролі в БД. Цей процес забезпечує безпеку та контроль над скиданням паролі користувачів, забезпечуючи можливість швидкого та надійного відновлення доступу до своїх облікових записів.

На рисунку 2.9 зображено метод `resetPassword`, що відповідає за зміну пароля. Він відповідає за зміну пароля користувача в системі, надає безпечний та контрольований спосіб зміни пароля для забезпечення безпеки облікового запису користувача.

```
public function ForgetPassword(ForgetRequest $request)
{
    $lang = $request->lang;
    $email = $request->email;

    if(User::where('email', $email)->doesntExist()) {
        return response([
            'message' => 'Email Invalid'
        ], 401);
    }

    $token = rand(10,1000000);

    try {
        DB::table('password_resets')->insert([
            //'lang' => $lang,
            'email' => $email,
            'token' => $token
        ]);

        Mail::to($email)->send(new ForgetMail($lang, $token, $email));

        return response([
            'message' => 'Reset Password Mail send on your email'
        ],200);
    }catch (Exception $exception) {
        return response([
            'message' => $exception->getMessage()
        ], 400);
    }
}
```

Рисунок 2.9 – Метод `resetPassword`, що відповідає за зміну пароля класу(`ForgetRepository`)

Наступний метод обробляє запит на додавання нової сесії до системи з інформацією про зал та фільм. Він перевіряє роль користувача (чи він має роль "admin"), перевіряє наявність сесії в залі і повертає відповідну відповідь. Після перевірки ролі користувача, метод перевіряє наявність сесії в залі. Цей метод забезпечує контроль та безпеку при додаванні нових сеансів до системи, дозволяючи тільки адміністраторам з необхідними правами виконувати цю операцію та уникати конфліктів з існуючими сесіями. На рисунку 2.10 зображено метод AddSession, що відповідає за додавання сесії.

```
public function AddSession(Request $request): \Illuminate\Http\JsonResponse
{
    try {
        $user = Auth::user();
        $role = $user->hasRole('admin');//role check
        $hall = Hall::where('id', $request[4])->first();
        $movie_id = Movies::where('id', $request[3])->first();
        $sessions = Sessions::where([
            'hall_id' => $hall->id,
            'start' => date('Y-m-d H:i', strtotime($request[1])),
        ]->first());
        if ($sessions) {
            return response()->json([
                'message' => "The session currently already exists"
            ], 400);
        }
        elseif ($role) {
            $session = Sessions::create([
                'hall_id' => $hall->id,
                'original_title' => $movie_id->original_title,
                'date_Time' => date('Y-m-d', strtotime($request[1])),
                'movie_id' => $movie_id->id,
                'start' => date('Y-m-d H:i', strtotime($request[1])),
                'end' => date('Y-m-d H:i', strtotime($request[2])),
                'price' => $request[5],
            ]);
            return response()->json([
                'hall' => $hall,
                'movie_id' => $movie_id,
                'session' => $session,
            ], 200);
        } else {
            return response()->json([
                'message' => "Not Admin"
            ], 404);
        }
    } catch (Exception $exception) {
        return response()->json([
            'message' => $exception->getMessage()
        ], 400);
    }
}
```

Рисунок 2.10 – Метод AddSession, що відповідає за додавання сесії класу(SessionRepository)

Цей метод повертає інформацію про фільм та його сеанси на основі переданих параметрів запиту, таких як ID фільму, мова та період часу.

Наступний метод Виводить інформацію за обраним проміжком часу. Цей метод дозволяє користувачам зосередитися на конкретному періоді, в якому вони зацікавлені, і отримати відповідну інформацію без необхідності переглядати весь розклад або доступні дані.

На рисунку 2.11 зображено метод CurrentMovie, що відповідає за виведення потрібних сеансів виходячи з зазначеного часового проміжку обраного користувачем.

```
public function CurrentMovie(Request $request)//$lang, $movie_id
{
    try {
        $m = Movies::where('id', $request->movie_id)->first();
        $languages = Language::where('prefix', $request->lang)->first();
        $movie = MoviesLanguages::where('language_id', $languages->id)-
>where('movie_id', $request->movie_id)->first();
        $locale = app()->getLocale();

        $sessions = Sessions::whereDate('start', '=', date("Y-m-d"))->get();
        if ($request->period === 'Today') {
            $sessions = Sessions::whereDate('start', '=', date("Y-m-d"))
->whereDate('start', '>', date("Y-m-d H:i"))->get();
        } elseif ($request->period === 'Tomorrow') {
            $sessions = Sessions::whereDate('start', '=', date("Y-m-d",
strtotime("+1 day")))->where('movie_id', $request->movie_id)->get();
        } elseif ($request->period === 'Week') {
            $sessions = Sessions::whereDate('start', '>=', date("Y-m-d",
strtotime("+1 day")))->whereDate('start', '<=', date("Y-m-d", strtotime("+1
week")))->where('movie_id', $request->movie_id)->get();
        } elseif ($request->period === 'Month') {
            $sessions = Sessions::whereDate('start', '>=', date("Y-m-d",
strtotime("+1 day")))->whereDate('start', '<=', date("Y-m-d", strtotime("+1
month")))->where('movie_id', $request->movie_id)->get();
        }

        if(count($sessions) == '0') {
            return response()->json([
                'movie' => $movie,
                'm' => $m,
                'sessions' => $sessions,
                'message' => count($sessions) == '0', //'There are no upcoming
sessions.',
            ], 200); //State Code;
        }
    }
}
```

Рисунок 2.11 – Метод CurrentMovie, що відповідає виведення потрібних сеансів класу(MovieRepository)

Система перевіряє актуальну інформацію про фільми, доступні для перегляду та виводить для користувача. Це спрощує пошук та полегшує вибір потрібного сеансу. Користувачам не потрібно перевіряти кілька джерел

інформації або контролювати оновлення даних про фільми, оскільки система автоматично надає їм актуальну інформацію. Вони можуть бачити назви фільмів, графік сеансів, тривалість та інші важливі деталі, що допомагають їм зробити інформований вибір.

### **Висновки до другого розділу**

1. Проведено аналіз потенційних сценаріїв використання системи та встановлено основні цілі її функціонування. Об'єктно-орієнтована структура системи була розроблена з урахуванням потреб користувачів та основних функцій сервісу.

2. Розроблено структуру бази даних, що включає необхідні таблиці та зв'язки для зберігання інформації про фільми, сеанси, квитки, користувачів та інші важливі дані.

3. Проведено проектування та реалізацію алгоритмів роботи системи, що включають процеси пошуку сеансів, обробку замовлень квитків, резервацію місць та інші важливі функції.

4. Здійснено реалізацію самого онлайн-сервісу замовлення квитків та послуг "Кінотеатр", де враховано всі попередні етапи та було створено зручний та функціональний інтерфейс для користувачів.

## РОЗДІЛ 3 ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ОНЛАЙН-СЕРВІСУ ЗАМОВЛЕННЯ КВИТКІВ ТА ДОДАТКОВИХ ПОСЛУГ “КІНОТЕАТР”

### 3.1 Порядок встановлення та налаштування параметрів системи

Для більш наочного розгорнення існує діаграма розгортання. Діаграма розгортання (Deployment diagram) [14] – це діаграма, яка відображає фізичну архітектуру системи, розташування її компонентів та зв'язків між ними. Вона показує, як програмні компоненти розгортаються на апаратному забезпеченні, такому як сервери, комп'ютери або мобільні пристрої. Вона допомагає зрозуміти, як фізичні компоненти системи взаємодіють між собою та як вони розташовані. Діаграма є важливим інструментом для планування та аналізу інфраструктури системи перед її розгортанням.

На рис. 3.1 зображено діаграму розгортання.

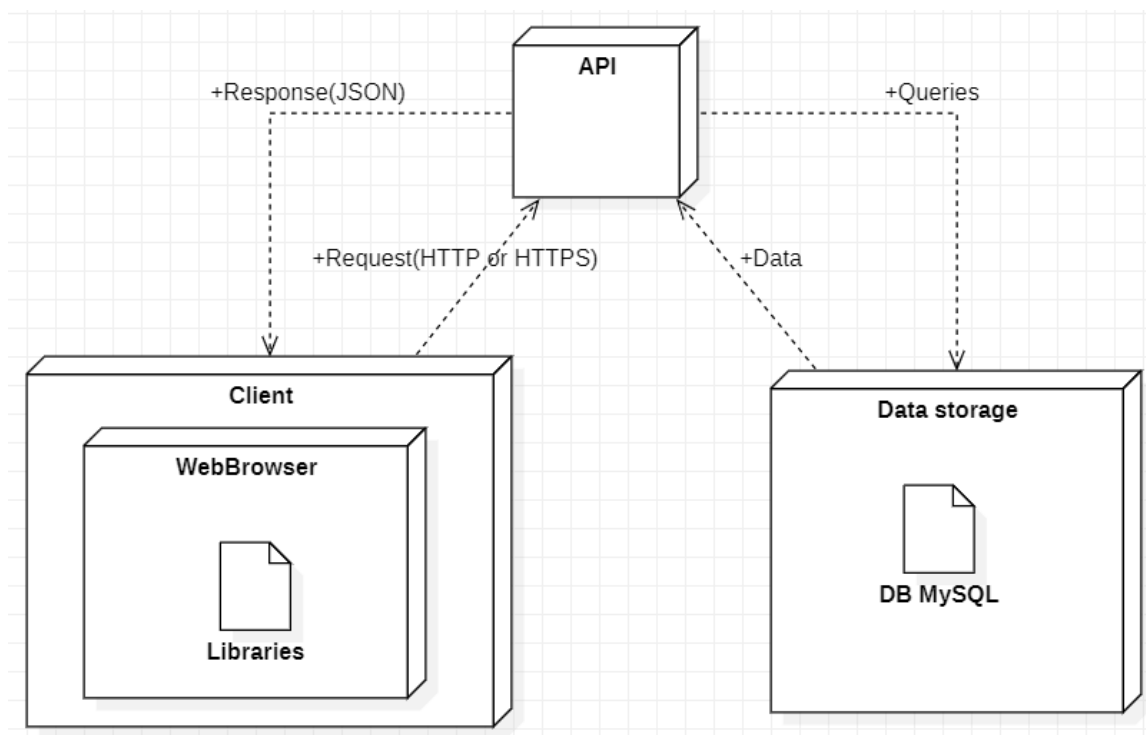


Рисунок 3.1 – Діаграма розгортання

Проаналізувавши дану діаграму можемо зробити наступні висновки:

- Бекенд частина звертається до сервера для отримання необхідних ресурсів.

- При запуску додатку клієнт отримує початкову HTML-розмітку, а після цього оновлює її за допомогою асинхронних звернень до API.
- Веб-застосунок має бекенд частину, яка розгортається на порті 8000, тоді як клієнтська частина працює на порті 3000.
- Бекенд частина звертається до сервера для отримання необхідних ресурсів.

Ця архітектура дозволяє розділити бекенд і клієнтську частини веб-застосунку, забезпечуючи їх незалежну розгортку та масштабування. Бекенд частина забезпечує доступ до ресурсів і виконує бізнес-логіку, а клієнтська частина відповідає за візуальне представлення та взаємодію з користувачем. Звернення до API дозволяє клієнтській частині оновлювати дані асинхронно, підтримуючи динамічне оновлення сторінок та забезпечуючи зручний користувацький досвід.

### 3.2 Структура інтерфейсу та порядок взаємодії з онлайн-сервісом

Коли користувач переходить на сторінку нашого застосунку, його зустрічає головна сторінка з фільмами, які є в прокаті. На рис. 3.2 зображено головну сторінку.

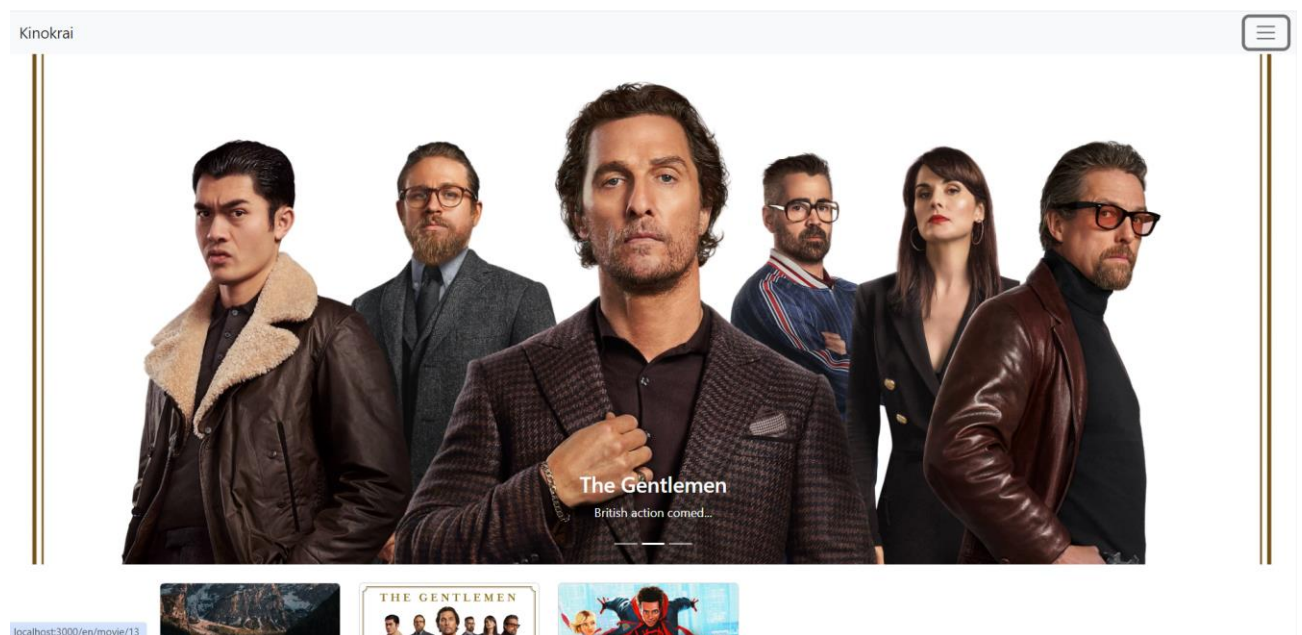


Рисунок 3.2 – Головна сторінка

Користувач має можливість зробити авторизацію, зареєструватись або ознайомитись з усіма доступними фільмами в прокаті. На рис. 3.3 зображено вікно реєстрації.

Kinokrai

Register

First Name \*

Last Name \*

Email Address \*

Password \*

Password confirmation \*

REGISTER

[Already have an account? Sign in](#)

Copyright © Cinema 2023.

Рисунок 3.3 – Вікно реєстрації

Користувач може зробити авторизацію на сайті ввівши інформацію у відповідні поля. На рис. 3.4 зображено вікно авторизації.

Kinokrai

Sign in

Email \*

kyl.lexa@gmail.com

Password \*

\*\*\*\*\*

SIGN IN

[Forgot password](#) [Register](#)

Copyright © Cinema 2023.

Рисунок 3.4 – Вікно авторизації

Після вибору фільму, користувач може обрати доступний сеанс для перегляду за зручним часом. На рис. 3.5 зображено вікно вибору сеансу.

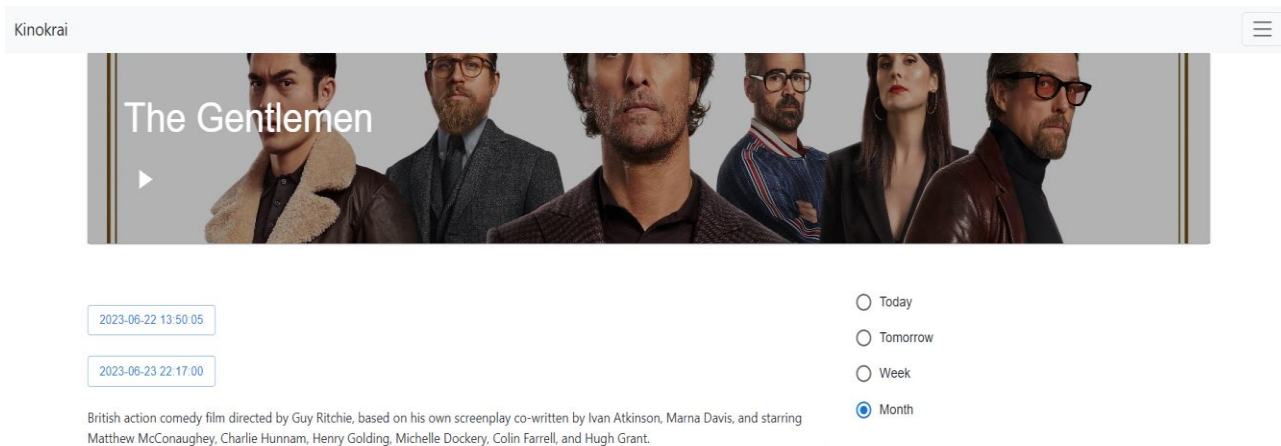


Рисунок 3.5 – Вікно авторизації

Також для більшого ознайомлення користувач може переглянути трейлер. На рис. 3.6 зображено вікно перегляду трейлера.

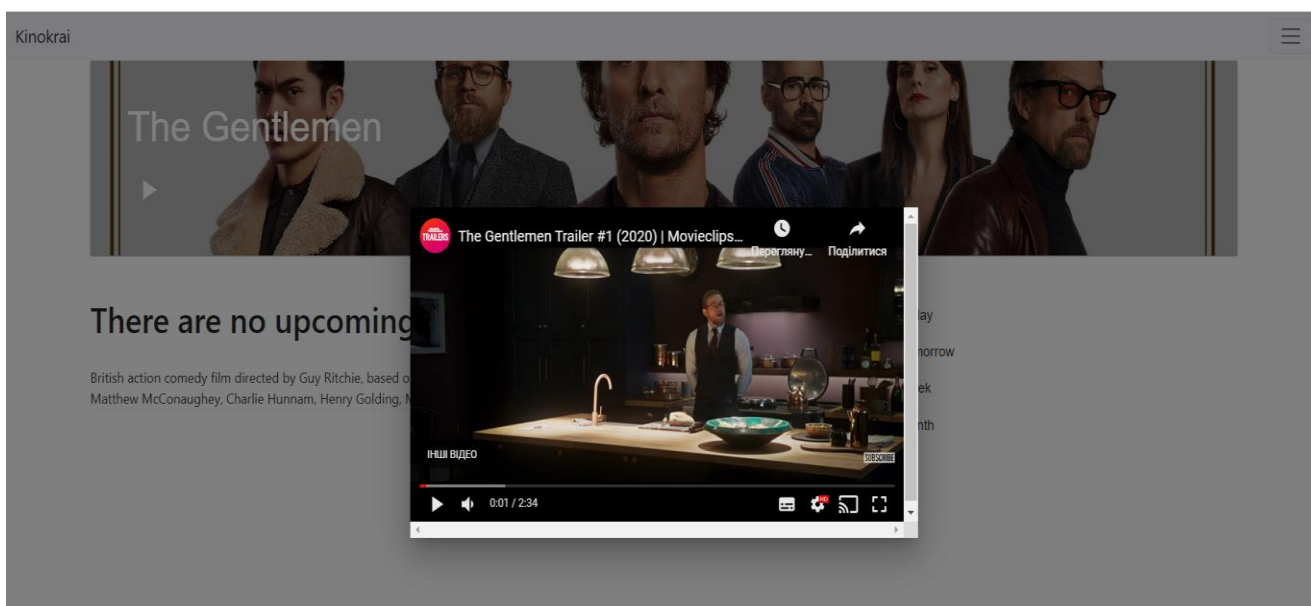


Рисунок 3.6 – Вікно перегляду трейлера

Після вибору фільму користувач може обрати доступне місце, після чого перейти до оплати. На рис. 3.7 зображено вікно вибору місць. Він може вибрати бажане місце, натиснувши на нього кліком миші.

Обране місце буде виділене окремим кольором або підсвічуватися, щоб показати, що воно вибране користувачем, після чого в додатковому полі буде додано інформацію про місце та ціну за квиток. Також у відповідному полі буде виведена інформація про кількість обраних місць та їх загальну ціну.

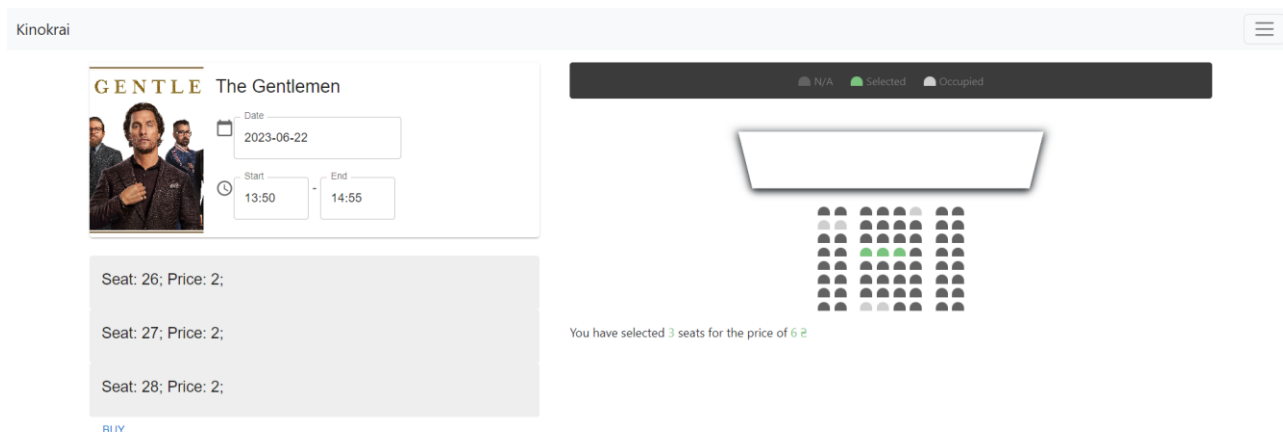


Рисунок 3.7 – Вікно вибору місць

Після вибору фільму користувач може обрати доступне місце в залі, після чого перейти до оплати. Це дозволяє користувачеві вибрати саме те місце, яке найбільше відповідає його вимогам та комфорту. Після того, як користувач обрав бажане місце, він може перейти до оплати, для введення необхідних платіжних даних та завершити процес бронювання квитків. Після успішної оплати, він отримує підтвердження про придбання та може отримати доступ до квитків для друку або збереження у форматі pdf. На рис. 3.8 зображено вікно вибору місць.

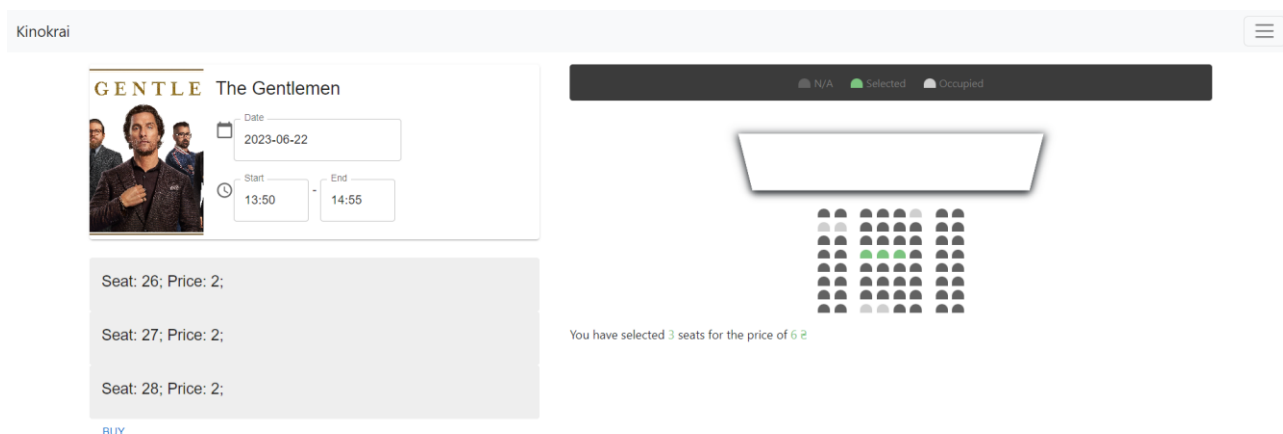


Рисунок 3.8 – Вікно вибору місць

Якщо користувач є адміністратором, його переносить на вікно адмін-панелі, де він може відслідковувати наявні фільми в прокаті, додавати та редагувати їх. Також адміністратор може вивести фільм з прокату натиснувши відповідну кнопку. На рис. 3.9 зображено вікно адміністратора.

|           |  |
|-----------|--|
| Movies    |  |
| Movies    |  |
| Add movie |  |
| Halls     |  |
| Halls     |  |
| Add hall  |  |
| Users     |  |
| Users     |  |

| ID | Title                   | Director        | Description                  | Content | Language  | Duration | Production | Studio |
|----|-------------------------|-----------------|------------------------------|---------|-----------|----------|------------|--------|
| 7  | Avatar EN               | Avatar EN       | Avatar ENAvatar ENAv...      |         | Ukrainian |          |            |        |
| 7  | Avatar UA               | Avatar UA       | Avatar UAAvatar UAAv...      |         | Ukrainian |          |            |        |
| 13 | The Gentlemen           | Guy Ritchie     | British action comedy fil... |         | Ukrainian |          |            |        |
| 13 | Джентльмени             | Джентльмени     | <p><em><u>Британсь...        |         | Ukrainian |          |            |        |
| 18 | Людина-паук: Крізь ...  | Хоакім Дос С... | комп'ютерно-анімаційн...     |         | Ukrainian |          |            |        |
| 18 | Spider-Man: Across t... | Хоакім Дос С... | is a 2023 computer-ani...    |         | English   |          |            |        |
| 21 | Спайд                   | Пітер Сон       | <p><span style="backg...     |         | Ukrainian |          |            |        |

Рисунок 3.9 – Вікно адміністратора

У адміністратора є можливість додати фільм заповнивши відповідні поля. Якщо поле не буде заповненим, система відобразить попередження, про необхідність додати інформацію. Якщо інформація пройшла валідацію, тоді відповідна інформація зберігається в БД. На рис. 3.10 зображено додавання фільму.

|           |  |
|-----------|--|
| Movies    |  |
| Movies    |  |
| Add movie |  |
| Halls     |  |
| Halls     |  |
| Add hall  |  |
| Users     |  |
| Users     |  |

New Movie

All movies

Localization

☒

true

Title

Original Title

Director

Content

Year

Enter title

Enter Original Title

Enter Director

Выбрать файлы

Ф...ан

MM / DD / YYYY

Description

Normal

B

I

U

Start rental period

Over rental period

Age

Language

Duration (Час)

MM / DD / YYYY

MM / DD / YYYY

hh:mm:ss

Trailer

Enter Trailer

Submit

Рисунок 3.10 – Вікно додавання фільму

Вибравши необхідний фільм, адміністратор може редагувати інформацію про фільм в залежності, на якій мові він бажає її змінити. На рис. 3.11 зображено вікно редагування фільму.

Рисунок 3.11 – Вікно редагування фільму

Перейшовши до вікна з залами перед адміністратором з’являться всі зали які наявні в кінотеатрі. На сторінці відображається коротка інформація про зал, його назва та кількість місць. На рис. 3.12 зображено вікно з залами.

Рисунок 3.12 – Вікно з залами

Адміністратор може додати сеанс до обраної зали. Під час додавання нового сеансу, адміністратор обирає відповідний фільм, який буде під час цього сеансу. Він також встановлює ціну за місце для даного сеансу, що дозволяє гнучко налаштувати вартість квитків в залежності від різних факторів, наприклад, популярності фільму або часу показу. На рис. 3.13 зображено вікно додавання сеансу до відповідної зали.

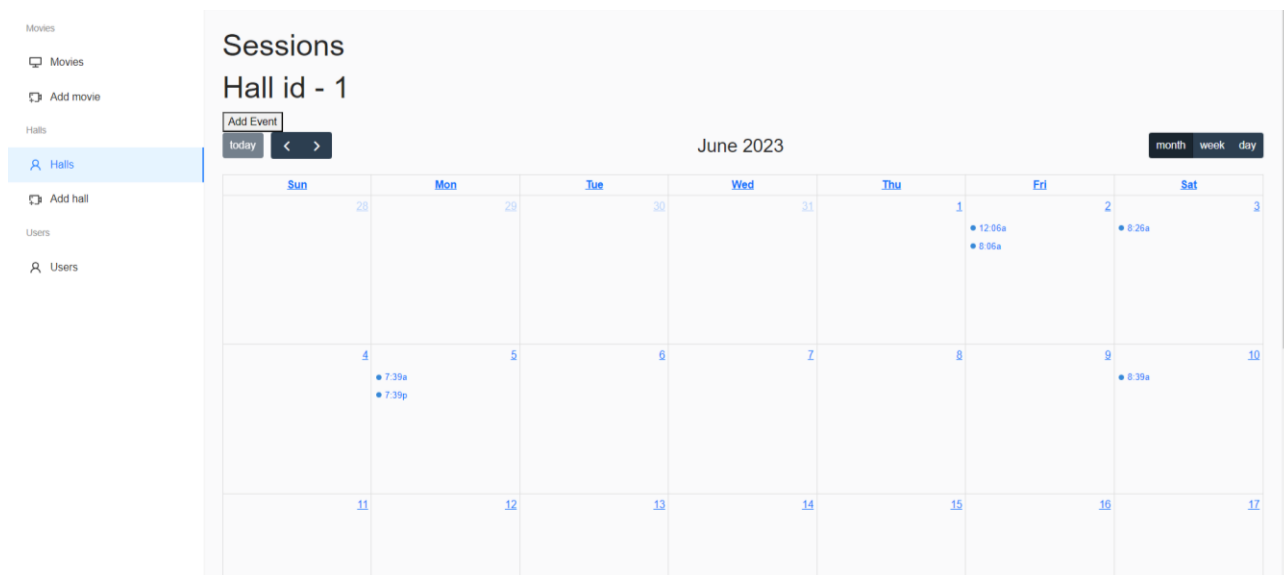


Рисунок 3.13 – Вікно додавання сеансу до відповідної зали

Адміністратор системи квиткового бронювання має можливість додавати нові зали до системи, надаючи їм унікальні імена та вказуючи кількість місць, що доступні у кожному залі.

Рисунок 3.14 – Вікно додавання зали

Крім того, адміністратор може додати короткий опис для кожного залу, що допомагатиме користувачам отримати загальне уявлення про характеристики та особливості цього залу.

### 3.3 Тестування роботи програмного продукту

Тестування продукту є важливою частиною розробки програмного забезпечення. Воно дозволяє перевірити правильність реалізації функціональності, ідентифікувати помилки та проблеми, а також забезпечити високу якість продукту перед його випуском.

Воно є неперервним процесом, який виконується на різних етапах розробки та після випуску продукту. Воно допомагає забезпечити якість та надійність продукту, знизити ризики виникнення помилок та недоліків, а також покращити задоволеність користувачів.

Таблиця 3.1

Test Case таблиця

| Test Case Description                         | Test Steps                                                                                                                                                  | Test Data                                                                                                                                                                                                                                                                                    | Expected Results                        | Actual Results   | Pass/Fail |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----------------------------------------|------------------|-----------|
| Перевірка авторизації з валідними даними      | 1. URL: <a href="http://localhost:3000/en/login">http://localhost:3000/en/login</a><br>2. Введіть email<br>3. Введіть пароль<br>4. Натисніть "SIGN IN"      | Email: <a href="mailto:kyl.lexa1@gmail.com">kyl.lexa1@gmail.com</a><br>Пароль: lexa1lexa1                                                                                                                                                                                                    | Користувач повинен увійти до системи    | Як і очікувалось | Pass      |
| Перевірка авторизації з недійсними даними     | 1. URL: <a href="http://localhost:3000/en/login">http://localhost:3000/en/login</a><br>2. Введіть email<br>3. Введіть пароль<br>4. Натисніть "SIGN IN"      | Email: <a href="mailto:kyl.lexa1@gmail.com">kyl.lexa1@gmail.com</a><br>Пароль: lexa1234567                                                                                                                                                                                                   | Користувач не повинен увійти до системи | Як і очікувалось | Pass      |
| Перевірка додавання фільму з валідними даними | 1. Увійти до системи у ролі адміністратора<br>2. Перейти до вкладки "Add movie"<br>3. Заповнити відповідні поля інформацією<br>4. Натисніть кнопку "Submit" | New Movie: true<br>Localization: Ukrainian<br>Title: Джентельмени<br>Original title: The Gentlemen<br>Content: фото<br>Year: 05/31/23<br>Description: британський...<br>Start rental period: 06/11/23<br>Over rental period: 06/29/23<br>Age: 18+<br>Language: Ukrainian<br>Trailer: http... | Повинен додаться фільм до БД            | Як і очікувалось | Pass      |

Продовження таблиці 3.1

| Test Case Description                         | Test Steps                                                                                                                                                                                                                                                                                                                                                                                                        | Test Data                                                                                                | Expected Results                       | Actual Results   | Pass/Fail |
|-----------------------------------------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|----------------------------------------------------------------------------------------------------------|----------------------------------------|------------------|-----------|
| Перевірка додавання сеансу з валідними даними | <ol style="list-style-type: none"> <li>1. Увійти до системи у ролі адміністратора</li> <li>2. Перейти до вкладки "Halls"</li> <li>3. Обрати бажаний зал для додавання сесії</li> <li>4. Натиснути "Add event"</li> <li>5. Заповнити необхідну інформацію</li> <li>6. Натиснути "Add event"</li> </ol>                                                                                                             | Select film: The Gentlemen<br>Price: 3<br>Start Date: 06/18/2023 7:50 pm<br>End Date: 06/18/2023 9:50 pm | Повинен додатись сеанс до БД           | Як і очікувалось | Pass      |
| Перевірка додавання зали                      | <ol style="list-style-type: none"> <li>1. Увійти до системи у ролі адміністратора</li> <li>2. Перейти до вкладки "Add Hall"</li> <li>3. Заповнити відповідні поля інформацією</li> <li>4. Натиснути кнопку "Submit"</li> </ol>                                                                                                                                                                                    | Name: Тарас Шевченко<br>Count of seats: 64<br>Description: Зручна зала...                                | Повиненна додатись зала до БД          | Як і очікувалось | Pass      |
| Перевірка зміни паролю користувача            | <ol style="list-style-type: none"> <li>1. URL:</li> <li>2. Перейти до сторіки входу</li> <li>3. Натиснути посилання "Forget password"</li> <li>4. Ввести пошту за якою було закріплено акаунт</li> <li>5. Натиснути кнопку "Submit"</li> <li>6. Увійти до пошти</li> <li>7. Перейти за унікальним посиланням яке прийде на пошту</li> <li>8. Ввести новий пароль</li> <li>9. Натиснути кнопку "Submit"</li> </ol> | Email: <a href="mailto:kyl.lexa1@gmail.com">kyl.lexa1@gmail.com</a><br>Пароль: lexa1 lexa1               | Повиненна змінитись пароль користувача | Як і очікувалось | Pass      |

Мета тестування полягає в перевірці продукту з метою забезпечення його якості, виявлення помилок та проблем, а також підтвердження відповідності вимогам та очікуванням.

### Висновки до третього розділу

1. Описано порядок встановлення та налаштування параметрів системи, що є необхідним для правильної роботи продукту.

2. Розглянуто структуру інтерфейсу та порядок взаємодії з онлайн-сервісом. Зазначено, що клієнтська частина представляє собою додаток, що використовує бібліотеки для побудови користувацького інтерфейсу та

налаштування логіки оновлення даних та сторінок. Користувач, який є адміністратором, переноситься на вікно адмін-панелі для виконання відповідних функцій.

3. Наведено процес тестування продукту. Тестування є важливою частиною розробки програмного забезпечення і має на меті перевірити правильність реалізації функціональності, виявити помилки та проблеми, а також забезпечити високу якість продукту перед його випуском. Воно є неперервним процесом, що виконується на різних етапах розробки та після випуску продукту.

4. Представлено порядок правильного встановлення та налаштування параметрів системи.

## ВИСНОВКИ

В рамках виконання випускної роботи бакалавра на тему "Онлайн-сервіс замовлення квитків та послуг "Кінотеатр"" була проведена детальна розробка та реалізація веб-орієнтованої системи для зручного та ефективного замовлення квитків.

Проведений аналіз аналогів програмного продукту показав, що існуючі рішення, такі як Multiplex та Planetakino, мають свої обмеження та недоліки, які можуть бути вирішені та удосконалені у розробленій системі.

Обґрунтування актуальності теми роботи показало, що у сучасному світі зростає популярність онлайн-сервісів та електронної комерції, в тому числі і у сфері культурних подій, таких як кінотеатри. Розробка зручної та функціональної системи для замовлення квитків має великий потенціал для задоволення потреб користувачів та покращення їх взаємодії з кінотеатром.

Вибір архітектури веб-орієнтованої системи було здійснено з використанням підходу SPA (Single-Page Application), що дозволяє забезпечити швидку та плавну роботу системи без перезавантаження сторінок. Ця архітектура дозволяє покращити користувацький досвід та забезпечити більш ефективне використання системи.

Проектування та реалізація алгоритмів роботи системи передбачає розробку алгоритмів пошуку фільмів та сеансів, обробки замовлень квитків, керування резерваціями та скасуваннями, а також забезпечення безпеки даних та запобігання шахрайству. Реалізація цих алгоритмів дозволить створити функціональну та надійну систему, яка забезпечить зручність та задоволення користувачів.

Отже, виконання випускної роботи бакалавра на тему "Онлайн сервіс замовлення квитків та послуг "Кінотеатр"" виявилось актуальним та цікавим завданням, що дозволило розробити та реалізувати функціональну веб-орієнтовану систему для зручного та ефективного замовлення квитків.

Результати цієї роботи можуть бути корисними для подальшого розвитку та вдосконалення подібних систем у сфері розваг та культури.

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Single-page application [Електронний ресурс] – Режим доступу до ресурсу: [https://en.wikipedia.org/wiki/Single-page\\_application](https://en.wikipedia.org/wiki/Single-page_application).
2. Документація JSON [Електронний ресурс] – Режим доступу до ресурсу: <http://json.com/>.
3. Laravel API — Repository Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.devgenius.io/laravel-api-repository-pattern-make-your-code-more-structured-the-simple-guide-5b770da766d7>.
4. Use Case Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/>.
5. Class Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>.
6. Патерни проектування [Електронний ресурс] – Режим доступу до ресурсу: <https://refactoring.guru/uk/design-patterns/what-is-pattern>.
7. Реалізація патерну "Репозиторій" [Електронний ресурс] – Режим доступу до ресурсу: <https://blog.devgenius.io/laravel-api-repository-pattern-make-your-code-more-structured-the-simple-guide-5b770da766d7>.
8. Патерн Unit Of Work [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@rahulrulz680/unit-of-work-in-repository-pattern-b48a862d06a2>.
9. Документація React JS [Електронний ресурс] – Режим доступу до ресурсу: <https://uk.reactjs.org/>.
10. Документація MySQL [Електронний ресурс] – Режим доступу до ресурсу: <https://www.mysql.com/>.
11. Activity Diagram [Електронний ресурс] – Режим доступу до ресурсу: <https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-activity-diagram/>.

12. Component Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/>.
13. Sequence Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/>.
14. Deployment Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-deployment-diagram/>.
15. UML для бізнес-моделювання: для чого потрібні діаграми процесів [Електронний ресурс] – Режим доступу до ресурсу:  
<https://evergreens.com.ua/ua/articles/uml-diagrams.html>.
16. Онлайн-платформа Udemу [Електронний ресурс] – Режим доступу до ресурсу: <https://ua.udemy.com/>.
17. Class Diagram [Електронний ресурс] – Режим доступу до ресурсу:  
<https://www.visual-paradigm.com/guide/uml-unified-modeling-language/uml-class-diagram-tutorial/>.
18. Repository Design Pattern [Електронний ресурс] – Режим доступу до ресурсу: <https://medium.com/@pererikbergman/repository-design-pattern-e28c0f3e4a30>.
19. Документація Laravel [Електронний ресурс] – Режим доступу до ресурсу:  
<https://laravel.com/docs/9.x>.

# ДОДАТКИ

|  |   |  |  |  |                           |    |
|--|---|--|--|--|---------------------------|----|
|  |   |  |  |  | ІІЗ.КР.Б – 121 – 23 – ІІЗ | .  |
|  |   |  |  |  |                           |    |
|  | . |  |  |  |                           | 59 |

Клас що відповідає за надсилання користувачу повідомлення з подальшою зміною пароля.

```
class ForgetRepository implements ForgetInterface
{
    // Use ResponseAPI Trait in this repository
    use AuthResponseAPI;

    public function ForgetPassword(ForgetRequest $request)
    {
        $lang = $request->lang;
        $email = $request->email;

        if(User::where('email', $email)->doesntExist()) {
            return response([
                'message' => 'Email Invalid'
            ], 401);
        }

        // generate Random Token
        $token = rand(10,100000);

        try {
            DB::table('password_resets')->insert([
                //'lang' => $lang,
                'email' => $email,
                'token' => $token
            ]);
            //dd($token);

            Mail::to($email)->send(new ForgetMail($lang, $token, $email));

            return response([
                'message' => 'Reset Password Mail send on your email'
            ],200);
        }catch (Exception $exception) {
            return response([
                'message' => $exception->getMessage()
            ], 400);
        }
    }
}
```

Клас що відповідає за введення або зняття фільму з прокату.

```
public function MovieSwitch(Request $request): \Illuminate\Http\JsonResponse
{
    try {
        $movie_id = Movies::where('id', $request->id)->first();
        $movies = MoviesLanguages::where('movie_id', $request->id)->get();
        if ($request->rolling === 'true') {
            $movie_id->update([
                'rolling' => 1,
            ]);
            foreach ($movies as $ms) {
                $ms->update([
                    'rolling' => 1,
                ]);
            }
        }
        if ($request->rolling === 'false') {
            $movie_id->update([
                'rolling' => 0,
            ]);
            foreach ($movies as $ms) {
                $ms->update([
                    'rolling' => 0,
                ]);
            }
        }
        return response()->json([
            'movie_id' => $movie_id,
            'movies' => $movies,
        ], 200); //State Code;
    } catch (Exception $exception) {
        return response()->json([
            'message' => $exception->getMessage()
        ], 400);
    }
}
```

Клас, що відповідає за додавання, виведення, перемикання та редагування фільму

```
<?php

namespace App\Repositories;

use App\Http\Requests\MovieRequest;
use App\Interfaces\MovieInterface;
use App\Models\Language;
use App\Models\Movies;
use App\Models\MoviesLanguages;
use App\Models\Sessions;
use App\Traits\AuthResponseAPI;
use Illuminate\Http\Request;
use Illuminate\Support\Facades\Auth;
use Exception;

class MovieRepository implements MovieInterface
{
    // Use ResponseAPI Trait in this repository
    use AuthResponseAPI;

    public function RentMovie($lang): \Illuminate\Http\JsonResponse
    {
        try {
            $alldata = [];
            $movie_id = Movies::where('rolling', '1');
            $languages = Language::where('prefix', $lang)->first();
            $movies = MoviesLanguages::where('language_id', $languages->id)-
            >where('rolling', 1)->get();

            return response()->json([
                'movie_id' => $movie_id,
                //'movie_id' => $movie_id,
                //'languages' => $movie_id->languages()->get(),
                'movies' => $movies,
                'array_push' => $alldata,
                'getLocale' => $languages,
                //'languages' => $languages->movies()->get(),
            ], 200); //State Code;

        } catch (Exception $exception) {
            return response()->json([
                'message' => $exception->getMessage()
            ], 400);
        }
    }

    public function MovieSwitch(Request $request): \Illuminate\Http\JsonResponse
    {
        try {
            $movie_id = Movies::where('id', $request->id)->first();
            $movies = MoviesLanguages::where('movie_id', $request->id)->get();
            if ($request->rolling === 'true') {
                $movie_id->update([
                    'rolling' => 1,
                ]);
                foreach ($movies as $ms) {

```

```

        $ms->update(
            [
                'rolling' => 1,
            ]
        );
    }
}
if ($request->rolling === 'false') {
    $movie_id->update(
        [
            'rolling' => 0,
        ]
    );
    foreach ($movies as $ms) {
        $ms->update(
            [
                'rolling' => 0,
            ]
        );
    }
}
return response()->json([
    'movie_id' => $movie_id,
    'movies' => $movies,
], 200); //State Code;

} catch (Exception $exception) {
    return response()->json([
        'message' => $exception->getMessage()
    ], 400);
}
}

public function CurrentMovie(Request $request):
\Illuminate\Http\JsonResponse// $lang, $movie_id
{
    try {
        $m = Movies::where('id', $request->movie_id)->first();
        $languages = Language::where('prefix', $request->lang)->first();
        $movie = MoviesLanguages::where('language_id', $languages->id)-
>where('movie_id', $request->movie_id)->first();
        $locale = app()->getLocale();

        $sessions = Sessions::query();
        if ($request->period === 'Today') {
            $sessions->whereDate('start', '=', date("Y-m-d"))
                ->whereDate('start', '>', date("Y-m-d H:i"));
        } elseif ($request->period === 'Tomorrow') {
            $sessions->whereDate('start', '=', date("Y-m-d", strtotime("+1
day"))) ->where('movie_id', $request->movie_id);
        } elseif ($request->period === 'Week') {
            $sessions->whereDate('start', '>=', date("Y-m-d", strtotime("+1
day"))) ->whereDate('start', '<=', date("Y-m-d", strtotime("+1 week"))) -
>where('movie_id', $request->movie_id);
        } elseif ($request->period === 'Month') {
            $sessions->whereDate('start', '>=', date("Y-m-d", strtotime("+1
day"))) ->whereDate('start', '<=', date("Y-m-d", strtotime("+1 month"))) -
>where('movie_id', $request->movie_id);
        }

        $screentype = explode(',', $request->screentype);
        if ($request->screentype) {
            $sessions->whereHas('cinema_hall', function ($q) use
($screentype, $request) {
                $q->whereIn('working', $screentype);
            });

```

```

    }

    if(count($sessions->get()) == '0') {
        return response()->json([
            'movie' => $movie,
            'm' => $m,
            'request->movie_id' => $request->movie_id,
            'request->lang' => $request->lang,
            'request->period' => $request->period,
            'sessions' => $sessions->get(),
            'date-today' => date("Y-m-d H:i:s"),
            'date-tomorrow' => date("Y-m-d", strtotime("+1 day")),
            'date-week' => date("Y-m-d", strtotime("+1 week")),
            'date-month' => date("Y-m-d", strtotime("+1 month")),
            'message' => count($sessions->get()) == '0', // 'There are no
upcoming sessions.',
        ], 200); //State Code;
    }

    return response()->json([
        'movie' => $movie,
        'm' => $m,
        'request->movie_id' => $request->movie_id,
        'request->lang' => $request->lang,
        'request->period' => $request->period,
        'sessions' => $sessions->get(),
        'date-today' => date("Y-m-d H:i:s"),
        'date-tomorrow' => date("Y-m-d", strtotime("+1 day")),
        'date-week' => date("Y-m-d", strtotime("+1 week")),
        'date-month' => date("Y-m-d", strtotime("+1 month")),
        'request->screentype' => $request->screentype,
        'message' => count($sessions->get()) == '0',
    ], 200); //State Code;

} catch (Exception $exception) {
    return response()->json([
        'message' => $exception->getMessage()
    ], 400);
}

}

public function AddMovie(Request $request):
\Illuminate\Http\JsonResponse//MovieRequest
{
    try {
        $user = Auth::user();
        $role = $user->hasRole('admin');//role check
        if ($role) {
            if ($request->new_movie !== 'false') {
                $createnewmovie = (Movies::create([
                    'original_title' => $request->original_title,
                    'age' => $request->age,
                    'year' => date('Y-m-d', strtotime($request->year)),
                    'start_rental_period' => date('Y-m-d',
strtotime($request->start_rental_period)),
                    'over_rental_period' => date('Y-m-d',
strtotime($request->over_rental_period)),
                    'duration' => date('h:i:s', strtotime($request-
>dur)), //date('H:i:s', strtotime($request->duration))
                    'rolling' => 0, //date('H:i:s', strtotime($request-
>duration))
                    'trailer' => $request->trailer,
                ]))->id;
            }
        }
    }
}

```

```

        $movie = MoviesLanguages::create([
            'title' => $request->title,
            'director' => $request->director,
            'description' => $request->description,
            'content' => ($request->hasFile('img') ? $request-
>file('img')->store('content', 'public') : null),
            'language' => $request->language,
            'movie_id' => $request->movie_id === null ? $createnewmovie
: $request->movie_id, //$request->movie_id === null
            'language_id' => $request->language_id,
            'rolling' => 0,
        ]);
        return response()->json([
            'dur' => strtotime($request->dur),
            'request' => $request,
        ], 200);
    } else {
        return response()->json([
            'message' => "Not Admin"
        ], 404);
    }
} catch (Exception $exception) {
    return response()->json([
        'message' => $exception->getMessage()
    ], 400);
}
}

public function EditMovie($id): \Illuminate\Http\JsonResponse
{
    try {
        $user = Auth::user();
        $role = $user->hasRole('admin');//role check
        if ($role) {
            $localization = Language::all();
            $movie = Movies::where('id', $id)->first();
            $movieTranslation = MoviesLanguages::where('movie_id', $id)-
>get();

            return response()->json([
                'localization' => $localization,
                'movie' => $movie,
                'movieTranslation' => $movieTranslation,
                'info' => $id,
            ], 200);
        } else {
            return response()->json([
                'message' => "Not Admin"
            ], 404);
        }
    } catch (Exception $exception) {
        return response()->json([
            'message' => $exception->getMessage()
        ], 400);
    }
}

public function UpdateMovie(MovieRequest $request):
\Illuminate\Http\JsonResponse
{
    try {
        $user = Auth::user();
        $role = $user->hasRole('admin');//role check

```

```

        $MoviesLanguages = MoviesLanguages::where(
            [
                ['movie_id', '=', $request->movie_id],
                ['language_id', '=', $request->language_id],
            ]
        )->get();
        if ($role) {
            Movies::where(
                [
                    ['id', '=', $request->movie_id],
                ]
            )->update(
                [
                    'original_title' => $request->original_title,
                    'age' => $request->age,
                    'year' => date('Y-m-d', strtotime($request->year)),
                    'start_rental_period' => date('Y-m-d',
                        strtotime($request->start_rental_period)),
                    'over_rental_period' => date('Y-m-d',
                        strtotime($request->over_rental_period)),
                    'duration' => date('H:i:s', strtotime($request->duration)),
                ]
            );
            MoviesLanguages::where(
                [
                    ['movie_id', '=', $request->movie_id],
                    ['language_id', '=', $request->language_id],
                ]
            )->update(
                [
                    'title' => $request->title,
                    'director' => $request->director,
                    'description' => $request->description,
                    'content' => ($request->hasFile('img') ? $request->file('img')->store('content', 'public') : null),
                ]
            );
            return response()->json([
                'request' => $request,
                'MoviesLanguages' => $MoviesLanguages,
            ], 200);
        } else {
            return response()->json([
                'message' => "Not Admin"
            ], 404);
        }
    } catch (Exception $exception) {
        return response()->json([
            'message' => $exception->getMessage()
        ], 400);
    }
}

public function AllMovie(): \Illuminate\Http\JsonResponse
{
    try {
        $user = Auth::user();
        $role = $user->hasRole('admin');//role check
        if ($role) {
            $movie_id = Movies::all();
            $movies = MoviesLanguages::all();
            return response()->json([
                'movie_id' => $movie_id,
                'movies' => $movies,
            ], 200); //State Code;
        }
    }
}

```

```

        } else {
            return response()->json([
                'message' => "Not Admin"
            ], 404);
        }
    } catch (Exception $exception) {
        return response()->json([
            'message' => $exception->getMessage()
        ], 400);
    }
}
}
}

```

## Головний компонент клієнтської частини

```

import './App.css';
import {Header} from "../common/Header";
import {createBrowserRouter, RouterProvider} from "react-router-dom";
import LogIn from "../components/LogIn";
import Home from "../components/Home";
import 'bootstrap/dist/css/bootstrap.css'
import Register from "../components/Register";
import ForgetPassword from "../components/ForgetPassword";
import axios from "axios";
import ProfilePage from "../components/ProfilePage";
import React, {Suspense} from "react";
import ResetPassword from "../components/ResetPassword";
import "react-datetime/css/react-datetime.css";
import Blog from "../components/blog/Blog";
import Booking from "../components/booking/Booking";
import EditProfilePage from "../components/EditProfilePage";

function App() {
  const router = createBrowserRouter(
    [
      {
        path: "/*:lang/",
        element: <Header/>,
        children: [
          {
            path: "/*:lang/",
            element: <Home/>,
          },
          {
            path: "/*:lang/register",
            element: <Register/>,
          },
          {
            path: "/*:lang/login",
            element: <LogIn/>,
          },
          {
            path: "/*:lang/forgetpassword",
            element: <ForgetPassword/>,
          },
          {
            path: "/*:lang/resetpassword/:token/:email",
            element: <ResetPassword/>,
          },
          {
            path: "/*:lang/profile",
            element: <ProfilePage/>,
          },
          {
            path: "/*:lang/editprofile",
            element: <EditProfilePage/>,
          },
          {
            path: "/*:lang/movie/:movie_id",
            element: <Blog/>,
          },
          {
            path: "/*:lang/movie/:movie_id/session/:session_id",

```

```

        element: <Booking/>,
      },
      {
        path: "*",
        element: <h1>404</h1>,
      },
    ],
  },
  {
    path: "/",
    element: <h1>Just a second</h1>,
  },
]
);

// Set main base url
axios.defaults.baseURL = "http://localhost:8000";
axios.defaults.withCredentials = true;
// Beare Token Save
axios.defaults.headers.common['Authorization'] = 'Bearer ' +
localStorage.getItem('token');

return (
  <Suspense>
    <RouterProvider fallback="<div>sdfsdf</div>" router={router}/>
  </Suspense>
);
}

export default App;

```