КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ імені ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра прикладних інформаційних систем

122 «Комп'ютерні науки»
(шифр і назва спеціальності)

«Прикладне програмування»

(назва освітньої програми)

Курсова робота

на тему: «Веб-сервіс пошуку квитків у кінотеатрі»

Виконала

(Підпис)

Боголій Владислава Олегівна
(прізвище, ім'я, по батькові)

Керівник _____
(прізвище, ім'я, по батькові)

(Резолюція «До захисту»)

Попередній захист:

(Висновок: "До захисту в екзаменаційній комісії ")

Завідувач кафедри _____

(Підпис) (Прізвище, ініціали) (Дата)

Київ – 2024

ЗМІСТ

ВСТУП

Актуальність дослідження. У сучасному світі за стрімкого розвитку інформаційних технологій все більше послуг в різних сферах стають доступними для використання віддалено з допомогою гаджетів. Розробка вебсервісу, спеціалізованого на пошуку квитків у кінотеатрі, відповідає вимогам користувачів, забезпечуючи їм простий і зручний інструмент для моніторингу фільмів та пошуку доступних сеансів. Це дозволяє користувачам заощадити час та зусилля, які раніше витрачалися на пошук інформації та візити до кінотеатрів.

Мета дослідження. Розробка вебсервісу для перегляду розкладу сеансів у кінотеатрі, а також резервації місць на обрані сеанси.

Завдання дослідження. Відповідно до мети, визначено наступні завдання дослідження:

- вивчення теорії розробки вебсервісів
- аналіз та проєктування архітектурних рішень
- розробка програмної частини

Об’єкт дослідження. Об’єктом дослідження є процес проєктування технічної частини та програмна реалізація вебсервісу для пошуку квитків у кінотеатрі.

Предмет дослідження. Предметом дослідження даної роботи є теоретичні відомості про вебсервіси, можливі архітектурні рішення та програмні засоби для створення вебсервісу.

Методи дослідження. Для досягнення поставленої мети буде використано наступні методи дослідження:

- Метод аналізу (вивчення наявних сервісів для виявлення потреб користувачів)
- Метод порівняння (вибір оптимальних технічних рішень для реалізації проєкту)

- Метод синтезу (поєднання всіх рішень для отримання єдиного результату)

Практичне значення одержаних результатів. Кінцевим результатом є вебсервіс, який спрощує процес отримання даних про актуальні фільми та розклад сеансів у кінотеатрі і надає можливість резервації квитків без потреби йти до кінотеатру.

РОЗДІЛ 1. ТЕОРІЯ РОЗРОБКИ ВЕБСЕРВІСІВ

1.1 Поняття вебсервісу

У сучасному світі більшу частина нашого життя пов’язана з технологіями. Ми весь час маємо взаємодію: на роботі, у побуті та під час відпочинку. Таким чином зі зростанням попиту на різні сервіси, відбувається стрімкий розвиток в галузі інформаційних технологій. За короткий час людство пройшло шлях від створення перших комп’ютера та програми до створення смартфонів з застосунками, призначеними для них.

Вебсервіс – це поняття, яке є подібним до вебзастосунку, проте головною відмінністю є те, що використання вебсервісу неодмінно передбачає роботу з даними, обмін ними та їх опрацювання.

Вебсервіс складається з двох частин: клієнта та сервера. Клієнтом є браузер користувача, функція якого полягає у відображенні інформації та легкій взаємодії з користувачем за допомогою зручного графічного інтерфейсу. Серверна частина забезпечує правильне виконання команд, отримання доступу до інформації, її обробку, а також передачу.

На відміну від комп’ютерних програм та мобільних застосунків, вебсервіс не потребує попереднього встановлення на пристрій, виділення пам’яті під нього та постійного завантаження оновлених версій користувачем, так як доступ до сервісу та взаємодія з ним здійснюється лише через браузер.

Від вебсайтів вебсервіси зазвичай візуально нічим не відрізняються, проте різниця між ними полягає у тому, що функціонал сервісу є більш широким, ніж сайту, та він завжди має з’єднання з програмними сервісами для обміну інформацією інформації. Вебсервіс зазвичай використовується для виконання певного завдання і має ряд вбудованих функцій, в той час як вебсайт є статичним, має більш інформативний характер і є просто візуальним відображенням якоїсь інформації.

Переваги вебсервісів:

- Вебсервіси не займають місце в пам’яті пристрою, оскільки вся інформація зберігається в зовнішніх джерелах, а браузер слугує лише для її відображення;
- Його робота не залежить від типу пристрою (приватний комп’ютер, мобільний телефон) та операційної системи (Windows, Linux, MacOS), так як для запуску важлива лише наявність браузера;
- Всі користувачі мають доступ до єдиної версії: користувач не потребує самостійно встановлювати оновлення, всі зміни відбуваються на єдиному сервері.

Створення вебсервісу – це розробка клієнтської та серверної частин, і для кожної зазвичай використовуються різні технології. Популярні технології, що реалізують клієнтську частину вебсервісів:

- Мова розмітки гіпертексту HTML. Використовується для створення макету візуальної частини, визначає розміщення всіх елементів на сторінці та їх атрибути. Більшість фреймворків, які можна використати при створенні вебсайтів та вебсервісів, мають за основу для візуалізації саме HTML;
- Мова стилю сторінок CSS (каскадні таблиці стилів). Задає стилі і зовнішній вигляд всіх елементів сторінки: написи, кнопки, зображення тощо;
- Мови програмування TypeScript, JavaScript. Відповідають за взаємодію з елементами сторінки, відслідковують дії користувача, натиснення кнопок, ввід даних та передають цю інформацію на бік серверу.

Для розробки серверної частини можна використати будь-яку з великої кількості мов програмування або їх поєднання. Серверна частина відповідає за виконання скриптів, дій над даними, відпрацювання алгоритмів, а отже може бути реалізована з використанням PHP, Python, Java, Node.js (environment), Dart, Ruby, ASP.NET (environment) тощо.

1.2 Типи вебсервісів

Вебсервіси можна по-різному класифікувати залежно від кількості сторінок та їх призначення, тощо. Так як вебсервіс за означенням дуже близький до вебзастосунку, то їх класифікація є подібною, проте, на відміну від застосунків, вебсервіси не можуть бути статичними, такими, що їх використання не викликає ніяких змін на сервері, так як користування вебсервісом має в основі роботу з даними, їх обмін та запис. Таким чином всі вебсервіси є динамічними.

Динамічний сервіс активно взаємодіє з сервером для миттєвого запису та зміни інформації, відповідно до дій користувача, а також оновлення та відображення її у режимі реального часу. При використанні такого вебсервісу, кожна взаємодія і подальша зміна елемента провокує оновлення сторінки та побудову макету заново, згідно нових даних.

Наступна класифікація заснована на кількості сторінок: одно- та багатосторінкові вебсервіси. Перший тип – SPA (Single-Page Application) – має лише одну сторінку, яка доступна для відображення, і саме вона має в собі весь графічний інтерфейс для простої взаємодії з користувачем. Перевагою такого типу є швидкість, так як всі елементи завантажуються одразу і при виконанні дій оновлюється лише інформація в них. Також це сприяє праці офлайн, оскільки виконується лише один запит на сервер, після чого дані можна просто зберегти в кеш. У такому випадку недоліком є лише те, що за відсутності постійного інтернет-з’єднання неможлива актуалізація даних, якщо на серверній стороні трапляться якісь зміни.

Багатосторінкові сервіси складаються з багатьох сторінок, з якими можна взаємодіяти. Зазвичай кожна зі сторінок має окреме призначення та виконує ряд певних функцій. Такий тип сервісу є ідеальним, коли необхідне зберігання великої кількості інформації та реалізація широкого функціоналу, який може бути не пов’язаний між собою. Маючи певні потреби, користувач може перейти на сторінку, яка їх задовольняє і не пропонує зайвого.

Залежно від призначення існує велика кількість сервісів, проте до основних можна віднести наступні:

- Електронна комерція – сервіси, призначені для комерційної діяльності, наприклад інтернет-магазини. Такі сервіси зазвичай мають розділений інтерфейс для користувачів-покупців та користувачів-менеджерів;
- Системи керування вмістом – такі сервіси мають конструктори для створення вебсайтів, доступні та зрозумілі для звичайних користувачів. Завдяки сервісам такого типу, можна створювати та розміщувати власні сайти без спеціальних технічних знань, а також розміщувати на них різні мультимедіа і ділитися інформацією;
- Вебпортали – використовуються для отримання зручного доступу до різноманітних послуг або інформації в межах одного сайту. Головною перевагою такого типу сервісу є пряме підключення до сервера та зазвичай обов'язкова реєстрація кожного користувача для отримання доступу до повного функціоналу, що забезпечує надійність сервісу;
- Анімація – сервіси, які зазвичай не мають в основі важливого функціоналу, проте створені для розваги користувача. Такі сервіси рідко використовуються як повноцінний окремий сервіс, проте найчастіше вони є інтеграцією у застосунки широкого призначення (комерційний, навчальний тощо);
- Розширені сервіси – подібні до комп'ютерних програм. Вони обходять обмеження браузера, підключаючись до встановлених користувачем плагінів. Завдяки цьому розширені сервіси мають кращу графіку та більш інтерактивний інтерфейс. Недоліком такого типу сервісів є залежність від плагінів, які можуть застаріти або більше не підтримуватися;
- Поступові сервіси – кросплатформні сервіси-гібриди вебсайту та мобільного сервісів і застосунків. Головна перевага таких сервісів в тому, що вони можуть синхронізуватися та працювати у фоновому режимі. Незважаючи на завантаження через браузер, вони є адаптивними і їх

використання схоже на користування встановленим мобільним застосунком, з розширеними можливостями, наприклад отримання push-повідомлень або додавання на робочий стіл для миттєвого запуску.

1.3 Вебсервіси різного призначення та їх використання

У сучасному світі кожна окрема частина нашого життя пов’язана з інформаційними технологіями. Маючи різне призначення та функціональність вони значно полегшують наше існування, допомагаючи вирішити велику кількість різних повсякденних задач. Ми користуємося програмами, застосунками та сервісами в роботі, для отримання та поширення інформації, навчання та розвитку, відпочинку тощо. Завдяки інтеграції таких технологій у різні сфери суспільства відбувається стрімка автоматизація багатьох речей, які раніше могли викликати незручності.

Прикладом можуть бути електронні поштові сервіси – їх принцип роботи є подібним до роботи звичайного поштового відділення, але з наявними очевидними перевагами:

- Незалежність від часу: відправка електронних листів можлива у будь-який час будь-якого дня;
- Миттєвість: отримувач майже одразу має змогу прочитати лист, час доставки якого не залежить від свят чи погодних умов, а лише від наявності інтернет-з’єднання;
- Вкладення: окрім тексту до листа можна додати зображення та різні типи документів і посилання;
- Масованість: можливість написати лише один лист та відправити його десяткам користувачів одразу.

Всім відомим прикладом такого вебсервісу є Gmail – електронна пошта від сервісів Google.

Портали для запису на прийом до лікаря. Більше немає необхідності відстоювати величезні черги, щоб спочатку записатися, а потім потрапити до

лікаря, завдяки вебсервісам це можна зручно зробити вдома. Такі сервіси дають користувачу можливість самостійно обрати лікаря та дату і час прийому. Прикладами такого типу вебсервісів в українській мережі є MedCard24 та Helsi.

Навчальні портали дають змогу отримувати нові знання та розвиватися для кожного і незалежно від місця знаходження. Такі сервіси мають в собі тисячі навчальних курсів та посібників з різних галузей: від філософії та мистецтва до медицини та інженерії. Завдяки таким вебсервісам, користувачі мають змогу розвиватися в різних напрямках, обравши найцікавіші для себе та зручно планувати свій час на навчання, адже доступність до застосунку теж залежить лише від наявності інтернет-з'єднання. Coursera, edX – приклади таких вебсервісів, відомі у всьому світі.

Інтернет-магазини. Вебсервіси комерційного типу наразі є найпопулярнішими у всьому просторі Всесвітньої павутини. Вони є зручними для всіх типів користувачів: як для покупців, так і для осіб, що пропонують свої товари. Призначення таких вебсервісів є дуже різноманітними: продаж-обмін одягу, книг, квитків до театру та кіно, тварин, послуг, житла, а іноді все це містить один сервіс. Інтеграція таких вебсервісів у бізнесі значно підвищує показники продажу закладу, що пропонує свої послуги, а також полегшує покупцям процес вибору та купівлі, тому є популярним рішенням. Наприклад, Amazon, Rozetka – вебсервіси, що пропонують товари з різних категорій; інтернет-магазини Vivat, Moyo – вебсервіси вузької галузі.

РОЗДІЛ 2. АНАЛІЗ ТА ПРОЄКТУВАННЯ АРХІТЕКТУРНИХ РІШЕНЬ

2.1 Поняття архітектури програмного забезпечення

В наш час будь-яке широко використовуване програмне забезпечення здебільшого складається з сотень та тисячів рядків коду, які відповідають за різні функції. Саме тому розібратися у коді може бути дуже складно, навіть якщо файли добре розділено та впорядковано. Безперечно, наочне відображення загальної картини розрозбляємого великого чи навіть маленького проєкту значно

спрощує розуміння того, як поєднуються між собою різні його частини та що має бути отримано в результаті виконання.

В 90-х роках XX століття проєктування архітектури програмного забезпечення стало одним з найважливіших етапів розробки ПЗ – його почали використовувати загалом у всіх сферах, що так чи інакше пов’язані зі створенням програмного продукту. З початком активного використання людством комп’ютерів почалося поширення технологічно складних проєктів, що складаються з тисячів рядків коду, який складно переписати та відтворити знову у випадку втрати. Саме стрімкий розвиток та попит у галузі розробки ПЗ сприяв поширенню в цій галузі такого поняття як архітектура та її різновиди, типи, шаблони.

Як правило, архітектура ПЗ описує та наочно демонструє взаємодію різних компонентів ПЗ, їх взаємодію, складові та результати виконання як окремих частин готового проєкту. Існують різні способи описати таку схему, зважаючи на те, які є вимоги до результату, а також на тип та структуру проєкту, що спричинило появу різних архітектурних шаблонів та рішень, які можуть бути використанні при розробці.

Визначення та проєктування архітектури є важливим етапом розробки ПЗ з багатьох причини. Наприклад, сам розробник при моделюванні чітко визначає, що є важливим у програмі і який має бути функціонал та компоненти, та їх взаємодія для отримання позитивного результату. Це допомагає впевнено рухатися далі, оскільки архітектурна схема дає розробнику інформацію про те, з чого почати програмну реалізацію та як крок за кроком, рухаючись за схемою, завершити завдання. Також така схема допомагає визначити можливе джерело помилки, якщо така виникає, що полегшує її пошук та усунення.

Необхідність різного представлення інформації, залежно від типу та завдання ПЗ, спричинила створення різних видів архітектурних схем (моделей), які звуться архітектурними стилями і є широко використовуваними шаблонами.

Зараз розробники можуть взяти за основу різні стандартні шаблони для моделювання структури свого проєкту, виходячи з того, яким має бути задовільний результат.

Отже, розробка архітектури програмного забезпечення є значним етапом у створенні певного проєкту. Вона допомагає краще розуміти складову окремих елементів та їх поведінку, взаємодію компонентів внутрішньої та зовнішньої частин програми, логічні зв'язки між ними, загальний функціонал, а також те, якою має бути фінальна реалізація для досягнення поставленої цілі.

2.2 Аналіз архітектурних шаблонів

Чітко визначеного поняття, що таке архітектура ПЗ немає, проте зрозуміло, якою є мета її проєктування: відображення структури, зв'язків та логіки ПЗ. Головна ідея полягає у тому, щоб зробити структуру ПЗ наочно зрозумілою та спростити вирішення задач, які виникатимуть протягом усього процесу розробки.

Залежно від різних потреб, існує кілька різних стандартних архітектурних шаблонів зі своїми перевагами.

– Багаторівнева архітектура (N-layer)

Найпоширеніший та найбільш використовуваний шаблон серед розробників, де всі компоненти на модулі, групи яких складають різні рівні (рис. 2.2.1). Обмежень в їх кількості немає, але зазвичай така архітектура складається з чотирьох рівнів: подання, бізнес-логіка, сервіси та база даних. Тобто, наприклад, до рівня подання відносяться всі модулі, які відповідають за візуальне подання контенту користувачеві (користувацький інтерфейс); до рівня бізнес-логіки – логічний функціонал; до сервісів – з'єднання з зовнішніми джерелами, файловою системою та базами даних; бази даних – бази, таблиці та файли, в яких зберігається необхідна інформація.

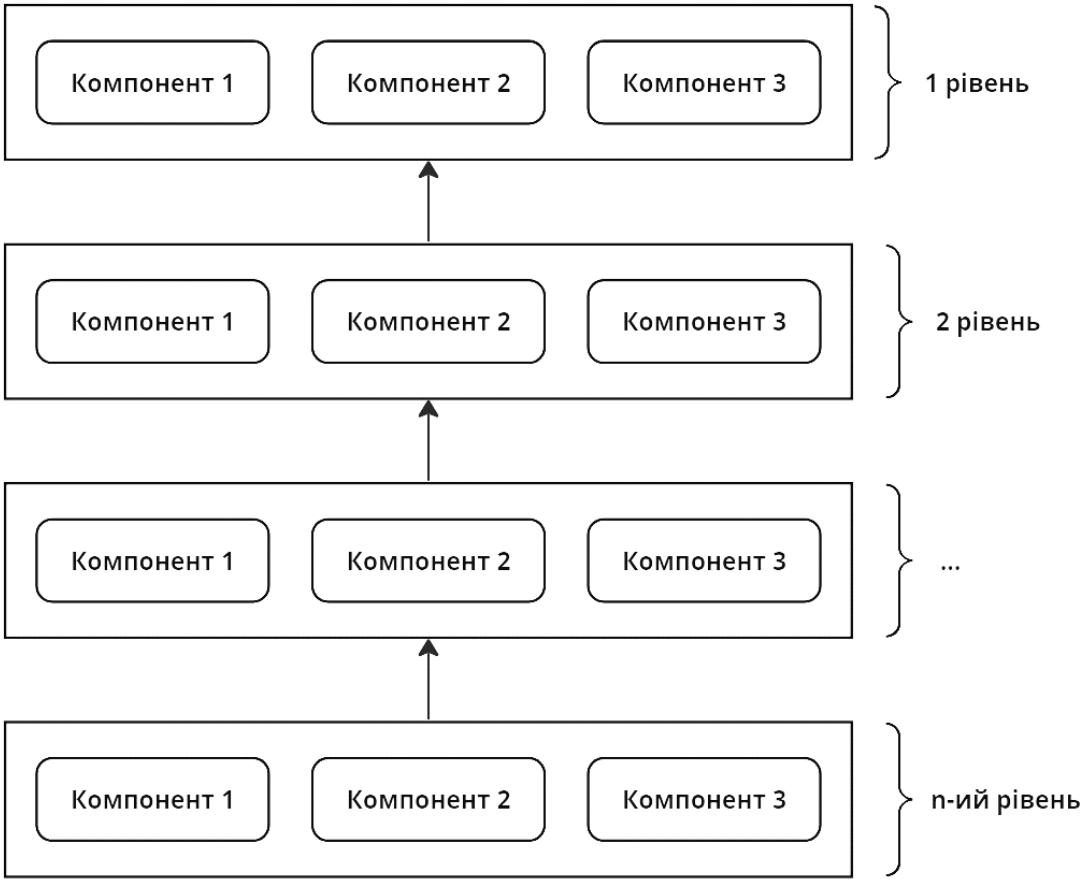

Рис. 2.2.1 – Приклад багаторівневої архітектури

– Клієнт-серверна архітектура (Client-server)

Цей шаблон складається з двох сторін: сервер та клієнт (рис. 2.2.2). Зв’язок компонентів полягає у тому, що сторона клієнта надсилає запити, а сервер їх отримує, обробляє та надсилає відповідь. В такій архітектурі за окремий компонент Клієнт можуть виступати один або декілька клієнтів, але, як правило, існує лише один сервер. Окрім того, база даних в такій архітектурі може існувати як винесений назовні компонент, до якого сервер має доступ. Зазвичай, використовується при розробці веб-сервісів, тому зв’язок між сторонами відбувається за допомогою інтернет-з’єднання.

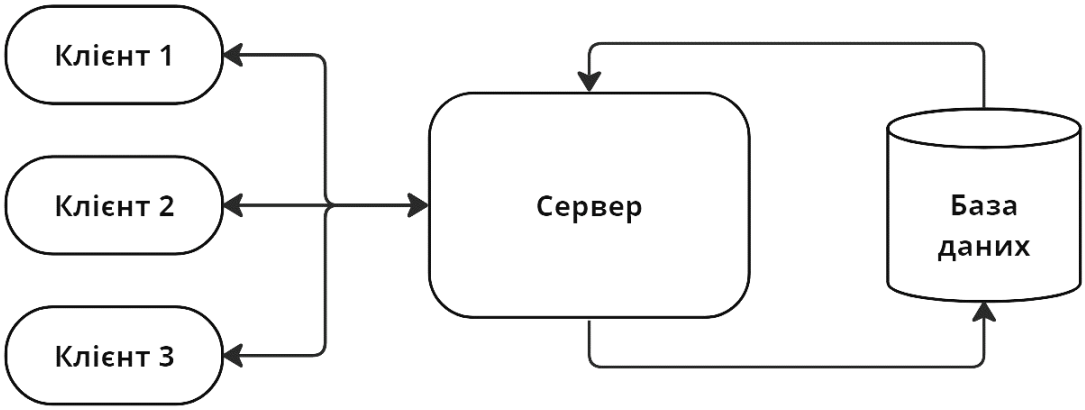

Рис. 2.2.2 – Приклад клієнт-серверної архітектури

– Моноліт (Monolith)

Ще один часто використовуваний шаблон – моноліт, в якому всі компоненти тісно пов’язані між собою (рис. 2.2.3). За такого типу архітектури немає певного поділу на групи, а візуалізація, сервіси та вся робота з даними активно взаємодіють один з одним.

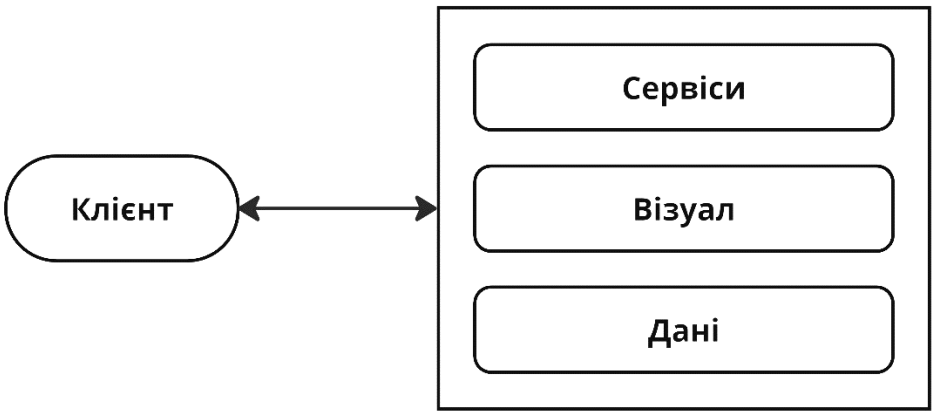

Рис. 2.2.3 – Приклад монолітної архітектури

– Головний-підлеглі (Master-slave)

Цей шаблон нагадує роботу в команді: існує один головний, хто визначає та розподіляє роботу, та підлеглі, хто роботу виконує. В такому типі архітектури «Головний» компонент відповідає за загальне керування, тобто визначає завдання, розподіляє їх та делегує компонентам підлеглим на виконання (рис. 2.2.4). Після завершення своїх процесів, підлеглі компоненти

надсилають результати назад головному для подальших дій. Такий тип архітектури є доволі ефективним за рахунок використання методу паралелізму та рівномірного поділу навантаження.

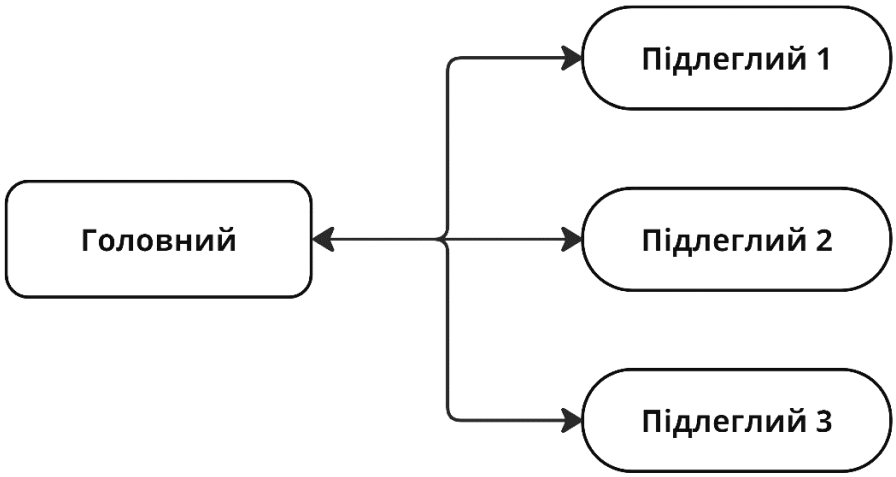

Рис. 2.2.4 – Приклад архітектури «Головний-підлеглі»

– Канали і фільтри (Pipes and filters)

Такий шаблон архітектури зазвичай використовується при розробці систем, які мають справу з потоками даних та їх обробкою. Його логіка полягає у тому, що кожен етап обробки даних є окремим фільтром, через який ці дані проходять, і після проходження всіх таких фільтрів ми отримуємо кінцевий результат (рис. 2.2.5).

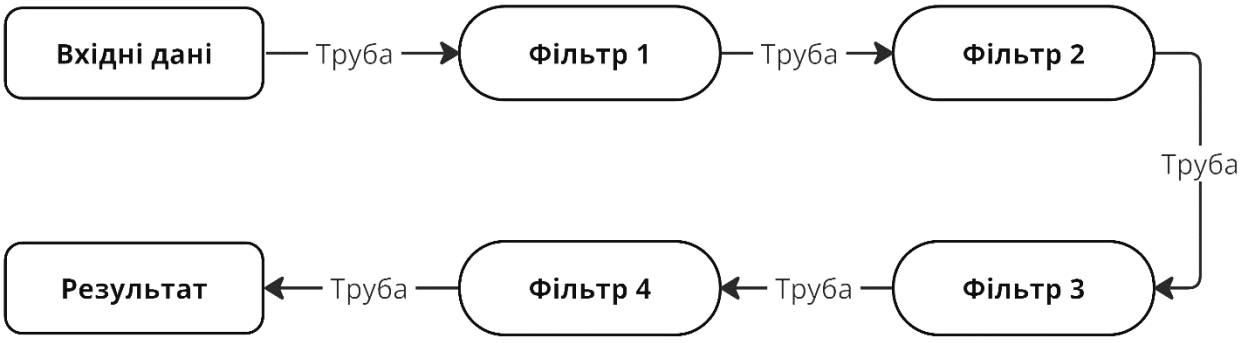

Рис 2.2.5 – Приклад архітектури «Канали та фільтри»

– Посередник (Broker)

Шаблон Посередник є подібним до клієнт-серверної архітектури, проте має у своєму складі додатковий компонент – посередника, та більшу кількість серверів (рис. 2.2.6). На стороні клієнта генерується запит, який надсилається посереднику, після чого посередник визначає, на який сервер перенаправити запит на виконання.

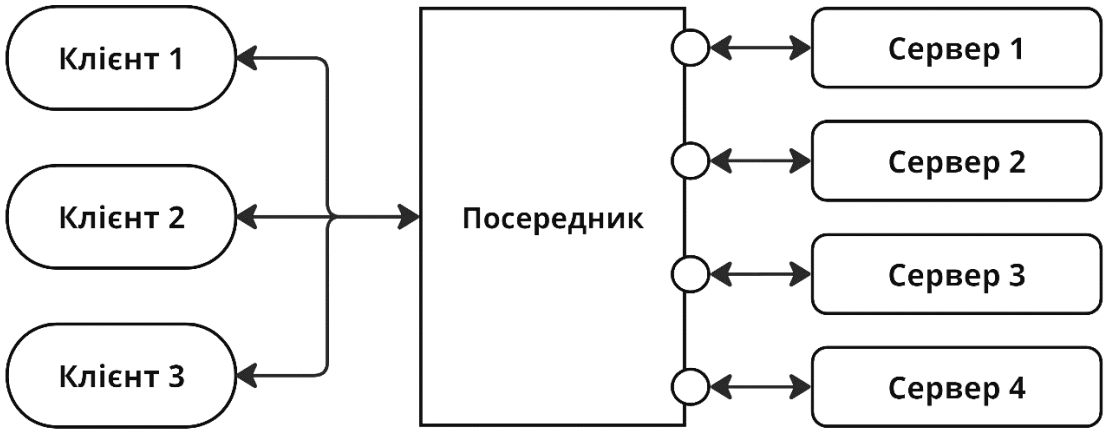

Рис. 2.2.6 – Приклад архітектури «Посередник»

– Дошка (Blackboard)

Така архітектура має у своєму складі три елементи: дошка, контролер та база знань (рис. 2.2.7). Елемент «Дошка» визначає завдання, «Контролер» вирішує, які з модулів в базі знань підходять для виконання поставленої задачі, розподіляє її між ними та відправляє на виконання, а після завершення виконання модулями, отримує результати та направляє їх назад на «Дошку», де результати об’єднуються. Цей шаблон схожий на архітектуру типу Головний-підлеглий, проте, на відміну від нього, де завдання розподілено рівномірно між багатьма модулями, в цій архітектурі задача розподіляється між модулями залежно від їх функціоналу та можливостей.

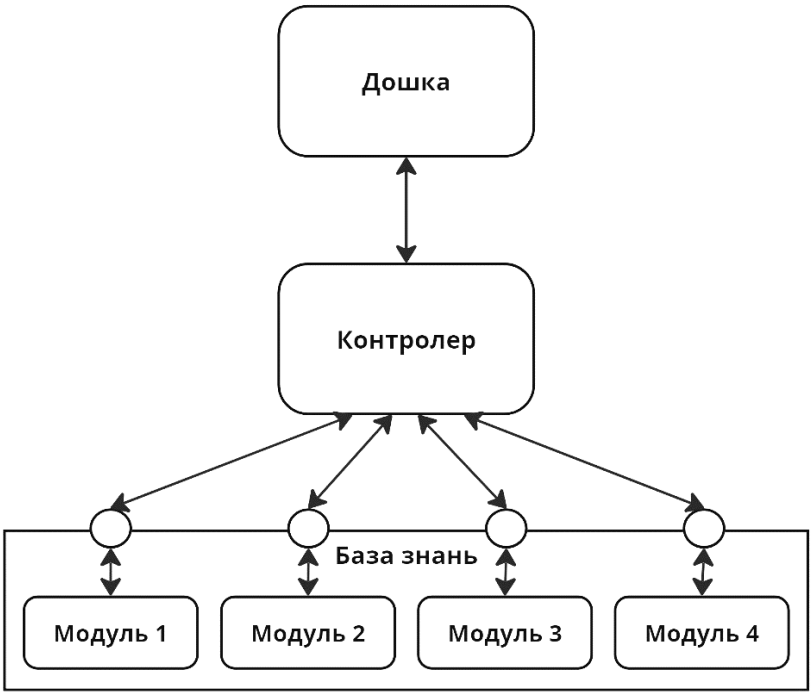

Рис. 2.2.7 – Приклад архітектури «Дошка»

– Модель-Представлення-Контролер (Model-View-Controller, MVC)

Всі елементи такого архітектурного шаблону поділено між трьома групами: модель, представлення та контролер (рис. 2.2.8). До першої групи відносяться всі файли, в які передаються та в яких зберігаються дані, до групи представлення – файли, що відповідають за візуалізацію та користувацький інтерфейс, а до останньої групи – функціонал, який виконує взаємодію між першими двома, тобто отримує дані, введені користувачем, обробляє їх, передає до баз даних, або вже наявні дані передає до групи візуалізації, для їх належного відображення користувачеві.

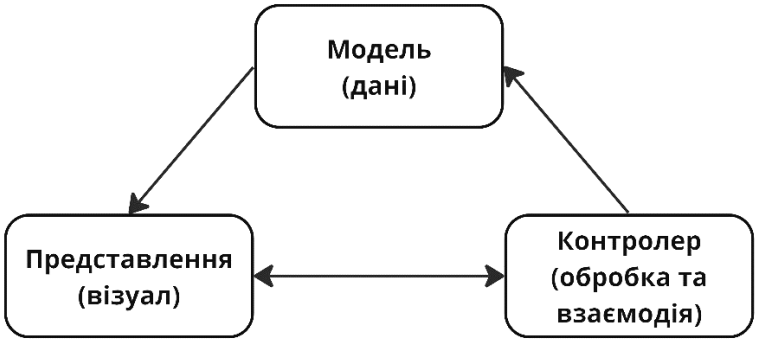

Рис. 2.2.8 – Приклад архітектури «Модель-Представлення-Контролер»

– Рівноправ’я (Peer-to-peer, P2P)

З назви цього шаблону зрозуміло, що всі компоненти в такому типі архітектури є рівноправними елементами, тобто вони не мають залежності один від одного або від центрального елемента, проте мають зв’язки для обміну інформацією між собою (рис. 2.2.9). За такої архітектури кожен окремий компонент виступає як клієнтом – може надсилати запити іншим учасникам схеми, – так і сервером – отримує та обробляє запити інших клієнтів.

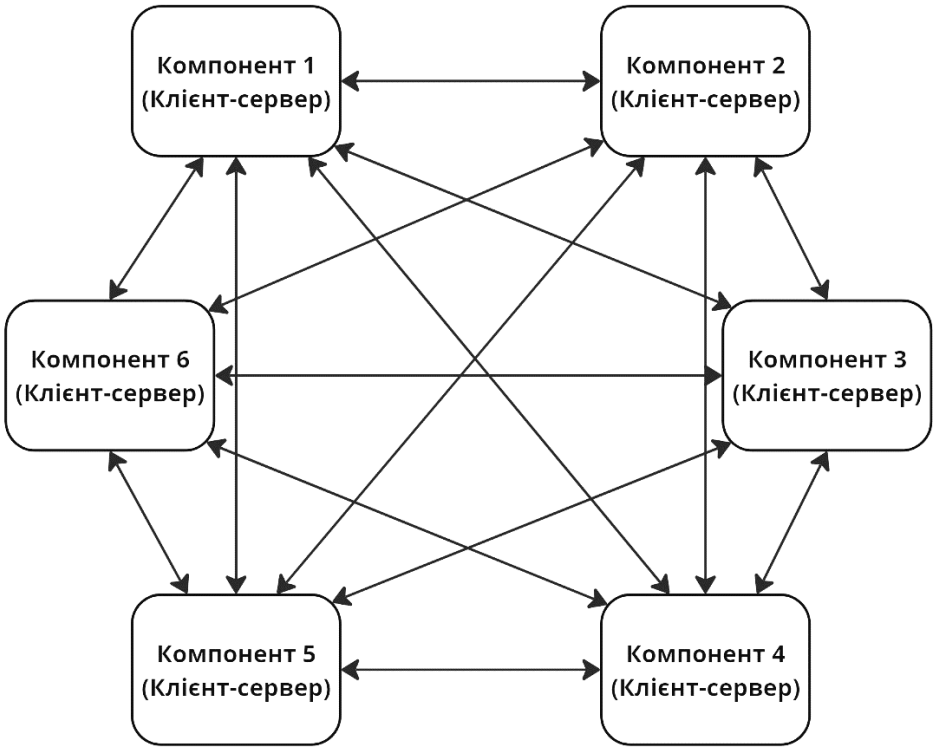

Рис. 2.2.9 – Приклад архітектури «Рівноправ’я»

– Керована подіями (Event-driven, event-bus)

Така архітектура повністю побудована на подіях. Компонентами такого шаблону є подія-початок, яка запускає процес, слухач подій, який є посередником та визначає, яку подію було виконано, та подія-результат (рис. 2.2.10). Виглядає це наступним чином: натискання кнопки на сайті – подію-початок – зчитує слухач подій та визначає, яка подія-результат – наприклад, створення нового користувача, – має бути виконана.

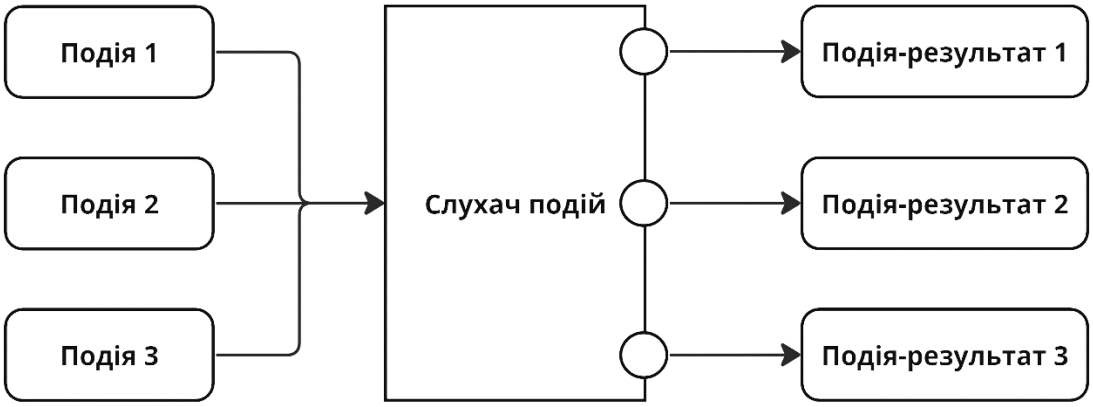

Рис. 2.2.10 – Приклад керованої подіями архітектури

– Мікросервіси (Microservices)

Наступний шаблон – шаблон на основі мікросервісів, – повністю заснований на ідеї поділу функціоналу розроблюваного програмного забезпечення на окремі сервіси – мікросервіси (рис. 2.2.11). Кожна функція, яка може бути виконана за запитом клієнта, є самостійним сервісом у такій архітектурі. Залежно він дій клієнта, відбувається звернення до певного мікросервісу, в якому обробляється отримана чи існуюча інформація та який повертає результат своїх дій на сторону клієнта.

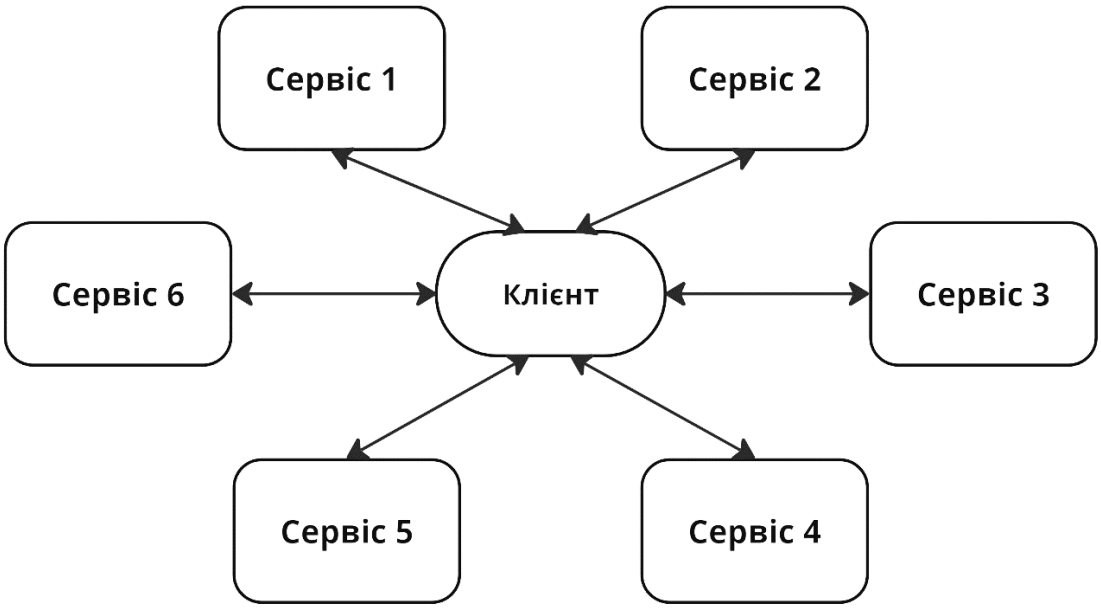

Рис. 2.2.11 – Приклад архітектури «Мікросервіси»

– Мікроядра (Microkernel)

Цей тип архітектури має два головні компоненти: основне ядро та підключені мікроядра (рис. 2.2.12). Основне ядро відповідає за головну функцію, яка лежить в основі всього розроблюваного ПЗ, а додаткові функції можуть бути підключені як мікроядра, які є розширенням головного ядра. Така архітектура, наприклад, може бути використана при розробці застосунків для листування – так званих «месенджерів», – де за відправлення текстових повідомлень відповідає головне ядро, а додаткові функції – голосові повідомлення, відео-повідомлення, надсилання мультимедіа – є підключеними мікроядрами.

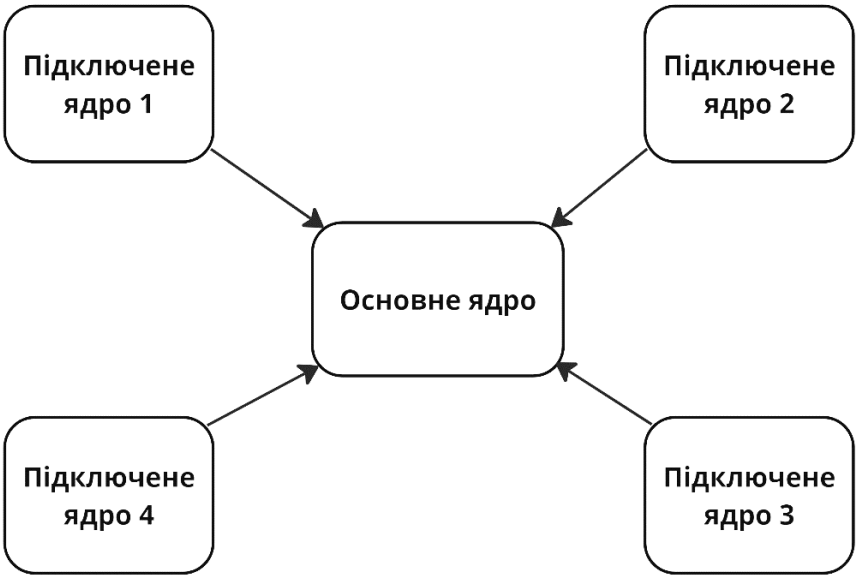

Рис. 2.2.12 – Приклад архітектури «Мікроядра»

– Безсерверна (Serverless)

Розробка ПЗ з безсерверним типом архітектури є дещо простішою за інші, так як вона виключає необхідність підтримки серверу через його відсутність. Така архітектура в основному зосереджена на розробці зовнішнього вигляду та логіки функціоналу, а для підтримки розробники зазвичай використовують хмарні сервери, яка керує запуском, виділенням ресурсів та масштабуванням (рис. 2.2.13).

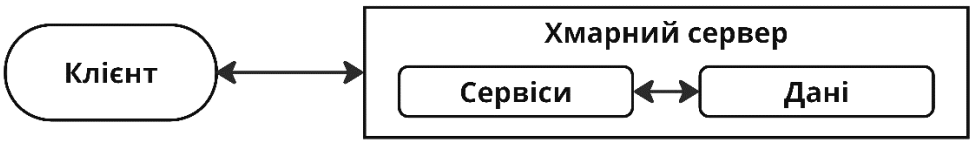

Рис. 2.2.13 – Приклад безсерверної архітектури

2.3 Вибір та проєктування архітектури вебсервісу для пошуку квитків у кінотеатрі

Головним призначенням нашого сервісу є спрощення процедури пошуку квитків у кінотеатрі. Користувач повинен мати змогу дізнатися розклад показу фільмів, інформацію про фільми, а також наявність вільних місць на обраний сеанс. Для реалізації такої технології найнеобхіднішими елементами є користувацький інтерфейс – для відображення даних клієнту – та розклад – дані, які користувач отримає.

Проаналізувавши поставлене завдання та визначивши загальні елементи, прийнято рішення про використання шаблону багаторівневої архітектури з поділом на наступні рівні:

- Візуальне подання
- Моделі
- Сервіси
- База даних

Організувавши проєкт таким чином, можна легко виправляти можливі помилки та змінювати функціонал без шкоди для інших елементів, оскільки рівні не будуть значною мірою впливати один на одного, а лише обмінюватися своїми даними для коректної роботи нашого застосунку.

Запуск сервісу буде супроводжуватися отриманням інформації у форматі JSON з бази даних з інформацією про розклад сеансів, їх час, ціну, та наявність вільних місць для вибору. Це буде відбуватися за допомогою сервісів, які звертаються до бази даних з запитом, після чого ці сервіси конвертують отриману інформацію з формату JSON відповідно до створених моделей. Після

проходження цих етапів ми отримуємо структуровану інформацію, яку передаємо до рівня візуального представлення – UX/UI – такий результат отримає користувач застосунку для подальшої роботи згідно своїх потреб.

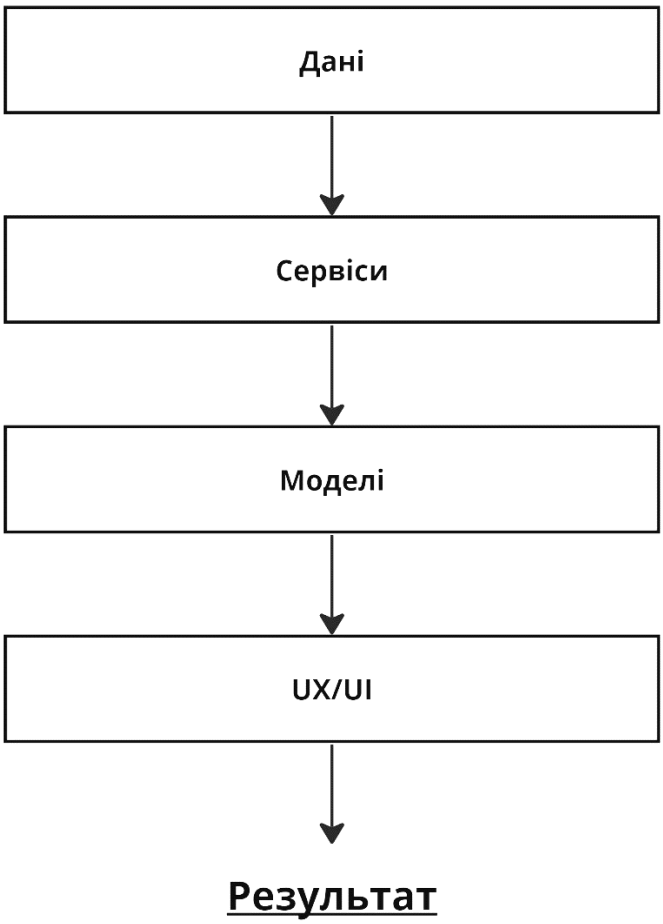

Рис. 2.3.1 – Архітектура веб-застосунку для пошуку квитків у кінотеатрі

РОЗДІЛ 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ВЕБСЕРВІСУ ДЛЯ ПОШУКУ КВИТКІВ У КІНОТЕАТРІ

3.1 Створення бази даних

Для програмної реалізації вебсервісу будемо використовувати мову програмування Dart, до якої підключимо фреймворк Flutter від Google, який дозволяє додати до коду візуальну обгортку з використанням вбудованих віджетів, а також дає змогу розроблювати кросплатформне програмне забезпечення: для запуску на персональних комп'ютерах, мобільних платформах та через браузер (вебсайти, вебзастосунки, вебсервіси).

Так як запуск сервісу буде супроводжуватися отриманням інформації про розклад та фільми, як було зазначено у підрозділі 2.3, почнемо зі створення структури та заповнення бази даних. Для наших потреб використаємо інший вебсервіс – Firestore database від Google, що знаходиться за наступним посиланням: https://firebase.google.com.

За допомогою цього ресурсу ми маємо змогу створити власну базу даних та в подальшому взаємодіяти з нею: отримувати, записувати та змінювати інформацію в ній.

База даних складається з двох документів: для збереження розкладу сеансів та для збереження інформації про фільм – Showtimes та Movies відповідно. Документ з інформацією про фільми рівнем нижче має перелік номерів фільмів – їхніх унікальних id. Для кожного такого id номеру існують стандартні поля з даними про певний фільм (табл. 3.1.1). Вся інформація та посилання на зображення взяті з безкоштовного вебресурсу TMDB (The Movie Database): https://www.themoviedb.org.

Таблиця 3.1.1 – Структура кожного id в документі Movies.

Поле	Тип	Призначення
age_rating	String	Інформація про віковий рейтинг

title_ua	String	Назва фільму українською
title_original	String	Оригінальна назва фільму
overview	String	Опис фільму
poster	String	Шлях до зображення постеру
runtime	Int	Тривалість фільму
tagline	String	Слоган фільму
genres	String[]	Жанри фільму

Документ з інформацією про розклад сеансів рівнем нижче має перелік дат, доступних для отримання: поточний день та три наступних. Кожна дата містить інформацію про фільми, які транслюються в кінотеатрі в цей день, а саме – id фільмів. Кожен фільм на наступному рівні має перелік, який вказує на години показу цього фільму в обраний день – Showtimes – і врешті кожен часовий показник має лише два стандартні для кожного такого елемента поля, які його характеризують (табл. 3.1.2).

Таблиця 3.1.2 – Структура Showtime в документі Showtimes.

Поле	Тип	Призначення
price	String	Ціна на сеанс
seats_taken	Int[]	Номери зарезервованих місць

3.2 Створення моделей

Для зручної структуризації та обробки даних, розробимо моделі. Це будуть класи об’єктів, потрібні для того, щоб в подальшому створювати їх прототипи та

працювати з ними. Відповідно до створеної нами бази даних, нам необхідні два нові класи: Фільм та Сеанс.

Перший клас Фільм – Movie – потрібний для створення об’єктів, які містять інформацію про фільм, тож такий клас має відповідні поля (табл. 3.2.1).

Таблиця 3.2.1 – Атрибути класу Movie.

Атрибут	Тип	Призначення
id	String	Унікальний код фільму
showtimes	Showtime[]	Години показу фільму
ageRating	String	Інформація про віковий рейтинг
titleUA	String	Назва фільму українською
titleOriginal	String	Оригінальна назва фільму
overview	String	Опис фільму
poster	String	Шлях до зображення постеру
runtime	Int	Тривалість фільму
tagline	String	Слоган фільму
genres	String[]	Жанри фільму

Наступний клас – Showtimes – необхідний для зберігання інформації про сеанс показу, тож містить відповідні поля (табл. 3.2.2). Також масив об’єктів цього класу присутній серед необхідних для заповнення атрибутів при створенні прототипу класу Movie, так як кожен фільм має один або перелік сеансів для показу в день.

Таблиця 3.2.2 – Атрибути класу Showtime.

Атрибут	Тип	Призначення
time	String	Час початку сеансу
price	String	Ціна за одне місце
seatsTaken	Int[]	Номери недоступних для резервації місць

3.3 Створення сервісів

Перш за все встановимо залежності з firebase_database, firebase_core та cloud_firestore для нашого проєкту та імпортуємо відповідні бібліотеки до наших сервісів, що забезпечить з'єднання з базою даних.

Так як наша база даних складається з двох документів, створюємо два сервіси: один для доступу до документу Movies, інший – до документу Showtimes.

Перший сервіс – MovieService – має лише одну функцію, яка приймає id фільму як аргумент, отримує інформацію з бази даних за шляхом «Movies/id/…», після чого повертає цю інформацію як результат свого виконання.

Другий сервіс з'єднується з документом Showtimes. Він складається з двох функцій: для отримання даних та для запису даних.

Перша функція – getSchedule – отримуючи день як аргумент, він отримує інформацію про фільми та їх сеанси, які доступні в обраний день. Для кожного id фільму в структурі документу, яку визначено в підрозділі 3.1, цей сервіс звертається до MovieService, передаючи цей унікальний номер та отримує від нього інформацію про фільм, після чого збирає всі дані та зберігає їх у масив

типу Movie, зазначеного у підрозділі 3.2, та повертає створений масив як результат виконання.

Друга функція – reserveSeats – отримує наступний перелік аргументів: час сеансу, день сеансу, ідентифікатор фільму та масив місць, які є зарезервованими для цього сеансу. Після чого за шляхом «Showtimes/day/id/showtime/seats_taken» записує нове значення.

3.4 Тестування вебсервісу для пошуку квитків у кінотеатрі

Останнім кроком реалізації нашого проєкту є створення візуальної обгортки для представлення даних користувачу та користувацького інтерфейсу для взаємодії з даними. З використанням стандартних віджетів використовуваного фреймворку Flutter, додаємо візуальне представлення наших моделей.

При запуску сервісу користувач бачить головну сторінку, на якій є перелік днів – «Сьогодні» є початковим значенням – для перегляду розкладу сеансів, а також список фільмів, які транслюються в обраний день. Для кожного фільму зазначено його основні відомості, такі як назва, слоган, жанри, опис, тривалість, постер, віковий рейтинг, а також наявні сеанси.

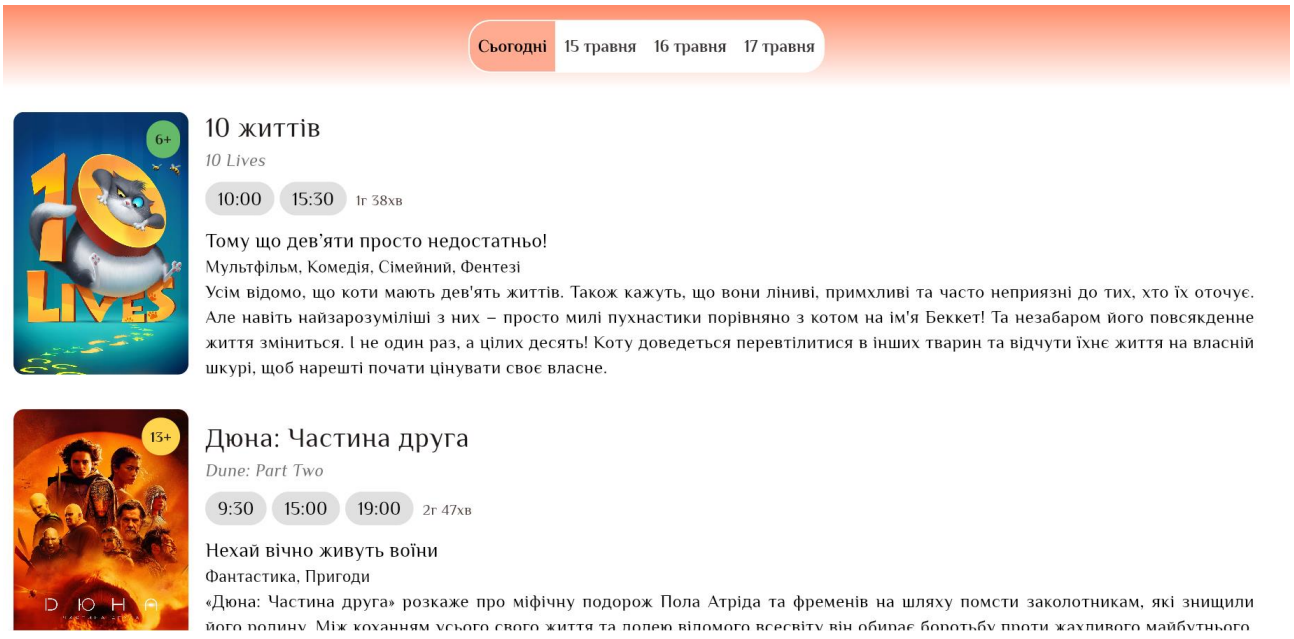

Рис. 3.4.1 – Головна сторінка вебсервісу

При наведенні курсору миші на сеанс користувач може дізнатися ціну одного квитка. При натисканні на сеанс, з’являється вікно для вибору та резервації місць на обраний сеанс. Сеанс, трансляція якого вже триває, обрати неможливо – вікно для резервації не з’явиться.

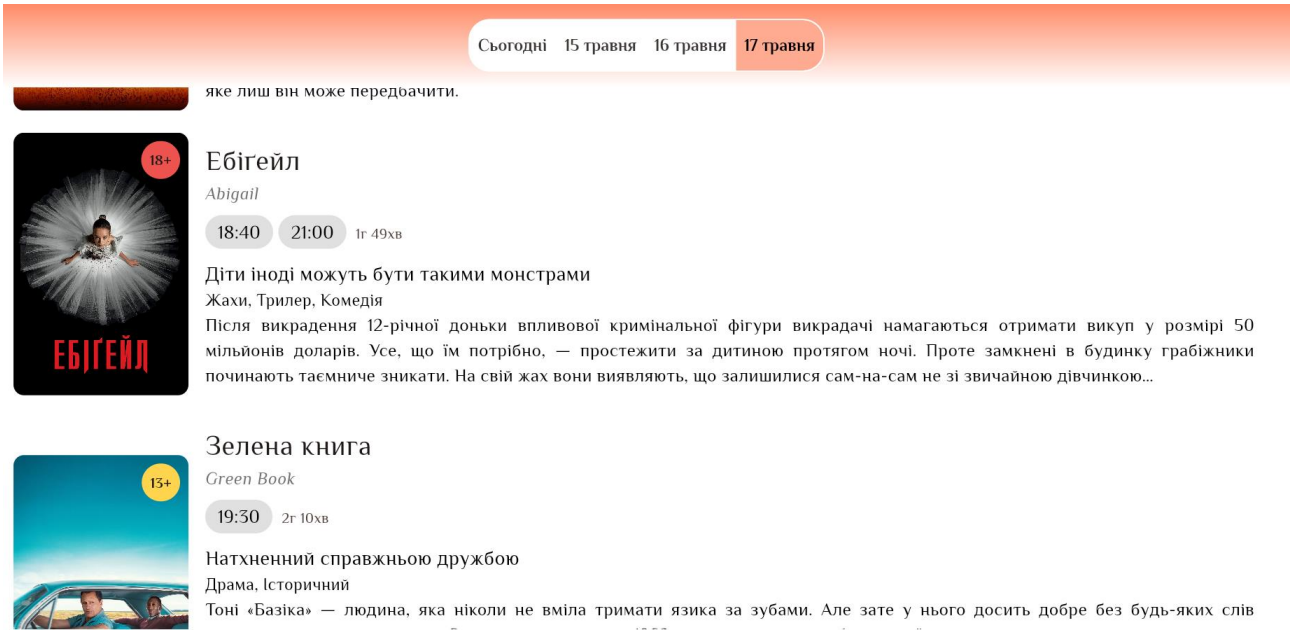

Рис. 3.4.2 – Розклад на обраний день

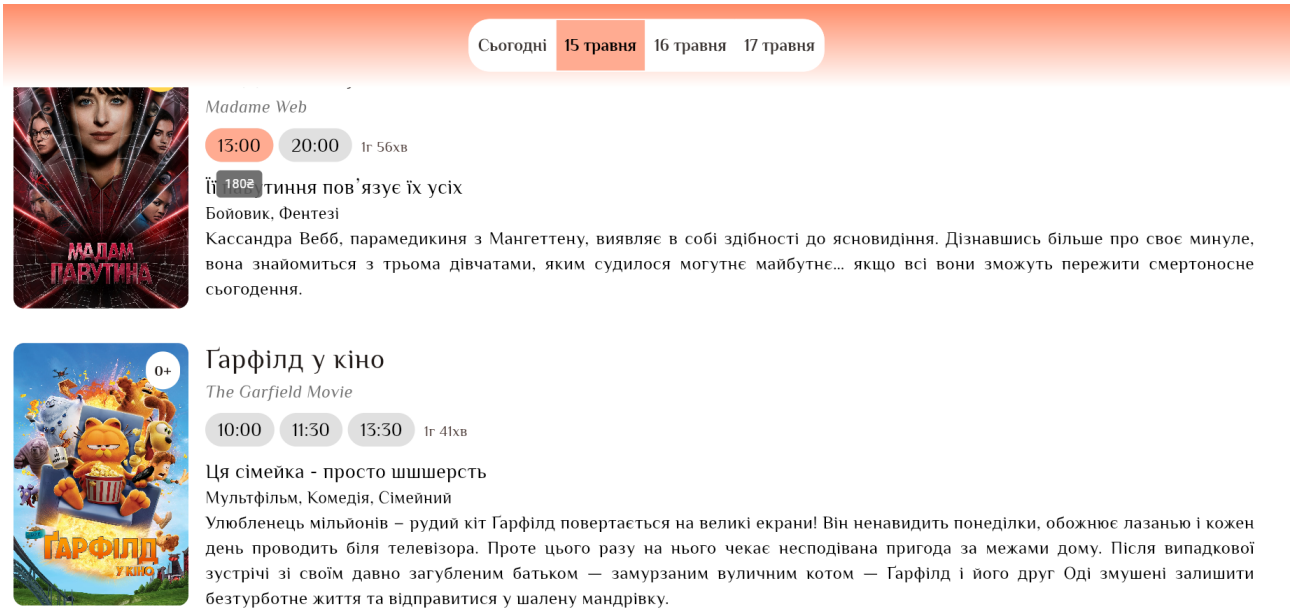

Рис. 3.4.3 – Ціна квитка на сеанс відображається при наведенні курсору миші

Віно резервації має коротку інформацію про сеанс – назву фільму, дату, час, ціну за один квиток – та сітку розташування місць у кінозалі з позначеними недоступними місцями. При виборі місця користувач бачить номери обраних місць та їх загальну ціну, яка залежить від кількості місць. Також доступні дві кнопки: повернутися назад та зарезервувати обрані місця.

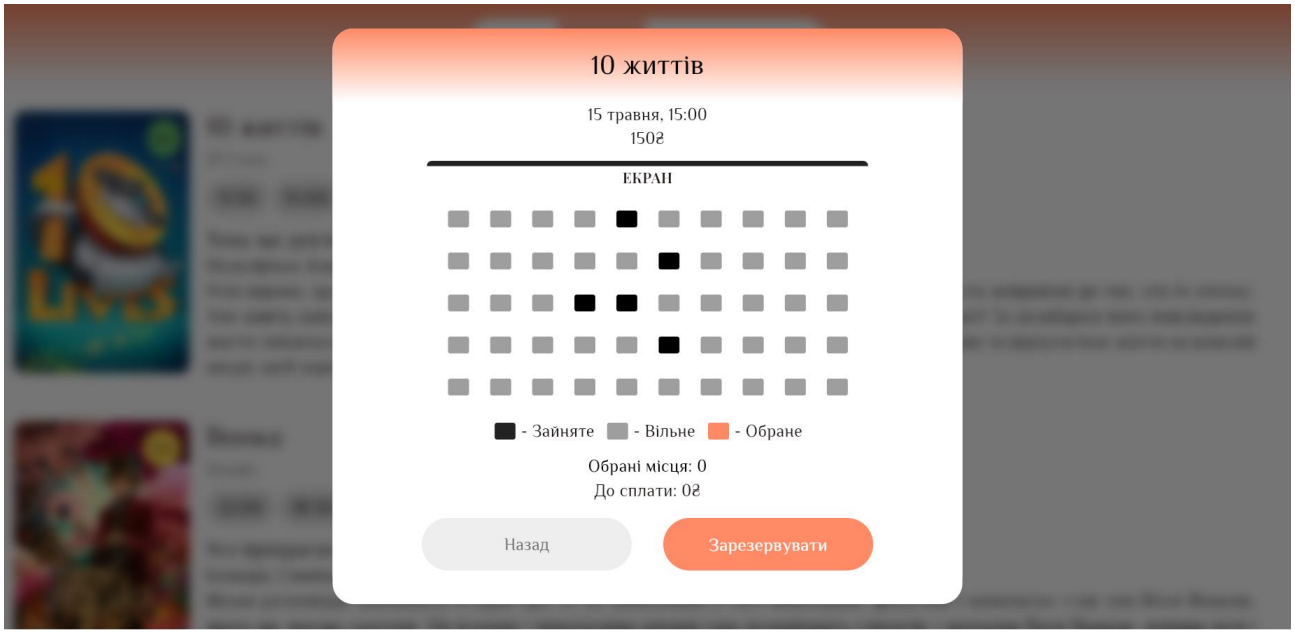

Рис. 3.4.4 – Вікно резервації місць

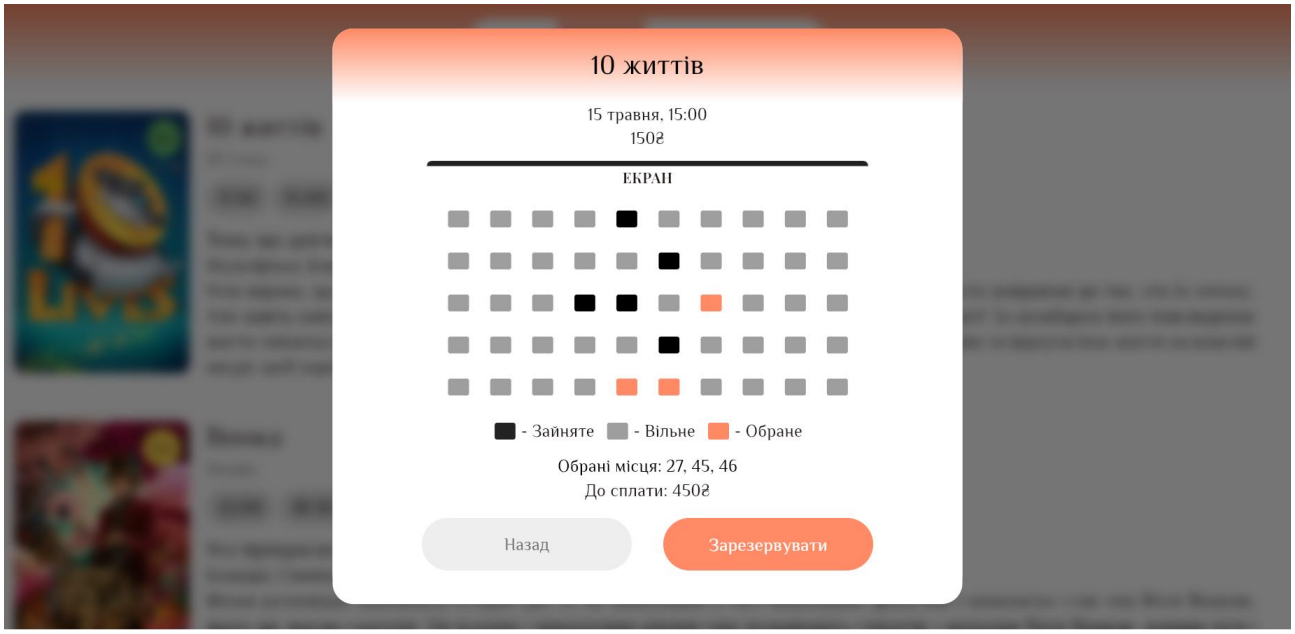

Рис. 3.4.5 – Обрані користувачем місця

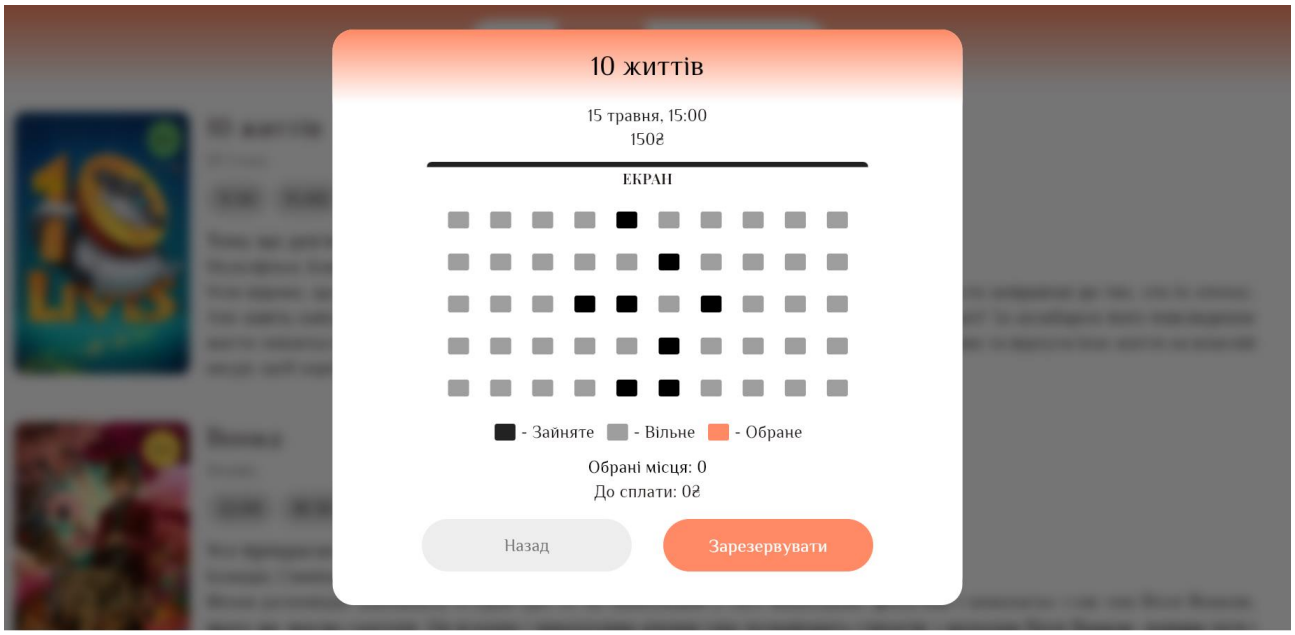

Рис. 3.4.6 – Після резервації місця недоступні

Рис. 3.4.7 – Користувач не може зарезервувати місця, не обравши жодного

ВИСНОВКИ

В процесі виконання курсової роботи було досліджено наступні питання:

1. Теоретичні відомості про вебсервіси, їх розробку, а також класифікацію в залежності від типу, структури та призначення.
2. Що таке архітектура програмного забезпечення, яке її значення при розробці проєкту. Які архітектурні шаблони існують, їхні особливості та наочні приклади.
3. Проєктування компонентів програми згідно обраного архітектурного рішення та їхньої взаємодії. Створення користувацького інтерфейсу для представлення даних користувачу у належному вигляді.

В результаті ми отримали повноцінний вебсервіс, який надає користувачам можливість отримання інформації про розклад сеансів у кінотеатрі та фільми, а також можливість вибору та резервації місць на обраний сеанс без необхідності йти до кінотеатру.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Архітектура веб-додатків – як вибрати? [Електронний ресурс] – Режим доступу до ресурсу: https://blog.ithillel.ua/articles/web-application-architecture
2. Архітектура програмного забезпечення. [Електронний ресурс] – Режим доступу до ресурсу: https://wezom.com.ua/ua/blog/arhitektura-programmnogo-obespecheniya
3. Найважливіші архітектурні шаблони, які необхідно знати. [Електронний ресурс] – Режим доступу до ресурсу: https://devzone.org.ua/post/nayvazlyvishi-arkhitekturni-shablony-iaki-neobkhidno-znaty
4. What is the blackboard design pattern? [Електронний ресурс] – Режим доступу до ресурсу: https://justgokus.medium.com/what-is-the-blackboard-design-pattern-c834227dc617
5. Jacyntho M. D., Schwabe D., Rossi G. A software architecture for structuring complex web applications. Journal of web engineering. 2002. Т. 1, №1. С. 2–6.
6. Software architecture [Електронний ресурс] – Режим доступу до ресурсу: https://www.freelancer.com/what-is-software-architecture
7. 10 common software architectural patterns in a nutshell. [Електронний ресурс] – Режим доступу до ресурсу: https://towardsdatascience.com/10-common-software-architectural-patterns-in-a-nutshell-a0b47a1e9013
8. 10 types of web applications and how you can use them. [Електронний ресурс] – Режим доступу до ресурсу: https://www.imaginarycloud.com/blog/10-types-of-web-applications-and-how-you-can-use-them
9. Types of software architecture patterns. [Електронний ресурс] – Режим доступу до ресурсу: https://www.geeksforgeeks.org/types-of-software-architecture-patterns/
10. Types of web applications. [Електронний ресурс] – Режим доступу до ресурсу: https://www.intelivita.com/blog/types-of-web-applications/

11. Web application architecture [Електронний ресурс] – Режим доступу до ресурсу: https://peiko.space/blog/article/web-application-architecture
12. База даних [Електронний ресурс] – Режим доступу до ресурсу: https://firebase.google.com/
13. Дані про фільми [Електронний ресурс] – Режим доступу до ресурсу: https://www.themoviedb.org/