МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
“____”_____2024 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСНИКА ОСВІТНЬОГО СТУПЕНЯ
“БАКАЛАВР”

Тема: «Мобільний застосунок для створення і роботи з персональним відео-контентом "Show & Movie Tracker"»

Виконавець: Григоренко Катерина Володимирівна

Керівник: к.т.н Талалаєв Володимир Опанасович

Нормоконтролер: Варнавський В’ячеслав Володимирович

Київ 2024

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь бакалавр
Спеціальність 121 «Інженерія програмного забезпечення»
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Катерина НЕСТЕРЕНКО
"___" _____ 2024 р.

ЗАВДАННЯ
на виконання кваліфікаційної роботи студентки
Григоренко Катерини Володимирівни

1. Тема проекту: «Мобільний застосунок для створення і роботи з персональним відео-контентом "Show & Movie Tracker"» затверджена наказом ректора від 8.12.2023 р. № 2483/ст.

2. Термін виконання проекту: з 3.01.2024 р. до 29.02.2024 р.

3. Вихідні дані до проекту: програмний продукт розробити у вигляді мобільного застосунку для операційної системи на базі Android.

4. Зміст пояснювальної записки:
   1. Аналіз існуючих додатків відслідковування відео-контенту .
   2. Вимоги до застосунку.
   3. Структура програмного застосунку.
   4. Реалізація програмного застосунку.

5. Перелік обов'язкових слайдів презентації:
   1. Актуальність проекту.
   2. Вимоги до програмного продукту.
   3. Реалізація застосунку.
   4. Інтерфейс проекту.
   5. Демонстрація роботи прототипу застосунку.

6. Календарний план-графік

№ пор	Завдання	Термін виконання	Відмітка про виконання
1.	Складання та затвердження графіку кваліфікаційної роботи Написання 1 розділу, представлення керівнику	03.01.24 – 09.01.24	
2.	Попередній друк 1 розділу та допоміжних сторінок (чорновик) - титульної, завдання, графіка, реферат, список скорочень, зміст, вступ, список джерел, 1-й нормо-контроль.	10.01.24 – 17.01.24	
3.	Написання 2 розділу, представлення керівнику	18.01.24 – 22.01.24	
4.	Написання 3 розділу, представлення керівнику	23.01.24 – 26.01.24	
5.	Написання 4 розділу, представлення керівнику	27.01.24 – 31.01.24	
6.	Загальне редагування та друк пояснювальної записки, графічного матеріалу	01.02.24 – 09.02.24	
7.	Проходження нормо-контролю, перевірка на антиплагіат, перепліт пояснювальної записки.	01.02.24 – 05.02.24	
8.	Розробка тексту доповіді. Оформлення графічного матеріалу для презентації	05.02.24 – 09.02.24	
9.	Отримання відгуку керівника, рецензії.	10.02.24 – 22.02.24	
10.	Підготовка матеріалів для передачі секретарю ДЕК (ПЗ, CD-R з електронними копіями ПЗ, презентації, відгук керівника, рецензія) в папці	23.02.24 – 29.02.24	

7. Дата видачі завдання 03.01.2024

Керівник: к.т.н. Володимир ТАЛАЛАЄВ

Завдання прийняв до виконання: Катерина ГРИГОРЕНКО

Дата

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи : «Мобільний застосунок для створення і роботи з персональним відео-контентом "Show & Movie Tracker"»: 45 с., 17 рис., 10 інформаційних джерел.

МОБІЛЬНИЙ ЗАСТОСУНОК, ВІДЕО-КОНТЕНТ, ІНТЕРФЕЙС, РОЗРОБКА

Об’єкт розробки – Мобільний застосунок "Show & Movie Tracker"

Мета роботи – створення мобільного застосунку для зручної і швидкої роботи з персональним відео-контентом.

Метод розробки – ООП – об'єктно-орієнтований підхід.

Прогнозові припущення щодо розвитку розроблюваної системи – можливе розширення, як на операційну систему iOS, так і функціональні системи.

ABSTRACT

Explanatory note to the thesis «Mobile application for creating and working with personal video content "Show & Movie Tracker"»: 45 p., 17 Fig., 10 information sources.

MOBILE APPLACATION, VIDEO CONTENT, INTERFACE, DESIGN

Object of development – Mobile application "Show & Movie Tracker"

The purpose of the work - to create a mobile application for convenient and fast work with personal video content.

Development method – OOP – object-oriented approach.

Predictive assumptions regarding the development of the developed system - possible expansion, both to the iOS operating system and functional systems.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 6
ВСТУП 7
РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ВІДСЛІДКОВУВАННЯ ВІДЕО-КОНТЕНТУ 9
1.1. Розвиток мобільних додатків 9
1.2. Аналіз потреб користувачів 10
1.3. Аналіз існуючих мобільних застосунків з відслідковування відео-контенту 11
1.3.1. Мобільний додаток TV Time 12
1.3.2. Мобільний додаток Hobi: TV Series Tracker 13
1.3.3. Мобільний додаток CLZ Movies 14
Висновки до розділу 15
РОЗДІЛ 2. ВИМОГИ ДО ЗАСТОСУНКУ «SHOW & MOVIE TRACKER» 16
2.1. Нефункціональні вимоги 16
2.2. Функціональні вимоги 18
2.3. Програмні вимоги 20
Висновки до розділу 25
РОЗДІЛ 3. СТРУКТУРА ЗАСТОСУНКУ «SHOW & MOVIE TRACKER» 26
3.1. Взаємодія частин програмного забезпечення застосунку 26
3.2. Архітектура мобільних додатків 28
3.3. Архітектура застосунку 29
3.4. Діаграма класів застосунку 32
3.5. Прототип застосунку 38
Висновки до розділу 44
ВИСНОВКИ 45
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 47

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

ОС - операційна система.

БД - база даних.

СУБД - система управління базами даних.

ООП - об’єктно-орієнтоване програмування.

ПЗ – програмне забезпечення.

API - Application Programming Interface (прикладний програмний інтерфейс).

TMDB API - The Movie Database API .

XML - Extensible Markup Language (розширена мова розмітки).

MVVM - Model-View-ViewModel.

ВСТУП

З врахуванням сучасної динаміки цифрового світу та наслідків пандемії 2019 року, сфера розваг переживає надзвичайно швидкий розвиток, що призводить до безпрецедентного доступу користувачів до широкого асортименту кінематографічного контенту. Обмеження на кількість відвідувачів кінотеатрів, ускладнення зйомок та масове закриття меж країн вимагають кардинальних змін у стратегії діяльності кіностудій. Відтак, популярність стрімінгових сервісів стрімко зросла, що відображається, наприклад, лише на Netflix кількість підписників збільшилася більш ніж на 60 мільйонів за два роки. Разом із цим, збільшується обсяг контенту, доступного для перегляду в онлайн-кінотеатрах.

У такому контексті вибір відповідного контенту стає справжнім викликом, що потребує систематизації та зручних інструментів управління. Даний проект має на меті оптимізувати цей досвід користувачів. Мета додатку полягає в тому аби надати ефективні та зручні інструменти для керування процесом перегляду кінокартин. Це включає відстеження переглядів, створення персональних списків, обмін рекомендаціями та інтеграцію з зовнішніми джерелами даних.

У рамках дослідження було розглянуто ключові технології, використані при розробці програми, а також рішення, спрямовані на забезпечення зручності використання, персоналізований підхід та соціальну взаємодію в галузі кінематографії. Аналіз ринку та конкурентної обстановки надається для повного розуміння контексту та перспектив проекту.

Задачі цієї роботи включають в себе визначення загальної структури програми, виявлення та описання вимог проекту, забезпечення ефективного зберігання та управління даними користувачів, розробку зручного та інтуїтивно зрозумілого інтерфейсу, забезпечення безпеки даних та реалізацію механізмів автентифікації, підключення до зовнішніх джерел даних, оптимізацію продуктивності, тестування, ефективне керування станом програми та

забезпечення цілісності даних, а також створення механізмів збирання зворотного зв'язку від користувачів.

Цей проект націлений на задоволення потреб сучасного глядача, який бажає максимально насолоджуватися переглядом великої кількості контенту без зайвих труднощів у пошуку та виборі. Механізми фільтрації, інтелектуальні рекомендації та можливості персоналізації допомагатимуть кожному користувачеві знайти саме те, що відповідає його смакам та перевагам.

Крім того, проект має за мету створити спільноту глядачів, яка може обмінюватися враженнями від перегляду, рекомендувати одне одному цікавий контент та підтримувати активну взаємодію серед шанувальників кіно та серіалів. Можливість обговорення фільмів та серіалів прямо в додатку роблять його центром спілкування для фанатів кіно.

Отже, цей проект впливає не лише на спосіб, яким ми споживаємо контент, але й намагається створити повноцінну спільноту, середовище для любителів кіно та серіалів у світі цифрового розвитку.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧИХ ДОДАТКІВ ВІДСЛІДКОВУВАННЯ ВІДЕО-КОНТЕНТУ

1.1. Розвиток мобільних додатків

В епоху мобільних телефонів спостерігається стрімкий розвиток технологій. Щороку вимоги до мобільних пристроїв зростають, і виробники конкурують за звання кращого флагману, облаштовуючи їх все більш потужними технічними характеристиками.

На тлі стрімкого розвитку мобільних телефонів, слід відзначити, що розвиток мобільних додатків також визначається постійним ростом функціональності та вдосконаленням технологій. Зокрема, спостерігається нарощування можливостей використання штучного інтелекту та машинного навчання. Розробники все частіше інтегрують ці технології для покращення персоналізації додатків та надання користувачам більш інтелектуального взаємодії.

Зростаючі вимоги користувачів обумовлені необхідністю комфортного та швидкого доступу до інформації. Сучасні розробники інформаційних технологій створюють мобільні додатки, які спрямовані на вирішення різноманітних завдань.

Розробники мобільних додатків фокусуються на оптимізації ефективності та економії ресурсів батареї та пам'яті телефону. Спрощення інтерфейсу для користувача та покращення продуктивності додатків є пріоритетом.

Відзначається також акцент на трендах дизайну, орієнтовані на максимальну простоту для зручності користувачів, а не на виразність для розробників.

Разом із тим, розробники враховують ресурсозатратність додатків, враховуючи бажання користувачів щодо мінімізації обсягу пам'яті та навантаження на процесор смартфона. Тенденції в дизайні мобільних додатків націлені на простоту і функціональність, надаючи перевагу зручності користувачів перед творчістю розробників.

Усі ці тенденції обумовлюють динамічний характер ринку мобільних додатків та свідчать про постійне прагнення розробників до інновацій та покращень для задоволення ростучих потреб сучасних користувачів.

1.2. Аналіз потреб користувачів

Головна мета додатку - задовольнити потреби звичайних любителів кіно, які насолоджуються переглядом фільмів та серіалів у вільний час. З інтенсивним збільшенням кількості фільмів і серіалів на ринку виникла складність відстеження новинок та вибору цікавих творів. У зв'язку з цим зросла потреба в застосунках, які сприяють систематизації інформації, пошуку нового контенту та отриманні персоналізованих рекомендацій.

Основна цільова аудиторія додатку - звичайні користувачі які полюбляють переглядати кінематографічний контент у свій вільний час. З ростом кількості фільмів і серіалів на ринку стало важко відстежувати новинки та знаходити цікаві твори. Тому зросла і потреба у застосунках, які допомагають організовувати інформацію, шукати новий контент і отримувати персоналізовані рекомендації.

Багато користувачів мріють про свій особистий каталог фільмів і серіалів, де можна відзначати переглянуті твори, виділяти улюблені, формувати списки для подальших переглядів та обмінюватися цим з іншими фанатами кіно.

Застосунки для відстеження фільмів і серіалів доступні на різних платформах, таких як смартфони, планшети або SmartTV. Це надає користувачам можливість зручно отримувати доступ до свого каталогу у будь-якому місці і в будь-який час. Такий аспект підсилює зручність та доступність для широкого кола аудиторії.

1.3. Аналіз існуючих мобільних застосунків з відслідковування відео-контенту

На момент написання роботи існує ціла низка мобільних додатків для відслідковування відео-контенту, які відповідають потребам і цілям користувачів. Кожен додаток має свої унікальні особливості та функції, що робить його адаптивним для різних потреб. На етапі аналізу доцільно оглянути та дослідити існуючі аналоги проектів, які безпосередньо або опосередковано включають функціонал концепції мобільних застосунків з відслідковування відео-контенту.

Для порівняльного огляду були обрані такі додатки:

1. TV Time. Кількість завантажень додатку 10 млн+.
2. Hobi: TV Series Tracker. Кількість завантажень додатку 1 млн+.
3. CLZ Movies. Кількість завантажень додатку 100 тис+.

Кожен застосунок є додатком відстеження та управління переглядом телевізійних програм, серіалів та фільмів. Усі додатки орієнтовані на роботу з контентом в мережі, окрім останнього, CLZ Movies, де акцент робиться на фізичні носії. В ході аналізу продуктів було досліджено їх функціонал, зручність інтерфейсу та використання. Нижче наведено позитивні та негативні моменти використання кожного з обраних додатків.

1.3.1. Мобільний додаток TV Time

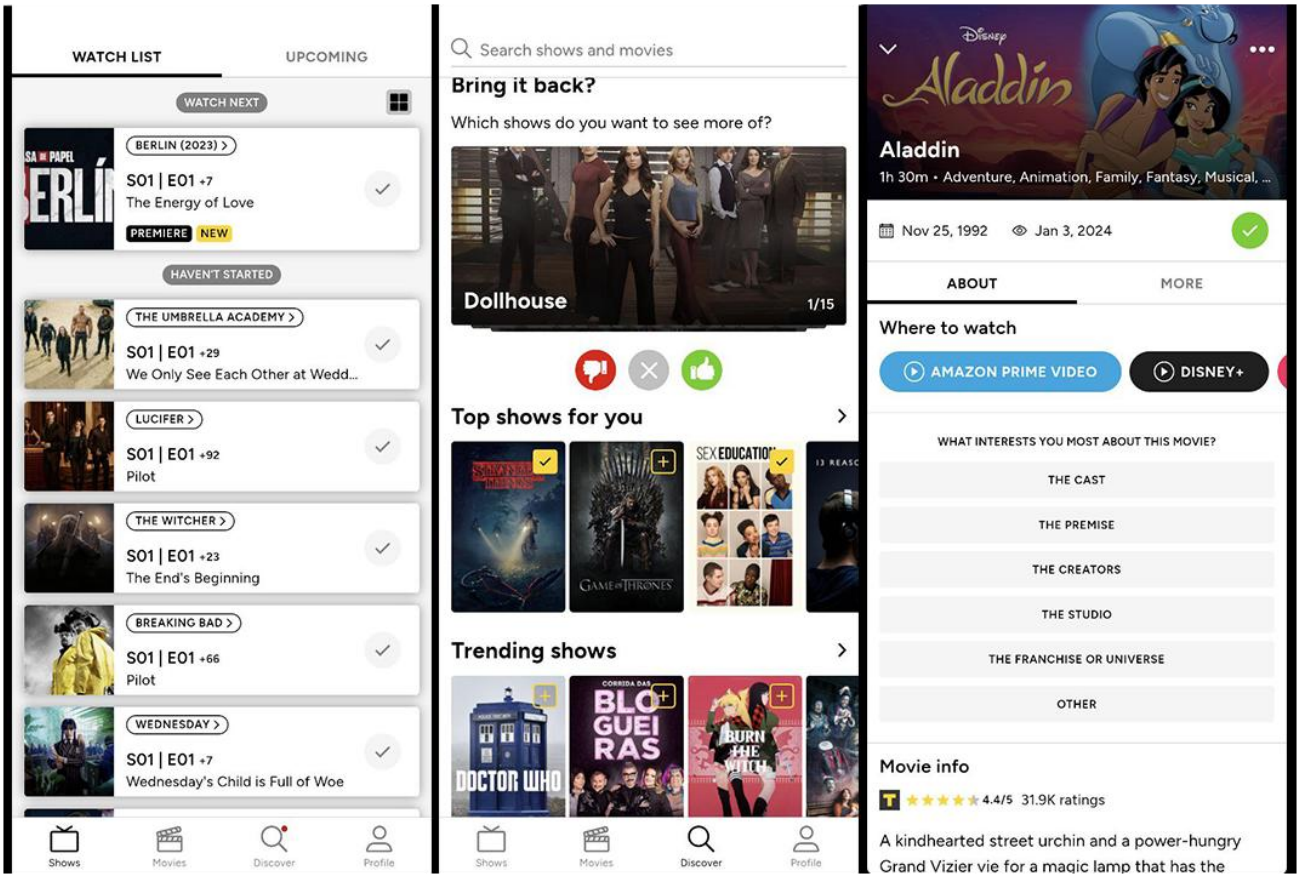


Рис. 1.3.1. інтерфейс додатку TV Time

З плюсів можна відзначити:

- Обширна інформація про кінокартини.
- Інформація на якій платформі доступний перегляд.
- Календар трансляцій.
- Відгуки користувачів.
- Персональні рекомендації.
- Статистика переглядів та досягнення.

До мінусів можна віднести:

- Обов’язкова реєстрація для функціонування додатку.
- Менш повний опис до деяких старих або менш відомих шоу.
- Реклама та монетизація.
- Немає вибору мови інтерфейсу в додатку або через налаштування системи. Відсутня підтримка української мови.

1.3.2. Мобільний додаток Hobi: TV Series Tracker

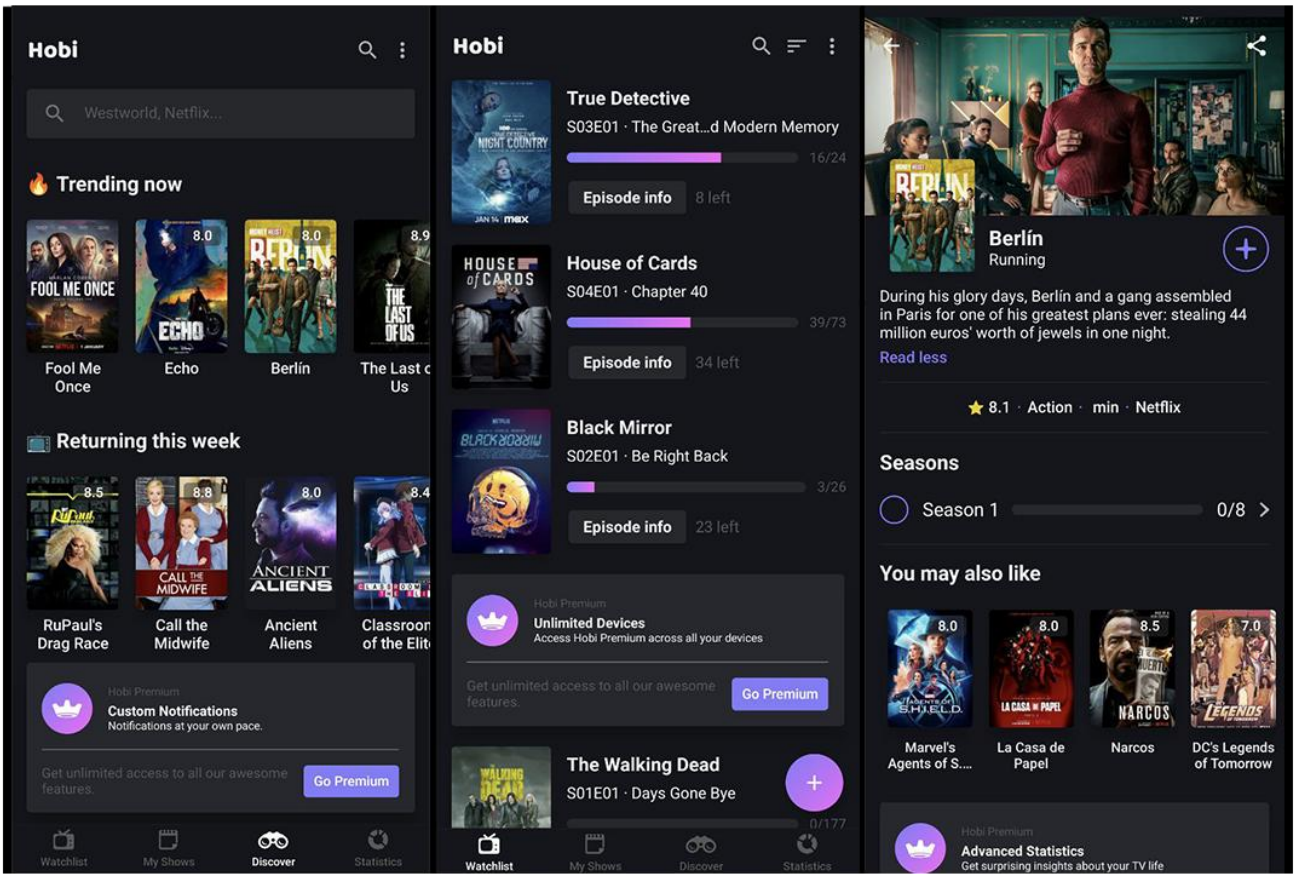


Рис. 1.3.2. інтерфейс додатку Hobi: TV Series Tracker

З плюсів можна відзначити:

- Календар трансляцій.
- Оцінки користувачів.
- Сповіщення про нові картини або епізоди.
- Статистика та аналітика переглядів.
- Персональні рекомендації.

До мінусів можна віднести:

- Нав’язлива реклама купівлі преміум акаунту.
- Куций опис фільмів та серіалів.
- Немає вибору мови інтерфейсу в додатку або через налаштування системи. Відсутня підтримка української мови.

1.3.3. Мобільний додаток CLZ Movies

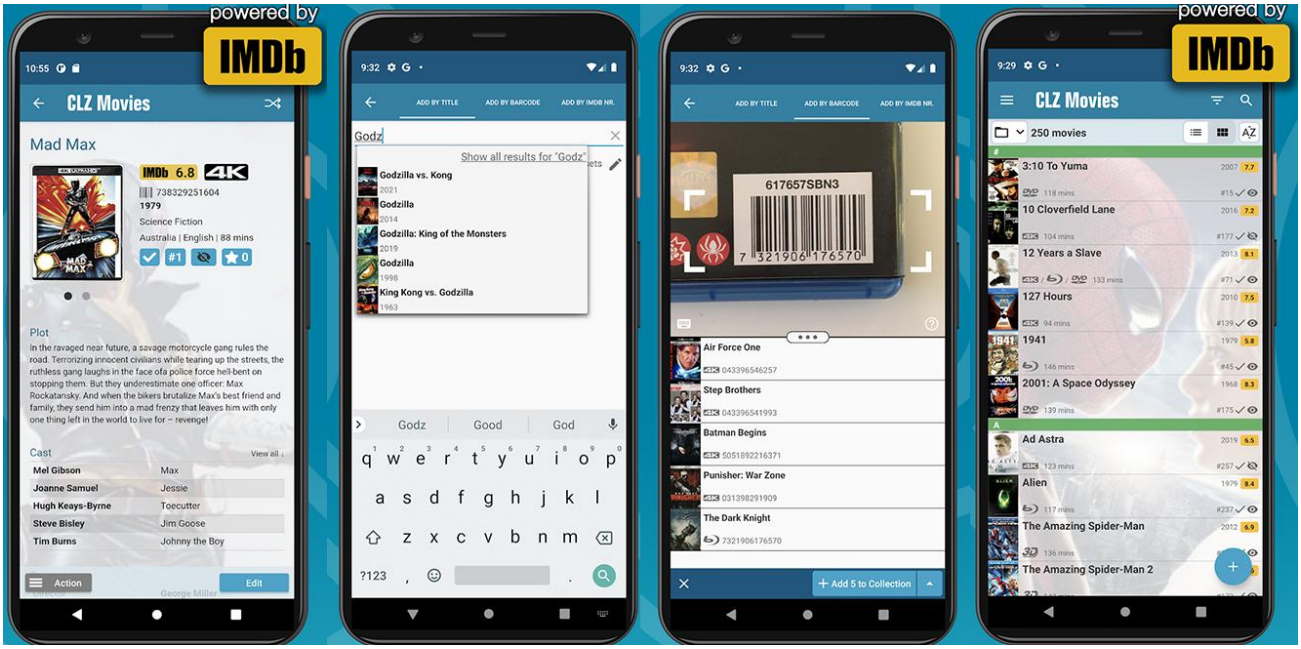


Рис. 1.3.3. інтерфейс додатку CLZ Movies

З плюсів можна відзначити:

- Організація колекції.
- Додавання фільмів (сканування штрих-коду або введення даних вручну).
- Інформація про фільми.
- Синхронізація та резервне копіювання даних.

До мінусів можна віднести:

- Додаток можна використовувати лише оформивши підписку.
- Застарівший інтерфейс.
- Орієнтованість на фізичні копії, ніж на потокові сервіси.
- Немає вибору мови інтерфейсу в додатку або через налаштування системи. Відсутня підтримка української мови.

Отже, ми бачимо що існують додатки для різних прошарків користувачів, з різним функціоналом та перевагами. Проте вони мають свої недоліки і хотілось би мати більш зручний та універсальний додаток у користуванні.

Висновки до розділу

У першому розділі було проведено аналіз потреб користувачів. Визначені причини зростання потреб користувачів у подібних програмних системах. Також було проведено аналіз подібних додатків з відслідковування відео-контенту, та визначені їхні основні переваги та недоліки.

РОЗДІЛ 2.
ВИМОГИ ДО ЗАСТОСУНКУ «SHOW & MOVIE TRACKER»

2.1. Нефункціональні вимоги

Нефункціональні вимоги описують програмні та системні вимоги, вимоги до дизайну інтерфейсу , а також визначають показники часу роботи, захисту даних і досягнення якості з урахуванням рекомендацій використовуваного стандарту.

Мобільний застосунок для відстеження відео-контенту повинен надавати користувачам інтуїтивно зрозумілий та естетичний інтерфейс, спрощений для зручного використання. Передбачається висока швидкість та ефективність роботи застосунку, з уникненням відчуття затримок чи лагів.

У контексті дизайну, важливо враховувати естетичність та інтуїтивність, щоб користувачі могли легко взаємодіяти з додатком. Безпека та конфіденційність даних також мають велике значення, і додаток повинен забезпечувати заходи для їх захисту.

Крім того, необхідно враховувати ефективні механізми пошуку та фільтрації відео-контенту, щоб користувачі могли легко знаходити та відслідковувати фільми та серіали за їхніми індивідуальними вподобаннями. Взаємодія з соціальними мережами та можливість обговорення контенту з іншими користувачами також є важливою вимогою.

Узагальнити нефункціональні вимоги до додатку можна таким чином:

- Продуктивність. Забезпечення швидкого доступу до функціоналу - запорука задоволення очікувань користувачів. Час завантаження програми не повинен перевищувати 3 секунди.
- Безпека. Застосунок повинен захищати інформацію та дані так, щоб невповноважені суб'єкти або системи не могли змінювати їх, а вповноважені суб'єкти не отримували відмову для доступу до них.
- Доступність. Застосування повинне бути доступним всім користувачам, які бажають його завантажити на свій мобільний пристрій.
- Зручність і простота. Мобільний застосунок повинен бути легко освоєним у використанні. Це забезпечує користувачу можливість його експлуатувати та керувати ним.
- Масштабованість. Програма має підтримувати збільшення кількості користувачів без значного зниження продуктивності.
- Надійність. Мобільний застосунок повинен підтримувати певну працездатність в заданих умовах.
- Сумісність. Програма має коректно працювати різних пристроях обраної операційної системи.
- Сумісність із різними дозволами екранів. Програма має коректно відображатися на екранах різних пристроїв з різною роздільною здатністю.
- Чуйність інтерфейсу. Час відповіді на дії користувача не повинен перевищувати 0,5 секунд.
- Використання обмежених ресурсів. Додаток не повинен перевищувати встановлені ліміти для використання оперативної пам'яті та процесорного часу.
- Цілісність даних. Застосування повинне легко отримувати доступ до будь-яких даних, що потрібні для його коректної роботи, які не виходять за межі зумовлених.

- Ефективність мережної взаємодії. Оптимізація запитів мережі для мінімізації часу завантаження даних.
- Оновлення та підтримка. Надання регулярних оновлень із виправленням помилок та новим функціоналом.

2.2. Функціональні вимоги

Функціональні вимоги - це вимоги до програмного забезпечення, які описують поведінку системи. Функціональні вимоги визначають конкретні функції та можливості які повинні бути реалізовані в проекті.

Функціональні вимоги для мобільного застосунку з відстеження відео-контенту включають забезпечення можливості реєстрації та створення облікового запису для кожного користувача. Застосунок повинен дозволяти користувачам додавати відео в свій каталог, позначати їх статус перегляду, а також визначати особисті враження та рейтинг.

Одна з ключових функцій - ефективний механізм пошуку та фільтрації відео-контенту за різними параметрами, такими як жанр, рік випуску, рейтинг тощо. Функціональність додатку також повинна включати можливість отримання персоналізованих рекомендацій на основі історії перегляду та вподобань. Передбачається можливість створення користувачами персональних списків для подальшого перегляду, а також обміну ними з іншими користувачами.

Функціональні вимоги для даного проекту є:

- Реєстрація та аутентифікація. Користувачі можуть створити обліковий запис із особистими даними або пропустити цей крок.
- Профіль користувача. Користувачі мають персоналізований профіль, де зберігаються їхні уподобання та історія переглядів.
- Відстеження переглядів. Користувачі можуть відзначати переглянуті фільми та серіали.

- Створення списку бажаного перегляду. Користувачі можуть додавати до бажаного фільми та серіали для перегляду.
- Тематичні плейлисти та колекції. Користувачі можуть створювати списки кінокартин та переглядати відкриті списки інших користувачів.
- Рекомендації та алгоритми. Система надає персональні рекомендації на основі переваг користувача.
- Огляди та рецензії. Користувачі можуть залишати свої огляди та рецензії на переглянуті фільми та серіали.
- Інтеграція із зовнішніми джерелами. Відображення актуальної інформації про фільми, серіали та акторів.
- Повідомлення та новини. Користувачі отримують повідомлення про новинки, релізи та події, пов'язані з їхніми інтересами.
- Календар. Календар виходу нових серій для відстежуваних серіалів.
- Інтерфейс пошуку. Користувачі можуть здійснювати пошук фільмів та серіалів за різними критеріями.
- Персональна статистика. Можливість подивитися свою статистику переглядів.

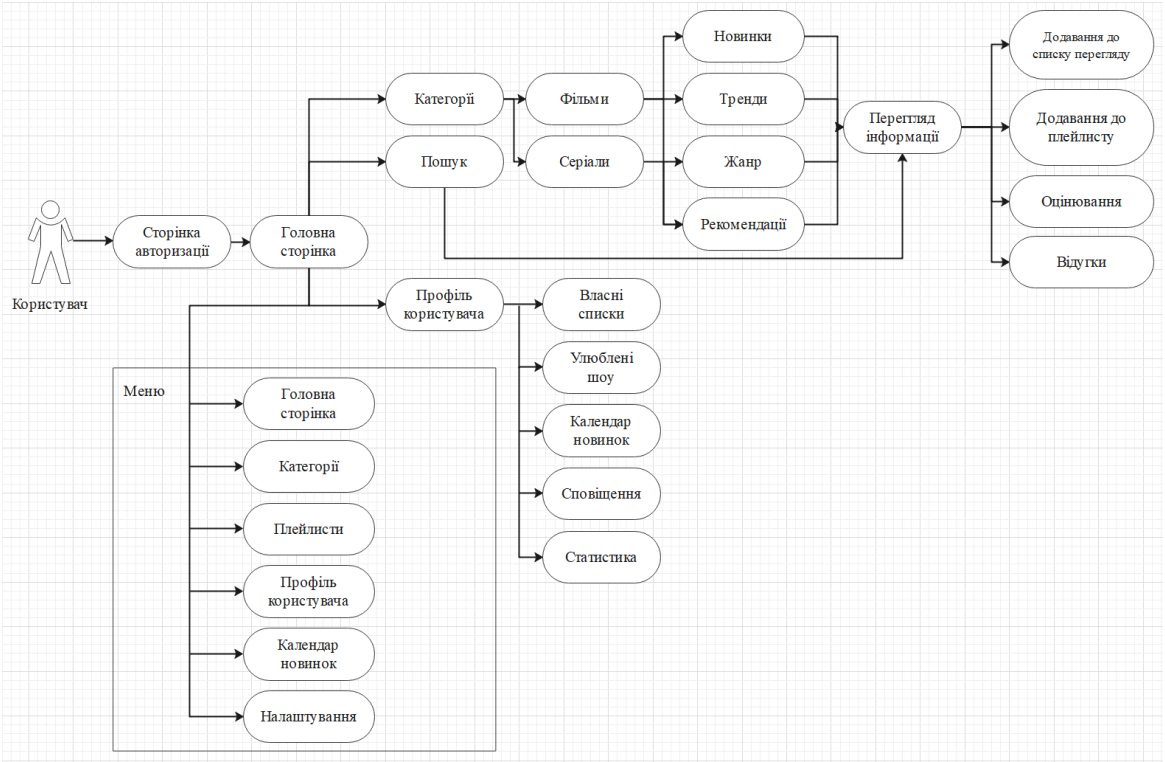


Рис. 2.2. Діаграма прецедентів застосунку «Show & Movie Tracker»

2.3. Програмні вимоги

Програмні вимоги - це набір характеристик та параметрів, які необхідні для правильної роботи програми або комп'ютерної системи. Ці вимоги визначають операційну систему застосунку, середовище та мову розробки, систему управління базами даних та інші компоненти для коректного функціонування програми.

Мобільний застосунок розробляється для пристроїв на базі Android OS. При успішності застосунку відбудеться подальше розширення на iOS та інші функціональні системи.

Мова розробки обрана Java; буде використана СУБД система PostgreSQL, з мовою запитів SQL; застосунок матиме підключення до зовнішнього ресурсу TMDB API; запуск прототипу відбуватиметься на локальному сервері. Середовище розробки було обрано IntelliJ Idea з використанням можливостей Android Studio.

Мова розробки Java

Java - це високорівнева, об'єктно-орієнтована мова програмування, що базується на принципах об'єктно-орієнтованого підходу. Це сприяє ефективному створенню модульних та повторно використовуваних блоків коду. Однією з ключових особливостей Java є можливість розробки програм на різні платформи, які можуть функціонувати на різних операційних системах, що робить її широко використовуваною у сфері розробки програмного забезпечення.

Java виділяється серед багатьох інших мов програмування завдяки особливості, за якої її програми компілюються у байт-код. Під час виконання програми байт-код інтерпретується віртуальною машиною Java. Цей підхід забезпечує велику переносимість, високий рівень безпеки та можливість масштабування для Java-програм. Іншими словами, байт-код створює проміжний шар, що дозволяє програмам працювати на будь-якій платформі, на якій встановлено відповідну віртуальну машину Java, забезпечуючи високий рівень універсальності.

У мові програмування Java всі об'єкти успадковують базову поведінку та властивості від головного об'єкта. Однією з ключових особливостей віртуальної машини Java є те, що помилки, які можуть виникнути під час виконання програми, не призводять до повного збою системи. До того ж, в середовищі виконання існують інструменти, які приєднуються автоматично і, в разі виникнення виключення, фіксують інформацію з пам'яті для подальшого аналізу програми.

Ці автоматизовані інструменти обробки виключень надають важливу інформацію щодо помилок у програмах на Java. Крім того, для ефективного управління пам'яттю під час життєвого циклу об'єкта використовується автоматичний збирач сміття. Коли на об'єкт не залишається посилань, збирач сміття автоматично видаляє його з пам'яті.

Важливо зауважити, що хоча автоматичний збирач сміття раціонально керує пам'яттю, витік пам'яті може виникнути, якщо програміст створює посилання на об'єкти, які вже не потрібні, наприклад, в діючих контейнерах.

Java визначається суворою типізацією, що означає, що кожна змінна та вираз мають тип, визначений на етапі компіляції. Типи даних у Java поділяються на дві основні категорії: прості та вказівникові. Прості типи включають булевий, числові та символьний, в той час як вказівникові типи включають класи, інтерфейси та масиви.

Прості типи дозволяють представляти базові дані, такі як істина/хиба, цілі числа чи символи. З іншого боку, вказівникові типи складаються з класів, які визначають об'єкти або масиви. Значенням вказівникового типу є вказівник на конкретний об'єкт або екземпляр класу.

Такий підхід до типізації не лише забезпечує безпеку типів, але й допомагає під час розробки, оскільки він дозволяє виявляти помилки на етапі компіляції, що сприяє більш високій надійності програм.

Середовище розробки

IntelliJ IDEA представляє собою інтегроване середовище розробки, призначене для використання різними мовами програмування, але основний

фокус робиться на мові програмування Java. За допомогою IntelliJ IDEA можна ефективно розробляти програми на Java завдяки численним потужним інструментам, таким як автоматичне завершення коду, можливості рефакторингу, аналіз коду, інтегрована система збірки та інші.

Окрім підтримки Java, IntelliJ IDEA також надає можливість працювати з іншими мовами програмування, такими як Kotlin, Groovy, Scala, JavaScript, TypeScript, HTML, CSS, SQL і інші. Це середовище розробки сприяє підтримці коду в чистому та ефективному стані завдяки розширеному набору інструментів рефакторингу. Ці інструменти дозволяють вам полегшити та покращити код, не втрачаючи його функціональності. Такий підхід допомагає підтримувати високий стандарт програмного забезпечення та забезпечує ефективну розробку проектів.

IntelliJ IDEA допомагає розробникам не лише засвоювати різні мови програмування, але й ефективно використовувати різноманітні фреймворки та технології. Серед підтримуваних фреймворків і технологій можна відзначити такі, як Spring, Hibernate, JavaFX, Android SDK та інші, що значно спрощує розробку застосунків на їхній основі.

IntelliJ IDEA також забезпечує зручність тестування для розробників, які працюють з Android-додатками. Зокрема, вона підтримує вбудований Android емулятор, що дозволяє проводити ефективне тестування застосунків безпосередньо в середовищі розробки. Крім того, розробники можуть використовувати свої власні пристрої для тестування, що робить процес розробки та налаштування більш гнучким та адаптованим до їхніх потреб.

Android Studios

Android Studio - офіційний інструмент для створення мобільних додатків, призначений для операційної системи Android. Це інтегроване середовище розробки, яке забезпечує розробникам комплексні можливості та інструменти для ефективної та зручної роботи.

Однією з ключових функцій Android Studio є наявність інтегрованих інструментів для розробки, таких як дизайнер інтерфейсів, редактор XML, дебагер, профілер продуктивності та система збірки. У програмі вбудований візуальний дизайнер, який дозволяє створювати інтерфейси користувача шляхом простого перетягування та розміщення елементів.

Також Android Studio надає потужну систему збірки та управління залежностями. Це дозволяє розробникам легко додавати бібліотеки та сторонні компоненти до свого проекту, спрощуючи тим самим роботу з розширенням функціональності додатків. Загалом, Android Studio створено з урахуванням потреб розробників для надання повноцінного інструменту для творення якісних мобільних застосунків.

Поміж потужного редактора коду та інструментів для розробників IntelliJ, Android Studio пропонує ряд додаткових функцій, які значно підвищують продуктивність при створенні програм для Android. До цих функцій включаються:

- Гнучка система побудови на основі Gradle, яка дозволяє ефективно налаштовувати та керувати процесом збірки проектів.
- Швидкий і багатофункціональний емулятор, який дозволяє тестувати додатки на різних віртуальних пристроях Android.
- Єдине середовище, де можна розробляти для всіх пристроїв Android, забезпечуючи єдність та зручність розробки.
- Можливість надсилання змін коду та ресурсів до запущеної програми без перезапуску, що спрощує налаштування та вдосконалення додатків.
- Використання шаблонів коду та інтеграція з GitHub для спрощення розробки та імпортування зразків коду.
- Широкий набір інструментів та основ тестування для забезпечення якості та надійності додатків.
- Інструменти Lint для виявлення проблем продуктивності, зручності використання, сумісності версій та інших аспектів.

- Вбудована підтримка Google Cloud Platform, що полегшує інтеграцію з Google Cloud Messaging та App Engine для розширення функціональності додатків.

Java Spring Framework

При розробці взаємодії з сервером використовувався Java код, і для його реалізації використано Spring Framework. Spring Framework – це високорівневий фреймворк для програмування та інверсії контейнера управління, спрямований на платформу Java. Відзначається тим, що його основні функції можуть бути використані в будь-яких додатках Java, а також є розширення для створення веб-додатків поверх платформи Java Enterprise Edition.

Spring Framework відзначається широким спектром можливостей для полегшення розробки та управління компонентами додатків. Основні характеристики та можливості цього фреймворку включають інверсію контролю, впровадження залежностей, модульність, аспектно-орієнтоване програмування, підтримку транзакцій, інтеграцію з різними технологіями та підтримку веб-розробки.

Однією з ключових особливостей є інверсія контролю, що дозволяє фреймворку приймати на себе відповідальність за управління життєвим циклом об'єктів, забезпечуючи більшу гнучкість та простоту в розробці. Впровадження залежностей сприяє створенню менш залежних компонентів, зробивши код більш придатним для тестування.

Модульність Spring дозволяє розробникам вибирати лише ті частини фреймворку, які їм потрібні, підтримуючи легкість та ефективність в розробці. AOP надає можливість виділяти та відокремлювати аспекти додатку, спрощуючи обробку справ зі спільною логікою.

Не вимагаючи конкретної моделі програмування, Spring Framework став вкрай популярним у Java-спільноті, особливо як доповнення до моделі Enterprise JavaBeans. Цей фреймворк відкритого коду відкриває широкі

можливості для розробників, надаючи зручні інструменти для створення різноманітних додатків та взаємодії з сервером на базі платформи Java.

Вимоги для супроводження системи

З огляду на безпеку даних, необхідно застосовувати передові практики шифрування для зберігання конфіденційної інформації, використовувати механізми автентифікації та авторизації для захисту від несанкціонованого доступу.

Застосунок повинен підтримувати одночасне підключення до серверів багатьох користувачів. SQL запити повинні бути оптимізовані для швидкої роботи. Необхідно також забезпечити можливість регулярного резервного копіювання та відновлення бази даних та інших важливих ресурсів додатку для забезпечення безпеки даних та швидкого відновлення в разі необхідності.

Важливим є також здатність додатку підтримувати розширення та оновлення без негативного впливу на користувачів, а також система моніторингу та діагностики для оперативного виявлення та усунення проблем.

Сумісність з різними версіями ОС Android вимагає ретельного тестування та виправлення помилок для забезпечення стабільної роботи додатку. Код програми повинен бути гнучким, для його легкої зміни, якщо потрібно буде для переходу до iOS або для функціонального або графічного розширення.

Висновки до розділу

У цьому розділі були описані основні функціональні вимоги для реалізації прототипу додатку з відстежування відео-контенту. Також були сформовані нефункціональні вимоги до застосунку, були обрані середовище, мова програмування та додаткові інструменти розробки. Розробка передбачає подальше розширення на iOS та інші функціональні системи в майбутньому.

РОЗДІЛ 3.
СТРУКТУРА ЗАСТОСУНКУ «SHOW & MOVIE TRACKER»

3.1. Взаємодія частин програмного забезпечення застосунку

У даному проекті використовуються мови програмування Java, PostgreSQL, XML для розмітки сторінок, а також зовнішня база даних TMBD API. У проекті Java використовується для реалізації різноманітних завдань, включаючи створення моделей об'єктів, які представляють дані, контролерів, які обробляють запити від користувачів, та мікросервісів, які взаємодіють з базою даних. Наприклад, за допомогою Java можна створити об'єкт користувача, який містить в собі інформацію про користувача, таку як ім'я, електронна адреса тощо.

Крім того, Java використовується для налаштування з'єднання з базою даних PostgreSQL та виконання запитів до неї, щоб отримати або зберегти дані. Після отримання даних від бази даних, Java може обробити ці дані та підготувати їх до відображення на сторінці.

Після цього, за допомогою XML, інформація з Java об'єкту вставляється у відповідні місця на сторінці, використовуючи ідентифікатори. Такий підхід дозволяє ефективно керувати даними та забезпечує їх правильне відображення на веб-сторінці для користувача.

TMDB API - це набір програмних інтерфейсів, який надає доступ до бази даних фільмів, телевізійних шоу і акторів.

Цей API дозволяє отримувати доступ до різноманітної інформації про фільми та телевізійні шоу, включаючи дані про рейтинги, описи, режисерів, акторів, постери та інше. Використання зовнішньої бази даних дуже спрощує та прискорює роботу створення проекту.

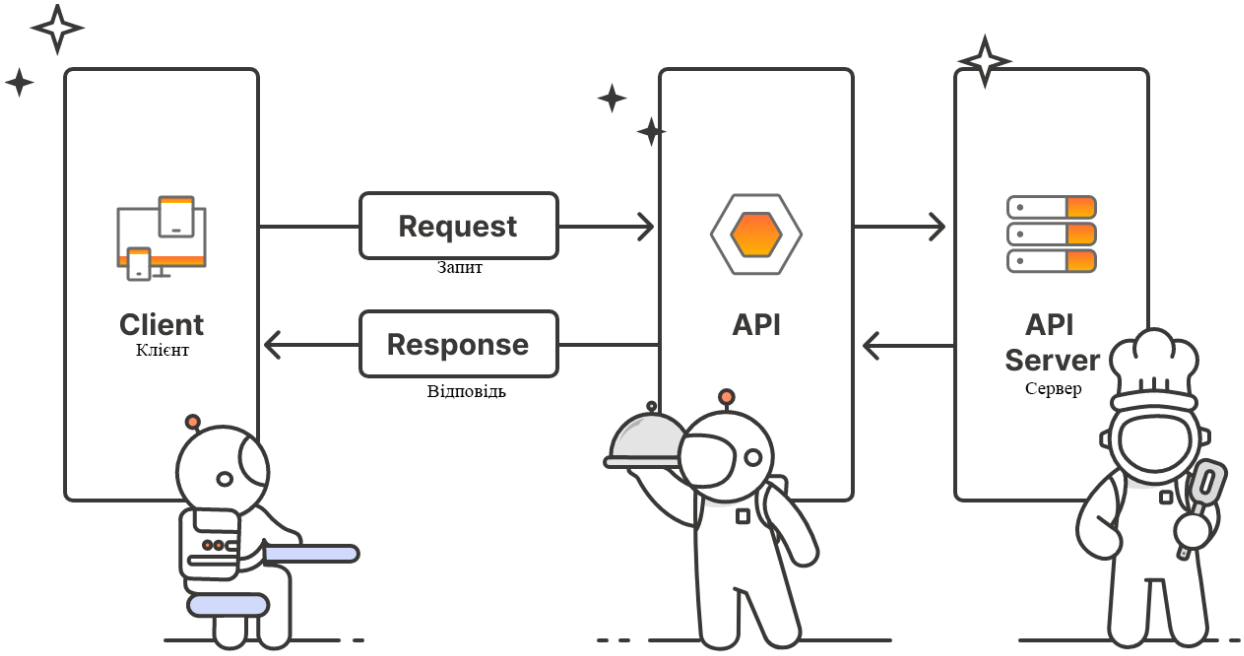


Рис. 3.1. Принцип взаємодії з API

Код застосунку складається з:

- Java – це backend (бекенд) – це код який знаходиться на серверній частині, тобто не доступний для користувача, відповідає за обробку даних, логіку додатку та взаємодію з базою даних.
- PostgreSQL – база даних де буде зберігатися інформація про користувачів, що дозволить оптимізувати запити та забезпечити безпеку даних.
- XML – frontend (фронтенд) – це частина ПЗ, що знаходиться на клієнтській стороні і відповідає за візуальний та інтерактивний інтерфейс додатку.

Взаємодія цих мов програмування дозволяє всій системі працювати відповідно до потреб користувача. Ці елементи узгоджено працюють, як у мобільному додатку, так і на локальному сервері. Мова Java використовується як на мобільних пристроях, так і на локальному сервері, забезпечуючи єдність

та сумісність у системі. База даних PostgreSQL розташована на локальному сервері, і мобільний додаток може взаємодіяти з нею через мікросервіси, надсилаючи запити та отримуючи відповіді для оптимальної роботи програми.

3.2. Архітектура мобільних додатків

Архітектура мобільних додатків - це структура та організація програмного коду мобільного додатку, яка визначає його функціональність, ефективність та можливості розширення. Така архітектура визначає, як різні компоненти додатку взаємодіють між собою і як вони організовані.

Архітектура мобільних додатків включає ряд складових, які забезпечують їхню функціональність та ефективність.

На клієнтській частині, або тій, що працює на боку користувача, розміщується користувацький інтерфейс, який відповідає за відображення графічного інтерфейсу та взаємодію з користувачем. Тут також розташовується бізнес-логіка, яка містить код, що визначає логіку додатку та взаємодію з сервером або базою даних.

Серверна частина забезпечує обробку запитів від клієнтів та взаємодію з базою даних. Вона включає в себе сервер, який обробляє запити, і базу даних, яка зберігає дані, необхідні для роботи додатку.

API дозволяє клієнтській та серверній частинам взаємодіяти між собою, визначаючи, які дані можна отримати або відправити та як вони повинні оброблятися.

Моделі даних визначають структуру даних, що використовується в додатку, такі як об'єкти, класи або структури, що представляють дані та їх взаємозв'язки.

Система авторизації та безпеки забезпечує захист даних та контроль доступу до функцій додатку, включаючи механізми аутентифікації та авторизації користувачів.

Додатки можуть використовувати різні архітектурні підходи, такі як Model-View-Controller, Model-View-ViewModel, Clean Architecture, або Flutter/Dart має свою власну архітектуру, названу Flutter Architecture. Кожен з цих

архітектурних підходів має свої переваги і недоліки, і може бути вибраний залежно від конкретних потреб проекту та вимог до нього.

Model-View-Controller розділяє додаток на модель (яка представляє дані та бізнес-логіку), представлення (яке відображає інтерфейс користувача) та контролер (який обробляє вхідні дані та керує взаємодією між моделлю та представленням).

Model-View-ViewModel є модифікацією Model-View-Controller, де модель відокремлена від представлення і замість цього використовується модель-вигляд-представлення, де модель взаємодіє з представленням через проміжний об'єкт - ViewModel. Цей підхід дозволяє краще управляти станом додатку та полегшує тестування.

Clean Architecture ставить основну увагу на розділення додатку на рівні відповідальності (бізнес-логіка, представлення, інтерфейси користувача, зовнішні інтерфейси, тощо) і забезпеченням, щоб залежності між цими рівнями були однонаправленими.

Flutter Architecture - це специфічний архітектурний підхід, який використовується в мобільній розробці за допомогою Flutter/Dart. Він має вбудовані інструменти й шаблони для створення ефективних та розширюваних додатків, що працюють на крос-платформеному фреймворку Flutter.

3.3. Архітектура застосунку

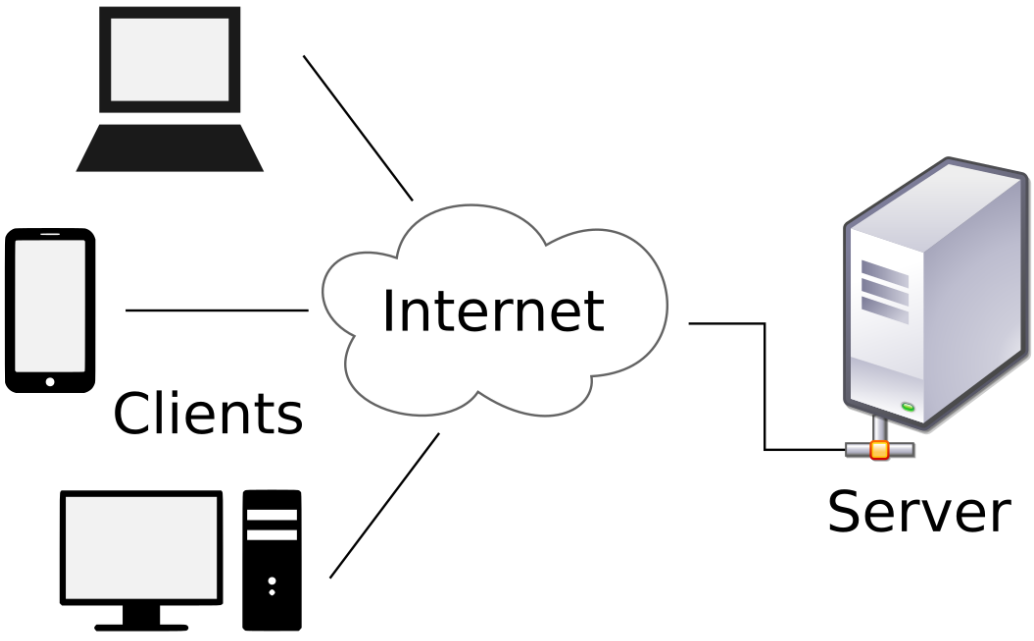
Для цього проекту планується використання шаблону Model-View-ViewModel у поєднанні з клієнт-серверною архітектурою. Цей підхід дозволить ефективно розділити логіку програми на компоненти, які відповідають за обробку даних (модель), представлення інформації для користувача (вигляд) та проміжний шар для управління даними та інтерфейсом (представлення). Крім цього, клієнт-серверна архітектура дозволить забезпечити взаємодію між клієнтською та серверною частинами додатку, що сприятиме ефективному обміну даними та забезпечить швидку та стабільну роботу програми.

Рис. 3.3.1. Архітектурний шаблон Model-View-ViewModel

Шаблон MVVM складається з трьох частин:

- Модель (Model) - являє собою основні дані, які необхідні для функціонування додатку.
- Вигляд (View) – це графічний інтерфейс, який включає в себе вікна, кнопки та інші елементи.
- Представлення (ViewModel) – поєднує в собі абстракцію Вигляду з даними з Моделі, які потрібно відобразити.

Клієнт-серверна архітектура – це спосіб організації програмного забезпечення, при якому один або декілька клієнтських пристроїв запитують інформацію від серверного пристрою. Сервер, як правило, відповідає запитаною інформацією або виконує певні операції. Ця архітектура дозволяє ефективно організовувати обмін інформацією між клієнтськими та серверними



компонентами для виконання різних завдань.

Рис. 3.3.2. Клієнт-серверна архітектура

Перевагами такої архітектури являються:

- Гнучкість та масштабованість – така архітектура дозволяє легко додавати або змінювати клієнтів і сервери, а також розподіляти навантаження між ними.
- Легкість обслуговування – розділення функціональності між клієнтами там серверами спрощує розробку та обслуговування програм.
- Безпека – клієнт серверна архітектура забезпечує захист даних і ресурсів на рівні сервера, а також можливість встановлювати різні рівні доступу для різних клієнтів.
- Продуктивність – дана архітектура підвищує продуктивність системи, оскільки сервери можуть виконувати ресурсномісткі операції, а клієнти можуть зосередитися на інтерфейсі користувача і локальній обробці даних.

З недоліків можна виділити:

- Несправність сервера – якщо сервер вийде з ладу, то це може призвести до недоступності мережевих ресурсів або зупинки роботи всієї системи.
- Залежність від мережі - взаємодія між клієнтом і сервером відбувається через мережу, тому необхідно враховувати можливість втрати з'єднання або затримки.
- Складність адміністрування – клієнт-серверна архітектура вимагає кваліфікованого персоналу для налаштування і підтримки серверів, та забезпечення безпеки.
- Безпека даних – така архітектура створює ризик витоку або злому даних, як від користувачів так і зовнішніх атак.

- Вартість – клієнт-серверна архітектура потребує високої вартості мережі та мережевого обладнання, а також витрат на обслуговування й оновлення серверів.

Мобільний застосунок складається з двох проектів – сам застосунок яким буде користуватися користувач, що є «клієнтом» у цій архітектурі, а другий – сам сервер. У даній роботі буде використано локальний сервер, для розгортання якого буде використаний ноутбук та програма Tomcat.

Як приклад в даному застосунку користувач захоче побачити детальну інформацію про обраний фільм, він відкриє список фільмів і при визові методу відкриття вікна детальної інформації буде відправлено запит до сервера, який в свою чергу поверне розширені дані про обрану кінокартину.

3.4. Діаграма класів застосунку

Діаграма класів надає візуальне уявлення структури програми шляхом відображення класів, їх атрибутів та методів. Даний проект складається з трьох пакетів: activities, adapters, domain.

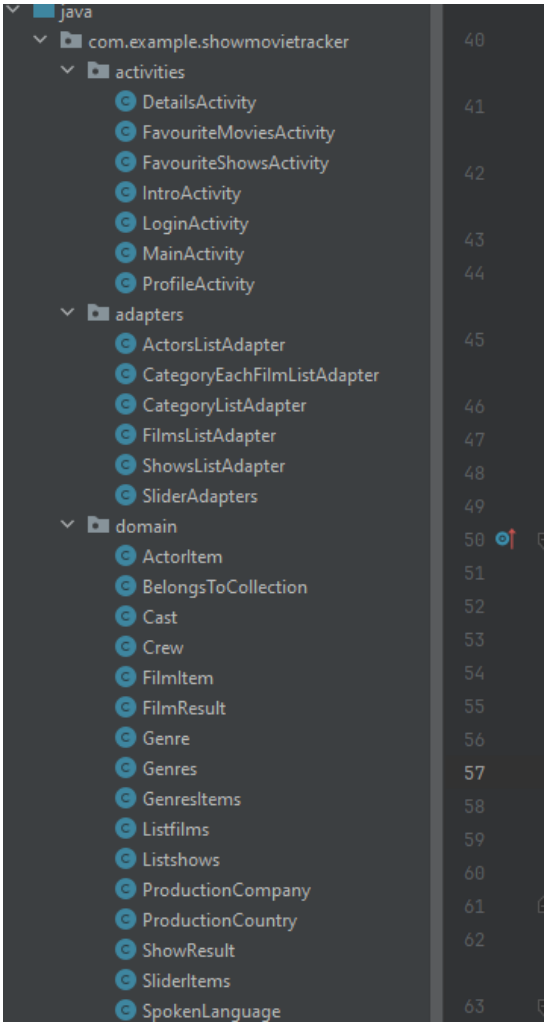


Рис. 3.4.1. Структура застосунку

Діаграма класів adapters описує структуру та взаємозв'язки класів адаптерів, що виступають посередниками між класом Модель і активністю, яка представляє візуальний інтерфейс користувача. Коли об'єкт класу моделі надходить, він обробляється адаптером, який передає дані у відповідні View-об'єкти, що відповідають за відображення цих даних на екрані. Адаптери дозволяють розподілити логіку обробки даних від класу моделі та їх відображення в інтерфейсі користувача, що сприяє підтримці чистоти коду та покращує його розширюваність.

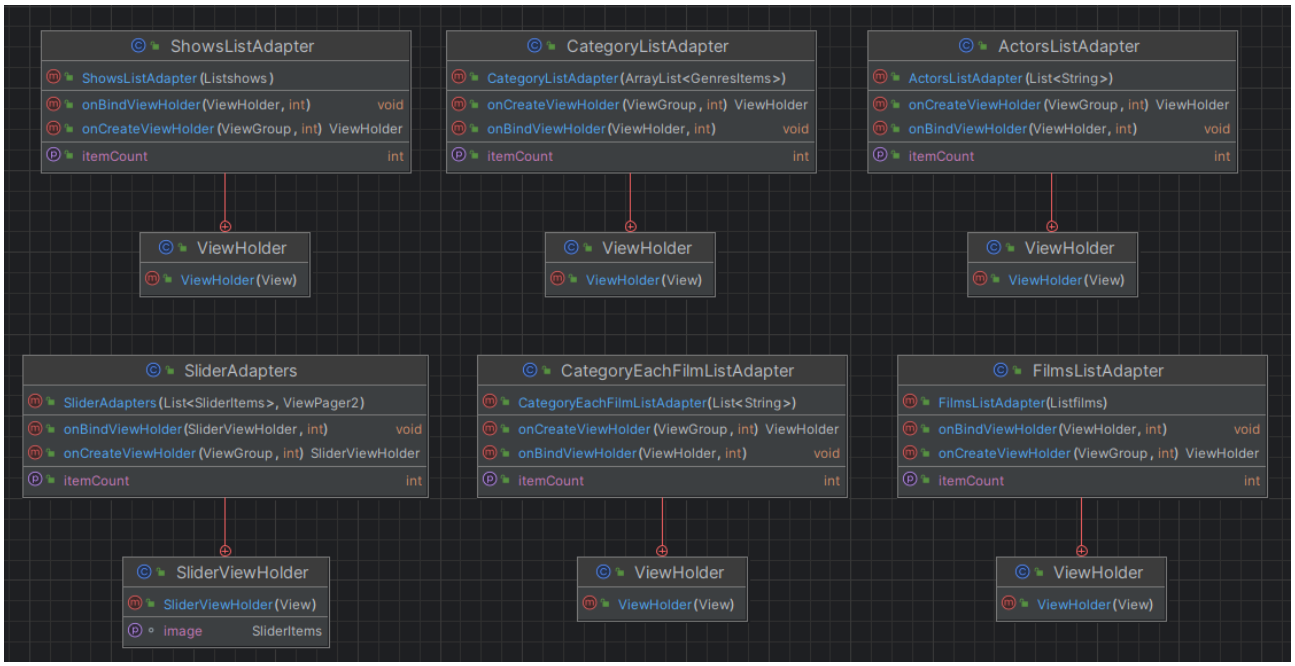


Рис. 3.4.2. Діаграма класів пакету adapters

Діаграма класів domain описує структуру класів, які представляють основні сутності системи, такі як фільми, серіали, жанри та інша додаткова інформація про кінокартини, таку як акторський склад, рейтинг і т. д. Ці класи відповідають за моделювання об'єктів, які використовуються у системі, та забезпечують доступ до них для роботи з іншими компонентами програми. Класи в цій діаграмі також відповідають за взаємодію з зовнішніми джерелами даних, такими як API, для отримання необхідної інформації. Після отримання

даних вони перетворюють цю інформацію в об'єкти відповідного типу, які в подальшому використовуються в компонентах пакету adapters для подальшої обробки та відображення у візуальному інтерфейсі користувача.

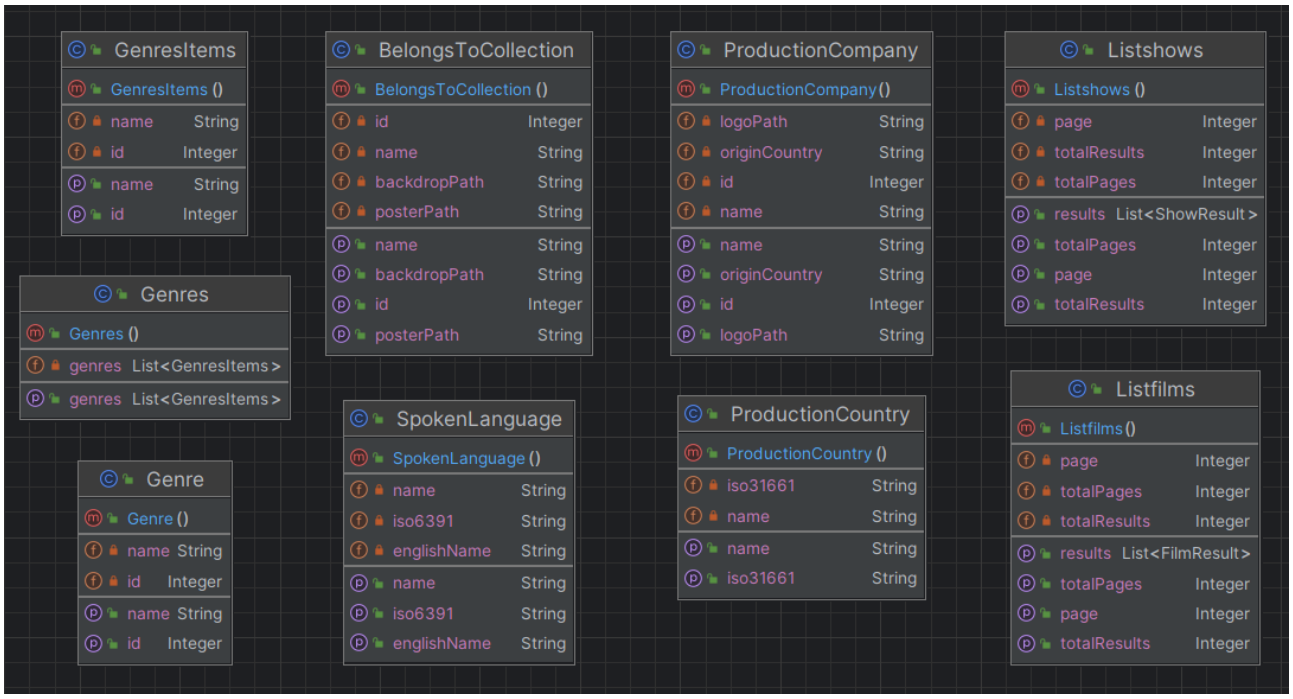


Рис. 3.4.3. Діаграма класів пакету domain

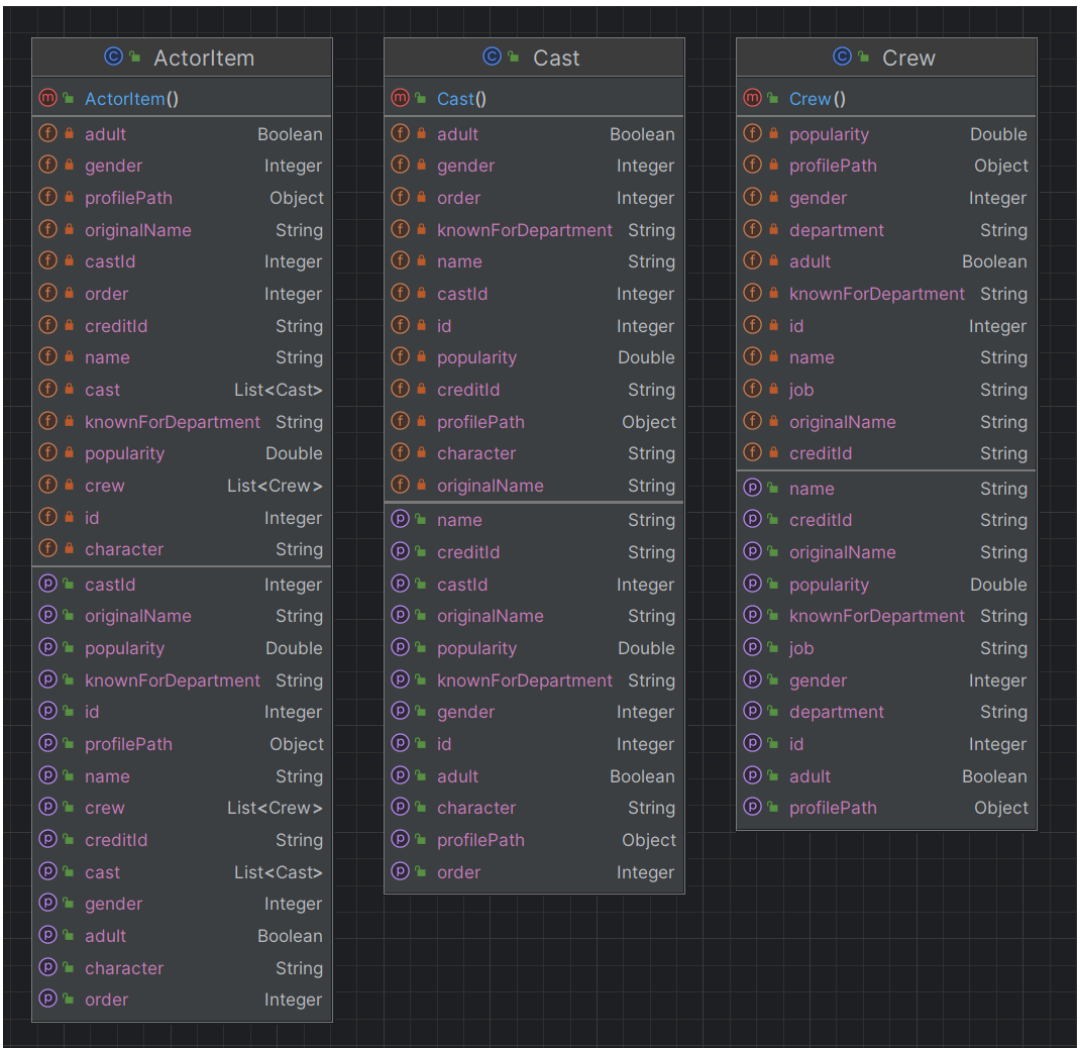


Рис. 3.4.4. Діаграма класів пакету domain

Рис. 3.4.5. Діаграма класів пакету domain

Діаграма класів activity визначає структуру класів активності, які відповідає за відображення кожного окремого об’єкту фільму, серіалу, жанру тощо. Цей клас приймає об’єкти класу від адаптерів, з яких отримує дані, які

необхідні для створення та відображення активності.

Рис. 3.4.6. Діаграма класу activities

3.5. Прототип застосунку

Процес створення мобільного застосунку відбувся за допомогою IntelliJ IDEA з можливостями Android Studio, використовуючи мову програмування

Java. Для тестування функціональних можливостей та інтерфейсу прототипу було обрано мобільний пристрій Google Pixel 5a з операційною системою Android 14.

Прототип було розроблено з метою перевірки концепцій та функціональності програми. Після тестування прототипу будуть виправлені всі виявлені проблеми, як функціональні, так і графічні.

Прототип мобільного застосунку використовує локальний сервер комп'ютера, на якому розміщена логіка мікросервісу, та зовнішню базу даних. Після випуску продукту у реліз, сервер буде замінений на онлайн хостинг, де буде зберігатися вся інформація для застосунку.

Далі, після успішного тестування та модифікації прототипу, застосунок буде випущено до релізу на платформі Google Play Market. Також після успішного запуску застосунку, буде розглянута можливість розробки версії застосунку для операційної системи iOS.

Прототип застосунку орієнтується на обрану мову системи смартфону. Якщо мова системи вказана українська, то вона відповідно і буде основною мовою додатку. Якщо обрано англійську чи будь-яку іншу мову, то за замовчуванням мова застосунку буде обрана англійська.

Початкова сторінка мобільного застосунку.

Після запуску мобільного додатку користувача зустрічає початкове вікно застосунку. На цій сторінці користувач може ознайомитися з стислим описом функціональних можливостей застосунку.

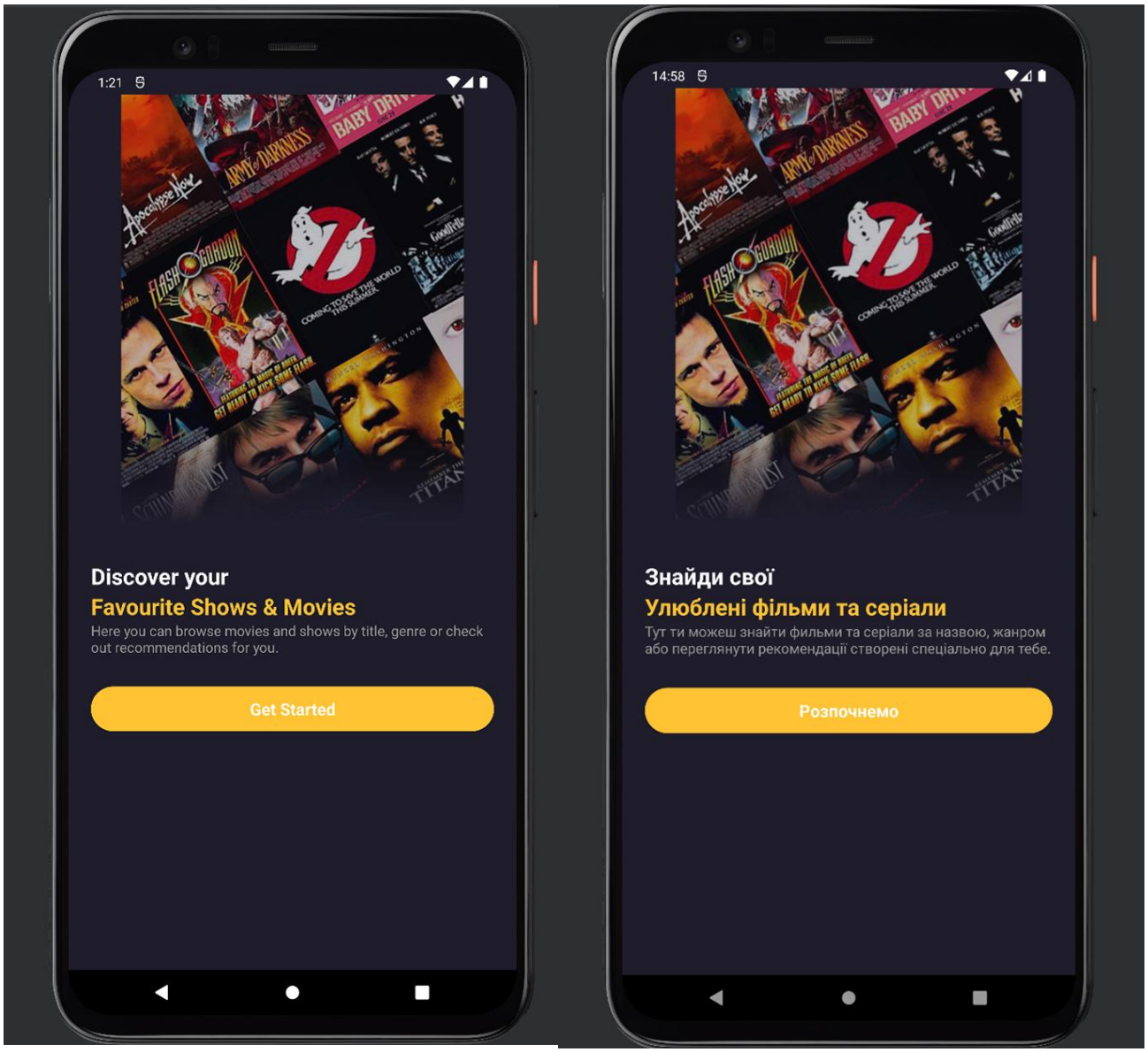


Рис. 3.5.1. Початкова сторінка застосунку на англійські та українській мовах

Сторінка авторизації.

Після початкової сторінки користувача зустрічає сторінка авторизації. На цій сторінці користувач може увійти до існуючого акаунту, зареєструвати новий або відновити забутий пароль.

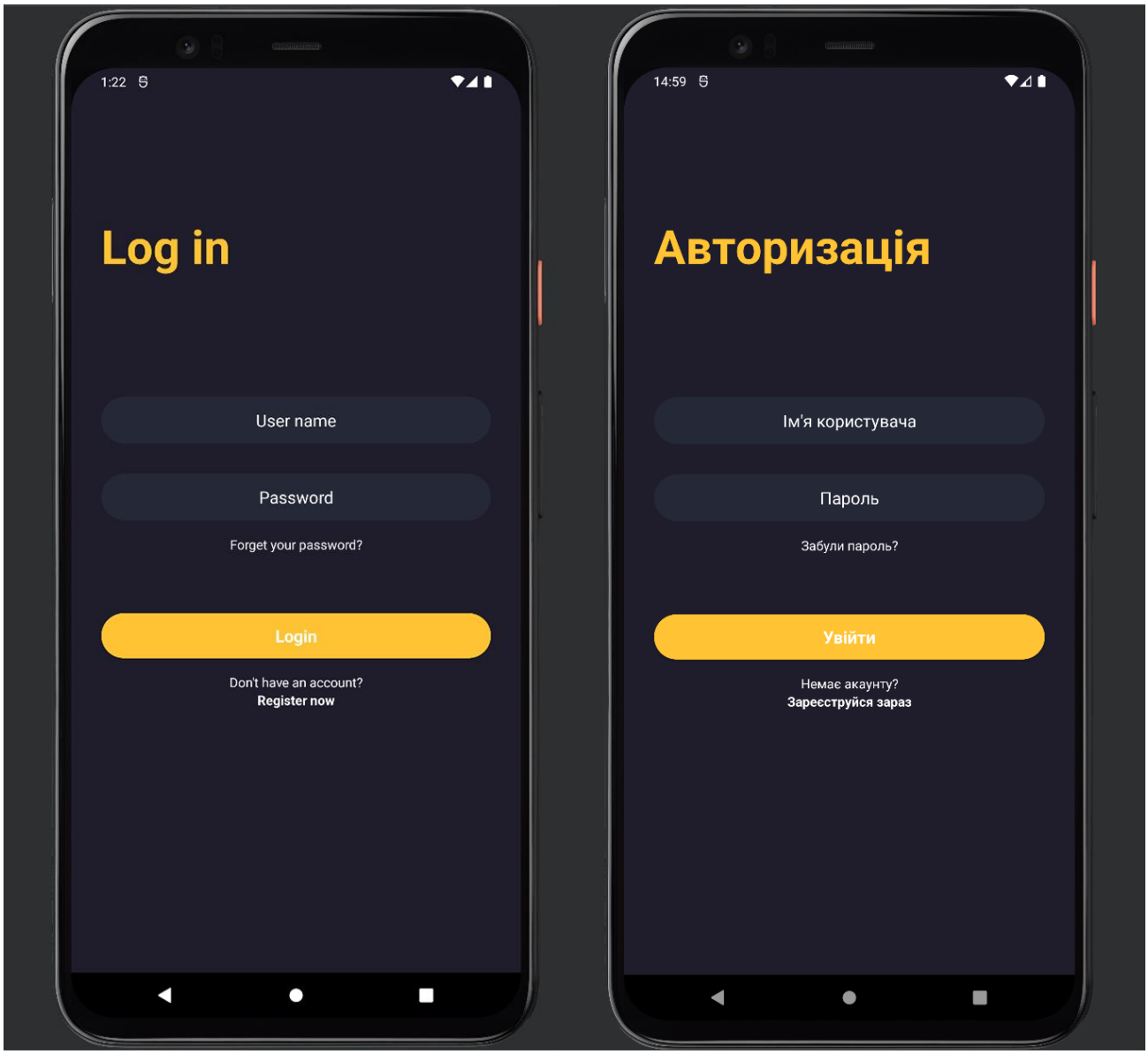


Рис. 3.5.2. Сторінка авторизації на англійські та українській мовах

Головна сторінка застосунку.

Головною сторінкою застосунку є сторінка пошуку та рекомендацій. На ній відображаються проекти, які зараз є популярними серед користувачів, майбутні релізи, списки популярних фільмів та серіалів, а також картин за категоріями та іншими тегами.

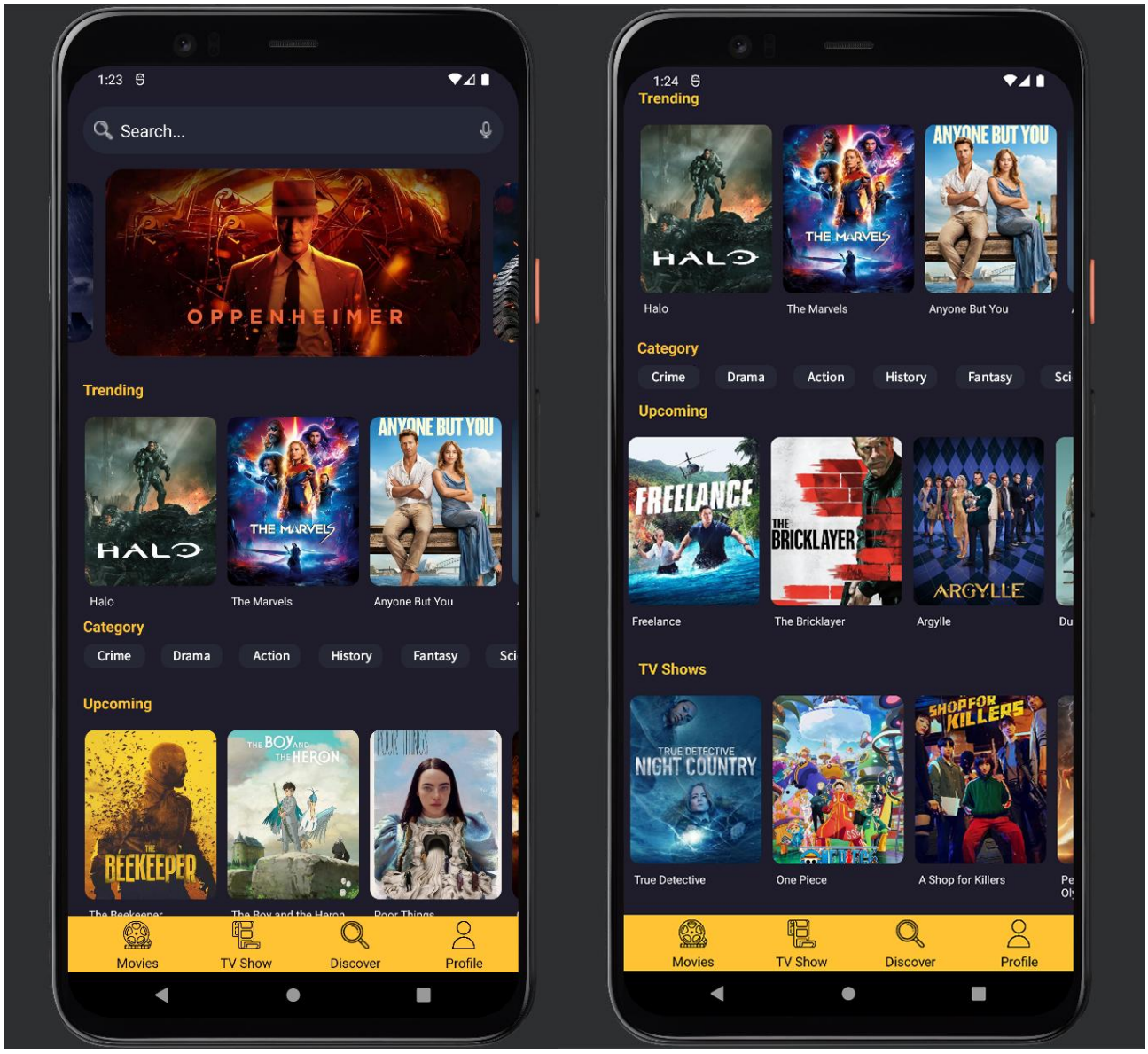


Рис. 3.5.3. Головна сторінка застосунку

Сторінка детальної інформації фільму або серіалу.

Сторінка детальної інформації про фільм або серіал відображається після вибору користувача конкретної кінокартини. Тут можна побачити опис обраного фільму, його оцінку, час необхідний для перегляду, список акторів які брали участь в роботі та приналежність до жанрів. Також на сторінці опису можна додати фільм до власного списку перегляду.

Рис. 3.5.4. Сторінка детальної інформації фільму

Висновки до розділу

У даному розділі було розглянуто види архітектури мобільних застосунків. Обрано структуру застосунку. Виявлено, що для розробки застосунку буде застосована клієнт-серверна архітектура. За допомогою діаграм класів було детально описано всі класи, які складають застосунок. Було розглянуто інструменти використані для розробки прототипу мобільного застосунку з відслідковування онлайн відео-контенту, середовище розробки, мову програмування, функціональні можливості та графічний інтерфейс.

ВИСНОВКИ

Під час виконання кваліфікаційної роботи було створено мобільний додаток на базі операційної системи Android, призначений для людей які полюбляють проводити свій вільний час за переглядом фільмів та серіалів. У процесі розробки були використані мови програмування такі як Java для реалізації логіки застосунку, PostgreSQL для роботи з базою даних, а також XML для оформлення інтерфейсу користувача.

Основною метою кваліфікаційної роботи було розробка Мобільного застосунку для створення і роботи з персональним відео-контентом "Show & Movie Tracker". Мобільний застосунок надає користувачам зручний інтерфейс для швидкого пошуку відомих фільмів та серіалів, а також відслідковування оновлень у світі кіно та телебачення. Крім того, застосунок враховує індивідуальні вподобання кожного користувача та надає персоналізовані рекомендації для забезпечення задоволення від перегляду контенту. Користувачі можуть створювати свої власні списки фільмів та серіалів, які хочуть подивитися у майбутньому, та зручно керувати цими списками через додаток.

У першому розділі роботи було проведено аналіз потреб користувачів та визначено чинники, які призводять до зростання популярності подібних програмних рішень. Крім того, був проведений огляд аналогічних додатків з відстежуванням відео-контенту, під час якого були виявлені їхні переваги та недоліки.

У другому розділі описано основні функціональні вимоги до прототипу додатку з відстежуванням відео-контенту, а також сформульовані нефункціональні вимоги до системи. Визначено середовище розробки, мову програмування та інші інструменти, необхідні для реалізації проекту. Крім того, було визначено можливість розширення додатку на інші платформи, такі як iOS.

У третьому розділі розглянуто різновиди архітектури мобільних застосунків та обрано структуру застосунку. Зазначено, що для розробки буде

використана клієнт-серверна архітектура. Крім того, описано вибір архітектурних компонентів і принципи їх взаємодії.

Було розглянуто інструменти використані для розробки прототипу мобільного застосунку з відслідковування онлайн відео-контенту. Описано середовище розробки, обрану мову програмування, функціональні можливості та графічний інтерфейс. Також було проведено докладний аналіз діаграм класів, які описують структуру та взаємодію компонентів додатку.

У даній кваліфікаційній роботі використані знання, набуті протягом усього періоду навчання в університеті. Засвоєні навички та знання включають в себе володіння мовою програмування Java, навички роботи з базами даних і використання SQL-синтаксису для операцій з даними, розуміння мови розмітки XML, вміння працювати з фреймворком Java Spring, освоєння методів локалізації програмного забезпечення, досвід використання локальних серверів та навички розробки мобільних застосунків.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Документація з програмування в середовищі розробки програмного забезпечення Android Studio // [Електронний ресурс].- https://developer.android.com/reference/android/graphics/pdf/PdfDocument.
2. Meet Android Studios // [Електронний ресурс].- https://developer.android.com/studio/intro
3. Maps SDK for Android // [Електронний ресурс].- https://developers.google.com/maps/documentation/android-sdk/map-with-marker?
4. Introduction to mobile Application Architectures // [Електронний ресурс].- https://www.informit.com/articles/article.aspx?p=336262
5. Mobile App Architecture // [Електронний ресурс].- https://www.intellectsoft.net/blog/mobile-app-architecture/
6. Client Server Architecture // [Електронний ресурс].- https://www.redswitches.com/blog/client-server-architecture/
7. Using Locale in Android Apps [Електронний ресурс].- https://developer.android.com/reference/java/util/Locale
8. Посібник Java// [Електронний ресурс].- https://metanit.com/java/
9. IntelliJ IDEA // [Електронний ресурс].- https://www.jetbrains.com/idea/
10. TMDP API // [Електронний ресурс].- https://www.themoviedb.org/