Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

Магістр
(назва освітнього ступеня)

на тему: Організація мультидисциплінарної платформи для організації будинків.

Виконав(ла): студент(ка) 6 курсу, групи СІм-61
спеціальності
123 „Комп'ютерна інженерія"
(шифр і назва спеціальності)

(підпис) Ураксузов Ю.В.
(прізвище та ініціали)

Керівник (підпис) Лупин В.В.
(прізвище та ініціали)

Нормоконтроль (підпис) Лупин В.С.
(прізвище та ініціали)

Завідувач кафедри (підпис) (прізвище та ініціали)

Рецензент (підпис) Козак Р.О.
(прізвище та ініціали)

Тернопіль
2020

Міністерство освіти і науки України

Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних систем та мереж
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (прізвище та ініціали)

« » 20___ р.

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Магістр
(назва освітнього ступеня)

за спеціальністю 123 "Комп'ютерна інженерія"
(шифр і назва спеціальності)

студенту Кравцову Юрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема роботи Організація мультифункціональної платформи для планування будинку

Керівник роботи к.т.н. доцент Бушинський Віталій Володимирович
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від «28» квітня 2020 року № 417-68Г

2. Термін подання студентом завершеної роботи _____

3. Вихідні дані до роботи Мобільний застосунок для мобільної платформи iOS на мові Swift для планування будинку.

4. Зміст роботи (перелік питань, які потрібно розробити)
Аналіз сучасного стану дослідження в області організації будинку. Методи побудови рекомендаційних сервісів для організації будинку. Організація мультифункціональної платформи для планування будинку. Охорона праці та безпека життєдіяльності. Висновки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1 Тема та актуальність роботи. 2. Мета і задачі. 3 Принципи роботи цифрового будинку. 3. Методи дослідження найбільших до поширення загроз будинку. 4 Карта технологій безпеки цифрового будинку. 4. Побудова системи ТОП-250. 5. Діаграма блоків систем управління 6. Середовище розробки коду на мові Swift, 7. Висновки

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека надзвичайних ситуацій	Клепінк В.М. ст. викладач		22.10.20
Охорона праці	Скуратівська Г.М.		

7. Дата видачі завдання _____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Огляд та аналіз існуючих рішень та досвідчень у сфері організації будинку	29.09.2020 – 10.10.2020	виконано
2.	Дослідження методів побудови рекомендаційних сервісів для організації будинку	12.10.2020 – 24.10.2020	виконано
3.	Розробка, впровадження та публікація застосунку.	26.10.2020 – 19.11.2020	виконано
4.	Охорона праці в надзвичайних ситуаціях	21.11.2020 – 04.12.2020	виконано
5.	Оформлення пояснювальної записки	04.12.2020 – 06.12.2020	виконано
6.	Оформлення графічного матеріалу	07.12.2020 – 11.12.2020	виконано
7.	Попередній захист кваліфікаційної роботи магістра	13.12.2020	виконано
8.	Захист дипломної роботи магістра		

Студент _____ Цимбалюк Ю.В.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Дунець Х.Б.
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Організація мультифункціональної платформи для планування відпочинку // Кваліфікаційна робота магістра // Крамаров Юрій Володимирович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп'ютерно-інформаційних систем та програмної інженерії, група СІМ-61 // Тернопіль, 2020 // с. – 78, рис. – 39, табл. – 3, аркушів А1 – __, додат. – 22, бібліогр. – 20.

Ключові слова: планування відпочинку, мобільний застосунок, ранжування.

Мета роботи полягає у дослідженні методів та розробка мобільного застосунку для планування відпочинку, дослідження мобільних застосунків, теорії їх розробки, інструментів для розробки мобільних застосунків, створення макету застосунку та вибір операційної системи і розроблення для неї застосунку.

У дипломній роботі було розроблено мобільний застосунок для пошуку та вибору фільмів, а також найближчих кінотеатрів. Проведено аналіз 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-250 фільмів з сайту IMDB та відображає на карті найближчі до користувача кінотеатри.

ANNOTATION

Organization of a multifunctional platform for recreation planning // Qualification work of the master // Kramarov Yuriy // Ternopil Ivan Pulyuy National Technical University, Faculty of Computer Information Systems and Software Engineering, CIM-61 Group // Ternopil, 2020 // p. - 78, fig. - 39, table. - 3, sheets of A1 - __, add. - 22, bibliogr. - 20.

Key words: recreation planning, mobile application, ranging.

The aim of the work is to examine the methods and develop the mobile applications for recreation planning, to research the mobile applications, theories of their development, tools for mobile application development, to create an application layout and select an operating system, developing an application for it.

The diploma thesis deals with a mobile application for searching and selecting movies, as well as the nearest cinemas. The analysis of features of development of mobile applications was carried out. Comparison with existing applications was made. The functionality of the system was detailed with the help of various diagrams. A working version of the system was built and tested, which creates a custom list, shows the TOP-250 movies from the IMDB site and displays the nearest cinemas on the map.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

ООП – об’єктно-орієнтоване програмування;

IDE – інтегроване середовище розробки;

OLE – зв’язування та впровадження об’єктів;

VCL (Visual Component Library) – візуальна бібліотека компонентів;

NET – програмна технологія, запропонована фірмою Microsoft, як платформа для створення, як звичайних програм, так і веб-застосунків;

ПК – персональний комп’ютер;

PDP–1 (Programmed Data Processor-1) – перший комп'ютер із серії PDP, вироблений Digital Equipment Corporation в 1960 році;

XML – запропонований консорціумом World Wide Web (W3C) стандарт побудови мов розмітки ієрархічно структурованих даних для обміну між різними застосунками, зокрема, через Інтернет;

SQL – декларативна мова програ мування для взаємодії користувача з базами даних, що застосовується для формування запитів, оновлення і керування реляційними базами даних;

MSDE – файл-серверна система управління версіями;

UML – діаграма, що відображає об'єкти та їх зв’язки в певний момент часу;

WCF – ім’я платформи наступного покоління, раніше відомої під кодом Indigo, для коннекційних систем з використанням розширення API .NET, WCF робить можливою побудову безпечних, надійних і транзакційних систем через спрощену уніфіковану програмну модель міжплатформної взаємодії;

UI (user interface) – призначений для користувача інтерфейс. Це певні елементи сайту або програми (оформлені графічно і органічно вписані в дизайн), що дозволяють користувачеві взаємодіяти з цим самим сайтом;

ВСТУП

Актуальність теми полягає у тому, що на сьогоднішній день сучасні смартфони, яких з кожним днем стає все більше використовуються у всіх сферах людської діяльності для обробки інформації і обсяг цієї інформації стає все більшим.

Навіть оргіназація відпочинку з друзями потребує обробки інформації і проблема створення різних засобів для спрощення поставленої задачі завжди була популярною серед користувачів.

Зараз мобільні пристрої, які завжди знаходяться під рукою проникли в усі сфери людського життя. Смартфон став незамінним помічником. Області застосування мобільних пристроїв постійно розширюються і захоплюють такі сфери людського життя в яких, як здавалося, обчислення не потрібні.

Спрощення задач – це одне з завдань сучасних смартфонів. Сьогодні все більшого значення має прискорення вирішення певних задач, що зекономить час та ресурси користувачів застосунку.

Мета і завдання дослідження - це дослідження методів та розробка мобільного застосунку для планування відпочинку, дослідження мобільних застосунків, теорії їх розробки, інструментів для розробки мобільних застосунків, створення макету застосунку та вибір операційної системи і розроблення для неї застосунку.

Для досягнення вказаної мети в роботі поставлено наступні задачі:

- аналіз наукових праць і практик реалізації і засобів ранжування контенту;
- аналіз принципів організації платформ для планування відпочинку;
- обґрунтування вибору ОС та вимог до платформи;
- розробка алгоритмів роботи модулів;
- Розробка мультифункціональної платформи для планування відпочинку;

Об’єктом дослідження є розробка мобільних застосунків, які допомагають орагнізовувати відпочинок та принципи їх створення.

Предмет дослідження. Вивчення літератури пов'язаної з розробкою мобільних застосунків. Розробка зручного для користувача програмного продукту,

побудованого відповідно до специфікацій, на мові високого рівня, в якому буде простий та зручний сервіс для справжніх кіноманів, в якому присутній короткий опис фільмів, щоб швидко зорієнтуватися у виборі, користуватись яким можна на вулиці, вдома, в кафе чи у транспорті.

Методи дослідження є середовище візуального програмування Xcode, мова програмування Swift 4.0, робота з Apple MapKit та IMDb API.

Для проведення даної роботи використовувались методи аналізу, збору інформації, обробки і аналізу даних, порівняння, аналогії, методи програмування, зокрема методи об’єктно-орієнтованого програмування.

Наукова новизна одержаних результатів.

- Уперше запропоновано архітектурне рішення з використанням методу інтелектуального ранжування місць відпочинку, що дало змогу на запит користувача формувати більш точні відповіді в порівнянні з іншими методами.
- Набув подальшого розвитку метод і засіб ранжування відеоконтенту та найближчих місць відпочинку, що дало змогу підвищити рівень якості надання послуг кінцевому користувачу.

Практичне значення одержаних результатів полягає тому, що розроблену систему можна буде використовуватись у планування відпочинку для користувачів, зручний та зрозумілий інтерфейс та доступність безкоштовних оновлень для користувачів мобільних пристроїв.

Публікації – “Інтегроване середовище розробки Apple Xcode”. Створення власної карти за допомогою Apple MapKit”.

Структура роботи. Робота складається з пояснювальної записки та графічної частини. Пояснювальна записка складається з вступу, чотирьох розділів, висновків, списку використаних джерел та додатку. Обсяг роботи: пояснювальна записка - __ аркушів формату А4, графічна частина -

РОЗДІЛ 1
АНАЛІЗ СУЧАСНОГО ЕТАПУ ДОСЛІДЖЕННЯ В ОБЛАСТІ ОРГАНІЗАЦІЇ ВІДПОЧИНКУ

1.1. Аналіз принципів організації платформ для планування відпочинку

Складно уявити сучасну людину без мобільного пристрою в руках. Будь-то телефон, смартфон або інші комунікатори – ці пристрої міцно і, судячи з усього, надовго, увійшли в суспільство, як незамінні повсякденні засоби зв'язку. З появою нових мобільних пристроїв і різноманітністю платформ, на яких вони розробляються (Android, Symbian, Bada, iOS, iPhone), активно розвивається бізнес з розробки мобільних додатків різного призначення [16].

Мобільний додаток являє собою програму, встановлену на тій чи іншій платформі, що володіє певним функціоналом, що дозволяє виконувати різні дії. Розробляються ці програми на мові програмування високого рівня та компілюються в код пристрою. Мобільні додатки зайняли свою нішу в багатьох сферах діяльності людини і сьогодні використовуються не тільки для розваг (як це переважно було раніше), але і для ведення бізнесу та проведення різних рекламних кампаній. Сучасні технології дозволяють максимально адаптувати програми під різні мобільні пристрої і зробити простими для сприйняття людиною.

Програми, що позиціонують себе як мобільні додатки, створюються, виходячи з потреб споживачів. Однакових додатків не буває, адже не буває користувачів з однаковими потребами та вимогами. Загалом можна виділити кілька видів мобільних додатків залежно від їх спрямованості і виконуваних функцій. Отже, виділяють[9]:

– промо – додатки для мобільних пристроїв. Такі програми, як правило, обмежені в своїх функціях, але є дуже креативними і популярними в наш час. Прикладом такого мобільного додатку може стати віртуальна запальничка від Zippo або додаток Magic Coke Bottle, який представляє собою пляшку “Coca-Cola”, що передбачає майбутнє. Такі додатки найчастіше використовуються як частина рекламної кампанії різних брендів.

– програми-події – це такі програми, що розробляються для трансляції тих чи інших подій. Наприклад, для перегляду Олімпіади онлайн.

– програми-служби є своєрідними аналогами сайтів. Такі мобільні додатки можуть розроблятися у вигляді каталогів, списків і т.д., що відображають діяльність тієї чи іншої організації.

– програми-ігри – це мобільні додатки, які найчастіше створюються для розваг, а також навчання дітей в ігровій формі. Ігри розробляються для найрізноманітніших платформ (Android, Symbian, Bada, iOS і ін.), їх кількість сьогодні настільки велика, що не піддається рахунку.

– Інтернет-магазини. Такі мобільні додатки створюються для здійснення онлайн-покупок одягу, аксесуарів, прикрас і т.д. набувають більшої популярності, тому що значно полегшують процес вибору та придбання необхідної речі.

– мобільні додатки для бізнесу – програми, що створюються як для спілкування та оптимізації роботи всередині організації (корпоративні додатки), так і для взаємодії з користувачами.

– інші види мобільних додатків: контентні додатки, додатки-соціальні мережі, системні програми і т.д.

Не можна дати чітку класифікацію мобільних додатків, адже деякі з них мають настільки багатий функціонал, що є своєрідним “міксом” різних видів додатків. Як правило, такі додатки, що пропонують користувачам широкі можливості, є платними. Безкоштовні ж додатки найчастіше представляють собою просте програмне забезпечення (ПЗ) з обмеженим набором можливостей, наприклад, для перегляду електронної пошти. Хоча є і винятки.

Сучасні мобільні додатки давно стали потужним маркетинговим інструментом, який дозволяє вирішувати безліч завдань: створювати імідж, підтримувати бренд і підвищувати лояльність до нього з боку споживачів, оптимізувати процеси комунікації, створювати певний інформаційний простір.

На сьогоднішній день існує багато платформ та сервісів для організації відпочинку. Серед них онлайн-кінотеатри та стрімінгові сервіси: YouTube, Spotify, Apple Music, Megogo, Netflix, YouTube Music, SoundCloud та інші.

Деякі з них пропонують свої послуги безкоштовно, але переважна більшість надає обмежений перелік функцій та дають можливість користувачеві розширити функціонал шляхом платної підписки на сервіс.

Також за допомогою деяких сервісів користувач має можливість ділитись з іншими користувачами своїм контентом (SoundCloud, Youtube, Spotify, Apple Music), інші ж пропонують тільки конкретний перелік ліцензованого контенту (Netflix, Megogo, Hulu).

Потокове (стрімінгове) мультимедіа – це мультимедіа, яке користувач отримує безпосередньо від провайдера потокового мовлення (радіо, телебачення).

Починаючи з 1980 років домашні комп'ютери, які були доступні користувачам вже могли відображати різні типи мультимедіа, проте мережі залишалися обмеженими, тому потокове медіа поступалося традиційному – CD-диски, касети, книги.

В період кінця. 1990 – початку. 2000 років користувачі отримали відносно високу пропускну здатність мережі, зросла загальна кількість користувачів інтернету, з'явилися стандартизовані протоколи та формати (TCP/IP, HTML, HTTP), в інтернеті почала з'являтись комерція. Саме в той час почали з'являтись перші сервіси потокового медіа.

Так як витрати на передачу та зберігання інформації завжди великі, а в основному мультимедіа завжди займає великі об'єми, тому в більшості випадків при передачі використовують стискання інформації в цілях збільшення швидкості та зменшення витрат на передачу.

Потокове мультимедіа буває двох типів: за запитом та живе. Мультимедіа за запитом зберігається на серверах тривалий період часу (YouTube, Spotify, Megogo, Netflix). Живе мультимедіа доступне короткий відрізок часу, наприклад, при передачі відео зі спортивних змагань (Twitch, YouTube, Mixer та інші).

1.1.1. Потокове мовлення і зберігання інформації. Необхідний розмір для зберігання інформації потокових мультимедіа обчислюється в залежності від швидкості переданої інформації а тривалості інформації за такою формулою (для одного користувача та одного файлу).

Розмір сховища = тривалість (секунд) * бітрейт (кбіт/с)/(8*1024)

Для прикладу одна година відео, що має розмір 320x240 пікселів, закодованого зі швидкістю 300кбіт/с буде займати:

(3600с * 300кбіт/с)/(8*1024) = 131.83 Мб місця на диску.

Щоб розрахувати необхідну пропускну здатність для сервера потокового медіа з режимом передачі за запитом для файла, який буде переглядати 1000 людей одночасно за протоколом Unicast (1 клієнт – 1 з'єднання), то можна використовувати таку формулу:

300кбіт/с * 1000 = 300.000кбіт/с = 300Мбіт/с мережевого інтерфейсу.

Це еквівалент близько 125Гб інформації на годину. Якщо використовується протокол Multicast, то навантаження на сервер буде набагато нижче, так як для передачі інформації всім клієнтам використовується єдиний потік, який буде займати всього. 300кбіт/с мережевого інтерфейсу сервера.

Протоколи Multicast розроблені для зменшення навантаження. З. серверів на підключення при перегляді потокового медіа великою кількістю клієнтів. Ці протоколи даних надсилають один потік даних цілій групі клієнтів. Проте один з недоліків такого протоколу є неможливість реалізувати функцію відео за запитом. Також безперервне потокове мовлення інформації робить неможливим управління відтворенням користувачем, але ця проблема вирішується впровадженням у мережу передачі даних кешуючих серверів.

1.2. 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 [11].

Розробка мобільних додатків складається з повного циклу, який включає в себе шість основних етапів. Жоден з них не можна обділити увагою і знехтувати, оскільки розробка мобільних додатків – вартісна річ, тому перш, ніж вкладати в нього час і кошти, слід чітко визначитися, чого саме потребує користувач. І перше, що потрібно зробити для розробки мобільного застосування – це вибрати тип програми та операційну систему для його реалізації.

Основні етапи розробки мобільних додатків наведено нижче [14].

Першим етапом є 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. Тому розробка детального макету – фундаментальна основа, яка перетворює мобільні додатки в маленькі шедеври. Блок-схема додатку наведена у додатку А.

До другого етапу відноситься створення прототипу і проектування структури UI/UX. Щоб зрозуміти, як користувач буде використовувати ваш додаток, створюється спеціальна карта взаємодії між екранами. На цьому ж етапі опрацьовується весь функціонал програми. Фактично, виконуються всі функції, прописані в технічному завданні: визначається, як буде працювати додаток, які кнопки і функціонал будуть розміщені на кожному екрані. Також важливо продумати навігацію – як користувач буде переходити на кожен з екранів.

Третій етап – це 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 кожну 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. Концепт програми показано у додатку Х.

Четвертим етапом є розробка самої програми. Розробка мобільних додатків на цьому етапі полягає в тому, що з'єднуються воєдино 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.

П'ятим етапом розробки є тестування додатку. Навіть у кращому програмному коді можуть виявитися незначні помилки, адже розробка мобільних додатків – тривалий процес. Отже, завдання дизайн-студії на цьому етапі їх виявити і усунути. 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.

Розміщення мобільного додатку в магазині є шостим етапом. Для того, щоб користувачі могли знайти мобільний додаток і скористатися ним, його розміщують у спеціальних магазинах. Наприклад, GooglePay для ОС Android, або AppStore для iOS.

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.

Магазин додатків для смартфонів на основі iOS – має скромнішу статистику, але приносить розробникам більший прибуток.

Більшість з сучасних платформ націлені на постійну взаємодію з користувачем, тому розробники надають перевагу мобільним додаткам, які краще оптимізовані під різне апаратне забезпечення, ніж звичайний сайт. Великим мінусом в їх розробці є необхідність врахувати велику кількість можливих апаратних платформ користувачів.

Також можуть використовувати Веб-додатки – це клієнт-серверний додаток, в якому клієнт взаємодіє з веб-сервером за допомогою браузера. Логіка таких додатків розподілена між сервером і клієнтом, дані зберігаються переважно на сервері, а обмін інформацією проводиться через мережу. Однією з переваг такого підходу є те, що клієнти не залежні від конкретної операційної системи користувача, тому веб-додатки являються універсальними.

Великі платформи при розробці додатків намагаються зробити його доступним якомога більшій кількості користувачів, тому і системні вимоги намагаються зробити якомога меншими.

Наприклад, якщо ви бажаєте дивитись відео через браузер на Youtube, то вам буде потрібна 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.

Мобільний додаток. Youtube останньої версії вимагає Android 5.0 або iOS 11 для установки.

1.3. Порівняння сервісів планування відпочинку

Той час, коли потрібно було платити за конкретний диск з фільмом чи музичним альбомом пройшли. Сучасні потокові сервіси дозволяють дивитись абсолютно все: від новин в прямому ефірі до останніх новинок серіалів чи класичні фільми в будь який момент часу, з будь якого пристрою. Користувач купує підписку на сервіс, а потім може дивитись все, що забажає на телевізорі, комп’ютері, смартфоні та на інших пристроях, які це дозволяють.

Найпопулярніші відео-сервіси:

- Netflix;
- YouTube;
- Hulu;
- Amazon;
- Twich;
- Vevo;

- Megogo;
- Disney+
- Apple TV+

Вибір сервісу залежить від того, що конкретно користувач бажає дивитись. Netflix обирають заради фільмів та серіалів (також знятих безпосередньо самим сервісом), Twitch призначений для геймерів, які транслюють свій ігровий процес в режимі онлайн, YouTube – найпопулярніший відеохостинг в світі, де користувачі можуть завантажувати свої відеозаписи, а також переглядати, оцінювати й. коментувати відеозаписи інших користувачів. На. YouTube представлені фільми, музичнікліпи, трейлери, новини, освітні передачі, відеоблоги та інше.

Музику також сьогодні вже майже ніхто не купує окремими дисками та не скачує цілими альбомами. Користувачам простіше підписатись на стрімінговий онлайн-сервіс та слухати все, що бажає користувач, в будь який час.

Найпопулярніші музичні сервіси:

- Spotify
- Google Play Music
- Apple Music
- SoundCloud
- Deezer

Всі ці сервіси можуть автоматично підбирати плейлист на основі схожих до вподобань користувача пісень, або ж вибрати вже існуючі. Зазвичай такі сервіси можна зв’язати через мобільний додаток з розумними годинниками, навушниками, домашніми музичними системами.

Більшість з наведених сервісів надають безкоштовний пробний період, після якого потрібно буде платити за повну підписку, або ж обмежують функції для користувачів, які не хочуть платити, але хочуть користуватись сервсіом.

Цінова політика сервісів дуже відрязняється, деякі сервіси пропонують декілька різних видів підписки: індивідуальний, сімейний, для двох, студентський. Деякі сервіси поділяють підписки по ціні, де дорожча підписка надає контент кращої якості.

Порівняння ціни підписки на місяць різних сервісів (Ціни вказані в $, так як не всі сервіси перелічені доступні на території України).

Таблиця 1.1

Відео сервіси

Netflix	Apple TV+	Disney+	Amazon Prime	HBO Max	Hulu
Базова –8.99$ Стандартна – 10.99$ Преміум – 12.99$	4.99$	6.99$	12.99$	14.99$	Базова – 5.99$ Преміум – 11.99$

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

Таблиця 1.2

Музичні сервіси

Spotify	Apple Music	YouTube Music	Deezer	SoundCloud
Преміум – 9.99$ Дуо – 12.99$ Сім’я – 14.99$ Студент – 4.99$	9.99$	Преміум – 9.99$ Сім’я- 14.99$	Базовий - безкоштовно Преміум – 4.99$ Сім’я – 6.99$ HiFi – 8.99$	Базовий – безкоштовно Go – 4.99$ Go + - 9.99$

Музичні сервіси зазвичай не дуже сильно відрізняються представленими виконавцями та дуже рідко мають ексклюзиви на відміну від відео сервісів тому, що кожен виконавець намагається охопити якомога більшу аудиторію.

1.4. Висновки до розділу 1

Результати, одержані в даному розділі полягають в наступному:

Стрімінгові сервіси набувають все більшої популярності в сфері планування та організації відпочинку серед молоді. Швидкий розвиток інтернет-технологій в 1990х дав початок потоковим медіа сервісам, які переросли в справжню заміну телебаченню, походу в кінотеатр, вони витіснили покупку окремого контенту на цифрових та фізичних носіях.

Конкуренція на ринку стрімінгових сервісів породжує великий вибір різних варіантів для кінцевого користувача. Сервіси стали відрізнятись не тільки ціною, але й якістю та різноманіттям контенту, який вони надають. Деякі сервіси надають безкоштовний доступ до всієї своєї бібліотеки безкоштовно, але платна підписка забирає рекламні оголошення, дозволяє кешувати контент для перегляду/прослуховування без підключення до мережі, надає музику або відео в кращій якості (вище розрішення, бітрейт).

Всі ці сервіси працюють майже на всіх пристроях, які мають підключення до мережі інтернет та не вимагають дуже потужної апаратної частини для коректної роботи. Більшість з них можуть працювати в браузері комп’ютера чи смартфона, проте не завжди всі функції сервісу будуть доступні в браузері, тому краще використовувати спеціально розроблений для цього додаток.

РОЗДІЛ 2
МЕТОДИ ПОБУДОВИ РЕКОМЕНДАЦІЙНИХ СЕРВІСІВ ДЛЯ ОРГАНІЗАЦІЇ ВІДПОЧИНКУ

2.1. Принципи ранжування контенту

Ранжування – це алгоритми, за якими контент видається та сортується в пошуку при запиті. Існує велика кількість факторів для ранжування, серед яких можна відмітити рейтинг, релевантність, актуальність, дата створення та інші, на основі яких пошукова система формує список. Алгоритми ранжування того чи іншого сервісу змінюються в процесі його функціонування.

Основними зачадами алгоритму ранжування пошуковими системами є видача користувачеві конкретних відповідей на поставлені запитання і побудова оптимальної послідовності тих чи інших результатів пошуку за запитом з застосуванням тих чи інших інструментів на кожному етапі пошуку для того, щоб гарантувати його ефективність.

Алгоритми ранжування змінюються пошуковими системами в процесі еволюції інтернету. Редактору достатньо вказати в метатегах тематику, опис та ключові слова, щоб пошукова машина порахувала ці дані, як відповідні заданій тематиці і почала високо ранжувати його за вказаними ключовими словами. Тоді сервіси стали змінювати алгоритми, для врахування вмісту та інших факторів ранжування.

2.1.1. Ранжування контенту на YouTube. На YouTube на ранжування впливають три основні параметри та ще декілька додаткових. Серед головних критеріїв:

- Релевантність – те, наскільки відео відповідає пошуковому запиту за наповненням та описом;
- Якість – алгоритми враховують експертність, авторитетність і надійність каналів, кількість підписників та переглядів;
- Залученість – те, як активно користувачі реагують на контент;

- А також:
- Персоналізація – відповідність контенту тому, що шукав і дивився користувач.

Авторитетність джерела – якщо це доречно, сервіс вибирає контент із джерел, вартих довіри. Це стосується таких сфер, як новини, політика, музика та наукова інформація. Наприклад, у відео з категорій музики та розваг алгоритми не так звертають увагу на цей параметр.

Кожного разу коли хтось взаємодіє з відео на YouTube (коментарі, вподобання, перегляди) – це сигналізує про те, що людям воно подобається. Таким чином відеоролики піднімаються в загальному рейтингу та стануть відображатись на початку списку при пошуковому запиті.

Важливими є також теги відео, вони призвані допомогти YouTube зрозуміти вміст. Якщо ж користувач почне використовувати велику кількість тегів, не пов’язаних з відео, то YouTube не буде його ранжувати, так як не розуміє про що саме йдеться в ролику.

Також, якщо відео користувача отримає кількість натискань, яка перевищу середні показники, то YouTube дасть йому поштовх в ранжуванні.

Великий опис також допомагає краще ранжувати відео на YouTube, адже завдяки йому сервіс краще розуміє про що йдеться в відео. Оптимальним розміром є 100-200 символів.

2.1.2. Ранжування фільмів. Однією з найпопулярніших є «Система рейтингів Американської кіноасоціації» (англ. MPAA film rating system) – прийнята в США система оцінки фільмів. В залежності від отриманої оцінки глядацька аудиторія може бути обмежена за рахунок виключення з неї дітей та підлітків. Рейтинг Американської кіноасоціації відіграє важливу роль в долі прокату кінокартини. Система введена в дію 1 листопада 1968 року та досі використовується з незначними змінами.

На сьогоднішній день фільм, який виходить в прокат може отримати один з п’яти рейтингів:

Рейтинг G (General audiences) – Стрічка демонструється без обмежень. Даний рейтинг показує, що оцінений фільм не містить нічого, що більшість батьків могли б вважати непридатним для перегляду чи прослуховування маленькими дітьми. Груба лексика в фільмах з рейтингом G використовуватись не може.

Рейтинг PG (Parental guidance suggested) – Дітям рекомендується перегляд з батьками. Деякі матеріали можуть не підходити для дітей. Цей рейтинг показує, що батьки можуть вважати деякі сцени в стрічці непідходящими для дітей. Можуть бути представлені сцени насильства, але в дуже помірних кількостях.

Рейтинг PG-13 (Parents strongly cautioned) – Не рекомендовано до перегляду дітям до 13 років. Такий рейтинг вказує на те, що оцінений фільм може не підходити для дітей. Батьки повинні бути обережними даючи дозвіл дітям переглядати таку стрічку.

Рейтинг R (Restricted) – Лиця, які не досягли віку 17 років допускаються на фільм тільки в присутності одного з батьків або законного представника. Даний рейтинг показує, що оціночна комісія визначила, що деякі матеріали оціненого фільму призначені тільки для дорослих.

Рейтинг NC-17 (No one 17 & Under Admitted) – Лиця 17-річного віку та молодше не допускаються на фільм. Це вказує на те, що оціночна комісія полягає, що на думку більшості батьків фільм явно для дорослих і дітей до 18 років допускати до перегляду не можна.

NC-17

Рейтинг NR (Not Rated) отримують картини, які не отримали рейтинг Американської асоціації, а для фільмів, які вийшли в прокат до введення системи рейтингів – U (Unrated).

2.2. Метод пробудови персональних рекомендацій

Високі оцінки визначають інтереси користувача, а низькі – антиінтереси. Користувачі можуть мати різні інтереси, але однаково оцінювати погані фільми, тому співпадіння двох десяток і одиниць повинно мати різну вагу.

Також низькі оцінки зазвичай ставляться менш обдумано. Наприклад для більшості користувачів різниця між 1 і 4 по десятибальній шкалі незначна, нелегко визначити гірше з поганого, а от різниця між 7 і 10 для користувачів велика.

Дуже часто для обчислення коефіцієнту близькості оцінок серед користувачів використовується коефіцієнт кореляції Пірсона, який розробли Карл Пірсон, Френсіс Еджуорт і Рафаель Уелдон в 90х роках XIX століття. Коефіцієнт кореляціх розраховується за формулою:

$$\mathbf{r}_{XY} = \frac{\mathbf{cov}_{XY}}{\sigma_X \sigma_Y} = \frac{\sum(X - \bar{X})(Y - \bar{Y})}{\sqrt{\sum(X - \bar{X})^2 \sum(Y - \bar{Y})^2}}.$$

Де $\overline{X} = \frac{1}{n}\sum_{t=1}^{n} X_t, \overline{Y} = \frac{1}{n}\sum_{t=1}^{n} Y_t$ - середнє значення вибірок.

Коефіцієнт кореляції вимірюється в межах від мінус одиниці до плюс одицині.

Також окремою неочевидною проблемою є співвідношення шкали з балами та з відсотками. Причина в психології, адже якщо взяти для 10-бальної шкали по 10% на кожен бал, то фільм, який отримав 8/10 отримає 80% рейтинг, що гірше сприймається користувачем, тому підбираються спеціальні функції, які переводять передбачення оцінки з балів в відсотки так, щоб при цьому зберігались очікування користувачів.

При складанні персональних рекомендацій потрібно також враховувати, що користувачі в більшості випадків не оцінюють кожен фільм, який переглянули, а тільки невелику частину з них. Також високі оцінки отримують далеко не всі фільми, яким користувач надає перевагу. Найвищі оцінки зазвичай отримують рідкісні кіношедеври чи класика, але це не заважає людям з задоволенням переглядати чергові комедії, бойовики чи мелодрами. А це сильно впливає на рекомендації, які користувач отримає в кінцевому результаті. Сервіс буде рекомендувати стрічку рекомендацій, яка буде виглядати логічно, користувачі навіть признають її якість, але дивитись нічого не стануть.

Саме тому великі сервіси вкладають кошти в розробку нових алгоритмів побудови персональних рекомендацій, які створять необхідний баланс між двома крайнощами та будуть брати до уваги потенціал для перегляду та сприйняття рекомендації користувачем.

Ранжувати всю базу фільмів недоцільно, тому на першому кроці відбираються декілька сотень кандидатів, тобто алгоритм знаходить фільми, які могли б бути цікаві глядачеві. Сюди потрапляють як популярні картини, так і близькі до користувача за певними критеріями.

На другому кроці визначаються фактори за фільмами, користувачем та в парі користувач/фільм та ранжуються кандидати за допомогою машинного навчання.

2.3. Метод знаходження найближчих до користувача локацій відпочинку

В сучасному світі існує необхідність в використанні картографічних сервісів, які не тільки можуть вказати шлях до конкретної точки на карті, але й знайти потрібне користувачеві місце, або надати перелік найближчих закладів за певними критеріями. Користувачі також можуть відслідковувати погоду, пожежі, затори, ремонтні роботи в реальному часі.

Розглянемо для прикладу карти Google.

Google Maps – це безкоштовний картографічний сервіс від компанії Google, а також набір різних застосунків, побудованих на базі цього сервісу.

Сервіс являє собою карту та супутникові знімки всього світу та надає користувачам аналіз трафіку в реальному часі, панорамний перегляд вулиці і прокладання маршруту автомобілем, велосипедом, пішки або громадським транспортом.

Користувачі можуть додавати відсутні на картах позначки (парки, сквери, маленькі вулички), які упустили розробники. Також підприємці можуть додавати на карту свої кафе, кінотеатри, торгові центри, ресторани, магазини та інші види бізнесу й відразу вказати контакти та графік, щоб користувачам було простіше їх знайти.

Google збирає списки підприємств з кількох онлайн та офлайн джерел. Щоб запобігти дублюванню в індексі, алгоритм Google автоматично об’єднує списки на основі адреси, номера телефону або геокоду. Google також залучає волонтерів для перевірки та виправлення даних в разі, якщо інформація для різних підприємств відображається як інформація одного підприємства, в результаті чого списки містять помилки.

Такий підхід дуже спрощує життя користувачам, адже вони можуть за лічені хвилини отримати перелік найближчих закладів до їхнього місця перебування, переглянути графік роботи, фотографії та відгуки від інших користувачів за запитом.

Користувачі отримують список закладів відсортованих на основі рейтингу, відгуків та заповнення інформації про організацію, що може сильно впливати на відвідуваність закладу.

Також існує Google Maps API для впровадження карт в свої додатки та веб-сторінки, витягування даних з Google Maps або отримання та передавання геолокації користувача (наприклад сервісу таксі чи доставки).

Для визначення найближчих закладів зазвичай використовується метод k-найближчих сусідів – це простий метричний алгоритм для автоматичної класифікації об’єктів. Основним принципом даного методу є те, що об’єкт присвоюється тому класу, який є найбільш поширеним серед сусідів даного елемента. Сусіди беруться виходячи з множини об’єктів, класи яких уже відомі, кожен об’єкт має кінцеву кількість атрибутів. З ключового для даного методу значення k, вираховується клас, який є найчисленнішим серед інших.

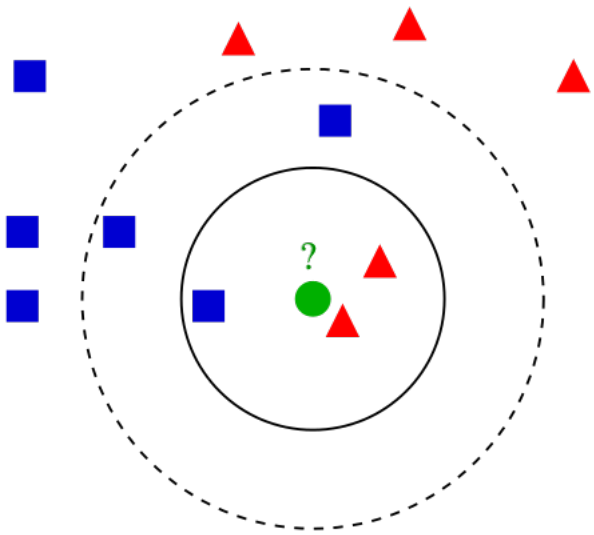


Рис. 2.1. Метод K-найближчих сусідів

2.4. Вибір операційної системи

Операційна система – це найважливіше програмне забезпечення, яке працює на комп'ютері. Воно керує пам'яттю, процесами та всім програмним та апаратним забезпеченням. Можна сказати, що ОС – це міст між комп'ютером і людиною.

Для комп'ютерів існує три найбільш популярні операційні системи: Microsoft Windows 10, Apple macOS та Linux.

На мобільних пристроях найпопулярнішими є Apple iOS, Google Android та Microsoft Windows 10.

Apple macOS та iOS встановлюються на всі комп'ютери та мобільні пристрої компанії Apple. Хоч Apple і заборонила, але встановити macOS на звичайний комп'ютер можливо, проте таке рішення не буде нормально працювати.

Linux, Android та Windows 10 є безкоштовними операційними системами.

Мобільні ОС за своїми функціями поступаються комп'ютерним, але все ж здатні виконати більшість основних завдань. Наприклад, перегляд фільмів, веб-сторінок, запуск застосунків, ігр та інше.

Графічний інтерфейс кожної ОС має свій зовнішній вигляд та функції, тому, якщо раптом змінити операційну систему на іншу, то на перший погляд та дотик незнайома операційна система може виявитись незвичною та незручною.

Більшість людей використовують ту операційну систему, яка вже була встановлена при купівлі комп’ютера, але за бажанням вони можуть її обновити, або встановити іншу.

На даний момент електронний світ заповнений масою мобільних пристроїв, планшетів і портативних комп'ютерів різних типів. Для більшості людей життя без засобів стільникового зв'язку вже немислима, а мешканці великих міст настільки активно користуються смартфонами, що вони стали для них просто порятунком в будь-якій ситуації. У зв'язку з цим істотно загострюється конкуренція на ринку програмного і апаратного забезпечення в даній сфері. Боротьба операційних систем може вважатися одним з найбільш рішучих протистоянь між мобільними гігантами. Саме питання про те, що краще: Android або iOS, став все частіше турбувати сучасних користувачів. Обидві системи представлені великими гравцями на ринку мобільних пристроїв, тому цілком зрозуміло, чому людям хочеться отримати кращий продукт.

Зараз не можна конкретно сказати, що краще: Android або iOS, так як перша система є більш новою в порівнянні з іншою. Вона динамічно розвивається, завойовуючи увагу все більшої кількості користувачів. Однак друга стала перевіреним помічником для багатьох шанувальників продукції Apple. Кожна система володіє своїми відчутними перевагами та недоліками, які зазвичай залежать від смаків користувачів. Такі люди не просто вибирають один продукт, але ще й ідентифікують себе як прихильників того чи іншого варіанту досить категорично.

Варто розглянути фундаментальні відмінності цих систем. У першу чергу варто сказати про дизайн, який подобається чи не подобається користувачам виключно в залежності від їх індивідуальних переваг. Якщо міркувати про зручність інтерфейсу, то перевага в даному випадку у Android, так як віджети та іконки встановлюються тут дуже просто. На всьому обладнанні від Apple віджети знаходяться в спеціальному меню, а змінити дизайн системи за своїм смаком практично неможливо.

В обох системах чудово реалізована підтримка браузерів, а також робота в соціальних мережах. Складно сказати, чому iOS краще Android, так як в обох платформах є підтримка безлічі програм для передачі відео та фото, а також

текстової інформації в реальному часі. Однак для продукту від Apple відзначається більш швидке з'єднання на протоколах Wi-Fi і 3G, що робить роботу з Інтернетом набагато зручнішою. Перевага iOS полягає у більш стабільній роботі застосунків і відсутності вірусів. Програмний код системи закритий, що захищає користувача від небажаного ПЗ. З Android ситуація трохи інша. Віруси тут – це звичайна ситуація, яка вирішується за допомогою установки антивірусного ПЗ, яке забезпечує захист ще під час інсталяції програм. Більшість хороших застосунків iOS є платними, а можливості їх взлому майже немає, тому iOS є кращою ОС для розробників, які бажають продавати свій програмний продукт, для конкурента є цілий набір розповсюджуваних вільно програм, або можливість безкоштовно встановити платні програми з сторонніх ресурсів, що забирає великий відсоток прибутку в розробників. Зрозуміло, що неможливо однозначно назвати ту чи іншу систему кращою, оскільки у кожної з них є переваги і недоліки, звернувши увагу на які, користувач зможе зробити для себе оптимальний вибір. Невелике порівняння iOS та Android показане на рисунку 2.4.1. та 2.4.2.

Рис. 2.2. Порівняння iOS та Android

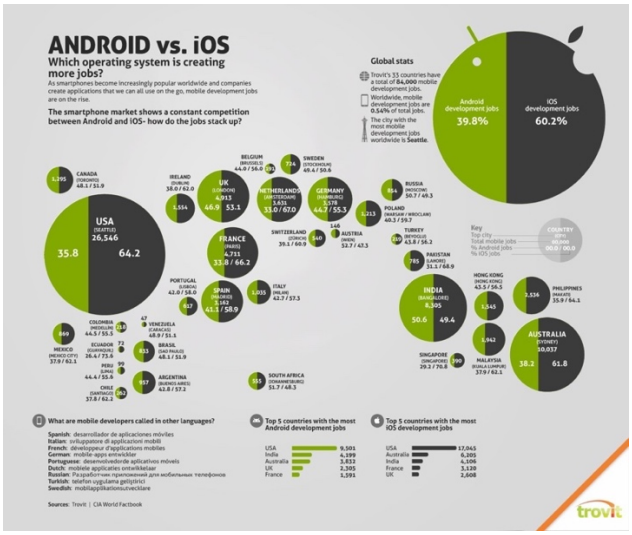


Рис. 2.3. Кількість нових розробників для кожної ОС

Для розробки програмного продукту для магістерської роботи було обрано операційну систему iOS. Дана операційна система завойовує собі фанатів простим та зрозумілим інтерфейсом, безпекою використання, швидкістю та закритою екосистемою Apple.

З моменту виходу першого телефона під управлінням iOS (Apple iPhone 29 червня 2007 року) було вироблено та продано більше ніж 1 млрд. пристроїв під управлінням операційної системи iOS (раніше iPhone OS).

За 13 років iOS зазнала багато змін як в дизайні, так і в своєму функціоналі, це зображено на рис. 2.4.3.

Рис. 2.4. Зміни в дизайні iPhone OS 1 – iOS 9

В першій версії iOS не було жодної підтримки застосунків. Всі зусилля розробників на iPhone OS 1 обмежувались сферою браузера Safari – HTML –

застосунками, які можна було розмістити на домашньому екрані. App Store з’явився лише в 2008 році з виходом iPhone 3G.

2.5. Висновки до розділу 2

В даному розділі проведено аналіз загальних принципів і методів ранжування відеоконтенту, зокрема фільмів та відео на найпопулярніших платформах.

Система оцінки фільмів прийнята в США, під назвою «Система рейтингів Американської кіноасоціації» є однією із найпопулярніших та відіграє важливу роль в долі прокату кінокартин. За цією системою фільм може отримати одну із п’яти рейтингів: G, PG, PG-13, R, NC-17, або NR (Not Rated) для тих фільмів, що не отримали рейтинг чи U (Unrated), для фільмів, які вийшли в прокат до введення системи рейтингів.

Досліджено методи побудови персональних рекомендацій для користувача на основі рейтингу.

Кожен користувач може ставити оцінки відповідно до своїх особистих вподобань. Високі оцінки визначають інтереси користувача, а низькі – антиінтереси. Користувачі можуть мати різні інтереси, але однаково оцінювати погані фільми, тому співпадіння двох десяток і одиниць має різну вагу. Також ще однією проблемою є співвідношення шкали з балами та з відсотками, тому підбираються спеціальні функції, які переводять передбачення оцінки з балів в відсотки так, щоб при цьому зберігались очікування користувачів. Саме тому великі служби вкладають значні кошти у розробку нових алгоритмів для створення особистих рекомендацій, щоб створити необхідний баланс між двома крайнощами та врахувати потенціал візуалізації та сприйняття рекомендації користувачем. Сортування всієї бази даних фільмів недоцільно, тому на першому кроці відбирають сотні претендентів, після цього, визначаються фактори за фільмами, користувачем та в парі користувач/фільм і ранжуються кандидати за допомогою машинного навчання.

Також було досліджено метод знаходження найближчих локацій на основі алгоритму k-найближчих сусідів.

Користувачі можуть відслідковувати погоду, пожежі, затори, ремонтні роботи в реальному часі за допомогою різних доступних сервісів. Одним з найвідоміших є Google Maps – це безкоштовний картографічний сервіс від компанії Google, а також набір різних застосунків, побудованих на базі цього сервісу, він являє собою карту та супутникові знімки всього світу та надає користувачам аналіз трафіку в реальному часі, панорамний перегляд вулиці і прокладання маршруту автомобілем, велосипедом, пішки або громадським транспортом. Також, користувачі можуть додавати відсутні на картах позначки (парки, сквери, маленькі вулички). Також існує Google Maps API для впровадження карт в свої додатки та веб-сторінки, витягування даних з Google Maps або отримання та передавання геолокації користувача. А для визначення найближчих закладів, зазвичай, використовується метод k-найближчих сусідів.

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

Вибір операційної система має дуже велике значення, адже це найважливіше програмне забезпечення, яке працює на комп’ютері. Для комп’ютерів існує три найбільш популярні операційні системи: Microsoft Windows 10, Apple macOS та Linux. На мобільних пристроях найпопулярнішими є Apple iOS, Google Android та Microsoft Windows 10. Графічний інтерфейс кожної ОС має свій зовнішній вигляд та ряд особливих функцій. Кожна система володіє своїми відчутними перевагами та недоліками, які зазвичай залежать від потреб та вподобань користувачів, звернувши увагу на них, користувач зможе зробити для себе оптимальний вибір.

Проведено дослідження ранжування контенту на YouTube.

На сьогоднішній час YouTube досяг досить високого рівня розвитку і навчився точно розуміти запити користувачів, та при ранжуванні видає максимально релевантні результати. Використання тегів, велика кількість натискань, надійний та точний опис відео, якість та надійність каналів та активність користувачів допомагають досягти найкращого ранжування.

РОЗДІЛ 3
РЕАЛІЗАЦІЯ МУЛЬТИФУНКЦІОНАЛЬНОЇ ПЛАТФОРМИ ДЛЯ ПЛАНУВАННЯ ВІДПОЧИНКУ

3.1 Вимоги до платформи

Для виконання поставлених задач магістерської роботи нам потрібно побудувати дерево проблем.

На рисунку 3.1. зображено дерево проблем розробки застосунку, де головною проблемою визначено власне сама розр обка, яка розбивається на складові: проблема створення списку, проблема створення карти та проблема створення списку ТОП-250. Кожна з цих проблем також містить свої компоненти.

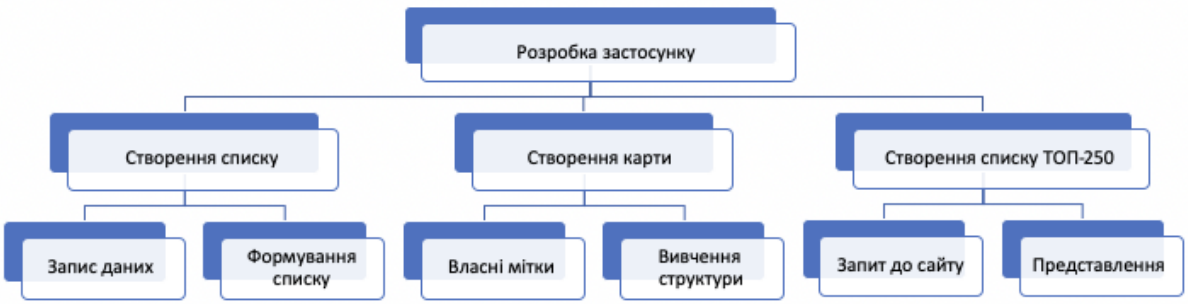


Рис 3.1. Дерево проблем

На підставі побудованого дерева проблем ми будуємо дерево цілей, яке зображено на малюнку 3.2. Дерево цілей також складається з головної цілі – розробки додатку та націлене на вирішення проблем. Ціль створення списку складається з аналізу літературних джерел та розробки програмного модуля. Карта також складається з аналізу літератури за темою побудови карт та побудови самої карти. ТОП-250 включає в себе написання коду для отримання даних з сайту iMDB.com та їх представлення. Тестування передбачає тестування основних

модулів та встановлення застосунку на пристрої. Впровадження має на меті підтримку застосунку та періодичне оновлення з доданням нових функцій.

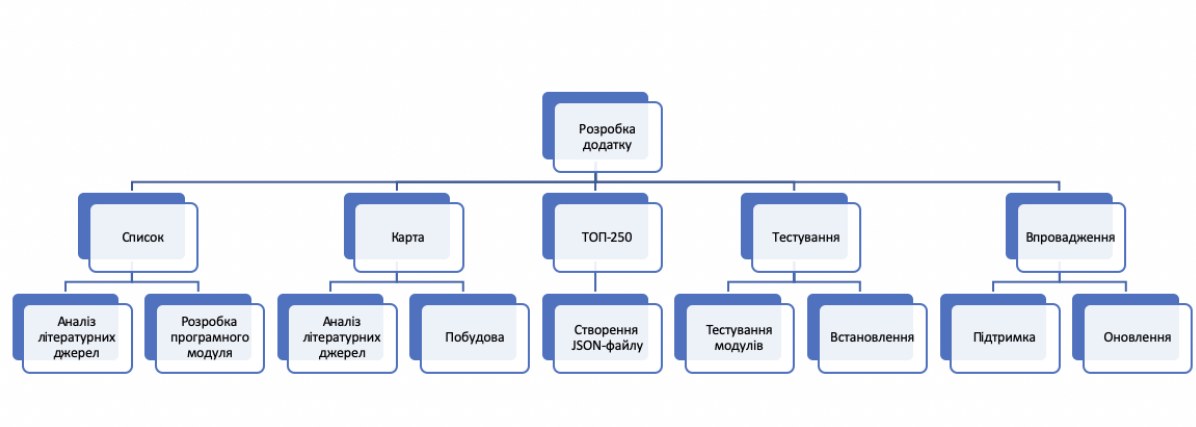


Рис 3.2. Дерево цілей

Досягнення поставлених цілей дасть можливість розробити бажаний програмний продукт з передбаченими певними функціональними можливостями.

3.1.1. Опис вибраної методології проектування та інструментальних засобів. Rapid Application Development (RAD) – методологія проектування ІС, особливістю якої являється швидкість та зручність програмування, тобто створення технологічного процесу, що надає можливість розробнику максимально швидко створювати ПЗ. Дана методологія ґрунтується на послідовності ітерацій прототипів, критичний аналіз яких обговорюється із замовником. Під час такого аналізу формуються вимоги до системи, що розробляється. Розроблення кожного інтегрованого продукту чітко обмежується певним періодом часу (зазвичай становить 60 днів). На відміну від інших підходів до розробки ПЗ, в RAD кожний прототип розвивається в частину майбутньої системи[6].

Можна виділити такі основні принципи методології RAD:

- CASE-засоби мають бути націлені на мінімізацію часу розробки;
- створення прототипу для уточнення вимог замовника;
- циклічність розробки (кожна нова версія програми ґрунтується на оцінці попередньої версії замовником);
- зменшення часу розробки версії за рахунок перенесення вже готових

модулів та функціоналу в нову версію;

- тісна співпраця команди розробників, та готовність кожного учасника розробки виконувати кілька завдань;
- управління проектом повинне мінімізувати тривалість циклу розробки.

RAD використовує спіральну модель життєвого циклу (ЖЦ) ПЗ. У даній методології виділяють такі 4 фази:

1. Фаза аналізу і планування вимог – визначаються функції системи та виділяються найбільш пріоритетні, описуються інформаційні потреби, обмежується масштаб проекту, визначаються часові рамки для кожної з наступних фаз. Результатом виконання даної фази є список функцій та їх пріоритетність, попередні функціональні та інформаційні моделі.

2. Фаза проектування – уточнюються і доповнюються вимоги до системи, більш докладно розглядаються процеси системи, оцінюється кількість функціональних елементів. Результати – загальна інформаційна модель системи, її функціональні модулі, точно визначені за допомогою CASE-засобів інтерфейси між підсистемами, побудовані прототипи екранів, звітів, діалогів і т. п.

3. Фаза побудови – безпосередньо виконання швидкої розробки системи, ітеративна побудова системи на основі попередньо створених моделей та функціональних вимог, здійснення тестування під час розробки, оцінка кінцевим користувачем системи і внесення коректив. Результатом є готова працююча система, що задовольняє всі узгоджені вимоги.

4. Фаза впровадження – проводиться навчання користувачів, організаційні зміни та паралельно з впровадженням нової системи здійснюється робота з існуючою системою.

Методологію RAD доцільно використовувати, коли чітко відомо напрями розробки проекту, але вимоги до ПЗ визначені загально, необхідно виконати проект в певні стислі терміни та при обмеженому бюджеті, також основна увага має приділятися інтерфейсу користувача, а обчислювальна складність ПЗ не повинна бути високою. Саме тому дана методологія була обрана для здійснення проектування системи, розглянутої у даній магістерській роботі [6].

Таблиця 3.1.

Порівняльна характеристика методологій проектування інформаційних систем

Методологія проектування	Переваги	Недоліки
1	2	3
Rapid Application Development (RAD)	1. Скорочений час циклу розробки завдяки використанню потужних інструментальних засобів. 2. Потрібна невелика команда фахівців. 3. Зменшення витрат завдяки скороченому ЖЦ.	1. Потрібна постійна участь користувачів у процесі для створення якісного проекту. 2. Потрібна висока кваліфікація та обізнаність фахівців.
	4. Зменшення витрат та ризиків, пов'язаних з дотриманням графіку. 5. До складу кожного часового блоку входить аналіз, проектування та впровадження. 6. Основну увагу перенесено з документації на код на розроблювальну систему.	3. Можливе невдале використання методології, якщо у системі відсутні компоненти, що можна використовувати повторно. 4. Для реалізації методології потрібні розробники та замовники, готові до швидкого виконання дій із жорсткими часовими обмеженнями.
Rational Unified Process (RUP)	1. Можливість зміни вимог, незалежно, чи вимагає цього клієнт чи зміни виникають у ході роботи над проектом. 2. Наявність точної та детальної документації. 3. RUP дозволяє отримати саме той рівень формалізації, який необхідний в проекті. 4.Інтеграція вимог відбувається протягом всього процесу розробки, в тому числі і на фазі побудови.	1. Ефективне виконання проекту залежить від вмінь професіоналів правильно розподіляти роботу між розробниками, які у свою чергу повинні видати заплановані результати. 2. Інтеграція нових вимог в процесі розробки може негативно позначиться на інших наступних етапах. 3. Впровадження даної методології є складним і дорогим, оскільки вимагає залучення спеціалістів та фахівців для здійснення навчання персоналу. 4. Використання детального документування вимог засобами UML, що розробникам інколи складно виконати якісно та змістовно.

Продовження Таблиці 3.1.

1	2	3
Гнучкі (Agile) методології	1. Здатність працювати в незвичній предметній області та швидко реагувати на зміни вимог замовником. 2. Завдяки тісній взаємодії з замовником, проект повністю відповідає його очікуванням. 3. Максимально швидкі терміни виконання проекту. 4. Можливість розробки з обмеженим бюджетом.	1. Відсутність чіткого уявлення про кінцевий продукт. 2. Потрібне постійне залучення замовника у процес розробки. 3. Неможливість застосовувати методологію у галузях, де від застосунків залежить безпека людей. 4. У зв’язку з “полегшеними” вимогами, можуть з’явитися проблеми з їх стабілізації та правильного тлумачення. 5. Можливі також проблеми, пов’язані з уривчастістю архітектури, ускладненням рефакторингу тощо.

Для безпосереднього проектування ІС використовують різноманітні CASE-засоби. CASE-засіб – програмне забезпечення, що підтримує процеси створення та супроводу ІС, а саме аналіз і формулювання вимог, проектування ПЗ та баз даних, генерацію коду, тестування, документування, керування та інші процеси. CASE-засоби разом із системним ПЗ і технічними засобами утворюють повне середовище розробки ІС[13].

Для проектування ІС у даній роботі використовуватиметься програмне середовище Enterprise Architect, All Fusion Process Modeler та Erwin Data Modeler.

Sparx Systems Enterprise Architect – це програмне забезпечення, що представляє собою набір UML-інструментів для виконання аналізу та дизайну моделей ПЗ та підтримки повного циклу розробки від аналізу до впровадження.

Даний програмний продукт використовується в багатьох галузях промисловості більше ніж в 160 країнах світу. За 15 років існування даний продукт постійно розвивається та доробляється згідно з потребами розробників, бізнес-аналітиків, архітекторів, менеджерів проектів, тестерів, дизайнерів та інших.

Заснований на відкритих стандартах, Enterprise Architect легко масштабується від невеликих моделей до користувацьких великих сховищ чи розподілених рішень на базі хмари.

AllFusion Process Modeler або Bpwin – програмний продукт, що створений для моделювання, аналізу, опису та оптимізації бізнес-процесів. За допомогою даного продукту можна створювати графічні моделі бізнес-процесів. Bpwin допомагає візуалізувати модель організації процесів у вигляді схеми виконання робіт, організації документообігу, обміну інформацією і т. д. Це дає можливість використовувати передові технології для вирішення задач управління організацією чи системою[14].

AllFusion Data Modeler або Erwin - це CASE-засіб для проектування та документування баз даних, що дозволяє створювати, документувати та супроводжувати різноманітні види баз даних. Даний продукт призначений для всіх компаній, що розробляють та використовують бази даних, для адміністраторів БД, системних аналітиків, проектувальників БД, розробників, керівників проекту. Erwin дозволяє керувати даними в процесі корпоративних змін та в умовах технологій, що стрімко розвиваються[14].

3.1.2. Моделювання процесів предметної області. Для того щоб змоделювати процеси будь-якої предметної області використовують різноманітні CASE-засоби, які підтримують основні стандарти моделювання та відповідні їм діаграми. До них належать IDEF0, IDEF3, UML та інші. Дані стандарти допомагають зрозуміти процеси, що відбуваються у проектованій системі, описати її документообіг, які взаємодії відбуваються між інформаційними потоками системи, описати функціонал системи з боку її користувачів та т. п.. Тож для повного розуміння проектованої системи та її процесів всередині неї, обов’язковим є створення наступних діаграм.

3.1.3. Концептуальна модель. Концептуальна модель системи містить користувача, який взаємодіє з системою, відправляє їй запити та отримує відповідь на них. На рис. 3.3. зображена спрощена модель функціонування системи.

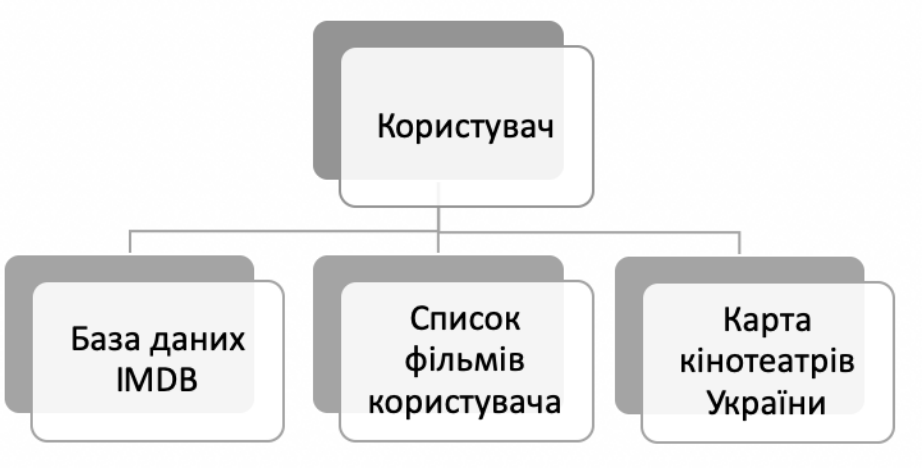


Рис. 3.3. Спрощена модель

У кожного користувача буде один і той самий інтерфейс. Взаємодіяти користувачі будуть з однією і тією ж базою, але будуть мати можливість створювати власні списки фільмів.

3.1.4. Структурна схема системи з врахуванням інформаційних потоків. На рис. 3.4. зображена концептуальна модель системи з описами запитів та відповідей на них.

Рис. 3.4. Концептуальна модель

При потраплянні на основну сторінку застосунку відображається список фільмів користувача, який він може змінювати (додавати, видаляти, редагувати елементи).

При переході на сторінку з ТОП-250 йде автоматичний запит про інформацію з сайту iMDB.com, яка відображається в вигляді списку, в якому можна детальніше дізнатись про фільм.

При переході на сторінку з картою, користувачеві відображає найближчі кінотеатри до його місця розташування.

3.2 Проектування архітектури

Діаграма послідовності (sequence diagram) - діаграма, на якій показані взаємодії об'єктів, впорядковані за часом їх прояву. За допомогою діаграми послідовності можна представити взаємодію елементів моделі як своєрідний часовий графік "життя" всієї сукупності об'єктів, пов'язаних між собою для реалізації варіанту використання програмної системи, досягнення бізнес-мети або виконання якої-небудь задачі[8].

Діаграма послідовності для мобільного застосунку, що проектується, зображена на рис. 3.5.

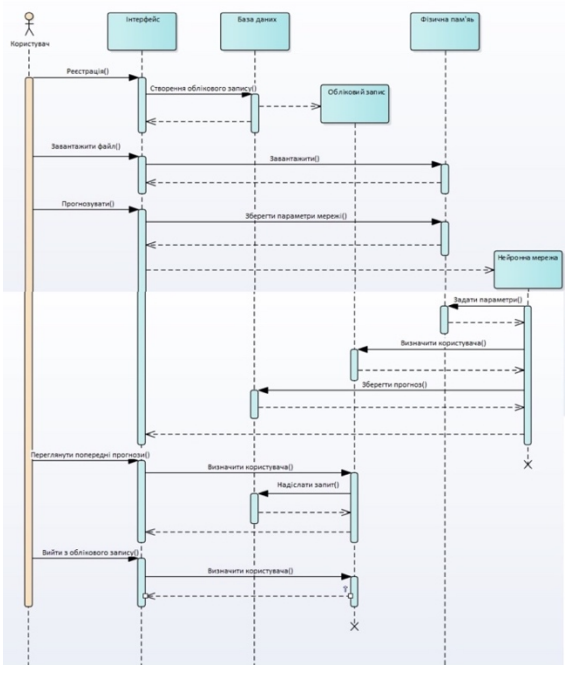


Рис. 3.5. Діаграма послідовності

На даній діаграмі у вигляді вертикальних ліній показано різні процеси або об'єкти, що існують водночас. Надіслані повідомлення зображуються у вигляді горизонтальних ліній, в порядку відправлення, а відповіді на повідомлення – у вигляді пунктирних ліній.

Однак не всі об’єкти існують від самого початку роботи системи. Наприклад обліковий запис користувача створюється у випадку, коли користувач за допомогою інтерфейсу обрав дію створення облікового запису, попередньо заповнивши форму реєстрації. Інтерфейс у відповідь на цю дію передав повідомлення про створення облікового засобу. У випадку створення об’єкту в списку інтерфейс надсилає повідомлення про збереження у фізичній пам’яті назви елемента, і очікує відповідь. Лише після цього інтерфейс ініціює створення. В даному випадку відповіддю буде новий елемент списку, який інтерфейс виведе на екран.

Діаграма послідовності по суті описує динамічний аспект функціонування системи, однак для розробки ІС важливо розуміти як статично зобразити систему. Для цього використовують діаграми класів.

На діаграмах класів показані різноманітні класи, які утворюють систему, а також їх взаємозв’язки. Діаграми класів називають “статичними діаграмами”, оскільки на них показано класи разом з атрибутами і операціями (методами), а також статичний взаємозв’язок між ними: те, яким класам «відомо» про існування яких класів, і те, які класи «є частиною» інших класів, — але не показано методи, які при цьому викликаються. Перевагами створення діаграми класів є те, що вони допомагають на концептуальному рівні формувати «словник предметної області», а на рівні специфікації і реалізації визначати структуру класів у програмній реалізації системи[8].

Для даного мобільного застосунку діаграму класів можна зобразити наступним чином (рис. 3.6).

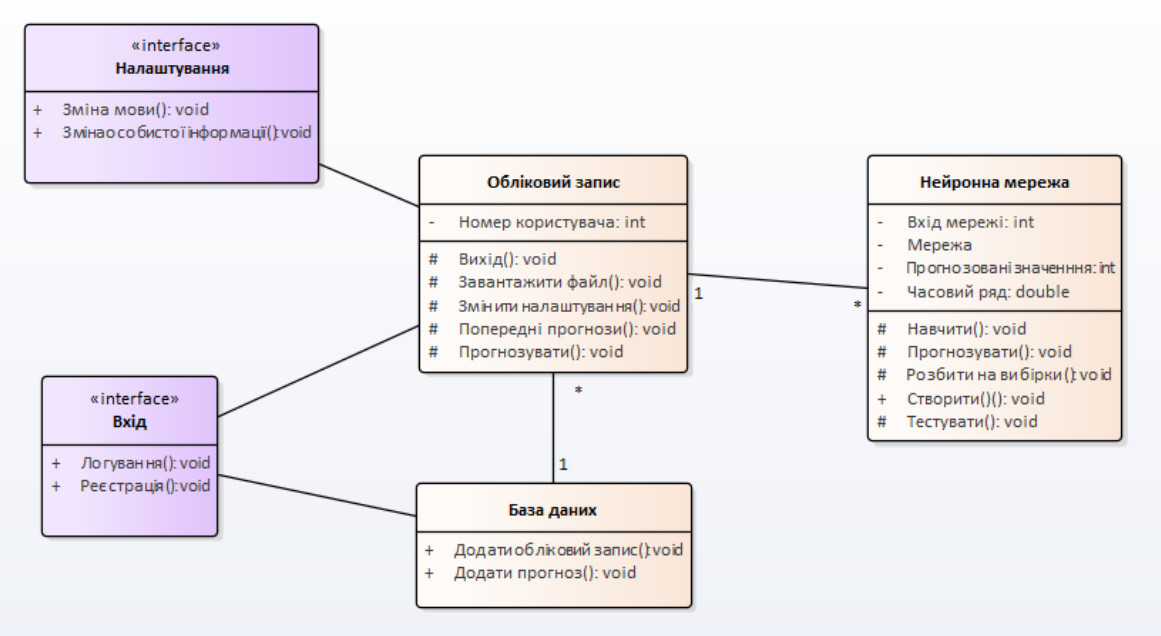


Рис. 3.6. Діаграма класів

Функціонал нашої ІС складається з 3-ох класів та 2-ох інтерфейсів. Інтерфейси — це абстрактні класи, тобто з них не можна напряму створювати екземпляри. У інтерфейсах можуть міститися операції, але не атрибути. В даному випадку інтерфейси використовуються для входу в систему та зміни налаштувань застосунку. Існуючі класи системи містять різноманітні атрибути та операції і взаємодіють між собою. Дані взаємодії зображені у вигляді зв’язків асоціації з зазначеною численністю, що визначає кількість об’єктів на відповідному кінці асоціації, які можуть мати зв’язок з одним з об’єктів на іншому кінці асоціації. Наприклад об’єкт класу «Обліковий запис» може бути зв’язаний з невизначеною кількістю об’єктів класу «Нейрона мережа». Також слід звернути увагу що операції (або методи) у різних класах та інтерфейсах мають різну область видимості.

3.3 Алгоритми роботи модулів

Для розуміння роботи програми необхідно побудувати алгоритми роботи. На наведеному нижче рисунку 3.7 показана блок-схема роботи застосунку.

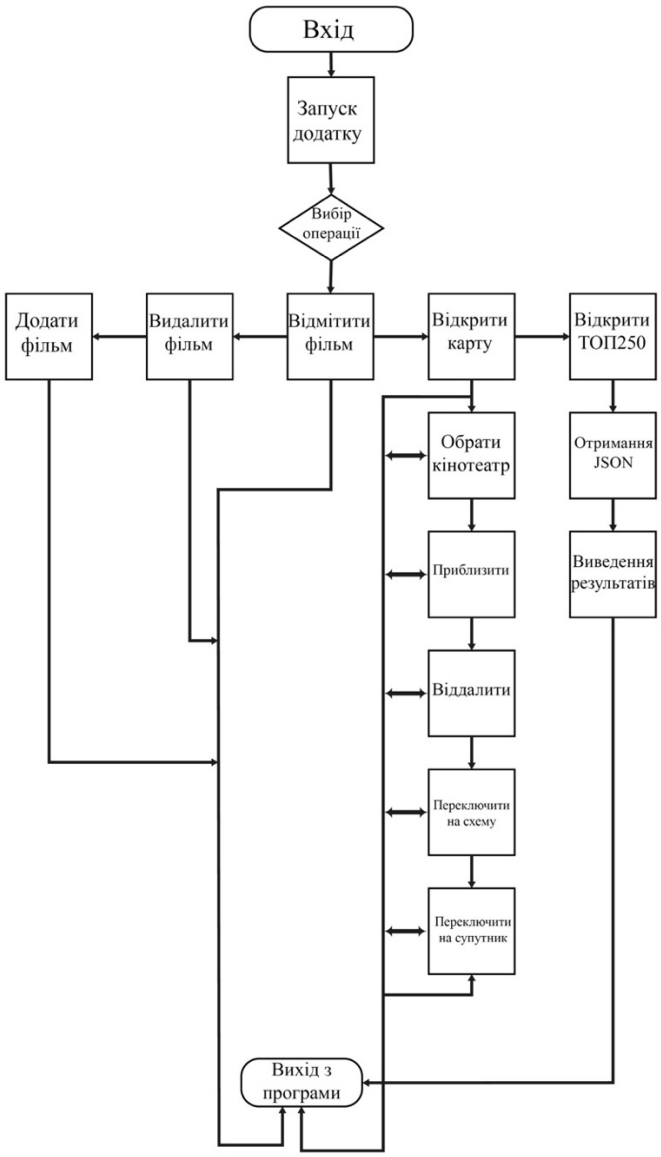


Рис. 3.7. Блок схема застосунку

Після запуску застосунку користувач має можливість обрати операцію з доступного переліку: додати фільм, видалити фільм, відмітити фільм, перейти до карти, перейти до списку ТОП-250.

При відкритті карти користувач може обрати кінотеатр, переглянути інформацію про нього та виконувати переміщення по карті.

При відкритті списку ТОП-250 користувач має змогу прочитати короткі відомості про фільм та переглянути оцінку користувачів та рейтинг.

Кожна з функцій отримує певний вхід і в результаті виконання видає вихід.

Після завершення вище описаних дій, користувач може повернутися до виконання іншої операції або завершити роботу.

3.3.1. Моделювання схеми даних проектованої системи. Тож, описавши дану ІС у вигляді вище розглянутих діаграм ми отримуємо чітке уявлення про всі процеси та потоки інформації, що в ній відбуваються та про їх взаємодію. Однак для розуміння системи недостатньо змоделювати усі інформаційні потоки чи потоки даних, потрібно описати саму структуру даних, тобто змоделювати схему даних, що в подальшому використовуватиметься для побудови бази даних ІС. Для вирішення цього завдання використовується методологія IDEF1x.

Методологія IDEF1X – це мова для семантичного моделювання даних , що ґрунтується на концепції «сутність – зв’язок». Діаграма «сутність – зв’язок» (entity-relationship diagram або ERD)призначена для розробки моделі даних та забезпечує стандартний спосіб визначення даних та відношень між ними. За допомогою ERD здійснюється деталізація сховищ даних системи, а також документуються сутності системи і способи їх взаємодії[10].

Рис. 3.8. Діаграма «сутність – зв’язок»

Діаграма «сутність – зв’язок» для ІС, що проектується, зображена на рис. 3.8. Отже діаграма описує дві сутності: Користувач та Фільм. Між сутностями існує зв’язок типу один до багатьох, це означає що один користувач може здійснити кілька прогнозів. Саме такою є схема даних системи, що проектується. Кожна з сутностей має ключовий атрибут та інші атрибути різних типів, здебільшого текстового.

3.3.2. Моделювання системи в стандарті UML. Ще одним аспектом опису ІС є опис системи з боку користувача, тобто опис випадків використання системи, функціонал, що буде надаватися користувачу з боку системи, та опис окремих

функціональних блоків системи, таких як класи, модулі чи підсистеми. Саме для цього використовують UML.

UML - уніфікована мова моделювання, що використовується у парадигмі об'єктно-орієнтованого програмування. Є невід'ємною частиною уніфікованого процесу розробки програмного забезпечення. UML є мовою широкого профілю, це відкритий стандарт, що використовує графічні позначення для створення абстрактної моделі системи, яка називається UML-моделлю. UML був створений для визначення, візуалізації, проектування й документування в основному програмних систем. В UML використовується 14 видів діаграм, однак для опису нашої системи використаємо три з них - діаграму прецедентів або випадків використання, діаграму послідовності та діаграму класів.

Діаграма прецедентів в UML - це діаграма, на якій зображено відношення між акторами та прецедентами в системі. Також перекладається як діаграма варіантів використання.

Діаграми прецедентів або діаграми варіантів використання (use-case diagrams) - це діаграми, які описують функціональність, що буде надаватись користувачам системи, яка проектується. Створюються шляхом використання прецедентів та акторів, а також відношень між ними. Набір усіх прецедентів діаграми фактично визначає функціональні вимоги, за допомогою яких може бути сформульоване технічне завдання[8].

Діаграма прецедентів для даної ІС зображена на рис. 3.9.

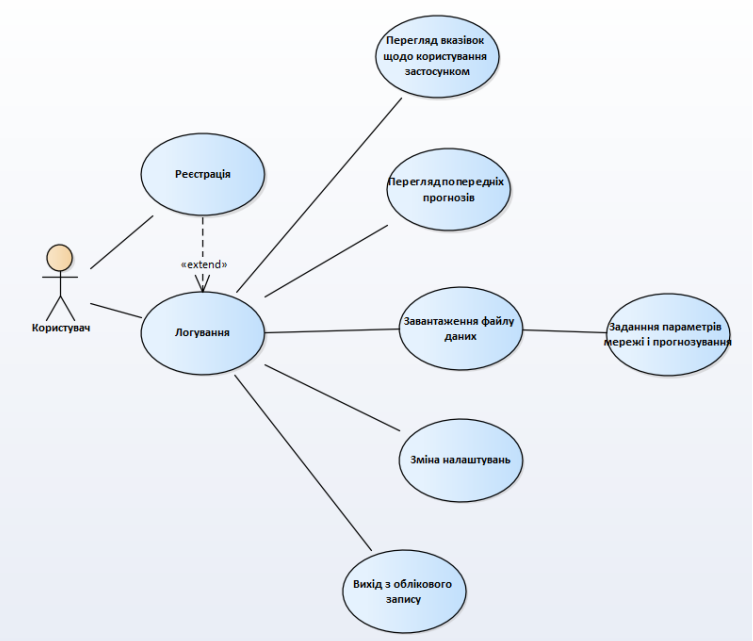


Рис. 3.9. Діаграма прецедентів

Як бачимо, на даній діаграмі вказаний увесь необхідний функціонал системи з боку його використання користувачем у тій послідовності, в якій це можна здійснити.

Відношення розширення (extend) застосовують для моделювання таких частин прецеденту, які користувач сприймає як необов'язкову поведінку системи. Тим самим можна розділити обов'язкову й необов'язкову поведінку. На діаграмі прецедентів відношення розширення зображують у вигляді залежності зі стереотипом extend. Пунктирна стрілка має бути спрямована до базового прецеденту (до прецеденту, який розширюється), від абстрактного (того, що розширює)[8].

Наступним типом діаграми мови UML, що використовується при моделюванні даної ІС, є діаграма послідовності.

3.4 Тестування та інтерфейс платформи

Тестування даної системи проводитиметься безпосередньо під час розробки програми. Участь у тестуванні прийматимуть тестувальник, системний аналітик та Junior developer. Буть використовуватися як динамічні так і статичні методи тестування.

На етапі статичного тестування перевіряється вся документація, отримана як результат життєвого циклу програми. Це і технічне завдання, і специфікація, і вихідний текст програми на мові програмування. Вся документація аналізується на предмет дотримання стандартів програмування. У результаті статичної перевірки встановлюється, наскільки програма відповідає заданим критеріям та вимогам замовника. Статичне тестування виконуватимуть системний аналітик та Junior developer.

Динамічні методи застосовуються в процесі безпосереднього виконання програми. Коректність програмного засобу перевіряється на безлічі тестів або наборів підготовлених вхідних даних. При прогоні кожного тесту збираються та аналізуються дані про відмови та збої в роботі програми. Динамічне тестування здійснюватиме тестувальник з допомогою Junior developer.

«Пілотне» впровадження системи відбуватиметься наступним чином. Для всіх класів користувачів необхідно здійснити встановлення застосунку на їхні мобільні пристрої, щоб забезпечити можливість працювати у розроблені системі. Для цього потрібно завантажити на пристрій застосунок та запустити його. Після першого запуску програми користувач має надати їй відповідні дозволи для коректної роботи. Також у майбутньому буде можливість завантажити даний застосунок з офіційного магазину програм iOS.

3.4.1. Середовище розробки застосунків Xcode. Інтегроване середовище розробки (IDE) – комплексне програмне забезпечення для розробки програмного забезпечення. Зазвичай складається з редактора початкового коду, інструментів для автоматизації, складання та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

Деякі середовища розробки містять компілятор, інтерпретатор (Eclipse) або ж обидва, інші не містять жодного з них (Lazarus). Деякі інтегровані середовища розробки містять систему керування версіями або інструменти для полегшення розробки графічного середовища користувача (GUI) (Xcode, Embarcadero, Delphi). Багато сучасних IDE містять інспектор класів, інспектор об’єктів, схему ієрархії класів для полегшення об’єктно-орієнтованої розробки програмного забезпечення.

Для розробки застосунку було використане середовище Xcode 10.

Xcode – інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання “універсальних” (Universal Binary) прикладних програм для Mac OS X.

Xcode включає в себе більшу частину документації розробника від Apple та Interface Builder – застосування, яке використовується для створення графічних інтерфейсів.

Пакет Xcode містить змінену версію вільного набору компіляторів GNU Compiler Collection і підтримує мови C, C++, Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java. Сторонніми розробниками реалізована підтримка GNU Pascal, Free Pascal, Ada, C #, Perl, Haskell і D. Пакет Xcode використовує GDB як back-end для свого відналагоджувача.

У серпні 2006 Apple оголосила про те, що DTrace, фреймворк динамічного трасування від Sun Microsystems, випущений як частина OpenSolaris, буде інтегрований в Xcode під назвою Xray. Пізніше Xray був перейменований в Instruments.

З червня 2014 року став доступний Xcode 6, в якому з’явилась підтримка нової мови програмування Swift з врахуванням особливостей 4000 нових API.

В червні 2015 року з’явився Xcode 7, де з’явилась оновлена версія мови Swift на якій був оновлений Interface Builder.

З вересня 2016 став доступний новий Xcode 8, основне вікно якого зображене на рисунку. 2.7. В ньому з’явилась підтримка iOS 10, Swift 3.0, оновлений і розширений дебаггер.

В червні 2018 року на WWDC показали Xcode 10 з підтримкою Swift 4.2 та технології Metal 2.1. З цієї версії Xcode більше не підтримує 32-бітні застосунки на macOS.

3 червня 2019 року Apple випустили бета-версію Xcode, 11 в яку додали підтримку Swift 5.0.

22 червня 2020 року було представлено Xcode 12 з підтримкою Swift 5.3.

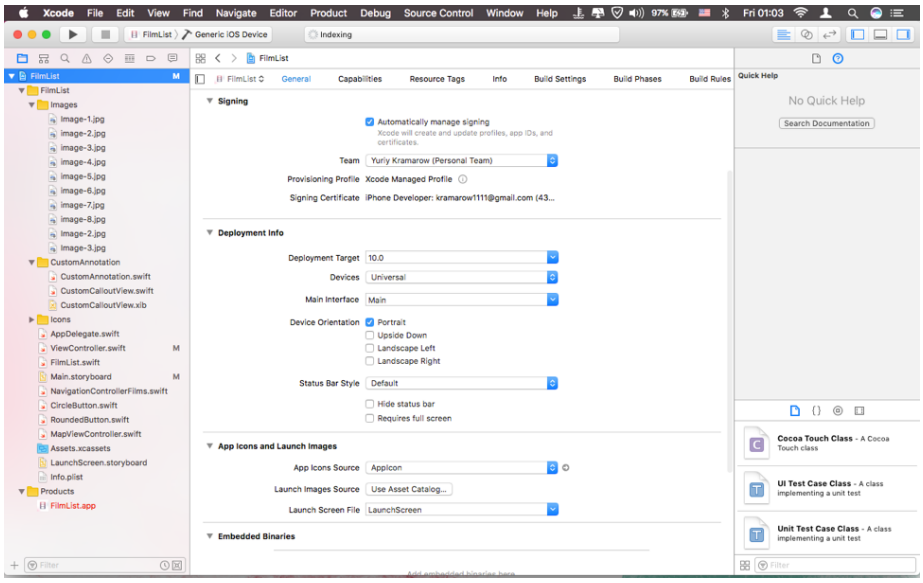


Рис 3.10. Основне вікно проекту в Xcode 10

Конструктор форм Xcode 10 показаний на рисунку 3.11.

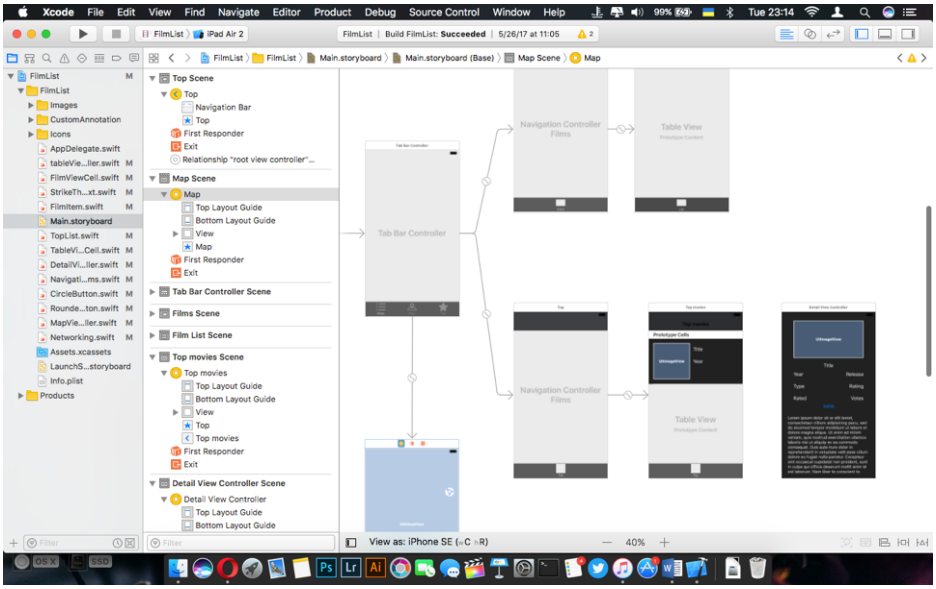


Рис 3.11. Конструктор форм Xcode

Редактор коду застосунку в Xcode 10 показаний на рисунку 3.12.

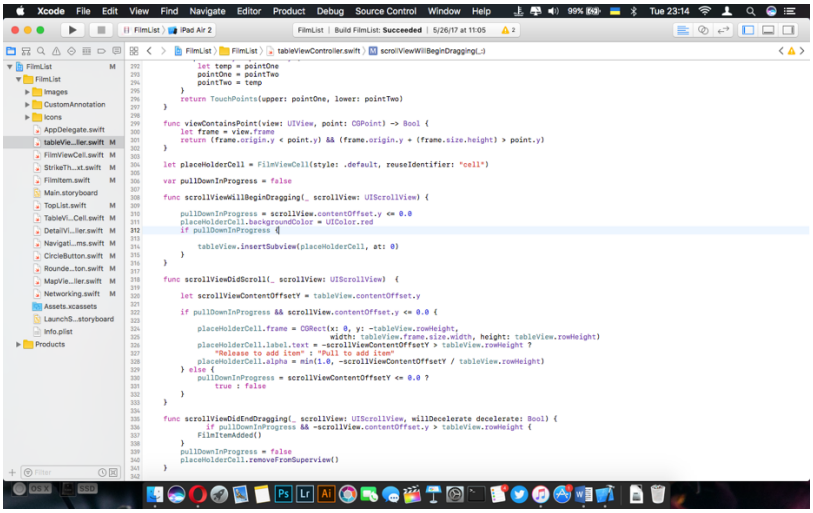


Рис 3.12. Редактор коду Xcode

Вбудований емулятор iOS дає змогу тестувати розроблений застосунок на найновіших ОС та найновіших версіях Mac, iPhone, iPad, iPod, Apple TV, Apple Watch. Список вибору пристроїв для тестування в Xcode зображено на рисунку 3.13.

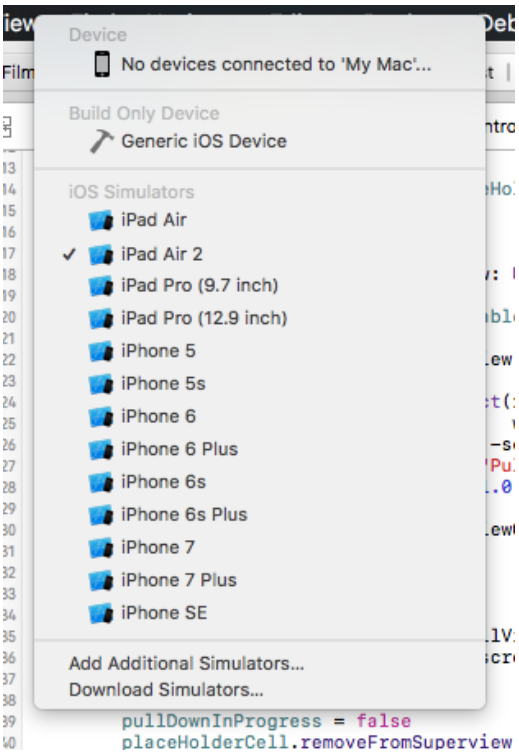


Рис 3.13. – Список пристроїв для тестування застосунку в Xcode

Як видно на рисунку 3.13, Xcode дозволяє проводити тестування застосунків на всіх актуальних пристроях під управлінням iOS.

3.4.2. Встановлення додатку для тестування. Для тестування розробник може використовувати вже наявні емулятори всіх версій операційної системи iOS та всіх пристроїв які її підтримують, які вмонтовані в середовище Xcode.

Також є можливість встановити свій проект на будь-який наявний iOS-пристрій для тестування, але такий застосунок пробуде на пристрої тільки тиждень, після чого його потрібно буде перевстановити тому, що в кожного застосунку є електронний підпис, який видається лише на тиждень якщо в розробника немає спеціального профіля.

Для подальшого поширення додатку для тестування чи вже для користування розробнику необхідно мати «Профіль розробника Apple», який коштує 100$ на рік. Такий профіль дасть змогу поширювати тестові версії свого застосунку через TestFlight для обраних користувачів через їх Apple ID, який дасть змогу отримувати відгуки користувачів та швидко вносити зміни та виправлення в застосунок для подальшого тестування. Також такий профіль розробника дасть змогу публікувати свої застосунки в Apple App Store для завантажування всіма бажаючими.

Для тестування розробник може використовувати вже наявні емулятори всіх версій операційної системи iOS та всіх пристроїв які її підтримують, які вмонтовані в середовище Xcode.

Також є можливість встановити свій проект на будь-який наявний iOS-пристрій для тестування, але такий застосунок пробуде на пристрої тільки тиждень, після чого його потрібно буде перевстановити тому, що в кожного застосунку є електронний підпис, який видається лише на тиждень якщо в розробника немає спецільного профіля.

Для подальшого поширення додатку для тестування чи вже для користування розробнику необхідно мати «Профіль розробника Apple», який коштує 100$ на рік. Такий профіль дасть змогу поширювати тестові версії свого застосунку через TestFligt для обраних користувачів через їх Apple ID, який дасть змогу отримувати відгуки користувачів та швидко вносити зміни та виправлення в застосунок для подальшого тестування. Також такий профіль розробника дасть змогу публікувати свої застосунки в Apple App Store для завантажування всіма бажаючими.

3.4.3. Вимоги до програмного та апаратного забезпечення та основні команди мови Swift. Apple Xcode дає змогу розробляти застосунки на всі версії iOS починаючи від 8.0 та всі пристрої, які її підтримують.

Даний проект розроблявся для iPhone та iPad на версії iOS 12 і вище тому, що існує багато технологій та бібліотек, які не підримує старіша версія ОС.

З виходом нової ОС кожного року з'являється нова версія Xcode і нова Swift, в якій інколи міняється синтаксис. В такому випадку після оновлення системи та середовища, Xcode сам пропонує перевести ваш проект на новішу версію мови без втручання розробника і показує де саме будуть внесені зміни.

В місцях, де середовище не може внести зміни самостійно, Xcode дає підказку користувачеві як варто змінити код, що показано на рисунку 3.14.

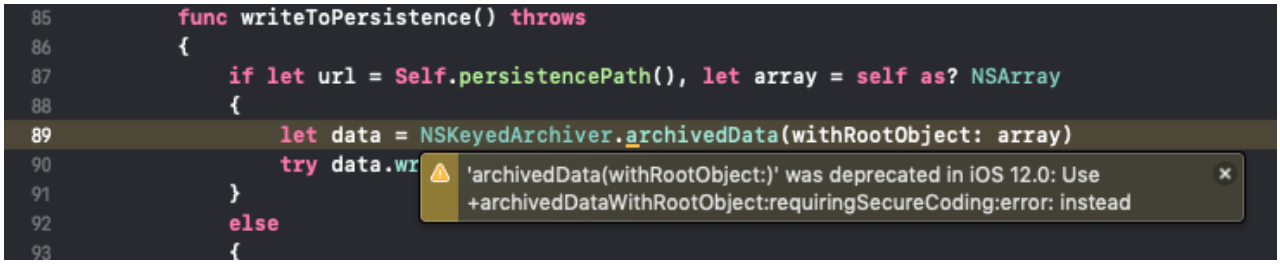


Рис. 3.14. Підказка Xcode щодо зміни коду

На вищепоказаному рисунку показана підказка середовища щодо зміни коду, відповідно до нових стандартів мови.

Мова Swift є дуже простою та інтуітивною, тому її синтаксис та команди будуть зрозумілі користувачам з низьким рівнем знань програмування. В Swift додані сучасні функції, які перетворюють створення застосунків в простий, більш гнучкий та цікавий процес.

В мові Swift коментарі можуть записуваитсь як зазвичай "//", або "/* */" для багаторядкового коментаря.

Для оголовшення змінних в Swift використовується службове слово "var", після якого повинні бути ім'я змінної, її тип та початкове значення:

```
var explicitDouble: Double = 70
```

Якщо тип змінної не було вказано, то Swift обере його автоматично на основі початкового значення:

```
var implicitInteger = 70
var implicitDouble = 70.0
```

Оголошення константи починається з службового слова “let”, після якого повинні бути ім’я змінної, тип змінної та початкове значення:

```
let numberOfBananas: Int = 10
```

Якщо тип константи не вказано, то його обере автоматично на основі початкового значення.

Значення змінних та констант можуть бути виставлені у рядки (змінні типу sring) наступним чином:

```
    let appleSummary = "I have \(numberOfApples)
apples."
    let fruitSummary = "I have \(numberOfApples +
numberOfOranges) pieces of fruit."
```

Оголошення масиву даних:

```
var fruits = ["mango", "kiwi", "avocado"]
```

Приклад оператора if, функцій .isEmpty та .count

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

Щоб оголосити “словник” (dictionary) з 4 елементів, кожен з яких містить ім’я та вік

```
    let people = ["Anna": 67, "Beto": 8, "Jack": 33,
"Sam": 25]
```

Використовуючи можливості мови Swift можна надрукувати обидва значення в єдиному циклі

```
    for (name, age) in people {
    print("\(name) is \(age) years old.")
    Оголошення методів починається із службового слова
"func"
    func sayHello(personName: String) -> String {
    let greeting = "Hello, " + personName + "!"
    return greeting
    }
```

3.4.4. Публікація застосунку в Apple App Store. Для того, щоб викласти застосунок в App Store, буде потрібен платний аккаунт розробника, середовище розробки Xcode та власне сам застосунок.

Всі застосунки перед завантаженням в App Store повинні бути підписані сертифікатор розробника. Це потрібно, щоб користувачі App Store були впевнені, що завантажують конкретний застосунок від конкретного розробника, а не підробку від чужого імені. Процедура підпису застосунку дозволяє операційній системі дізнатись, хто являється розробником та впевнитись в тому, що застосунок не був змінений з моменту його зібрання.

Сертифікат предствляє собою пару ключів асимметричного шифрування: приватний та публічний. В процесі зібрання Xcode формує цифровий підпис збірки, оснований на даних приватного ключа. Перевірити підпис можна за допомогою публічного ключа, який доступний і для Apple, яка цей сертифікат і видала.

Одного сертифікату достатньо для публікації необмеженої кількості застосунків.

Після отримання профілю розробника потрібно додати застосунок до iTunes Connect в розділ MyApps, що показано на рисунку 3.15.

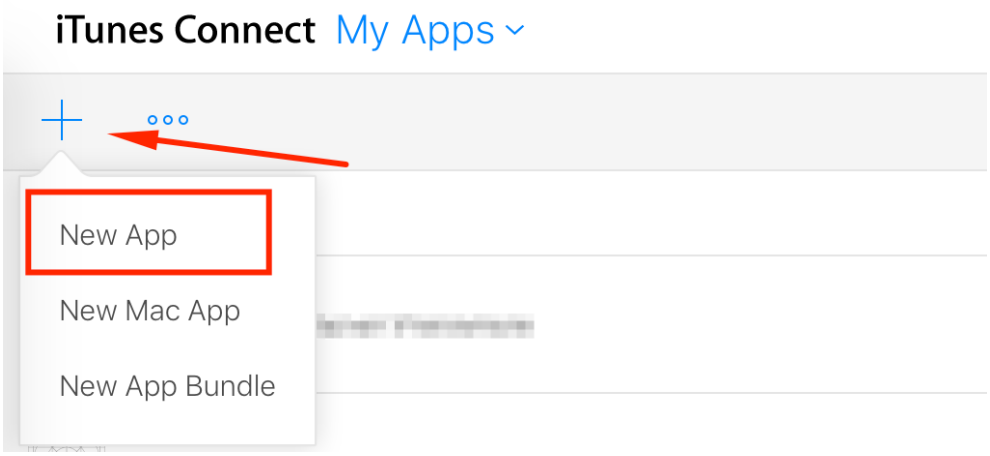


Рис. 3.15. Вкладка додання застосунку на сайті iTunes Connect

В формі, яка відкриється необхідно вказати платформу, назву застосунку, яка буде відображатись в App Store (до 30 знаків), вказати основну мову застосунку та вказати AppID застосунку – унікальний ідентифікатор для фінансових звітів та аналітики.

Після цього попадемо на сторінку застосунку. Потрібно заповнити інформацію про вартість застосунку в магазині та заповнити маркетингову інформацію (скріншоти, відео, ключові слова для пошуку, опис, віковий рейтинг, найменування правовласника, контакну інформацію розробника) в пункті «Pricing and Availability».

Також є секція для того, щоб можна було вказати певну версію застосунку, яку буде відправлено на огляд, а після чого в магазин.

Варто зазначити, що без всіх необхідних іконок, застосунок не пройде автоматичну перевірку в iTunes Connect.

Тепер в Xcode необхідно перейти в меню Product – Archive для того, щоб зібрати та відправити проект в iTunes Connect. Після зібрання застосунку буде показане вікно органайзера Xcode та кнопка «Upload to App Store», яка зображена на рисунку 3.16.

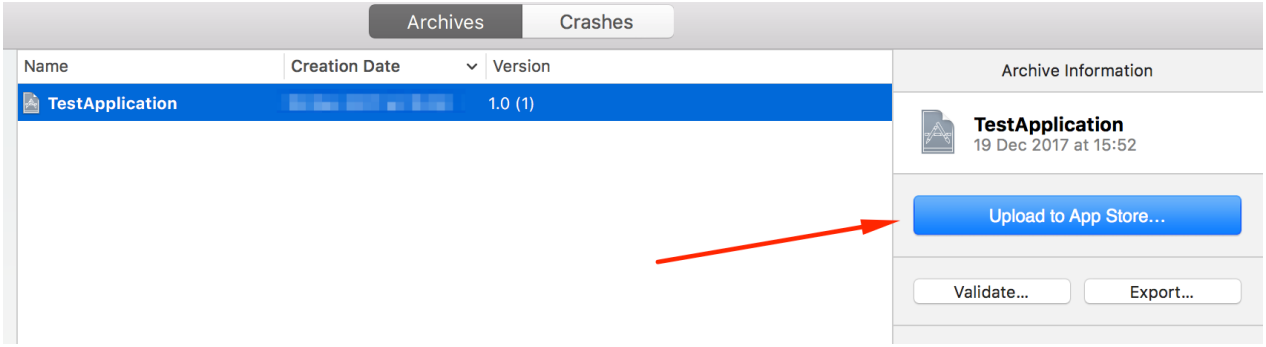


Рис. 3.16. Меню архівування проекту в Xcode.

Після цього відкриється вікно налаштувань завантаження в App Store, налаштування можна залишити без змін та продовжити. На наступному кроці необхідно буде вибрати provisioning profile з випадаючого списку. Xcode підготує архів для відвантаження в iTunes Connect, зображений на рисунку 3.17, а також ще раз показує вибрані нами параметри.

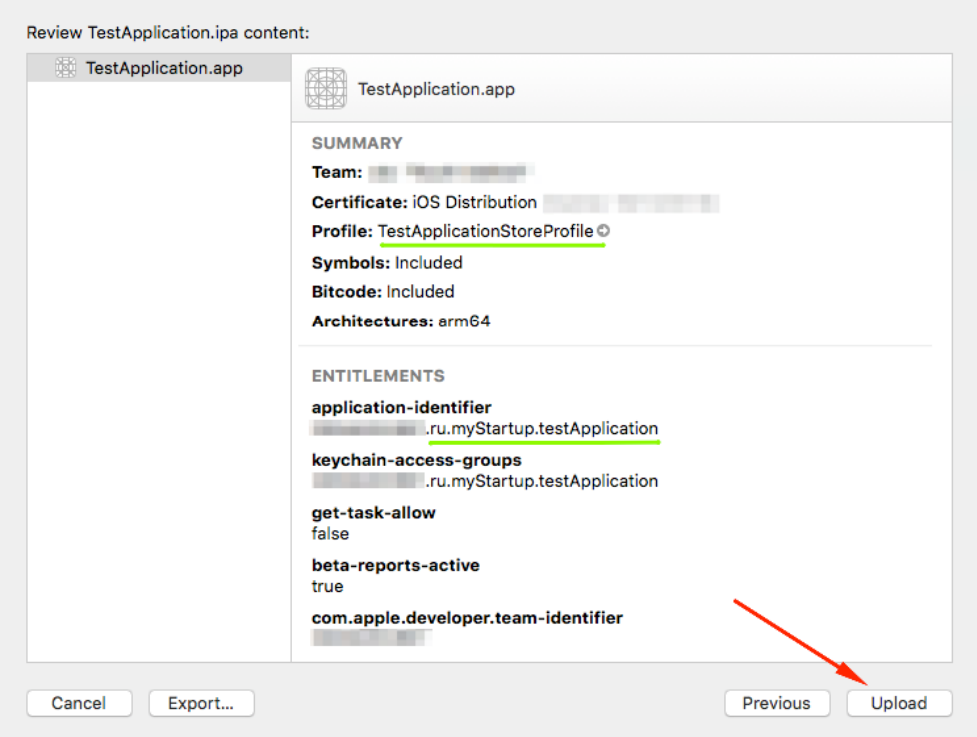


Рис 3.17. Архів для відвантаження в iTunes Connect.

Після натискання кнопки «Upload» в залежності від швидкості з'єднання потрібно буде зачекати деякий час. Якщо все добре, то Xcode повідомить про успішене відвантаження до iTunes Connect.

Після цього кроку необхідно відправити застосунок на огляд. Зробити це можна в iTunes Connect на вкладці Activity. Для Swift-проектів автоматична перевірка займає приблизно 30хв. До цього часу застосунок буде зі статусом «Processing».

Коли завершиться перевірка, застосунок буде доступний для вибору на сторінці інформації про версію застосунка, після цього в iTunes Connect з'явиться іконка, необхідно зберегти зміни та можна відправити застосунок на огляд.

На протязі декількох днів ви отримаєте відповідь, щодо вашого застосунку, де вам повідомлять чи з'явиться він в App Store або повідомлять, що він порушує певні норми або правила та вам доведеться вносити зміни.

3.4.5. Розробка додатку. Створення проекту в середовищі Xcode:

Для створення проекту в Xcode потрібно у вікні привітання, яке зображене на рисунку 3.18 слід вибрати пункт "Create a new Xcode project" (Створити новий проект Xcode) [5].

Рис. 3.18. Вікно привітання Xcode

Після цього можна 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 3.19.

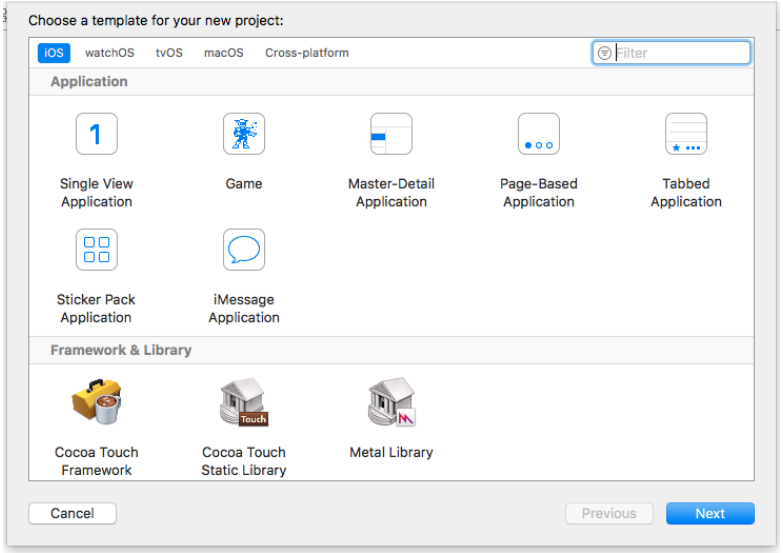


Рис. 3.19. Вікно вибору шаблонів в Xcode

Основним файлом, що створює Xcode в папці проекту є файл з розширенням .xcodeproj, який зображений на рисунку 3.20.

Рис. 3.20. Вигляд файлу .xcodeproj

Далі 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. На рисунку 3.21 зображене вікно створення проекту. У додатках Б та В наведено скріншоти iPad-версії застосунку, що розроблявся для iPhone.

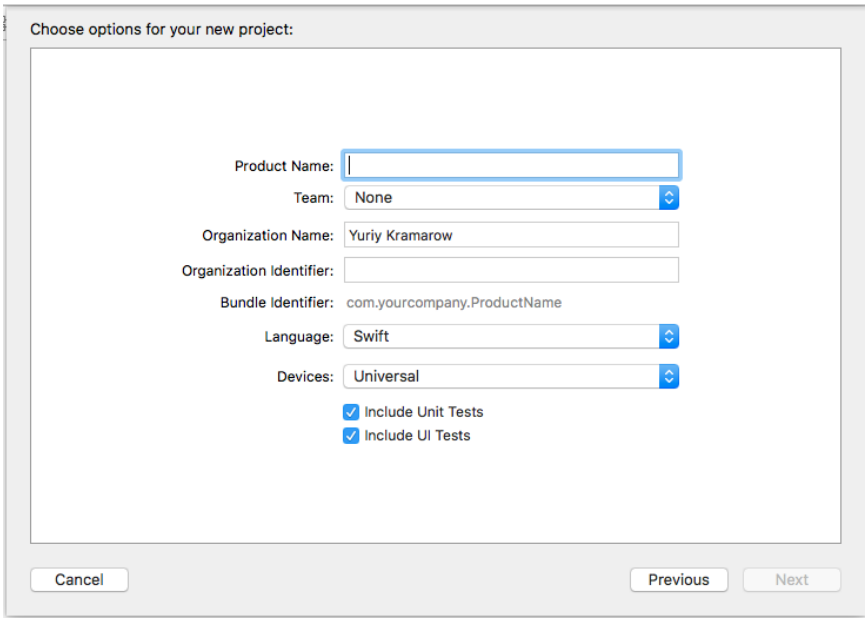


Рис. 3.21. – Вікно створення проекту Xcode

З рисунку 3.21 видно, що Xcode дає змогу розробляти проект цілій команді, обирати мову та тип пристроїв.

3.4.6. Опис коду додатку. Для того, щоб надати застосунку приємного вигляду на його фон замість заливки певним кольором можна встановити зображення з певною текстурою або фотографію. Робиться це за допомогою коду, що наведений на рисунку 3.22. Код усіх файлів проекту показано у додатках Ж-Ф.

```
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
```

Рис 3.22. – Код встановлення фону застосунку з файлу

Іноді з’являється потреба вивести на екран повідомлення чи попередження під час певної дії. Код для виведення повідомлень наведений на рис 3.23.

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

Рис. 3.23. – Код для виведення повідомлення при натисканні на «+»

Необхідною також є перевірка того, чи поле введення порожнє. Приклад такої перевірки показано у коді, який дає змогу отримати назву, яку ввів користувач, але тільки якщо це не порожнє значення.

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

Рис 3.24. – Код для перевірки заповненості поля.

В повідомлення іноді потрібно додавати текстові поля для введення чи кнопки. Вони створюються за допомогою коду, показаного на рис. 3.25.

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

Рис. 3.25. – Код додавання текстового поля, кнопки «Відмінити» та «Підтвердити» до повідомлення

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

Рис. 3.26. – Функція для додавання нового елементу в список

В даному коді описана функція, що відповідає за додавання нових елементів у список застосунку.

3.4.7. Додання власних міток на карті за допомогою MapKit. MapKit дозволяє відображати карти та супутникові знімки, призначені для користувача. Структура MapKit надає інтерфейс для вбудовування карти безпосередньо у власні вікна програми. Ця система також забезпечує підтримку аннотацій карти, додавання накладень, а також виконання зворотного геокодування вибірок для визначення міток інформації для заданих координат карти [4].

Для того, щоб позначити будь-яке місце, якого на карті немає є можливість створювати власні мітки, що робиться за допомогою коду, наведеного на рис 3.27. Повний код файлу MapViewController.swift наведено у додатку У.

```
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]))
```

Рис. 3.27. - Встановлення власних міток на карті

У створених мітках є можливість показувати зображення за допомогою масиву.

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

Рис. 3.29. – Код для показу зображень в мітках

Для використання даних геолокації, користувача необхідно про це попередити та запросити доступ, робиться це за допомогою коду, показаного на рис. 3.29.

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

Рис. 3.29. - Запит доступу до даних геолокації

За допомогою коду, наведеного на рис. 3.30. 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
  }
```

Рис. 3.30. - Кнопка для переключення типу карти

Щоб швидко знайти місця поблизу можна відразу переміститись до місця розташування користувача, що робиться за допомогою коду:

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

Рис. 3.31. – Кнопка повернення до локації користувача

Керувати картою можна не тільки за допомогою екранних жестів, але й за допомогою кнопок, створення яких показано на рис. 3.32.

```
@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)
  }
```

Рис. 3.32. – Код створення кнопок управління картою.

Вище наведений код відповідає за створення наекранних кнопок для збільшення/зменшення карти, зміну типу карти та повернення до місцезнаходження користувача.

3.4.8. Додання інформації «ТОП250» з сайту IMDb. База даних фільмів в Інтернеті (Internet Movie Database, IMDb) – найбільша база даних та веб-сайт про кінематограф. База даних в більшості поповнюється добровольцями, це чимось нагадує принцип роботи вікі.

Також на сайті є стрічка новин, яка постійно оновлюється статтями, а система персональних рекомендацій допомагає підібрати найкращий фільм за жанром, новизною чи на основі останніх фільмів, які переглядав користувач.

У базі зараз зібрана детальна інформація про більш ніж 3 мільйони кінострічок і телесеріалів, є дані майже про 6,9 мільйонів акторів, режисерів та інших професіоналів кіно зі всього світу. Скріншот сайту зображено на рисунку 3.33.

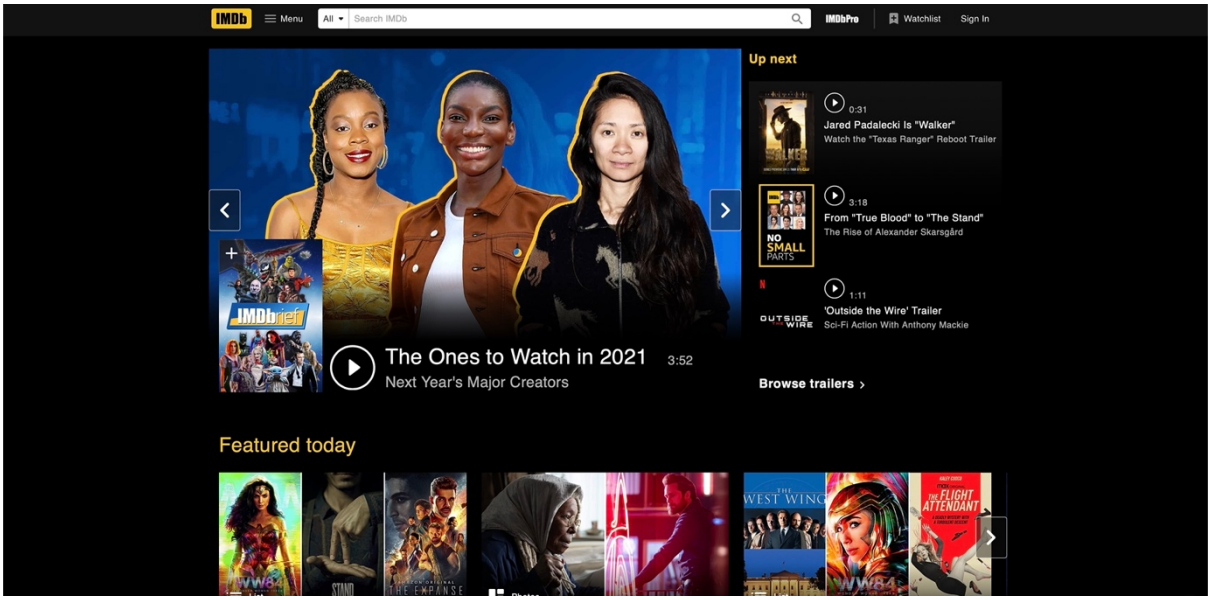


Рис. 3.33. Скріншот сайту IMDb

З 1998 року базою даних IMDb володіє компанія “Amazon.com”. Майже уся інформація IMDb знаходиться в безкоштовному доступі, сайт IMDb.com функціонує на основі безкоштовного програмного забезпечення (Apache, Perl та ін.). За даними рейтингового агентства ”Alexa”, сайт imdb.com постійно входить в сотню найпопулярніших сайтів мережі Інтернет. Основна мова інтерфейсу сайту – англійська, українського інтерфейсу сайт не має.

Для того, щоб отримати дані про список ТОП250 з сайту IMDb.com використовується код, що наведений у на рис. 3.34.

```
class TopList:
UIViewController,UITableViewDataSource,UITableViewDelegate {
    final let urlString = "https://api.myjson.com/bins/pvitx"
    //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()
    }
      func downloadJsonWithTask() {
let url = NSURL(string: urlString)
var downloadTask = URLRequest(url: (url as? URL)!,
cachePolicy: URLRequest.CachePolicy.returnCacheDataElseLoad,
timeoutInterval: 60)
downloadTask.httpMethod = "GET"
URLSession.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")!)
```

Рис. 3.34. – Частина коду з файлу TopList.swift

Повний код файлу TopList.swift показано у Додаткуку М. Даний файл відповідає за отримання даних з сайту IMDb.com та представлення їх у застосунку.

3.4.9. Оцінювання та аналіз результатів

Отримано застосунок “FilmList”, готовий для використання користувачами. Застосунок задовільняє всі вимоги, які були поставлені на етапі постановки завдання.

Було проведено дослідження компонентів програмного середовища Xcode 10, яке використовувалось для створення застосунку.

Застосунок представляє собою чотири вкладки з графічним інтерфейсом, що мають спеціальне призначення. При запуску застосунку відкривається головна вкладка з списком фільмів.

Перевагою застосунку є те, що він досить простий в експлуатації і має дуже низькі системні вимоги, але як будь-який застосунок він може бути покращений за рахунок допрацювання.

Використання засобів програмування дало змогу справитись з поставленою задачею. У середовищі програмування Xcode 10 є всі необхідні інструменти для того, щоб створювати повноцінні програми та мобільні застосунки.

Елементи Xcode дали змогу створити гнучкий графічний інтерфейс для поставленої задачі за допомогою використання засобів візуального програмування.

3.4.10. Графічний редактор Adobe Photoshop CC. Adobe Photoshop CC використовувався для створення попереднього макету застосунку та більшості елементів дизайну, а саме для фону, іконки застосунку та кнопок.

Adobe Photoshop – графічний редактор, розроблений і поширюваний фірмою Adobe Systems. Цей продукт є лідером ринку в області комерційних засобів редагування растрових зображень, і найвідомішим продуктом фірми Adobe. Зараз редактор доступний на платформах Apple Mac OS, Microsoft Windows, iOS та Android. Основне його вікно зображене на рис. 3.35.

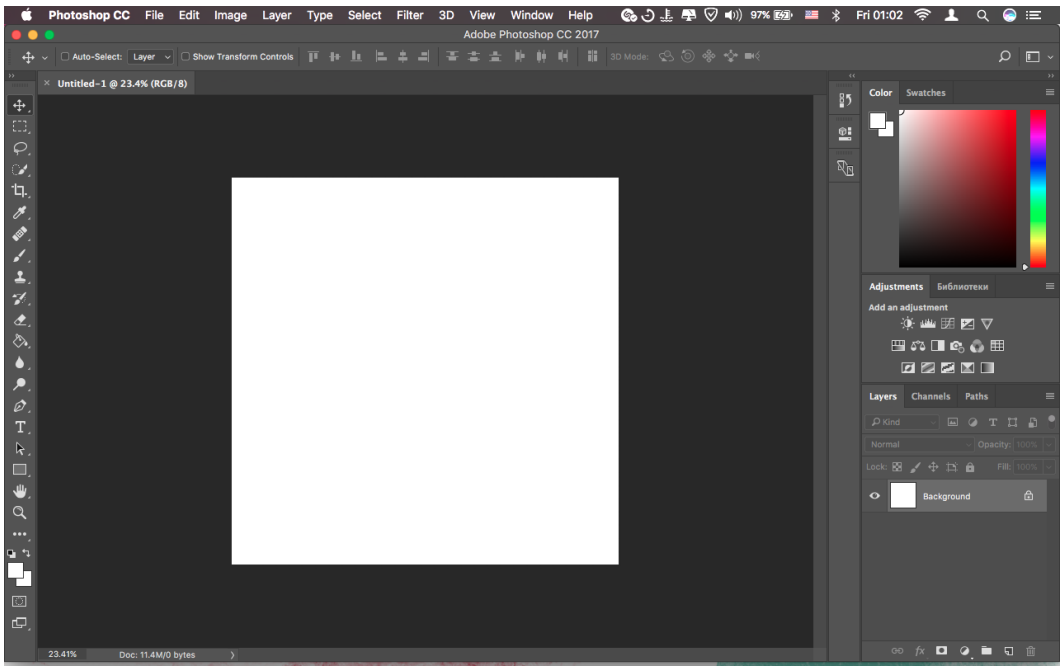


Рис. 3.35. Основне вікно Adobe Photoshop

Деякі функції Photoshop включають в себе:

– висока якість обробки графічних зображень;

– гнучкість інтерфейсу користуввача;

– зручність і простота в експлуатації;

– великі можливості, які надають змогу виконувати будь-які операції створення і обробки та редагування зображень;

– широкі можливості автоматичної обробки растрових зображень, за допомогою використання сценаріїв;

– сучасний механізм роботи з кольоровими профілями, які допускають їх втілення в файли зображень з метою автоматичної корекції кольорових параметрів при виводі на друк для різних пристроїв;

– великий набір фільтрів, за допомогою яких можна створювати різні художні ефекти та сцени.

3.5 Висновки до розділу 3

В цьому розділі було досліджено методи реалізації мультифункціональної платформи для планування відпочинку

Побудовано блок-схему застосунку, дерево проблем, дерево цілей, концептуальну модель та діаграму прецедентів.

Головною проблемою є власне сама розробка, яка поділяється вже на більшу кількість менших проблем: побудова списку, створення карти та впровадження IMDb API та інші. З наявним деревом проблем, ми будуємо дерево цілей, досягнення яких дасть можливість отримати бажаний програмний продукт з необхідними функціями.

Досліджено методологію проектування інформаційних систем RAD (Rapid Application Development). Така методологія ґрунтується на великій кількості прототипів, які на кожній ітерації аналізуються і обговорюються з замовником, для продовження розробки. Кожен прототип в подальшому розвивається в частину майбутньої системи.

Досліджено створення діаграми класів та концептуальної моделі ІС. Діаграма класів показує класи, які утворюють загальну систему, а також взаємозв’язки між ними.

Розглянуто етап тестування, пілотного впровадження системи та публікація застосунку в магазині Apple AppStore.

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

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.

Після розробки та всіх необхідних тестів розробник має змогу купити «Профіль розробника Apple», щоб отримати можливість публікувати свої додатки в Apple AppStore.

Визначено вимоги до програмного та апаратного забезпечення.

Проект розроблявся для iPhone і iPad з версією iOS не нижче 12, хоча є можливість розробляти для версії від iOS 8, але така розробка буде мати ряд недоліків.

Проведено розробку додатку, детально описано його елементи коду та середовище розробки. Окремо розглянуто впровадження IMDb API та Apple MapKit.

Розглянуто графічний редактор Adobe Photoshop, який використовувався для створення прототипу, елементів інтерфейсу та іконки додатку.

РОЗДІЛ 4

ОХОРОНА ПРАЦІ ТА БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ

4.1 Охорона праці та безпека в надзвичаних ситуаціях

Державні нормативно-правові акти з охорони праці (ДНАОП) -– це правила, стандарти, норми, регламенти, положення, інструкції та інші документи, яким надано чинність правових норм, обов'язкових для виконання.

За сферою дії ДНАОП можуть бути міжгалузевими, галузевими та актами підприємств.

Державний міжгалузевий нормативний акт про охорону праці — це ДНАОП загальнодержавного користування, дія якого поширюється на всі підприємства, установи, організації народного господарства України незалежно від їх відомчої (галузевої) належності та форм власності.

Державний галузевий нормативний акт про охорону праці — це ДНАОП, дія якого поширюється на підприємства, установи і організації незалежно від форм власності, що належать до певної галузі.

Нормативна база України з охорони праці узагальнена і систематизована у „Державному реєстрі міжгалузевих та галузевих актів про охорону праці”. Реєстр включає в себе понад 2000 нормативних актів, а також 350 міждержавних стандартів безпеки праці та близько 40 державних стандартів України (ДСТУ).

Відповідно до вимог ДНАОП власники підприємств, установ, організацій або уповноважені ними органи розробляють і затверджують власні положення, інструкції або інші нормативні акти про охорону праці, що діють у межах підприємства, установи, організації. Нормативною підставою для цього є «Порядок опрацювання і затвердження власником нормативних актів про охорону праці, що діють на підприємстві», згідно якому до основних нормативних актів підприємства належать, наприклад:

– Положення про систему управління охороною праці на підприємстві;

– Положення про службу охорони праці підприємства;

– Положення про комісію з питань охорони праці підприємства;

– Положення про роботу уповноважених трудового колективу з питань охорони праці;

– Положення про навчання, інструктаж і перевірку знань працівників з питань охорони праці;

– Перелік робіт з підвищеною небезпекою та ін.

Враховуючи специфіку виробництва та вимоги чинного законодавства, власник затверджує нормативно-правові акти, що регламентують питання охорони праці та охоплюють завдання і функції системи управління охороною праці.

У Законі України "Про охорону праці" задекларовані основні принципи державної політики в галузі охорони праці:

– пріоритет життя і здоров'я працівників щодо результатів виробничої діяльності підприємства;

– повна відповідальність роботодавця за створення належних, безпечних і здорових умов праці;

– підвищення рівня промислової безпеки шляхом забезпечення суцільного технічного контролю за станом виробництв, технологій та продукції;

– обов'язковий соціальний захист працівників, повне відшкодування шкоди особам, які потерпіли від нещасних випадків на виробництві та професійних захворювань;

– використання економічних методів управління охороною праці;

– комплексне розв'язання завдань охорони праці на основі загальнодержавної, галузевих, регіональних програм з цього питання та з урахуванням інших напрямків економічної та соціальної політики, досягнень у галузі науки і техніки та охорони довкілля;

– запровадження єдиних нормативів з охорони праці для всіх підприємств та суб'єктів підприємницької діяльності незалежно від форм власності й виду діяльності;

– інформування населення, проведення навчання, професійної підготовки і підвищення кваліфікації працівників з питань охорони праці;

– співробітництво і проведення консультацій між роботодавцями та працівниками (їх представниками), між усіма соціальними групами під час прийняття рішень з охорони праці;

– міжнародне співробітництво в галузі охорони праці, використання світового досвіду організації роботи щодо поліпшення умов і підвищення безпеки праці.

Для реалізації цих принципів було створено Національну раду з питань безпечної життєдіяльності населення при Кабінеті Міністрів України, Держгірпромнагляд та його територіальні органи, Фонд соціального страхування від нещасних випадків, Національний науково-дослідний інститут промислової безпеки та охорони праці, навчально-методичні центри.

Відповідно до Закону України "Про охорону праці" за порушення законів та інших нормативно-правових актів з охорони праці, створення перешкод у діяльності посадових осіб органів державного нагляду за охороною праці, а також представників профспілок, їх організацій та об'єднань винні особи притягаються до дисциплінарної, адміністративної, матеріальної, кримінальної відповідальності згідно із законом.

4.2 Безпека в надзвичайних ситуаціях

Безпека – це збалансований, за експертною оцінкою, стан людини, соціуму, держави, природних, антропогенних систем тощо.

Надзвичайна ситуація – порушення нормальних умов життя і діяльності людей на об'єкті або території, спричинене аварією, катастрофою, стихійним лихом, епідемією, епізоотією, епіфітотією, великою пожежею, застосуванням за собі в ураження, що призвели або можуть призвести до людських і матеріальних втрат.

Основою життєдіяльності є чинники і параметри навколишнього середовища (сонця, повітря, води, ґрунту, біосфери) та штучного середовища життя (житлові та виробничі будівлі, споруди, транспортні та повітряні комунікації, системи забезпечення енергоресурсами, продуктами харчування) і багато іншого, що створено руками людини для забезпечення життя.

Загальнодержавні, відомчі і територіальні заходи організаційно-економічного характеру запобігання та ліквідації наслідків надзвичайних ситуацій.

Запобігання виникненню надзвичайних ситуацій — це підготовка та реалізація комплексу правових, соціально-економічних, політичних, організаційно-технічних, санітарно-гігієнічних та інших заходів, спрямованих на регулювання безпеки, проведення оцінки рівнів ризику, завчасне реагування на загрозу виникнення надзвичайної ситуації на основі даних моніторингу (спостережень), експертизи, досліджень та прогнозів щодо можливого перебігу подій із метою недопущення їх переростання у надзвичайну ситуацію або пом'якшення її можливих наслідків.

Ліквідація наслідків надзвичайної ситуації проводиться з метою відновлення роботи підприємства організації, навчальних закладів тощо. Вона включає:

– розвідку осередків надзвичайних ситуацій;
– аварійно-рятувальні й лікувально-евакуаційні заходи;
– локалізацію й гасіння пожеж;
– відбудову споруд і шляхів сполучення;
– проведення ізоляційно-обмежувальних заходів в осередках біологічного зараження;
– проведення спеціальної обробки населення;
– дезактивації, дегазації техніки, доріг, місцевості тощо.

Алгоритм класифікації надзвичайних ситуацій:

Етап - віднесення події за пороговим значенням до надзвичайної ситуації

Етап - класифікація її за походженням:

– техногенного характеру,
– природного характеру,
– соціально-політичного характеру
– воєнного характеру

Етап - класифікація її за рівнем:

– Загальнодержавного
– Регіонального
– Місцевого
– Об’єктового.

Найбільш небезпечні надзвичайні ситуації природного характеру в Україні.

На території України можливе виникнення практично всього спектру небезпечних природних явищ і процесів геологічного, гідрогеологічного та метеорологічного походження. До них належать великі повені, катастрофічні затоплення, землетруси та зсувні процеси, лісові та польові пожежі, великі снігопади та ожеледі, урагани, смерчі та шквальні вітри тощо.

Серед надзвичайних ситуацій природного походження в Україні найчастіше трапляються:

– Геологічна небезпечні явища, такі, як зсуви, обвали та осипи, просадки земної поверхні різного походження та ін.;

– Метеорологічна небезпечні явища, такі, як зливи, урагани, сильні снігопади, сильний град, ожеледь;

– Гідрологічна небезпечні явища, такі, як повені, паводки, підвищення рівня ґрунтових вод та ін.;

– Природні пожежі лісових та хлібних масивів;

– Масові інфекції та хвороби людей, тварин і рослин.

Оцінка нанесеного збитку від надзвичайних ситуацій.

Часто оцінка нанесеного збитку може виражатися не тільки шкодою, завданою життю і здоров’ю людини, але і іншими втратами, наприклад, матеріальними.

Оцінка обстановки – порядок визначення ступеню ураженості об’єкта чи території, можливих об’ємів завданих збитків та вплив вторинних факторів на проведення рятувальних та інших невідкладних робіт в осередку ураження від надзвичайних ситуацій.

4.3 Вплив виробничого середовища на працездатність та здоров’я користувачів комп’ютерів

Стомлення і перевтома. Будь-яка діяльність, якщо вона оптимальна для організму по інтенсивності і тривалості та проходить у сприятливих виробничих умовах, позитивно впливає на організм і сприяє його удосконалюванню.

Ефективність діяльності людини базується на рівні психічної напруги, яка прямо пропорційна складності завдання.

Психічна напруга - це фізіологічна реакція організму, яка мобілізує його ресурси (біологічно і соціально корисна реакція). Під впливом психічної напруги змінюються життєво важливі функції організму: обмін речовин, кровообіг, дихання. В поведінці людини спостерігається загальна зібраність, дії стають більш чіткими, підвищується швидкість рухових реакцій, зростає фізична працездатність. При цьому загострюється сприйняття, прискорюється процес мислення, поліпшується пам'ять, підвищується концентрація уваги.

При надмірній інтенсивності чи тривалості робота приводить до розвитку вираженого стомлення, зниження продуктивності, неповного відновлення за період відпочинку. Стомлення - загальний фізіологічний процес, яким супроводжуються усі види активної діяльності людини. З біологічної точки зору стомлення - це тимчасове погіршення функціонального стану організму людини, що виявляється в змінах фізіологічних функцій і є захисною реакцією організму. Відомо, що розвиток втоми та перевтоми веде до порушення координації рухів, зорових розладів, неуважності, втрати пильності та контролю реальної ситуації. Втома породжує у працівника стан, який призводить до помилок в роботі, небезпечним ситуаціям і нещасним випадкам.

Робочий стрес - хворобливий психоемоційний стан, що утворюється в процесі праці. Поняття охоплює великий набір розладів, включаючи психічні захворювання (депресію, тривогу) та інші типи емоційних розладів (незадоволеність, стомлення, стрес, і так далі), неадекватна поведінка (агресія, наркоманія), порушення пам'яті або концентрації. Ці порушення можуть призвести до незадовільного виконання працівником своїх обов'язків і завдати шкоди його здоров'ю. Робочий стрес - важлива проблема сучасного робочого місця. До його схильні близько третини працівників.

Причиною робочого стресу є умови праці. Питання про те, що має більший вплив - умови роботи або особисті характеристики працівника, є спірним. Різні відповіді на 26 це питання породжують різні способи вирішення проблеми. Якщо

вважати, що персональні особливості важливіше, то на перший план виходять пристосовуваність і навички спілкування.

Робота на ПК під час виконання виробничих завдань, за умови дотримання вимог ДСанПіН 3.3.2-007-98 "Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин", НПАОП 0.00-1.28-10 "Правила охорони праці під час експлуатації електронно-обчислювальних машин", не належить до категорії шкідливих і важких. Але у користувачів, які інтенсивно використовують ПК в умовах значних розумових напружень досить часто (40—70%) виникають психологічні та поведінкові порушення (нервозність, роздратування, тривога, нерішучість, замкнутість тощо). Серед користувачів ВДТ в США і Європі значного поширення набуло специфічне 69 захворювання, яке отримало назву синдром комп'ютерного стресу (СКС). СКС супроводжується головним болем, запаленням очей, алергією, роздратованістю, млявістю і депресією.

Постійне і значне напруження функцій зорового аналізатора, обумовлене необхідністю розрізнення об'єктів (символів, знаків, ліній, зображень і т.п.) в умовах:

– нечіткого зображення та мерехтіння на екрані;
– недостатньої освітленості поля екрану;
– недостатньою контрастності об'єктів розрізнення;
– постійна преадаптація зорового апарату до різних рівнів освітленості екрану, оригіналу, клавіатури.

Нечітке зображення та мерехтіння на екрані збільшують імовірність порушення функції зору. Користувач може навіть звикнути до незначного мерехтіння тексту чи картинки, однак очі автоматично реагують на нього. Напружуються зорові нерви та відповідні зорові центри кори головного мозку, при цьому гострота зору неминуче знижується.

Особливості інформаційної взаємодії людини та ПК. Діяльність, коли в процесі роботи необхідно приймати рішення за відсутності заздалегідь відомого алгоритму, оперативне очікування інформації, необхідної для прийняття відповідних рішень та збої в її отриманні, психологічні особливості роботи

оператора (пошук, отримання, введення та обробка інформації), пов'язані з емоційно-вольовою сферою, спричинює виникнення стресових ситуацій у користувача ПК. В умовах значних розумових напружень досить часто виникають психологічні та поведінкові порушення (нервозність, тривога, нерішучість) і як наслідок нездатність прийняти правильне рішення. Монотонність і гіподинамія. Спостереження за діяльністю користувачів ПК, показує, що гіподинамія (періодичне перебування в незручній позі (робота з нахилом або поворотом тулуба, незручним розташуванням кінцівок та у фіксованій позі (неможливість зміни взаємного розташування різних частин тіла відносно одна одної) і праця з особливо тривалою монотонністю (стереотипні робочі рухи при локальному навантаженні (за участю м'язів кистей та пальців рук) – спричиняє погіршання мозкового кровообігу у вигляді підвищеної загальної втоми, порушення концентрації уваги, точності і координації рухів.

Порушення функцій репродуктивної системи користувачів ПК.

Праця за комп'ютером є шкідливою для вагітних жінок, особливо в перші три місяці вагітності. Робота за комп'ютером порушує нормальний перебіг вагітності, підвищує імовірність спонтанного аборту, може бути причиною появи на світ дітей з вродженими вадами, із них найбільш суттєвими бувають дефекти розвитку головного мозку. При цьому якщо раніше вважалось що причиною є електромагнітні поля ВДТ відповідної інтенсивності які здатні змінювати і переривати клітинний розвиток, то в теперішній час все більше спеціалістів приходять до висновку що на жінку впливає весь комплекс діючих факторів, включаючи тривале перебування у незмінній позі, напруження скелетно-м'язової системи і нервово емоційні перевантаження при роботі на ПК. Крім порушення функцій чотирьох найбільш вразливих систем організму людини при роботі на ПК впливу зазнають і інші системи організму (слуховий аналізатор, серцево-судинна система, підвищується схильність до вірусних і багатьох інфекційних захворювань та хвороб органів травлення). До числа факторів, що погіршують стан здоров'я користувачів, слід віднести електромагнітне і електростатичні поля, акустичний шум, зміна іонного складу повітря та параметрів мікроклімату в приміщенні. На стан користувачів роблять вплив і 73 ергономічні параметри розташування екрану

монітора, які ведуть до зміни контрастності зображення в умовах інтенсивної засвічення, появі дзеркальних відблисків від передньої поверхні екрана монітора і т.д. Важливу роль грає і стан освітленості на робочому місці, параметри меблів і характеристики приміщення, де розташована комп'ютерна техніка. Враховуючи вищесказане можливо зробити висновок що суттєве покращення умов праці користувачів ПК можна досягти шляхом врахування:

- параметрів приміщень та взаємного розташування робочих місць;
- параметрів виробничого середовища приміщень;
- параметрів обладнання та організації робочих місць;
- режимів праці і відпочинку користувачів ПК;
- стану здоров‘я користувача ПК.

4.4 Висновки до розділу 4

Проаналізовано вимоги з охорони праці та безпеки в надзвичайних ситуаціях, що дало змогу визначити шляхи мінімізації негативного впливу комп’ютерної техніки на користувачів програмного засобу виявлення.

Досліджено питання безпеки в надзвичайних ситуацій та запобігання їх винекнення.

Розглянуто найбільш небезпечні надзвичайні ситуації на території України, до яких належать великі повені, катастрофічні затоплення, землетруси та зсувні процеси, лісові та польові пожежі, великі снігопади та ожеледі, урагани, смерчі та шквальні вітри тощо.

Досліджено вплив виробничого середовища на працездатність та здоров’я користувачів комп’ютерів.

Розглянуто такі причини стомлення і перевтоми:

- психічна напруга
- інтенсивність роботи
- тривалість роботи
- робочий стрес
- умови праці

- постійне напруження зору
- монотонність
- гіподинамія

Надмірна робота з ПК може привезти до порушення функції репродуктивної системи користувача через електромагнітні та електростатичні поля, акустичний шум, зміну іонного складу повітря та параметрів мікроклімату в приміщенні, а також ергономічні параметри розташування екрану монітора.

ВИСНОВКИ

Завдання, поставлене на магістерську роботу було виконано в повному обсязі.

Проведено аналіз літературних джерел за темою магістерської роботи, побудовано дерево проблем та дерево цілей.

Досліджено компоненти програмного мови програмування. Swift середовища Xcode 10, яке використовувалось для створення застосунку.

Виконано проектування застосунку та написання коду.

Розроблено застосунок для мобільної платформи iOS, готовий для використання користувачами. Застосунок задовільняє всі вимоги, які були поставлені на етапі постановки завдання.

Застосунок можна реалізувати в магазині «App Store», як безкоштовно, так і за встановлену ціну.

Проведено дослідження можливості оновлення. Застосунок може в майбутньому вдосконалюватись, оновлення будуть поширюватись через стандатний магазин iOS.

Також в перспективі є можливість співпрацювати з різними мережами кінотеатрів, для впровадження магазину квитків в застосунку та розкладу сеансів.

Написання магістерської роботи дало багато досвіду в розробці мобільного програмного забезпечення, вивчення нового матеріалу, вдосконалення вмінь та навичок роботи із об’єктно-орієнтовним програмуванням в середовищі Xcode 10.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Бойчик І.М. Економіка підприємства: Навч. посіб. – К.: Атака, 2002. – 480 с. – ISBN 0-596-60234-4;

2. Кардаш В.Я. Маркетингова товарна політика: Підручник. – К.: КНЕУ, 2001. – 240 с. – ISBN 0-596-30915-3;

3. Язык UML. Руководство пользователю / Грейди Буч, Джеймс Рамбо, Айвар Джекобсон. 2-е изд., Питер 2004.

4. Apple Inc. The Swift Programming Language: Навч. посіб. – 1 Infinite Loop Cupertino, CA 95014 – 1127 с.

5. MapKit | Apple Developer Documentation [Електронний ресурс]. https://developer.apple.com/reference/mapkit

6. Start Developing iOS Apps (Swift): Jump Right In – Apple Developer [Електронний ресурс]. https://developer.apple.com/ library/content/referencelibrary/GettingStarted/DevelopiOSAppsSwift/

7. iOS – Википедия [Електронний ресурс]. https://ru.wikipedia.org/wiki/IOS

8. Xcode – Википедия [Електронний ресурс]. https://ru.wikipedia.org/wiki/Xcode

9. Swift (язык программирования) – Википедия [Електронний ресурс]. https://ru.wikipedia.org/wiki/Swift_ (язык_программирования)

10. Як створюти мобільні додатки самостійно [Електронний ресурс]. http://hi–news.pp.ua/kompyuteri/174–mobln–yak–stvoryuvati–dodatki–dlya–iphone–android–samostyno.html

11. Що краще Android або iOS? [Електронний ресурс]. http://ukr–article.com/index.php?newsid=2235

12. Розробка мобільних додатків [Електронний ресурс]. http://voroninstudio.eu/uk/service/razrabotka–mobilnih–prilozheniy

13. Що таке мобільний додаток [Електронний ресурс]. http://ladyfacts.xyz/hi–tech/mobilni–telefoni/1069–shho–take–mobilnij–dodatok.html

14. Розробка додатків для мобільних пристроїв – Вікіпедія [Електронний ресурс]. https://uk.wikipedia.org/wiki/Розробка_ додатків_для_мобільних_пристроїв

15. Розробка мобільних додатків [Електронний ресурс]. Режим доступу: https://nakitel.com/poslugy/mobilni–dodatky/

16. Adobe Photoshop – Википедия [Електронний ресурс]. https://ru.wikipedia.org/wiki/Adobe_Photoshop

17. InCube відкрита регіональна платформа [Електронний ресурс]. http://www.incubeplatform.com.ua/project-viewer/show/38

18. Методологія RAD [Електронний ресурс]. https://studopedia.ru/7_144443_metodologiya-RAD.html

19. Тенденции и перспективы рынка мобильных приложений: поговорим о деньгах [Електронний ресурс]. https://habr.com/company/alconost/blog/323020/

20. Microsoft Object [Електронний ресурс]. https://uk.wikipedia.org/wiki/Microsoft_Project

ДОДАТКИ

ДОДАТКИ

Додаток А

Тексти наукових публікацій

УДК 004.4’23
Крамаров Ю. – ст. гр. СІм-61
Тернопільський національний технічний університет імені Івана Пулюя

СТВОРЕННЯ ВЛАСНОЇ КАРТИ ЗА ДОПОМОГОЮ APPLE MAPKIT

Науковий керівник: к.т.н., доцент Яцишин В.В.

Kramarov Y.
Ternopil Ivan Puluj National Technical University

CREATING CUSTOM MAP WITH APPLE MAPKIT

Supervisor: PhD, Assoc. Prof. Yatsyshyn V.

Ключові слова: апаратне забезпечення, надійність, ефективність

Keywords: hardware, reliability, efficiency

MapKit дозволяє відображати карти та супутникові знімки, призначені для користувача. Структура MapKit надає інтерфейс для вбудовування карти безпосередньо у власні вікна програми. Ця система також забезпечує підтримку анотацій карти, додавання накладень, а також виконання зворотного геокодування вибірок для визначення міток інформації для заданих координат карти.

Для того, щоб позначити будь-яке місце, якого немає на карті існує можливість створювати власні мітки, що робиться за допомогою коду:.

```
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]))
```

У створених мітках є можливість показувати зображення за допомогою масиву.

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

Для використання даних геолокації, користувача необхідно про це попередити та запросити доступ, робиться це за допомогою коду, наведеного нижче

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

Кнопки переключення з карти на супутник реалізорвуються за допомогою коду:

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

Щоб швидко знайти місця поблизу, можна відразу переміститись до місця розташування користувача, що робиться за допомогою коду:

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

Створення коду для керування картою:

```
@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)
  }
```

Вище наведений код відповідає за створення наекранних кнопок для збільшення/зменшення карти, зміну типу карти та повернення до місцезнаходження користувача.

УДК 004.4’23
Крамаров Ю. – ст. гр. СІм-61
Тернопільський національний технічний університет імені Івана Пулюя

ІНТЕГРОВАНЕ СЕРЕДОВИЩЕ РОЗРОБКИ APPLE XCODE

Науковий керівник: к.т.н., доцент Яцишин В.В.

Kramarov Y.
Ternopil Ivan Puluj National Technical University

INTEGRATED DEVELOPMENT ENVIRONMENT APPLE XCODE

Supervisor: PhD, Assoc. Prof. Yatsyshyn V.

Ключові слова: апаратне забезпечення, надійність, ефективність
Keywords: hardware, reliability, efficiency

Інтегроване середовище розробки (IDE) представляє собою комплексний інструмент для розробки програмного забезпечення. Зазвичай до складу IDE входить редактор початкового коду, засоби автоматизації, тестування та відлагодження програм. Більшість сучасних середовищ розробки мають можливість автодоповнення коду.

Середовища розробки можуть включати компілятор, інтерпретатор (Eclipse) або одночасно обидва. Деякі з них містять систему керування версіями або інструменти для полегшення розробки графічного середовища користувача (GUI) (Xcode, Embarcadero, Delphi). Багато сучасних IDE містять інспектор класів, інспектор об’єктів, схему ієрархії класів для полегшення об’єктно-орієнтованої розробки програмного забезпечення.

Xcode – інтегроване середовище розробки (IDE) виробництва Apple. Дозволяє створювати програмне забезпечення з використанням таких технологій як GCC, GDB, Java та ін. На сьогодні є єдиним засобом написання “універсальних” (Universal Binary) прикладних програм для Mac OS X.

Xcode включає в себе більшу частину документації розробника від Apple та Interface Builder – застосування, яке використовується для створення графічних інтерфейсів.

До складу пакету Xcode входить змінена версія вільного набору компіляторів GNU Compiler Collection. В Xcode підтримуються мови C, C++, Objective-C, Swift, Java, AppleScript, Python і Ruby з різними моделями програмування, включаючи (але не обмежуючись) Cocoa, Carbon і Java. Сторонніми розробниками реалізована підтримка GNU Pascal, Free Pascal, Ada, C #, Perl, Haskell і D. Пакет Xcode використовує GDB як back-end для відналагоджувача.

Для тестування розробники можуть використовувати вже наявні емулятори всіх версій операційної системи iOS та всіх пристроїв, які її підтримують, що інтегровані у середовище Xcode. Вбудований емулятор iOS дає змогу тестувати розроблений застосунок на найновіших ОС та версіях Mac, iPhone, iPad, iPod, Apple TV, Apple Watch.

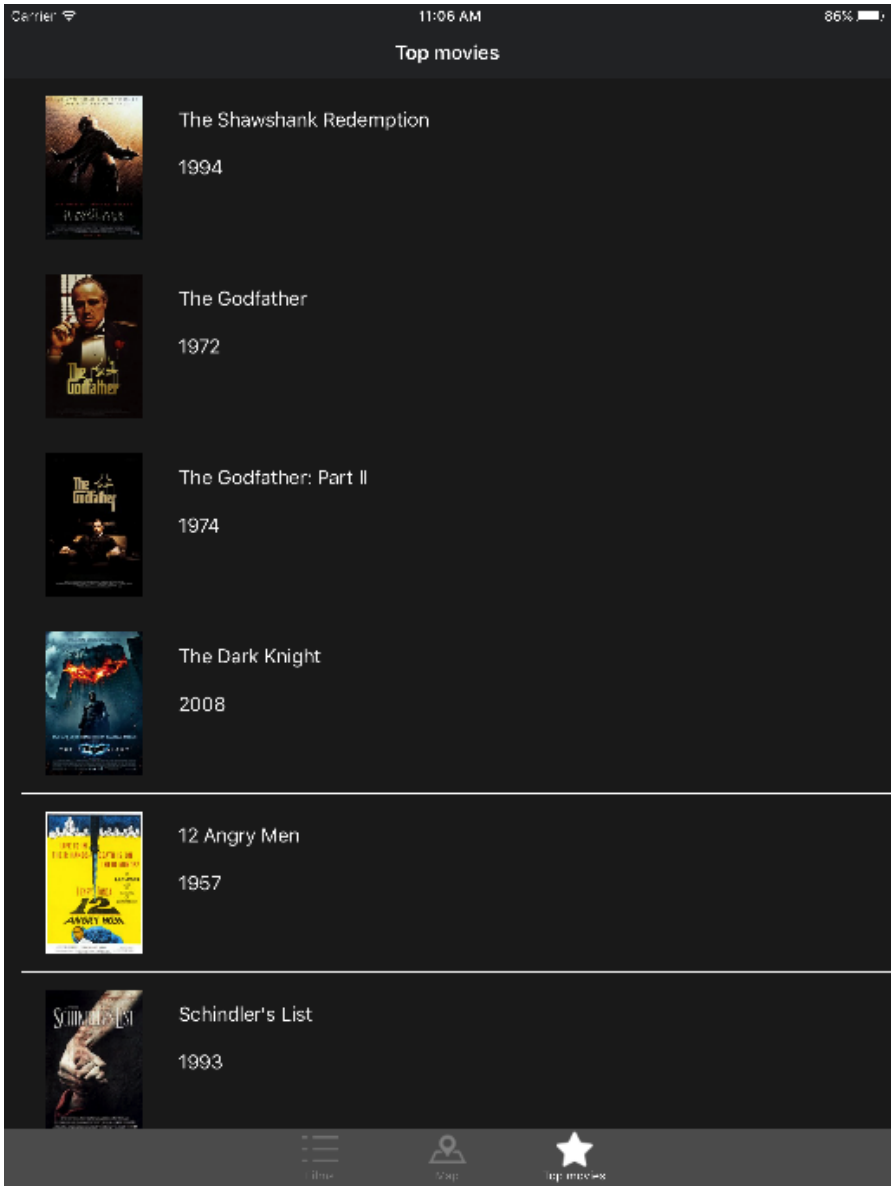
Apple Xcode дає змогу розробляти застосунки на всі версії iOS починаючи від 8.0 та всі пристрої, які її підтримують. Дане середовище можна ефективно використати при розробці мультифункціональної платформи для організації відпочинку.

Додаток Б

Скріншот iPad – версії додатку

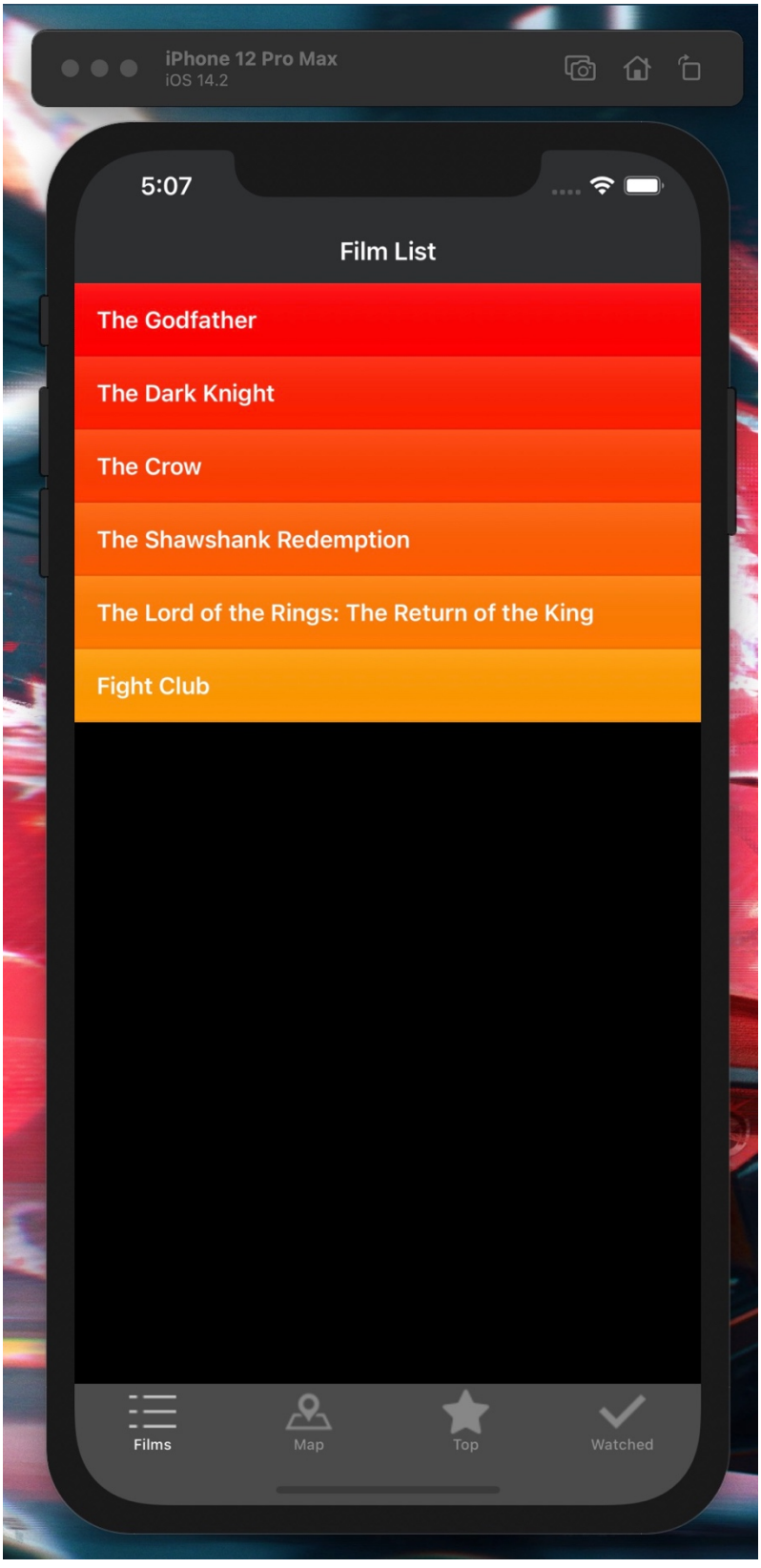
Додаток В

Скріншот iPad – версії додатку



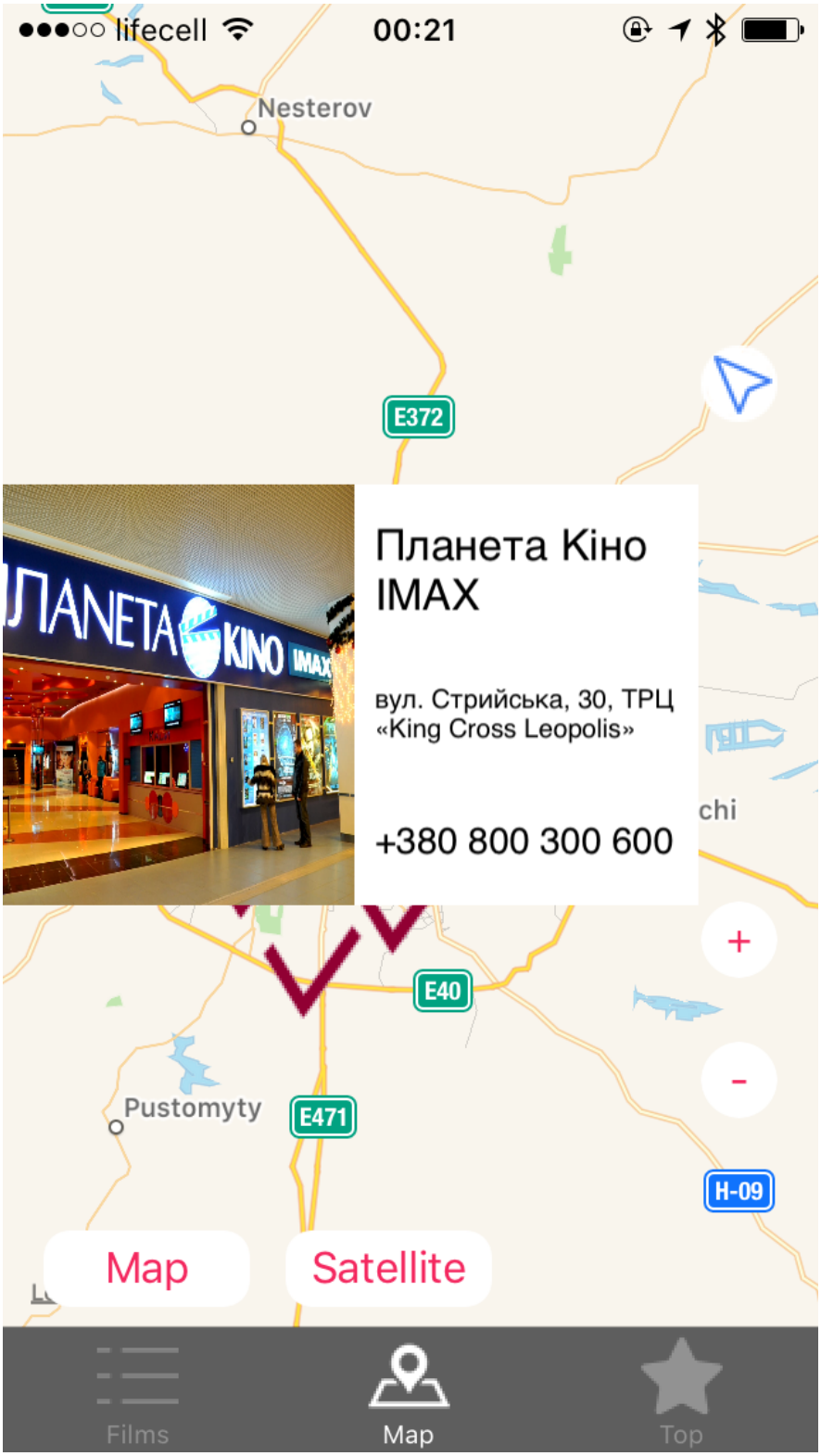
Додаток Г

Скріншот iPhone – версії додатку



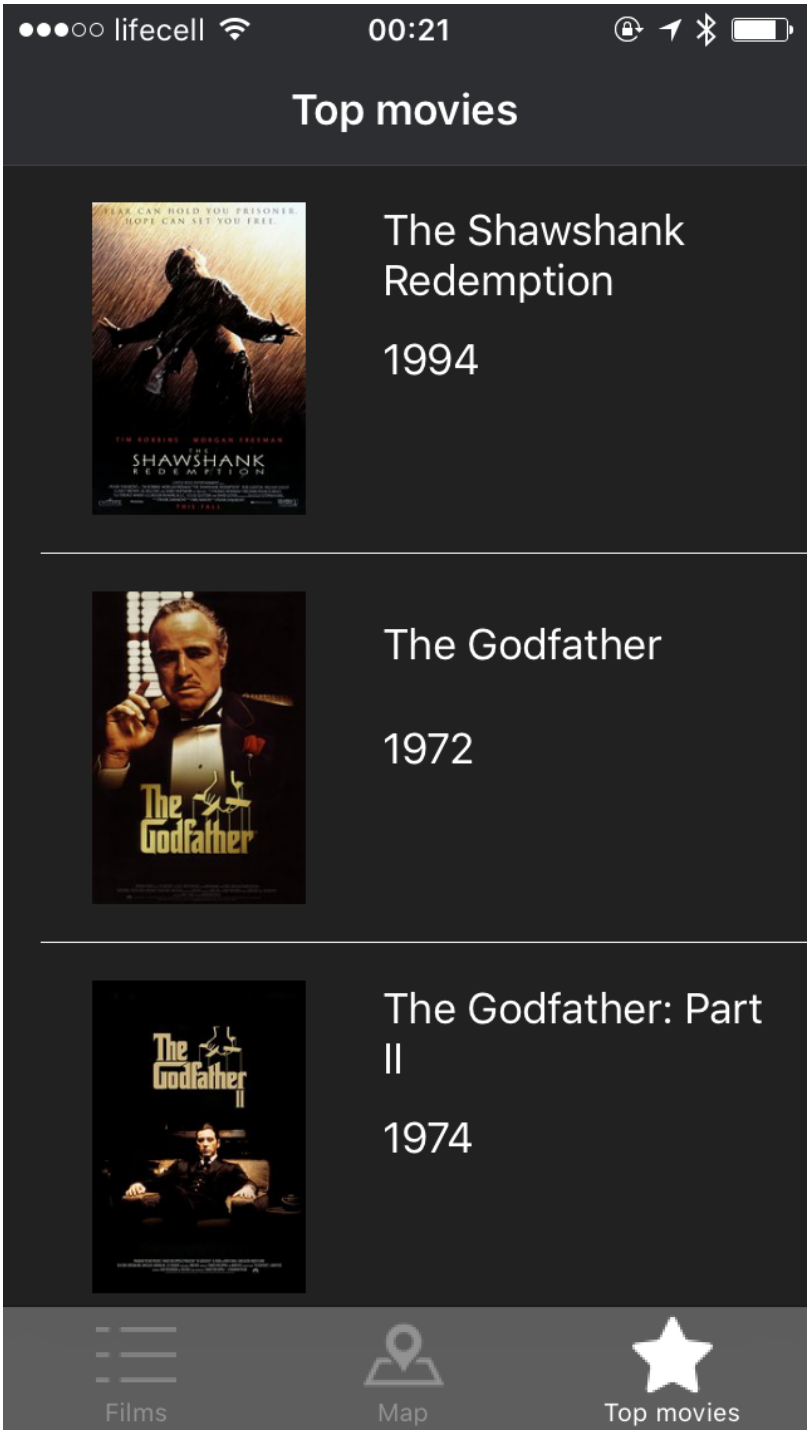
Додаток Д

Скріншот iPhone – версії додатку



Додаток Е

Скріншот iPhone – версії додатку



Додаток Ж
Код файлу tableViewController.swift

```
import UIKit

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, not
necesserially 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(FilmViewCell.self,
forCellReuseIdentifier: "cell")
        tableView.separatorStyle = .none
        tableView.backgroundColor = UIColor.black
```

```
  tableView.rowHeight = 50
if 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 Return
of 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.clear
        let item = filmItems[indexPath.row]
        //            cell.textLabel?.text = item.text
        cell.delegate = self
        cell.filmItem = item
        return 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..<filmItems.count {
            if filmItems[i] === filmItem {
                index = i
                break
            }
        }
                filmItems.remove(at: index)
                       let visibleCells = tableView.visibleCells as!
[FilmViewCell]
        let lastView = visibleCells[visibleCells.count - 1] as
FilmViewCell
        var delay = 0.0
        var startAnimating = false
        for i in 0..<visibleCells.count {
            let cell = visibleCells[i]
            if startAnimating {
                UIView.animate(withDuration: 0.3, delay: delay,
options: .curveEaseInOut,
                                                animations: {() in
                                                 cell.frame =
cell.frame.offsetBy(dx: 0.0, dy: -cell.frame.size.height)},
                                                completion: {(finished:
Bool) in if (cell == lastView) {

self.tableView.reloadData()
                                                 }
                }
                )
                delay += 0.03
            }
            if cell.filmItem === filmItem {
                startAnimating = true
```

```
        cell.isHidden = true
            }
        }
tableView.beginUpdates()
        let indexPathForRow = NSIndexPath(row: index, section: 0)
        tableView.deleteRows(at: [indexPathForRow as IndexPath],
with: .fade)
        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: CGPoint
        var lower: CGPoint
    }
    var upperCellIndex = -100
    var lowerCellIndex = -100
    var initialTouchPoints: TouchPoints!
    var pinchExceededRequiredDistance = false
    var 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..<visibleCells.count {
            let cell = visibleCells[i]
            if viewContainsPoint(view: cell, point:
initialTouchPoints.upper) {
                upperCellIndex = i
            }
            if viewContainsPoint(view: cell, point:
initialTouchPoints.lower) {
                lowerCellIndex = i
            }
        }
        if abs(upperCellIndex - lowerCellIndex) == 1 {
            pinchInProgress = 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..<visibleCells.count {
            let cell = visibleCells[i]
            if i <= upperCellIndex {
                cell.transform = CGAffineTransform(translationX: 0,
y: -delta)
            }
            if i >= lowerCellIndex {
                cell.transform = CGAffineTransform(translationX: 0,
y: delta)
            }
        }
        let gapSize = delta * 2
        let cappedGapSize = min(gapSize, tableView.rowHeight)
        placeHolderCell.transform = CGAffineTransform(scaleX: 1.0,
y: cappedGapSize / tableView.rowHeight)
        placeHolderCell.label.text = gapSize > tableView.rowHeight ?
"Release to add item" : "Pull apart to add item"
        placeHolderCell.alpha = min(1.0, gapSize /
tableView.rowHeight)
        pinchExceededRequiredDistance = gapSize >
tableView.rowHeight
    }
    func pinchEnded(recognizer: UIPinchGestureRecognizer) {
        pinchInProgress = false
```

```
 placeHolderCell.transform = CGAffineTransform.identity
        placeHolderCell.removeFromSuperview()
if pinchExceededRequiredDistance {
            pinchExceededRequiredDistance = false
            let visibleCells = self.tableView.visibleCells as!
[FilmViewCell]
            for cell in visibleCells {
                cell.transform = CGAffineTransform.identity
            }
let indexOffset = Int(floor(tableView.contentOffset.y /
tableView.rowHeight))
            FilmItemAddedAtIndex(index: lowerCellIndex +
indexOffset)
        } else {
            UIView.animate(withDuration: 0.2, delay: 0.0,
options: .curveEaseInOut, animations: {() in
                let visibleCells = self.tableView.visibleCells as!
[FilmViewCell]
                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 = FilmViewCell(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
            }
        }
    }
}
```

Додаток 3

Код файлу FilmViewCell.swift

```
import UIKit
import QuartzCore

protocol TableViewCellDelegate {
    func FilmItemDeleted(filmItem: FilmItem)
    func cellDidBeginEditing(editingCell: FilmViewCell)
    func cellDidEndEditing(editingCell: FilmViewCell)
}
class FilmViewCell: UITableViewCell, 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 = .right
        crossLabel = 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
CGColor
        let color2 = UIColor(white: 1.0, alpha: 0.1).cgColor as
CGColor
        let color3 = UIColor.clear.cgColor as CGColor
        let color4 = UIColor(white: 0.0, alpha: 0.1).cgColor as
CGColor


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 К

Код файлу StrikeThrougText.swift

```
import UIKit
import QuartzCore

class StrikeThroughText: UITextField {
    let strikeThroughLayer: CALayer
    var strikeThrough : Bool {
        didSet {
            strikeThroughLayer.isHidden = !strikeThrough
            if 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 = 2.0
    func resizeStrikeThrough() {
        let textSize = text!.size(attributes:
[NSFontAttributeName:font!])
        strikeThroughLayer.frame = CGRect(x: 0, y:
bounds.size.height/2,
                                          width: textSize.width,
height: kStrikeOutThickness)
    }
}
```

Додаток Л

Код файлу FilmItem.swift

```
import Foundation

import 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?
String
            {
                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)
            }
        }
}
```

Додаток М

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

self.urlIMDBArray.append(name as! String)
                                        }
                                        if let name =
FilmsDict.value(forKey: "rating") {
                                            self.RatingArray.append(name
as! String)
                                        }
                                        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,
numberOfRowsInSection section: 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 imgURL = NSURL(string: imgURLArray[indexPath.row])

            if imgURL != nil {
                let data = NSData(contentsOf: (imgURL as? URL)!)
                cell.imgView.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 = imgURLArray[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 Н

Код файлу TableViewCell.swift

```
import UIKit

class TableViewCell: UITableViewCell {

    @IBOutlet weak var imgView: 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 П

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.didReceiveMemoryWarning()
    }

    func updateUI() {
        self.titleLabel.text = titleString
        self.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 viewDidAppear(_ animated: Bool) {
        plotView.scrollRangeToVisible(NSMakeRange(0,0))
    }
}
```

Додаток Р

Код файлу NavigationControllerFilms.swift

```
import Foundation
import UIKit

class NavigationControllerFilms: UINavigationController,
UIViewControllerTransitioningDelegate {

    override func viewDidLoad() {
        super.viewDidLoad()

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

Додаток С

Код файлу CircleButton.swift

```
import Foundation
import UIKit

class CircleButton: UIButton {

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        //layer.borderWidth = 1.0
        layer.borderColor = tintColor.cgColor
        layer.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 Т

Код файлу RoundedButton.swift

```
import Foundation
import UIKit

class BorderedButton: UIButton {

    required init?(coder aDecoder: NSCoder) {
        super.init(coder: aDecoder)
        //layer.borderWidth = 1.0
        layer.borderColor = tintColor.cgColor
        layer.cornerRadius = 10.0
        clipsToBounds = true
        contentEdgeInsets = UIEdgeInsets(top: 8, left: 8, bottom: 8,
right: 8)
        setTitleColor(tintColor, for: .normal)
        setTitleColor(UIColor.white, for: .highlighted)
        //setBackgroundImage(UIImage(color: tintColor),
forState: .Highlighted)
    }
}
```

Додаток У

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 viewDidAppear(_ animated: Bool) {
        locationAuthStatus()
    }
    func locationAuthStatus() {
        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 Червоної
Калини, 81","вул. Стрийська, 30, ТРЦ «King Cross Leopolis»"]
        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(_ viewFormapView: 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 Ф

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 X

Концепт додатку