МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ
КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка та реалізація iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача**

Виконав: студент 4 курсу, групи 6.1210-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Д. В. Фролов
(ініціали та прізвище)
Керівник к.т.н., доцент, доцент кафедри ЕІС та ПЗ
Н. П. Полякова
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дісітел»
П. О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра Електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти перший(бакалаврський)
Спеціальність 121 Інженерія програмного забезпечення
(код та назва)
Освітня програма Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ

Завідувач кафедри Т. В. Критська
“ 01 ” 2024 року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Фролову Данілу Володимировичу
(прізвище, ім’я, по батькові)

1.Тема роботи Розробка та реалізація iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача
керівник роботи Полякова Наталія Петрівна, к.т.н., доцент
(прізвище ім’я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від 26.12.2023 № 2215-с
2. Строк подання студентом кваліфікаційної роботи 12.06.2023
3. Вихідні дані кваліфікаційної роботи бакалавра

- комплект нормативних документів ;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)

- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- створення програмного продукту та його опис;
- дослідження поставленої проблеми та розробка висновків.

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)
слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання 01.03.2024

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів бакалаврської роботи	Примітка
1	Аналіз предметної області	02.01 – 24.01.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	25.01 – 29.01.24	виконано
3	Аналіз існуючих методів розв’язання та аналогів	01.01 – 05.02.24	виконано
4	Огляд та збір літератури щодо теми кваліфікаційної роботи	06.02 – 20.02.24	виконано
5	Огляд та аналіз існуючих рішень та аналогів	21.02 – 28.02.24	виконано
6	Аналіз сучасних технологій для створення iOS-застосунку	01.03 – 03.03.24	виконано
7	Аналіз API штучного інтелекту	04.03 – 13.03.24	виконано
8	Аналіз існуючих API для отримання актуального рейтигу фільмів	14.03 – 19.03.24	виконано
9	Опрацювання функціональних вимог до системи	20.03 – 24.03.24	виконано
10	Розробка та впровадження алгоритмів персоналізації пошуку фільмів	25.03 – 20.04.24	виконано
11	Програмна реалізація застосунку	21.04 – 14.05.24	виконано
12	Оформлення звіту	15.05 – 31.05.24	виконано

Студент _____ Фролов Д. В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Полякова Н.П.
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ Скрипник І.А.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок — 73

Рисунків — 18

Таблиць — 1

Джерел — 20

Фролов Д. В. Розробка та реалізація iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник Н. П. Полякова. Запоріжжя : ЗНУ, 2024. 73 с.

У даній роботі представлено розробку та реалізацію iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача. Застосунок використовує OpenAI та TMDB API для забезпечення інтелектуального пошуку та надання користувачам релевантних рекомендацій.

Основна мета роботи – створення інструменту, який дозволяє користувачам знаходити фільми, що максимально відповідають їхнім особистим вподобанням, тим самим зменшуючи час, витрачений на пошук. Інтеграція штучного інтелекту (ШІ) у процес персоналізації дозволяє додатку аналізувати історію переглядів, оцінки та вподобання користувачів для надання більш точних рекомендацій.

Розробка додатку передбачає кілька ключових етапів: аналіз предметної області, огляд існуючих методів рішення, розробка архітектури системи та програмна реалізація. Особлива увага приділялася використанню сучасних технологій, таких як обробка природної мови та машинне навчання, для забезпечення високої якості рекомендацій та зручності використання додатку.

Ключові слова: *iOS-додаток, OpenAI та TMDB API, інтелектуальний пошук, штучний інтелект, фільми.*

ABSTRACT

Pages — 73

Drawings — 18

Tables — 1

Sources — 20

Frolov D. V. Development of an iOS application for personalized movie search using user preferences: Bachelor's qualification work in the specialty 121 "Software Engineering" / Scientific supervisor N. P. Polyakova. Zaporizhzhia: ZNU, 2024. 73 p.

This work presents the development and implementation of an iOS application for personalized movie search using user preferences. The application utilizes OpenAI and TMDB APIs to provide intelligent search and deliver relevant recommendations to users.

The main goal of the work is to create a tool that allows users to find movies that best match their personal preferences, thereby reducing the time spent searching. The integration of artificial intelligence (AI) into the personalization process enables the application to analyze users' viewing history, ratings, and preferences to provide more accurate recommendations.

The development of the application involves several key stages: analysis of the subject area, review of existing solutions, system architecture design, and software implementation. Special attention was paid to the use of modern technologies such as natural language processing and machine learning to ensure high-quality recommendations and ease of use.

Keywords: *iOS application, OpenAI and TMDB APIs, intelligent search, artificial intelligence, movies.*

ЗМІСТ

ВСТУП 7

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ 12

1.1 Огляд літературних джерел 12

1.2 Аналіз програмних продуктів-аналогів 13

1.3 Постановка задачі 20

2 ДОСЛІДЖЕННЯ ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ MOVIE+ 22

2.1 Огляд мов програмування 22

2.2 Огляд інструментів розробки 24

2.3 Огляд архітектурних паттернів 26

2.4 Огляд управліннь залежностями 31

2.5 Огляд методів зберігання даних 33

2.6 Огляд API 35

2.7 Висновок 38

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ 40

3.1 Опис предметної області 40

3.2 Архітектура системи 42

3.3 Функціональні вимоги системи 44

3.4 Вимоги до апаратного та програмного забезпечення 46

3.5 Модулі і алгоритми 47

3.6 Структури даних 59

3.7 Проєкт інтерфейсу 63

3.8 Реалізація і тестування 68

ВИСНОВКИ 71

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 72

ВСТУП

Актуальність теми

В умовах цифровізації та інформаційного перенасичення важливість розробки інноваційних технологічних рішень у медіа-індустрії стає незаперечною. Індустрія розваг, зокрема кінематограф, щодня генерує величезні обсяги контенту, що вимагають від користувачів все більше часу та зусиль для пошуку та відбору відповідних фільмів. Розробка iOS-додатку з інтеграцією штучного інтелекту для персоналізованого пошуку фільмів відповідає на ці виклики, надаючи користувачам унікальний інструмент для ефективного та захоплюючого вибору контенту.

Інтеграція штучного інтелекту, зокрема технологій, заснованих на обробці природної мови та машинному навчанні, відкриває нові горизонти для персоналізованих рекомендацій фільмів. Системи, що навчаються із кожним запитом користувача, можуть точніше підбирати фільми, які відповідають індивідуальним смакам та перевагам. Це не тільки підвищує задоволеність користувача від додатку, але й сприяє ефективнішому використанню часу, забезпечуючи більш релевантний і значущий досвід взаємодії з медіа-контентом.

Штучний інтелект надає можливість не просто аналізувати існуючі дані, але й прогнозувати майбутні вподобання користувачів, використовуючи складні алгоритми для виявлення шаблонів та трендів [1]. Це особливо актуально в контексті iOS-платформи, де висока конкуренція та підвищені вимоги до якості додатків вимагають інновацій та креативного підходу до розробки програмного забезпечення.

Крім того, використання ШІ для персоналізації пошуку фільмів може забезпечити користувачам унікальну цінність, яку не можуть запропонувати стандартні алгоритми пошуку. Завдяки інтелектуальному аналізу великих даних, можливо знайти неочікувані зв’язки між жанрами, акторським складом, режисерсь-

кими преференціями та іншими факторами, що впливають на вибір фільму. Такий підхід дозволяє не лише розширити горизонти вибору для глядача, але й надати більше можливостей для виявлення менш відомих, але цінних кінематографічних творів, що можуть відповідати унікальним смакам користувача.

З огляду на постійне зростання кількості доступного відеоконтенту, час на пошук і відбір фільмів стає все більш цінним ресурсом. Тому здатність додатку з штучним інтелектом вивчати патерни перегляду і вподобань користувачів, пропонуючи контент, який відповідає їхнім індивідуальним очікуванням, не лише підвищує загальну задоволеність користувачів, але й сприяє формуванню більш осмисленого та глибокого досвіду перегляду.

Розробка такого додатку також сприяє розвитку нових бізнес-моделей та можливостей для кінокомпаній і дистриб'юторів контенту. Завдяки більш точним рекомендаціям і персоналізованому підходу, компанії можуть краще розуміти уподобання своїх клієнтів і ефективніше просувати свій контент. Це також відкриває нові перспективи для співпраці між розробниками програмного забезпечення, компаніями, що займаються штучним інтелектом, та кіновиробниками.

Узагальнюючи, можна сказати, що розробка iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача є не лише актуальною, але й необхідною в умовах сучасного інформаційного середовища. Вона сприяє підвищенню якості взаємодії з медіа-контентом, ефективності використання часу та загальному задоволенню користувачів, а також відкриває нові можливості для бізнесу в сфері розваг.

Мета дослідження

Створення iOS-застосунку, який забезпечує персоналізований пошук фільмів на основі вподобань користувача.

Завдання дослідження

Дослідити існуючі технологічні рішення та аналогічні додатки для персоналізованого пошуку фільмів. Провести аналіз сучасних технологій та методів, що використовуються для розробки таких додатків. Розробити та протестувати прототип додатку. Оцінити продуктивність та ефективність створеного додатку, а також надати рекомендації щодо його вдосконалення.

Об’єкт дослідження

Процес створення iOS-додатку для персоналізованого пошуку фільмів з використанням сучасних технологій штучного інтелекту та мобільної розробки.

Предмет дослідження

Технології розробки мобільних додатків для персоналізованого пошуку фільмів, методи інтеграції штучного інтелекту, а також підходи до забезпечення високої точності та релевантності рекомендацій.

Методи дослідження

Аналіз наукової літератури та існуючих рішень у сфері персоналізованого пошуку фільмів. Моделювання та розробка архітектури додатку з використанням мов програмування та інструментів для iOS. Проведення функціонального, інтеграційного та користувацького тестування для оцінки ефективності рекомендацій. Використання статистичних методів для аналізу даних і визначення точності алгоритмів рекомендацій.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає у створенні ефективного інструменту для персоналізованого пошуку фільмів, підвищенні якості користувацького досвіду, підтримці бізнес-моделей кінокомпаній, а також у покращенні освітнього процесу в галузі програмної інженерії.

Апробація

Роботу було представлено на конференції «Молода наука-2024» [2].

Глосарій

iOS-додаток — програмне забезпечення, розроблене для роботи на операційній системі iOS, яка використовується на пристроях Apple, таких як iPhone та iPad. Цей додаток спеціально створений для персоналізованого пошуку фільмів з використанням вподобань користувачів.

Штучний інтелект (ШІ) — галузь комп'ютерних наук, що займається створенням систем, здатних виконувати завдання, які зазвичай вимагають людського інтелекту, такі як розпізнавання мови, навчання, планування та прийняття рішень. У цьому контексті ШІ використовується для аналізу вподобань користувачів та надання персоналізованих рекомендацій.

OpenAI API — інтерфейс програмування додатків, розроблений компанією OpenAI, який надає доступ до різноманітних моделей ШІ для виконання завдань, пов'язаних з обробкою природної мови, генерацією тексту, перекладом та іншими завданнями.

TMDB API — інтерфейс програмування додатків, наданий базою даних фільмів The Movie Database (TMDB), який дозволяє отримувати інформацію про фільми, акторів, рейтинги та інші дані, необхідні для створення додатків для перегляду та пошуку фільмів.

Обробка природної мови — підгалузь штучного інтелекту, яка займається взаємодією між комп’ютерами та людьми за допомогою природної мови. Вона включає такі завдання, як розуміння, інтерпретація та генерація тексту.

Машинне навчання — метод аналізу даних, що автоматизує побудову аналітичних моделей. Це галузь штучного інтелекту, яка базується на ідеї, що системи можуть вчитися з даних, ідентифікувати шаблони та приймати рішення з мінімальним втручанням людини.

Кіномани — люди, які захоплюються переглядом фільмів і мають великий інтерес до кінематографу, часто шукають нові та цікаві фільми для перегляду.

Система керування базами даних (СКБД) — програмне забезпечення, яке використовується для створення, управління та адміністрування баз даних. СКБД забезпечує інструменти для зберігання, пошуку, оновлення та видалення даних, а також забезпечує безпеку та цілісність даних. У контексті даного додатку СКБД використовується для зберігання списків переглянутих фільмів та іншої інформації, що стосується користувачів та їх вподобань.

1 ОГЛЯД ІСНУЮЧИХ РІШЕНЬ

1.1 Огляд літературних джерел

У даному розділі буде представлено аналіз наукових та технічних джерел, присвячених розробці мобільних додатків, персоналізованих систем рекомендацій та використанню штучного інтелекту для покращення користувацького досвіду. Основна увага приділятиметься дослідженням, які описують архітектурні рішення, алгоритми машинного навчання, методи обробки природної мови та інші технології, що використовуються для створення інтелектуальних мобільних застосунків. Огляд літературних джерел допоможе визначити актуальні тенденції, існуючі методи та підходи, а також виявити можливі напрямки для подальших досліджень та вдосконалення розробки застосунку Movie+.

Розробка мобільних додатків значно еволюціонувала за останні роки, зокрема завдяки впровадженню передових архітектурних шаблонів. У дослідженні Di Francesco, Lago та Malavolta [3] детально розглянуто мікросервісні архітектурні рішення, які сприяють підвищенню масштабованості та підтримуваності програмного забезпечення. Важливість модульної архітектури підкреслюється у багатьох джерелах, оскільки вона дозволяє легше оновлювати та модифікувати систему.

Застосування машинного навчання для персоналізації контенту є ключовою темою у сучасних дослідженнях. Стаття на платформі Sitecore [4] описує різні методи та алгоритми машинного навчання, які використовуються для надання персоналізованих рекомендацій користувачам. Зокрема, підкреслюється роль обробки природної мови та машинного навчання у створенні інтелектуальних систем, що враховують вподобання користувачів для поліпшення їхнього досвіду.

Одним із провідних інструментів у сфері штучного інтелекту є OpenAI API, який надає доступ до потужних моделей машинного навчання, таких як GPT-3. В дослідженні OpenAI [5] розглядається використання GPT-3 для створення інтелектуальних систем, здатних розуміти та генерувати текст на природній мові. Використання OpenAI API дозволяє інтегрувати передові алгоритми штучного інтелекту в мобільні додатки, надаючи можливість створювати високоякісні рекомендаційні системи.

Платформа Moments Log також надає важливі інсайти щодо переваг модульної архітектури для покращення масштабованості та підтримуваності програмних систем [6]. Дослідження підкреслює важливість розподілу системи на незалежні модулі, що значно полегшує процес оновлення та модифікації програмного забезпечення.

Результати досліджень показують, що використання мікросервісної архітектури та алгоритмів машинного навчання значно підвищує ефективність, надійність і зручність у розробці сучасних мобільних застосунків. Впровадження передових технологій, таких як OpenAI API, дозволяє створювати інноваційні рішення, що забезпечують високий рівень персоналізації та задоволення користувачів.

1.2 Аналіз програмних продуктів-аналогів

Для розробки ефективного iOS-додатку для персоналізованого пошуку фільмів важливо проаналізувати існуючі програмні рішення. Вивчення аналогічних додатків дозволяє визначити їхні сильні та слабкі сторони, а також запозичити найкращі практики для впровадження у власному проєкті. У цьому розділі розглянемо декілька популярних додатків, що пропонують подібний функціонал, та проведемо їх детальний аналіз з метою визначення основних аспектів, які сприяють успішному виконанню їхніх задач.

Netflix

Використовує складні алгоритми рекомендацій, які аналізують перегляди та вподобання користувачів [7].

Інтерфейс застосунку можемо побачити на рисунку 1.

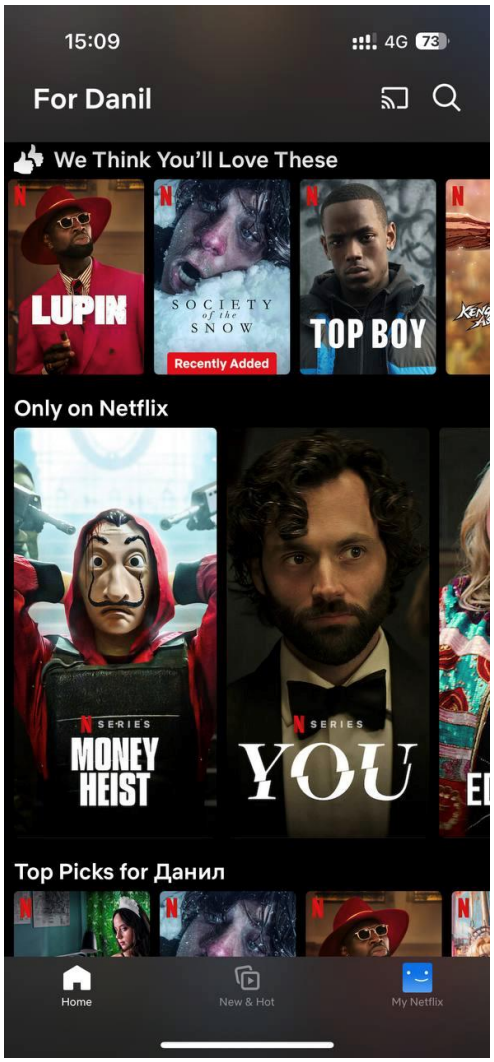

Рисунок 1 — Інтерфейс застосунку Netflix

Letterboxd

Соціальна мережа для любителів кіно, яка дозволяє вести облік переглянутих фільмів і отримувати рекомендації на основі вподобань інших користувачів [8].

Інтерфейс застосунку можемо побачити на рисунку 2.

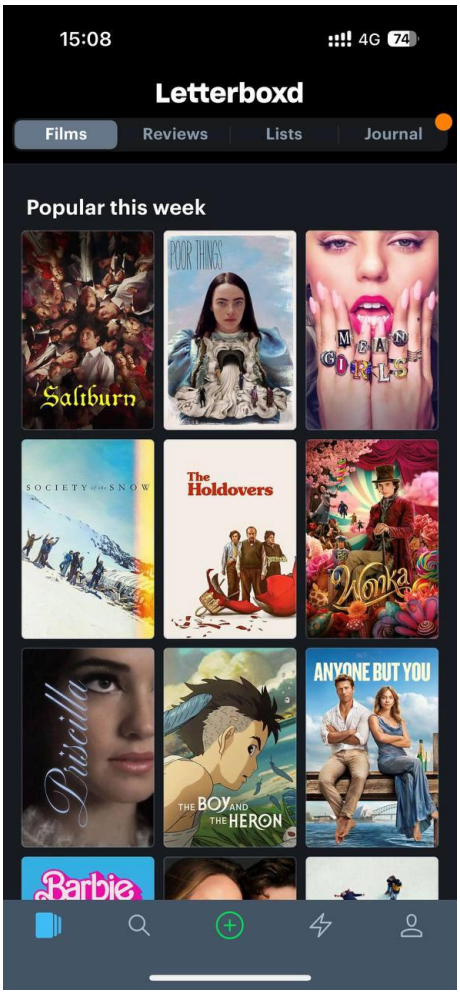

Рисунок 2 — Інтерфейс застосунку Letterboxd

MovieLens

Сервіс використовує алгоритми машинного навчання для надання персоналізованих пропозицій на основі індивідуальних переваг користувачів. Його ключовими перевагами є не тільки персоналізація рекомендацій, але й використання категоризації фільмів та вбудований список бажань, що робить відбір фільмів зручнішим [9].

Інтерфейс веб-сервісу MovieLens можна побачити на рисунку 3.

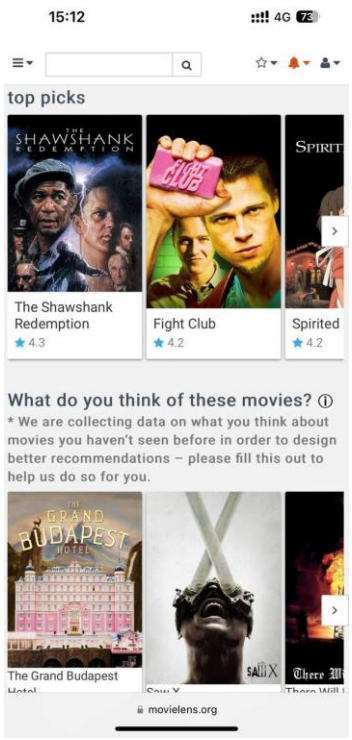

Рисунок 3 — Інтерфейс веб-застосунку MovieLens

При порівнянні існуючих рішень, таких як Letterboxd, MovieLens та Netflix, важливо враховувати декілька ключових факторів:

Функціонал

Netflix — сервіс, який пропонує широкий вибір фільмів і серіалів, включаючи ексклюзивні виробництва та оригінальний контент, доступний тільки на Netflix. Система рекомендацій заснована на алгоритмах, які аналізують перегляди та вподобання усіх користувачів, формуючи списки рекомендованих фільмів на основі популярних трендів та взаємозв'язків у виборі контенту. Однак, оскільки фокус зроблено на промоції власних виробництв та ексклюзивного контенту, вибір фільмів може бути обмеженим і не завжди повністю відповідати індивідуальним вподобанням користувачів.

Letterboxd — платформа функціонує як соціальна мережа для кіноентузіастів, де користувачі можуть вести записи фільмів, які вони переглянули чи планують переглянути, а також відстежувати рекомендації від інших користувачів. Вона дозволяє користувачам створювати та ділитися власними списками та відгуками, але може не мати такої ж потужності персоналізації рекомендацій, як системи, що базуються на ШІ.

MovieLens — платформа пропонує персоналізовані рекомендації, використовуючи алгоритми машинного навчання для аналізу індивідуальних переваг користувачів. MovieLens збирає оцінки фільмів від користувачів і застосовує складні математичні моделі для прогнозування, які фільми можуть сподобатися конкретному користувачеві. Відмінною особливістю MovieLens є те, що сервіс не переслідує комерційну мету і не містить реклами, надаючи високоякісні рекомендації без будь-якого зовнішнього впливу.

Цінова політика

Netflix вимагає платної підписки, вартість якої залежить від кількості екранів та рівня доступу до HD чи Ultra HD контенту.

Letterboxd пропонує базову безкоштовну версію, але для доступу до додаткових функцій, таких як статистика переглядів та відсутність реклами, потрібно оформити платну підписку.

MovieLens є повністю безкоштовним сервісом, що є великою перевагою для користувачів, які не хочуть витрачати кошти на підписку. Це робить MovieLens особливо привабливим для тих, хто шукає персоналізовані рекомендації без будь-яких фінансових зобов’язань.

Політика розповсюдження

Netflix має власну політику розповсюдження контенту, яка іноді обмежує доступ до певних фільмів та серіалів в залежності від географічного розташування користувача. Це може створювати незручності для міжнародних користувачів, які шукають специфічний контент.

Letterboxd, зосереджуючись на соціальному аспекті та веденні списків фільмів, не займається розповсюдженням контенту, тому не стикається з такими обмеженнями.

MovieLens, будучи некомерційним сервісом, також уникає проблем, пов'язаних із ліцензуванням контенту, оскільки він не розповсюджує фільми, а лише надає рекомендації.

Додаткові функції

Netflix пропонує такі додаткові функції, як можливість створення декількох профілів на одному акаунті, що дозволяє користувачам мати індивідуалізовані списки перегляду та рекомендацій. Також є функції батьківського контролю та різноманітні налаштування для покращення досвіду перегляду.

Letterboxd відзначається своїми соціальними функціями, дозволяючи користувачам створювати і ділитися списками фільмів, писати відгуки та стежити за оцінками та рекомендаціями інших. Крім того, користувачі можуть взаємодіяти один з одним, обговорювати фільми та відстежувати активність друзів.

MovieLens пропонує детальні аналітичні звіти про перегляди та оцінки, що дає користувачам змогу глибше аналізувати свої переваги. Сервіс також підтримує функцію експорту даних, що дозволяє користувачам зберігати свою історію переглядів та оцінок.

Мобільна оптимізація

Netflix має високоякісний мобільний додаток, який доступний на більшості платформ та пристроїв. Цей додаток забезпечує зручний інтерфейс, оптимізований для перегляду в дорозі, і включає всі основні функції настільної версії.

Letterboxd також має мобільний додаток, який дозволяє користувачам вести список переглянутих фільмів, читати та писати відгуки, та взаємодіяти зі спільнотою. Цей додаток є зручним інструментом для кіноентузіастів, що хочуть залишатися на зв'язку зі світом кіно в будь-який час.

MovieLens не має спеціалізованого мобільного додатку, що може обмежувати його використання на мобільних пристроях. Веб-версія MovieLens хоча і доступна через мобільні браузери, може не забезпечувати такої ж зручності та інтуїтивності, як спеціалізовані мобільні додатки.

Підсумок порівняння

Netflix — хоча сервіс славиться своїм широким асортиментом контенту та сильними алгоритмами рекомендацій, основним недоліком є вимога платної підписки. Це може бути фінансово обтяжливо для деяких користувачів. Крім того, алгоритми рекомендацій Netflix можуть не завжди точно відображати індивідуальні уподобання, оскільки вони схильні зосереджуватися на загальних трендах і популярності контенту.

Letterboxd — сервіс пропонує відмінну соціальну платформу для обговорення та оцінювання фільмів, але основним недоліком є відсутність персоналізованих рекомендацій на рівні Netflix чи MovieLens. Це може обмежувати користувачів у відкритті нових фільмів, які точно відповідають їхнім персональним уподобанням. Також, платна підписка для доступу до додаткових функцій може бути не до вподоби деяким користувачам.

MovieLens — найбільшим недоліком MovieLens є відсутність мобільного додатку, що робить сервіс менш зручним для користувачів, які переважають мобільний перегляд. Також інтерфейс MovieLens може бути сприйнятий як менш інтуїтивний порівняно з іншими сервісами. Хоча MovieLens пропонує персоналізовані рекомендації, відсутність реального доступу до перегляду фільмів може бути обмеженням для користувачів, які шукають все-в-одному рішення для відкриття та перегляду фільмів.

Підсумовуючи, вибір між Netflix, Letterboxd та MovieLens залежить від особистих переваг користувача та того, наскільки важливими є для них певні аспекти, такі як широкий вибір контенту, соціальна взаємодія, точність персоналізованих рекомендацій, вартість підписки, наявність мобільного додатку, та зручність користувацького інтерфейсу. Кожен з цих сервісів має свої сильні сторони, але також та свої обмеження, які можуть впливати на вибір користувачів.

1.3 Постановка задачі

Розробка ефективного iOS-застосунку для персоналізованого пошуку фільмів потребує чіткого визначення задач, які необхідно вирішити для досягнення основної мети. Головним завданням є створення сучасного застосунку, який забезпечить користувачам можливість швидко та легко знаходити фільми, що відповідають їхнім індивідуальним вподобанням.

Основним пріоритетом є розробка зручного інтерфейсу користувача (UI) та користувацького досвіду (UX), що забезпечить інтуїтивну та приємну взаємодію з застосунком. Важливо створити дизайн, який буде не тільки привабливим, але й функціональним, з легким доступом до всіх необхідних функцій і зручною навігацією.

Крім того, необхідно забезпечити ефективний підбір фільмів за допомогою сучасних технологій штучного інтелекту. Використання алгоритмів машинного

навчання та обробки природної мови дозволить аналізувати вподобання користувачів та надавати точні й релевантні рекомендації. Інтеграція з OpenAI та TMDB API дозволить використовувати великі бази даних для аналізу та надання рекомендацій.

Важливо також розробити архітектуру системи, яка забезпечить ефективну роботу з великими обсягами даних та високою продуктивністю. Це включає вибір відповідних технологій для бази даних, алгоритмів машинного навчання та інтеграцію зовнішніх API.

Після розробки прототипу необхідно провести його тестування, щоб перевірити працездатність основних функцій та виявити можливі помилки. Проведення тестування допоможе оптимізувати роботу системи та забезпечити її надійність і стабільність.

Останнім, але не менш важливим кроком, є оцінка ефективності роботи застосунку та його здатність задовольняти потреби користувачів. На основі результатів тестування слід внести необхідні корективи та вдосконалення для підвищення продуктивності та якості рекомендацій.

Вирішення цих задач дозволить створити високоякісний продукт, який не тільки задовольнить потреби користувачів, але й забезпечить конкурентоспроможність на ринку мобільних застосунків.

2 ДОСЛІДЖЕННЯ ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ MOVIE+

2.1 Огляд мов програмування

Для розробки сучасного iOS-застосунку Movie+, який забезпечує персоналізований пошук фільмів, важливо ретельно обрати засоби програмної реалізації. У контексті платформи iOS існують дві основні мови програмування, які використовуються для створення застосунків: Objective-C та Swift.

Objective-C була розроблена в 1980Сх роках Бредом Коксом і Томом Лавом як надбудова над мовою C з додаванням об’єктно-орієнтованих можливостей. Ця мова була офіційно прийнята компанією Apple і стала основною для розробки програмного забезпечення під платформи macOS та iOS [10].

Основні відмінності Objective-C:

1. Сумісність з C: Objective-C є розширенням мови C, що дозволяє використовувати існуючі бібліотеки та код, написані на C. Це забезпечує високу продуктивність і доступ до низькорівневих системних ресурсів.
2. Об’єктно-орієнтоване програмування (ООП): Objective-C додає концепції ООП до C, включаючи класи, об’єкти, методи та успадкування, що дозволяє створювати модульний та багаторазово використовуваний код.
3. Динамічна типізація і повідомлення: Objective-C використовує динамічну типізацію та механізм повідомлень, що дозволяє визначати методи та викликати їх під час виконання програми. Це забезпечує гнучкість у розробці та дозволяє створювати складні програмні системи.
4. Синтаксис: Синтаксис Objective-C є складним через використання специфічних конструкцій.

Swift була розроблена компанією Apple і представлена в 2014 році як заміна для Objective-C. Мова була створена під керівництвом Кріса Латтнера, який також працював над проєктом LLVM. Swift призначена для забезпечення більшої простоти, безпеки та продуктивності в розробці програмного забезпечення [11].

Основні відмінності Swift:

1. Сучасний синтаксис: Swift має сучасний і зручний синтаксис, який є більш читабельним та інтуїтивно зрозумілим. Це робить мову більш доступною для новачків і спрощує процес написання коду.
2. Безпека і управління пам’яттю: Swift має вбудовані засоби для обробки помилок і автоматичного управління пам’яттю за допомогою ARC (Automatic Reference Counting), що зменшує ймовірність витоків пам’яті і покращує надійність коду.
3. Продуктивність: Swift була спроєктована з урахуванням високої продуктивності та оптимізації, що дозволяє створювати швидкі та ефективні застосунки. Компілятор Swift використовує сучасні технології оптимізації, що підвищує швидкість виконання коду.
4. Функціональне програмування: Swift підтримує функціональне програмування, що дозволяє розробникам писати компактний та читабельний код за допомогою функцій вищого порядку, замикань та інших функціональних конструкцій.
5. Розширення і підтримка: Swift активно розвивається і підтримується спільнотою розробників по всьому світу. Це забезпечує постійне вдосконалення мови, додавання нових функцій та виправлення помилок.

Основні відмінності між Objective-C та Swift

Objective-C має динамічну типізацію, тоді як Swift використовує статичну типізацію, що забезпечує більшу безпеку типів під час компіляції. Синтаксис Objective-C може бути складним для новачків, тоді як Swift має більш зрозумілий та сучасний синтаксис. Swift забезпечує кращу продуктивність та оптимізацію

коду, а також має вбудовані засоби для управління пам’яттю та обробки помилок, що підвищує надійність і безпеку коду.

Вибір мови програмування

Для реалізації проєкту Movie+ доцільно обрати Swift як основну мову програмування. Вона забезпечує високу продуктивність, зручний синтаксис, потужні інструменти для розробки та активну підтримку з боку спільноти. Swift дозволяє створювати сучасні, швидкі та надійні застосунки, що є важливим для досягнення цілей проєкту Movie+.

2.2 Огляд інструментів розробки

Для вибору відповідних інструментів розробки інтерфейсу користувача (UI) необхідно розглянути основні опції, які забезпечать зручний користувацький досвід та високу продуктивність застосунку. Два основні інструменти, які використовуються для цієї мети, — це UIKit та SwiftUI. Розглянемо їхні особливості, переваги та недоліки.

UIKit є основним фреймворком для розробки інтерфейсів користувача на платформі iOS, який використовується з моменту запуску iOS. UIKit забезпечує багатий набір компонентів для створення інтерфейсу користувача, включаючи кнопки, таблиці, колекції, навігаційні контролери та багато інших елементів [12].

Однією з ключових переваг UIKit є його зрілість і стабільність. Оскільки UIKit використовується протягом багатьох років, він має велику спільноту розробників і багату документацію. Це полегшує вирішення проблем, з якими можуть стикатися розробники, і забезпечує доступ до великої кількості ресурсів та прикладів.

UIKit підтримує широкий спектр версій iOS, що дозволяє створювати застосунки, сумісні з більшістю пристроїв, що використовуються сьогодні. Це особливо важливо для забезпечення максимальної аудиторії користувачів, які зможуть використовувати застосунок без обмежень.

SwiftUI — це новий фреймворк для розробки інтерфейсів користувача, представлений Apple в 2019 році. SwiftUI використовує декларативний підхід до створення UI, що дозволяє розробникам описувати, як має виглядати і функціонувати інтерфейс користувача, використовуючи зрозумілий і компактний синтаксис [13].

SwiftUI має багато переваг, включаючи спрощений синтаксис, тісну інтеграцію з мовою Swift та підтримку автоматичних оновлень інтерфейсу. Завдяки цьому розробка стає швидшою і більш ефективною. Крім того, SwiftUI забезпечує зручний попередній перегляд інтерфейсу в режимі реального часу, що дозволяє розробникам миттєво бачити результати своїх змін.

Однак, SwiftUI має певні обмеження. Одним з основних недоліків є те, що SwiftUI підтримується лише на iOS 13 і вище, а деякі з його функцій доступні тільки з iOS 15. Це означає, що застосунки, розроблені з використанням SwiftUI, не будуть сумісні з пристроями, що працюють на старіших версіях iOS. Це може значно обмежити аудиторію користувачів.

Вибір інструменту розробки інтерфейсу

Для проєкту Movie+ доцільно обрати UIKit як основний інструмент для розробки інтерфейсу користувача. UIKit забезпечує широку підтримку пристроїв, стабільність і зрілість, а також доступ до великої кількості ресурсів і прикладів. Це дозволить створити застосунок, який буде сумісний з більшістю пристроїв iOS і забезпечить кращий досвід для користувачів.

SwiftUI, хоч і є перспективним інструментом з багатьма перевагами, має обмежену підтримку пристроїв через вимогу нових версій iOS. Тому, для забезпечення максимальної аудиторії користувачів, використання UIKit є оптимальним вибором для проєкту Movie+.

2.3 Огляд архітектурних паттернів

Вибір архітектурного паттерну є критично важливим для розробки iOS-застосунків, оскільки він визначає гнучкість, тестованість та легкість підтримки коду. Розглянемо чотири популярні архітектурні паттерни: MVC (Model-View-Controller), MVVM (Model-View-ViewModel), MVP (Model-View-Presenter) та VIPER (View-Interactor-Presenter-Entity-Router), щоб визначити найбільш підходящий для проєкту Movie+ [14].

MVC є одним з найстаріших і найбільш використовуваних архітектурних паттернів. У цій архітектурі Model відповідає за дані та бізнес-логіку, View відображає дані та взаємодіє з користувачем, а Controller керує взаємодією між Model та View, обробляючи логіку відображення.

Зображення схеми архітектури MVC представлено на рисунку 4.

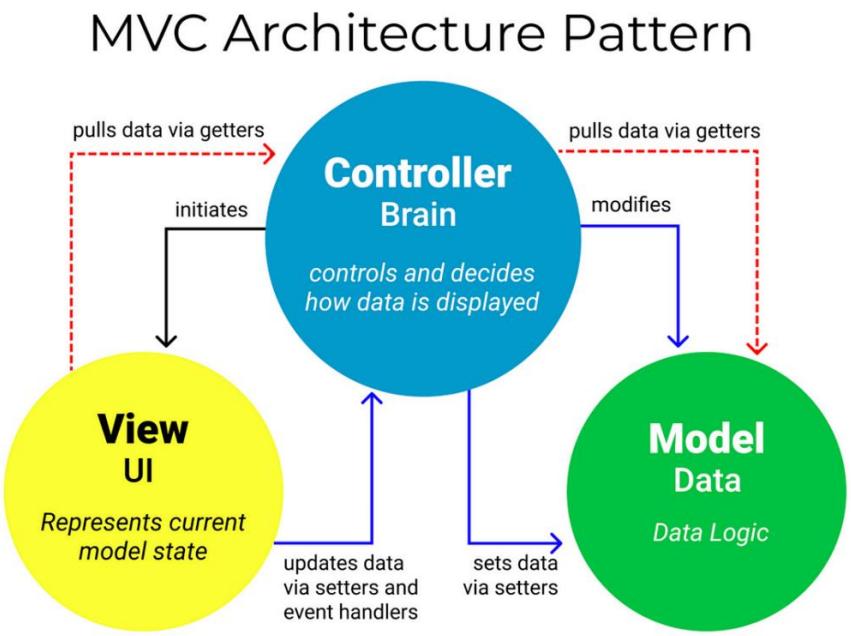

Рисунок 4 — Схема архітектури MVC

Переваги MVC:

1. Простота реалізації та розуміння.
2. Широко використовується та добре підтримується в спільноті.
3. Добре підходить для невеликих та середніх проєктів.

Недоліки MVC:

1. Може призводити до перевантаження Controller при великій кількості логіки.
2. Відсутність чіткого розділення відповідальностей може ускладнювати підтримку коду в великих проєктах.

MVVM є архітектурним паттерном, який відокремлює бізнес-логіку та представлення даних від інтерфейсу користувача. У цій архітектурі Model відповідає за дані та бізнес-логіку, View відображає дані та взаємодіє з користувачем, а ViewModel забезпечує зв’язок між Model та View, обробляючи дані для відображення.

Зображення схеми архітектури MVVM представлено на рисунку 5.

Рисунок 5 — Схема архітектури MVVM

Переваги MVVM:

1. Легка тестованість бізнес-логіки.
2. Добра підтримка двостороннього зв’язку між View та ViewModel.
3. Відокремлення представлення даних від бізнес-логіки.

Недоліки MVVM:

1. Може бути складним для реалізації в проєктах зі складною логікою.
2. Перевантаження ViewModel при великій кількості бізнес-логіки.

MVP є архітектурним паттерном, що дозволяє розділити інтерфейс користувача та бізнес-логіку за допомогою Presenter. Model відповідає за дані та бізнес-логіку, View відображає дані та взаємодіє з користувачем, а Presenter керує взаємодією між Model та View, обробляючи логіку відображення.

Зображення схеми архітектури MVP представлено на рисунку 6.

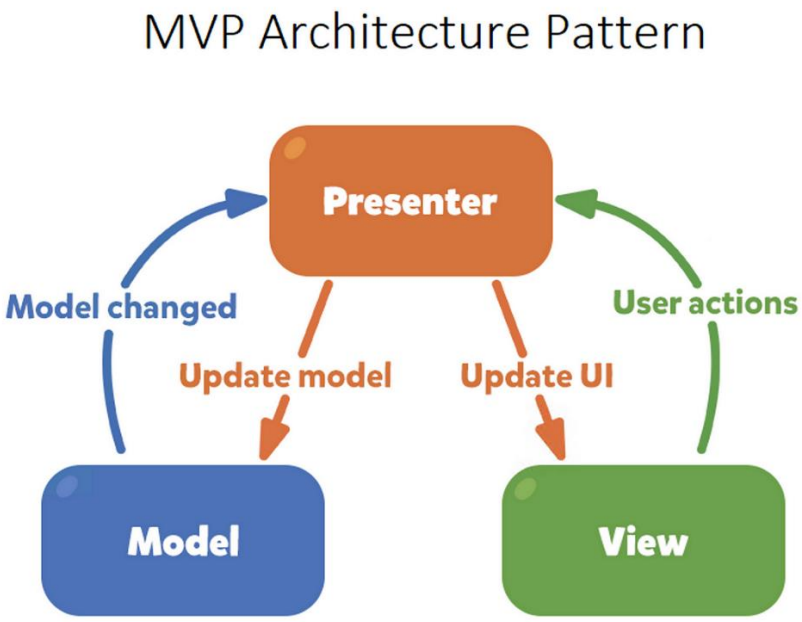

Рисунок 6 — Схема архітектури MVP

Переваги MVP:

1. Легка тестованість Presenter.
2. Добре відокремлення бізнес-логіки від інтерфейсу користувача.
3. Простота реалізації та розуміння.

Недоліки MVP:

1. Можливе перевантаження Presenter при великій кількості логіки.
2. Проблеми з масштабованістю для великих проєктів.

VIPER є архітектурним паттерном, який забезпечує чітке розділення відповідальностей між різними компонентами. View відповідає за відображення даних та взаємодію з користувачем, Interactor містить бізнес-логіку, Presenter керує взаємодією між View та Interactor, Entity представляє моделі даних, а Router відповідає за навігацію між екранами.

Зображення схеми архітектури VIPER представлено на рисунку 7.

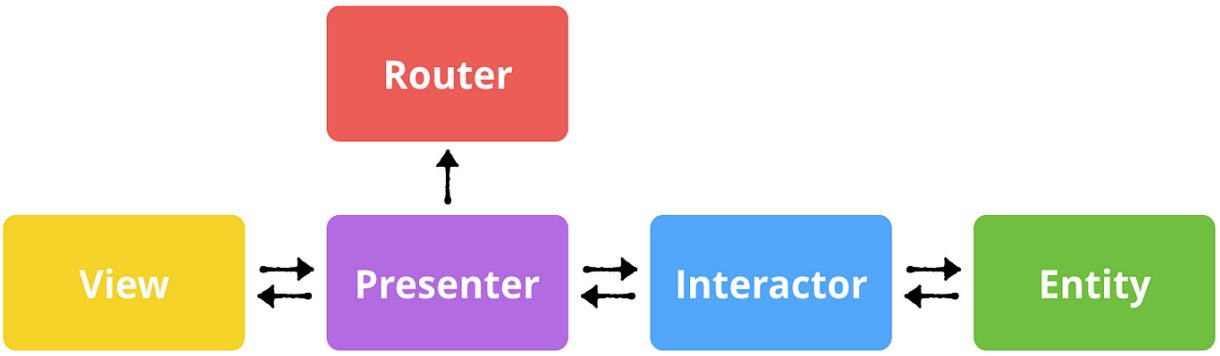

Рисунок 7 — Схема архітектури VIPER

Переваги VIPER:

1. Чітке розділення відповідальностей між компонентами.
2. Легка тестованість кожного компонента.
3. Висока масштабованість для великих проєктів.
4. Структурована архітектура, що полегшує підтримку та розширення проєкту.

Недоліки VIPER:

1. Складність реалізації через велику кількість компонентів.
2. Можлива складність в розумінні архітектури новачками.

Вибір архітектурного паттерну

Враховуючи потреби проєкту Movie+, архітектура VIPER є найбільш підходящою. Вона забезпечує чітке розділення відповідальностей, що є особливо важливим у проєктах з великою кількістю взаємодій з мережею. VIPER дозволяє легко тестувати кожен компонент окремо, що підвищує надійність та якість коду. Крім того, VIPER забезпечує високу масштабованість, що дозволяє ефективно працювати з великими обсягами даних та складними бізнес-логіками. Незважаючи на складність реалізації, переваги VIPER значно переважають, особливо у

контексті створення високоякісного застосунку з інтенсивною мережею взаємодій.

2.4 Огляд управліннь залежностями

Управління залежностями відіграє ключову роль у розробці iOS-застосунків, оскільки дозволяє легко інтегрувати сторонні бібліотеки та фреймворки, що прискорює процес розробки та забезпечує високу якість коду. Серед популярних інструментів для управління залежностями на платформі iOS виділяються CocoaPods, Carthage та Swift Package Manager (SPM) [15]. Розглянемо їхні особливості, переваги та недоліки, щоб обрати найкращий варіант для проєкту Movie+.

Порівняння засобів для управління залежностями наведено в таблиці 1.

Таблиця 1 — Порівняння інструментів CocoaPods, Carthage та SPM

Параметр	**CocoaPods**	**Carthage**	**Swift Package Manager (SPM)**
Інтеграція з проєктом	Додає додатковий крок у процесі налаштування проєкту через генерацію файлу .xcworkspace	Менше втручання в проєктну структуру у порівнянні з CocoaPods	Інтеграція безпосередньо в Xcode
Простота використання	Широке використання у спільноті розробників, добре задокументоване	Більше ручної роботи з налаштуванням та управлінням залежностями	Простота у використанні та мінімальний вплив на структуру проєкту
Швидкість збірки	Повільніша збірка проєкту у порівнянні з іншими інструментами	Використання бінарних фреймворків, що може пришвидшити збірку проєкту	Висока швидкість збірки завдяки інтеграції з Xcode та використанню сучасних технологій оптимізації
Підтримка мов програмування	Swift, Objective-C, C	Swift, Objective-C, C	Swift, C
Налаштування та інтеграція	Вимагає встановлення та налаштування окремо від Xcode	Складніша інтеграція з деякими бібліотеками	Інтеграція з Xcode, що спрощує налаштування та управління залежностями
Автоматичне оновлення залежностей	Так	Ні	Так

Вибір засобу управління залежностями

На основі аналізу інструментів управління залежностями, Swift Package Manager (SPM) є найбільш доцільним вибором для проєкту Movie+. Його інтеграція з Xcode спрощує процес налаштування та управління залежностями, забезпечуючи при цьому високий рівень автоматизації та сумісності з новими версіями Swift. Хоча SPM має деякі обмеження щодо підтримки бібліотек, написаних

не на Swift, його переваги значно переважають у контексті створення сучасного та надійного iOS-застосунку.

2.5 Огляд методів зберігання даних

Для зберігання списку переглянутих фільмів у iOS-додатку Movie+ розглянемо чотири основні способи збереження даних: Core Data, Realm, UserDefaults та Keychain. Кожен з цих способів не є традиційною системою керування базами даних (СКБД), але надає ефективні методи зберігання даних, які відповідають специфічним вимогам мобільних додатків. Вибір правильного способу зберігання залежить від вимог до продуктивності, безпеки та зручності використання.

Core Data

Core Data — це потужний фреймворк для управління об’єктами та збереження їх у локальній базі даних. Він надає розробникам інструменти для управління даними, збереження їх на диск, відновлення та забезпечення цілісності.

Переваги:

- Потужний і гнучкий інструмент для роботи з даними.
- Підтримка складних запитів та обробки даних.
- Автоматичне управління зв'язками між об'єктами та цілісністю даних.
- Можливість створення моделей даних з графічним інтерфейсом у Xcode.

Недоліки:

- Високий поріг входу для новачків через складність фреймворку.
- Порівняно низька продуктивність для великих обсягів даних.
- Великий обсяг коду для простих операцій з даними.

Realm

Realm — це сучасна база даних для мобільних додатків, яка забезпечує високу продуктивність та простоту використання.

Переваги:

- Висока продуктивність та швидкість роботи.
- Простий та інтуїтивно зрозумілий API.
- Підтримка багатопотоковості та синхронізації даних у реальному часі.
- Автоматичне оновлення даних у реальному часі.

Недоліки:

- Необхідність додавання зовнішньої бібліотеки до проєкту.
- Обмежена підтримка складних запитів та обробки даних порівняно з Core Data.
- Відсутність графічного інтерфейсу для моделювання даних у Xcode.

UserDefaults

UserDefaults — це інструмент для зберігання невеликих обсягів даних користувача, таких як налаштування та переваги.

Переваги:

- Простота використання та інтеграції.
- Висока швидкість збереження та отримання даних.
- Ідеально підходить для зберігання невеликих обсягів даних.

Недоліки:

- Не підходить для зберігання великих обсягів даних.
- Відсутність підтримки складних структур даних.
- Низький рівень безпеки для конфіденційних даних.

Keychain

Keychain — це безпечне сховище для збереження конфіденційних даних, таких як паролі та токени.

Переваги**:**

- Високий рівень безпеки для зберігання конфіденційних даних.
- Захист даних на рівні апаратного забезпечення.
- Автоматичне шифрування та декодування даних.

Недоліки:

- Складність використання та інтеграції у порівнянні з UserDefaults.
- Не підходить для зберігання великих обсягів даних або складних структур.
- Висока продуктивність не є ключовою перевагою.

Вибір методу зберігання даних для списку переглянутих фільмів

Для зберігання списку переглянутих фільмів у додатку Movie+ доцільно обрати Core Data. Він забезпечує потужний і гнучкий інструмент для роботи з даними, підтримку складних запитів та обробки даних. Використання Core Data дозволяє уникнути додавання зайвих зовнішніх залежностей, знижуючи складність проєкту, та забезпечує зручний графічний інтерфейс для моделювання даних у Xcode.

2.6 Огляд API

Для створення застосунку, який надає персоналізовані рекомендації фільмів, важливо обрати відповідні API, що забезпечать точні дані про фільми та можливості штучного інтелекту. Розглянемо два основні напрямки: API для отримання рейтингу фільмів та API для використання штучного інтелекту.

API для отримання рейтингу фільмів

Вибір API для отримання даних про фільми є критичним для забезпечення актуальної та достовірної інформації для користувачів. Серед популярних варіантів є OMDb API, Rotten Tomatoes API та TMDB (The Movie Database) API.

OMDb (Open Movie Database) API надає доступ до великої бази даних фільмів, серіалів та іншого відеоконтенту. Він дозволяє отримувати інформацію про назви, режисерів, акторів, рейтинги та багато іншого. [16]

Переваги OMDb API:

1. Відкрита база даних з великою кількістю інформації.
2. Підтримка різноманітних форматів запитів.

Недоліки OMDb API:

1. Обмежена кількість запитів безкоштовно.
2. Порівняно менша точність даних у порівнянні з іншими сервісам.

Rotten Tomatoes API надає доступ до рейтингів та рецензій фільмів від критиків та користувачів. Цей сервіс відомий своїми високими стандартами та об’єктивністю [17].

Переваги Rotten Tomatoes API:

1. Висока точність і надійність даних.
2. Відомі рейтинги та рецензії від критиків.

Недоліки Rotten Tomatoes API:

1. Обмежений доступ до API для сторонніх розробників.
2. Вимагає комерційної ліцензії для більшості функцій.

TMDB API надає доступ до великої бази даних фільмів, серіалів, акторів, рейтингів та багато іншого. Це один з найпопулярніших і найточніших сервісів для отримання інформації про фільми [18].

Переваги TMDB API:

1. Велика і постійно оновлювана база даних.
2. Підтримка багатьох мов та регіонів.

3. Безкоштовний доступ з можливістю розширених функцій.

Вибір API для отримання рейтингу фільмів

Для проєкту Movie+ доцільно обрати TMDB API. Він забезпечує доступ до великої та актуальної бази даних, підтримує багатомовність і надає безкоштовний доступ до більшості функцій, що є критично важливим для створення якісного продукту.

API для штучного інтелекту

Інтеграція штучного інтелекту у додаток дозволяє створювати персоналізовані рекомендації для користувачів. Розглянемо два популярних варіанти: Google Cloud AI API та OpenAI API.

Google Cloud AI API надає широкий спектр інструментів для машинного навчання, включаючи обробку природної мови, комп'ютерний зір, автоматичне навчання моделей та багато іншого [19].

Переваги Google Cloud AI API:

1. Потужні інструменти для машинного навчання.
2. Інтеграція з іншими сервісами Google Cloud.
3. Висока продуктивність і масштабованість.

Недоліки Google Cloud AI API:

1. Висока вартість для великих обсягів запитів.
2. Складність налаштування та інтеграції.

OpenAI API надає доступ до потужних моделей машинного навчання, таких як GPT-3, для створення інтелектуальних застосунків з можливістю обробки природної мови, генерації тексту та інших завдань [20].

Переваги OpenAI API:

1. Висока якість і точність моделей машинного навчання.
2. Широкий спектр можливостей для обробки природної мови.

3. Простота інтеграції та використання.

Недоліки OpenAI API:

1. Обмежена кількість безкоштовних запитів.

Вибір API для штучного інтелекту

Для проєкту Movie+ доцільно обрати OpenAI API. Він надає потужні інструменти для обробки природної мови та створення персоналізованих рекомендацій, що є важливим для досягнення мети проєкту. Незважаючи на обмеження безкоштовних запитів, його висока якість і простота інтеграції роблять його найкращим вибором.

2.7 Висновок

Дослідження засобів програмної реалізації для проєкту Movie+ охопило аналіз кількох важливих аспектів, необхідних для забезпечення ефективної розробки та функціональності застосунку. Вибір мови програмування показав, що Swift є найкращим варіантом завдяки своїй сучасності, безпеці, продуктивності та активній підтримці з боку спільноти розробників. Ця мова дозволяє створювати швидкі та надійні застосунки, що є критично важливим для досягнення цілей проєкту Movie+.

Аналіз інструментів розробки інтерфейсу користувача виявив, що UIKit є оптимальним вибором. Його зрілість, стабільність та сумісність з більшістю пристроїв iOS забезпечують широку аудиторію користувачів. Хоча SwiftUI є перспективним інструментом, його обмежена підтримка старіших версій iOS може стати значною перешкодою для охоплення широкої аудиторії.

Розгляд архітектурних паттернів, таких як MVC, MVVM, MVP та VIPER, дозволив визначити VIPER як найбільш підходящий варіант для проєкту Movie+.

Цей паттерн забезпечує чітке розділення відповідальностей, високу масштабованість та можливість ефективної роботи з великими обсягами даних і складними бізнес-логіками. VIPER також сприяє легкості тестування кожного компонента окремо, що підвищує надійність та якість коду.

Щодо управління залежностями, аналіз CocoaPods, Carthage та Swift Package Manager (SPM) показав, що SPM є найбільш доцільним вибором. Його інтеграція з Xcode, простота використання та високий рівень автоматизації роблять SPM ідеальним інструментом для проєкту Movie+.

Огляд API для отримання даних про фільми та можливостей штучного інтелекту показав, що TMDB API є найкращим варіантом для забезпечення актуальних та достовірних даних про фільми. Тим часом, OpenAI API забезпечує потужні інструменти для обробки природної мови та створення персоналізованих рекомендацій, що є важливим для досягнення мети проєкту.

Загалом, обрані засоби програмної реалізації забезпечують створення сучасного, надійного та ефективного iOS-застосунку Movie+, який надає користувачам персоналізовані рекомендації фільмів, зменшуючи час на пошук та покращуючи загальний користувацький досвід.

3 ПРОЄКТУВАННЯ ТА РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Опис предметної області

З розвитком технологій та збільшенням кількості кіноконтенту, знайти фільми, що відповідають індивідуальним смакам користувачів, стає все складніше. Саме тому виникає необхідність у розробці ефективних інструментів, які б спрощували процес пошуку та підбору фільмів, враховуючи особисті вподобання користувачів. Основною метою даного дослідження є створення iOS-застосунку, який використовує сучасні технології штучного інтелекту для надання персоналізованих рекомендацій щодо фільмів.

У контексті цього дослідження, предметна область включає аналіз та використання сучасних технологій штучного інтелекту та машинного навчання для персоналізації рекомендацій кіноконтенту. Головною метою є розробка iOS-застосунку, який здатен ефективно використовувати алгоритми для аналізу вподобань користувача та надання релевантних рекомендацій.

Основні компоненти предметної області:

1. Штучний інтелект та машинне навчання:

- Обробка природної мови: Використовується для аналізу текстових даних, таких як відгуки користувачів або описи фільмів, що дозволяє краще розуміти вподобання користувача.
- Рекомендаційні системи: Включають алгоритми, які аналізують минулі перегляди та взаємодії користувача для надання персоналізованих рекомендацій.

2. API для отримання даних про фільми:

- TMDB API: Надає доступ до великої бази даних фільмів, серіалів, акторів та рейтингової інформації, що є ключовим джерелом даних для рекомендаційного алгоритму.
- OpenAI API: Використовується для складних аналітичних задач, таких як аналіз відгуків або прогнозування вподобань на основі великих обсягів даних.

3. Мобільна платформа iOS:

- Swift: Основна мова програмування для розробки iOS-застосунків, яка забезпечує високу продуктивність та безпеку коду.
- UIKit та SwiftUI: Інструменти для створення інтерфейсу користувача, що забезпечують гнучкість та високу якість взаємодії з користувачем.

Виклики та можливості:

- Стабільність та продуктивність: Забезпечення швидкої та стабільної роботи застосунку навіть при обробці великих обсягів даних.
- Інтуїтивний інтерфейс: Розробка зручного та інтуїтивно зрозумілого інтерфейсу для користувачів різного рівня технічної підготовки.
- Персоналізація: Використання даних про вподобання користувачів для надання максимально точних рекомендацій, що підвищує задоволеність користувачів.

Висновок:

Дослідження предметної області включає аналіз сучасних технологій та методів, які дозволяють створити ефективний та зручний iOS-застосунок для персоналізованого пошуку фільмів. Використання штучного інтелекту та машинного навчання в поєднанні з потужними API дозволяє забезпечити високу якість рекомендацій та задоволеність користувачів, створюючи нові можливості для розвитку медіа-індустрії.

3.2 Архітектура системи

Архітектура iOS-додатку Movie+ для персоналізованого пошуку фільмів ґрунтується на сучасних технологіях та передових архітектурних шаблонах, що забезпечують високу продуктивність, зручність використання та точність даних. Ключовими елементами цієї архітектури є мова програмування Swift, архітектурний шаблон VIPER, фреймворк UIKit, а також додаткові бібліотеки та API для взаємодії з зовнішніми сервісами.

Основу додатку складає мова програмування Swift, яка забезпечує високу продуктивність, безпеку та легкість у підтримці коду. Використання Swift дозволяє розробляти ефективний та стійкий до помилок код, що особливо важливо для додатків, які обробляють персональні дані користувачів. Фреймворк UIKit надає потужний набір інструментів для створення інтуїтивно зрозумілих та естетично привабливих інтерфейсів. Завдяки цьому розробники можуть створювати гнучкі та адаптивні інтерфейси, які підлаштовуються під різні розміри екранів.

Основним архітектурним підходом у додатку є шаблон VIPER (View-Interactor-Presenter-Entity-Router). VIPER забезпечує чітке розділення відповідальностей між компонентами, що сприяє підвищенню модульності та тестованості коду. VIPER дозволяє розділити логіку додатку на окремі модулі, що полегшує розробку, тестування та підтримку.

Для взаємодії з зовнішніми API та обробки HTTP-запитів використовується бібліотека Alamofire. Вона забезпечує простий і зручний спосіб відправки та обробки запитів, що дозволяє ефективно працювати з зовнішніми сервісами, такими як TMDB API та OpenAI API. TMDB API надає доступ до великої бази даних фільмів, включаючи їх рейтинги, акторський склад та іншу інформацію. OpenAI API використовується для створення персоналізованих рекомендацій на основі вподобань користувача.

Для обробки та відображення зображень використовується бібліотека Kingfisher. Вона дозволяє завантажувати зображення через URL, кешувати їх та відображати у додатку. Це значно покращує користувацький досвід, оскільки зображення завантажуються швидко та без затримок.

Важливим елементом додатку є анімації, які роблять взаємодію з інтерфейсом більш привабливою та інтуїтивно зрозумілою. Для створення анімацій використовується бібліотека Lottie, яка дозволяє легко інтегрувати складні анімації у додаток.

Діаграму розгортання можна побачити на рисунку 8.

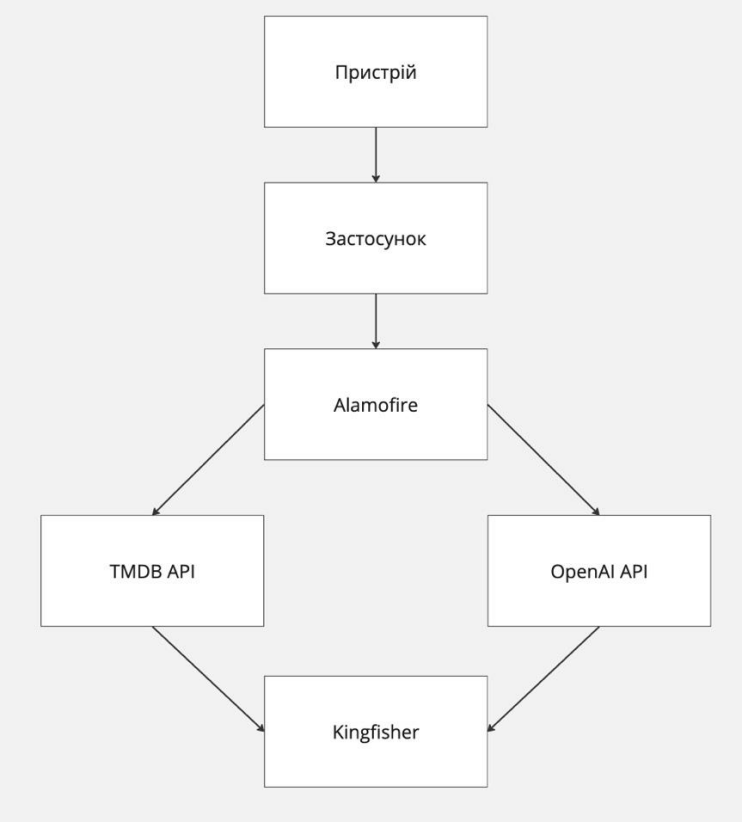

Рисунок 8 — *Архітектура системи*

3.3 Функціональні вимоги системи

iOS-додаток Movie+ для персоналізованого пошуку фільмів забезпечує користувачам ефективний та зручний спосіб знаходження фільмів, які відповідають їхнім вподобанням. Для досягнення цієї мети, система має відповідати наступним функціональним вимогам:

1. Пошук фільмів
- Додаток повинен надавати користувачам можливість здійснювати пошук фільмів за назвою, жанром, акторами та іншими критеріями.
- Використання TMDB API для отримання інформації про фільми та їх рейтингів.
2. Персоналізовані рекомендації
- Додаток повинен генерувати персоналізовані рекомендації фільмів на основі вподобань та історії переглядів користувачів.
- Використання OpenAI API для аналізу вподобань користувачів та створення рекомендацій.
3. Перегляд деталей фільму
- Користувачі повинні мати можливість переглядати детальну інформацію про фільми, включаючи опис, акторський склад, рейтинг та трейлери.
4. Додавання фільмів до списків
- Додаток повинен дозволяти користувачам додавати фільми до списків бажаних переглядів, переглянутих фільмів та інших персоналізованих списків.
- Збереження даних списків локально на пристрої.
5. Перегляд трейлерів
- Додаток повинен дозволяти користувачам переглядати трейлери фільмів.
- Використання TMDB API для отримання посилань на трейлери.
6. Інтуїтивний та зручний інтерфейс

- Додаток повинен мати інтуїтивно зрозумілий та зручний інтерфейс, що забезпечує легкість навігації та взаємодії.
- Використання UIKit для створення адаптивного та привабливого інтерфейсу.

  UseCase діаграму можна побачити на рисунку 9.

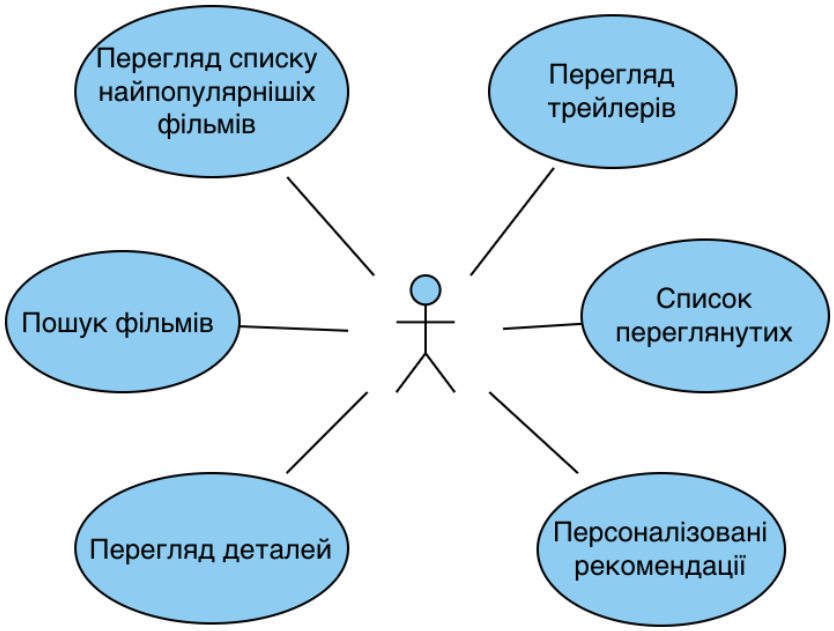

Рисунок 9 — *UseCase діаграма можливостей взаємодії користувача з застосунком*

Висновок

Ці вимоги включають ефективний пошук та персоналізовані рекомендації фільмів, детальний перегляд інформації, збереження фільмів у списки, а також можливість перегляду трейлерів. Зосередженість на інтуїтивному та зручному ін-

терфейсі, а також використання сучасних технологій, забезпечує створення додатку, який не тільки відповідає потребам користувачів, але й надає їм зручний інструмент для пошуку та насолоди улюбленими фільмами.

3.4 Вимоги до апаратного та програмного забезпечення

Для роботи додатку Movie+ потрібна операційна система iOS 13 або вище. Враховуючи важливість стабільної роботи додатку, особливо при роботі з великою кількістю даних та інтеграцією з зовнішніми сервісами, а також використовуючи останні технології та підходи, дотримання мінімальних вимог є обов'язковим.

Мінімальні вимоги**:**

- Операційна система: iOS 13.0 або вище.
- Пристрій: iPhone 6s або новіший.
- Процесор: A9 чип або новіший.
- Оперативна пам'ять: 2 ГБ RAM.
- Пам'ять: 30 МБ для встановлення додатку та забезпечення його функціонування.

Дотримання цих вимог гарантує стабільну та ефективну роботу додатку. Це забезпечить користувачам швидкий та надійний доступ до всіх функцій, включаючи перегляд трейлерів, збереження та синхронізацію даних. Відповідність цим вимогам сприятиме високій якості користувацького досвіду, мінімізуючи ризики технічних проблем та забезпечуючи безперебійну роботу додатку навіть при інтенсивному використанні.

3.5 Модулі і алгоритми

Проєктування та реалізація iOS-застосунку Movie+ базується на використанні різноманітних алгоритмів та модулів, що забезпечують його ефективне функціонування. У цьому розділі буде розглянуто такі аспекти:

1. Реалізація алгоритму – ключові процеси та методи обробки даних, які забезпечують функціонування застосунку. Перелік класів та їх функцій — короткий опис кожного класу та його ролі в структурі застосунку.
2. Структура модуля — перелік всіх класів архітектури з описом їх призначень.
3. Аналіз основних класів — приклади коду, що ілюструють реалізацію та взаємодію основних компонентів та огляд окремих функцій, які відіграють значну роль у загальній архітектурі застосунку.

Цей розділ надасть глибоке розуміння структури та принципів роботи застосунку Movie+, що є критично важливим для його успішної розробки та подальшого вдосконалення.

Опис роботи OpenAI API

OpenAI API — це інтерфейс програмування додатків, розроблений компанією OpenAI, що надає доступ до різноманітних моделей штучного інтелекту для виконання завдань, пов’язаних з обробкою природної мови, генерацією тексту, перекладом та іншими завданнями. OpenAI використовує передові моделі машинного навчання, такі як GPT-3, що здатні розуміти та генерувати текст на основі заданих запитів (prompts).

Як працює OpenAI API:

1. Модель: Основою OpenAI API є модель штучного інтелекту GPT-3, яка тренується на великому обсязі текстових даних, що дозволяє їй розуміти контекст та генерувати логічно пов'язані відповіді.
2. Запит (Prompt): Користувач формулює текстовий запит (prompt), що описує завдання, яке необхідно виконати. Наприклад, це може бути запит на генерацію рекомендацій фільмів.
3. Параметри запиту: Користувач також може вказати додаткові параметри, такі як temperature, max_tokens, top_p, frequency_penalty, presence_penalty, що впливають на характер та якість відповіді.
4. Обробка запиту: Модель обробляє запит та генерує відповідь, використовуючи свою здатність розуміти контекст та генерувати релевантний текст.
5. Відповідь: OpenAI API повертає згенерований текст, який можна використовувати у додатку для виконання конкретних завдань, таких як рекомендації фільмів.

Реалізація алгоритму

Для реалізації персоналізованих рекомендацій у застосунку Movie+ використовується OpenAI API у поєднанні з TMDB API. Цей підхід дозволяє забезпечити точні рекомендації фільмів, відповідно до вподобань користувачів.

1. Отримання даних від користувача.

По натисканню на кнопку запиту беремо фільм, з якого був надісланий запит, і список вже переглянутих, який зберегли на основі запитів користувача.

2. Формування запиту до OpenAI API.

На основі введених даних формується запит до OpenAI API. Запит містить текстовий prompt, що визначає завдання для моделі, а також параметри, які впливають на характер відповіді.

Параметри запиту:

- prompt: Текстовий запит, що формулює завдання для моделі.
- temperature: Визначає рівень варіативності відповідей. Низькі значення (наприклад, 0.2) призводять до більш передбачуваних і консервативних відповідей. Вищі значення (до 1.0) додають більше креативності і випадковості. Для нашого запиту встановлено значення 0.7, що забезпечує баланс між креативністю та передбачуваністю.
- max_tokens (1 - 2048): Задає максимальну кількість токенів у відповіді. У цьому випадку, 200 токенів достатньо для отримання детальних рекомендацій.
- top_p (0 - 1): Обмежує вибір токенів до найбільш ймовірних, сума ймовірностей яких складає це значення. Значення 1.0 включає всі ймовірні варіанти.
- frequency_penalty (0 - 2): Регулює частоту появи тих самих токенів у відповіді. Значення 0.5 зменшує ймовірність повторення тих самих фраз. Мінімальне значення 0 означає відсутність штрафу за частоту, максимальне 2 накладає сильний штраф.
- presence_penalty (0 - 2): Контролює, чи будуть у відповіді присутні нові токени. Значення 0.0 означає, що нові токени не будуть пріоритетними, тоді як максимальне значення 2 стимулює появу нових токенів.

  Приклад запиту в json форматі можна побачити на лістингу 1.

Лістинг 1 *Приклад тіла запиту до OpenAI API*

```
{
     "prompt": "Recommend movies similar to those with
the following actors or in the same genre, excluding the
following watched movies. Actors: [actor1, actor2, actor3].
Genre: [genre1, genre2]. Watched movies: [movie1, movie2,
movie3].",
```

```
        "temperature": 0.7,
        "max_tokens": 200,
        "top_p": 1.0,
        "frequency_penalty": 0.5,
        "presence_penalty": 0.0
    }
```

3. Отримання результатів від OpenAI API.

OpenAI API повертає список рекомендованих фільмів у текстовому форматі.

4. Інтеграція з TMDB API для отримання ID фільмів.

Використовуючи отримані назви фільмів, асинхронно надсилаються запити до TMDB API для отримання їхніх ID та додаткової інформації. Наприклад, запит до TMDB API можна побачити на лістингу 2:

Лістинг 2 *Приклад запиту до* TMDB API

```
https://api.themoviedb.org/3/search/movie?query=MovieName
First&api_key=API_KEY
```

5. Обробка відповідей від TMDB API.

TMDB API повертає інформацію про фільми, включаючи їхні ID.

6. Перевірка відповідності категорії та акторів на клієнті.

Після отримання даних від TMDB API, на клієнті перевіряється, чи відповідають фільми заданим категоріям або акторам. Якщо фільм відповідає вимогам, він додається до списку рекомендацій для користувача.

7. Відображення рекомендацій користувачу.

Виведення списку рекомендованих фільмів у зручному форматі. Надання можливості перегляду додаткової інформації про фільми та додавання їх до списку для перегляду.

Цей підхід забезпечує точність та релевантність рекомендацій, одночасно зручну для користувача перевірку відповідності категорії та акторів. Інтеграція з TMDB API дозволяє отримати детальні дані про фільми, включаючи їхні ID, що значно спрощує процес створення персоналізованих рекомендацій.

Структура модуля

У VIPER архітектурі кожен екран інтерфейсу iOS-додатку складається з семи ключових компонентів: ViewController, Interactor, Presenter, Entity, Router, DataSource та ViewState. Далі наведемо детальний опис кожного з цих компонентів.

ViewController

ViewController відповідає за відображення інтерфейсу користувача та обробку взаємодії з ним. Основні функції включають відображення даних, отриманих від Presenter, обробку подій користувача, таких як натискання кнопок або введення тексту, та виклик методів Presenter для оновлення даних або виконання певних дій. ViewController містить об’єкт відповідного Presenter та взаємодіє безпосередньо з його методами.

Interactor

Interactor відповідає за бізнес-логіку додатку та обробку даних. Він отримує дані від Entity, обробляє їх та передає Presenter. Interactor також відповідає за виконання запитів до зовнішніх сервісів або бази даних, обробку результатів та виконання необхідних бізнес-процесів. Interactor не взаємодіє безпосередньо з View, що забезпечує чітке розділення відповідальностей.

Presenter

Presenter виступає посередником між View та Interactor. Він отримує дані від Interactor, обробляє їх, форматує для відображення у View та керує бізнес-логікою програми. Presenter забезпечує логіку відображення та обробки даних, звільняючи View від цих завдань. Presenter тримає у собі об’єкт View для виклику необхідних оновлень інтерфейсу.

Entity

Entity відповідає за управління моделями даних додатку. Вона містить структури та методи для зберігання даних, їх обробки, включаючи обчислення, перетворення та валідацію. Entity забезпечує Interactor усіма необхідними даними для обробки та виконання бізнес-логіки.

Router

Router відповідає за навігацію між екранами додатку. Його основні функції включають виконання переходів між екранами, підготовку даних для наступного екрану та взаємодію з іншими роутерами системи для забезпечення очікуваної навігації. Router полегшує переміщення користувача між різними частинами додатку, забезпечуючи послідовну структуровану розробку.

DataSource

DataSource відповідає за постачання даних для відображення у View. Він може включати методи для завантаження, кешування та управління даними, які необхідні для роботи з інтерфейсом користувача. DataSource забезпечує Presenter актуальними даними, необхідними для відображення та обробки.

ViewState

ViewState відповідає за зберігання стану інтерфейсу користувача. Він містить інформацію про поточний стан екрану, включаючи дані, які відображаються, стан завантаження, помилки та інші властивості, що впливають на відображення інтерфейсу. ViewState допомагає зберігати консистентний стан екрану під час взаємодії користувача з додатком.

Висновок

Такий підхід до організації коду в архітектурі VIPER з додатковими компонентами DataSource та ViewState забезпечує чітке розділення відповідальностей, покращує тестованість та полегшує підтримку коду. Це сприяє більш структурованій та надійній розробці додатку.

Аналіз основних класів

Розглянемо основні модулі додатку:

1. MovieList — модуль для відображення списку фільмів. Користувач може переглядати список доступних фільмів та вибирати для перегляду детальнішої інформації. Модуль включає функції для фільтрації та сортування фільмів за різними критеріями.
2. MovieDetails — модуль для відображення детальної інформації про обраний фільм. Він включає опис, акторський склад, рейтинг та інші дані. Користувач може переглядати рецензії та додаткові матеріали про фільм.
3. VideoPlayer — модуль для відтворення відео. Він дозволяє користувачам переглядати трейлери, кліпи та інші відеоматеріали, пов’язані з фільмом. Модуль підтримує функції керування відтворенням, такі як пауза, відмотування та зміна якості відео.

4. Recommendations — модуль для відображення рекомендацій фільмів на основі вподобань користувача. Він використовує алгоритми машинного навчання та API OpenAI для надання персоналізованих рекомендацій. Користувач може переглядати список рекомендованих фільмів та додавати їх до свого списку перегляду.

Ці модулі забезпечують основні функції додатку, дозволяючи користувачам зручно переглядати, вибирати та відтворювати фільми, а також отримувати персоналізовані рекомендації на основі їхніх уподобань.

Структуру основних модулів можна побачити на рисунку 10.

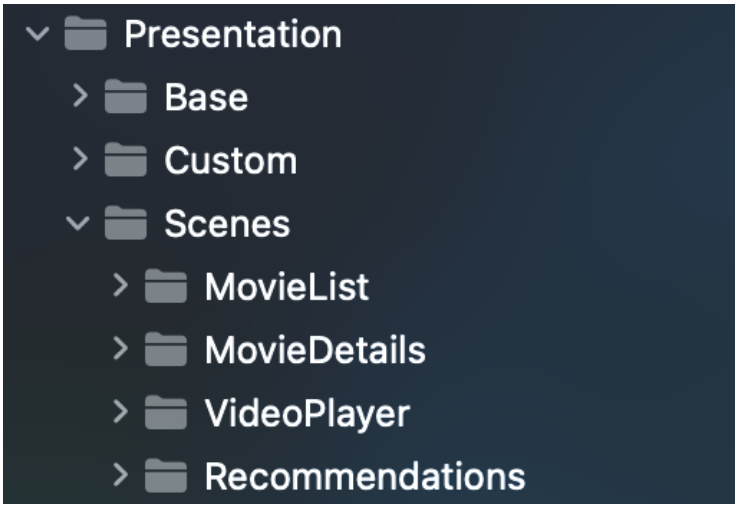

Рисунок 10 — *Структура модулів додатку*

MovieDetailsScreen

MovieDetailsScreen — модуль призначений для надання користувачам детальної інформації про обраний фільм. Інтерфейс користувача забезпечує зручний доступ до інформації про фільм, таких як опис, акторський склад, рейтинг, а також надає можливість користувачам виконувати додаткові дії за допомогою трьох кнопок: переглянути трейлер, знайти подібні фільми та знайти фільми з таким самим акторським складом.

Коли користувач відкриває екран деталей фільму, ViewController викликає метод viewDidLoad, який ініціалізує об’єкт Presenter і викликає метод perform(action:) з параметром viewDidLoad. Presenter обробляє цей запит і передає його до Interactor для отримання деталей фільму та списку відео. Під час завантаження даних відображається індикатор завантаження. Після отримання даних Presenter або відправляє команду до ViewState для побудови інтерфейсу з отриманими даними, або відображає повідомлення про помилку, якщо дані не вдалося завантажити. Після цього ViewState формує представлення екрану, яке передається до View, що оновлює інтерфейс користувача та забезпечує доступ до всіх необхідних даних.

Опис та реалізація класів і методів у модулі MovieDetailsScreen формують чітке та структуроване бізнес-представлення. Це забезпечує зручність та ефективність взаємодії користувачів з додатком, сприяючи швидкому доступу до релевантної інформації про фільми, а також надає можливість легкого розширення та підтримки додатку в майбутньому.

Повний перелік методів класу MovieDetailsViewController модулю MovieDetails можна побачити на лістингу 3.

Лістинг 3 *Перелік методів класу* MovieDetailsViewController

```
final class MovieDetailsViewController {
    override func viewDidLoad()
    func render(with title: String)
    func setDataSource(snapshot: Snapshot
    func showAlert(title: String?, onDismiss: @escaping
(() -> Void))
    private func makeCollectionViewLayout() ->
        UICollectionViewCompositionalLayout
}
```

Переглянемо кожен з методів класу MovieDetailsViewController:

1. override func viewDidLoad() — викликається при завантаженні представлення, ініціалізує Presenter та викликає метод для завантаження даних.
2. func render(with title: String) — встановлює заголовок екрану з назвою фільму.
3. func setDataSource(snapshot: Snapshot) — оновлює джерело даних колекційного представлення.
4. func showAlert(title: String?, message: String?, onDismiss: @escaping (() -> Void)) — відображає повідомлення про помилку або успіх операції.
5. private func makeCollectionViewLayout() -> UICollectionViewCompositionalLayout — створює і налаштовує макет для колекційного представлення.

На лістингу 4 можна побачити повний перелік методів класу MovieDetailsPresenter модулю MovieDetailsScreen.

Лістинг 4 *Перелік методів класу* MovieDetailsPresenter

```
final class MovieDetailsPresenterImpl {
    init(
         router: MovieDetailsRouter,
         interactor: MovieDetailsInteractor,
         viewStateFactory: MovieDetailsViewStateFactory,
         configuration: MovieDetailsConfiguration
         )
    func inject(view: MovieDetailsView)
    func perform(action: MovieDetailsAction)
    private func performViewDidLoadAction()
    private func performPlaybackAction()
    private func fetchMovieData() async
    @MainActor private func updateView() async
```

```
        @MainActor private func updateDataSource(viewState:
MovieDetailsViewState) async
    }
```

Переглянемо кожен з методів класу MovieDetailsPresenterImpl:

1. init(router: MovieDetailsRouter, interactor: MovieDetailsInteractor, viewStateFactory: MovieDetailsViewStateFactory, configuration: MovieDetailsConfiguration) — ініціалізує екземпляр класу з необхідними залежностями.
2. func inject(view: MovieDetailsView) — встановлює View для Presenter.
3. func perform(action: MovieDetailsAction) — виконує дії на основі переданого параметра.
4. private func performViewDidLoadAction() — викликається при завантаженні екрану, ініціалізує завантаження даних.
5. private func performPlaybackAction() — виконує дію відтворення відео.
6. private func fetchMovieData() async — завантажує дані про фільм та відео з мережі.
7. @MainActor private func updateView() async — оновлює інтерфейс користувача після отримання даних.
8. @MainActor private func updateDataSource(viewState: MovieDetailsViewState) async — оновлює джерело даних для колекційного представлення.

Переглянути діаграму класів модулю MovieDetails можна на рисунку 11 та 12.

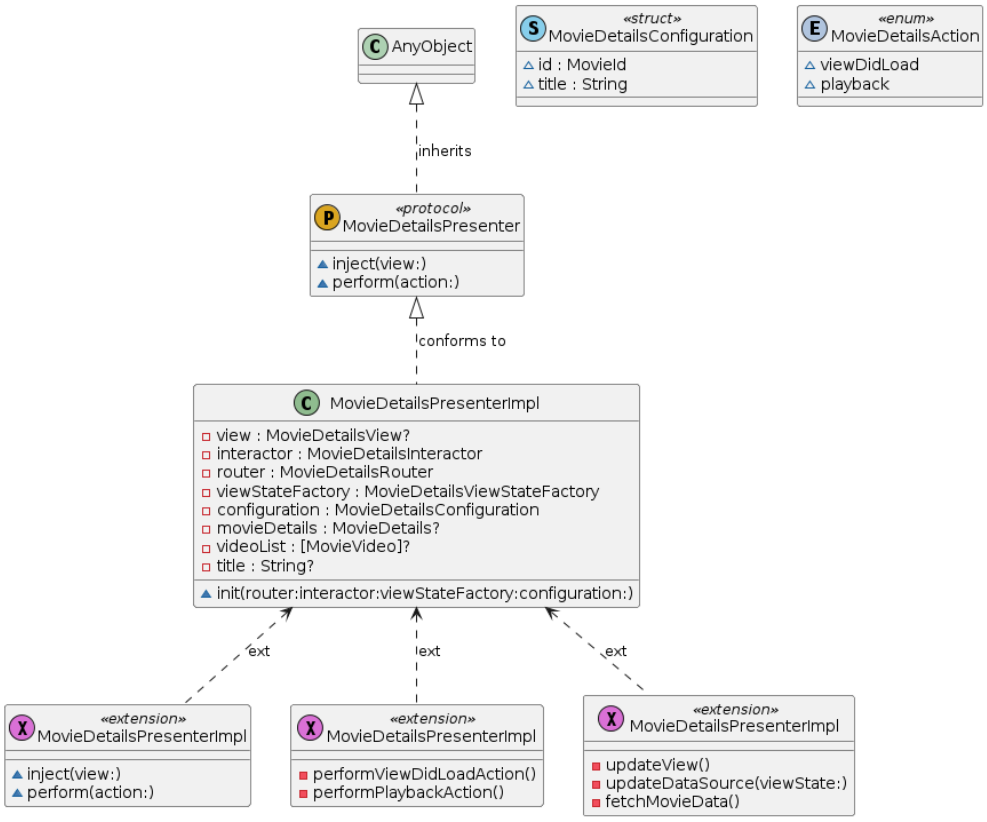

Рисунок 11 — *Діаграма класу presenter модуля* MovieDetails

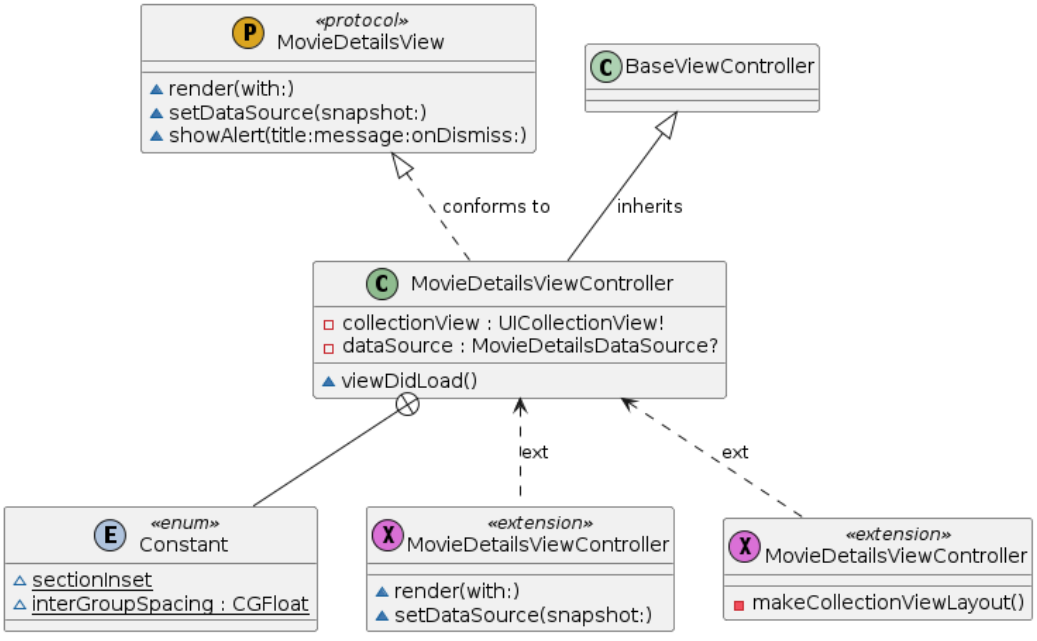

Рисунок 12 — *Діаграма класу ViewController модуля* MovieDetails

3.6 Структури даних

Враховуючи вибір Core Data як основного способу зберігання даних у додатку Movie+, у цьому розділі буде детально описано структури даних, які використовуються для зберігання та обробки інформації про фільми. Використання Core Data дозволяє ефективно організувати зберігання даних у вигляді об'єктів та їх зв'язків, що забезпечує високу продуктивність та зручність роботи з даними.

На лістингу 5 можна побачити модель даних фільму.

Лістинг 5 Модель даних фільму

```
struct MoviesResponse: Codable {
    let results: [Movie]
}

typealias MovieId = Int

struct Movie: Codable {
    let backdropPath: String?
    let genreIDS: [Int]
    let id: MovieId
    let releaseDate: String?
    let title: String
    let voteAverage: Double

    enum CodingKeys: String, CodingKey {
        case backdropPath = "backdrop_path"
        case genreIDS = "genre_ids"
        case id
        case releaseDate = "release_date"
        case title
```

```
        case voteAverage = "vote_average"
    }
}
```

Модель Movie зберігає основну інформацію про фільм, включаючи його ідентифікатор, назву, опис, дату виходу, шляхи до постерів, жанри та середню оцінку. Ця структура використовується для відображення фільмів у списках рекомендацій та результатах пошуку. Модель MoviesResponse використовується для отримання списку фільмів з API.

На лістингу 6 можна побачити модель детальної інформації про фільм.

Лістинг 6 Модель детальної інформації про фільм

```
struct MovieDetails: Codable {
    struct CountryName: Codable {
        var name: String
    }

    let id: MovieId
    let budget: Int
    let revenue: Int
    let productionCountries: [CountryName]
    let releaseDate: String?
    let backdropPath: String?
    let title: String
    let originalTitle: String
    let overview: String
    let voteAverage: Double

    enum CodingKeys: String, CodingKey {
        case id
```

```
            case budget
            case revenue
            case productionCountries = "production_countries"
            case releaseDate = "release_date"
            case backdropPath = "backdrop_path"
            case title
            case originalTitle = "original_title"
            case overview
            case voteAverage = "vote_average"
        }
    }
```

Модель MovieDetails зберігає детальну інформацію про фільм, включаючи оригінальну назву, опис, дату виходу, жанри, країни виробництва, тривалість, бюджет, доходи, середню оцінку та шляхи до зображень. Ця структура використовується для відображення повної інформації про фільм на окремій сторінці.

Переглянути ER діаграму моделі MovieVideo можна на рисунку 13.

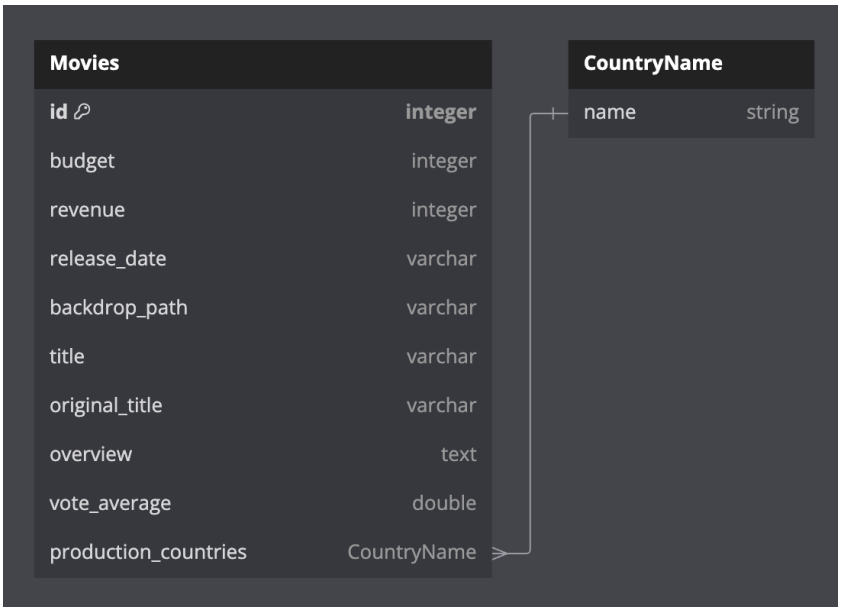

Рисунок 13 — *ER діаграма*

На лістингу 7 можна побачити модель відео фільму.

Лістинг 7 Модель відео фільму

```
struct MovieVideosResponse: Codable {
    let results: [MovieVideo]
}

struct MovieVideo: Codable {
    let name: String
    let key: String
    let official: Bool
    let publicationDate: Date?

    enum CodingKeys: String, CodingKey {
        case name
        case key
        case official
        case publicationDate = "published_at"
    }

    init(from decoder: Decoder) throws {
        let container = try decoder.container(keyedBy:
CodingKeys.self)
        self.name = try container.decode(String.self,
forKey: .name)
        self.key = try container.decode(String.self,
forKey: .key)
        self.official = try container.decode(Bool.self,
forKey: .official)
```

```
            let publicationDateString = try container.de-
code(String.self, forKey: .publicationDate)
            self.publicationDate = DateHelper.toServ-
erDate(from: publicationDateString)
        }
    }
```

3.7 Проєкт інтерфейсу

Проєкт інтерфейсу iOS-додатку Movie+ має на меті забезпечити користувачам зручний, інтуїтивно зрозумілий та естетично привабливий досвід взаємодії з додатком. Використання сучасного стеку технологій, таких як Swift, UIKit, VIPER архітектура, а також бібліотек Alamofire та Kingfisher, дозволяє створити продуктивний та стабільний додаток з привабливим інтерфейсом. Інтерфейс включає кілька ключових екранів, які охоплюють основні функціональні можливості додатку, такі як завантаження, перегляд популярних фільмів, пошук, деталі фільму та перегляд трейлерів. Завдяки продуманій архітектурі та використанню передових технологій, додаток забезпечує високий рівень користувацького досвіду.

Екран завантаження

Перший екран, який бачить користувач при відкритті додатку, є екран завантаження. Він відображає анімацію завантаження, що інформує користувача про процес підготовки додатку до роботи.

Інтерфейс екрану завантаження зображено на рисунку 14.

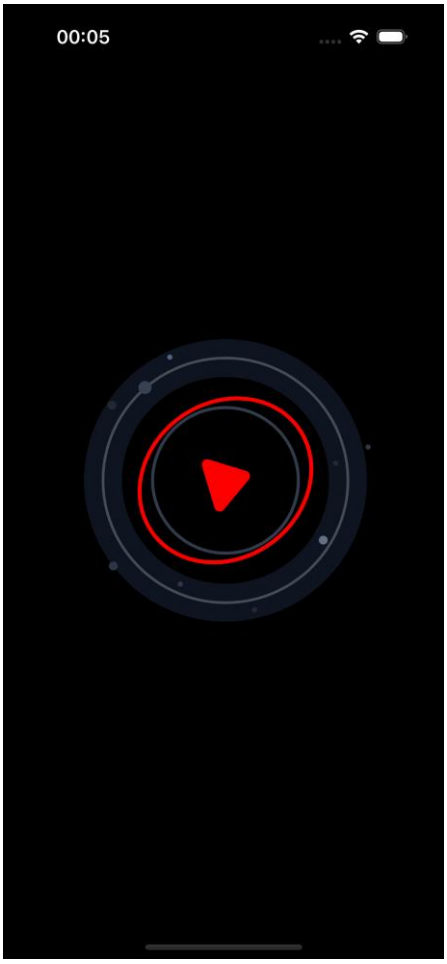

Рисунок 14 — *Екран завантаження*

Екран перегляду популярних фільмів

Після завантаження користувач потрапляє на екран перегляду популярних фільмів. Цей екран показує список фільмів з зображеннями, назвами та рейтингами. Додано пагінацію, яка автоматично завантажує нові елементи, коли користувач прокручує до останніх п'яти елементів списку.

Інтерфейс екрану перегляду популярних фільмів зображено на рисунку 15.

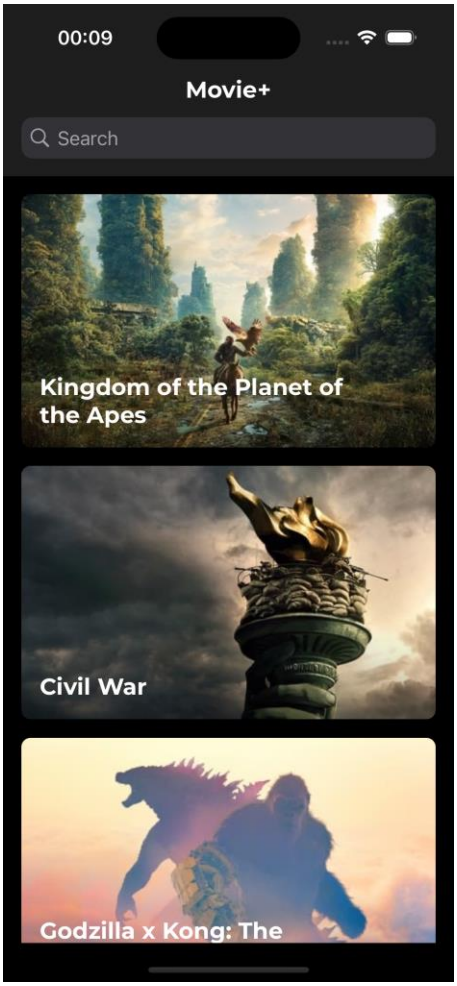

Рисунок 15 — *Екран перегляду популярних фільмів*

Стан пошуку

Екран пошуку дозволяє користувачам знаходити фільми за ключовими словами. Пошук виконується кожні 0,5 секунд після введення нового символу, що запобігає надмірному навантаженню на сервер запитами. Пошукові результати відображаються в реальному часі під полем введення.

Інтерфейс екрану пошуку зображено на рисунку 16.

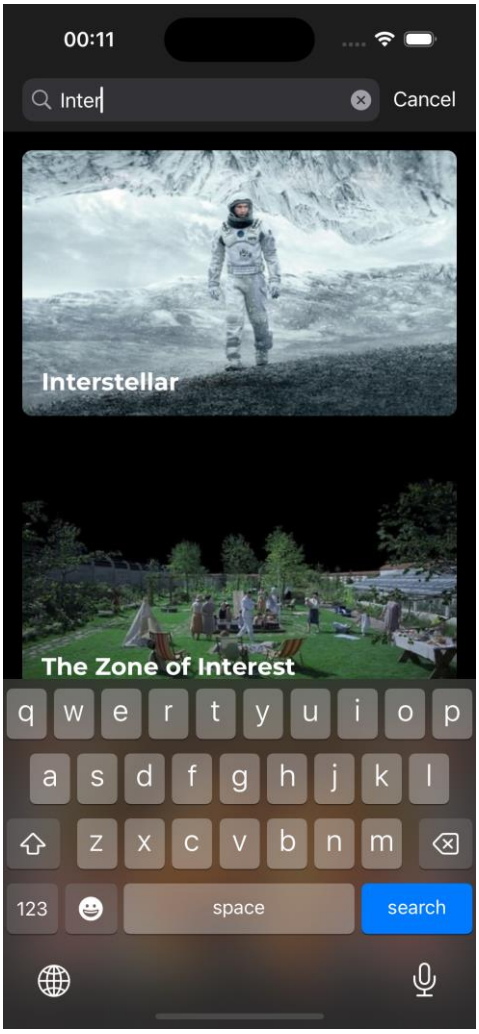

Рисунок 16 — *Інтерфейс екрану пошуку*

Екран деталей фільму

На екрані деталей фільму користувач може переглянути докладну інформацію про обраний фільм. Інформація включає опис, рейтинг, дату релізу, країни виробництва, бюджет та доходи. Крім того, є кнопки для перегляду трейлера та пошуку подібних фільмів з використанням алгоритмів штучного інтелекту.

Інтерфейс екрану деталей зображено на рисунку 17.

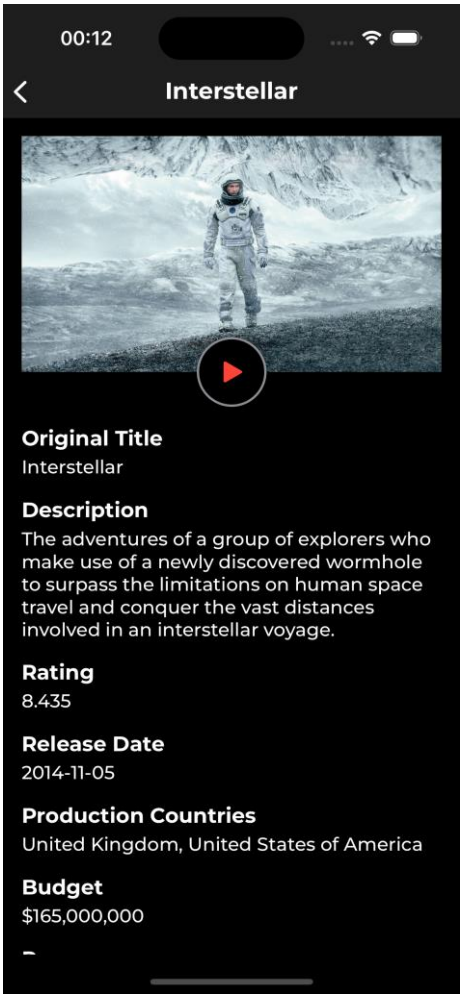

Рисунок 17 — *Екран деталей фільму*

Інтерфейс перегляду трейлера

Інтерфейс перегляду трейлера забезпечує зручне відтворення відео з можливістю перемотування та переходу на повний екран.

Інтерфейс перегляду трейлера зображено на рисунку 18.

Рисунок 18 — *Інтерфейс перегляду трейлера*

Висновок

Проєкт інтерфейсу iOS-додатку Movie+ був розроблений з урахуванням потреб користувачів та сучасних тенденцій в дизайні мобільних застосунків. Кожен екран і стан інтерфейсу забезпечують зручність, швидкодію та естетичну привабливість, сприяючи позитивному користувацькому досвіду.

3.8 Реалізація і тестування

Реалізація

Для розробки iOS-застосунку Movie+ використовувалися сучасні технології та архітектурні підходи, що забезпечують високу продуктивність та масштабованість. Поточна версія додатку має наступні характеристики:

- Об'єм коду: приблизно 4 000 рядків.
- Розмір: 25 мегабайти.

Тестування

Для забезпечення високої якості та стабільності додатку було проведено комплексне тестування на різних пристроях з версіями iOS від 13 до 17. Тестування включало перевірку основних сценаріїв взаємодії користувача, таких як:

1. Перегляд деталей фільмів.
2. Пошук та фільтрація фільмів.
3. Перегляд трейлерів фільмів.
4. Збереження вподобаних фільмів.
5. Рекомендації на основі переглянутих фільмів.

Тестування здійснювалося наступними методами:

- Функціональне тестування: Перевірка кожної функціональної можливості додатку, щоб переконатися в її правильній роботі відповідно до вимог.
- Тестування інтерфейсу користувача: Оцінка зручності використання, перевірка коректності відображення інтерфейсу на різних розмірах екранів та при різних налаштуваннях системи.
- Memory Leak тестування: Перевірка на наявність витоків пам'яті під час виконання основних сценаріїв використання додатку. Виявлено та усунено всі виявлені проблеми з витоками пам'яті, що дозволило забезпечити стабільну роботу додатку навіть при тривалому використанні.

Описані функціональні вимоги визначають ключові можливості iOS-додатку Movie+, які включають перегляд деталей фільмів, пошук та фільтрацію фільмів, перегляд трейлерів, збереження вподобаних фільмів та отримання персоналізованих рекомендацій. Додаток пройшов всебічне тестування, включаючи функціональні тестування, та тестування на витоки пам’ять, що підтвердило його високу стабільність та продуктивність. Це забезпечує користувачам надійну та

зручну платформу для ефективного пошуку та перегляду фільмів, що відповідає їхнім очікуванням та вимогам.

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було проведено комплексне дослідження технологій розробки мобільних застосунків з використанням штучного інтелекту для персоналізованого пошуку фільмів. Аналіз сучасних аналогів дозволив визначити ефективні методи і підходи для реалізації даного проєкту.

Основною мовою програмування для розробки було обрано Swift, завдяки її продуктивності, безпеці та сучасному синтаксису. Це забезпечило створення надійного та високопродуктивного застосунку. Для реалізації користувацького інтерфейсу було використано UIKit, який надав широкий набір інструментів і компонентів для створення інтуїтивно зрозумілого і привабливого інтерфейсу. Використання Swift Package Manager (SPM) як інструменту управління залежностями забезпечило легку інтеграцію з Xcode та автоматичне оновлення бібліотек, що значно спростило процес розробки та підтримки проєкту.

Для отримання актуальних даних про фільми було обрано TMDB API, який надав доступ до великої бази даних з детальною інформацією про фільми та серіали. Для реалізації персоналізованих рекомендацій було використано OpenAI API, який забезпечив високоякісні моделі для обробки природної мови та машинного навчання.

При розробці архітектури застосунку було вирішено використовувати патерн VIPER, який забезпечує чітке розділення відповідальностей між різними компонентами системи. Це сприяло підвищенню тестованості, гнучкості та масштабованості застосунку, а також полегшило його підтримку та розвиток.

Завдяки обраним технологіям та інструментам, вдалося створити сучасний та надійний iOS-застосунок Movie+, який надає користувачам можливість швидкого та зручного пошуку фільмів відповідно до їхніх особистих вподобань, забезпечуючи високу якість користувацького досвіду.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Machine Learning in Content Personalization URL: https://www.sitecore.com/blog/ai/machine-learning-for-personalization (дата звернення 16.01.2024).
2. Фролов Д. В., Полякова Н. П., доцент, к.т.н.. Розробка та реалізація iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача. Збірник наукових праць студентів, аспірантів, докторантів і молодих вчених «Молода наука-2024» / Запорізький національний університет. Запоріжжя: ЗНУ, 2024. Т.5. С. 125-126.
3. Di Francesco, P., Lago, P., & Malavolta, I. (2019). Architecting with microservices: A systematic mapping study. Journal of Systems and Software. URL: https://doi.org/10.1016/j.jss.2019.01.001 (дата звернення 20.02.2024).
4. Machine Learning for Personalization. Sitecore. URL: https://www.sitecore.com/blog/ai/machine-learning-for-personalization (дата звернення 20.02.2024).
5. OpenAI API. OpenAI. URL: https://openai.com/api/ (дата звернення 20.02.2024).
6. The Power of Modular Software Architecture: Boosting Reusability and Maintainability. Moments Log. URL: https://www.momentslog.com/2020/01/02/power-of-modular-software-architecture/ (дата звернення 20.02.2024).
7. Netflix Media Center. URL: https://media.netflix.com/ (дата звернення 15.03.2024).
8. Letterboxd Limited. URL: https://letterboxd.com/ (дата звернення 15.03.2024).
9. MovieLens. URL: https://movielens.org/ (дата звернення 15.03.2024).
10. Objective-C. Wikipedia site. URL: https://uk.wikipedia.org/wiki/Objective-C (дата звернення 19.03.2024)

11. Swift. Wikipedia site. URL: https://uk.wikipedia.org/wiki/Swift (дата звернення 19.03.2024)
12. UIKit. Apple Developer site. URL: https://developer.apple.com/documentation/uikit (дата звернення 25.03.2024)
13. SwiftUI. Apple Developer site. URL: https://developer.apple.com/xcode/swiftui/ (дата звернення 25.03.2024)
14. MVC, MVP, MVI, MVVM and VIPER Design Patterns. Medium site. URL: https://medium.com/@pinarkocak/mvc-mvp-and-mvvm-design-patterns-82317d6efeac (дата звернення 26.03.2024)
15. CocoaPods vs Carthage vs SPM (Dependency manager in Swift). URL: https://manasaprema04.medium.com/dependency-managers-in-swift-d6a01e7a29a8 (дата звернення 26.03.2024)
16. OMDb API. URL: https://www.omdbapi.com/ (дата звернення 04.04.2024)
17. Rotten Tomatoes API. URL: https://rapidapi.com/collection/rotten-tomatoes-api (дата звернення 04.04.2024)
18. TMDB API. URL: https://www.themoviedb.org/ (дата звернення 04.04.2024)
19. AI APIs for Google Cloud. URL: https://cloud.google.com/ai/apis (дата звернення 04.04.2024)
20. OpenAI API. URL: https://openai.com/index/openai-api/ (дата звернення 04.04.2024)

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Фролов Данніл Володимирович, студент 4 курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-214@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «Розробка та реалізація iOS-додатку для персоналізованого пошуку фільмів із використанням вподобань користувача» відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлений.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

- згоден на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-системи, а також на архівування моєї роботи в базі даних цієї системи.

Дата 14.06.2024 _____ Фролов Данніл Володимирович
(підпис) (прізвище та ініціали) (студент)

Дата 15.06.2024 _____ Полякова Наталія Петрівна
(підпис) (прізвище та ініціали) (керівник)