

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ЗАПОРІЗЬКА ПОЛІТЕХНІКА»

Факультет комп'ютерних наук та технологій
Кафедра комп'ютерних систем та мереж

Пояснювальна записка

до дипломного проекту (роботи)

бакалавра

(ступінь вищої освіти (освітній ступінь))

на тему РОЗРОБКА МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КІНОМАНІВ

Виконав: студент 4 курсу, групи КНТ-520
спеціальності _____

123 Комп'ютерна інженерія

Освітня програма (спеціалізація)

Комп'ютерна інженерія

(код і назва спеціальності)

ЧЕРВОНЮК Д.І.

(прізвище та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(прізвище та ініціали)

Рецензент СТЕПАНЕНКО О.О.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування вищого навчального закладу)

Факультет Комп'ютерних наук і технологій
Кафедра «Комп'ютерні системи та мережі»
Ступінь вищої освіти (освітній ступінь) бакалаврський
Спеціальність 123 Комп'ютерна інженерія
(код і назва)
Освітня програма (спеціалізація) Комп'ютерна інженерія
(назва)

ЗАТВЕРДЖУЮ

Зав. кафедри Кудерметов Р.К.

“ ” 2024 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТУ

ЧЕРВОНІЮК Дмитру Ігоревичу

(прізвище, ім'я, по батькові)

1. Тема проекту (роботи) Розробка мобільного застосунку для кіноманів

керівник проекту (роботи) СКРУПСЬКИЙ С.Ю., к.т.н., доцент

(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від “01” квітня 2024 року № 105

2. Строк подання студентом проекту (роботи) 01.06.2024 року

3. Вихідні дані до проекту (роботи) операційна система iOS, RAD, Xcode 10,
мова програмування Swift, Adobe Photoshop

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

1) Аналіз методів організації платформ для відпочинку;

2) Методи побудови сервісів;

3) Реалізація мультифункціональної платформи для планування відпочинку;

4) Розробка мобільного додатку;

5) Тестування платформи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

1) Популярні відеосервіси для відпочинку;

2) Популярні музичні сервіси;

3) Схема структури розробки додатку;

4) Діаграма послідовності для мобільного додатку.

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5	СКРУПСЬКИЙ С.Ю., к.т.н., доцент		
Нормоконтроль	ПОЛЬСЬКА О.В., ст. викладач		

7. Дата видачі завдання 01.04.2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз принципів організації платформ	17.04.2024	
2	Аналіз програмно-апаратних платформ	20.04.2024	
3	Методи побудови сервісів	23.04.2024	
4	Вибір операційної системи	28.04.2024	
5	Визначення методології проектування	03.05.2024	
6	Визначення вимог для реалізації розробки	09.05.2024	
7	Розробка мобільного додатку	19.05.2024	
8	Тестування роботи системи	25.05.2024	
9	Оформлення пояснювальної записки	29.05.2024	

Студент  Дмитро ЧЕРВОНІЮК
(підпис) (ініціали та прізвище)

Керівник проекту (роботи)  Степан СКРУПСЬКИЙ
(підпис) (ініціали та прізвище)

РЕФЕРАТ

ПЗ: 92 с., 47 рис., 2 табл., 22 джерел, 9 додатків.

ТОП, МУЛЬТІМЕДІА, СЕРВІС, РАНЖУВАННЯ, ПЛАТФОРМА, XCODE, APPLE APP STORE, IOS

Мета роботи полягає розробці мобільного застосунку для планування відпочинку.

Для реалізації даної роботи використовувались методи аналізу, збору інформації, обробки і аналізу даних, порівняння, аналогії, методи програмування.

По результатам виконання було розроблено мобільний застосунок для пошуку та вибору фільмів, а також найближчих кінотеатрів. Проаналізовано особливості розробки мобільних застосунків, здійснено порівняння з існуючими застосунками. Деталізовано функціональність системи за допомогою різних діаграм. Було створено та протестовано робочу версію системи, яка створює користувацький список, показує ТОП фільмів з сайту IMDb та відображає на карті найближчі до користувача кінотеатри.

ЗМІСТ

Вступ.....	8
1 Аналіз методів організації платформ для відпочинку.....	10
1.1 Аналіз принципів організації платформ для планування відпочинку.....	10
1.2 Мультимедіа і зберігання інформації.....	12
1.3 Аналіз програмно-апаратних платформ	15
1.4 Сервіси для планування відпочинку	18
2 Методи побудови сервісів	21
2.1 Ранжування контенту.....	21
2.1.1 Ранжування фільмів	22
2.1.2 Ранжування контенту на YouTube.....	23
2.2 Метод побудови персональних рекомендацій користувача	24
2.3 Метод знаходження найближчих до користувача локацій відпочинку	25
2.4 Вибір операційної системи.....	26
3 Реалізація мультифункціональної платформи для планування відпочинку	28
3.1 Вимоги до платформи.....	28
3.2 Методологія проектування.....	29
3.3 Проектування архітектури.....	31
3.4 Алгоритми роботи модулів	34
3.5 Моделювання системи в стандарті UML.....	34
3.6 Середовище розробки застосунків Xcode.....	38
3.7 Мова програмування Swift	40
3.8 Графічний редактор Adobe Photoshop CC	42
4 Розробка мобільного додатку	44
4.1 Створення проекту в середовищі Xcode	44
4.2 Опис коду додатку	45
4.3 Додання власних міток на карті за допомогою MapKit	48
5 Тестування платформи	51

5.1 Встановлення програми для проведення тестування	52
5.2 Публікація застосунку в Apple App Store	52
5.3 Додання інформації до «ТОП»	62
Висновки	65
Перелік джерел посилання	66
Додаток А. Інтерфейс користувача	69
Додаток Б. Відображення елементів у списку фільмів	76
Додаток В. Відображення тексту з перекресленим форматуванням	80
Додаток Г. Налаштування вигляду елементів інтерфейсу	81
Додаток Д. Відображення детальної інформації про об'єкти	86
Додаток Ж. Налаштування навігації у мобільному додатку	88
Додаток К. Встановлення розміру, колірів, тіней та інших атрибутів кнопки	91
Додаток Л. Управління мережевою взаємодією додатку	92
Додаток М. Інтерфейс створеного додатку	93
Графічна частина:	
Пл.1 - Популярні відеосервіси для відпочинку	
Пл.2 - Популярні музичні сервіси	
Пл.3 - Схема структури розробки додатку	
Пл.4 - Діаграма послідовності для мобільного додатку	

ПЕРЕЛІК ОСНОВНИХ ПОЗНАЧЕНЬ І СКОРОЧЕНЬ

ООП	об'єктно-орієнтоване програмування
IDE	інтегроване середовище розробки
OLE	зв'язування та впровадження об'єктів
VCL	візуальна бібліотека компонентів;
NET	програмна технологія для створення, як звичайних програм, так і веб-застосунків
XML	стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками
SQL	мова програмування для взаємодії користувача з базами
MSDE	файл-серверна система управління версіями

ВСТУП

На сучасний день, сучасні мобільні пристрої, які поширюються з кожним днем, активно використовуються у всіх сферах людської діяльності для обробки зростаючого обсягу інформації. Організація відпочинку з друзями потребує обробки інформації, і створення різних інструментів для спрощення цього процесу завжди було популярним серед користувачів. Сучасні мобільні пристрої, які завжди під рукою, проникли в усі сфери життя, ставши незамінними помічниками. Сфери застосування мобільних пристроїв постійно розширюються, включаючи ті, де раніше не відчувалася потреба в обчисленнях.

Однією з головних функцій сучасних смартфонів є спрощення задач. Значення швидкого вирішення завдань, що може дійсно зекономити час та ресурси користувачів, стає все більшим. Метою цієї роботи є розробка мобільного застосунку для планування відпочинку.

Для досягнення цієї мети передбачені такі завдання:

- аналіз методів організації платформ для планування відпочинку;
- аналіз програмно-апаратних платформ;
- методи та вимоги побудови сервісів для створення платформ;
- розробка алгоритмів роботи модулів;
- розробка мультифункціональної платформи для планування відпочинку.

Розробка зручного для користувача програмного продукту, що відповідає специфікаціям, на мові високого рівня, що надає простий та зручний сервіс для справжніх кіноманів. Цей продукт буде містити короткі описи фільмів для швидкого орієнтування в виборі та може бути використаний на вулиці, вдома, в кафе чи в транспорті.

В ході роботи було використано інтегроване середовище візуального програмування Xcode, мова програмування Swift 4.0, робота з Apple MapKit та IMDb API, графічний редактор Adobe Photoshop.

Одержані результати показують, що розроблену систему можна буде

використовуватись у планування відпочинку для користувачів, зручний та зрозумілий інтерфейс та доступність безкоштовних оновлень для користувачів мобільних пристроїв.

1 АНАЛІЗ МЕТОДІВ ОРГАНІЗАЦІЇ ПЛАТФОРМ ДЛЯ ВІДПОЧИНКУ

1.1 Аналіз принципів організації платформ для планування відпочинку

Сучасну людину без мобільного пристрою в руках важко уявити. Чи то телефон, смартфон, чи інші комунікатори - ці пристрої стали невід'ємною частиною життя і, здається, залишаться такими на довгий час. З'явлення нових мобільних пристроїв та різноманітність платформ, таких як Android, Symbian, Bada, iOS або iPhone, активно стимулює розвиток бізнесу з розробки мобільних додатків різного призначення.

Мобільний додаток - це програма, що встановлюється на конкретній платформі та має свій функціонал для виконання різних дій. Ці програми розробляються на мовах програмування високого рівня та компілюються у код пристрою. Мобільні додатки зайняли свою нішу в різних сферах людської діяльності і зараз використовуються не тільки для розваг, як це було раніше, але й для бізнесу та проведення рекламних кампаній. Сучасні технології дозволяють максимально адаптувати програми під різні мобільні пристрої і зробити їх легкими для сприйняття користувачем [1].

Мобільні додатки, які позиціонують себе як рішення для споживачів, розробляються з урахуванням їхніх потреб. Оскільки потреби та вимоги користувачів різні, не існує універсальних додатків. Загалом можна виділити декілька типів мобільних додатків в залежності від їхньої спрямованості та функцій:

- програми-служби: це додатки, які можуть представляти собою каталоги, списки тощо, відображаючи діяльність певної організації;
- програми-події: створені для трансляції різноманітних подій, наприклад, онлайн-трансляція футбольних матчів;
- промо-додатки: обмежені за функціональністю, але дуже креативні й популярні, часто використовуються як частина рекламних кампаній;

- програми-ігри: призначені для розваг та навчання дітей, розроблені для різних платформ;
- мобільні додатки для бізнесу: як для внутрішньої комунікації та оптимізації роботи організації, так і для взаємодії з користувачами;
- інтернет-магазини: для здійснення онлайн-покупок різних товарів;
- інші види мобільних додатків: соціальні мережі, системні програми тощо.

Чітка класифікація мобільних додатків неможлива, оскільки деякі з них об'єднують у собі різноманітні функції, утворюючи свого роду "мікс". Зазвичай ці додатки, які пропонують розширені можливості, є платними, тоді як безкоштовні додатки зазвичай мають обмежений функціонал, наприклад, для перегляду електронної пошти. Хоча існують винятки.

Сучасні мобільні додатки вже давно стали потужним інструментом маркетингу, який вирішує безліч завдань: формування іміджу, підтримка бренду та збільшення лояльності споживачів, оптимізація комунікаційних процесів та створення інформаційного простору.

У сучасному світі існує багато платформ і сервісів для організації дозвілля. Онлайн-кінотеатри та стрімінгові сервіси, такі як YouTube, Spotify, Apple Music, Megogo, Netflix, YouTube Music, SoundCloud та інші, є серед них. Деякі з них пропонують безкоштовний доступ, але більшість забезпечує лише обмежений функціонал і надає можливість користувачам розширити його за допомогою платної підписки.

Також деякі сервіси дозволяють користувачам спільно використовувати власний контент (SoundCloud, YouTube, Spotify, Apple Music), тоді як інші пропонують лише конкретний набір ліцензованого контенту (Netflix, Megogo, Hulu).

Платформи для планування відпочинку може включати наступні аспекти:

- зручність і доступність інтерфейсу - платформа повинна мати зрозумілий та легко доступний інтерфейс, що дозволяє користувачам швидко та ефективно планувати свій відпочинок. Це може включати можливість швидкого пошуку, фільтрації та вибору різноманітних варіантів відпочинку;

- різноманітність пропозицій - платформа повинна пропонувати різноманітні варіанти відпочинку, що відповідають потребам та бажанням різних користувачів. Це може включати різні типи подорожей (екскурсії, круїзи, пляжний відпочинок), а також різні види розваг та активного відпочинку;

- персоналізація - платформа може надавати можливість персоналізації планування відпочинку відповідно до індивідуальних потреб та вподобань користувача. Наприклад, система рекомендацій може пропонувати варіанти відпочинку, які відповідають історії пошуку та попереднім виборам користувача;

- інтеграція з іншими сервісами - платформа може інтегруватися з іншими сервісами, такими як системи бронювання готелів, авіаквитків, транспорту тощо, для забезпечення повного і комплексного планування подорожі;

- відгуки і рейтинги - наявність відгуків та рейтингів від інших користувачів може допомогти прийняти рішення щодо вибору місця відпочинку. Платформа може надавати можливість користувачам залишати свої відгуки та оцінювати послуги, що надаються;

- підтримка клієнтів - наявність служби підтримки клієнтів, яка може вирішувати будь-які питання та проблеми користувачів під час планування та під час подорожі, є важливим аспектом організації платформи для планування відпочинку.

Аналіз цих принципів допоможе побудувати ефективну та зручну платформу для користувачів, яка відповідає їхнім потребам та очікуванням щодо планування відпочинку.

1.2 Мультимедіа і зберігання інформації

Потокове мультимедіа - це формат мультимедійного вмісту, який передається через мережу Інтернет у режимі реального часу. У цьому форматі аудіо, відео або інші мультимедійні дані не потрібно завантажувати повністю на

пристрій користувача перед відтворенням. Замість цього, дані передаються по мережі порціями, а пристрій відтворення може починати відтворення вмісту майже одразу після початку завантаження. Потокове мультимедіа дозволяє користувачам споживати вміст в режимі реального часу без необхідності чекати на завершення завантаження. Такі послуги включають стрімінгове відео (наприклад, YouTube, Netflix), аудіострімінг (наприклад, Spotify, Apple Music) та інші мультимедійні сервіси, які надають змогу відтворювати контент у режимі реального часу через Інтернет [2].

Потокове мультимедіа поділяють на два типи:

- за запитом;
- живе.

Мультимедіа за запитом - це формат мультимедійного вмісту, який користувач може отримати або відтворити за власним запитом чи побажанням. Цей тип мультимедіа може включати в себе різноманітні файли, такі як відео, аудіо, фотографії, анімацію тощо, які можуть бути збережені на пристрої користувача або отримані через мережу Інтернет за допомогою різних сервісів і платформ.

Мультимедійний контент за запитом може включати в себе велику різноманітність вмісту, наприклад:

- відео за запитом: фільми, телепередачі, короткі відеоролики тощо, які можуть бути переглянуті користувачем за власним вибором;
- аудіо за запитом: музика, подкасти, аудіокниги, радіопередачі тощо, які можуть бути відтворені або завантажені користувачем відповідно до їхніх вподобань;
- фотографії та зображення: колекції фотографій, ілюстрації, меми, артистика тощо, які можуть бути переглянуті або завантажені з різних джерел;
- інші форми мультимедійного контенту: анімація, ігри, віртуальна реальність тощо, які можуть бути доступні користувачам за їхнім власним запитом.

Отже, мультимедіа за запитом дозволяє користувачам отримувати доступ

до різноманітного вмісту, який вони шукають або в якому зацікавлені у конкретний момент часу і може зберігатися на серверах тривалий період часу (YouTube, Spotify, Megogo, Netflix).

Живе мультимедіа - це формат мультимедійного вмісту, який надається в реальному часі, зазвичай через пряму трансляцію або живий стрім. У цьому форматі користувачі можуть споживати вміст у реальному часі, отримуючи доступ до подій, виступів, концертів, подій в мережі, демонстрацій, вебінарів та інших подій, що відбуваються в даний момент.

Живе мультимедіа може бути представлене у різних форматах, включаючи відео, аудіо, тексти та зображення, і надає користувачам можливість бути в центрі подій без необхідності фізичного присутності. Цей формат надає можливість спілкуватися в реальному часі, взаємодіяти з іншими учасниками або взяти участь у дискусіях під час трансляції.

Прикладами живого мультимедіа можуть бути онлайн-трансляції спортивних подій, конференцій, музичних виступів, вебінарів, стрімів на платформах для геймінгу та інші онлайн-події, які відбуваються у реальному часі і доступні для перегляду аудиторії в цей самий момент. (Twitch, YouTube, Mixer та інші).

Розмір для зберігання інформації потокових мультимедіа може значно варіюватися в залежності від кількості контенту, який включається в потік, та якості відео або аудіо.

Для відео потоків, розмір файлу зазвичай залежить від роздільної здатності (наприклад, HD, Full HD, 4K), бітрейту (кількості бітів на секунду) та тривалості відео. Наприклад, відео потік стандартної якості може мати розмір приблизно 1-2 мегабайти на хвилину, в той час як відео високої якості може потребувати від 10 до 20 мегабайтів на хвилину або більше.

Для аудіо потоків, розмір файлу також залежить від бітрейту та тривалості аудіо. Зазвичай, аудіо потік стандартної якості (наприклад, MP3 з бітрейтом 128 kbps) може займати приблизно 1 мегабайт на хвилину, але високоякісні аудіоформати, такі як FLAC або WAV, можуть мати більші розміри

файлів.

Однак, важливо зауважити, що у випадку потокових мультимедіа дані не зберігаються локально на пристрої користувача, а передаються через мережу в реальному часі. Таким чином, для перегляду або прослуховування потокового вмісту не потрібно значного місця для зберігання на пристрої.

1.3 Аналіз програмно-апаратних платформ

Сьогодні кожна компанія або фірма, яка йде в ногу з часом, розуміє, що створити хороший сайт або надрукувати якісну поліграфію – недостатньо. У час, коли більшість людей мають під рукою свій портативний мікросвіт в смартфоні, єдине рішення для тих, хто хоче бути серед лідерів – створити якісні та зручні мобільні додатки.

Розробка мобільних додатків складний і багатоетапний процес, тому, починаючи його, слід визначити потреби своїх користувачів і функції, що додаток повинен виконувати, а також визначитися з його типом і мобільною платформою [3].

Розробка мобільних додатків може складатися з наступних етапів та компонентів:

- **аналіз вимог:** цей етап включає збір та аналіз вимог користувачів та бізнесу, визначення функціональних та нефункціональних вимог до додатка;
- **розробка макету додатку:** цьому етапу варто приділити особливу увагу, адже саме від технічного завдання залежить кінцевий результат. Не приділивши увагу навіть незначній деталі і не заклавши її в архітектуру програми, можна зіткнутися з необхідністю переробляти все практично з нуля (додаток А);
- **проектування інтерфейсу користувача (UI/UX):** розробка дизайну та інтерфейсу користувача, який буде зручним та привабливим для цільової аудиторії;

- розробка функціональності: створення функціональності додатка відповідно до вимог, включаючи роботу з базами даних, обробку даних, реалізацію функцій тощо;

- дизайн мобільних додатків: відштовхуючись від цілей мобільного додатку, його аудиторії та функціоналу, створюється відповідний дизайн. На цьому етапі дизайн-студія детально опрацьовує кожен з екранів і кожен його деталь. Для початку розробляється дизайн перших трьох сторінок, який є своєрідним фундаментом для всіх інших. За бажанням користувача, можна створити кілька варіантів дизайну (оцінивши які, ви можете вибрати саме той, який найбільше припаде до душі), тільки після затвердження дизайну промальовуються інші екрани, кнопки та іконки;

- розробка програми: розробка мобільних додатків на цьому етапі полягає в тому, що з'єднуються воедино всі промальовані і затверджені елементи – екрани, кнопки, іконки, підказки. Фактично, зі статичної картинки створюється інтерактивна, рухлива модель. Крім того, з'єднується серверна і користувацька частини програми, щоб вона повноцінно працювала;

- тестування: проведення різних видів тестування, включаючи модульне тестування, інтеграційне тестування, тестування продуктивності та тестування на користувацькому досвіді (UX). Кожен мобільний додаток, унікальний, а значить, проектуючи його важко передбачити всі нюанси його живої роботи. Для цього протягом декількох днів формується таблиця з помилками. За бажанням, користувач теж може отримати тестову версію програми, щоб на власному досвіді переконатися, як вона працює. Адже кінцевий споживач повинен отримати бездоганний продукт;

- оптимізація та вдосконалення: проведення оптимізації додатка для підвищення продуктивності, зменшення споживання ресурсів та вдосконалення користувацького досвіду;

- реліз та підтримка: після завершення розробки та тестування додаток готовий до релізу в магазинах додатків (App Store для iOS, Google Play для Android). Після релізу може відбуватися підтримка, оновлення та вдосконалення

додатка. Для того, щоб користувачі могли знайти мобільний додаток і скористатися ним, його розміщують у спеціальних магазинах. Наприклад, GooglePlay для ОС Android, або AppStore для iOS;

- маркетинг: просування додатку серед цільової аудиторії, включаючи стратегії маркетингу, реклами та взаємодію з користувачами.

Кожен із цих етапів важливий для успішного створення та впровадження мобільного додатка.

За статистикою, Google Play, де розміщені програми під операційні системи на базі Android, лідирує за кількістю скачувань. Він також продовжує набирати популярність на ринку мобільних додатків.

Більшість сучасних платформ орієнтовані на постійний взаємозв'язок із користувачем. Тому розробники віддають перевагу мобільним додаткам, які краще пристосовані до різного обладнання, ніж звичайні веб-сайти. Однак в розробці цих додатків є певні складнощі, пов'язані з необхідністю враховувати різноманітні апаратні платформи користувачів.

Крім цього, можуть використовуватися веб-додатки - це клієнт-серверні додатки, у яких клієнт взаємодіє з веб-сервером через браузер. У таких додатках логіка розподіляється між сервером і клієнтом, а дані зберігаються в основному на сервері, з обміном інформації через мережу. Однією з переваг цього підходу є те, що клієнти не залежать від конкретної операційної системи користувача, тому веб-додатки є універсальними.

Великі платформи, розробляючи додатки, прагнуть зробити їх доступними якомога більшій кількості користувачів, тому системні вимоги намагаються зробити якомога меншими.

Наприклад, для перегляду відео через браузер на YouTube вам знадобиться остання версія браузера Chrome, Opera, Firefox, Safari або Edge, інтернет-з'єднання зі швидкістю не менше 500 Кбіт/с та операційна система Windows 7, MacOS 10.7, Ubuntu 10 або їх новіші версії.

1.4 Сервіси для планування відпочинку

Сучасні стрімінгові сервіси дозволяють переглядати широкий спектр контенту: від прямих трансляцій новин до останніх серіалів та класичних фільмів у будь-який зручний для користувача час і з будь-якого пристрою. Користувач придбавши підписку на сервіс, отримує можливість перегляду будь-якого контенту на телевізорі, комп'ютері, смартфоні та інших сумісних пристроях [4].

Найпопулярніші відео-сервіси:

- YouTube - найбільший і найпопулярніший відеохостинг, який містить величезну кількість відео з різних тематик, від розважального контенту до навчальних відео та музичних кліпів;

- Netflix - популярний стрімінговий сервіс, який надає доступ до широкого асортименту фільмів, серіалів і документальних фільмів, включаючи оригінальний контент;

- Amazon Prime Video - стрімінговий сервіс, який пропонує широкий вибір відеоконтенту, включаючи фільми, серіали та оригінальний контент;

- Disney - сервіс відео-стрімінгу, який пропонує широкий вибір фільмів і серіалів від Disney, Pixar, Marvel, Star Wars та National Geographic;

- Hulu - американський сервіс відео-стрімінгу, який пропонує широкий вибір фільмів, серіалів та оригінального контенту, а також можливість перегляду телевізійних шоу та прямих ефірів;

- YouTube TV - платна підписка, яка надає доступ до телевізійних каналів, прямих ефірів та DVR-функції через платформу YouTube;

- Vimeo - відеохостинг, який призначений переважно для професіональних відеомейкерів та кінематографістів, а також для творчої спільноти, яка шукає якісний відеоконтент.

Ці сервіси надають різноманітний відеоконтент і відповідають на потреби різних категорій глядачів.

Вибір сервісу залежить від того, що конкретно користувач бажає дивитись.

Сучасні користувачі все рідше купують окремі диски або завантажують цілі альбоми музики. Вони віддають перевагу підписці на онлайн-стрімінговий сервіс, що дає змогу слухати будь-яку музику у будь-який час [5].

Ось деякі з найпопулярніших музичних сервісів:

- Spotify -цей сервіс надає доступ до величезної бібліотеки музики з усього світу, де користувачі можуть слухати улюблені пісні, створювати власні плейлисти та відкривати нові виконавці;

- Apple Music - розроблений компанією Apple, цей сервіс також пропонує широкий вибір музики, включаючи оригінальний контент та радіостанції;

- Amazon Music - цей сервіс надає доступ до мільйонів пісень для підписників Amazon Prime, а також пропонує різноманітні плейлисти та радіостанції;

- YouTube Music - розроблений компанією Google, цей сервіс дозволяє слухати музику з YouTube, а також використовувати функції, спеціально розроблені для музичного сприятливого середовища;

- Tidal - цей сервіс відомий своєю високою якістю звуку та ексклюзивним контентом від відомих виконавців;

- Deezer – цей сервіс пропонує доступ до мільйонів треків, плейлисти, радіостанції та рекомендації на основі музичних вподобань користувачів;

- SoundCloud - цей сервіс відомий своїм спрямуванням на нових та незалежних виконавців, які можуть розміщувати свою музику та взаємодіяти з фанатами.

Ці сервіси дозволяють користувачам насолоджуватися музикою в будь-який час та зручно з будь-якого пристрою.

Усі ці платформи можуть автоматично створювати плейлисти, враховуючи музичні вподобання користувача, або ж вибирати існуючі варіанти. Зазвичай їх можна синхронізувати з мобільним додатком, який працює з розумними годинниками, навушниками та домашніми аудіосистемами.

Більшість з них пропонують безкоштовний пробний період, після якого необхідно оформити повну підписку або обмежують функціонал для неоплачених

користувачів, які хочуть продовжувати користуватися сервісом.

Цінова політика сервісів різниться: деякі пропонують різні види підписок, такі як індивідуальні, сімейні, для пар чи студентів. Інші визначають різні рівні доступу в залежності від ціни, пропонуючи вищу цінову категорію за контент в кращій якості [6].

Ціни підписки на місяць у відео-сервісів можуть варіюватися залежно від різних факторів, таких як країна проживання, тип підписки та доступний контент, таблиця 1.1.

Таблиця 1.1 – Ціни підписки на відео сервіси за місяць

Netflix	YouTube Premium	Disney+	Amazon Prime	Hulu
\$8.99 - \$17.99	\$11.99	\$7.99	\$12.99	\$5.99 - \$11.99

Кожен з цих сервісів пропонує свій унікальний контент, тому обрати один конкретний може бути дуже важко для кінцевого користувача.

Ось орієнтовані ціни на місячну підписку у деяких з найпопулярніших музичних сервісів, таблиця 1.2).

Таблиця 1.2 – Ціни підписки на музичні сервіси на місяць

Spotify Premium	Apple Music	YouTube Music	Amazon Music Unlimited	Tidal Premium
9.99\$	9.99\$	9.99\$	9.99\$	- 9.99\$

Знову ж таки, ціни можуть змінюватися залежно від регіону і можуть бути доступні різні пакети підписки, такі як сімейні або студентські плани, які можуть бути дешевшими.

Стрімінгові послуги стають все більш популярними серед молоді, яка шукає способи планування та організації відпочинку. З розвитком інтернет-технологій потокові медіа сервіси стали справжньою альтернативою телебаченню та виходу в кінотеатр, змінивши підхід до придбання контенту на цифрових та фізичних носіях.

Конкуренція серед стрімінгових сервісів спричинила широкий вибір для користувача. Тепер вони відрізняються не тільки за ціною, але й за якістю та різноманіттям контенту, який вони пропонують. Деякі сервіси надають безкоштовний доступ до всієї бібліотеки з рекламними вставками, але платна підписка прибирає рекламу, дозволяє кешувати контент для перегляду без інтернет-підключення та пропонує вищу якість аудіо або відео.

Ці сервіси працюють майже на всіх пристроях з доступом до мережі Інтернет і не вимагають великої апаратної потужності. Більшість з них можуть працювати у веб-браузері, але не всі функції можуть бути доступні, тому рекомендується використовувати спеціальний додаток для цього.

2 МЕТОДИ ПОБУДОВИ СЕРВІСІВ

2.1 Ранжування контенту

Ранжування в пошукових системах - це процес, за допомогою якого контент впорядковується та представляється користувачу під час його пошукового запиту. Цей процес ґрунтується на різних критеріях, таких як оцінка якості, відповідність, актуальність, дата створення тощо. За допомогою цих критеріїв пошукова система формує список результатів. Алгоритми ранжування піддаються постійній модифікації для оптимізації функціонування сервісу[7].

Основним завданням алгоритмів ранжування є надання користувачеві конкретних відповідей на його запити та створення оптимальної послідовності результатів пошуку. Ці алгоритми використовують різні інструменти на кожному етапі пошуку для забезпечення його ефективності.

Алгоритми ранжування постійно змінюються відповідно до еволюції Інтернету. Редакторам контенту достатньо вказати тематику, опис та ключові слова у метатегах для того, щоб пошукова система вважала їх відповідними тематиці і високо їх ранжувала. Проте сервіси постійно вдосконалюють свої

алгоритми для урахування різних факторів, що впливають на ранжування контенту.

2.1.1 Ранжування фільмів

Ранжування фільмів - це процес впорядкування та визначення популярності фільмів на основі різних критеріїв. Цей процес може включати оцінки глядачів, критиків, касові збори, нагороди, рейтинги на платформах, таких як IMDb або Rotten Tomatoes, а також інші фактори, такі як жанр, режисер, акторський склад тощо. Ранжування фільмів може допомогти глядачам знаходити якісні та цікаві фільми великого або малого бюджету, що відповідають їхнім вподобанням і інтересам [8].

Наразі фільм, що виходить в кінопрокат, може бути оцінений однією з п'яти категорій рейтингів.

Рейтинг G (General audiences) означає, що фільм, який виходить в прокат з рейтингом G, зазвичай призначений для перегляду аудиторією будь-якого віку. Цей рейтинг означає, що матеріал фільму не містить нічого, що може бути шкідливим для дітей, і він підходить для всіх вікових категорій. Фільми з таким рейтингом зазвичай не мають насильства, сексуального контенту, грубої мови або іншого матеріалу, який може бути несприйнятливим для молоді.

Рейтинг PG (Parental guidance suggested) означає, що дорослим рекомендується надавати дитині певну підтримку або вказівки щодо перегляду фільму.

Рейтинг PG-13 (Parents strongly cautioned) означає, що батькам настійно рекомендується бути обережними та ретельно обговорити з дітьми перегляд такого фільму, оскільки він може містити матеріал, який не підходить для дітей молодшого віку.

Рейтинг R (Restricted) означає, що фільм призначений для дорослої аудиторії. До перегляду такого фільму рекомендується особам, які досягли певного віку, і він може містити матеріал, що не підходить для неповнолітніх без супроводу дорослих.

Рейтинг NC-17 (No one 17 & Under Admitted) означає, що особам до 17

років вхід заборонено. Цей рейтинг позначає фільми з високим рівнем контенту, який може бути неприйнятним для дітей і підлітків, навіть з наявністю батьківського контролю.

Рейтинг NR (Not Rated) означає, що фільм не має офіційного рейтингу. Це може статися, коли фільм ще не був пройшов оцінку або коли він не відповідає стандартним критеріям для існуючих рейтингів. Фільми з таким рейтингом можуть бути представлені в різних форматах, і вони можуть включати матеріал, який відрізняється за своєю придатністю для різних аудиторій.

2.1.2 Ранжування контенту на YouTube

Google та YouTube використовують алгоритми для визначення контенту, який показувати у відповідь на пошукові запити користувачів. Ці алгоритми також збирають інформацію про користувачів, що означає, що результати пошуку та рейтинги адаптовані до кожної конкретної особи [9]..

Пошукові системи часто оновлюють свої алгоритми, але є фактори, за допомогою яких алгоритми оцінюють вміст відео та загалом контент сторінки.

Одним з найсуттєвіших критеріїв для класифікації відео є тривалість перегляду, тобто час, який глядачі проводять, переглядаючи контент. Високий показник тривалості перегляду вказує на те, що відео містить цінний контент, оскільки глядачі не припиняють перегляду через короткий проміжок часу.. А от короткий час перегляду означає, що вміст, швидше за все, не релевантний запиту та не повинен бути ранжованим.

На ранжування YouTube впливають наступні аспекти:

- релевантність: оцінюється, наскільки відео відповідає запиту користувача за змістом та описом;
- якість: алгоритми враховують експертність, авторитетність і надійність каналів, а також кількість підписників та переглядів;
- залученість: враховується активність користувачів у відношенні до контенту;
- персоналізація: контент адаптується до індивідуальних інтересів користувача, враховуючи його попередні запити та перегляди;

- авторитетність джерела: якщо доцільно, обирається контент із джерел, які мають високий рівень довіри. Це особливо стосується сфер, таких як новини, політика, музика та наукова інформація.

Алгоритм YouTube має 2 мети:

- надати користувачам цікавий контент (той, котрий релевантний їх запитам);
- заохотити людей переглядати більше відео, які, ймовірно, їм сподобаються.

2.2 Метод побудови персональних рекомендацій користувача

При формуванні рекомендацій для користувача використовується інформація про його реакції на попередні відео. На основі цих даних формуються та калібруються контентні вподобання глядача, які допомагають у майбутньому підібрати для його інтерфейсу більш точні рекомендації. Метрики взаємодії (Interactions) позитивно впливають на поширення майбутніх відео, тематика яких близька до тієї, з якою у користувача була взаємодія [10]. Такий контент частіше з'являтиметься у рекомендаціях тих глядачів, які вже дивилися контент з цього каналу раніше (або каналу, близькому за тематикою), це позитивно впливає на процес формування постійної аудиторії, утримання глядачів, але не має прямого ефекту на пошук нових.

При формулюванні особистих рекомендацій важливо враховувати, що більшість користувачів не оцінюють кожен фільм, який дивилися, а лише деяку частину. Також не всі фільми, які вони віддають перевагу, отримують високі оцінки. Зазвичай найвищі бали отримують рідкісні шедеври або класика, але це не заважає людям насолоджуватися звичайними комедіями, бойовиками або мелодрамами. Це має значний вплив на рекомендації, які користувач отримує в результаті. Тому сервіс має намір рекомендувати такі фільми, які будуть

логічними та прийнятними для користувача, але вони, можливо, не будуть переглядатися.

Тому великі сервіси вкладають кошти в розробку нових алгоритмів для створення персональних рекомендацій, які збалансують обидва підходи та врахують потенціал перегляду та сприйняття рекомендацій користувачем. Ранжування всієї бази фільмів неефективно, тому спочатку відбирається декілька сотень потенційних кандидатів, які можуть бути цікавими глядачам. Сюди включаються як популярні фільми, так і ті, що відповідають певним критеріям користувача. Після цього визначаються фактори, пов'язані з фільмами, користувачами або парою користувач/фільм, і кандидати ранжуються за допомогою машинного навчання.

2.3 Метод знаходження найближчих до користувача локацій відпочинку

Зазвичай для визначення найближчих закладів використовується метод k -найближчих сусідів, що є простим метричним алгоритмом для автоматичної класифікації об'єктів. Основна ідея полягає в тому, що об'єкт отримує клас, який найбільш поширений серед його сусідів. Сусіди вибираються з множини об'єктів, класи яких вже відомі, і кожен об'єкт має фіксовану кількість атрибутів. За допомогою ключового параметра k визначається клас, який є найбільш представленим серед інших [11].

Google Maps, наприклад, надає можливість аналізу трафіку в реальному часі, панорамного перегляду вулиць та прокладання маршрутів для різних видів транспорту. Користувачі можуть шукати та додавати на карту різні об'єкти, які можуть бути пропущені розробниками, такі як парки, вулички тощо. Підприємці мають можливість додавати свої бізнеси на карту, включаючи контактну інформацію та графік роботи. Цей підхід дуже зручний для користувачів, які

швидко можуть отримати список найближчих закладів, переглянути їх фотографії та відгуки інших клієнтів за кілька хвилин. Заклади ранжуються за рейтингом, відгуками та повнотою інформації про них, що може суттєво вплинути на кількість їхніх відвідувачів

2.4 Вибір операційної системи

Операційна система, що працює на комп'ютері, є ключовим програмним забезпеченням, яке керує пам'яттю, процесами та усім програмним та апаратним забезпеченням.

Можна розглядати ОС як зв'язок між комп'ютером і користувачем. На ринку комп'ютерів найпопулярніші три операційні системи: Microsoft Windows 10, Apple macOS та Linux. На мобільних пристроях лідирують Apple iOS, Google Android та Microsoft Windows 10.

Apple macOS та iOS можуть бути встановленими як на комп'ютери так і на мобільні пристрої. Linux, Android та Windows 10 є безкоштовними операційними системами [12].

Мобільні операційні системи здатні виконувати більшість основних завдань, таких як перегляд веб-сторінок, фільмів, запуск застосунків тощо. Кожна операційна система має свій унікальний графічний інтерфейс та функціонал, тому перехід від однієї до іншої може здатися складним і незвичним.

Більшість користувачів використовують операційну систему, що вже встановлена на їхньому пристрої, але вони можуть змінити її за бажанням. Завдяки популярності мобільних пристроїв конкуренція на ринку програмного та апаратного забезпечення зростає, а боротьба між операційними системами стає все активнішою.

Операційні системи Android або iOS, мають свої переваги та недоліки. Це залежить від індивідуальних вподобань користувача (рис.2.1, 2.2).

 VS 		
iOS		Android
<i>Високоякісні додатки і магазин додатків</i>	Якість додатків	<i>Різноманітність програм</i>
<i>Виходять регулярно</i>	Швидке оновлення	
<i>Скоріше</i>	Кращі програми доступні першими	<i>Таніше</i>
95%	Дружній інтерфейс	5 %
<i>Значно надійніший</i>	Надійність	<i>півтора-два роки</i>
<i>Набагато захищеніша</i>	Велика безпека	<i>Загроза збільшуватиметься</i>
<i>Може використовувати в якості модему</i>	Continuity	
<i>Дорожчий</i>	Цінова тенденція	<i>Більш доступний</i>

Рисунок 2.1 - Порівняння iOS та Android



Рисунок 2.2 – Захист даних iOS та Android

Для розробки програмного продукту у дипломній роботі була обрана операційна система iOS. Ця ОС відзначається привабливим для користувачів інтерфейсом, високим рівнем безпеки, швидкістю та інтегрованою екосистемою Apple.

Проведено аналіз загальних принципів та методів ранжування відеоконтенту, включаючи фільми та відео на найпопулярніших платформах. Розглянуто методи створення персональних рекомендацій для користувачів на основі рейтингів.

Виконано порівняння операційних систем, які потенційно можуть бути використані для розробки та експлуатації додатку.

3 РЕАЛІЗАЦІЯ МУЛЬТИФУНКЦІОНАЛЬНОЇ ПЛАТФОРМИ ДЛЯ ПЛАНУВАННЯ ВІДПОЧИНКУ

3.1 Вимоги до платформи

Для виконання завдання дипломного проєкту необхідно побудувати структурну схему.

На рисунку представлено структурну схему розробки застосунку, яка розбивається на складові: створення списку, створення карти та створення списку ТОП. Кожна складова містить свої компоненти (рис.3.1).



Рисунок 3.1 - Схема компонентів застосунку

Наступним кроком буде відображення структури додатку на підставі побудованої вищенаведеної структурної схеми застосунку (рис.3.2). Ця структура направлена на основну складову – розробку додатку.



Рисунок 3.2 – Схема структури розробки додатку

Мета створення списку полягає у проведенні аналізу літературних джерел і розробці програмного модуля. Карта включає в себе аналіз літературних джерел щодо побудови карт та розробку самої карти. Реалізація ТОП включає написання програмного коду для отримання даних з сайту iMDB.com та їх відображення. Тестування передбачає перевірку основних модулів та встановлення програми на різних пристроях. Впровадження має на меті підтримку програмного забезпечення та регулярне оновлення з включенням нових функціональностей.

Структура додатку дозволяє розробити певний програмний продукт з певними функціональними можливостями.

3.2 Методологія проектування

Методологія проектування в ІТ сфері - це систематичний підхід до розробки програмного забезпечення або інформаційних систем, який включає в

себе набір принципів, методів, інструментів та практик для ефективного керування процесом розробки [13].

Методологія проєктування включає наступні етапи:

- планування - визначення цілей, термінів, бюджету та ресурсів проєкту;
- організація - формування команди, встановлення ролей і комунікаційних процесів;
- виконання - фактичне виконання завдань з використанням ресурсів;
- контроль і моніторинг - відстеження прогресу, управління ризиками та корекція плану;
- завершення - перевірка цілей і здача проєкту.

Існує безліч інших підходів, які можуть застосовуватися в управлінні проєктами залежно від конкретних вимог і характеру проєкту:

На основі проведеного аналізу для даного проєктування було обрано методологію RAD (Rapid Application Development) яка націлена на швидке створення і розгортання програмних додатків. Основний принцип цього метода - максимальна участь замовника і розробників у процесі. RAD підходить для проєктів, де можуть бути швидко оцінені та переглянуті користувачами, що дозволяє швидше реагувати на їх потреби та зміни.

Основні принципи дії методології RAD [14]:

- ітеративність - розробка програми відбувається через короткі ітерації, кожна з яких призводить до створення функціонального прототипу;
- рішучість - зосередження на швидкому прийнятті рішень та просуванні вперед, уникнення перебільшеної деталізації на ранніх етапах;
- тісне співробітництво з клієнтом - забезпечення постійного зворотного зв'язку з клієнтом, щоб забезпечити відповідність розробленого продукту їх потребам;
- прототипування - швидке створення прототипів або моделей продукту для визначення вимог та уточнення специфікацій;
- реюзабельність - використання готових компонентів або бібліотек, що дозволяє прискорити процес розробки;

- гнучкість - здатність змінювати вимоги та адаптуватися до нової інформації під час розробки.

Методологія RAD дозволяє швидше виходити на ринок з новими продуктами або функціональними оновленнями завдяки підвищеній швидкості розробки та впровадження. Однак вона може бути менш підходящою для складних або великих проєктів, де потрібно більше деталізованих аналізів і планування.

Саме тому дана методологія була обрана для здійснення проектування системи, розглянутої у даній роботі.

3.3 Проектування архітектури

Щоб представити взаємодію елементів моделі для реалізації виконання розробки мобільного застосунку буде використано діаграму послідовності.

Діаграма послідовності - це вид діаграми, що використовується в UML (Unified Modeling Language) для моделювання взаємодії між об'єктами в системі в певній послідовності часу. Вона відображає послідовність обміну повідомленнями між об'єктами або компонентами системи за часовою шкалою, що дозволяє зрозуміти порядок виконання різних операцій чи активностей. Діаграма послідовності допомагає візуалізувати взаємодію між об'єктами системи та виявляти потенційні проблеми в логіці програми [15].

Діаграма послідовності для мобільного застосунку, що проектується, зображена на рис. 3.3.

На цій діаграмі показано різні процеси або об'єкти, що функціонують одночасно, у вигляді вертикальних ліній. Відправлені повідомлення відображаються у вигляді горизонтальних ліній, відповідно до їх послідовного відправлення, а відповіді на повідомлення позначаються пунктирними лініями.

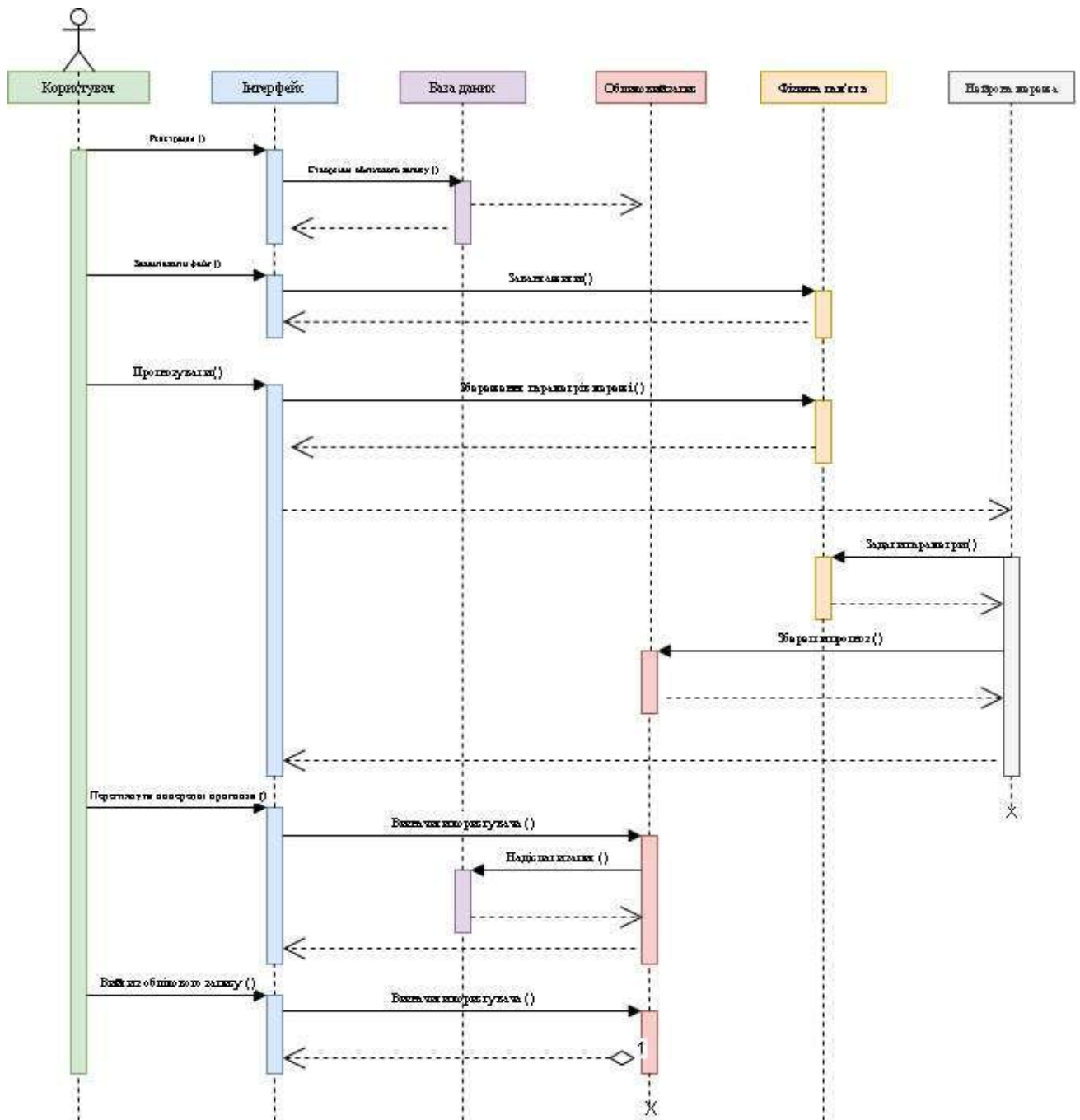


Рисунок 3.3 - Діаграма послідовності для мобільного додатку

При створенні облікового запису користувачем він взаємодіє з інтерфейсом, вибираючи опцію створення та заповнюючи реєстраційну форму. Після цього інтерфейс відправляє повідомлення про створення облікового запису. Якщо користувач створює новий об'єкт у списку, інтерфейс зберігає назву елемента у фізичній пам'яті та очікує відповіді. Тільки після цього інтерфейс ініціює процес створення. У відповідь на це буде створений новий елемент

списку, який інтерфейс відображає на екрані.

Для статичного опису системи, показуючи різні класи та їх взаємозв'язки зазвичай використовують діаграму класів [16].

На цих діаграмах відображаються класи, їх атрибути і методи, а також статичні зв'язки між ними. Вони допомагають формувати "словник предметної області" на концептуальному рівні і визначати структуру класів у програмній реалізації системи на рівні специфікації і реалізації. Діаграма класів є статичною, оскільки вона відображає лише статичні аспекти системи, не показуючи при цьому методів, які викликаються.

Для проєктованого мобільного застосунку цю діаграму можна зобразити наступним чином (рис3.4).



Рисунок 3.4 - Діаграма класів для мобільного застосунку

Функціонал інформаційної системи складається з трьох класів та двох інтерфейсів. Інтерфейси є абстрактними класами, що означає, що від них не можна безпосередньо створювати екземпляри. У них можуть бути операції, але не атрибути. У цьому випадку інтерфейси використовуються для входу в систему та зміни налаштувань додатку. Існуючі класи системи мають різноманітні атрибути та операції і взаємодіють між собою. Ці взаємодії показані у вигляді зв'язків асоціації з вказаною чисельністю, яка визначає кількість об'єктів на кожному кінці асоціації, які можуть бути пов'язані з об'єктами на іншому кінці. Об'єкт класу може бути пов'язаний з неозначеною кількістю об'єктів іншого класу. Також

важливо зазначити, що операції (або методи) у різних класах та інтерфейсах мають різний рівень обізнаності.

3.4 Алгоритми роботи модулів

Для здобуття адекватного рівня розуміння роботи програми необхідно розробити послідовності кроків, що визначатимуть, як саме вона буде функціонувати. Саме для цього було створено алгоритм дії застосунку (рис.3.5).

Після запуску застосунку користувач має можливість обрати одну з п'яти операцій із наданого переліку: додати фільм, видалити фільм, відмітити фільм, перейти до карти, перейти до списку ТОП.

При відкритті карти, користувач може як вибрати кінотеатр так і переглянути інформацію про нього та здійснювати переміщення по карті.

При відкритті списку «ТОП» користувач додатку має змогу ознайомитися з короткою інформацією про фільм, оцінку та рейтинг інших користувачів.

Після виконання описаних вище кроків, користувач може повернутися до виконання іншої дії або завершити роботу.

3.5 Моделювання системи в стандарті UML

Ще одним аспектом опису інформаційної системи є опис з точки зору користувача, який включає в себе випадки використання системи, функціонал, доступний для користувача з боку системи, а також опис окремих функціональних блоків системи, таких як класи, модулі або підсистеми. Для цього використовується мова моделювання UML [17].

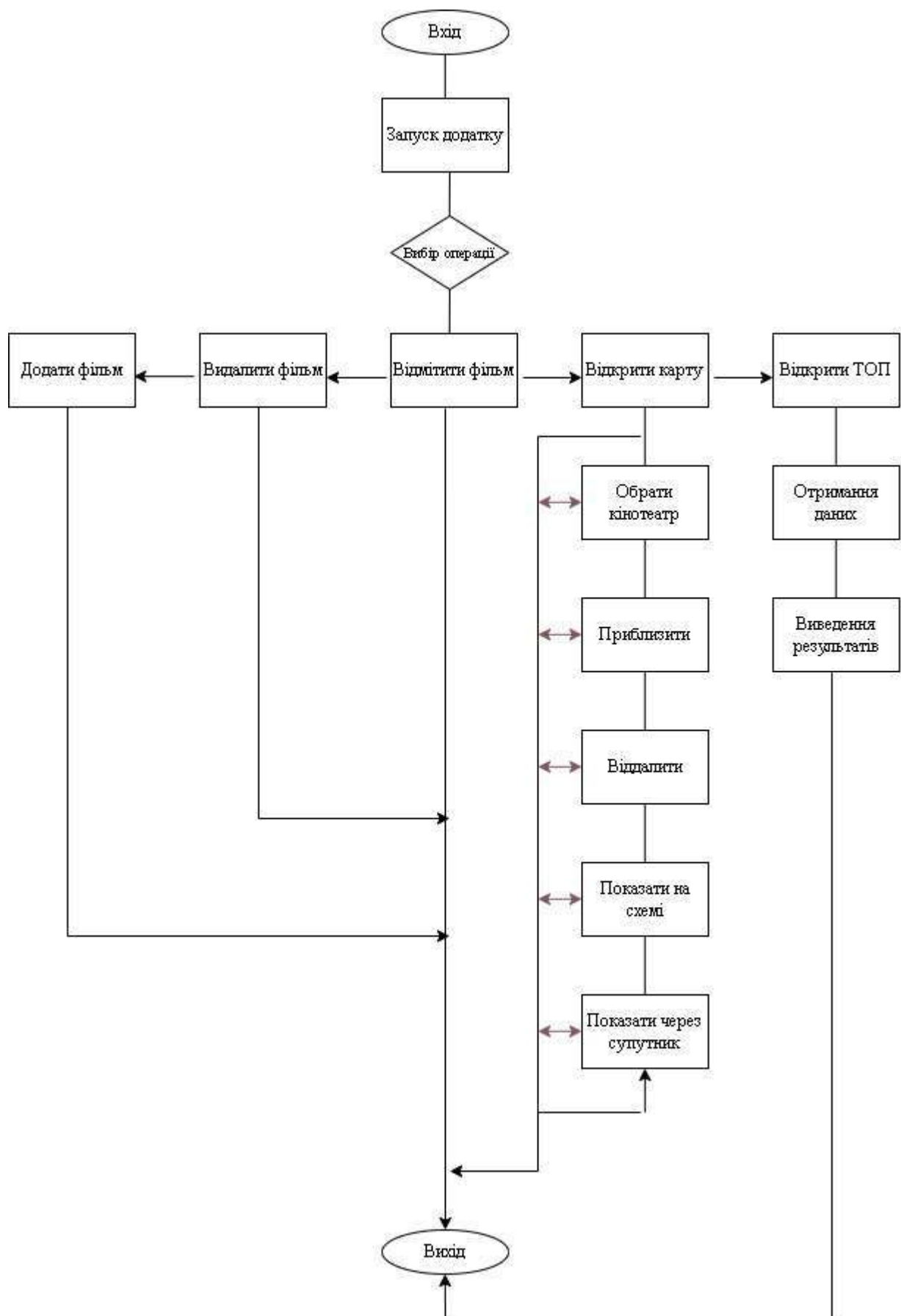


Рисунок 3.5 - Блок схема мобільного застосунку

Мова моделювання UML використовується для створення структурних і поведінкових моделей програмних систем. Вона дозволяє інженерам, розробникам програмного забезпечення та іншим учасникам процесу розробки спілкуватися і взаємодіяти за допомогою спільного графічного і текстового опису системи. Основні цілі використання UML полягають у наступному:

- стандартизація комунікації - UML надає стандартні нотації та правила для опису різних аспектів системи, це дозволяє розробникам спілкуватися між собою, якщо вони використовують різні технології чи працюють на різних етапах розробки;

- візуалізація системи - UML дозволяє візуалізувати структуру та поведінку системи за допомогою діаграм, це робить аналіз, розуміння та моделювання системи більш простим та зрозумілим;

- документування - UML може слугувати інструментом для створення документації, яка описує систему, це дозволяє зберігати та передавати інформацію про систему для майбутніх потреб;

- аналіз та проектування - UML дозволяє проводити аналіз вимог до системи та проектувати її архітектуру, він допомагає виявляти проблеми та визначати оптимальні рішення щодо розробки програмного забезпечення;

- підтримка різних етапів розробки - UML може бути використана на різних етапах життєвого циклу розробки програмного забезпечення, від аналізу вимог до тестування та розгортання системи.

Узагальнюючи, мова моделювання UML є важливим інструментом для розробки програмного забезпечення, який сприяє покращенню якості та продуктивності розробки, а також полегшує спілкування між учасниками проекту.

Діаграми прецедентів або діаграми варіантів використання - це графічні зображення, які використовуються в рамках мови моделювання UML для відображення взаємодії між акторами (користувачами) та системою. Ці діаграми дозволяють моделювати функціональність системи з точки зору користувача, виокремлюючи основні сценарії використання, які включають у себе послідовності подій та дій, що відбуваються в системі. Діаграми прецедентів

допомагають у розумінні та документуванні вимог до системи, а також визначають область функціональності, яка потребує реалізації.

Діаграма прецедентів для даної системи зображена на рис. 3.6.



Рисунок 3.6 - Діаграма прецедентів

На діаграмі представлено весь функціонал системи, який користувач може використовувати в певній послідовності. Відношення розширення (extend) використовуються для моделювання таких частин прецеденту, які можуть бути сприйняті користувачем як необов'язкова поведінка системи. Це дозволяє відокремити обов'язкову і необов'язкову поведінку. На діаграмі прецедентів відношення розширення позначаються у вигляді залежності зі стереотипом extend. Пунктирна стрілка спрямована до базового прецеденту (прецеденту, який розширюється) від абстрактного (того, що розширює).

3.6 Середовище розробки застосунків Xcode

Для втілення в життя розробки мобільного застосунку потрібно застосувати платформа для створення програмних додатків IDE.

Інтегроване середовище розробки (IDE) є комплексним програмним забезпеченням, призначеним для створення програмного забезпечення. Зазвичай, воно включає в себе редактор початкового коду, інструменти для автоматизації, збірки та налагодження програм. У більшості сучасних IDE є функціонал автодоповнення коду [18].

Зазвичай вони включають в себе редактор початкового коду, інструменти для автоматизації процесів розробки, засоби для компіляції та відлагодження програм. Багато сучасних ІСР також мають можливість автодоповнення коду, що сприяє швидшій та більш точній роботі розробників.

Для розробки мобільного застосунку було використане інтегроване середовище Xcode 10.

Xcode 10 - це інтегроване середовище розробки (ІСР), призначене для створення програмного забезпечення для платформ Apple, зокрема для iOS, macOS, watchOS та tvOS.

Основні характеристики Xcode 10 включають:

- редактор початкового коду - має потужний редактор з підсвічуванням синтаксису, автоматичним вирівнюванням та можливістю швидкого доступу до ресурсів та бібліотек;
- інструменти розробки - надає широкий набір інструментів для створення, відлагодження та тестування програм, включає в себе інтегровану систему контролю версій, засоби для профілювання та аналізу продуктивності, інструменти для створення інтерфейсу користувача та багато іншого;
- інтеграція зі середовищами розробки Apple - підтримує різні мови програмування, включаючи Swift, Objective-C та C++, та забезпечує інтеграцію з іншими середовищами розробки Apple, такими як Interface Builder для розробки

інтерфейсів користувача та SwiftUI для декларативної розробки інтерфейсу;

- підтримка веб-розробки - надає можливості для веб-розробки, зокрема підтримку JavaScript, HTML та CSS;

- інтеграція зі службами Apple - інтегровано з різними сервісами Apple, такими як App Store Connect для публікації програм в App Store, TestFlight для бета-тестування, а також засобами для розробки мобільних додатків для Apple Watch та Apple TV.

В цілому, Xcode 10 (рис3.7) є потужним інструментом для розробки програмного забезпечення для платформ Apple, який надає розробникам широкий набір інструментів та можливостей для створення високоякісних додатків.



Рисунок 3.7 – Початкове вікно Xcode 10

Використання вбудованого емулятора iOS у Xcode дозволяє перевірити створений додаток на останніх версіях операційних систем, а також на найновіших моделях Mac, iPhone, iPad, iPod, Apple TV та Apple Watch. Таким чином, Xcode забезпечує можливість тестування додатків на всіх поточних пристроях, що працюють під управлінням iOS.

3.7 Мова програмування Swift

На сьогоднішній день існує багато мов програмування. Деякі з них використовуються в обмеженому колі людей. Деякими мовами дуже зручно писати ігри чи створювати різноманітні додатки.

Swift - це нова мова програмування для додатків для iOS, macOS, watchOS, та tvOS, котра базується на кращому з C та Objective-C, без обмежень пов'язаних з сумісністю з C. Swift увібрав у себе шаблони безпечного програмування та додав сучасні можливості щоб зробити програмування простішим, гнучкішим (рис.3.8).



Рисунок 3.8 – Зображення іконки Swift

Мова Swift поєднує сучасну мову з інженерною культурою Apple, що робить її ідеальною інвестицією в майбутнє для розробників та компанії [19]. Її компілятор оптимізований для швидкодії, а сама мова — для розробки та масштабування. Ці фактори роблять Swift інструментом для створення додатків для різних платформ, таких як iOS, macOS, watchOS і tvOS.

Swift постійно розвивається, додаючи нові функції та можливості, що робить його важливим інструментом для розробників. Вона допомагає уникнути передачі неточних даних або змінних, не маючи правильної форми, що може зекономити багато часу у майбутньому.

Ця мова була розроблена як заміна мовам на основі C, зберігаючи продуктивність C у більшості завдань, але підтримуючи всі функції та системи iOS. Завдяки технології компілятора LLVM, код у Swift є швидким і ефективним,

що робить його ідеальним для розробки швидких та продуктивних додатків для різних пристроїв, таких як телевізори, годинники або мобільні пристрої.

Основний синтаксис Swift є простим та інтуїтивно зрозумілим, що полегшує написання виразного коду для розробників iOS. Він має функції, які забезпечують безпеку та чистоту коду, що робить його більш безпечним та легким для розуміння.

Крім того, у Swift широко використовуються константи, які забезпечують безпеку та зрозумілість коду, коли потрібно працювати зі значеннями, які не повинні змінюватись. Це зроблено для того, щоб забезпечити більшу надійність програм та полегшити їхнє розуміння.

Коментарі у мові Swift дуже схожі на коментарі в мові C. Однорядкові коментарі починаються з подвійного прямого слешу (`//`):

`//` - це коментар.

Багаторядкові коментарі починаються із символів слешу та зірочки (`/*`) і закінчуються символами зірочки та слешу (`*/`):

`/*` - це коментар, який написано у кілька рядків `*/`

Константи оголошуються за допомогою ключового слова `let`, а змінні – за допомогою ключового слова `var`.

Наприклад, для того, щоб відслідковувати кількість спроб користувача увійти в систему буде написано наступний код (рис.3.9):

```
let maximumNumberOfLoginAttempts = 10
var currentLoginAttempt = 0
```

Рисунок 3.9 – Код для входу у систему

Щоб оголосити “словник” (dictionary) з 4 елементів, кожен з яких містить ім’я та вік використано код (рис.3.10):

```
Let people = ("Anna": 67, "Beto": 8, "Jack": 33)
```

Рисунок 3.10 – Код для 4 елементів

Можна оголосити кілька констант чи змінних в один рядок, відділивши їх комами:

```
var x = 0.0, y = 0.0, z = 0.0
```

Оголошення масиву даних:

```
var fruits = ("mango", "apple", "kiwi")
```

Приклад оператора if, функцій .isEmpty та .count представлено на рисунку 3.11.

```
If fruits.isEmpty {
  print ("Фрути відсутні у масиві даних")
} else {
  print ("В масиві даних є\ (fruits.count) фруктів")
}
```

Рисунок 3.11 – Код для оператора if, .isEmpty та .count

Можна надрукувати поточне значення константи чи змінної за допомогою функції print(_:separator:terminator:)

```
Print (friendlyWelcome)
// Надрукую "Вітаю!"
```

Програмування Swift було розроблено для того, щоб розробник міг легко вводити і перевіряти текст, дозволяє бачити результати одразу, без витрат часу на збірку і запуск додатка. Мовою Swift можна створювати додатки для iOS, macOS, watchOS, та tvOS.

3.8 Графічний редактор Adobe Photoshop CC

Adobe Photoshop CC використовувався для створення вихідного макету програми та більшості елементів дизайну, таких як фон, іконки програми та кнопки.

Adobe Photoshop є графічним редактором, розробленим та розповсюджуваним компанією Adobe Systems [20]. Цей продукт є визнаним лідером на ринку комерційних засобів редагування растрових зображень і є найбільш відомим продуктом в арсеналі Adobe. На сьогоднішній день редактор доступний для платформ Apple Mac OS, Microsoft Windows, iOS та Android. Основне вікно програми зображено на рисунку 3.12.



Рисунок 3.12 - Вікно Adobe Photoshop

Функції Adobe Photoshop включають широкий спектр інструментів та можливостей для редагування та обробки растрових зображень. Деякі з найбільш важливих функцій включають:

- має набір інструментів для виправлення дефектів зображення, таких як видалення плям, розгладження зморшок або виправлення червоних очей;
- монтаж та об'єднання зображень - можна об'єднувати кілька зображень в одне, створювати композиції, використовуючи шари та маски;
- великий вибір фільтрів та ефектів, що дозволяють створювати різноманітні стилі та настроювати вигляд зображень;
- додавати та форматування тексту на зображеннях, створення ефективних надписів та додавати текстові елементи;
- корекція кольорів та збалансування - зміна колірну гаму, коригування

експозицію та налаштування інших параметрів для досягнення оптимального кольорового балансу. нням обробкою зображень та редагуванням без змін на вихідному зображенні;

- робота з 3D-зображеннями - має інструменти для редагування та створення тривимірних зображень, а також можливості для роботи з текстурами та освітленням.

В процесі роботи над даним розділом проєкту було визначено вимоги до платформи реалізації розробки. На основі проведеного аналізу було обрано методологію RAD яка націлена на швидке створення і розгортання програмних додатків. Торкнулися питання архітектури додатку, створено алгоритм роботи модуля, обрано середовище розробки застосунків Xcode, мову програмування Swift та графічного редактора Adobe Photoshop, який буде використовувався для створення прототипу, елементів .

4 РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ

4.1 Створення проєкту в середовищі Xcode

Для створення проєкту у середовищі Xcode потрібно у вікні привітання, яке зображене на рисунку 4.1 обрати пункт “Create a new Xcode project”.



Рисунок 4.1 - Вікно створення нового проєкту у Xcode 10

Далі потрібно вибрати один з шаблонів програми та операційну систему, для якої дана програма розробляється (iOS, macOS, watchOS, tvOS або Cross-platform (для всіх)). Вікно вибору шаблонів зображене на рисунку 4.2.

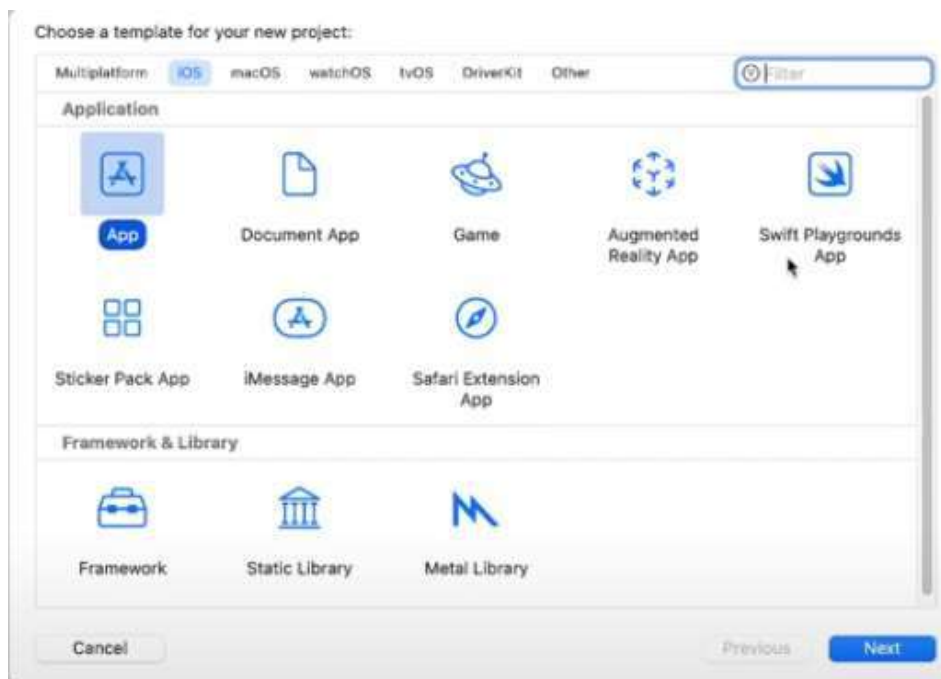


Рисунок 4.2 - Вікно вибору шаблонів в Xcode

Основним файлом, що створює Xcode в папці проекту є файл з розширенням *.xcodeproj.

Наступним кроком слід назвати проект, вказати розробника, ідентифікатор організації, вибрати мову, на якій розробляється проект та пристрій, iPhone чи iPad у випадку розробки для iOS, після чого потрібно буде всього лиш вказати, де слід зберігати проект на комп'ютері і можна починати з ним працювати (рис.4.3).

4.2 Опис коду додатку

Для досягнення естетичного вигляду вашого застосунку, ви можете встановити зображення з певною текстурою або фотографію на його фон замість

простого фону кольору. Це реалізовується за допомогою коду, який показано на прикладі, рисунок 4.4.

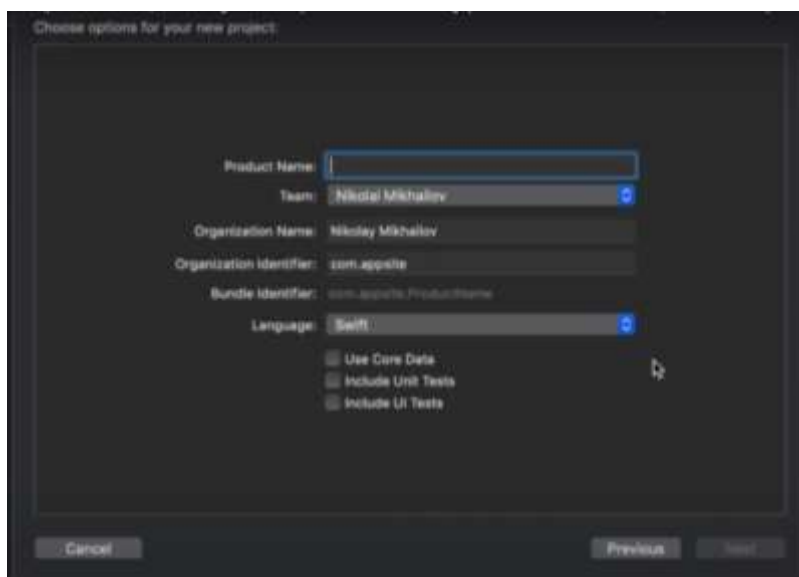


Рисунок 4.3 – Вікно інформації для створення нового проєкту

Код усіх файлів розробки даного проєкту наведено у додатках А - К.

```
override func viewWillAppear(_ animated: Bool) {
    //setting the background image to the tableview
    let backgroundImage = UIImage(named: "background.jpg")
    let imageView = UIImageView(image: backgroundImage)
    self.tableView.backgroundView = imageView
    //hiding unusable lines in empty cells
    tableView.tableFooterView = UIView(frame: CGRect.zero)
    // center and scale background image
    imageView.contentMode = .scaleAspectFill
}
```

Рисунок 4.4 – Код встановлення фону застосунку з файлу

Для необхідності виведення на екран повідомлення чи попередження під час певної дії створено наступний код (рис.4.5).

Також важливо перевірити, чи поле введення не залишилося порожнім. Приклад такої перевірки наведено у вихідному коді (рис. 4.6), який дозволяє отримати назву, введену користувачем, але лише якщо воно не є порожнім.

```
func didTapAddItemButton(_ sender: UIBarButtonItem)
{
    let alert = UIAlertController(
        title: "Movie Title:",
        message: "",
        preferredStyle: .alert
    )
```

Рисунок 4.5 – Код для виведення повідомлення

```
    if let title = alert.textFields?[0].text,
        title.characters.count > 0
    {
        self.addNewFilmItem(title: title)
```

Рисунок 4.6 – Код для перевірки заповненості поля

Для додавання в повідомленнях текстового поля чи кнопки використовується наступний код (рис.4.7):

```
        alert.addTextField(configurationHandler: nil)
        alert.addAction(UIAlertAction(title: "Cancel", style:
        .cancel, handler: nil))
        alert.addAction(UIAlertAction(title: "Confirm", style:
        .default, handler:
            { (_) in
```

Рисунок 4.7 – Код додавання текстового поля, кнопки «Відмінити» та «Підтвердити» до повідомлення

На рисунку 4.8 наведено код функцію, яка відповідає за додавання нових елементів до списку в програмі.

```
private
func addNewFilmItem(title: String)
{
    let newIndex = filmItems.count
    filmItems.append(FilmItem(title: title))
    tableView.insertRows(at: [IndexPath(row: newIndex,
    section: 0)], with: .top)
}
```

Рисунок 4.8 – Функція для додавання нового елементу в список

4.3 Додання власних міток на карті за допомогою MapKit

MapKit - це фреймворк, розроблений компанією Apple для роботи з картами та геолокацією в програмах, створених для платформ iOS, macOS, watchOS та tvOS. Цей фреймворк надає розробникам інструменти для відображення карт, взаємодії з ними, відслідковування місцеположення користувача та багато іншого [21]. MapKit дозволяє інтегрувати функціональність карт прямо в програму, щоб користувачі могли використовувати карти та отримувати інформацію про місця з інших джерел, таких як Apple Maps (рис.4.9).



Рисунок 4.9 – Взаємозв'язок MapKit – Swift - iOS

Ця система також дозволяє додавати анотації на карту, розміщувати накладення та виконувати зворотне геокодування вибірок для визначення міток інформації за заданими координатами на карті. Щоб позначити будь-яке місце, якого немає на карті, є можливість створювати власні мітки, що можна зробити за допомогою коду, наведеного на зображенні нижче.(рис.4.10).

Повний код файлу MapViewController.swift наведено у додатку Ж.2.

За допомогою створених міток є можливість показувати зображення за допомогою масиву це відображає наступні рядки коду (рис.4.11, додаток Д).

Для використання геолокаційних даних, необхідно заздалегідь повідомити користувача та запросити його згоду на доступ. Це виконується за допомогою коду, який представлено на рисунку 4.12.

```

let MapCenter =
CLLocationCoordinate2DMake(49.85096864,24.03086744)
    let region = MKCoordinateRegionMakeWithDistance(MapCenter,
regionRadius * 2, regionRadius * 2)
    Map.setRegion(region, animated: true)

    coordinates = [[Довгота, Широта]]
    names = ["Назва мітки"]
    addresses = ["Адреса"]
    phones = ["Номер телефону закладу"]
    self.Map.delegate = self
    for i in 0...7
    {
        let coordinate:[Double] = coordinates[i] as! [Double]
        let point = CustomAnnotation(coordinate:
CLLocationCoordinate2D(latitude: coordinate[0], longitude:
coordinate[1]))

```

Рисунок 4.10 - Встановлення власних міток на карті

```

        point.image = UIImage(named: "image-\(i+1).jpg")
        point.name = names[i]
        point.address = addresses[i]
        point.phone = phones[i]
        self.Map.addAnnotation(point)

```

Рисунок 4.11 – Код для показу зображень в мітках

```

override func viewWillAppear(_ animated: Bool) {
    locationAuthStatus()
}
func locationAuthStatus() {
    if CLLocationManager.authorizationStatus() ==
    .authorizedWhenInUse {
        Map.showsUserLocation = true
    } else {
        locationManager.requestWhenInUseAuthorization()
    }
}

```

Рисунок 4.12 - Запит доступу до даних геолокації

За допомогою коду, наведеного нижче (рис.4.13) створюються кнопки для переключення з карти на супутник.

```

@IBAction func type_satellite(sender: UIButton) {
    Map.mapType = MKMapType.satellite
}
@IBAction func type_standard(sender: UIButton) {
    Map.mapType = MKMapType.standard
}

```

Рисунок 4.13 - Кнопка для переключення типу карти

Для швидкого знаходження місць поблизу можна миттєво перейти до поточного місцезнаходження користувача за допомогою відповідного коду (рис.4.14).

```

@IBAction func getLocationButton(sender: UIButton) {
    Map.setCenter(Map.userLocation.coordinate, animated: true)
}

```

Рисунок 4.14 – Кнопка повернення до локації користувача

Картою можна маніпулювати не лише за допомогою екранних жестів, але й за допомогою кнопок (рис.4.15). За допомогою цього коду також можна робити зміну типу мапи, збільшення/зменшення мапи та повернення до місцезнаходження користувача.

```

@IBAction func zoomIn(sender: UIButton) {
    let span = MKCoordinateSpan(latitudeDelta:
Map.region.span.latitudeDelta/2, longitudeDelta:
Map.region.span.longitudeDelta/2)
    let region = MKCoordinateRegion(center: Map.region.center,
span: span)
    Map.setRegion(region, animated: true)
}
@IBAction func zoomOut(sender: UIButton) {
    let span = MKCoordinateSpan(latitudeDelta:
Map.region.span.latitudeDelta*2, longitudeDelta:
Map.region.span.longitudeDelta*2)
    let region = MKCoordinateRegion(center: Map.region.center,
span: span)
    Map.setRegion(region, animated: true)
}

```

Рисунок 4.15 – Код створення кнопок управління картою

Проведено розробку додатку за допомогою середовища Xcode та фреймворку MapKit. Детально описані елементи з наведеними кодами представлені у додатках А-К.

5 ТЕСТУВАННЯ ПЛАТФОРМИ

Тестування цієї системи буде проводитися безпосередньо під час розробки програми. У ньому візьмуть участь тестувальник, системний аналітик та молодший розробник. Планується використання як динамічних, так і статичних методів тестування.

Під час статичного тестування аналізується вся документація, що надійшла відповідно до життєвого циклу програми. Це охоплює технічне завдання, специфікації та вихідний код програми на мові програмування. Вся ця документація перевіряється на відповідність стандартам програмування. Статичне тестування дозволяє встановити, наскільки програма відповідає критеріям та вимогам замовника. Цей процес проводять системний аналітик та молодший розробник.

У динамічних методах перевірки програми проводиться в процесі її активного виконання. Ця перевірка включає в себе велику кількість тестів або наборів підготовлених вхідних даних. Кожен тест запускається для програми, і під час його виконання збираються та аналізуються дані щодо можливих відмов та помилок. Динамічне тестування виконується тестувальником спільно з молодшим розробником.

Введення системи в роботу буде здійснюватися в пробному режимі наступним чином: для всіх категорій користувачів необхідно встановити додаток на свої мобільні пристрої, щоб отримати доступ до розробленої системи. Для цього треба завантажити додаток на пристрій та виконати його запуск. Після першого використання програми користувачеві буде запропоновано надати

відповідні дозволи для її правильної роботи. Крім того, у майбутньому буде можливість завантажити цей додаток з офіційного магазину додатків для iOS.

5.1 Встановлення програми для проведення тестування

Для проведення тестування можна скористатися наявними емуляторами для всіх версій операційної системи iOS та всіх сумісних пристроїв, що доступні в середовищі Xcode.

Для подальшого розповсюдження або навіть простого використання свого додатку необхідно мати "Профіль розробника Apple". Цей профіль дозволить розповсюджувати тестові версії свого додатку через TestFlight для обраних користувачів за їх Apple ID. Це дозволить отримувати відгуки користувачів та швидко вносити зміни та виправлення у додаток для подальшого тестування. Крім того, такий профіль розробника дозволяє публікувати свої додатки в Apple App Store для завантаження всіма бажаючими.

5.2 Публікація застосунку в Apple App Store

Для розміщення застосунка в App Store необхідно мати платний розробницький обліковий запис, використовувати середовище розробки Xcode та, звісно, сам застосунок[22].

Перед тим як застосунок буде завантажено до App Store, він має бути підписаний сертифікатом розробника. Це необхідно, щоб користувачі магазину могли бути впевнені, що завантажують додаток саме від конкретного розробника, а не від когось іншого. Процедура підписування дозволяє операційній системі

ідентифікувати розробника та переконатися, що застосунок не був змінений після його компіляції.

Щоб створити сертифікат потрібно виконати наступні дії:

- перейти до developer.apple.com, перейти до розділу «сертифікати», натиснути на плюсик (рис.5.1);
- вибрати тип сертифіката Production — App Store and Ad Hoc;
- отримуючись наведених інструкцій сформувати Certificate request та завантажити його на портал.



Рисунок 5.1 – Підписання оформлення сертифікату

Сертифікат складається з пари ключів асиметричного шифрування: приватного та публічного. Під час збирання застосунку в Xcode створюється цифровий підпис, заснований на приватному ключі. Публічний ключ, доступний також Apple, є використаним для перевірки цього підпису.

Це унікальний ідентифікатор програми, який складається з двох частин: TeamID, який є ідентифікатором розробника і надається Apple під час реєстрації облікового запису розробника, і BundleID, який визначається розробником при реєстрації програми в обліковому записі (рис.5.2). Цей ідентифікатор потрібний для однозначної ідентифікації програми у всіх системах, таких як iTunes Connect, App Store і інші.

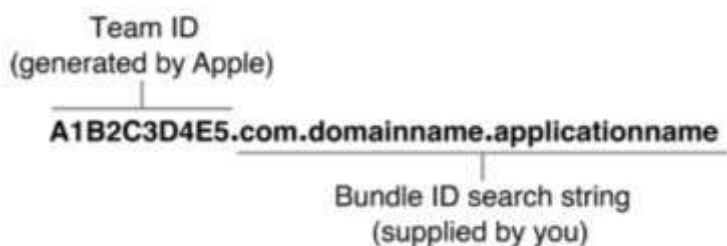


Рисунок 5.2 – Склад ідентифікатора програми

Для його створення виконаємо наступні кроки:

- перейдіть до секції AppIDs і клацніть на «+» в правому верхньому куті;
- введіть у поле «Опис App ID» - умовна назва програми для розробника, яка не буде відображатися в App Store, наприклад, TestApplication;
- заповніть поле «Явний ID програми» - це BundleID, наприклад, ru.myStartup.testApplication;
- в секції «Послуги програми» активуйте сервіси, які були необхідні під час розробки програми.

Для створення профілю Provisioning profile, який зв'язує AppID та сертифікат розробника (публічний ключ сертифіката). У разі розвитку provisioning profile він також містить UDID всіх пристроїв, на яких можливий запуск програми.

Потрібно виконати наступні дії:

- перейти до розділу iOS Provisioning Profiles, натисніть на «+» праворуч вгорі;
- обрати тип профайлу — «App Store»;
- вказати AppID, для якого створюється профайл — вкажіть раніше створений AppID;
- обрати «Виберіть сертифікат» - позначаємо галочкою сертифікат, який був створений раніше;
- дати назву профайлу, наприклад TestApplicationStoreProfile.

Збережіть згенерований профайл на комп'ютер, відкрийте його, двічі клацнувши. Тепер сертифікат доступний для XCode.

Для публікації будь-якої кількості застосунків достатньо одного сертифіката. Після отримання профілю розробника, додавання застосунку до iTunes Connect здійснюється в розділі MyApps (рис.5.3).

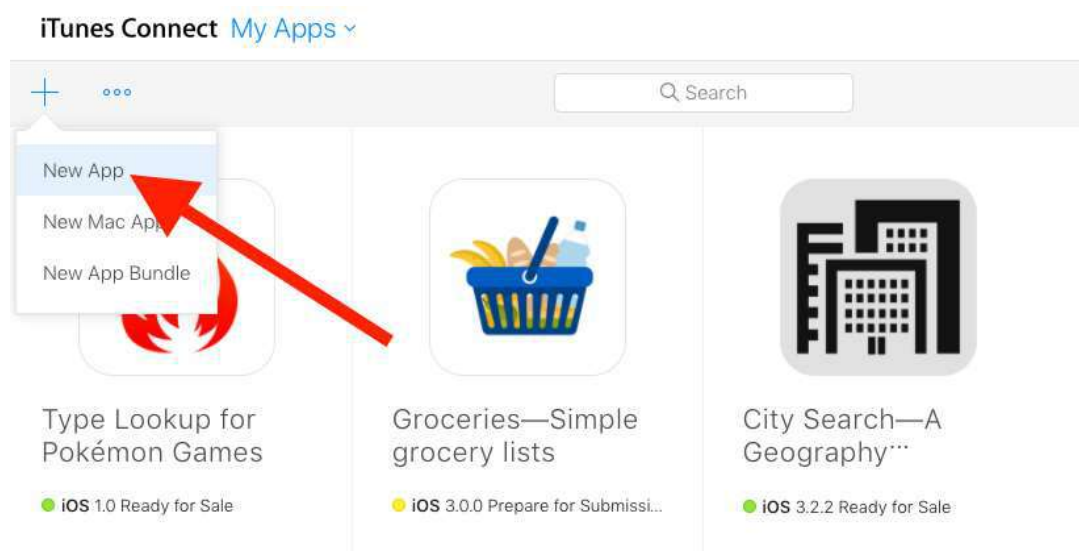


Рисунок 5.3 - Вкладка додавання застосунку на сайті iTunes Connect

У відкритій формі потрібно вказати платформу - iOS , назву застосунку для відображення в App Store, основну мову застосунку та унікальний ідентифікатор застосунку (AppID), який використовується для фінансових звітів та аналітики (рис.5.4).

Якщо форма заповнена вірно, відразу відкривається сторінка додатку (рис.5.5).

Після цього ми перейдемо на сторінку застосунку. Нам потрібно буде внести інформацію про ціну застосунку в магазині та надати маркетингові матеріали (скріншоти, відео, ключові слова для пошуку, опис, віковий рейтинг, назву власника авторських прав, контактну інформацію розробника) у розділі "Ціни та доступність" (Pricing and Availability). На вкладці ліворуч «Pricing and Availability» заповнюємо інформацію про вартість програми у сторі. На вкладці зліва з номером версії заповнюємо рекламну інформацію:

New App

Platforms ?

☒ iOS ☐ tvOS

Name ?

MyStartupTestApplication

Primary Language ?

Bundle ID ?

TestApplication - ru.myStartup.testApplication

SKU ?

ru.myStartup.testApplication

Cancel Create

Рисунок 5.4 – Заповнення форми

App Store Features TestFlight Activity

APP STORE INFORMATION

App Information Save

This information is used for all platforms of this app. Any changes will be released with your next app version.

Localizable Information Russian ?

Name ?

MyStartupTestApplication

Privacy Policy URL ?

http://example.com (optional)

Subtitle ?

Optional

General Information

Bundle ID ? Register a new bundle ID.

TestApplication - ru.myStartup.testApplication

Your Bundle ID ru.myStartup.testApplication

SKU ?

ru.myStartup.testApplication

Apple ID ?

Primary Language ?

Russian

Category ?

Primary

Secondary (optional)

License Agreement Edit

Apple's Standard License Agreement

Rating ?

No Rating

Additional Information

[View on App Store](#)

Рисунок 5.5 – Сторінка додатку

Щоб зберегти зміни, використовуйте кнопку Save нагорі (рис.5.6):



Рисунок 5.6 – Збереження змін після внесення інформації

Також існує розділ, де можна вказати конкретну версію застосунку, яку буде надіслано на перевірку, і після цього - у магазин. Важливо відзначити, що без всіх обов'язкових іконок застосунків не пройде автоматичну перевірку в iTunes Connect.

Відкрийте ваш проект у XCode та перейдіть до налаштувань проекту. Важливо, щоб значення Bundle Identifier відповідало значенню BundleID, яке ви вказали при створенні AppID. Крім того, рекомендується вимкнути автоматичне керування підписуванням у XCode (рис.5.7). У розкритому списку Provisioning Profile оберіть той, який ви недавно створили.

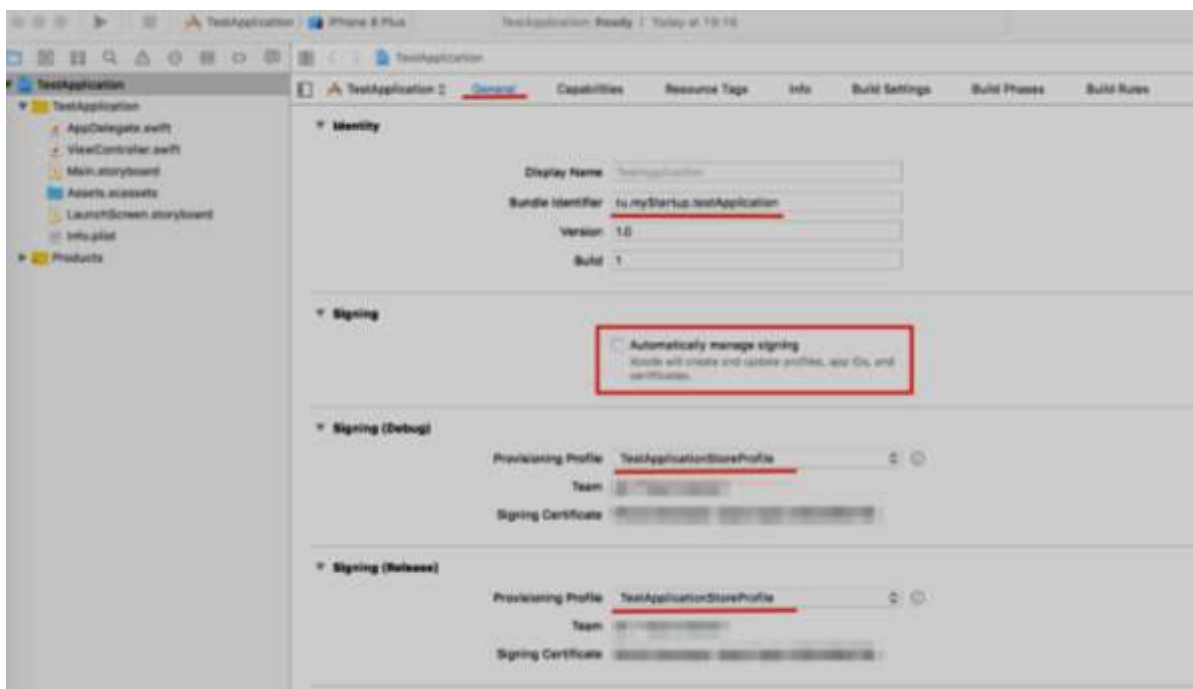


Рисунок 5.7 – Налаштування профілю

Тепер у Xcode потрібно перейти до меню "Product" та обрати опцію "Archive", щоб скомпілювати та відправити проєкт в iTunes Connect. Після компіляції застосунку відкриється вікно органайзера Xcode, де буде доступна кнопка "Upload to App Store", яку можна побачити на рисунку 5.8.



Рисунок 5.8 - Компілювання та відправка проєкту в iTunes Connect

Після цього відкриється вікно налаштувань для завантаження в App Store. Можна залишити налаштування без змін та перейти далі (рис.5.9).

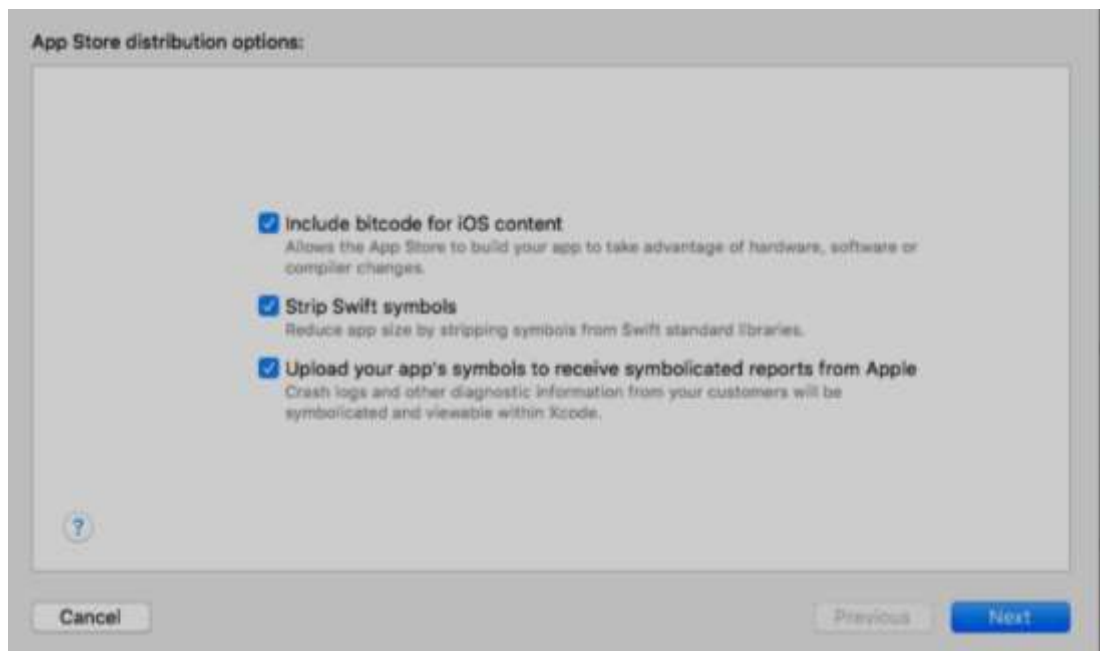


Рисунок 5.9 – Вікно налаштувань для завантаження в App Store

На наступному кроці потрібно вибрати provisioning profile з списку відповідних (рис.5.10). Xcode підготує архів для завантаження в iTunes Connect,

який можна побачити на рисунку 5.11 і ще раз показує обрані нами параметри.

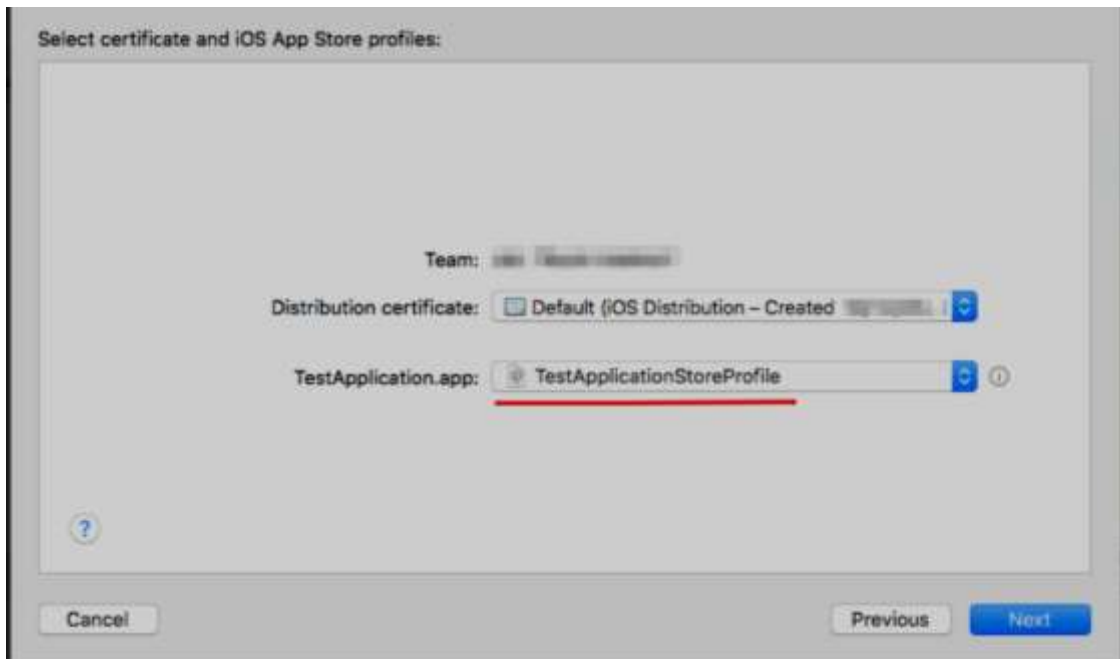


Рисунок 5.10 - Обрання provisioning profile із переліку

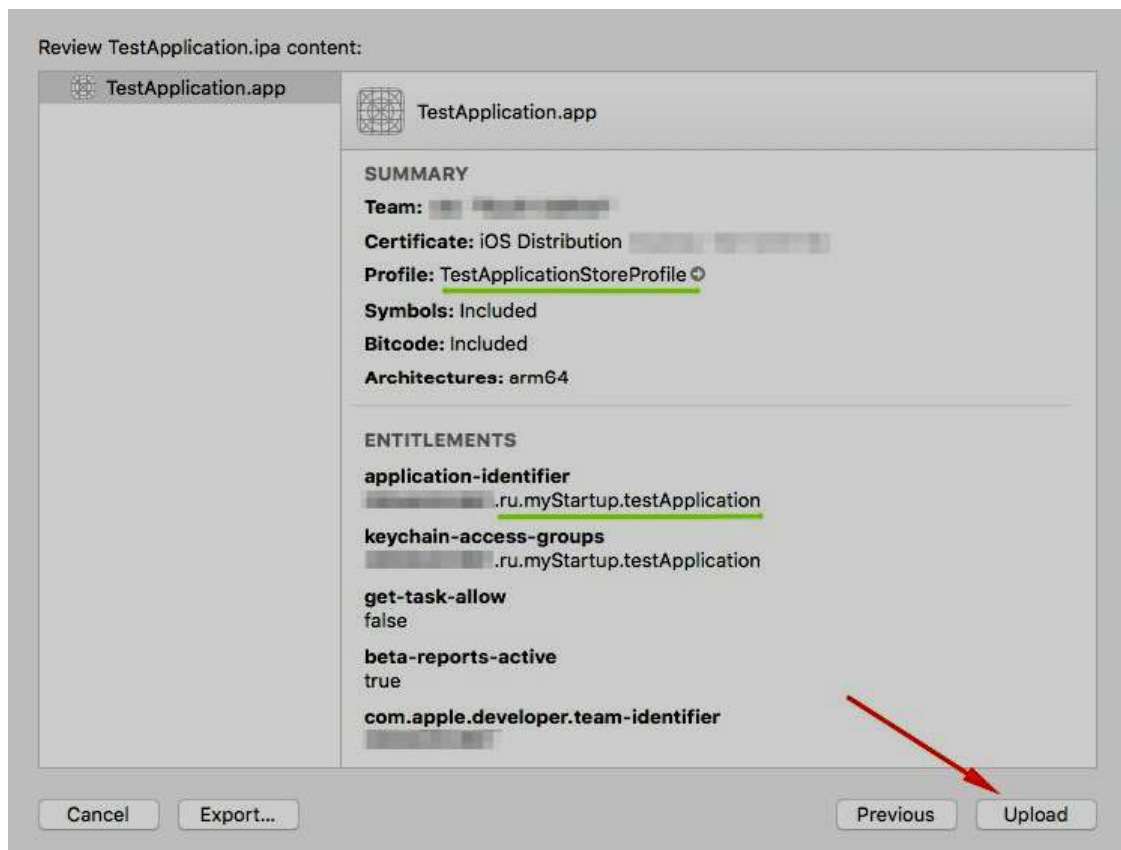


Рисунок 5.11 - Архів для відвантаження в iTunes Connect

Після натискання кнопки "Upload", час очікування залежить від швидкості з'єднання. Якщо все пройшло успішно, Xcode повідомить про вдале завантаження в iTunes Connect (рис.5.12).

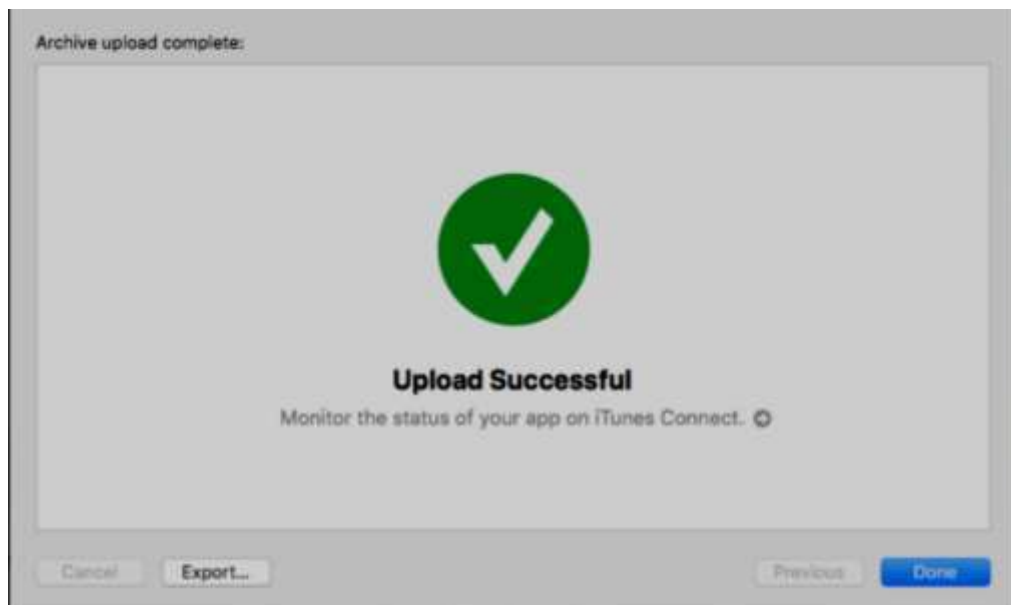


Рисунок 5.12 – Результат успішної перевірки

Далі, потрібно відправити застосунок на огляд. Це можна зробити в iTunes Connect, перейшовши на вкладку "Activity" (рис.5.13). Для проектів на Swift автоматична перевірка зазвичай займає близько 30 хвилин. Протягом цього часу стан застосунка буде "Processing".

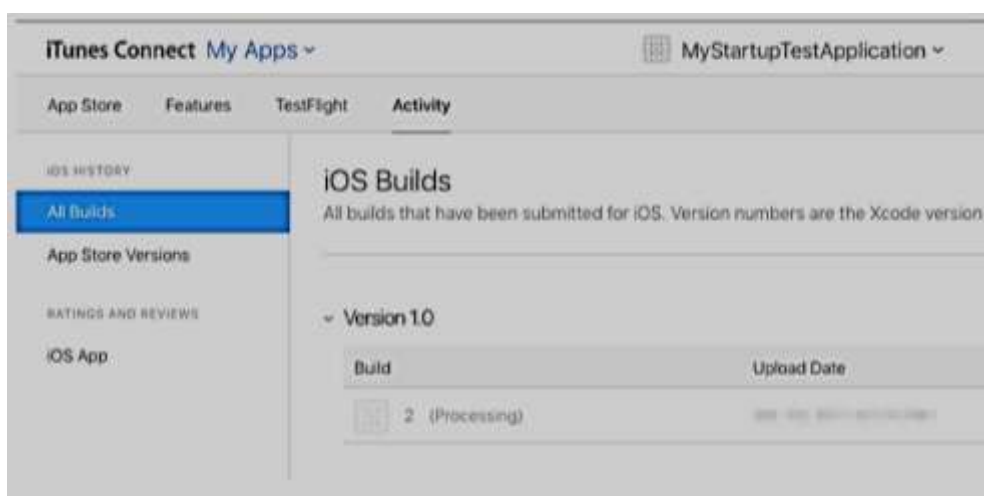


Рисунок 5.13 – Відправка застосунка на перевірку

Після завершення перевірки застосунк буде доступний для вибору на сторінці інформації про його версію (рис.5.14).

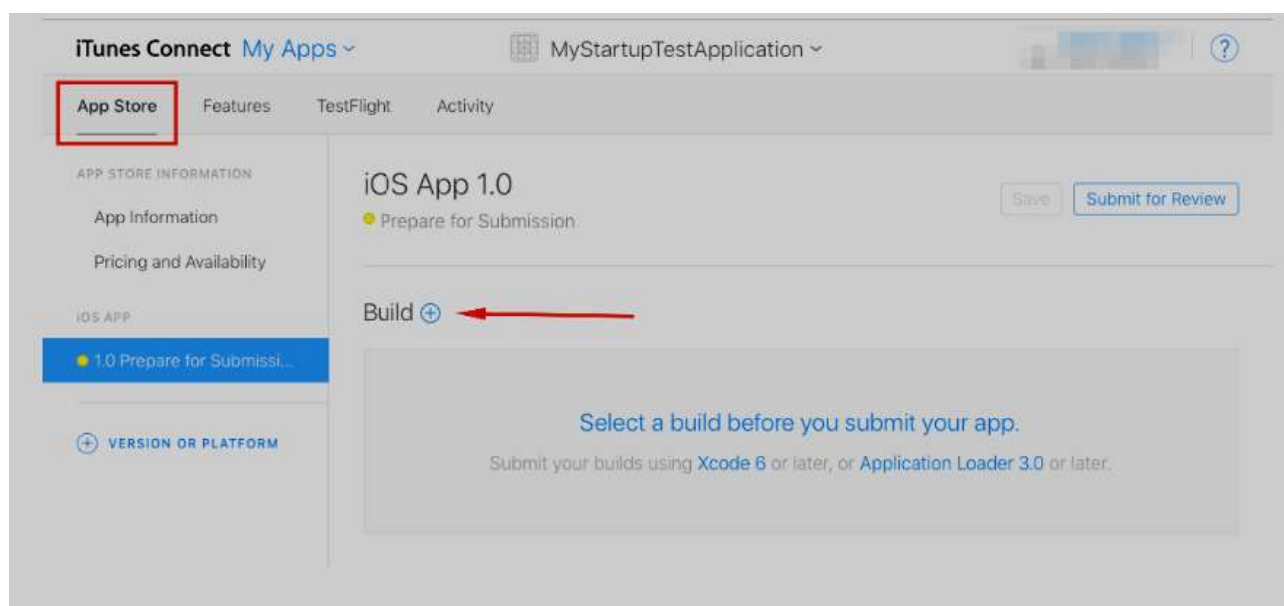


Рисунок 5.14 - Інформація про вибору на сторінці інформації

Наступним кроком буде обирання типу зборки (рис.5.15).

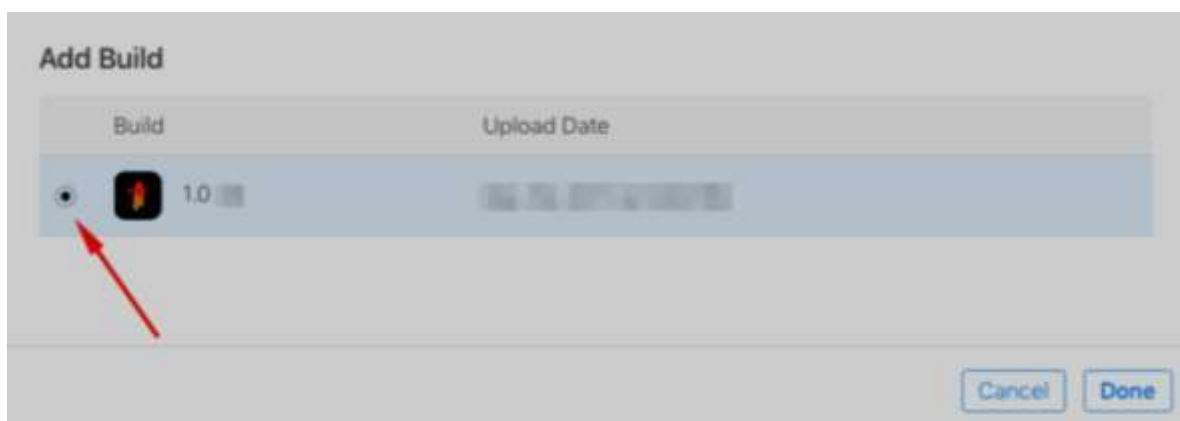


Рисунок 5.15 – Обрання типу зборки

Після цього в iTunes Connect з'явиться іконка, потрібно зберегти зміни та можна відправити застосунок на огляд (рис.5.16).

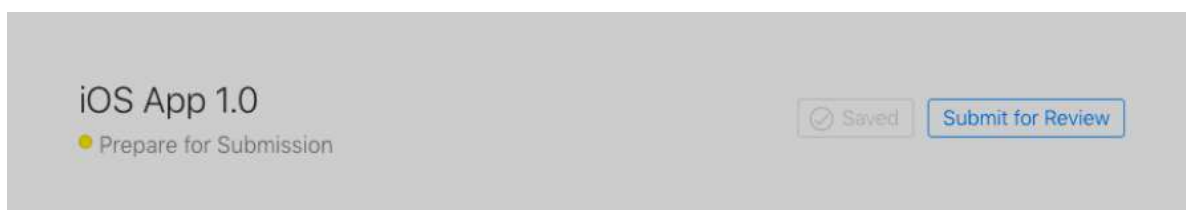


Рисунок 5.16 –Відправлення застосунку на огляд

Протягом кількох днів буде отримано відповідь щодо нашого застосунку, повідомлять, чи буде він доступний в App Store, або повідомлять про порушення певних норм або правил, що вимагають змін.

5.3 Додання інформації до «ТОП»

Internet Movie Database (IMDb) - це великий онлайн-ресурс, який містить інформацію про фільми, телепередачі, акторів, режисерів, продюсерів, сценаристів та інших учасників індустрії розваг. Вона є одним з найбільших та найвпливовіших джерел інформації про кіно у світі.

IMDb містить обширну базу даних фільмів з різних країн та різних епох. Кожен фільм має свою сторінку з докладною інформацією про нього, включаючи заголовок, рік виходу, жанр, рейтинги, огляди, акторський склад, інформацію про студію та бюджет. Крім того, IMDb також надає користувачам можливість додавати свої відгуки та оцінки для фільмів, створювати списки перегляду, обговорювати фільми на форумах та отримувати оновлення про нові релізи.

Ця платформа стала невід'ємною частиною кіноманів та професіоналів у галузі кінематографу, надаючи доступ до широкого спектру інформації про світ кіно.

Сайт також пропонує стрічку новин, яка постійно оновлюється статтями, і використовує систему персональних рекомендацій для підбору найкращого фільму за жанром, новизною або на основі останніх переглянутих користувачем фільмів.

У базі даних наразі міститься докладна інформація про понад 3 мільйони кінострічок і телесеріалів, а також дані про майже 6,9 мільйонів акторів, режисерів та інших фахівців зі всього світу. На рисунку 5.17 наведено знімок екрана сайту.

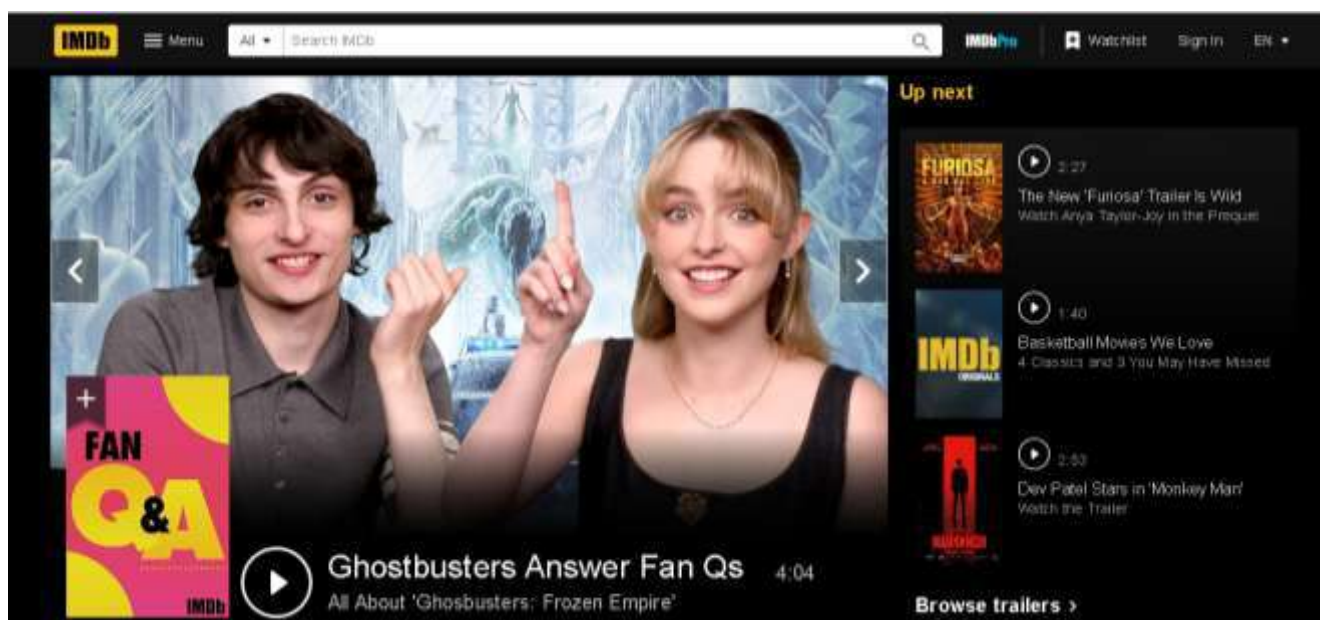


Рисунок 5.17 - Скріншот сайту IMDb

Щоб отримати інформацію про ТОП з веб-сайту IMDb.com, використовується код, який поданий на рисунку 5.18.

```
class TopList:
    UIViewController, UITableViewDataSource, UITableViewDelegate {
        final let urlString = "https://api.myjson.com/bins/pvityx"
        //Old url
        "http://api.myapifilms.com/imdb/top?start=1&end=10&token=67fc7
        a85-30c9-4031-a604-f126d958e077&format=json&data=0"
        @IBOutlet weak var tableView: UITableView!
        var titleArray = [String]()
        var yearArray = [String]()
        var imgURLArray = [String]()
        override func viewDidLoad() {
            super.viewDidLoad()
            self.title = "Top movies"

            self.navigationController?.navigationBar.titleTextAttributes?
            = [NSForegroundColorAttributeName: UIColor.white]
            self.downloadJsonWithTask()
        }
    }
```

Рисунок 5.18 – Частина коду з файлу TopList.swift

Повний код файлу TopList.swift додається в Додаток Г. Цей файл відповідає за взаємодію з веб-сайтом IMDb.com для отримання даних і їх подальшого відображення в застосунку.

Застосунок під назвою "FilmList" готовий до використання користувачами. Він повністю відповідає всім поставленим вимогам на етапі постановки завдання.

Тестування додатку виконується безпосередньо при його розробці за допомогою вбудованих в Xcode емуляторів всіх версій iOS та пристроїв для яких ведеться розробка.

Впроваджено систему та публікацію застосунку в магазині Apple AppStore. Для подальшого поширення може використовуватись сервіс TestFlight, який надає змогу розробнику контролювати кількість користувачів, які можуть використовувати додаток та отримувати всі необхідні звіти про роботу продукту.

ВИСНОВКИ

За результатами виконання дипломного проєкту було розроблено мобільний застосунок для пошуку та вибору фільмів.

Було проаналізовано особливості розробки мобільних застосунків, здійснено порівняння з існуючими застосунками. Розглянуті питання методів побудови засобів мобільних застосунків та сервісів для планування відпочинку. Створено алгоритм роботи модуля, діаграму прецедентів. Обґрунтовано інтегроване середовище розробки застосунків Xcode 10.

При створенні проєкту була використана мова програмування Swift та графічний редактор Adobe Photoshop.

Було створено та протестовано робочу версію системи, яка створює користувацький список, показує ТОП фільмів з сайту IMDb та відображає на карті найближчі до користувача кінотеатри.

В майбутньому створений застосунок може вдосконалюватись, за рахунок оновлення iOS.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Мобільний додаток і мобільний сайт: в чому різниця, і що краще? [Електронний ресурс] – Режим доступу: <https://webexpert.com.ua/ua/mobilnij-dodatok-i-mobilnij-sajt> (дата звернення 30.03.2024).
2. Поняття мультимедіа Кодування аудіо та відео даних [Електронний ресурс] – Режим доступу: <https://sites.google.com/view/54-> (дата звернення 01.04.2024).
3. Принципи розробки мобільного додатка [Електронний ресурс] – Режим доступу: <https://webcase.com.ua/uk/razrabotka-mobilnyh-prilozhenij/> (дата звернення 28.03.2024).
4. ТОП 50 найпопулярніших сайтів у світі [Електронний ресурс] – Режим доступу: <https://marketer.ua/ua/top-50-sites-of-world/> (дата звернення 29.04.2024).
5. Spotify, Apple music чи Youtube music - обираємо найкращий музичний стрімінговий сервіс [Електронний ресурс] – Режим доступу: <https://www.itbox.ua/ua/blog/Spotify-Apple-Music-chi-YouTube-Music--Obiraemo-naykraschiiy-muzichniy-strimingoviy-servis/> (дата звернення 29.03.2024).
6. 8 стрімінгових сервісів, з якими музика завжди під рукою [Електронний ресурс] – Режим доступу: <https://obrii.com.ua/informatsiya/46210-8-strimingovyh-servisov-s-kotorymi-muzyka-vsegda-pod-rukoj.html> (дата звернення 21.03.2024)
7. Ранжування сайту: 10 причин, що заважають потрапити в ТОП [Електронний ресурс] – Режим доступу <https://www.taina.com.ua/ranzhuvannya-sajtu-10-prychyn-shho-zavazhayut-potrapytu-v-top> (дата звернення 11.04.2024).
8. Топ факторів ранжування сайту в пошуку [Електронний ресурс] – Режим доступу: <https://uaateam.agency/blog/top-faktoriv-ranzhuvannya-sajtu-v-poshuku-v-2019-rotsi> (дата звернення 15.04.2024).
9. Алгоритми пошукових систем Google та YouTube [Електронний ресурс] – Режим доступу: <https://itforce.ua/blog/iak-optymizuvaty-youtube-dlia-poshuku/> (дата звернення 01.04.2024).

10. Основи персональних товарних рекомендацій [Електронний ресурс] – Режим доступу: <https://esputnik.com/uk/blog/gajd-z-personalnih-tovarnih-rekomendacij-tehnologiya-ta-kejsi> (дата звернення 11.04.2024).
11. Алгоритм найближчого сусіда [Електронний ресурс] – Режим доступу: https://uk.wikipedia.org/wiki/Метод_найближчого_сусіда#:~:text= (дата звернення 02.04.2024).
12. Частка ринку Android та iOS [Електронний ресурс] – Режим доступу: <https://root-nation.com/ua/news-ua/it-news-ua/ua-android-ios-statistika> (дата звернення 03.04.2024).
13. Які існують методи управління проєктами у сфері IT? [Електронний ресурс] – Режим доступу: <https://indevlab.com/uk/blog-ua/yaki-isnuyut-metodi-upravlinnya-proektami-u-sferi-it/> (дата звернення 17.04.2024).
14. Технологія швидкого проектування RAD [Електронний ресурс] – Режим доступу: https://elearning.sumdu.edu.ua/free_content/lectured:de1c9452f2/index.html (дата звернення 01.04.2024).
15. Діаграма послідовності [Електронний ресурс] – Режим доступу: <https://dou.ua/forums/topic/40575//> (дата звернення 2.04.2024).
16. Зв'язки використання , діаграма класів [Електронний ресурс] – Режим доступу: <https://www.dstu.dp.ua/Portal/Data/3/19/3-19-kl34.pdf> (дата звернення 15.04.2024).
17. Створення схеми діаграми станів UML [Електронний ресурс] – Режим доступу: <https://support.microsoft.com/uk-ua/office> (дата звернення 25.04.2024).
18. Встановлення IDE (Інтегрованого Середовища Розробки) [Електронний ресурс] – Режим доступу: <https://acode.com.ua/urok-4-vstanovlennya-ide-integrovanogo-seredovyshha-rozrobky/> (дата звернення 30.04.2024).
19. Про Swift [Електронний ресурс] – Режим доступу: https://book.swift.org.ua/book/0_welcome_to_swift/about_swift (дата звернення 30.04.2024).
20. Як використовувати Photoshop [Електронний ресурс] – Режим доступу: <https://designtalk.club/yak-vykorystovuvaty-photoshop-12-bazovyh-funktsij-chastyna->

1/ (дата звернення 03.05.2024).

21. Розробка мобільних додатків [Електронний ресурс] – Режим доступу: <https://brander.ua/technologies/mapkit> (дата звернення 07.05.2024).

22. Викладаємо додаток до App Store [Електронний ресурс] – Режим доступу: <https://habr.com/companies/touchinstinct/articles/345336/> (дата звернення 20.05.2024).

22. MapKit для новачків: розробляємо карти в Android-додатку [Електронний ресурс] – Режим доступу: <https://habr.com/companies/joydev/articles/754800/#2> (дата звернення 17.05.2024).

ДОДАТОК А

Інтерфейс користувача

Лістинг А.1 - Управління і відображення вмісту користувача. Код файлу tableViewController.swift

```

class tableViewController: UIViewController,
UITableViewDataSource, UITableViewDelegate, TableViewCellDelegate {
    @IBOutlet weak var tableView: UITableView! var filmItems =
[FilmItem]()
    let pinchRecognizer = UIPinchGestureRecognizer()

    override func viewDidLoad() {super.viewDidLoad()
        pinchRecognizer.addTarget(self, action:
#selector(tableViewController.handlePinch(recognizer:)))
        tableView.addGestureRecognizer(pinchRecognizer)
        NotificationCenter.default.addObserver(
            self, selector:
            #selector(UIApplicationDelegate.applicationDidEnterBackground(_
:)), name:
            NSNotification.Name.UIApplicationDidEnterBackground, object:
nil)

        do
        {
            self.filmItems = try
[FilmItem].readFromPersistence()
        }
        catch let error as NSError
        {
            if error.domain == NSCocoaErrorDomain && error.code
            == NSFileReadNoSuchFileError
            {
                NSLog("No persistence file found, notnecesserially an
error...")
            }
            else
            {
                let alert = UIAlertController(title: "Error",
                message: "Could not load the list items!", preferredStyle:
.alert)
                alert.addAction(UIAlertAction(title: "OK", style: .default,
handler: nil))
                self.present(alert, animated: true, completion: nil)
                NSLog("Error loading from persistence: \(error)")
            }
        }

        tableView.dataSource = self tableView.delegate = self
        tableView.register(FilmTableViewCell.self,

```

```

        forCellReuseIdentifier: "cell") tableView.separatorStyle =
.none tableView.backgroundColor = UIColor.black
        tableView.rowHeight = 50if filmItems.count > 0 {
            return
        }
        filmItems.append(FilmItem(text: "The Godfather"))
        filmItems.append(FilmItem(text: "The Dark Knight"))
        filmItems.append(FilmItem(text: "The Crow"))
        filmItems.append(FilmItem(text: "The Shawshank Redemption"))
        filmItems.append(FilmItem(text: "The Lord of the Rings: The
Returnof the King"))
        filmItems.append(FilmItem(text: "Fight Club"))
    }
    @objc
    public func applicationDidEnterBackground(_ notification:
NSNotification)
    {
        do
        {
            try filmItems.writeToPersistence()
        }
        catch let error
        {
            NSLog("Error writing to persistence: \(error)")
        }
    }
    func numberOfSectionsInTableView(tableView: UITableView) -> Int
    {
        return 1
    }
    func tableView(_ tableView: UITableView, numberOfRowsInSection
section: Int) -> Int {
        return filmItems.count
    }
    func tableView(_ tableView: UITableView,
cellForRowAt indexPath: IndexPath) ->
UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier:
"cell", for: indexPath as IndexPath) as! FilmViewCell
        cell.selectionStyle = .none cell.textLabel?.backgroundColor =
UIColor.clearlet item = filmItems[indexPath.row]
        // cell.textLabel?.text = item.textcell.delegate
= self
        cell.filmItem = itemreturn cell
    }
    func cellDidBeginEditing(editingCell: FilmViewCell) { let
editingOffset = tableView.contentOffset.y -
editingCell.frame.origin.y as CGFloat
        let visibleCells = tableView.visibleCells as! [FilmViewCell]
        for cell in visibleCells {
            UIView.animate(withDuration: 0.3, animations: {() in
cell.transform = CGAffineTransform(translationX: 0,
y: editingOffset)

```

```

        if cell != editingCell {
            cell.alpha = 0.3
        }
    })
}
}
func cellDidEndEditing(editingCell: FilmViewCell) {
    let visibleCells = tableView.visibleCells as! [FilmViewCell]
    for cell: FilmViewCell in visibleCells {
        UIView.animate(withDuration: 0.3, animations: {() in
            cell.transform = CGAffineTransform.identity
            if cell != editingCell {cell.alpha = 1.0
            }
        })
    }
    if editingCell.filmItem!.text == "" {
        FilmItemDeleted(filmItem: editingCell.filmItem!)
    }
    func FilmItemDeleted(filmItem: FilmItem) {var index = 0
        for i in 0..

```

```

        tableView.endUpdates()
    }
    func colorForIndex(index: Int) -> UIColor {let itemCount =
filmItems.count - 1
    let val = (CGFloat(index) / CGFloat(itemCount)) * 0.6 return
UIColor(red: 1.0, green: val, blue: 0.0, alpha: 1.0)
    }
    let kRowHeight: CGFloat = 50.0
    func tableView(_ tableView: UITableView,
heightForRowAt indexPath: IndexPath) -> CGFloat {return
kRowHeight
    }
    func tableView(_ tableView: UITableView, willDisplay cell:
UITableViewCell,
forRowAt indexPath: IndexPath) {
cell.backgroundColor = colorForIndex(index: indexPath.row)
    }
    struct TouchPoints {
var upper: CGPointvar lower: CGPoint
    }
    var upperCellIndex = -100var lowerCellIndex = -100
    var initialTouchPoints: TouchPoints!
    var pinchExceededRequiredDistance = falsevar pinchInProgress =
false
    func handlePinch(recognizer: UIPinchGestureRecognizer) {if
recognizer.state == .began {
    pinchStarted(recognizer: recognizer)
    }
    if recognizer.state == .changed&& pinchInProgress
    && recognizer.numberOfTouches == 2 {pinchChanged(recognizer:
recognizer)
    }
    if recognizer.state == .ended { pinchEnded(recognizer:
recognizer)
    }
    }
    func pinchStarted(recognizer: UIPinchGestureRecognizer) {
initialTouchPoints = getNormalizedTouchPoints(recognizer:
recognizer)
    upperCellIndex = -100
    lowerCellIndex = -100
    let visibleCells = tableView.visibleCells as![FilmViewCell]
    for i in 0..

```

```

= true
    let precedingCell = visibleCells[upperCellIndex]
placeholderCell.frame = precedingCell.frame.offsetBy(dx:
    0.0, dy: kRowHeight / 2.0)
placeholderCell.backgroundColor = precedingCell.backgroundColor
tableView.insertSubview(placeholderCell, at: 0)
}
}
func pinchChanged(recognizer: UIPinchGestureRecognizer) {let
currentTouchPoints =
    getNormalizedTouchPoints(recognizer: recognizer)
    let upperDelta = currentTouchPoints.upper.y -
initialTouchPoints.upper.y
    let lowerDelta = initialTouchPoints.lower.y -
currentTouchPoints.lower.y
    let delta = -min(0, min(upperDelta, lowerDelta))
    let visibleCells = tableView.visibleCells as! [FilmViewCell]
for i in 0..

```

```

[FilmTableViewCell
for cell in visibleCells {
cell.transform = CGAffineTransform.identity
}
}, completion: nil)
}
}

func getNormalizedTouchPoints(recognizer: UIGestureRecognizer)
->TouchPoints {
    var pointOne = recognizer.location(ofTouch: 0, in:tableView)
    var pointTwo = recognizer.location(ofTouch: 1, in:tableView)
    if pointOne.y > pointTwo.y {let temp = pointOne pointOne =
pointTwo pointTwo = temp
    }
    return TouchPoints(upper: pointOne, lower: pointTwo)
}

func viewContainsPoint(view: UIView, point: CGPoint) -> Bool {
let frame = view.frame
    return (frame.origin.y < point.y) && (frame.origin.y +
(frame.size.height) > point.y)
}

let placeholderCell = FilmTableViewCell(style: .default,
reuseIdentifier: "cell")
var pullDownInProgress = false
func scrollViewWillBeginDragging(_ scrollView: UIScrollView) {

    pullDownInProgress = scrollView.contentOffset.y <= 0.0
placeholderCell.backgroundColor = UIColor.red
    if pullDownInProgress {
tableView.insertSubview(placeholderCell, at: 0)
    }
    func scrollViewDidScroll(_ scrollView: UIScrollView) {
        let scrollViewContentOffsetY = tableView.contentOffset.y
        if pullDownInProgress && scrollView.contentOffset.y <= 0.0 {
placeholderCell.frame = CGRect(x: 0, y: -
tableView.rowHeight,
width: tableView.frame.size.width, height: tableView.rowHeight)
placeholderCell.label.text = -scrollViewContentOffsetY >
tableView.rowHeight ?
        "Release to add item" : "Pull to add item"
placeholderCell.alpha = min(1.0, -
scrollViewContentOffsetY / tableView.rowHeight)
        } else {
pullDownInProgress = scrollViewContentOffsetY <= 0.0 ?true :
false
        }
    }
    func scrollViewDidEndDragging(_ scrollView: UIScrollView,
willDecelerate decelerate: Bool) {
        if pullDownInProgress && -scrollView.contentOffset.y >
tableView.rowHeight {

```

```

        FilmItemAdded()
    }
    pullDownInProgress = false
placeholderCell.removeFromSuperview()
    }
    func FilmItemAdded() {
        FilmItemAddedAtIndex(index: 0)
    }
    func FilmItemAddedAtIndex(index: Int) { let filmItem =
FilmItem(text: "")
        filmItems.insert(filmItem, at: index) tableView.reloadData()
        var editCell: FilmViewCell
        let visibleCells = tableView.visibleCells as! [FilmViewCell]
for cell in visibleCells {
            if (cell.filmItem === filmItem) { editCell = cell
editCell.label.becomeFirstResponder() break
            }
        }
    }
}

```

**Лістинг A.2 – Визначення зовнішнього вигляду клітинка користувача.
Код файлу TableViewCell.swift**

```

import UIKit
class TableViewCell: UITableViewCell { @IBOutlet weak var
imageView: UIImageView! @IBOutlet weak var titleLabel: UILabel!
@IBOutlet weak var yearLabel: UILabel!
    override func awakeFromNib() {super.awakeFromNib()
    }

    override func setSelected(_ selected: Bool, animated: Bool) {
super.setSelected(selected, animated: animated)
    }
}

```

ДОДАТОК Б

Відображення елементів у списку фільмів

Лістинг Б.1 - Код файлу FilmViewCell.swift

```
protocol TableViewCellDelegate {
    func FilmItemDeleted(filmItem: FilmItem)
    func cellDidBeginEditing(editingCell: FilmViewCell) func
    cellDidEndEditing(editingCell: FilmViewCell)
}
class FilmViewCell: UITableViewController, UITextFieldDelegate {let
gradientLayer = CAGradientLayer()
    var originalCenter = CGPoint()
    var deleteOnDragRelease = false, completeOnDragRelease = false
var tickLabel: UILabel, crossLabel: UILabel
    let label: StrikeThroughText
    var itemCompleteLayer = CALayer() var delegate:
TableViewCellDelegate?var filmItem: FilmItem? {
    didSet {
        label.text = filmItem!.text label.strikeThrough =
filmItem!.completed
        itemCompleteLayer.isHidden = !label.strikeThrough
    }
}
    required init?(coder aDecoder: NSCoder) {fatalError("NSCoding
not supported")}
    override init(style: UITableViewCellStyle,
reuseIdentifier: String?) {
        label = StrikeThroughText(frame: CGRect.null)label.textColor =
UIColor.white
        label.font = UIFont.boldSystemFont(ofSize: 16)
label.backgroundColor = UIColor.clear
        func createCueLabel() -> UILabel {
            let label = UILabel(frame: CGRect.null)label.textColor =
UIColor.white
            label.font = UIFont.boldSystemFont(ofSize: 32.0)
label.backgroundColor = UIColor.clear
            return label
        }
        tickLabel = createCueLabel() tickLabel.text = "\u{2713}"
tickLabel.textAlignment = .rightcrossLabel = createCueLabel()
crossLabel.text = "\u{2717}" crossLabel.textAlignment = .left
        super.init(style: style, reuseIdentifier: reuseIdentifier)
label.delegate = self
        label.contentVerticalAlignment = .center
        addSubview(label)
        addSubview(tickLabel)
        addSubview(crossLabel) selectionStyle = .none
    }
}
```

```

gradientLayer.frame = bounds
    let color1 = UIColor(white: 1.0, alpha: 0.2).cgColor as
    let color2 = UIColor(white: 1.0, alpha: 0.1).cgColor as let
color3 = UIColor.clear.cgColor as CGColor
    let color4 = UIColor(white: 0.0, alpha: 0.1).cgColor as
    gradientLayer.colors = [color1, color2, color3, color4]
gradientLayer.locations = [0.0, 0.01, 0.95, 1.0]
    layer.insertSublayer(gradientLayer, at: 0) itemCompleteLayer =
CALayer(layer: layer)
    itemCompleteLayer.backgroundColor = UIColor(red: 0.0, green:
0.6, blue: 0.0, alpha: 1.0).cgColor
    itemCompleteLayer.isHidden = true
layer.insertSublayer(itemCompleteLayer, at: 0)
    let recognizer = UIPanGestureRecognizer(target: self, action:
#selector(FilmViewCell.handlePan(recognizer:)))
    recognizer.delegate = self addGestureRecognizer(recognizer)
}
    let kLabelLeftMargin: CGFloat = 15.0
    let kUICuesMargin: CGFloat = 10.0, kUICuesWidth: CGFloat =
50.0 override func layoutSubviews() {
    super.layoutSubviews() gradientLayer.frame = bounds
itemCompleteLayer.frame = bounds
    label.frame = CGRect(x: kLabelLeftMargin, y: 0,
width: bounds.size.width - kLabelLeftMargin, height:
bounds.size.height)
    tickLabel.frame = CGRect(x: -kUICuesWidth - kUICuesMargin
y: 0,
width: kUICuesWidth, height:
bounds.size.height)
    crossLabel.frame = CGRect(x: bounds.size.width + kUICuesMargin,
y: 0,
width: kUICuesWidth, height:
bounds.size.height)
}
    func handlePan(recognizer: UIPanGestureRecognizer) {
// 1
    if recognizer.state == .began { originalCenter = center
    }
// 2
    if recognizer.state == .changed {
    let translation = recognizer.translation(in: self) center =
CGPoint(x: originalCenter.x + translation.x, y:
originalCenter.y)
    deleteOnDragRelease = frame.origin.x < -frame.size.width / 2.0
    completeOnDragRelease = frame.origin.x >
frame.size.width / 2.0
    let cueAlpha = fabs(frame.origin.x) / (frame.size.width / 2.0)
    tickLabel.alpha = cueAlpha crossLabel.alpha = cueAlpha
    tickLabel.textColor = completeOnDragRelease ?
UIColor.green : UIColor.white
    crossLabel.textColor = deleteOnDragRelease ? UIColor.red
: UIColor.white
    }
}

```

```

// 3
if recognizer.state == .ended {
    let originalFrame = CGRect(x: 0, y: frame.origin.y,
    width: bounds.size.width,
    height: bounds.size.height)
    if deleteOnDragRelease {
        if delegate != nil && filmItem != nil {
            delegate!.FilmItemDeleted(filmItem: filmItem!)
        }
    } else if completeOnDragRelease {if filmItem != nil {
        filmItem!.completed = true
    }
    label.strikeThrough = true itemCompleteLayer.isHidden = false
        UIView.animate(withDuration: 0.2, animations:
            {self.frame =
                originalFrame})
    } else {
        UIView.animate(withDuration: 0.2, animations:
        {self.frame = originalFrame})
    }
    }
    override func gestureRecognizerShouldBegin(_
gestureRecognizer: UIGestureRecognizer) -> Bool {
    if let panGestureRecognizer = gestureRecognizer as?
    UIPanGestureRecognizer {
        let translation = panGestureRecognizer.translation(in:
        superview!)
        if fabs(translation.x) > fabs(translation.y) {
            return true
        }
        return false
    }
    return false
    }
    func textFieldShouldReturn(_ textField: UITextField) -> Bool {
        textField.resignFirstResponder()
        return false
    }
    func textFieldShouldBeginEditing(_ textField: UITextField) ->
Bool
    {
        if filmItem != nil {
            return !filmItem!.completed
        }
        return false
    }
    func textFieldDidEndEditing(_ textField: UITextField) {if
filmItem != nil {
        filmItem!.text = textField.text!
    }
    if delegate != nil { delegate!.cellDidEndEditing(editingCell:

```

```
self)
    }
    }
    func textFieldDidBeginEditing(_ textField: UITextField) {if
delegate != nil {
    delegate!.cellDidBeginEditing(editingCell: self)
    }
    }

    }
```

ДОДАТОК В

Відображення тексту з перекресленим форматуванням

Лістинг В.1 - Код файлу StrikeThrougText.swift

```
import UIKit
import QuartzCore

class StrikeThroughText: UITextField {let strikeThroughLayer:
CALayer var strikeThrough : Bool {
    didSet {
        strikeThroughLayer.isHidden = !strikeThroughif strikeThrough {
            resizeStrikeThrough()
        }
    }
}
    required init?(coder aDecoder: NSCoder) {fatalError("NSCoding
not supported")}
    override init(frame: CGRect) { strikeThroughLayer = CALayer()
        strikeThroughLayer.backgroundColor = UIColor.white.cgColor
        strikeThroughLayer.isHidden = true
        strikeThrough = false

        super.init(frame: frame) layer.addSublayer(strikeThroughLayer)
    }
    override func layoutSubviews() {super.layoutSubviews()
        resizeStrikeThrough()
    }
    let kStrikeOutThickness: CGFloat = 2.0func
resizeStrikeThrough() {
    let textSize = text!.size(attributes:
[NSFontAttributeName:font!])
    strikeThroughLayer.frame = CGRect(x: 0, y:
bounds.size.height/2,
        width: textSize.width,
        height: kStrikeOutThickness)
    }
}
```

ДОДАТОК Г

Налаштування вигляду елементів інтерфейсу

Лістинг Г.1 - Модель даних про фільм. Код файлу FilmItem.swift

```
import Foundationimport UIKit
class FilmItem: NSObject, NSCoding {var text: String
var completed: Bool
public init(text: String) {self.text = text self.completed =
false
}
required init?(coder aDecoder: NSCoder)
{
if let text = aDecoder.decodeObject(forKey: "text") as?

{
self.text = text
}
else
{
return nil
}
if aDecoder.containsValue(forKey: "completed")
{
self.completed = aDecoder.decodeBool(forKey:
"completed")
}
else
{
return nil
}
}

func encode(with aCoder: NSCoder)
{
aCoder.encode(self.text, forKey: "text")
aCoder.encode(self.completed, forKey: "completed")
}
}
extension FilmItem
{
public class func getMockData() -> [FilmItem]
{
return [
FilmItem(text: ""),
FilmItem(text: ""),
FilmItem(text: ""),FilmItem(text: "")
]
}
```

```

    }
    extension Collection where Iterator.Element == FilmItem
    {
        private static func persistencePath() -> URL?
        {
            let url = try? FileManager.default.url(for:
                .applicationSupportDirectory, in: .userDomainMask, appropriateFor:
                nil,
                create: true)
            return url?.appendingPathComponent("filmitems.bin")
        }
        func writeToPersistence() throws
        {
            if let url = Self.persistencePath(), let array = self as?
                NSArray
            {
                let data =
                    NSKeyedArchiver.archivedData(withRootObject: array)
                try data.write(to: url)
            }
            else
            {
                throw NSError(domain: "ru.wylde.FilmList", code: 10,
                    userInfo: nil)
            }
        }
        static func readFromPersistence() throws -> [FilmItem]
        {
            if let url = persistencePath(), let data = (try
                Data(contentsOf: url) as Data?)
            {
                if let array = NSKeyedUnarchiver.unarchiveObject(with: data)
                    as? [FilmItem]
                {
                    return array
                }
            }
            else
            {
                throw NSError(domain: "ru.wylde.FilmList", code:
                    11, userInfo: nil)
            }
        }
        else
        {
            throw NSError(domain: "ru.wylde.FilmList", code: 12,
                userInfo: nil)
        }
    }
}

```

Лістинг Г.2 - Представлення елементу ТОП-списку. Код файлу TopList.swift

```
import UIKit
import Foundation
class TopList:
UIViewController,UITableViewDataSource,UITableViewDelegate {
final let urlString = "https://api.myjson.com/bins/128p7x"

@IBOutlet weak var tableView: UITableView!var titleArray =
[String]()
var yearArray = [String]() var imgURLArray = [String]() var
ReleaseArray = [String]()var PlotArray = [String]() var urlIMDBArray
= [String]()var RatingArray = [String]() var RatedArray = [String]()
var VotesArray = [String]() var TypeArray = [String]()

override func viewDidLoad() {super.viewDidLoad() self.title =
"Top movies"

self.navigationController?.navigationBar.titleTextAttributes? =
[NSForegroundColorAttributeName: UIColor.white]
self.downloadJsonWithTask()
}
func downloadJsonWithTask() {
let url = NSURL(string: urlString)
if Reachability.isConnectedToNetwork() == true
{
print("Internet Connection Available!")
var downloadTask = URLRequest(url: (url as URL?)!,cachePolicy:
URLRequest.CachePolicy.returnCacheDataElseLoad, timeoutInterval: 60)
downloadTask.httpMethod = "GET"
NSURLSession.shared.dataTask(with: downloadTask,
completionHandler: {(data, response, error) -> Void in
if let jsonData = try?
JSONSerialization.jsonObject(with: data!, options:
.allowFragments)as? NSDictionary
print(jsonData!.value(forKey: "movies")!) if let FilmsArray =
jsonData!.value(forKey:
"movies") as? NSArray {
for actor in FilmsArray{
if let FilmsDict = actor as?
NSDictionary {
if let name =
FilmsDict.value(forKey: "title") {
self.titleArray.append(name
as! String)
}

if let name =
FilmsDict.value(forKey: "year") {
self.yearArray.append(name
as! String)
}
if let name =
```

```

FilmsDict.value(forKey: "urlPoster") {

as! String)
}
self.imgURLArray.append(name
if let name =FilmsDict.value(forKey: "releaseDate") {

        self.ReleaseArray.append(name as! String)
    }
    if let name = FilmsDict.value(forKey: "plot") {
self.PlotArray.append(name
as! String)

FilmsDict.value(forKey: "urlIMDB") {
}
if let name =
self.urlIMDBArray.append(name as! String)
}
if let name =
FilmsDict.value(forKey: "rating") {

as! String)
}
self.RatingArray.append(name
if let name =
FilmsDict.value(forKey: "rated") {
self.RatedArray.append(name
as! String)
}
if let name =
FilmsDict.value(forKey: "votes") {
self.VotesArray.append(name
as! String)
}
if let name =
FilmsDict.value(forKey: "type") {
self.TypeArray.append(name
as! String)
}
}
}
}
}
OperationQueue.main.addOperation({self.tableView.reloadData()
                                })
                                })
}).resume()
}else {
    print("Internet Connection not Available!")
    let alert = UIAlertController(title: "Warning",
                                message: "No
internet connection",
preferredStyle: .alert)

```

```

        alert.addAction(UIAlertAction(title: "Ok",
        style: .default))

    }}

    present(alert, animated: true)
    func tableView(_ tableView: UITableView, numberOfRowsInSectionSection: Int) -> Int {
        return titleArray.count
    }
    func tableView(_ tableView: UITableView, cellForRowAt indexPath: IndexPath) -> UITableViewCell {
        let cell = tableView.dequeueReusableCell(withIdentifier: "cell") as! TableViewCell
        cell.titleLabel.text = titleArray[indexPath.row]
        cell.yearLabel.text = yearArray[indexPath.row]
        let imageURL = NSURL(string: imageURLArray[indexPath.row]) if
        imageURL != nil {
            let data = NSData(contentsOf: (imageURL as? URL)!)
            cell.imageView.image = UIImage(data: data as! Data)
        }
        return cell
    }
    func tableView(_ tableView: UITableView, didSelectRowAt indexPath: IndexPath) {
        let vc =
        self.storyboard?.instantiateViewController(withIdentifier:
        "DetailViewController") as! DetailViewController
        vc.imageString = imageURLArray[indexPath.row] vc.titleString =
        titleArray[indexPath.row] vc.yearString = yearArray[indexPath.row]
        vc.plotString = PlotArray[indexPath.row] vc.releaseDateString =
        ReleaseArray[indexPath.row] vc.ratingString =
        RatingArray[indexPath.row] vc.ratedString =
        RatedArray[indexPath.row] vc.votesString = VotesArray[indexPath.row]
        vc.urlIMDBString = urlIMDBArray[indexPath.row] vc.typeString =
        TypeArray[indexPath.row]

        self.navigationController?.pushViewController(vc, animated:
        true)
        tableView.deselectRow(at: indexPath, animated: true)}

```

ДОДАТОК Д

Відображення детальної інформації про об'єкти

Лістинг Д.1 - Код файлу DetailViewController.swift

```
import UIKit

class DetailViewController: UIViewController {

    @IBOutlet weak var imageView: UIImageView! @IBOutlet weak var
    titleLabel: UILabel! @IBOutlet weak var yearLabel: UILabel!
    @IBOutlet weak var releaseDateLabel: UILabel! @IBOutlet weak var
    plotView: UITextView! @IBOutlet weak var urlIMDBLabel: UIButton!
    @IBOutlet weak var ratingLabel: UILabel! @IBOutlet weak var
    ratedLabel: UILabel! @IBOutlet weak var votesLabel: UILabel!
    @IBOutlet weak var typeLabel: UILabel!

    @IBAction func urlIMDBButtonPressed(sender: UIButton) {
        UIApplication.shared.openURL(NSURL(string: urlIMDBString)!
            as URL)
    }

    @IBAction func PosterTapped(sender: UITapGestureRecognizer){
        let imageView = sender.view as! UIImageView
        let newImageView = UIImageView(image: imageView.image)
        newImageView.frame = UIScreen.main.bounds
        newImageView.backgroundColor = .black newImageView.contentMode =
        .scaleAspectFit newImageView.isUserInteractionEnabled = true
        let tap = UITapGestureRecognizer(target: self, action:
        #selector(dismissFullscreenImage))
        newImageView.addGestureRecognizer(tap)
        self.view.addSubview(newImageView)
        self.navigationController?.isNavigationBarHidden = true
        self.tabBarController?.tabBar.isHidden = true
    }

    func dismissFullscreenImage(_ sender: UITapGestureRecognizer)
    {self.navigationController?.isNavigationBarHidden = false
    self.tabBarController?.tabBar.isHidden = false
    sender.view?.removeFromSuperview()
    }

    var titleString:String! var yearString:String! var
    imageString:String!
    var releaseDateString:String! var plotString:String!
    var urlIMDBString:String! var ratingString:String! var
    ratedString:String! var votesString:String! var typeString:String!

    override func viewDidLoad() {super.viewDidLoad()
    self.updateUI()
    }

    override func didReceiveMemoryWarning() {
    super.didReceiveMemoryWarningWarning()
    }
}
```

```

func updateUI() {
    self.titleLabel.text = titleStringself.yearLabel.text =
yearString self.plotView.text = plotString
    releaseDateString.insert(".", at:
releaseDateString.index(releaseDateString.startIndex, offsetBy: +4))
    releaseDateString.insert(".", at:
releaseDateString.index(releaseDateString.startIndex, offsetBy: +7))
    self.releaseDateLabel.text = "Released: \(releaseDateString!)"
    self.urlIMDBLabel.setTitle("IMDB Link", for:
UIControlState.normal)
    self.ratingLabel.text = "Rating: \(ratingString!)"
    self.ratedLabel.text = "Rated: \(ratedString!)" self.votesLabel.text
= "Votes: \(votesString!)" self.typeLabel.text = "Type:
\(typeString!)"

    let imgURL = URL(string:imageString) let data =
NSData(contentsOf: (imgURL!))
    self.imageView.image = UIImage(data: data as! Data)
}
func urlIMDBButtonPressed() {

}
override func viewDidLoad(_ animated: Bool) {
plotView.scrollToRangeToVisible(NSMakeRange(0,0))
}
}

```

ДОДАТОК Ж

Налаштування навігації у мобільному додатку

Лістинг Ж.1 - Навігаційний контролер. Код файлу

NavigationControllerFilms.swift

```
import Foundation
import UIKit

class NavigationControllerFilms: UINavigationController,
    UINavigationControllerTransitioningDelegate {

    override func viewDidLoad() {super.viewDidLoad()}

    self.navigationController.navigationBar.barStyle = UIBarStyle.black
    self.navigationController.navigationBar.tintColor = UIColor.white
}
}
```

Лістинг Ж.2 - Управління картою у мобільному додатку. Код файлу MapViewController.swift

```
import UIKit
import MapKit
import CoreLocation

class MapViewController: UIViewController, MKMapViewDelegate,
    CLLocationManagerDelegate {

    @IBOutlet weak var Map: MKMapView!
    var coordinates: [Any]!
    var names: [String]!
    var addresses: [String]!
    var phones: [String]!
    let locationManager = CLLocationManager()
    var regionRadius: CLLocationDistance = 5000

    override func viewWillAppear(_ animated: Bool) {
        locationManager.requestWhenInUseAuthorization()
    }

    func locationManager(CLLocationManager) {
        if CLLocationManager.authorizationStatus()
            == .authorizedWhenInUse {
            Map.showsUserLocation = true
        } else {
            locationManager.requestWhenInUseAuthorization()
        }
    }

    override func viewDidLoad() {
        super.viewDidLoad()
        Map.delegate = self
        locationManager.delegate = self
        let MapCenter =
            CLLocationCoordinate2DMake(49.85096864, 24.03086744)
        let region = MKCoordinateRegionMakeWithDistance(MapCenter,
            regionRadius * 2, regionRadius * 2)
    }
}
```

```

Map.setRegion(region, animated: true)
// 1
coordinates = [[49.85038,24.02202],[49.84327,24.02895],
[49.83867,24.02712],[49.828059,24.007894],[49.81203,24.00431],
[49.80741,23.978630],[49.79532,24.05779],[49.77376,24.01084]]
//
Latitude,Longitude
names = ["Планета Кіно - Львів Forum","Кінопалац","Кінопалац
«Копернік»","Антикінотеатр Rockfellow","Львівський
кіноцентр","Multiplex Victoria Gardens","Кінопалац ім.
Довженка","Планета Кіно IMAX"]
addresses = ["Під Дубом, 7А","вулиця Театральна, 22","вулиця
Коперника, 9","вулиця Генерала Чупринки","18, вулиця Володимира
Великого, 14А","Кульпарківська, 226А","Проспект ЧервоноїКалини,
81","вул. Стрийська, 30, ТРЦ «King Cross Leopoldis»"]
phones = ["+380-800-300-600","+380-322-975-050","+380 322
975 177","+380 68 404 0350","+380 322 644 353","+380 322 593
300","+380 3222 16131","+380 800 300 600"]
self.Map.delegate = self
// 2
for i in 0...7
{
let coordinate:[Double] = coordinates[i] as! [Double]let point
= CustomAnnotation(coordinate:
CLLocationCoordinate2D(latitude: coordinate[0], longitude:
coordinate[1]))
point.image = UIImage(named: "image-\(i+1).jpg")point.name =
names[i]
point.address = addresses[i] point.phone = phones[i]
self.Map.addAnnotation(point)
}
}
@IBAction func type_satellite(sender: UIButton) {Map.mapType =
MKMapType.satellite
}
@IBAction func type_standard(sender: UIButton) {Map.mapType =
MKMapType.standard
}
@IBAction func getLocationButton(sender: UIButton) {
Map.setCenter(Map.userLocation.coordinate, animated: true)
}
@IBAction func zoomIn(sender: UIButton) {
let span = MKCoordinateSpan(latitudeDelta:
Map.region.span.latitudeDelta/2, longitudeDelta:
Map.region.span.longitudeDelta/2)

let region = MKCoordinateRegion(center: Map.region.center,
span: span)
Map.setRegion(region, animated: true)
}
@IBAction func zoomOut(sender: UIButton) { let span =
MKCoordinateSpan(latitudeDelta:
Map.region.span.latitudeDelta*2, longitudeDelta:

```

```

Map.region.span.longitudeDelta*2)
    let region = MKCoordinateRegion(center: Map.region.center,
span: span)
    Map.setRegion(region, animated: true)
    }
    func mapView(_ mapView: MKMapView, viewFor annotation:
MKAnnotation) -> MKAnnotationView? {
        if annotation is MKUserLocation
        {
            return nil
        }
        var annotationView =
self.Map.dequeueReusableAnnotationView(withIdentifier: "Pin")
        if annotationView == nil{
            annotationView = MKAnnotationView(annotation:annotation,
reuseIdentifier: "Pin")
            annotationView?.canShowCallout = false
        }else{
            annotationView?.annotation = annotation
        }
        annotationView?.image = UIImage(named: "map_pin")return
annotationView
    }
    func mapView(_ mapView: MKMapView,
didSelect view: MKAnnotationView)
    {
        // 1
        if view.annotation is MKUserLocation
        {
            return
        }
        // 2
        let CalloutAnnotation = view.annotation as! CustomAnnotation
let views = Bundle.main.loadNibNamed("CustomCalloutView",
owner: nil, options: nil)
        let calloutView = views![0] as! CustomCalloutView
calloutView.CalloutName.text = CalloutAnnotation.name
calloutView.CalloutAddress.text = CalloutAnnotation.address
calloutView.CalloutPhone.text = CalloutAnnotation.phone
calloutView.CalloutImage.image = CalloutAnnotation.image
        // 3
        calloutView.center = CGPoint(x: view.bounds.size.width / 2,y:
-calloutView.bounds.size.height*0.52)
        view.addSubview(calloutView)
    }

    func mapView(_ mapView: MKMapView, didDeselect view:
MKAnnotationView) {
        if view.isKind(of: MKAnnotationView.self)
        {
            for subview in view.subviews
            {
                subview.removeFromSuperview() }
        }
    }

```

ДОДАТОК К

Встановлення розміру, колірів, тіней та інших атрибутів кнопки

Лістинг К.1 - Код файлу CircleButton.swift

```
import Foundationimport UIKit

class CircleButton: UIButton {

    required init?(coder aDecoder: NSCoder) {super.init(coder:
aDecoder)
        //layer.borderWidth = 1.0 layer.borderColor =
tintColor.cgColorlayer.cornerRadius = 15.0 clipsToBounds = true
        contentEdgeInsets = UIEdgeInsets(top: 8, left: 8, bottom: 8,
right: 8)
        setTitleColor(tintColor, for: .normal)
setTitleColor(UIColor.white, for: .highlighted)
    }
}
```

ДОДАТОК Л

Управління мережевою взаємодією додатку

Лістинг Л.1- Код файлу Networking.swift

```
import SystemConfiguration
public class Reachability {
    class func isConnectedToNetwork() -> Bool {

        var zeroAddress = sockaddr_in(sin_len: 0, sin_family: 0,
sin_port: 0, sin_addr: in_addr(s_addr: 0), sin_zero: (0, 0, 0, 0, 0,
0, 0, 0))
        zeroAddress.sin_len = UInt8(MemoryLayout.size(ofValue:
zeroAddress))
        zeroAddress.sin_family = sa_family_t(AF_INET)

        let defaultRouteReachability = withUnsafePointer(to:
&zeroAddress) {
            $0.withMemoryRebound(to: sockaddr.self, capacity: 1)
            {zeroSockAddress in
                SCNetworkReachabilityCreateWithAddress(nil,
zeroSockAddress)
            }
        }
        var flags: SCNetworkReachabilityFlags =
SCNetworkReachabilityFlags(rawValue: 0)
        if SCNetworkReachabilityGetFlags(defaultRouteReachability!,
&flags) == false {
            return false
        }
        let isReachable = (flags.rawValue &
UInt32(kSCNetworkFlagsReachable)) != 0
        let needsConnection = (flags.rawValue &
UInt32(kSCNetworkFlagsConnectionRequired)) != 0
        let ret = (isReachable && !needsConnection)

        return ret
    }
}
```

ДОДАТОК М

Інтерфейс створеного додатку

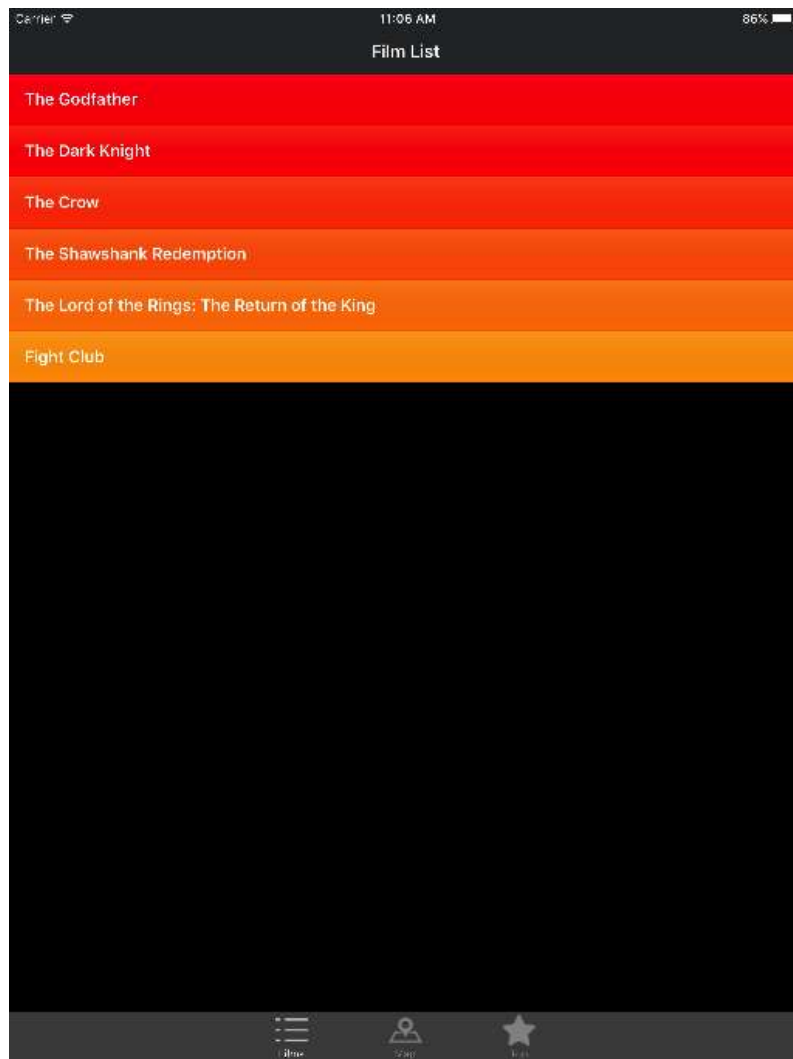


Рисунок М.1 - Скріншот iPad – версії додатку

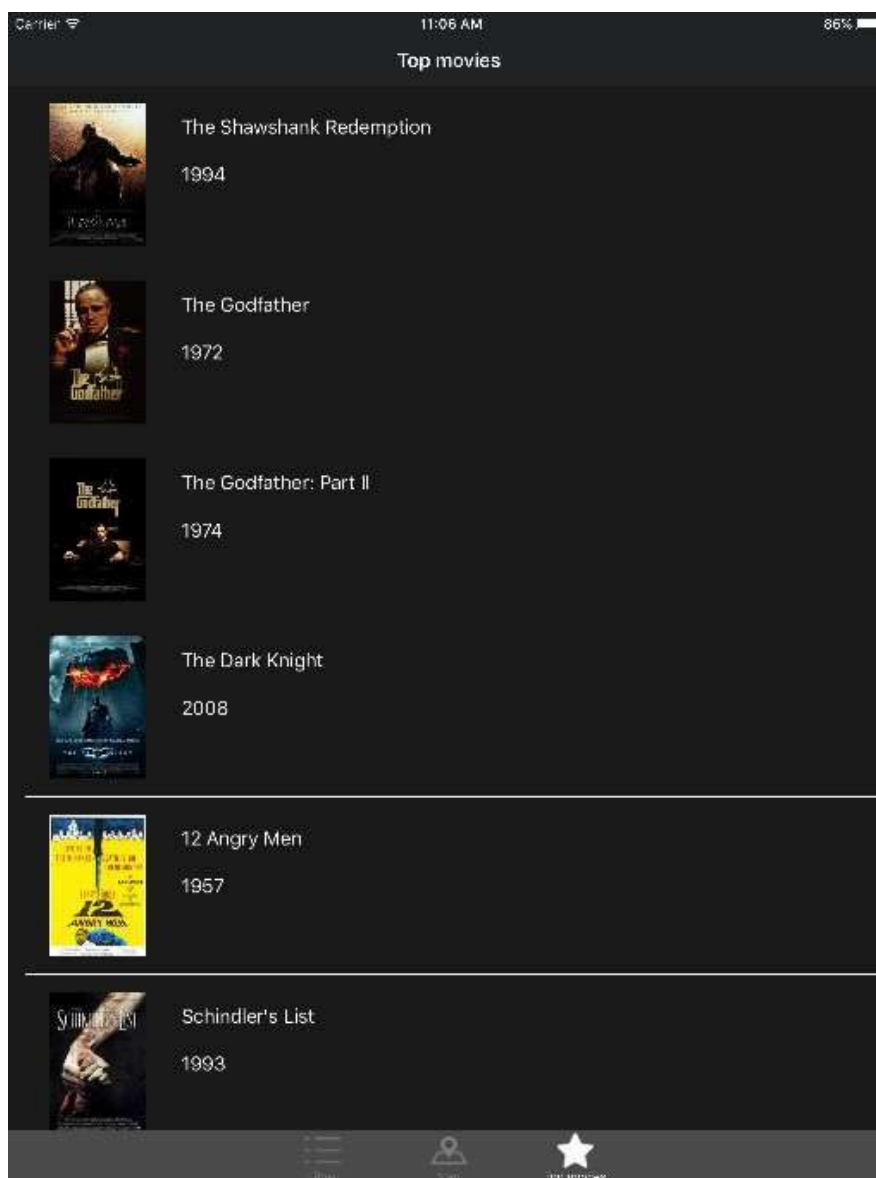


Рисунок М.2 - Скріншот iPad – ТОП

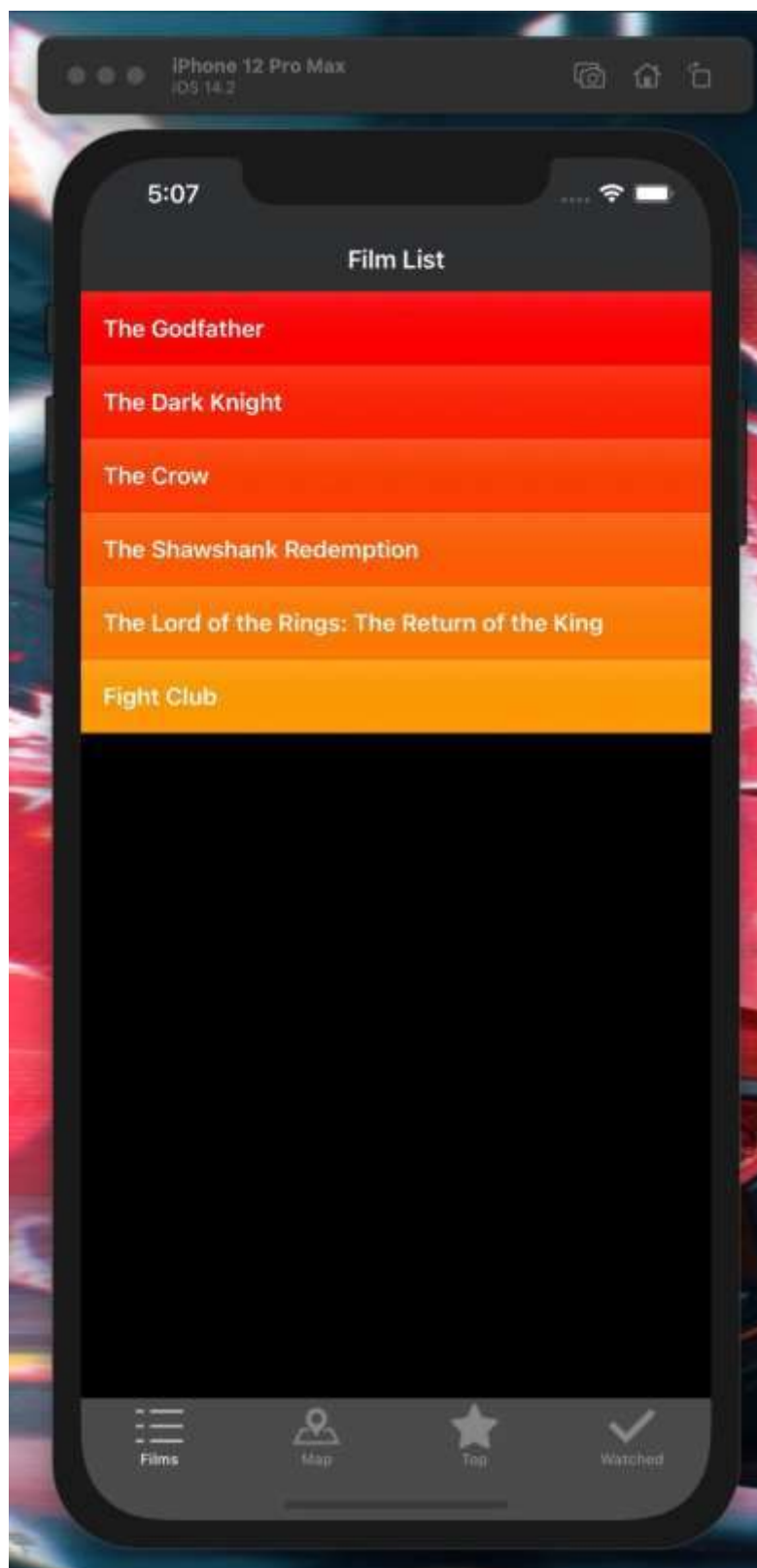
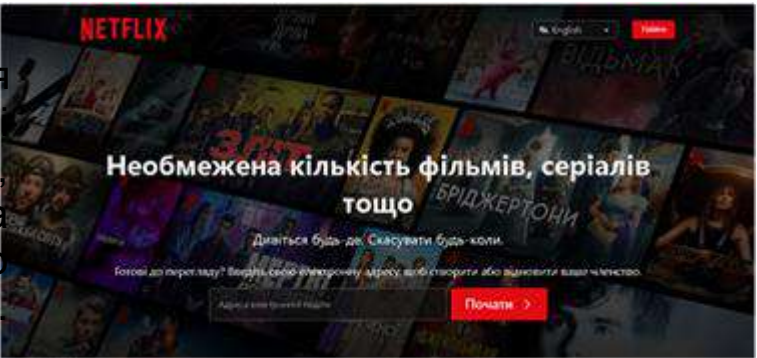


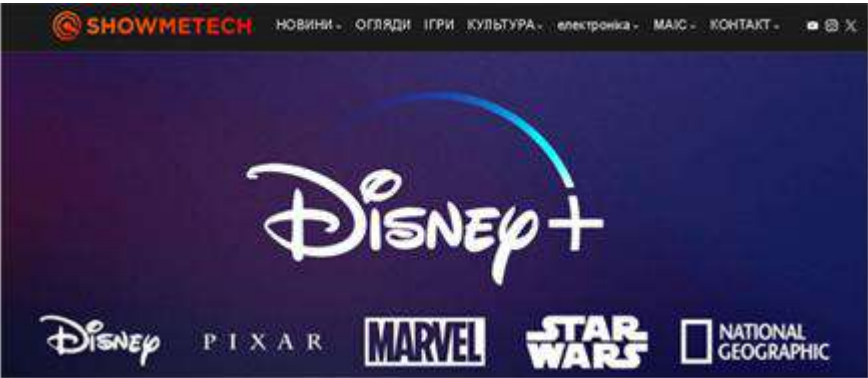
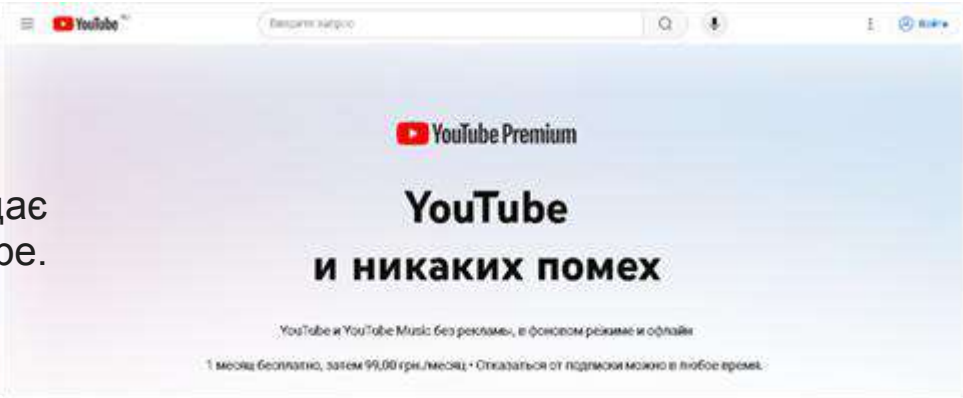
Рисунок М.3 – Скріншот Film List

Популярні відеосервіси для відпочинку

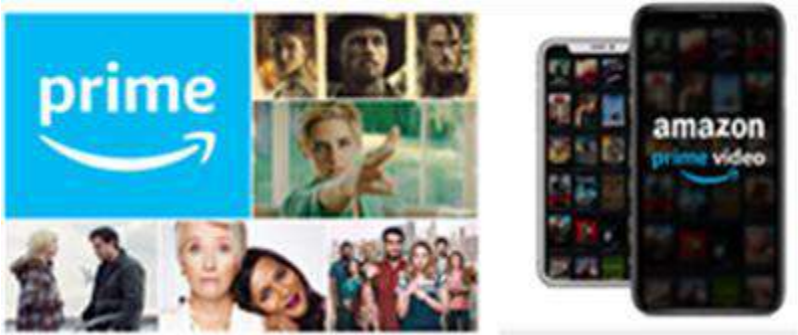
Netflix — це сервіс потокового передавання відео, що дає змогу дивитися різноманітні відзначені нагородами телесеріали, фільми, аніме, документалістику й інший контент на тисячах пристроїв із підключенням до Інтернету



YouTube Premium – це платна підписка, яка дає користувачам додаткові переваги на YouTube.



Disney+ — це дім потокового передавання ваших улюблених історій. Транслює останні випуски фільмів, ексклюзивні оригінальні серіали та каталог класичних фільмів із додаванням нових історій щотижня.



Amazon Prime сервіс «відео за запитом», пропонує телевізійні шоу та фільми в оренду або для купівлі як частину підписки Amazon Prime, деякі відео доступні для перегляду виключно повним членам Prime або Prime Video.



Hulu — американський потоковий VOD-сервіс, що повністю контролюється і належить The Walt Disney Company.

					13.02070849.00052 ПЛ1							
					Розробка мобільного застосунку для кіноманів	Лім.			Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата								
Розробив	Червонюк Д.І.											
Перевішив	Скрупський С.Ю.											
Т.контр.						Аркуш 1			Аркушів 1			
Н.контр.	Польська О.В.				Популярні відеосервіси для відпочинку	НУ «Запорізька політехніка», гр. КНТ-520						
Затвердив	Кудерметов Р.К.											

Популярні музичні сервіси для відпочинку



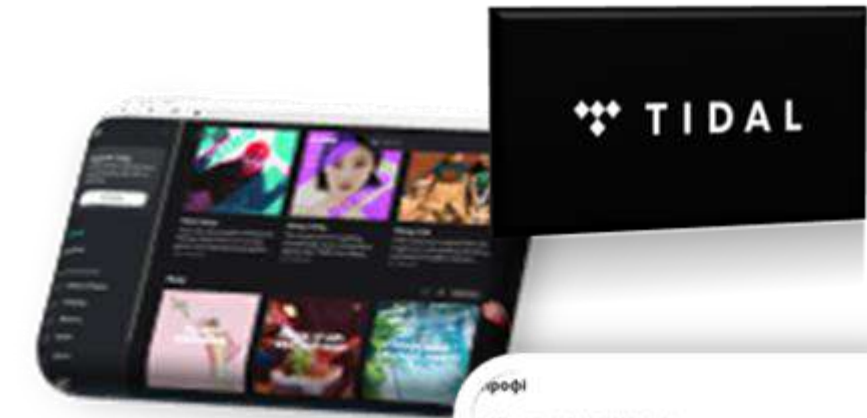
- профі**
- Велика бібліотека пісень і подкастів
 - Інтеграція з Alexa
 - Доступне цюктуєрення
 - Параметри потокового передавання у високій роздільній здатності
- мінуси**
- Обмежені соціальні можливості
 - Обмежені можливості персоналізації



- профі**
- Велика бібліотека пісень, музичних відео та живих виступів
 - Індивідуальні рекомендації на основі звичок слухачів
 - Гарна інтеграція з YouTube
- мінуси**
- Обмежені можливості високоякісного аудіо
 - Обмежені соціальні можливості



- профі**
- Велика бібліотека пісень і подкастів
 - Персоналізовані списки відтворення та рекомендації на основі алгоритму
 - Соціальні функції для обміну музикою
 - Доступне цюктуєрення
 - Хороша інтеграція з пристроями та розумними колонками
- мінуси**
- Обмежені можливості високоякісного аудіо
 - Обмежені можливості подкастів



- профі**
- Високі параметри аудіо
 - Ексклюзивний контент, наприклад живі виступи та музичні відео
 - Персоналізовані списки відтворення та рекомендації
 - Дружній до користувача інтерфейс
- мінуси**
- Обмежена бібліотека порівняно з іншими сервісами
 - Дорожче за інші послуги



- профі**
- Велика бібліотека пісень, подкастів і музичних відео
 - Персоналізовані плейлисти та радіостанції
 - Дружній до користувача інтерфейс
 - Високоякісний звук
 - Хороша інтеграція з пристроями Apple
- мінуси**
- Обмежена безкоштовна версія
 - Може бути дорожчим за інші послуги

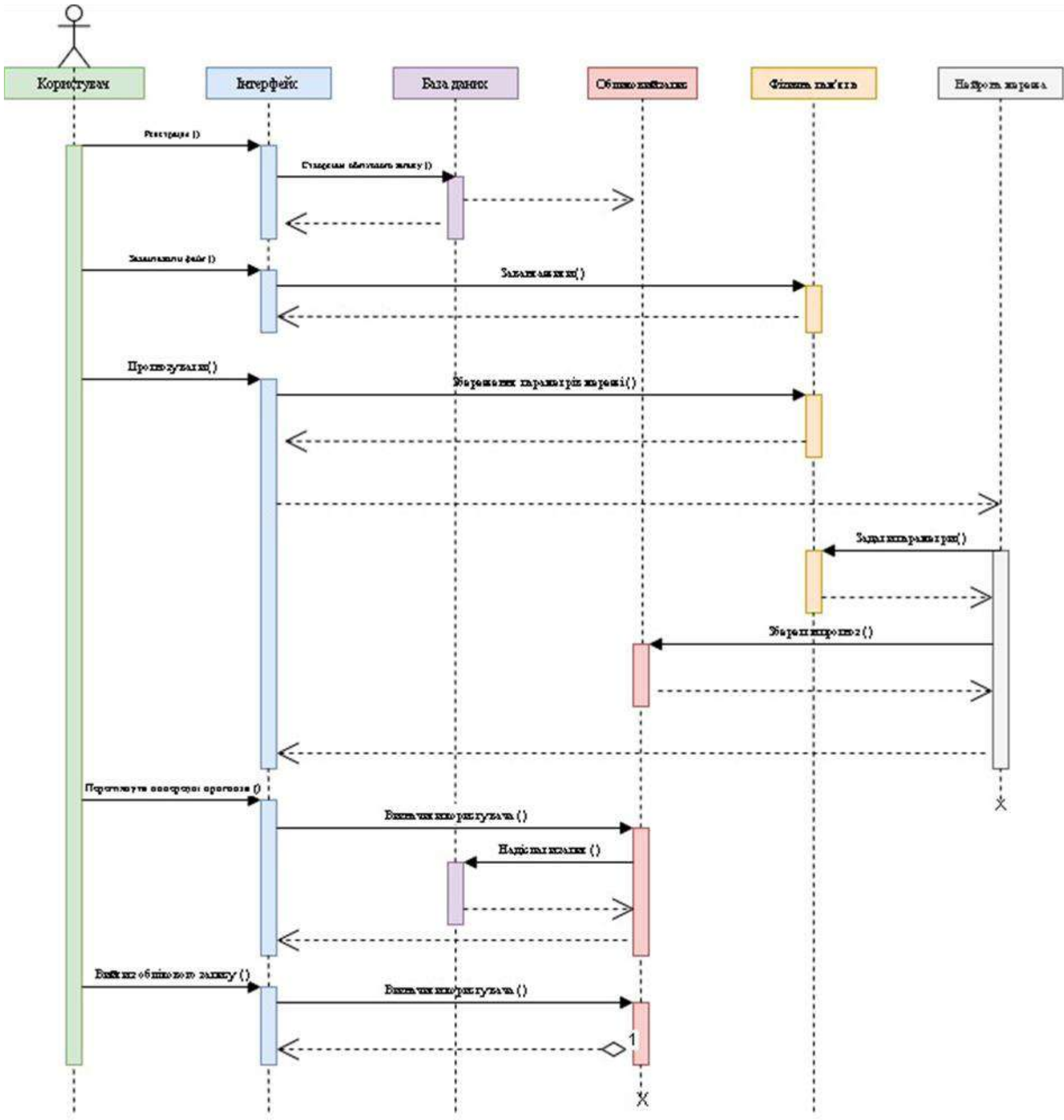
					13.02070849.00052 ПЛІ2				
					Розробка мобільного застосунку для кіноманів				
Зм.	Арк.	№ документа	Підпис	Дата	Найпопулярніші музичні сервіси				
Розробив	Червонюк Д.І.								
Перевірів	Скрупський С.Ю.								
Т.контр.					НУ «Запорізька політехніка», гр. КНТ-510				
Н.контр.	Польська О.В.								
Затвердив	Кудерметов Р.І.								

Схема структури розробки додатку



					13.02070849.00052 ПЛЗ											
					Розробка мобільного застосунку для кіноманів						Літ.		Маса		Масштаб	
Зм.	Арк.	№ документа	Підпис	Дата												
Розробив		Червонюк Д.І.														
Перевірив		Скруський С.Ю.														
Т.контр.					Схема структури розробки додатку						Аркуш 1		Аркушів 1			
Н.контр.		Тольська О.В.									НУ «Запорізька політехніка», гр. КНТ-520					
Затвердив		Кудерметов Р.К.														

Діаграма послідовності для мобільного додатку



					13.02070849.00052 ПЛ4			
Зм.	Арк.	№ документа	Підпис	Дата	Розробка мобільного застосунку для кіноманів	Літ.	Маса	Масштаб
Розробив		Червонюк Д.І.						
Перевірив		Скруський С.Ю.						
Т.контр.						Аркуш 1	Аркушів 1	
Н.контр.		Польська О.В.			Діаграма послідовності для мобільного додатку	НУ «Запорізька політехніка», гр. КНТ-520		
Затвердив		Кудерметов Р.Г.						