МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ЖЕРЕБЦОВ ОЛЕКСАНДР МАКСИМОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д. т. н., доцент
_____ Т. В. Нескородєва
«_____» _____ 2022р.

ДОСЛІДЖЕННЯ ТА МЕТОДІВ АНАЛІЗУ ДАНИХ ІНТЕРНЕТ-МАГАЗИНУ МУЗИЧНИХ ІНСТРУМЕНТІВ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

Науковий керівник:
Нескородєва Т. В., завідувач
кафедри інформаційних технологій
д-р техн. наук, доцент
_____
(підпис)

Оцінка: _____/_____/_____
(бали за шкалою ЄКТС/за національною шкалою)
Голова ЕК: _____
(підпис)

Вінниця – 2022

АНОТАЦІЯ

Жеребцов О.М. Дослідження методів аналізу даних інтернет-магазину музичних інструментів. Спеціальність 122 «Комп'ютерні науки», Освітня програма «Комп'ютерні технології обробки даних (Data Science)». Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній (магістерській) роботі досліджена проблема створення сучасних додатків та дослідження методів аналізу даних сайту. Розроблений додаток для продажу музичних інструментів, під різні платформи, який аналізує дані користувачів з метою адаптації до їх вподобань на основі методів формування рекомендацій. Показано застосування сучасних інструментів з метою забезпечення функціональності, адаптивності, цікавості і зручності для користувача. Встановлені актуальні напрямки та перспективи розширення функціональності додатку.

91 с., 14 рис., 1 табл., 60 джерел.

Ключові слова: мультиплатформенний додаток, аналіз даних, інтернет-магазин, дошка оголошень, Flutter SDK, мова програмування Dart, Firebase, Android Studio, Android, iOS, macOS, WEB.

ABSTRACT

Zherebtsov O.M. Research of data analysis methods of online store of musical instruments. Specialty 122 "Computer science", Educational program "Computer data processing technologies (Data Science)". Vasyl' Stus Donetsk National University, Vinnytsia, 2022.

In the qualification (master's) work the problem of creation of modern applications and researching methods of data analysis is investigated. Developed application for the musical instrument's sale, for various platforms, which analyzes user data to adapt to their liking based on methods of forming recommendations. The application of modern tools is shown to ensure functionality, adaptability, interest, and convenience for the user. Current directions and prospects for expanding the functionality of the application have been established.

91 p., 14 figures., 1 tab., 60 ref.

Keywords: cross-platform application, data analysis, internet-shop, bulletin board, Flutter SDK, Dart programming language, Firebase, Android Studio, Android, iOS, macOS, WEB.

ЗМІСТ

ВСТУП

Мультиплатформенна розробка додатків стає все більш актуальнішою не те що з кожним роком, а і з кожним місяцем. Сотні нових додатків виходять на різні онлайн-платформи кожен день. В умовах такої високої конкуренції складно пропонувати щось нове, але креативні розробки у співпраці з продуманим маркетингом можуть принести великі прибутки своїм розробникам.

Через пандемію COVID-19, на тлі карантину рекордно зросло використання різних додатків на різних платформах. Місяці в ізоляції і перехід бізнесу та навчання в онлайн сильно вплинули на наш спосіб життя та наші звички. Ми частіше купуємо в інтернеті, замовляємо їжу онлайн, більше проводимо часу у смартфонах, витрачаємо більше часу на ігри. Це призвело до зростання попиту на різні додатки. Щомісячний час, що витрачається на мобільні додатки, зріс на 40 % у 2020 році, порівняно з 2019 роком. Така тенденція зберігається і у 2021 і у 2022 році. Відбулося зростання завантажень додатків для ігор (43%), для бізнесу (105%) і доставки їжі (73%). 2020-й також був роком онлайн відео-конференцій: кількість завантажень Zoom та інших додатків з можливістю онлайн-зустрічей зросла на 2000% порівняно з 2019 роком. Додатки перетворилися в найбільшу споживчу систему на планеті: у 2021 року загальносвітова виручка від додатків досягла $6,3 трлн, а за прогнозами на 2022 прибутки будуть ще більші. [1].

З іншого боку, людина кожного дня отримує занадто багато інформації, що створює проблему пошуку та вибору необхідних товарів чи послуг серед великої кількості альтернатив. Тому при розробці мультиплатформених додатків велика увага приділяється методам збору та обробки інформації які покликані опрацьовувати данні користувачів, їх вподобання, дії та поведінку на онлайн ресурсах, тощо, для створення найкращих персональних рекомендацій.

Отже, в сьогоднішніх умовах безпрецедентного розвитку IT-сфери та попиту на IT-продукти, нагальною стає проблема розробки якісних майданчиків для онлайн торгівлі та розміщення оголошень.

Розробка сучасного мультиплатформного додатку «Musicians Shop» покликана допомогти вирішити цю проблему, створивши спеціалізоване середовище для розміщення онлайн оголошень про купівлю-продаж музичних інструментів та супутніх товарів.

У світі великий успіх має reverb.com, онлайн-магазин нового, вживаного та старовинного музичного обладнання, який у серпні 2017 року посів 18 місце в списку Inc 5000 приватних компаній, що найшвидше розвиваються, з 12 327 відсотками трирічного зростання (2013 - 2016 рр.).

З вищесказаного можна зробити висновок про те, що проблема розробки сучасного додатку є важливим питанням, вирішенню якого і присвячується дана робота.

Актуальність теми дослідження. Найпопулярнішою тенденцією IT-ринку сьогодні є розробка додатків та аналіз даних. Сучасний додаток має декілька вирішальних переваг: швидкість, та надзвичайна зручність для користувача, а також додаток має заохочувати користувача переглядати додаток регулярно за рахунок подання актуального контенту. Створення такого додатку - нетривіальне завдання та існує безліч варіантів реалізації через кількість технологій, що швидко зростає. Тому дана тема є найактуальнішою на сьогоднішній день.

Мета. Розробка додатку «Musicians Shop» під Android, iOS, macOS та WEB з можливістю перегляду та створення оголошень, збору статистики додатку та адаптивного підбору оголошень під кожного користувача. Користувач онлайн-магазину, має мати однакові можливості додатку на будь-якій платформі, однакові інтерфейси користувача але з урахуванням особливостей кожної з платформ, основними функціями якого мають бути такі можливості: користувач може створювати, редагувати та видаляти свої оголошення або переглядати та ставити вподобання оголошенням інших користувачів, а також мати можливість перегляду сторінок авторів, користувач може бачити статистику своїх оголошень і переглядати свої вподобані оголошення, також користувачу мають рекомендуватися найбільш актуальні для нього оголошення

відносно даних які були вказані і здійснюватися пошук оголошень. Додаток має реалізовуватися з урахуванням усіх сучасних стандартів розробки та повинен бути максимально приємним і зрозумілим для використання.

Завдання дослідження.

- Вивчити та з'ясувати як працюють Dart, Flutter SDK, Git, GitHub, Android Studio, Xcode, Android Emulator, iOS Simulator, Firebase, FluterFire CLI, Firebase Authentication, Firebase Firestore, Firebase Storage, Firebase Messaging, Firebase Crashlytics, Firebase Analytics, Firebase Hosting.
- З'ясувати як розробляються мультиплатформенні додатки, використовуючи SDK Flutter.

Об’єкт дослідження. Методи аналізу даних, методи проектування та технології створення мультиплатформенних додатків.

Предмет дослідження. Популярні інструменти, бібліотеки та фреймворки для створення сучасних додатків.

Для досягнення поставленої мети необхідно вирішити такі дослідницькі завдання:

- Розглянути та дослідити поняття сучасних додатків.
- Обрати інструменти для створення додатку.
- Створити сучасний додаток, який буде задовольняти поставлену мету роботи.

Практична значущість роботи полягає в тому, що її рекомендації можуть бути використані при розробці мультиплатформенних додатків та вивчені мови програмування Dart, SDK Flutter, технологій Firebase а розроблений додаток знайти конкретну практичну реалізацію в своїй галузі.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МУЛЬТИПЛАТФОРМНИХ ДОДАТКІВ

1.1. Веб-додатки, мобільні додатки (Android, iOS), додатки під macOS (Desktop додатки) та їх відмінність і переваги.

Мобільна індустрія з кожним роком лише зростає. Сьогодні важко уявити своє життя без смартфону: спілкування, робочі моменти, шопінг і навіть подорожі - усе це перенеслося до невеликого ґаджету. Воно й не дивно: зараз у світі - 4,32 мільярда активних мобільних користувачів. Уже немає вибору. Пандемія COVID-19 лише пришвидшила процес переходу в інтернет-площину. Саме тому цей ринок продовжує надавати користувачам з усього світу потужне програмне забезпечення. Веб-додатки значно змінили наше життя, оскільки вони просто проникли у всі сфери життя людей: починаючи з покупки товарів так і до отримання державних послуг онлайн. Веб-додаток - це додаток , який працює в браузері, як сайт. Мобільний додаток - має бути завантажений з платформ цифрової дистрибуції (AppStore, GooglePlay тощо).

Мобільний додаток для запуску роботи має окрему іконку на робочому столі пристрою та може працювати частково чи повністю без доступу до інтернету. На відміну від нього , щоб скористатися функціями веб-додатку, потрібно за. Хоча цикл розробки у веб-додатків та мобільних дуже схожий, а підсумкові проекти можуть бути зовні майже ідентичними, створенням програм займаються абсолютно інші програмісти, які володіють іншими мовами. Більше того, усі інші фахівці, задіяні в розробці, від дизайнерів, до техпідтримки, і навіть маркетологів, теж будуть вирішувати свої завдання іншими способами, не такими, як для веб-додатку.

Може скластися враження, що веб-додатки це ще одна назва для сайтів, тому що браузери відкривають сайти. Але це не так. Сучасні браузери відкривають і сайти і веб-додатки. До перших відносяться простіші- новинні портали, особисті блоги, корпоративні та лендінг сторінки. До других - більш

складніші- сторінка Facebook, кабінет GoogleAnalytics, веб-версіяZoom. Натомість інтернет-магазини можуть бути в обох категоріях.

1.2. Мобільні додатки (Android, iOS).

Мобільний застосунок (додаток) - це програмне забезпечення, призначене для роботи на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних програм або встановлюються на сам пристрій, або можуть бути завантажені на пристрій безкоштовно або за плату з інтернет-магазинів мобільних програм, таких як App Store, Google Play і Windows Phone Store.

Спочатку мобільні програми використовувалися для швидкої перевірки електронної пошти, але їх високий попит призвів до їх використання в інших областях, таких як ігри для мобільних телефонів, GPS, зв'язок, перегляд відео та використання Інтернету. Програми зазвичай переслідують як мінімум дві різні цілі. Тобто щоб власник не сумував і полегшував життя. Іншими словами, мобільні програми, які корисні, навіть якщо вони марні. Розповсюдження доступних мобільних пристроїв, зручних мультисенсорних екранів, доступ до безкоштовного, а іноді і безкоштовного інтернету, а також легкий обмін інформацією створили неймовірну аудиторію потенційних клієнтів, тому мобільні навіть обговорювати необхідність програми недоречно. будь-якого бізнесу.

Багато мобільних пристроїв продаються з уже встановленим набором мобільних програм, таких як веб-браузери, поштові клієнти, календарі (наприклад, Google Calendar) та програми для покупки та прослуховування музики. Деякі встановлені програми можуть бути видалені користувачем з мобільного пристрою за допомогою звичайного процесу видалення, що звільняє більше місця для зберігання інших (переважних) програм.

Додатки, що відразу не встановлені на мобільний пристрій, доступні для завантаження та встановлення через платформи їх розповсюдження. Такі платформи називаються магазинами додатків. Вони почали з'являтися у 2008 році, та зазвичай, управляються компанією власником мобільних операційних систем: Apple App Store, Google Play, Windows Phone Store і Black Berry App

World. Проте існують свого роду незалежні магазини додатків, такі як Cydia, Get Jar або F-Droid. Значна кількість мобільних додатків є безкоштовною для встановлення та користування, проте існують і платні. Всі додатки зазвичай завантажуються з платформи одразу на цільовий пристрій, але іноді, вони можуть бути завантаженими на ноутбуки чи комп'ютери. Як правило, 20..30% вартості платних додатків надходить до компанії-дистриб'ютора (наприклад, iTunes), а решта - до виробника. Тому той самий додаток може мати різну вартість, в залежності від мобільної платформи.

З мобільними програмами все набагато простіше, ніж з веб-програмами. Як і комп'ютерні програми, вони доступні тільки для смартфонів та планшетів і можуть бути завантажені з безкоштовних джерел, а також фірмових магазинів Apple і Android. Існує безліч технологій, які можна використовувати для створення мобільних програм, але в цілому їх можна розділити на дві категорії:

1. Нативні додатки. Ключ до розуміння цього підходу криється в самій назві. В основному для розробки нативного додатку використовується рідна мова пристрою. Таким чином, нативний додаток - це додаток, створений певною мовою програмування для конкретної платформи. Програми написані рідними мовами мобільних платформ (Java та Kotlin для Android, Swift та Objective-C для iOS/iPadOS). Легкий доступ до технічних частин вашого смартфона, таких як мікрофон та камера. Це забезпечує відмінний і знайомий досвід користувача.

2. Мультиплатформні. Ці програми іноді називають кросплатформовими. Універсальні програми, які не залежать від платформи, розробляються з використанням як власних, так і веб-технологій. Порівняно з нативними додатками гібриди дешевші і найчастіше зручніші. Ви можете заощадити гроші, не платячи за розробку двох версій для iOS і Android, але в той же час це не підходить для дуже складних проектів, і досвід користувача не дуже хороший. Найчастіше це робиться на платформах розробки, таких як React Native, Flutter та Ionic.

Переваги мобільних додатків:

- Повна взаємодія з користувачем. На відміну від мобільної версії сайту, з мобільним додатком у вас є можливість надсилати push-повідомлення. Наприклад, якщо ви опублікуєте щось нове на своєму сайті, користувачі дізнаються про це лише тоді, коли зайдуть туди. За допомогою мобільного додатка ви можете просто надіслати повідомлення всім активним користувачам. Обмеження кількості символів недоступні для SMS-компаній.
- Локальні повідомлення працюють приблизно так само, тільки встановлюються на пристрій. Якщо користувач виконує в додатку дію, яка очікує від користувача реакції через певну кількість часу, програма нагадає користувачеві про це.
- Більше можливостей зворотного зв'язку. Користувачі можуть залишати повідомлення та відгуки в магазині програм або приватно в соціальних мережах через програму. Зворотній зв'язок автоматично транслюється на всі ресурси.
- Поліпшений інтерфейс. Всі елементи керування: кнопки, текстові поля, посилання мають бути зручними для натискання пальцями на мобільних пристроях, а не курсорами миші. Екрани пристроїв відрізняються як за розміром, так і пікселями. Екрани телефонів можуть мати майже такий самий дозвіл, як і ноутбуки, але при цьому мати менший фізичний розмір. Так що я не знаю, як сайт виглядатиме на тому чи іншому пристрої, чи буде він корисним і чи взагалі практично його використовувати. Мобільні операційні системи для таких ситуацій мають власну спеціально навчену логіку, щоб розрізняти, з яким пристроєм вони взаємодіють, і обробляти наслідки натискання пальцем крихітної кнопки. Однак ця логіка поводиться по-різному на різних платформах та пристроях.
- Якість інтерфейсу також відображається у навігації. Кожна мобільна операційна система має власну логіку перемикання між робочими екранами програм. Кнопка «Назад» на Android проведіть пальцем від лівого краю екрана на iOS. Користувачі кожної операційної системи звикли до однакової поведінки у кожному додатку. У кожного сайту свій спосіб навігації і при

переході на інший сайт доводиться шукати кнопки типу "ОК", "Назад", "Скасування", яких може зовсім не бути.

- Високий рівень персоналізації. Мобільний телефон багато знає про свого власника, і використання цієї інформації для підвищення рівня обслуговування є однією з основних причин швидкого зростання мобільних додатків. Мобільні програми завжди мають можливість запам'ятовувати всі дані користувача та змінювати інтерфейс відповідно до потреб користувача. Якщо користувач ввів дані (наприклад, домашню адресу в додатку для доставки), цю адресу не потрібно вводити повторно. Навіть на різних пристроях, якщо синхронізація увімкнена, користувач побачить програму, заповнену особистими даними. Будь-яка iOS запам'ятовує основні часові інтервали дій власника пристрою та запам'ятовує значки мобільних програм на екрані саме в цій послідовності.
- Те саме стосується пристроїв. Залежно від типу програми вона отримуватиме дозволи, дані з камер, акселерометрів, компасів, барометрів тощо.
- Локалізація. Наприклад, якщо вашому додатку необхідно відобразити список ресторанів, він, швидше за все, знатиме поточне розташування користувача з урахуванням відстані до нього. Логіка роботи мобільного сайту завжди проста. Він може не враховувати множину даних, які можуть надати мобільні пристрої.
- Працювати офлайн. Інтернет є скрізь, але не завжди. Навіть за наявності мобільного інтернет-з'єднання його якість у середньому поступається домашньому чи офісному інтернет-з'єднанню. Якщо ваша мета полягає в тому, щоб ваші користувачі залишалися підключеними до вашого продукту, як тільки ваше інтернет-з'єднання переривається, розробка мобільного додатка є єдиним можливим рішенням.
- Розробка мобільних програм компаніями також є ознакою престижу. Компанія піклується про своїх клієнтів та пропонує свободу вибору ресурсів та більш високий рівень комфорту, особливо коли необхідно побудувати з

ними довгострокові, якісні та продуктивні відносини. Крім того, фірмовий мобільний додаток - це додатковий рекламний канал і, навпаки, відсутність нав'язливої реклами - привід переконатись у його перевагах.

Недоліки мобільних додатків:

- Розміщення QR-коду або прив'язки на вашому сайті та в додатку завжди призводить до більшої кількості відвідувань сайту, ніж встановлення мобільної програми. Деякі користувачі не встановлюються навіть після переходу за посиланням на програму в магазині.
- Кросплатформність. На відміну від веб-сайтів, доступних на всіх пристроях, мобільні програми доступні не на всіх пристроях.
- Миттєві оновлення. На відміну від веб-сайтів, перевірка оновлень програми у магазині завжди займає більше часу. Оновлений сайт відразу доступний для користувачів.
- Витрати на розробку та супровід мобільних додатків, як правило, завжди вищі, ніж їхні сайтові аналоги.
- Рекламувати мобільний додаток явно дорожче, ніж сайт. Ціна одного користувача безпосередньо залежить від його порога входу, що вищий поріг, то вища ціна його залучення. Щоб користувачі могли потрапити на ваш сайт, встановіть певне значення для кілків по рекламному блоку і незабаром отримаєте відвідувачів. Те саме потрібно зробити і з мобільними додатками, але не факт, що користувач встановить їх після відвідування магазину додатків. В результаті ціна установки завжди набагато вища за ціну кліка. Додатки також можуть бути встановлені власноруч користувачем, як наприклад, запуск пакету програм для Android на пристроях з операційною системою Android.

Спочатку мобільні програми були надані як інструменти для контролю та управління спільними інформаційними потоками, такими як електронна пошта, календарі, контакти, фондова біржа та інформація про погоду. Проте затребуваність та доступність інструментів для розробників призвели до швидкого розповсюдження програм для інших категорій електронних

пристроїв, що працюють за допомогою настільних програм. Як і у випадку з будь-яким програмним забезпеченням, стрімке зростання кількості та різноманітності мобільних додатків призвело до створення величезної кількості ресурсів знань про них. Огляди, схвалення, огляди, блоги, журнали та спеціалізовані служби пошуку програм в Інтернеті.

Використання мобільних програм серед користувачів мобільних пристроїв стає все більш поширеним явищем. Згідно з даними дослідження App Annie, в 2017 році кількість завантажень додатків збільшилася на 60%, споживчі витрати зросли більш ніж удвічі, а кожен користувач проводив у додатках близько 43 днів на рік. Загальна кількість завантажень додатків у 2019 році збільшилась до 115 мільярдів. 31 мільярд у App Store і 84 мільярди у Google Play. Дослідники виявили, що використання мобільного додатка позитивно корелює з його змістом залежно від часу доби та розташування користувача..

1.3. Розробка додатків під Android.

Java, безперечно, є лідером у цій галузі та сміливо може називатися основною мовою розробки під Android. Той факт, що майже всі офіційні курси та навчальна документація з програмування під Android засновані цією мовою, свідчить про її популярність. Для Java можна знайти безліч вихідного коду на GitHub, і самі розробники відзначають, що ця мова дуже зручна для написання мобільних додатків. Java широко поширилася в десятках галузей програмування, тому розробникам Android-початківцям непогано було б почати з вивчення Java в контексті екосистеми Android.

Kotlin дозволяє програмним продуктам мати всі сильні сторони Java без недоліків. Синтаксис програмування Kotlin та інші функції є сучасними, простими для розуміння та швидкими у простому середовищі розробки.

У 2019 році Google призначив Kotlin кращою мовою для платформи Android, зробивши її однією з офіційних програмованих мов для розробки під Android. Більше того, він повністю сумісний із Java. C# взяв найкраще з Java та додав кілька своїх цікавих функцій. Програмування під Android на C# має на

увазі використання Xamarin. Його можна використовувати для створення однієї логіки програми на C# для Android та iOS.

C++ - це кроссплатформенна мова програмування, яку можна використовувати для створення високопродуктивних мобільних та настільних додатків. Ця універсальна мова високого рівня була представлена як розширена версія мови програмування. Це дає розробникам повний контроль над пам'яттю та системними ресурсами. Для розробки Android-програм можна використовувати безліч бібліотек, доступних для цієї мови. Щоб код, написаний на C++, працював у AndroidStudio, вам потрібний власний набір програмного забезпечення, відомий як NDK. NDK дозволяє підвищити продуктивність, що особливо необхідно для хорошої реалізації програми на Android.

Basic4Android. Ця мова - найкращий вибір для новачка. Можна використовувати його у середовищі розробки B4A. Сам синтаксис Basic4Android дуже нагадує популярний VisualBasic. При розробці програм на ньому ви зможете використовувати безліч додаткових бібліотек з різним функціоналом, і вам не потрібні спеціальні інструменти runtime для запуску програм. Basic4Android має потужний візуальний конструктор, який підтримує різні орієнтації та розміри екрану. Ви можете навчитися багатьом новим речам без особливих зусиль і створювати базові програми, але для створення просунутих програм вам потрібно вивчити одну з вищезгаданих мов.

1.4. Процес розробки мобільних додатків під iOS.

IOS - одна з найпопулярніших платформ у світі. Financial Times повідомляє, що корпорація Apple досягла рекордної кількості проданих смартфонів в один мільярд пристроїв. Фактично кожна восьма людина на планеті могла б мати смартфон від Apple. Це величезна цільова аудиторія.

Розробка мобільних додатків для iOS виконується достатньо стандартним і в той же час унікальним для кожного випадку способом. Процес включає в себе планування, дослідження та чітку направленість. Щоб додаток став успішним,

необхідно під час його розробки правильно виконати певні кроки. Як правило, весь цикл складається із 7 етапів.

Етапи розробки:

1. Планування і написання Технічного завдання. Перед початком роботи необхідно знати, для якої цільової аудиторії буде розроблятись iOS-додаток, які завдання йому доведеться вирішувати та які функції будуть потрібні для функціонування програми. Крім того, необхідно комплексно вивчити ринок, щоб розуміти наскільки додаток буде затребуваним і конкурентноспроможним. Цей етап включає в себе й підготовку технічного завдання на розробку.

2. Проектування та дизайн. Наступним кроком є прототипування і дизайн інтерфейсу - UX/UI. Після узгодження прототипів, дизайнер створює візуалізацію екранів, які бачать користувачі на своїх мобільних пристроях.

3. Front-end розробка. Користувач напряму взаємодіє із зовнішнім інтерфейсом застосунку, тому на цьому етапі увага концентрується на роботі з елементами дизайну. Добре пропрацьований інтерфейс дозволяє додатку ефективно функціонувати на екранах з різними характеристиками та розмірами. Це дозволяє йому правильно відображатись і на більш ранніх версіях iPhone, і на нових моделях.

4. Back-end розробка. Початковий крок в розробці, який відповідає за функціонування iOS-додатку. Етап передбачає розробку на боці сервера, а конкретніше: аутентифікацію, функціонування облікових записів користувачів, налаштування взаємодії між ними, інтеграцію з соціальними мережами, пуш-повідомлення і багато іншого.

5. Створення API. Програмний інтерфейс або API додатку зв’язує внутрішній і зовнішній інтерфейси мобільного додатку. API є одним з найважливіших елементів для створення iOS-додатку.

6. Тестування. Після завершення кожного етапу розробки додаток обов’язково тестується, щоб переконатися в його повній роботоспроможності без помилок і збоїв.

7. Запуск та обслуговування. Після тестування додаток можна публікувати в AppStore. Але додавання в магазин не означає завершення робіт. Будь-яке ПЗ потребує обслуговування і підтримки, а також покращення поточних функцій і додавання нових можливостей.

Мови програмування iOS додатків.

При розробці програм для пристроїв на базі iOS, iPadOS, tvOS, macOS, watchOS використовуються мови програмування Objective-C і Swift Вони є основними мовами для написання iOS-застосунків. Ці мови об'єктно-орієнтовані і дають можливість під час написання коду групувати схожі завдання, що суттєво прискорює та спрощує роботу розробників.

Мова програмування Objective-C.

Objective-C є більш старою мовою програмування. Вона вперше була представлена в 80-х роках минулого століття. Поступово вона допрацьовувалася і стала основною для пристроїв Apple. Тому за допомого Objective-C можна створювати додатки під будь яку техніку "яблучної" корпорації. Серед переваг цієї мови можна назвати високу ступінь підтримки коду, величезну базу навчальних матеріалів і велику спільноту, схожість з сімейством C, сумісність з "молодшою" мовою Swift.

Мова Objective-C також відома як ObjC чи Obj-C. Вона є компільованою об'єктно-орієнтованою мовою програмування корпорації Apple і побудована на основі мови C-і та парадигм Smalltalk. Мова Objective-C повністю сумісна з C-і і код на C-і компілюється. Об'єктна модель побудована в стилі Smalltalk, тобто об'єктам надсилаються повідомлення. Компілятор Objective-C входить в GCC та доступний на більшості основних платформ. В першу чергу мова використовується для двох реалізацій об'єктно-орієнтованого інтерфейсу OpenStep - MacOS X (Cocoa) і GNUstep.

Слід зазначити, що мова також є message-oriented на відміну від C++, який є function-oriented. Виклики методу в Objective-C інтерпертуються не як виклик функції, а саме як надсилання повідомлення (з іменем і аргументами) об'єкту, подібно тому, як це відбувається в Smalltalk. Дана система забезпечує ряд

переваг - будь-який об’єкт може надсилати будь-яке повідомлення. Таким чином, замість обробки повідомлення об’єкт може делегувати свої повноваження іншому об’єкту для проведення обробки. Це дозволяє легко реалізувати розподілені об’єкти, які знаходяться в різних просторах. Повідомлення прив’язуються до відповідних функцій зразу на етапі виконання.

Поняття інтерфейсу об’єкта і протоколу в мові мають чітке розділення. Для протоколів підтримується множинне наслідування, а для об’єктів – не множинне. Об’єкт може успадковуватись від другого об’єкта і підтримувати зразу кілька протоколів.

Мова програмування Swift.

В 2014 році корпорація Apple представила нову мову програмування, яка отримала назву Swift. За словами розробників, ця мова увібрала в себе краще від популярної Objective-C і C. При цьому вона отримала зручний функціонал і більш сучасний інструментарій. Головними перевагами цієї мови можна назвати високу швидкість розробки програм, зменшення коду, кращу читаність, підтримку динамічних бібліотек, посилену безпеку. Objective-C та Swift добре сумісні, тому їх можна використовувати та використовують навіть в одному проекті. Однією з особливостей Swift є безпека. Завдяки інноваціям і оновленням синтаксису, ця мова набагато безпечніша за obj-c. Покращення роботи з пам’яттю звела можливість несанкціонованого доступу до даних до мінімуму. Ефективна обробка сценаріїв, в свою чергу, зменшила кількість критичних сценаріїв. Візуалізація результату – це також важливий момент, на який необхідно звернути увагу. Свіфт має особливу пісочницю Playground, яка дає можливість демонструвати роботу програми. Така система дозволяє скоротити час розробки і забезпечує швидке знаходження проблем в коді. Мова Swift продовжує розвиватися і все більше розробників переходять на неї чи взагалі починають свою трудову діяльність саме з неї. Apple робить ставку на Swift, але повністю відмовлятись від Objective-C точно не варто.

Середовище розробки.

Тепер ви знаєте на чому розробляють iOS-додатки і можна переходити до знайомства з використовуваними програмістами середовищами розробки. Найпопулярніше інтегроване середовище розробки (IDE) - це продукт XCode, який надається на безоплатній основі і створений компанією Apple.

XCode - основне середовище розробки XCode являє собою зручний додаток з великим набором корисних інструментів, які суттєво прискорюють і спрощують процес написання програм. В одному середовищі ви можете створити свою програму, протестувати та оптимізувати її, а потім відразу ж зібрати на цільовому пристрої iOS. Інтерфейс єдиного вікна суттєво спрощує роботу розробника. Більш того, під час написання коду програма вкаже програмісту на помилку, якщо вона з'явиться. В XCode інтегрований додаток IB (InterfaceBuilder), котрий дозволяє розробляти графічні інтерфейси, настроювати стилі і шрифти. Якщо для створення додатку доводиться працювати з мапами, в XCode передбачена функція для імітації геолокацій. Завдяки вбудованому симулятору програмісти можуть протестувати свою розробку, а конфігуратор допомагає знайти помилки верстки та можливі помилки.

Перераховані переваги лише головні з багатьох, які має дане середовище розробки iOS-застосунків. Головним конкурентом XCode є розробка JetBrains - AppCode з чудовою роботою автодоповнення, хорошою інтеграцією з issue-трекерами, докладним описом помилок. Але навряд вийде використовувати цю IDE в якості основної і єдиної, в більшості випадків розробники повертаються до XCode.

1.5. Веб-додатки.

Три базові відмінності веб-додатку від сайту:

1. Розширена аутентифікація. Користувачі веб-додатків можуть не тільки переглядати але і створювати, завантажувати, змінювати контент.

2. Складні функції. Для складної бізнес-логіки , наприклад, бронювання готелю, обмін повідомленнями, банківські послуги недостатньо базових

функцій інтернет-магазину (відсортувати, відправити в корзину, сплатити), тому створюється веб-додаток на основі програмної платформи.

3. Зробити зміни у сайті просто, як оновити HTML-код сторінки. Веб-додаток вимагає повноцінної компіляції та розгортання нового програмного забезпечення.[8]

Веб-додатки прийшли на заміну веб-сайтам. Це відбулось у де-кілька етапів:

1. Мобільна версія. У двотисячних роках , коли на мобільний трафік припадало 4% від усього світового обсягу трафіку почали створювати перші мобільні версії сайтів. Легкі, з меншою кількістю функцій вони вирішували найголовнішу проблему - давали користувачам мобільних пристроїв доступ до сервісів. Зараз вони не використовуються.[32]

2. Чутливий сайт. Сайти з чутливим дизайном, які стискалися або розтягувалися, залежно від розміру вікна браузера. Тоді з'явився принцип mobile-first, згідно з яким сайт спочатку робився під перегляд із мобільних , і в другу чергу для браузера. Такі шаблонні рішення на основі різних CMS використовуються й зараз, наприклад, у маленьких сайтах-візитках та лендингах.[30, 33]

3. Адаптивна верстка. З 2011 всі нескладні сайти, що заслуговують на увагу, за замовчуванням створюються з адаптивним дизайном. У них є кілька варіантів верстки для різних пристроїв, які забезпечують високий ступінь юзабіліті. Зараз це базове рішення для всіх проєктів із невеликим та середнім навантаженням. Тих самих корпоративних та новинних сайтів, а також середніх інтернет-магазинів. Сучасні веб-додатки відрізняються від усього, описаного вище й бувають трьох видів.[31]

4. PWA. Прогресивний веб-додаток - майже як нативний мобільний додаток. Працює скрізь, але що новіша версія ОС, то краще, оскільки може поліпшуватися пропорційно технологіям. Зручний інтерфейс користувача, як у повноцінної програми. Одночасно індексується в пошукових мережах та

дозволяє зберегти на екран іконку швидкого доступу. Вимагає підключення за захищеним протоколом HTTPS, але може частково працювати без інтернету.

5. HTML-5. Веб-сторінка, що імітує роботу програми, також доступна в будь-яких браузерах. Не запускається без інтернету, не підходить для суперскладних проектів, зате ідеальна, коли треба швидко створити якісний веб-додаток, котрий відповідає користувальницьким очікуванням.

6. SPA. SinglePageApplication - односторінковий веб-додаток із динамічним оновленням. Статична основа залишається незмінною, змінюються ті дані, із якими взаємодіє користувач. SPA-технологію використовують, як для адаптивних сайтів, так і для PWA веб-додатків.

Приклади веб-додатків:

1. Онлайн-редактори. GoogleDocs і Sheets, а також аналогічні їм Word та Excel 360, якими можна користуватися онлайн у браузері. Сюди ж відносяться вебпрограми для роботи з графікою, від Photoshop і Canva, до всіляких простих сервісів для накладання фільтрів на фото.

2. Соціальні мережі та месенджери. FB та його Messenger, Instagram, Twitter, Телеграм, Skype, Zoom та web-версії всього подібного їм.

3. E-commerce. Усі маркетплейси як Amazon і Prom.ua, а також великі торгові майданчики.

4. Банки, біржі та аукціони. Сайти всіх фінансових установ, від e-Bay та трейдингових платформ.

5. Кур’єрські служби, сервіси бронювання авіаквитків та житла, краудфандингові платформи тощо. писок можна продовжувати дуже довго, головне, щоби проєкт був складним, нестандартним та передбачав авторизацію [9].

1.6. Desktop Додатки.

Різниця між сайтами , веб- та декстопними додатками - вагома, адже для сайтів та веб-додатків використовується браузер. Десктопна програма повністю встановлюється на комп'ютер і не вимагає додаткового Інтернет-з'єднання (за умови, що програма не вимагає доступу до мережі). Програма дозволяє

зберігати всі файли на одному або кількох локальних комп'ютерах, а тому доступ до неї буде не у всіх користувачів [10].

Розробка Desktop додатків для бізнесу відкриває нові горизонти. Запровадження таких продуктів дозволяє підвищити ефективність, автоматизовує процеси та оптимізує роботу компанії. Десктопні програми мають цілу низку можливостей:

1. Автономна робота без Інтернету, що гарантує безпеку даних та високу продуктивність;

2. Підключення до Інтернету відкриває можливість використання хмари для збереження даних, доступу кількох користувачів до одного проекту, автономне оновлення програми без переустановки або тривалих налаштувань;

3. Можливість працювати в локальній мережі;

4. Швидкий запуск без постійного оновлення даних і завантаження параметрів з мережі.

5. Десктоп програми мають доступ до всіх пристроїв, які підключаються до комп'ютера, тобто інтегруються з принтерами, сканерами, фіскальними апаратами та іншими [11].

Кожна група має свої особливості і дозволяє повністю розкрити можливості бізнесу, розважальних чи громадських проектів.

1.7. Переваги нативних додатків.

1. Просте використання і опанування. Власні інтерфейси додатків та графічні компоненти наслідують філософію дизайну, вбудовану в конкретну операційну систему. Розташування елементів управління, палітри кольорів, анімація - все це легко впізнається. Тому така програма дуже проста в освоєнні і її використання не створює жодного дисонансу з існуючим користувальницьким досвідом.

2. Висока швидкість завантаження і роботи. Нативні програми оптимізовані під конкретну мобільну ОС, тому працюють швидко і дуже стабільно. Крім того, дані таких програм в основному зберігаються на пристрої, що робить їх роботу більш швидкою і менш залежною від інтернету.

3. Широкий спектр можливостей. Додаткам цього типу доступні всі функції ОС та компонентів пристрою (GPS, камера, календар, адресна книга тощо). Завдяки цьому нативні програми мають широкий функціонал і легко інтегруються один з одним.

1.8. Переваги мультиплатформних додатків.

1. Універсальність. Мультиплатформенні додатки пишуться для кількох платформ одночасно, а потім легко адаптуються до кожної з них. Сюди входять версії для iOS та Android, а також веб-версії та версії для ПК. Це робить багатоплатформні програми доступними для більш широкої аудиторії..

2. Зменшити витрати на розробку. Цей тип програм має меншу початкову вартість, ніж нативні програми, тому що його простіше та швидше розробляти. Це робить кросплатформні програми привабливим рішенням для компаній з обмеженим стартовим бюджетом.

3. Швидший вихід на ринок. Спрощення розробки багатоплатформних програм дає ще одну перевагу. Завдяки скороченню часу розробки, ви можете відносно швидко випустити свій продукт, щоб отримати перші переваги. Цей тип програми є розумним рішенням, якщо компанія не має можливості витрачати багато часу на розробку.

Економія часу і коштів - не єдині переваги розробки мультиплатформних додатків. З їхньою допомогою ще можна «перевірити» ідею та швидко створити прототип, аби переконатися, що платформа виглядатиме й працюватиме так, як того хоче замовник [12].

1.9. Переваги мультиплатформних додатків над нативними.

Основну різницю між цими додатками можна описати з точки зору їх вартості розробки. Мультиплатформні є більш доступними, ніж власне програми, які створені спеціально для iOS або Android. До того ж, вони можуть поєднати у собі нативні елементи та елементи веб-розробки. Саме тому мультиплатформні додатки зараз є хорошим вибором для бізнесу - незалежно від потреб компанії, вони якісно та легко виконують свої функції.

Ще кілька років тому вважалося, що мультиплатформенний додаток – це не найкращий варіант для бізнесу, тому що він занадто повільний і низькопродуктивний. Однак за достатньо короткий час та з появою сервісів для розробки мультиплатформних додатків вони перетворилися на достатньо непогану пропозицію для бізнесу.

Чи можуть все ж IOS та Android функціонувати на одному Native додатку? Ні. Це неможливо. Програми, створені для різних операційних систем, не збігаються. Отже, додатки для Android не зможуть працювати на IOS і навпаки. Єдиний виняток - це мультиплатформенний додаток, який враховує особливості двох систем.

Приклади мультиплатформних додатків:

1. Інстаграм - одна із найвідоміших соціальних мереж у світі. Додаток працює на HTML-5 і може підтримувати офлайн-дані, виконувати дії з мультимедійними файлами. До того ж, соцмережа пропонує користувачам низку інструментів для обміну інтерактивними даними.

2. Twitter - соцмережа для коротких повідомлень , один із найпопулярніших прикладів мультиплатформного додатка. Для обробки високого рівня трафіку платформа використовує гібридний підхід. Пропонує користувачам низку інструментів, доступна як у додатку, так і в браузері на мобільному пристрої чи комп'ютері.

3. Gmail - найпопулярніший поштовий сервіс теж використовує гібридний підхід. Компанія Google на цій платформі поєднала HTML та власні нативні елементи. Це забезпечує безперебійну роботу сервісу.

4. Uber - Мобільний додаток з таксі перевезень є хорошим прикладом мультиплатформного додатку. Сервіс містить просту навігацію, хороший та якісний користувацький інтерфейс. За рахунок цих технологій та зручних інструментів додаток випереджає своїх конкурентів.

5. Apple App Store - теж має гібридний підхід. Це хороший приклад використання нативних елементів для навігаційної та верхньої панелей пошуку. Це покращує взаємодію з користувачем і спрощує функції для нього.

Прикладів мультиплатформних додатків набагато більше, вони буквально полонять ринок розробки мобільних додатків, адже мультиплатформні додатки пропонують зручність, якість, економлять часові та фінансові затрати.

Висновок до розділу 1.

В цьому розділі було розглянуто загальне поняття додатків та сучасних мультиплатформних застосунків. Визначено необхідність використання різних додатків в сучасних реаліях, висвітлено усіх їх актуальні переваги та недоліки, а також перспективу розвитку ринку різних сучасних додатків.

РОЗДІЛ 2
МЕТОДИ АНАЛІЗУ ДАННИХ ДЛЯ РОЗРОБКИ МУЛЬТИПЛАТФОРМНИХ ДОДАТКІВ

2.1. Дослідження характеристик рекомендаційних систем.

З появою мережі Інтернет кількість інформації, з якою люди щодня стикаються, вагомо зросла. Це означає, що люди повинні орієнтуватися серед гігантської кількості можливих альтернатив, коли хочуть щось знайти. З іншого боку, однак, власники інтернет-магазинів та сервісів зацікавлені у персональній рекламі та рекомендаціях конкретним користувачам, тому що такий підхід може значно збільшити прибуток компанії. В результаті, за останні роки інтерес до розробки і поліпшення існуючих рекомендаційних систем помітно виріс.

У наші дні рекомендаційні системи вже досить поширені і мають багато додатків. Насамперед рекомендаційні системи використовуються в онлайн-торгівлі, щоб допомогти користувачам вибрати відповідні товари. Такі сервіси збирають інформацію про ваші уподобання та намагаються надати вам корисні продукти. Існує багато методів для формування рекомендацій, але всі ці методи мають як свої переваги так і свої недоліки. Ось саме чому дослідження в даній області актуальні.

Рекомендації формуються персонально для кожної людини, опираючись на її попередні дії на конкретному веб-ресурсі чи на основі минулої активності. Крім того, важлива й поведінка попередніх учасників процесу.

Це важлива функція для інтернет-магазинів та один із небагатьох способів ефективної роботи великих каталогів, таких як Amazon. Переважний спосіб в даному випадку забезпечує користувачеві зручність навігації по веб-ресурсам, а не додаткові звичні можливості. Якщо у вашому електронному каталозі понад 20 000 найменувань товарів, орієнтація вже видається непомірно важкою, не говорячи уже про мільйони товарів. На допомогу приходить віджет пошуку товарів, візуально схожих на шуканий, або належить одній групі виробів, або

компліментарна продукція. Такі рішення не лише збільшують кількість переглядів, а й позитивно впливають на конверсії. Такі методи, як свідчить практика, використовують як інтернет-магазини. Подібні прийоми можна легко побачити практично скрізь: різні соціальні платформи, портали, ресурси новин, інтернет-магазини. Така методика є дійсно дуже популярна.

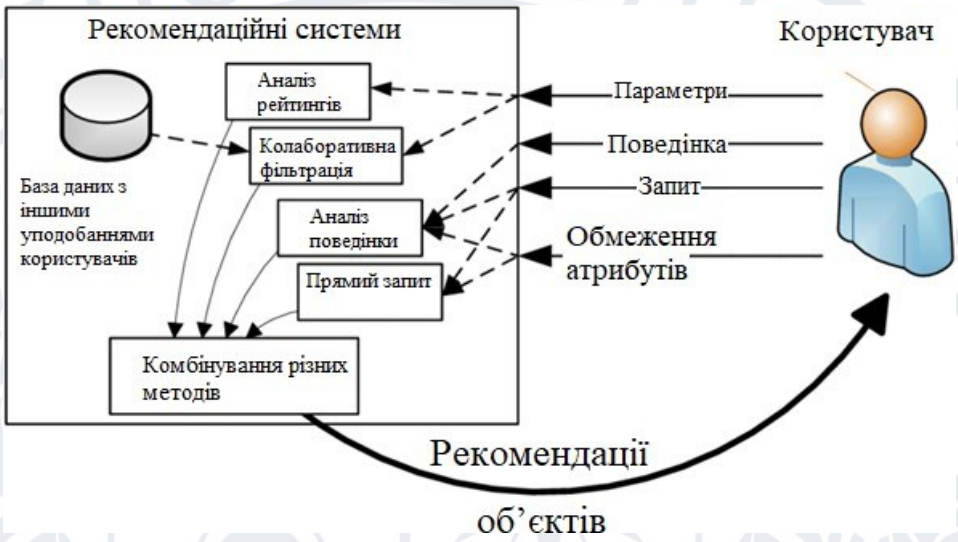

Рис. 2.1 - Структура рекомендаційної системи

Сервіси рекомендацій збирають різну інформацію людей, використовуючи кілька методів, які поділяються на системи.

Отже, перший тип - це явний збір даних. Як можна здогадатися з назви, необхідні роботи матеріали користувач готує сам. Наприклад, коли Google та інші системи рекомендацій пошукових систем просять вас оцінити різні фактори, створити список обраного у певній області або відповісти на кілька запитань. Якщо ж людина відмовляється надати інформацію самостійно, то тоді буде актуальною наступна методика. Другий тип - неявний збір даних. Це «шпигунська місія», де дії учасників процесу записуються програмно для подальшої обробки та застосування.

Програма розпізнає покупки, оцінки на сайті, збирає інформацію про перегляди та коментарі. Безумовно, такий вибір викликає етичні питання, оскільки захист персональних даних є однією з основних вимог, які висувають користувачі до пошуку на сайті. Але поки факт залишається фактом - своєрідне

стеження можливо, і користувачі сайтів не можуть перевірити, чи дійсно ведуться подібні заходи.

2.2. Аналіз процесу роботи рекомендаційних систем.

Процес роботи рекомендаційної системи- це збір даних про користувача. Дані збираються за допомогою різних методів, які розділяють на явні та неявні. Кожна система має специфічні вимоги до того, які дані повинні бути представлені для аналізу.

Вибір конкретних даних для вивчення може сильно вплинути на модель, з цих причин підготовка даних є занадто важливою частиною для будь-якої системи.

Підготовка даних часто є найбільш трудомісткою частиною будь-якого проекту. Вибір конкретного типу фільтрації або комбінації з декількох методів безпосередньо залежить від двох факторів – складності проекту і розміру його фінансування. Наприклад, створити алгоритм для системи з тематичних, пересічних один з одним блогів-завдання відносно проста і помірно витратна.

Більш масштабні і неоднорідні проекти, такі як онлайн-магазини чи торговельні майданчики, наприклад Rozetka чи Amazon, вимагають великих витрат, особливо в тому випадку, якщо стоїть завдання збільшити конверсію на дійсно значущі величини.

Як правило, в таких проектах не виходить обмежуватися одним видом рекомендаційного алгоритму і доводиться використовувати гібридну фільтрацію, в результаті чого вартість і складність розробки збільшується на порядки.

Так чи інакше, при розробці проекту, що надає користувачеві можливість вибору конкретного об'єкта із загального набору, необхідно враховувати стрімкий прогрес юзабіліті абсолютно у всіх сферах життя людини - від оптимізації сну за допомогою пристрою, який аналізує всі процеси, що відбуваються під час сну, видає рекомендації щодо їх поліпшення та автоматично підбирає предмети першої необхідності

виходячи з поточних потреб користувача.

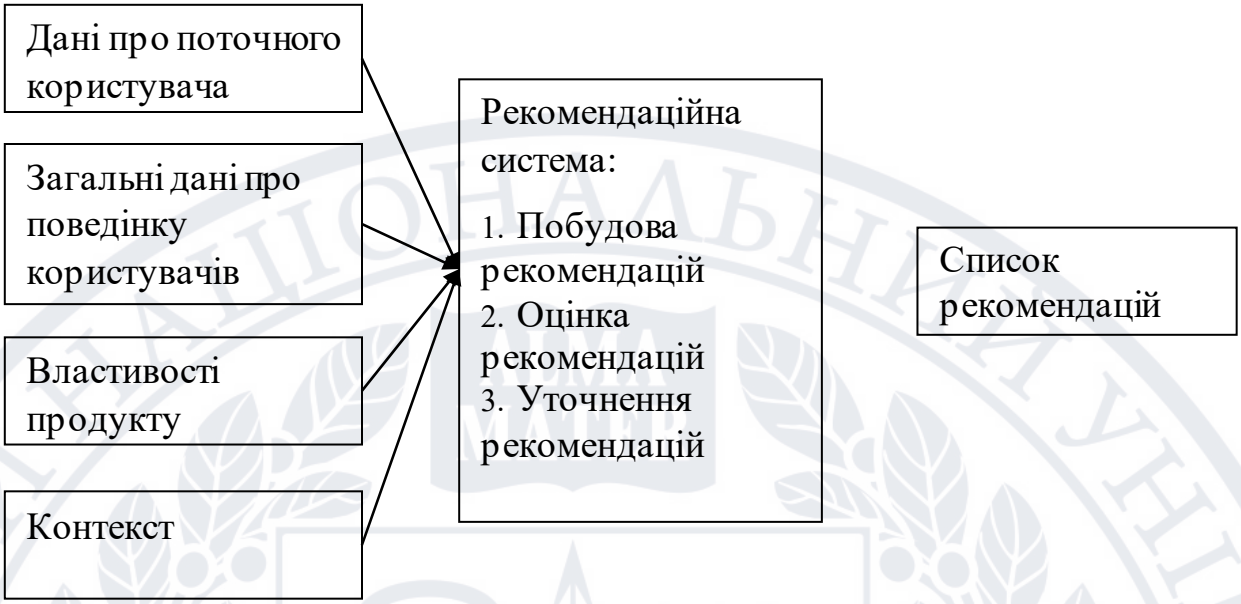

Рис. 2.2 - Процес роботи рекомендаційних систем

На сьогоднішній день для створення рекомендаційних систем використовуються дві основні стратегії: контентна фільтрація та спільна фільтрація. Зазначимо, що на практиці зазвичай використовується гібридна методика, що поєднує в собі переваги наступних підходів.

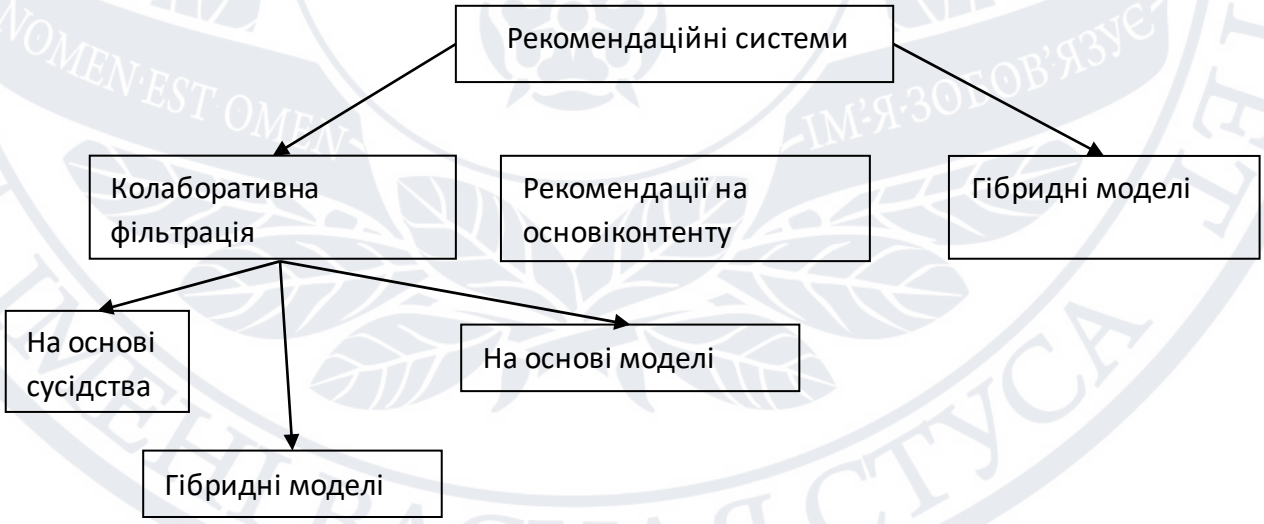

Рис.2.3 - Класифікація рекомендаційних систем

Крім евристики, заснованої здебільшого на методах інформаційного пошуку, достатньо часто використовуються і інші техніки рекомендацій з фільтрацією вмісту, наприклад, різні методи машинного навчання, такі як

прості байєсівські класифікатори, нейронні мережі, дерева рішень та кластеризація. Однак, на відміну від методів пошуку інформації, ці методи намагаються передбачити задоволеність користувачів, використовуючи як основу евристику косинусного коефіцієнта. Наприклад, у музичному аудіо-стрімінзі можна, використовуючи інформацію про музичні композиції які сподобались та ні, через наївний байесовский класифікатор виділити композиції, які користувач ще не чув і оцінити ймовірність належності композиції p_j до певного класу C_i (вподобані та ні).

2.3. Загальний підхід до оцінювання рекомендаційних систем і їх порівняльний аналіз

Рекомендаційної системи складається з алгоритму і набору даних D, як показано на рис. 2.1. Алгоритм рекомендації використовує набір даних для обчислення моделі користувацьких переваг. Потім ця модель може бути використана для обчислення рекомендацій для користувачів системи.

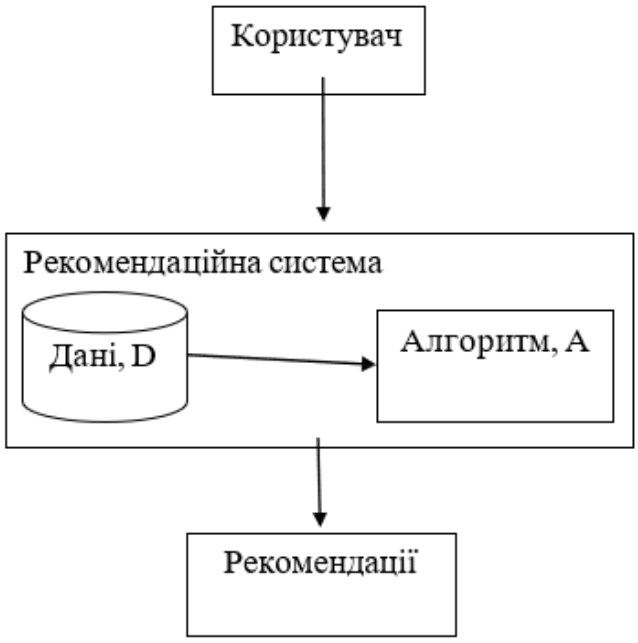

Рис. 2.4 - Модель рекомендаційної системи

Колаборативна фільтрація

Колаборативна фільтрація використовується в рекомендаційних системах. Рекомендаційні системи традиційно використовуються для прогнозування вибору цікавих користувачеві об'єктів на основі наявної інформації, як про поточного користувача, так і про інших клієнтів. В якості об'єктів, що представляють інтерес для користувача, розглядаються: новини, веб-сайти, фільми, товари і т. п. Залежно від типу вхідних даних рекомендаційні системи застосовують два види зворотного зв'язку: явний і неявний. Явний заснований на оцінках користувачів, що відображають їх інтерес до книги, фільму або іншого об'єкта. Такий зворотний зв'язок використовує, зокрема, компанія Netflix при формуванні рейтингу для фільмів і телевізійних програм. Користувачі мають можливість виставляти оцінки в даному сервісі. Однак в більшості випадків отримання явних оцінок користувачів пов'язано з труднощами, що і призводить до необхідності використання неявного зворотного зв'язку від користувача. Неявний зв’язок відображає переваги користувача на основі спостережень за його поведінкою. В якості вхідних даних при неявному зв'язку зазвичай використовують: історію покупок; переглянутих сайтів; шаблони пошуку; шаблони руху миші, і т. п. Ключовим підходом, який використовується при побудові рекомендаційних систем з неявним зворотним зв'язком, є колаборативна фільтрація. Даний підхід ґрунтується на принципах відбору за схожістю користувачів або за схожістю об'єктів, з якими взаємодіють користувачі. При колаборативної фільтрації з неявним зворотним зв'язком виявляють приховані фактори, які впливають на вибір користувача. При реалізації колаборативної фільтрації виникають проблеми, пов'язані з розрідженістю даних, а також наявністю релевантних даних. Розрідженість даних пов'язана з тим, що більшість комерційних рекомендаційних систем заснована на великій кількості даних про товари і незначній кількості оцінок користувачів.

В результаті матриця «об'єкт-користувач» виходить дуже великою з малою кількістю даних про покупки/перегляди. Це не дозволяє підвищити

точність рекомендацій. Зазначена проблема особливо актуальна для нових, щойно створених систем. Проблема релевантності оцінок часто виникає в разі холодного старту, оскільки нові об'єкти або користувачі ускладнюють створення релевантних рекомендацій.

Для вирішення зазначених проблем доцільно використовувати не тільки розріджені «позитивні» дані-інформацію про кількість покупок, переглядів, але і даних про «негативні» переваги. Однак існуючі версії колаборативної фільтрації орієнтовані на позитивні дані. Це не дає можливість отримати генералізовану модель і може призводити до виникнення проблеми перенавчання.

Методи, які зазвичай використовуються в колаборативній фільтрації це методи пам'яті чи сусідства. Ці методи обчислюють відсутні значення матриці ранжирування на основі їхніх сусідів. Ці околиці можуть бути визначені одним із двох способів:

- колаборативна фільтрація на основі користувачів. Основна ідея полягає в тому, щоб визначити райони (групи користувачів, схожих на цільового користувача) та обчислити кожне відсутнє значення у матриці оцінок цільового користувача як середньозважене значення рейтингів його району. Функція подібності обчислюється між рядками матриці ранжування виявлення схожих користувачів;

- колаборативна фільтрація на основі предметів. Основна ідея полягає в тому, щоб визначити сусідство (групу елементів, найбільш схожих на цільовий елемент) та обчислити кожне відсутнє значення у матриці оцінок цільового елемента як середньозважене значення оцінок у цьому сусідстві. Функція подібності обчислюється стовпчиками матриці ранжування виявлення схожих елементів.

Колаборативна фільтрація на основі елементів підвищує точність та стабільність при зміні оцінок. Колаборативна фільтрація на основі користувачів збільшує різноманітність.

Для колаборативної фільтрації на основі користувачів рейтингові

оцінки здебільшого розраховуються на основі коефіцієнта кореляції Пірсона.

$$\hat{r}_{ai} = \frac{\sum_{b \in P_a(i)} Pearson(a,b)(r_{bi} + \mu_a - \mu_b)}{\sum_{b \in P_a(i)} Pearson(a,b)},$$

$$Pearson(a,b) = \frac{\sum_{i \in I_a \cap I_b}(r_{ai} - \mu_a)(r_{bi} - \mu_b)}{\sqrt{\sum_{i \in I_a \cap I_b}(r_{ai} - \mu_a)^2}\sqrt{\sum_{i \in I_a \cap I_b}(r_{bi} - \mu_b)^2}},$$

$$\mu_a = \frac{\sum_{i \in I_a} r_{ai}}{|I_a|}$$

де

$\hat{r}_{ai}$ - оцінка рейтингу за моделлю, для користувача a за предметом i,

r_{ai} , r_{bi}- спостережуваний рейтинг,

$P_a(i)$ - множини k -найближчих юсерів для юсеру a, що виставив рейтинг предмету i ,

Pearson (a,b) - кореляція користувачів a і b з множини загально оцінених предметів $I_a \cap I_b$,

I_a - множина індексів предметів, яким виставив рейтинг користувач a ,

μ_a - середній рейтинг користувача a , за множиною оцінених предметів I_a.

Для колаборативної фільтрації на основі елементів оцінка рангу переважно обчислюється на основі скоригованої косинусної подібності.

$$\hat{r}_{ui} = \frac{\sum_{c \in Q_d(u)} AjustedCosine(c,d) r_{uc}}{\sum_{c \in Q_d(u)} AjustedCosine(c,d)},$$

$$AjustedCosine(c,d) = \frac{\sum_{u \in U_c \cap U_d}(r_{uc} - \mu_u)(r_{ud} - \mu_u)}{\sqrt{\sum_{u \in U_c \cap U_d}(r_{uc} - \mu_u)^2}\sqrt{\sum_{u \in U_c \cap U_d}(r_{ud} - \mu_u)^2}},$$

$$\mu_a = \frac{\sum_{i \in I_u} r_{ui}}{|I_u|}$$

де

$Q_d(u)$- множина k - ближчих предметів для предмета d, що виставив рейтинг користувач u,

AjustedCosine(c,d) - кореляція предметів c і d за множиною спільно-оцінюючих їх юсерів $U_c \cap U_d$,

U_c - множина індексів користувачів, які виставили рейтинг предмету c.

Контентна фільтрація

У рекомендаційній системі, яка використовує фільтрацію контенту (content filtering), користувачі незалежні від інших користувачів системи. Для формування рекомендацій системі потрібний профіль користувача, що містить інформацію про інтереси користувача. Інформація про об'єкти, що цікавлять користувача, зберігається у спеціальній формі у профілі. В системі також міститься інформація про всі товари, які можна рекомендувати. Така система використовує опис об'єктів у профілі користувача, щоб знайти схожі об'єкти в базі даних та рекомендувати їх користувачеві.

Застосування такого роду фільтрації дуже зручне, коли користувачі мають конкретні, чітко визначені інтереси, і вони шукають схожі рекомендації. Перевага контентної фільтрації полягає в тому, що для надання рекомендацій не потрібна велика кількість зареєстрованих користувачів. Тобто, рекомендації не залежать від інших користувачів системи. Основне обмеження цього методу полягає в тому, що система з таким типом фільтрації не може рекомендувати нові об'єкти, які не задовольняють інтереси користувача.

Для того, щоб система мала високоякісні рекомендації, необхідно постійно оновлювати інформацію про інтереси користувача. Але існує й інша проблема - користувачі завжди хочуть мінімізувати кількість часу, яку вони витрачають на взаємодію з системою, і неохоче діляться інформацією для зворотного зв'язку з системою. Це типова проблема для всіх рекомендаційних систем, і оскільки рекомендації засновані виключно на інформації профілю користувача, тобто чим менше інформації користувач надає, тим менше відповідний набір рекомендацій користувач отримає. В такому випадку для вирішення цього завдання використовуються автоматичні системи збору інформації для підтримки профілю, але це

ускладнює реалізацію системи.

Система рекомендаційної контент-фільтрації, що обробляє текстові дані, включає наступні компоненти:

1. Попередня обробка даних.

1.1. З текстових даних виводяться слова.

1.2. Проводиться очищення та фільтрація (відбір ключових слів) відповідно таким етапам:

- видалити стоп-слова. Стоп-слова не належать до якогось конкретного предмета, а є загальною частиною словникового запасу мови. Стоп-слова зазвичай є високочастотними словами, тому їх можна класифікувати як ключові слова. Артиклі, прийменники, спілки та займенники зазвичай вважаються стоп-словами (наприклад, в англійській мові стоп-словами є такі слова, як «a», «an» та «the»).

- стемінг (виймаються кореня зі слів і поєднуються слова з одним коренем);

- витяг фраз (Виявлення таких фраз, як «hot-dog»).

1.3. Вибрані ключові слова перетворюються на вектор і кожному слову надається вага у вигляді BoW або TF-IDF.

$$BoW(t,d) = n_t,$$

$$tf - idf\,(t,d,D) = tf\,(t,d) idf\,(t,D),$$

$$tf(t,d) = \frac{n_t}{\sum_{k=1}^{l_d} n_k}, idf\,(t,D) = ln\frac{|D|}{|\{d_i \in D | t \in d_i\}|},$$

де

BoW мішок слів (Bag-of-words) - кількість входжень певного слова у документі.

TF частота слова (term frequency) - відношення кількості входжень певного слова до загальної кількості слів у документі. Таким чином, оцінюється важливість слів в іншому документі.,

IDF зворотна частота документа (inverse document frequency) - інверсія

частоти, з якою деяке слово зустрічається в документах колекції. Облік IDF знижує вагу слів, що часто використовуються. Існує лише одне значення IDF для кожного унікального слова у цьому наборі документів,

n_t - число входжень слова t до документа d,

l_d - число слів у документ d,

|D| - кількість документів у колекції,

$|\{d_i \in D | t \in d_i\}|$ - число документів з списку D, в яких є слова t (коли $n_t \neq 0$).

2. Навчання моделі. Відбувається використовуючи явні та неявні рейтинги та атрибути елементів (зважені ключові слова BoW або TF-IDF) як навчальні дані, атрибути елементів як вхідні дані для навчання та рейтинги як вихідні дані для навчання. Цей крок аналогічний до навчання моделі класифікації або апроксимації.

3. Прогнозування. На цьому етапі вивчена модель використовується для формування тематичних рекомендацій для заданого користувача (розраховуються рейтинги позицій).

Фільтрація на основі знань

Фільтрація на основі знань - пропонує продукти на основі результатів досліджень про потреби та уподобання юсерів, вибір товарів та обґрунтування рекомендацій.

Методи фільтрації на основі вмісту ґрунтуються на описах елементів та профілях уподобань користувача. Ці методи найкраще підходять, коли ви знаєте дані про предмет (назву, місце розташування, опис тощо), але не про користувача. Рекомендація на основі контенту розглядає рекомендацію як проблему класифікації конкретного користувача, вивчаючи класифікатор переваг користувача на основі характеристик елемента (див. рис. 2.5).

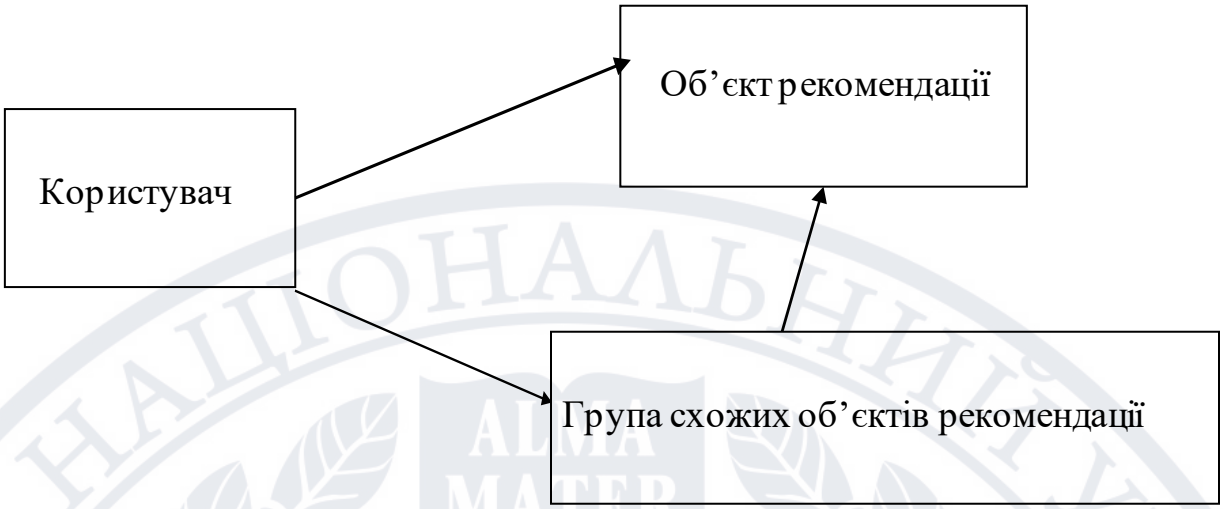

Рисунок 2.5 - Основна ідея методу фільтрації на основі вмісту

Такі системи використовують ключові слова для опису елементів і створюють профілі користувачів, щоб вказати типи елементів, які користувачі віддають перевагу. Інакше кажучи, ці алгоритми намагаються рекомендувати елементи, схожі елементи, які подобалися користувачам у минулому чи розглядають нині. У багатьох випадках він не використовує механізми входу користувачів для створення тимчасових профілів. Зокрема, різні елементи-кандидати 17 порівнюються з елементами, попередньо оціненими користувачем, та рекомендуються найбільш підходящі елементи. Цей підхід сягає своїм корінням у вивчення пошуку та фільтрації інформації. Наприклад, для створення профілю користувача, система повинна фокусуватися на двох типах інформації: модель уподобань користувача та історія взаємодії користувача з рекомендаційною системою. По суті ці методи використовують профілі елементів (тобто набори різних атрибутів і характеристик), які характеризують об'єкти в системі. Алгоритми, які представляють об'єкти, використовуються абстрагування функціональності елементів у системі.

Широко використовуваний алгоритм - це уявлення tf-idf (також зване уявленням векторного простору). Система створює профіль користувача на основі контенту на основі зважених векторів ознак елементів. Ваги вказують на важливість кожної функції для користувача і можуть бути розраховані на

основі векторів контенту, які оцінюються індивідуально з використанням різних методів. Прості підходи використовують векторне усереднення оцінюваних елементів, у той час як інші просунуті методи використовують методи машинного навчання, такі як класифікатори байєса, кластерний аналіз, дерева рішень, штучні нейронні мережі і т. д. Оцініть ймовірність того, що елемент сподобається. Ключова проблема у фільтрації на основі контенту полягає в тому, чи може система дізнаватися переваги користувача з дій користувача в одному джерелі контенту і використовувати їх для інших типів контенту. Якщо система обмежена рекомендацією того ж типу контенту, який вже використовує користувач, цінність рекомендаційної системи набагато вища, ніж якщо вона могла рекомендувати інші типи контенту з інших сервісів або менше. Наприклад, корисно рекомендувати статті новин на основі переглядів новин. Однак було б набагато корисніше мати можливість рекомендувати музику, відео, продукти, обговорення тощо з різних сервісів на основі вашого перегляду новин. Щоб подолати це більшість систем рекомендацій на основі контенту в даний час використовують деяку форму гібридної системи.

Системи рекомендацій на основі змісту можуть також включати системи рекомендацій на основі думок. У деяких випадках користувачі можуть залишати текстові відгуки та відгуки про товари. Ці створені користувачами тексти є неявними даними для рекомендаційних систем, оскільки є потенційно багатим джерелом як характеристик/аспектів продукту, і оцінок/відносин користувачів до цього елемента. Функції, витягнуті з оглядів, створених користувачами, є розширеними метаданими елементами, оскільки вони також відображають аспекти таких елементів, як метадані. Вилучений функціонал викликає широкий інтерес у користувачів. Налаштування з оглядів можна розглядати як рейтинги користувачів для кожної функції.

Загальні підходи до системи рекомендацій на основі думок використовують різні методи, такі як аналіз тексту, пошук інформації, аналіз

настроїв та глибоке навчання.

Рекомендаційні системи, що базуються на знаннях, можуть бути засновані на обмеженнях (системи, що базуються на обмеженнях) або на прикладах (системи, засновані на прикладах). У рекомендаційній системі на основі обмежень (малюнок х) користувачі спочатку задають обмеження на атрибути об'єкта. Потім система перетворює ці обмеження у внутрішнє уявлення та використовує правила, засновані на знаннях, що відображають перетворені обмеження. Об'єкт (наприклад, внутрішнє обмеження відображення «вартість < $10 000 ∧ make = Damage» відображатиме набір автомобілів, що задовольняють цьому обмеженню) поверне результат, і користувач може змінити обмеження (наприклад, змінити колір автомобіля на In the У рекомендаційній системі, заснованій на прикладі (рис. у), користувач спочатку вказує цільові об'єкти, схожі на бажаний об'єкт, потім система перетворює опис цільового об'єкта у внутрішнє уявлення, і створюється база знань, що відображає перетворений опис цільового об'єкта на об'єкт (наприклад, опис цільового об'єкта у внутрішньому поданні «Автомобіль Джеймса Бонда») представлений цією метою, зіставляється з набором автомобілів, які задовольняють об'єкту) повертає результат. Потім користувач може змінити атрибути цільового об'єкта (наприклад, додати марку автомобіля Джеймса Бонда). Такі зміни називають критикою. Відсутність методів, заснованих на знаннях, що налаштовуються тільки для конкретних областей, на відміну від методів спільної роботи та фільтрації контенту.

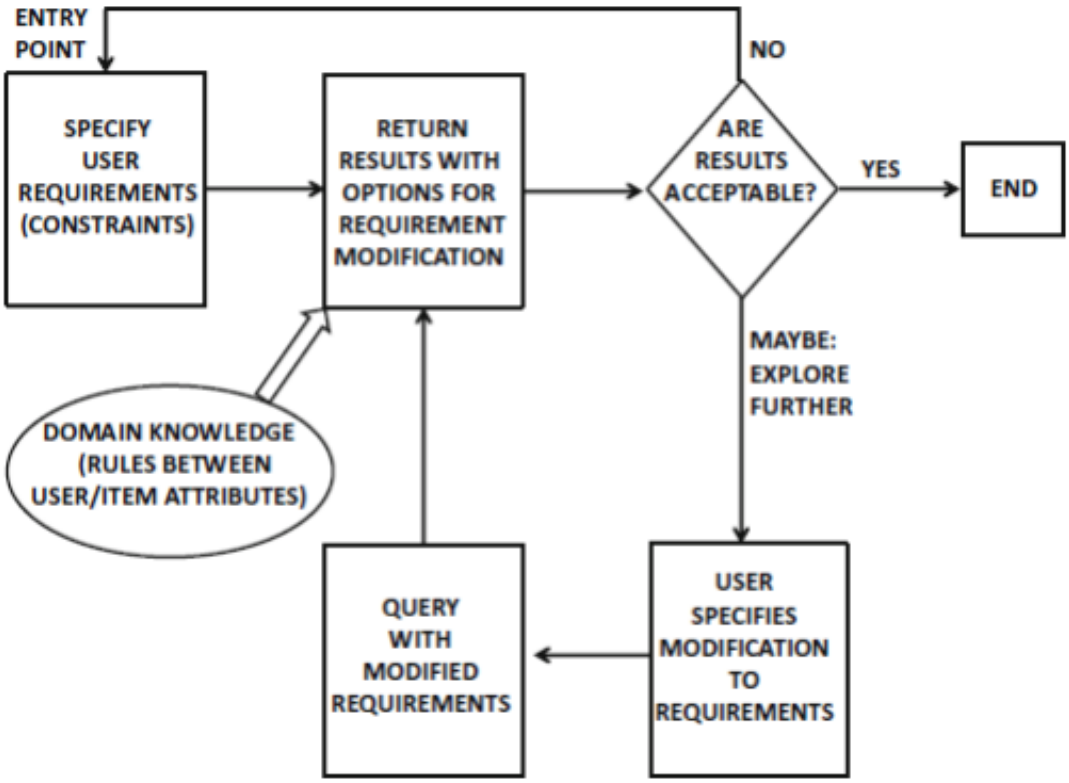

Рисунок 2.6 - Інтерактивний процес у рекомендаційній системі на основі обмежень.

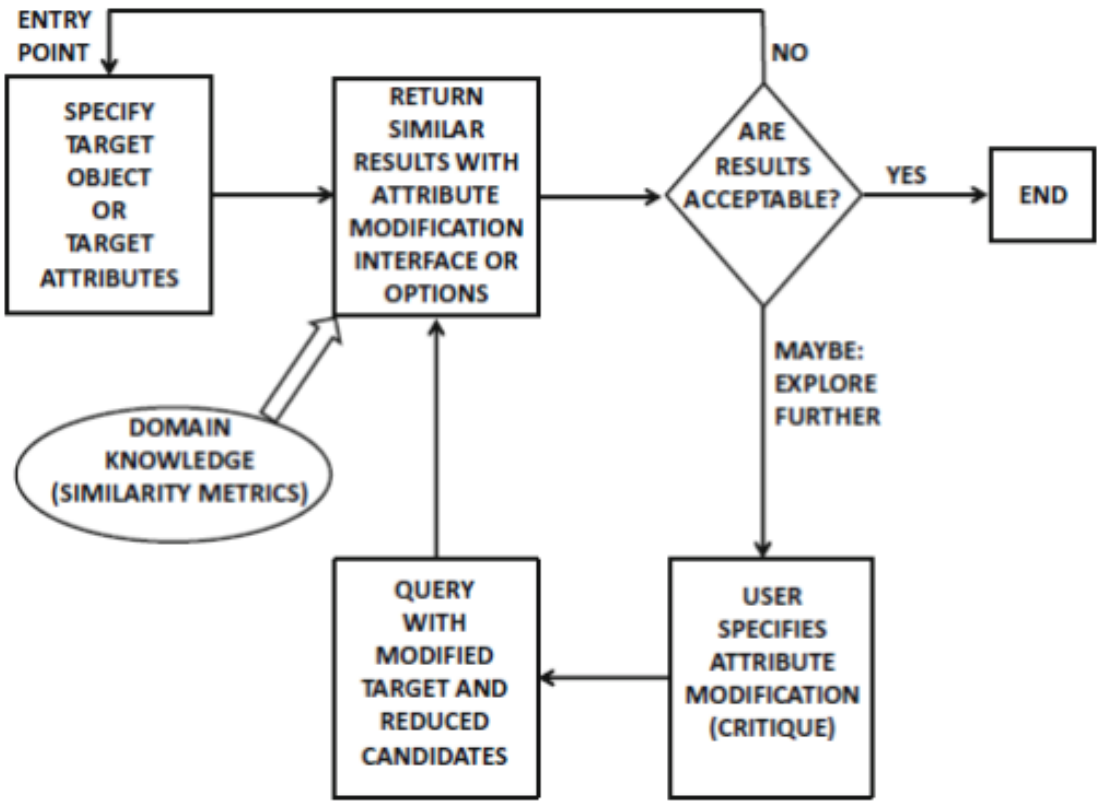

Рисунок 2.7 - Інтерактивний процес у рекомендаційній системі на основі прикладів

Гібридні системи

Гібридні системи зазвичай поєднують колаборативну фільтрацію і контентну фільтрацію. Це дозволяє вирішити ряд проблем, які виникають при використанні цих методів окремо. В гібридній системі інформація про інтереси користувачів представлена в профілі в двох видах - як перелік властивостей певного об'єкта і його оцінка користувачем. Ця характеристика є як перевагою, так і недоліком системи. Перевагою є повнота інформації, що дозволяє використовувати ефективні алгоритми фільтрації і формувати необхідні рекомендації. Недоліком є те, що користувач повинен ввести більше інформації в свій профіль, яка завжди дуже неохоче, або не робиться взагалі

	Контекстна фільтрація	**Колаборативна фільтрація**	**Фільтрація на знаннях**	**Гібридна фільтрація**
Проблема холодного старту	Відсутня	Присутня	Відсутня	Відсутня
Складність реалізації	Низька	Низька	Середня	Висока
Специфіка роботи	Інтернет-магазини, портали кіно, музики та ін.	Інтернет-магазини, портали кіно, музики та ін.	Інтернет-магазини	Будь-яка область
Точність рекомендацій	Середня	Середня	Висока	Висока
Потенційна швидкість роботи	Висока	Середня	Середня	Висока
Залежність від інших користувачів	Відсутня	Присутня	Відсутня	Залежить від реалізації

Таблиця 2.1 - Порівняльний аналіз методів підбору рекомендацій

Висновок до розділу 2

На підставі проведеного порівняльного аналізу можна зробити висновок про недоцільність розробки гібридної рекомендаційної системи через високу складність реалізації і відсутності команди розробників, між якими можна було б розділити виконувані завдання. Колаборативна фільтрація реалізована в соціальних мережах і показує досить сумнівні результати. Також великими мінусами є проблема так званого холодно старту, коли систему необхідно навчити шляхом вираження користувацьких симпатій/антипатій до результатів, що видаються, і залежність алгоритму аналізу від інших користувачів. Таким чином, ми приходимо до двох можливих варіантів, які по суті аналізують дані про сам об'єкт - це контентна фільтрація і фільтрація заснована на знаннях. Спочатку система працює, використовуючи контентну фільтрацію - вона підбирає новини, схожі з тими, які читає користувач, але після видачікористувач може зреагувати на підібрані рекомендації – поставити позначки «Мені подобається» Додатокзапам'ятовує дії користувача і враховує їх надалі.

РОЗДІЛ 3
ПОСТАНОВКА ЗАДАЧІ ТА ІНСТРУМЕНТИ

3.1. Постановка задачі.

Розробити мультиплатформенний додаток.

Розробити мобільний додаток для таких ОС: Android, iOS, macOS, WEB.

Етапи створення мобільних додатків:

- Визначення користувацької аудиторії додатку;
- вибір основних платформ додатку;
- розробка концепції додатку;
- формування технічного завдання;
- дизайн інтерфейсу;
- розробка;
- тестування;
- влагодження;
- реєстрація та реліз на онлайн платформах;
- взаємодія з користувачами;
- періодичні оновлення та покращення;

Завданням та ціллю даної роботи є розробка мультиплатформного додатку для пошуку музичних інструментів «Musicians Shop». Отже, мультиплатформенний додаток буде являти собою онлайн дошку оголошень, з адаптивним підбором даних під кожного користувача окремо, що надає можливість переглядати актуальні оголошення з можливістю пошуку.

Електронна дошка оголошень функціонально подібна звичайній: це додаток, де кожен бажаючий може вивісити своє оголошення, а всі відвідувачі додатку - прочитати та зберегти його, а також зв'язатися з автором. Електронна дошка оголошень, зазвичай, поділена на кілька тематичних розділів, відповідно до змісту оголошень.

Більшість дошок оголошень - безкоштовні але з деяким платним функціоналом. Для розміщення оголошення користувачеві достатньо ввести в спеціальне поле тему, ім'я/нікнейм або назву організації та координати (адреса

електронної пошти, поштова адреса, номер телефону, URL сайту тощо). форма. (Набори даних залежить від конкретних ресурсів). Як правило, відображається лише ім'я автора та тематика оголошення, а для перегляду повного тексту оголошення користувачеві необхідно перейти за посиланням, що веде до нього.. У деяких дошках оголошення можуть подавати тільки зареєстровані користувачі, в деяких - всі. Зараз в інтернеті існує тисячі і навіть десятки тисяч дошок оголошень. Зазвичай кожна з них присвячується якомусь одного виду оголошень. Існують національні дошки оголошень, призначених для жителів конкретної місцевості.

Електронні дошки оголошень бувають двох видів: модеровані (ті, у яких є так званий модератор - людина, яка контролює роботу цієї дошки) і немодеровані - працюють автоматично.

Більшість дошок оголошень в інтернеті припускають розміщення оголошень про продаж і купівлю товарів і послуг. Але є і такі сервіси, які пропонують розміщувати оголошення про передачу різних речей в дар, а також про пошук нових господарів для домашніх тварин, які передаються в добрі руки безкоштовно. Безкоштовні оголошення - це найкращий спосіб заявити про себе, свої товари і послуги в мережі Інтернет. На сторінках дошок оголошень представлені тисячі безкоштовних оголошень про нерухомість, авто, послуги, пошук роботи, знайомства, купівлю та продаж обладнання, комп'ютерів, засобів зв'язку, тощо. Дошки оголошень дозволяють розмістити оголошення з фотографією і без додаткової реєстрації. Ви можете розмістити не тільки безкоштовне оголошення, але і оголошення на комерційній основі - для максимальної ефективності.

Таким чином, основний функціонал для користувачів:

- Реєстрація користувачів у додатку (пошта та пароль).
- Перегляд усіх оголошень на головній сторінці які адаптивно підбираються в залежності від вподобань користувача.
- Можливість пошуку оголошення.
- Можливість вподобання оголошення.

- Можливість створювати, редагувати та видаляти свої оголошення.
- Можливість перегляду своїх оголошень, та вподобаних оголошень.
- Можливість перегляду сторінки автора оголошення.
- Перегляд статистики вподобань своїх оголошень.
- Користувач може редагувати або видалити свій профіль.
- Валідація текстових полів.
- Адаптивність під різні особливості платформи ТВ девайсу користувача.

3.2. Дослідження можливостей і огляд мови програмування Dart.

Мова програмування є сполучною ланкою між розробником та додатком. Існує безліч мов, якими пишуть мобільні додатки. Сьогодні неможливо працювати у сфері розробки, знаючи лише одну мову програмування. Щороку з'являються нові мови, але всі вони створюються для досягнення максимальної ефективності конкретного завдання. А головне затребувано розширюється та просувається.

Рисунок 3.1 - Логотип мови Dart

Dart - Мова програмування розроблена Google. Дартс позиціонується як якісний Один із розробників заміни/альтернативної мови JavaScript Марк Міллер (Mark Miller) С. Міллер) писав, що JavaScript «принципово недосконалий», «Javascript має фундаментальні недоліки..., які не можуть виправлені». Ось чому Dart був створений.[2]

Першу публічну інформацію про цю мову програмування було опубліковано на конференції розробників Goto 12 вересня 2011 року. 10 жовтня 2011 відбулася офіційна презентація мови Google Dart.

Завдання, які були поставлені розробникам мови:

- Створіть структуровану, але гнучку мову для веб-програмування.
- Для спрощеного вивчиння, зробити мову схожу на вже існуючі.
- Висока продуктивність отриманої програми, як у браузерах, так і в інших середовищах, від смартфонів до серверів.
- Спочатку було запропоновано два методи запуску Dart-програм: за допомогою віртуальної машини (у браузерах, що підтримують мову) або за допомогою проміжного перекладу Javascript (універсальний)..

15 листопада 2013 компанія Google випустилі Dart SDK 1.0, першу стабільну версію мови програмування.

4 липня 2014 року ECMA затвердила першу редакцію мовного стандарту, і стандарт отримав назву ECMA-408.

Dart - це оптимізована для клієнта мова, яка використовується для розробки швидких програм на будь-якій платформі. Мета створення Dart – надати найбільш продуктивну мову програмування для мультиплатформної розробки у поєднанні з гнучкою платформою часу виконання для фреймворків додатків. [59]

Мова визначається його технічною оболонкою, виборами, зробленими під час розробки, які визначають особливості та сильні сторони мови. Dart розроблений як технічна оболонка, яка особливо добре підходить для розробки на стороні клієнта, забезпечуючи як розробку (hot restart до секунди), так і високоякісний виробничий досвід для широкого спектру компіляцій (веб, мобільні та настільні комп'ютери).

Dart також є основою Flutter. Dart надає мову та середовище виконання, в якому працюють програми Flutter, але Dart також підтримує багато основних завдань розробників, такі як форматування коду, аналіз та тестування.

Мова Dart є типобезпечною і використовує статичну перевірку типів, щоб гарантувати, що значення змінної завжди відповідає статичному типу змінної. Це іноді називають набором звуків. Типи обов'язкові, але інструкції типів необов'язкові через визначення типів. Система типізації Dart також є гнучкою, що дозволяє використовувати динамічну типізацію разом із перевіркою під час

виконання. Це корисно для експериментів та коду, який має бути особливо динамічним.

Dart пропонує зручну та надійну нульову безпеку (null safety), тобто значення не можуть бути нульовими (null), якщо не вказати, що вони можуть бути нульовими (<T?>). Надійна нульова безпека дозволяє Dart захистити вас від нульових винятків під час виконання за допомогою статичного аналізу коду. На відміну від багатьох інших нульових мов, Dart визначає, що змінні не можуть бути нульовими, ця змінна завжди не допускає нуль. Якщо перевірити запущений код у налагоджувачі, виявиться, що відсутність нульових значень зберігається під час виконання, отже, надійна нульова безпека.

3.3. Огляд і дослідження можливостей Flutter SDK, плагінів, пакетів та бібліотек.

Рисунок 3.2 - Логотип Flutter SDK

Flutter - SDK, що містить відкритий вихідний код для створення мобільних програм від Google. Він використовується для розробки програм для Android та iOS, веб-сайтів, але через деякий час дозволяє створювати настільні програми, а також є єдиним способом розробки програм для Google Fuchsia. [3]

Перша версія Flutter називалася Sky і працювала тільки на Android. Про це було оголошено на саміті розробників Dart у 2015 році, і заявлена здатність відображати 120 кадрів на секунду. 4 грудня 2018 року FlutterLive анонсувала першу стабільну версію 1.0.

Архітектура:

- платформа/мова Dart;
- рушій Flutter SDK;

- основна бібліотека Foundation;
- набори Flutter Material/Cupertino віджетів;

Flutter додатки пишуться виключно на мові Dart.

В Android, а також Windows, macOS і Linux з Flutter Desktop Embedding Flutter працює на віртуальній машині Dart з використанням JIT-компілятора. Flutter на iOS використовує компіляцію AOT через обмеження виконання динамічного коду в AppStore.

Однією з основних переваг платформи Dart є «гарячий перезапуск». Це коли зміни у вихідному коді застосовуються негайно у програмі, що працює, без необхідності перезапуску.

Рушій Flutter написаний в основному на C++ та підтримує низькорівневий рендеринг із використанням графічної бібліотеки GoogleSkia. Він також має можливість взаємодіяти із залежними від платформи SDK для Android та iOS.

Бібліотека Foundation написана на Dart і містить основні класи та методи для створення додатків Flutter та взаємодії з двигуном Flutter.

Розробка інтерфейсу користувача для програми Flutter зазвичай включає використання і створення різних віджетів. Віджет Flutter - це постійний опис будь-якої частини інтерфейсу користувача. Усі графічні об'єкти, включаючи текст, фігури та анімацію, створюються за допомогою віджетів. Складні віджети створюються шляхом поєднання простіших віджетів. [46]

Однак є можливість створювати програми Flutter без віджетів, безпосередньо викликаючи методи бібліотеки Foundation для керування canvas.

Фреймворк Flutter складається із двох наборів віджетів, що відповідають конкретним заявам про дизайн, Material Design від Google та Cupertino для імітації дизайну додатків на Apple macOS та iOS.

Flutter DevTools - набір інструментів налагодження та продуктивності програмного забезпечення для Dart та Flutter.

Можливості що надає DevTools:

- Перегляд та аналіз UI та стан програми Flutter.
- Діагностика продуктивності UI у програмі Flutter.

- Профілювання ЦП для програми Flutter або Dart.
- Профілювання та аналіз мережі для програми Flutter.
- Можливість налагодження вихідного коду програми Flutter або Dart.
- Усунення проблем з виділенням пам’яті в програмі Flutter або Dart.
- Перегляд поточного журналу (логгера) та інформацію під час діагностики запущеної програми Flutter або Dart.
- Аналіз поточного коду програми та її розміру.

GetX (get) - Дуже легке та потужне рішення Flutter. Він поєднує в собі високоефективне управління станом, інтелектуальне впровадження залежностей і швидке та прагматичне керування маршрутами.

Рисунок 3.3 - Логотип GetX

GetX має три основні принципи. Це означає, що пріоритетом для всіх ресурсів бібліотеки є продуктивність, ефективність і організованість.

Продуктивність: GetX фокусується на продуктивності та мінімальному споживанні ресурсів. GetX не використовує Streams або ChangeNotifier.

Ефективність: GetX використовує простий та елегантний синтаксис. Незалежно від того, що вам потрібно зробити, GetX завжди є простий вихід. Це заощаджує час розробки та забезпечує максимальну продуктивність програми.

Загалом розробники повинні подбати про видалення контролерів з пам'яті. Це не потрібно GetX, оскільки ресурси видаляються з пам'яті, якщо вони не використовуються за умовчанням. Якщо потрібно щось зберегти в пам'яті, ви можете явно оголосити «permanent: true» у своїй залежності. Таким чином, ви

не тільки заощаджуєте час, а й знижуєте ризик виникнення непотрібних залежностей від пам'яті. За замовчуванням він також ліниво завантажує залежності. [63]

Організація: GetX дозволяє чітко розділити уявлення, логіку уявлення, бізнес-логіку, реалізацію залежностей та навігацію. Це рішення не залежить від дерева віджетів (візуалізації), оскільки переходу між корінням не потрібно ніякого контексту. Для доступу до контролерів/блоків через внесенийВиджет контекст не потрібно, тому логіка подання та бізнес-логіка повністю відокремлені від рівня рендерингу. Немає потреби впроваджувати класи контролера/моделі/блоку у дерево віджетів через мультипровайдер. Для цього GetX використовує власну функцію застосування залежностей, щоб повністю відокремити DI від подання.

З GetX розробники точно знають, де знаходяться всі функції їхнього застосування, і за умовчанням бачать чистий код. GetX не тільки полегшує розробку, але і дозволяє спільно використовувати модулі, які раніше було неможливо у Flutter. BLoC - це відправна точка для організації коду Flutter, що відокремлює бізнес-логіку від рендерингу. GetX - це природна еволюція цього, що відокремлює як бізнес-логіку, а й логіку уявлення. Додаткові реалізації залежностей і маршрутів також відокремлені один від одного, і шар даних знаходиться поза його межами. Ви знаєте, де все і все дуже просто.[55]

GetX - це найпростіший, практичніший і масштабованіший спосіб створення високопродуктивних додатків за допомогою Flutter SDK. Він має велику екосистему, яка добре працює разом, що спрощує роботу як для новачків, так і для експертів.

GetX має безліч функцій, що дозволяють почати програмування, ні про що не турбуючись, але кожна з цих функцій знаходиться в окремому контейнері і виконується тільки після використання. Якщо використовується лише керування станом, компілюється лише керування станом. Нічого з керуванням станом компілюватися не буде, якщо використовуються лише маршрути.

GetX має величезну екосистему, велику спільноту, велику кількість учасників і підтримуватиме Flutter, поки він існує. GetX може запускати той самий код на Android, iOS, в Інтернеті, Mac, Linux, Windows і навіть на серверах. Get Server дозволяє коду, написаному у вашому додатку, бути повністю доступним на сервері. [54]

Крім того, Get CLI можна використовувати для повної автоматизації всього процесу розробки як на сервері, так і в додатку.

Існують розширення для VSCode, Android Studio та Intellij для подальшого підвищення вашої продуктивності.

Несподівані помилки часто виникають після оновлення Flutter. Можуть виникати помилки компіляції, і часто ще існують неусунені помилки. Розробники повинні знати причину помилки, відстежувати помилку та відкривати проблему у відповідному репозиторії, щоб побачити рішення. Get централізує основні ресурси розробки (стан, залежності та управління маршрутами) і дозволяє вам додати один пакет у вашу pubspec і приступити до роботи. Після оновлення Flutter просто оновіть залежність Get, і все готово. Get також вирішує проблеми сумісності. Одна версія використовує залежність, а інша версія іншу залежність, тому часто версія пакета несумісна з іншою версією пакета. Це також не проблема при використанні Get, оскільки все знаходиться в одному пакеті та повністю сумісне. [56]

Flutter - це простий і дуже сучасний фреймворк, але у Flutter є шаблони, які можуть не сподобатися більшості розробників, наприклад Navigator.of(context).push(context, builder [...]). Get спрощує розробку. Замість того, щоб писати 8 рядків коду лише для виклику маршруту, просто виконайте Get.to(Home()) і все готове. Наступна сторінка готова. Якщо Flutter спрощує завдання, GetX робить її дуже простою. Управління станом та управління залежностями у Flutter також є предметом обговорення, оскільки на pub.dev є сотні шаблонів. Але немає нічого простішого, ніж додати ".obs" в кінець змінної і помістити її у віджет Obx. Будь-які оновлення цієї змінної автоматично оновлюватимуться на екрані.

GetX спрощує розробку, не переймаючись продуктивністю. Продуктивність Flutter вже вражає, але бувають ситуації, коли розробники використовують менеджери станів та локатори для виділення блоків, сховищ, контролерів тощо. У цьому випадку ви повинні вручну викликати виняток для цієї залежності, коли вона більше не потрібна. GetX просто використовує контролер, і коли контролер більше не використовується, він просто видаляється з пам'яті. Завдяки SmartManagement, все, що не використовується, видаляється з пам'яті, тому вам не потрібно турбуватися ні про що, крім програмування. Розробники завжди стежать, щоб вони споживали мінімально необхідну кількість ресурсів, не прописуючи при цьому логіку.

Real Separation - концепція «відділення презентації від бізнес-логіки». Це не особливість BLoC, MVC, MVVM, інші стандарти ринку мають цю концепцію. Однак цю концепцію часто можна пом'якшити у Flutter за рахунок використання контекстів. Якщо вам потрібний контекст для пошуку InheritedWidget, він знадобиться вам або у вашому поданні, або вам потрібно буде передати контекст у параметрі. Це рішення не підходить, оскільки завжди є залежність від бізнес-логіки View. GetX не повністю забороняє використання StatefulWidgets, InitState і т. д. Контролер має життєвий цикл і не залежить ні від чого у поданні, наприклад, коли йому потрібно зробити запит REST API. Коли ви використовуєте onInit для запуску http-дзвінка, змінні встановлюються при надходженні даних. GetX є повністю реактивним (працює в потоці), тому при завантаженні елемента всі віджети, що використовують цю змінну, автоматично оновляться у своєму поданні. Це дозволяє вам працювати виключно в інтерфейсі користувача без необхідності відправляти що-небудь у вашу бізнес-логіку, крім користувацьких подій (наприклад, натискання кнопок). Це дає вам свободу самостійно писати та тестувати бізнес-логіку. [60]

GetX постійно оновлюється та впроваджуються нові функції. [13]

Overlay Support (plugin: overlay_support) - плагін Flutter з підтримкою накладень дозволяє легко програмно створювати тости та сповіщення. [14]

Carousel Slider (package: carousel_slider) - пакет Flutter для створення віджету слайдера каруселі, який підтримує нескінченну прокручування та власні дочірні віджети.. [15]

Cupertino Icons (package: cupertino_icons) - бібліотека Flutter, що містить стандартний набір значків, що використовуються віджетами Flutter із Cupertino (iOS). [16]

Syncfusion Flutter Charts (package: syncfusion_flutter_charts) - бібліотека флаттер-діаграм, що містить віджети візуалізації даних, такі як декартові діаграми та кругові діаграми. Створюйте інтерактивні високопродуктивні анімовані діаграми в реальному часі. [17]

MIME (plugin: mime) - пакет Flutter керує визначеннями типів MIME та обробки мультимедійних потоків типів MIME. Ви можете використовувати клас MimeTypeResolver для визначення MIME-типу файлу. Він підтримує як використання розширень імен файлів, і байтове представлення файлів. Існує вбудований екземпляр MimeTypeResolver, доступний через функцію верхнього рівня lookupMimeType. Цей вбудований екземпляр має найпоширеніші розширення імен файлів та зареєстровані байти. Перетворювачі користувача можуть бути створені шляхом створення екземпляра MimeTypeResolver і додавання розширень імен файлів і байтів за допомогою addExtension і addMagicNumber. [18]

Logger (package: logger) - невеликий пакет логів Flutter, який простий у використанні і може бути розширений і налаштований відповідно до ваших потреб. Цей logger дуже нагадує реєстратор Android. [19]

File Picker (plugin: file_picker) - це плагін Flutter, який дозволяє вам вибирати один або кілька файлів з підтримкою фільтрації та зміни розширень за допомогою провідника файлів. [20]

Url Strategy (plugin: url_strategy) - пакет веб-застосунків Flutter, який дозволяє вам встановити веб-URL. [21]

Url Launcher (plugin: url_launcher) - плагін Flutter для запуску URL-адрес. Підтримує схеми мережі, телефону, SMS та електронної пошти. [22]

Path Provider (plugin: path_provider) - це плагін Flutter для пошуку часто використовуваних місць у файловій системі хост-платформи, таких як каталоги тимчасових даних та даних додатків. Підтримує Windows, Linux, MacOS, iOS, і Android. Не всі методи підтримуються на всіх платформах. [23]

Android Path Provider (plugin: android_path_provider) - плагін Flutter для отримання каталогу Android. (Завантаження, відео, зображення ...). Цей плагін працює тільки на платформі Android. [24]

Package Info Plus (plugin: package_info_plus) - плагін Flutter для запиту інформації про набір програм, такий як CFBundleVersion для iOS та versionCode для Android. [25]

Flutter Local Notifications (plugin: flutter_local_notifications) - плагін Flutter який є міжплатформенним для відображення та планування локальних повідомлень для програм Flutter з індивідуальним налаштуванням для кожної платформи. [26]

Flutter Lints (package: flutter_lints) - цей пакет Flutter містить набір рекомендованих лінтерів для додатків, пакетів та плагінів Flutter для просування хороших методів кодування. Цей пакет базується на рекомендованому Dart наборі lints, знайденому в package:lints. Лінти виявляються синтаксичним аналізатором дротика, який статично перевіряє код дротика. У IDE з підтримкою Dart проблеми, виявлені аналізаторами, відображаються в інтерфейсі користувача. Крім того, ви можете запустити аналіз Flutter та викликати аналізатор вручну. [27]

Flutter Launcher Icons Maker (flutter_launcher_icons_maker plugin) - це пакет Flutter, який полегшує завдання оновлення значків запуску Flutter. Інструмент повністю гнучкий і дозволяє вибирати платформи, на яких ви хочете, щоб ваші значки запуску оновлювалися. Якщо ви хочете, ви також можете зберегти свої старі значки запуску на випадок, якщо вони знадобляться вам в майбутньому. [28]

3.4. Огляд технологій Firebase

Firebase - американський постачальник хмарних послуг, заснований Ендрю Лі та Джеймсом Темпліном у 2011 році та придбаний Google у 2014 році.[5]

У травні 2012 року він отримав 1,4 мільйона доларів від Flybridge Capital Partners, Greylock Partners та NEA, а в червні 2013 року залучив 5,6 мільйона доларів від Union Square Ventures та Flybridge Capital Partners.

Рисунок 3.4 - логотип Firebase

Основний сервіс - це хмарна СУБД класу NoSQL, яка дозволяє розробникам додатків зберігати та синхронізувати дані між кількома клієнтами. Він підтримує інтеграцію з додатками для операційних систем Android та iOS та реалізує API для додатків JavaScript, Java, Objective-C та Node.js. Також можна безпосередньо працювати з базами даних у стилі REST із ряду фреймворків JavaScript. , AngularJS, ReactJS, Vue.js, Ember.js та Backbone.js. Надає API для шифрування даних. [50]

Пропонує хостинг для зберігання статичних файлів (CSS, HTML, JavaScript тощо), доставку через CDN та службу автентифікації клієнта з використанням коду лише на стороні клієнта, запущений 13 травня 2014 р., підтримка входу до системи. Facebook, GitHub, Twitter, Google Firebase Простий вхід.

Firebase пропонує аналітику, бази даних, обмін повідомленнями, звіти про збої та багато іншого, щоб ви могли працювати швидко та зосередитися на своїх користувачах.

Firebase побудований на інфраструктурі Google і автоматично масштабується навіть для великих програм.

Продукти Firebase добре працюють окремо або разом, чудово обмінюючись даними.

Firebase зберігає та синхронізує глобально додані дані, швидко та безпечно обслуговує ресурси додатків, а також забезпечує просту та безпечну автентифікацію користувачів.

Firebase допомагає створювати якісні програми, розширювати базу користувачів та заробляти більше грошей. Кожна функція працює окремо, але разом вони працюють ще краще.

Firebase Authentication спрямована на те, щоб спростити створення безпечних систем автентифікації, а також покращити вхід і адаптацію для кінцевих користувачів. Він надає наскрізне рішення ідентифікації, підтримуючи облікові записи електронної пошти та пароля, авторизацію через телефон, а також є можливість входу у Google, Facebook, Twitter і GitHub тощо. [5]

Більшості програм потрібно знати особу користувача. Знання особи користувача дозволяє програмі безпечно зберігати дані користувача у хмарі та забезпечувати однакову персоналізацію на всіх пристроях користувача.

Firebase Authentication надає серверну службу, простий у використанні SDK і готову до використання бібліотеку інтерфейсу користувача для автентифікації користувачів у вашому додатку. Інструмент підтримує автентифікацію за допомогою паролів, номерів телефонів, популярних федеративних постачальників ідентифікаційної інформації, таких як Facebook, Google, Twitter, та інших. [49]

Автентифікація Firebase тісно інтегрована з іншими сервісами Firebase та використовує галузеві стандарти, такі як OAuth 2.0. і OpenID Connect, тож її можна легко інтегрувати з вашим власним сервером.

Коли оновлюється Firebase Authentication з Identity Platform, розблоковуються додаткові функції, такі як багатофакторна автентифікація,

функції блокування, журнал активності користувачів і аудиту, підтримка SAML і загальна підтримка OpenID Connect, мультитенантність і підтримка корпоративного рівня.

Firebase Authentication з Identity Platform - це додаткове оновлення, яке додає кілька нових функцій до Firebase Authentication.

Це оновлення не потребує міграції - існуючий клієнтський SDK і код SDK адміністратора працюватимуть, як і раніше, і додаток отримає миттєвий доступ до таких функцій, як розширене журналювання та підтримка корпоративного рівня та угоди про рівень обслуговування. За допомогою додаткового коду можна додати багатофакторну автентифікацію, функції блокування та підтримку постачальників SAML і OpenID Connect.

Firebase Authentication with Identity Platform має іншу схему ціноутворення порівняно з базовим продуктом. Після оновлення проекти безкоштовного плану (Spark) будуть обмежені до 3000 щоденних активних користувачів, а проекти плану з оплатою за використання (Blaze) стягуватимуться плату за використання понад безкоштовний рівень у 50 000 активних користувачів щомісяця. Перед оновленням переконайтеся, що ви розумієте наслідки для оплати.

Можна входити в систему користувачів Firebase за допомогою FirebaseUI як повного рішення для автентифікації або за допомогою Firebase Authentication SDK, щоб вручну інтегрувати один чи кілька методів входу у додаток.

FirebaseUI надає рішення для автоматичної автентифікації, яке обробляє потік користувальницького інтерфейсу входу в систему. наприклад за допомогою адрес електронної пошти та паролів.

FirebaseUI це настроюване рішення для авторизації з відкритим кодом, яке обробляє потоки інтерфейсу користувача для входу користувачів. Компонент FirebaseUI Auth реалізує найкращі методи автентифікації в мобільних та десктоп додатках і веб-сайтах, що може максимізувати конверсію під час входу та реєстрації для вашої програми.

Компонент FirebaseUI Auth реалізує найкращі методи автентифікації на мобільних пристроях і веб-сайтах, що може максимізувати конверсію під час входу та реєстрації для вашої програми. Він також обробляє крайні випадки, як-от відновлення облікового запису та зв'язування облікових записів, які можуть бути вразливими до безпеки та які можуть бути схильними до ексепшинів та помилок, щоб правильно їх обробити.

FirebaseUI легко настроюється відповідно до інших візуальних стилів у вашому додатку, і він є відкритим вихідним кодом, тому ви не обмежені в реалізації бажаного досвіду користувача.

Firebase Auth було створено однією і тією ж командою розробників Google, яка також розробила Google Sign-in, Smart Lock і Chrome Password Manager, система безпеки Firebase застосовує внутрішній досвід Google щодо керування однією з найбільших баз даних облікових записів у світі.

Налаштування вашої власної системи автентифікації може зайняти місяці, і для подальшого обслуговування цієї системи потрібна команда інженерів. Налаштування всієї системи автентифікації за допомогою Firebase Auth для додатку можна зробити дуже швидко використовуючи менш ніж за 10 рядків коду, що може обробити навіть такі складні випадки, як злиття облікових записів.

Автентифікуйте користувачів за допомогою їхніх електронних адрес і паролів. Firebase Auth SDK дозволяє створювати та керувати користувачами, які входять до системи зі своєю адресою електронної пошти та паролем. Аутентифікація Firebase також обробляє надсилання електронних листів для скидання паролю.

Щоб увійти користувачем у програму, спочатку потрібно отримати від нього облікові дані для автентифікації. Цими обліковими даними можуть бути адреса електронної пошти та пароль користувача або токен OAuth від відповідного постачальника посвідчень. Потім потрібно передати ці облікові дані Firebase Authentication SDK. Потім серверні служби перевірять ці облікові дані та повернуть відповідь клієнту.

Після успішного входу в систему ви можете отримати доступ до основної інформації профілю користувача та контролювати доступ користувачів до даних, які зберігаються в інших продуктах Firebase. Також є можливість перевірити особистість користувача за допомогою власної серверної служби, використовуючи наданий токен автентифікації.

Для адреси електронної пошти, пароля чи номера телефону для входу та будь-яких федеративних постачальників ідентифікаційної інформації, які потрібно підтримувати, треба активувати їх в консолі Firebase і виконати будь-яку конфігурацію, яку вимагається постачальником ідентифікаційної інформації, наприклад налаштування URL-адреси переспрямування OAuth.

Для входу з адресою електронної пошти та паролем зробіть flow, який пропонує користувачам ввести свої адреси електронної пошти та паролі. Для входу за номером телефону створіть flow, який запропонує користувачам ввести їхній номер телефону, а потім код із отриманого SMS-повідомлення. Для об'єднаного входу запровадьте flow, необхідний кожному з методів. Для коректної роботи потрібно передавати адресу електронної пошти та пароль користувача або маркер OAuth, отриманий від федеративного постачальника ідентифікаційної інформації, до Firebase Authentication SDK.

Є мложливість налаштувати інтерфейс входу, встановивши параметри FirebaseUI, або розгалужити код на GitHub, щоб ще більше налаштувати процес входу.

Можна імпортувати бібліотеку FirebaseUI, вказати методи входу, які потрібно підтримувати, і запустити процес входу FirebaseUI.

Для адреси електронної пошти, пароля чи номера телефону для входу та будь-яких федеративних постачальників ідентифікаційної інформації, які ви хочете підтримувати, увімкніть їх на консолі Firebase і виконайте будь-яку конфігурацію, яку вимагає постачальник ідентифікаційної інформації, як-от налаштування URL-адреси переспрямування OAuth.

Розробники Firebase Auth пропонують плавне оновлення до платформи Google Cloud Identity Platform для отримання додаткових платних переваг і

функцій, таких як багатофакторна автентифікація, функції блокування та корпоративні угоди про рівень обслуговування.

Cloud Firestore - хмарна база даних NoSQL для зберігання та синхронізації даних для клієнтської та серверної розробки. Це гнучка масштабована база даних для мобільних, веб-розробок і серверних розробок від Firebase і Google Cloud. Подібно до бази даних реального часу Firebase, вона використовує прослуховувачі в реальному часі для синхронізації даних між клієнтськими програмами і забезпечує автономну підтримку для мобільних пристроїв та Інтернету, тому вона швидко реагує і працює незалежно від затримки в мережі або підключення до Інтернету. Ви можете створити свою програму . Cloud Firestore також пропонує безшовну інтеграцію з іншими продуктами Firebase та Google Cloud, включаючи хмарні функції. [5]

Cloud Firestore чудово підходить для Apple, Android і веб-додатків. Такі додатки можуть отримати прямий доступ через власні SDK. Cloud Firestore також доступний з власними пакетами Python, Java, Node.js, Unity, C++ та Go SDK на додаток до RPC і REST API.

Також є можливість зберігати дані в документах з полями, зіставленими зі значеннями відповідно до моделі даних NoSQL Cloud Firestore. Ці документи зберігаються у колекціях. Колекції - це контейнери документів, які можна використовувати для організації даних та побудови запитів. Документи підтримують широкий спектр типів даних, від простих рядків та чисел до складних вкладених об'єктів. Ви також можете створювати підколекції у своїх документах для створення ієрархічних структур даних, які масштабуються в міру зростання вашої бази даних. Модель даних Cloud Firestore підтримує будь-яку структуру даних, яка підходить для будь-якої програми. Крім того, надсилання запитів у Cloud Firestore є експресивним, ефективним і гнучким. Є можливість писати неглибокі запити для отримання даних на рівні документа без вилучення цілих колекцій або вкладених підколекцій. Можна додати сортування, фільтрацію та обмеження до своїх запитів або курсорів, щоб розбивати результати на сторінки. Щоб підтримувати актуальність даних у

ваших програмах, без завантаження всієї бази даних щоразу, коли відбувається оновлення, потрібно додати прослуховувачі в реальному часі. Додавання прослуховувачів у реальному часі до додатка відправлятиме сповіщення за допомогою знімка даних щоразу, коли дані, які прослуховують клієнтські додатки, змінюватимуться, отримуючи лише нові зміни.

У Cloud Firestore є можливість захистити доступ до даних за допомогою автентифікації Firebase і правил безпеки Cloud Firestore для платформ, Apple (iOS, macOS), JavaScript (Web) і Android або керування доступом та ідентифікацією (IAM) на стороні сервера.

Хмарне сховище для Firebase (Cloud Storage) призначене для розробників додатків, яким необхідно зберігати та обслуговувати контент користувача, такий як фотографії та відео.

Хмарне сховище для Firebase - це потужна, проста та економічна служба зберігання об'єктів, створена командою Google враховуючи масштабні проекти. Пакети Firebase SDK для клауду додають захист Google для завантажень і скачувань файлів для програм Firebase, незалежно від якості мережі.

Розробники використовують Firebase SDK для хмарного сховища, щоб завантажувати на сервер та завантажувати на пристрій файли безпосередньо з клієнта. Якщо мережне з'єднання нестабільне, клієнт може відновити роботу з того місця, де він був зупинений, заощаджуючи час і пропускну здатність ваших користувачів.

Хмарне сховище для Firebase зберігає файли у Google Cloud Storage, що робить їх доступними через Firebase і Google Cloud. Це дає можливість завантажувати та скачувати файли з мобільних, десктом та веб клієнтів через Firebase SDK з хмарного сховища. Крім того, ви можете використовувати Google Cloud Storage API для обробки на стороні сервера, як-от фільтрація зображень і транскодування відео. Хмарне сховище масштабується автоматично, що означає, що немає необхідності перемикатися на іншого провайдера.

Cloud Firestore дозволяє використовувати запити для отримання окремих, конкретних документів або для отримання всіх документів у колекції, які відповідають параметрам запиту. Запити можуть мати кілька ланцюжкових фільтрів і поєднувати фільтрацію та сортування. Вони також індексуються за умовчанням, отже, продуктивність запиту пропорційна розміру набору результатів, а чи не набору даних..

Як і база даних реального часу, Cloud Firestore використовує синхронізацію даних для оновлення даних на підключених пристроях. Однак він також створений для ефективного виконання простих одноразових запитів.

Cloud Firestore кешує дані, які активно використовує ваша програма, тому програми можуть записувати, читати, прослуховувати та запитувати дані, навіть якщо пристрій знаходиться в автономному режимі. Коли пристрій знову підключається до Інтернету, то Cloud Firestore запускає синхронізацію усіх локальних зміни з змінами які вже є на Cloud Firestore.

Пакети Firebase SDK для хмарного сховища бездоганно відпрацьовує інтеграція з Firebase Authentication для ідентифікації користувачів, і надається декларативна мову безпеки, що дозволяє встановити контроль доступу для окремих файлів або груп файлів, тож можна зробити файли загальнодоступними чи приватними.

Cloud Firestore надає найкраще з потужної інфраструктури Google Cloud: підтримка автоматичної реплікації даних у кількох регіонах, надійні гарантії узгодженості, атомарні пакетні операції та реальні транзакції. Google розробили Cloud Firestore, щоб справлятися з найскладнішими навантаженнями на базу даних від найбільших програм світу.

Cloud Firestore - це гнучка масштабована база даних для мобільних, веб-розробок і серверних розробок від Firebase і Google Cloud.

База даних Firebase Realtime зберігає дані програми JSON, як-от стан гри чи повідомлення чату, і миттєво синхронізує зміни на всіх підключених пристроях. Щоб дізнатися більше про відмінності між параметрами бази даних, перегляньте розділ Виберіть базу даних: Cloud Firestore або Realtime Database.

Firebase Remote Config зберігає вказані розробником пари ключ-значення, щоб змінити поведінку та зовнішній вигляд вашої програми, не вимагаючи від користувачів завантажувати оновлення.

Firebase Hosting надає швидкий і безпечний хостинг для вашої веб-програми, статичного та динамічного вмісту та мікросервісів, цей інструмент розміщує HTML, CSS і JavaScript для веб-сайту, а також інші ресурси, надані розробниками, наприклад графіку, шрифти та значки. [5]

Firebase Hosting - це веб-хостинг продуктового рівня для веб-розробників. Технологія швидко розгортає веб-програми за допомогою однієї команди та обслуговується як статичний, так і динамічний контент у глобальній CDN (мережі доставки вмісту). Також можна поєднати Firebase Hosting із Cloud Functions або Cloud Run, щоб створити та розмістити мікросервіси на Firebase.

Firebase Hosting створено для сучасного веб-розробника. Веб-сайти та програми стали потужнішими, ніж будь-коли, завдяки появі зовнішніх фреймворків JavaScript, як Angular, React та Flutter і інструментів статичного генератора, як Jekyll. Незалежно від того, чи розгортається проста цільова сторінка програми або складний прогресивний веб-додаток (PWA), хостинг пропонує інфраструктуру, інструменти і функції, спеціально розроблені для розгортання та керування веб-сайтами та додатками.

Використовуючи Firebase CLI, на серверах хостингу розгортаються файли з локальних каталогів. Окрім розміщення статичного вмісту, можна використовувати Cloud Functions for Firebase або Cloud Run для надання динамічного вмісту та розміщення мікросервісів на сайтах. Увесь вміст подається через з'єднання SSL із найближчого периферійного сервера на глобальному CDN.

Також можна переглянути та протестувати зміни перед опублікуванням. Використовуючи Firebase Local Emulator Suite, можна емулювати написану програму та серверні ресурси за локальною URL-адресою. Також можна поділитися змінами за тимчасовою URL-адресою попереднього перегляду та налаштувати інтеграцію GitHub для легкої ітерації під час розробки.

Сучасний web безпечний. SSL без конфігурації вбудовано в Firebase Hosting, тому вміст завжди доставляється безпечно.

Кожен файл, який завантажується, кешується на твердотільних накопичувачах на периферії CDN по всьому світу та подається як gzip або Brotli. Firebase Hosting автоматично вибирає найкращий метод стиснення для вмісту додатку. Де б не були користувачі додатку, вміст доставляється швидко.

Використовуючи Firebase CLI, можна запустити програму за лічені секунди. Інструменти командного рядка спрощують розгортання в процес збирання, а якщо потрібно скасувати розгортання, Firebase Hosting забезпечує відкат одним клацанням миші.

Firebase Hosting має легкі варіанти конфігурації хостингу, за допомогою яких можна створювати складні PWA. Можна легко переписати URL-адреси для маршрутизації на стороні клієнта, налаштувати власні заголовки та навіть надавати локалізований вміст.

Для обслуговування вмісту Firebase пропонує кілька варіантів доменів і субдоменів:

- За умовчанням кожен проект Firebase має безкоштовні субдомени в доменах web.app і firebaseapp.com. Ці два сайти обслуговують той самий розгорнутий вміст і конфігурацію.
- Можна створити кілька сайтів, якщо у є пов’язані сайти та додатки, які обслуговують різний вміст, але все ще спільно використовують ті самі ресурси проекту Firebase (наприклад, якщо у є блог, панель адміністратора та загальнодоступна програма).
- Можна підключити власне доменне ім’я до сайту, розміщеного на Firebase.
- Firebase автоматично надає сертифікати SSL для всіх доменів, щоб увесь вміст обслуговувався безпечно.

Пов’язавши сайт із веб-додатком Firebase, зможна використовувати Google Analytics для збору даних про використання та поведінку додатка та

використовувати моніторинг продуктивності Firebase, щоб отримати уявлення про характеристики продуктивності додатка.

Firebase Cloud Messaging (FCM) - це міжплатформне рішення для обміну повідомленнями, яке дозволяє надійно надсилати повідомлення безкоштовно.

FCM можна використовувати для сповіщення клієнтських програм про те, що нова електронна пошта або інші дані доступні для синхронізації. Можна надсилати сповіщення, щоб стимулювати повторне залучення та утримання користувачів. Для таких випадків використання, як обмін миттєвими повідомленнями, може передавати корисне навантаження до 4000 байт до клієнтської програми. [5]

Реалізація FCM включає в себе два основні компоненти для відправлення та отримання:

- Надійне середовище, як Cloud Functions для Firebase або сервер додатку, на якому можна створювати, налаштовувати та надсилати повідомлення.
- Клієнтська програма Apple, Android або веб-програма (JavaScript), яка отримує повідомлення через відповідну транспортну службу для певної платформи.

Є можливість надсилати повідомлення через Firebase Admin SDK або протоколи сервера FCM. Можна використовувати редактор сповіщень для тестування та надсилання маркетингових повідомлень або повідомлень про взаємодію за допомогою потужних вбудованих засобів націлювання та аналітики або спеціальних імпортованих сегментів.

Ключові можливості Firebase Cloud Messaging (FCM):

- Надсилає сповіщення, які відображаються користувачеві або надсилає повідомлення з даними які визначають, що відбувається в коді програми.
- Розповсюджує повідомлення в клієнтську програму будь-яким із 3 способів: на окремі пристрої, на групи пристроїв або на пристрої, підписані на теми.
- Надсилайте підтвердження, чати та інші повідомлення з пристроїв назад на сервер через надійний і енергозберігаючий канал підключення FCM.

Firebase Google Analytics - безкоштовне рішення для аналізування додатків, що надає статистичну інформацію про використання додатка та залучення користувачів. [5]

Firebase Google Analytics це необмежене аналітичне рішення, доступне безкоштовно. Analytics інтегрує всі функції Firebase і надає необмежену кількість звітів для до 500 окремих подій, що можна визначати за допомогою Firebase SDK. Google Analytics reports допомагають чітко зрозуміти, як поводяться користувачі додатку, що дає змогу приймати обґрунтовані рішення щодо маркетингу додатків і оптимізації ефективності.

Google Analytics допоможе зрозуміти, як люди використовують веб-програму, десктоп додаток чи програму Apple або Android. Пакет SDK автоматично фіксує низку подій і властивостей користувача, а також дозволяє визначати власні події, що настроюються для вимірювання речей, характерних для бізнес-логіки застосунку. Як тільки дані будуть зібрані, вони будуть доступні на вашій інформаційній панелі через консоль Firebase. Ця інформаційна панель надає докладну інформацію про зібрані дані, від агрегованих даних, таких як активні користувачі та демографічні дані, до більш докладних даних, таких як ідентифікатори продуктів, які найчастіше переглядаються або купуються.

Analytics також інтегрується з низкою інших функцій Firebase. Наприклад, analytics автоматично реєструє події, які відповідають сповіщенням, надісланим через редактор сповіщень, і надає відповідний звіт.

Analytics допомагає зрозуміти, як поводяться користувачі додатку, щоб можна було приймати зважені рішення про те, як рекламувати та розвивати свою програму. Переглядати ефективність збору інформації можна у звичайних і платних каналах, які допомагають зрозуміти, які методи найефективніші для залучення цінних користувачів. Якщо потрібно виконати спеціальний аналіз або об’єднати дані з іншими джерелами, можена пов’язати дані Analytics із BigQuery, що дає змогу виконувати складніший аналіз, наприклад надсилати запити до великих наборів даних і об’єднувати кілька джерел даних.

Використовуючи визначення Analytics, можна змінювати поведінку та зовнішній вигляд програми для різних аудиторій, не розповсюджуючи кілька версій програми а інтеграція менеджера тегів Google із Google Analytics дає змогу дистанційно керувати впровадженням Analytics із веб-інтерфейсу після розповсюдження програми.

Почати роботу з Analytics легко. Потрібно додати пакет SDK Firebase до нового чи існуючого додатка, і збір даних почнеться автоматично. Можна переглянути дані аналітики на консолі Firebase протягом кількох годин. Можна використовувати Analytics для реєстрації спеціальних подій, які мають значення для програми, як покупки в електронній комерції або досягнення а також можна визначити важливі аудиторії в консолі Firebase, які можна використовувати для відправки спеціальних на повідомлень, рекламних акції чи нових функцій програми за допомогою інших функцій Firebase, як FCM або Remote Config.

Firebase Crashlytics - легкі звіти про збої в режимі реального часу, які допомагають відстежувати, розставляти пріоритети та усувати проблеми зі стабільністю, які погіршують якість вашого додатка. Crashlytics економить час на усунення несправностей, інтелектуально групуючи збої та висвітлюючи обставини, які призвели до них та встановлює, які рядки коду викликають збою.

Firebase Crashlytics дозволяє отримати чітке та практичне уявлення про проблеми з програмою. Це дуже хороше рішення рішення для звітів про збої для Apple, Android, Flutter і Unity. [5]

Цей інструмант має дуже корисну можливість, а саме є можливість встановити, чи певний збій впливає на багатьох користувачів та надсилає сповіщення, коли проблема раптово стає серйознішою та яка може потребувати негайної уваги.

Crashlytics синтезує збої у керований список проблем, надає контекстну інформацію та підкреслює серйозність і поширеність збоїв, щоб розробники могли швидше визначити першопричину збоїв.

Crashlytics пропонує Crash Insights, корисні поради, які висвітлюють поширені проблеми стабільності та надають ресурси, які полегшують їх усунення, сортування та вирішення.

Crashlytics може фіксувати помилки додатка як події app_exception в Analytics. Події спрощують налагодження, надаючи доступ до списку інших подій, що призводять до кожного збою, і надають статистику аудиторії, дозволяючи отримувати звіти Analytics для користувачів із збоями.

FlutterFire CLI - це CLI, який допоможе використовувати FlutterFire у програмах Flutter, цей інструмент надає команди, які полегшують процес встановлення FlutterFire на всіх підтримуваних платформах. [5]

3.5. Огляд IDE Android Studio

Android Studio IDE - інтегроване середовище розробки (IDE) для роботи з платформою Android, анонсоване на конференції Google I/O 16 травня 2013 року.[4]

IDE знаходиться у відкритому доступі з версії 0.1, випущеної у травні 2013 року, і перебуває на стадії бета-тестування з версії 0.8, випущеної у червні 2014 року. Першу стабільну версію 1.0 було випущено в грудні 2014 року, після чого підтримка плагіна Android Development Tools (ADT) для Eclipse була припинена.

Заснована на програмному забезпеченні IntelliJ IDEA від JetBrains, Android Studio є офіційним інструментом для розробки програм для Android. Це середовище розробки доступне для macOS, Windows та Linux. На щорічній конференції Google I/O 17 травня 2017 року Google оголосила, що окрім Java та C++ буде підтримувати мову Kotlin, яку використовує Android Studio як офіційну мову програмування для платформи Android.

Нові функції з'являються з кожною новою версією Android Studio. На даний момент доступні такі функції:

- Розширений редактор макетів: WYSIWYG, можливість маніпулювати компонентами інтерфейсу користувача за допомогою перетягування,

можливість попереднього перегляду макета в конфігураціях з кількома екранами.

- Збірка додатків, базується на Gradle.
- Різні види зборок і генерація декількох типів .apk або .aab файлів
- Рефакторинг та аналіз поточного коду
- Аналізатор коду Lints, який має можливість знаходити проблеми продуктивності додатку, несумісності поточних версій та інші проблеми.
- Вбудований Pro Guard та Android утиліта для підписування застосунків.
- Шаблони базових компонентів та макетів під Android.
- Можливість розробки застосунків для Android TV та Android Wear.
- Підтримка Google Cloud Platform, що включає в себе інтеграцію з сервісами App Engine та Google Cloud Messaging.
- Android Studio 2.1 підтримує Android N Preview SDK, це означає, що розробники можуть розпочати писати програми для нових програмних платформ.
- Нова версія Android Studio 2.1 працює з оновленим компілятором Jack, покращеною підтримкою Java 8 та покращеною функціональністю Instant Run.
- З Platform-tools версії 23.1.0 для Linux є виключно 64-розрядна.
- AndroidStudio 3.0 стандартно поставляється з мовними інструментами Kotlin і ґрунтується на інтегрованому середовищі розробки JetBrains.

3.6. Огляд Git та GitHub

Git - це система керування версіями яка розповсюджується на безоплатній основі з відкритим вихідним кодом, призначена для швидкої та ефективної обробки будь-якого малого або великого проекту.

Git простий у освоєнні, має невеликий розмір та блискавичну продуктивність. Такі функції, як оптимізовані локальні гілки, зручні проміжні області та кілька робочих процесів, перевершують інструменти SCM, такі як Subversion, CVS, Perforce та Clear Case. [7]

GitHub - найбільший веб-сервіс для розміщення IT-проектів та їхньої спільної розробки. Цей веб-сервіс заснований на системі контролю версій Git та

розроблений за допомогою Rubyon Rails та Erlang компанією GitHub, Inc (раніше Logical Awesome). Сервіс безкоштовний для проектів з відкритим вихідним кодом та (станом на 2019 рік) невеликих приватних проектів та пропонує повну функціональність (включаючи SSL). Найбільші корпоративні проекти пропонують різноманітні платні плани. Автори цього сайту називають GitHub «соціальною мережею для розробників». Окрім публікації коду, учасники можуть спілкуватися, коментувати зміни один одного та стежити за новинами своїх друзів. Широкі можливості Git дозволяють програмістам поєднувати репозиторії. GitHub надає зручний інтерфейс, що показує внесок кожного учасника у форматі дерева. Є персональна сторінка проекту, невелика вікі та система відстеження помилок. Ви можете переглядати файли свого проекту з підсвічуванням синтаксису для більшості мов програмування прямо на своєму сайті. Ви можете створювати приватні репозиторії, які можете бачити лише ви та вибрані вами люди. Раніше за можливість створення приватних репозиторіїв стягувалась плата. Нові файли можна додавати прямо до репозиторію через веб-інтерфейс сервісу. Код проекту можна скопіювати через Git, а також завантажити у вигляді звичайного архіву із сайту. Сервіс підтримує отримання та редагування коду через SVN та Mercurial, окрім Git. На сайті є сервіс pastebin gist.github.com для швидкого опублікування фрагментів коду. [29]

Висновок до розділу 3

Правильний вибір необхідних інструментів розробки та розуміння принципів їхньої роботи - ключові моменти, які сильно вплинуть на створення та якість майбутніх проектів.

РОЗДІЛ 4
РОЗРОБКА МУЛЬТИПЛАТФОРМЕННОГО ДОДАТКУ «Musicians Shop»

4.1. Створення архітектури та UI проєкту

Добре відпрацьована архітектура потрібна всім додаткам, і складним, і шаблонним. З її допомогою заощаджується час, зусилля і гроші. Архітектура мобільних додатків - сукупність рішень, як організувати програму. У неї входять: структурні елементи і інтерфейси, зв'язок між обраними елементами, загальний стиль програми.

Гарна архітектура означає вигоду: простота і ефективність. Програму з такою архітектурою легше змінювати, тестувати і налагоджувати. Як зрозуміти, чи гарна архітектура у додатка?

Гнучкість: вибране рішення легко змінювати. Можна змінити один елемент, і це не стане фатальним для інших складових.

Масштабованість: час на розробку і доповнення зменшується. Гарна архітектура дозволяє направити розробку в декілька паралельних потоків.

Тестованість: додаток легко тестується, а значить, зменшується кількість помилок і збільшується його надійність.

Повторне використання: Елементи та структурні конструкції можуть бути використані в інших проектах.

Зрозумілість: код має бути зрозумілий якомога більшій кількості людей. Над додатком працює багато людей. Гарна архітектура дозволяє новачкам швидко розібратися в проекті.

Архітектура додатків - це структурний принцип, яким створено додаток. Кожному архітектурному вигляду властиві свої характеристики, властивості та відносини між компонентами. Компонент - дрібна або велика логічна та незалежна частина архітектурної системи програми.

Кожна програма навколо нас побудована по будь-якій архітектурі. Навіть у тих випадках, де принципово не дотримувалися жодного її видів, все одно

додаток буде працювати в рамках одного з видів архітектури. Інакше не виходить.

Багато хто поєднує терміни «архітектура» та «структура» в один. Але це не те саме, хоча структура додатку безпосередньо залежить від архітектури. Коли ми вимовляємо «архітектура додатків», то йдеться у широкому значенні про компоненти майбутньої системи додатку. Однак кожен архітектурний компонент може бути організований та представлений різними рішеннями. Ці рішення утворюють структуру докладання. Тобто структура більш точно описує інструменти та модулі майбутньої програми.

Схема як у проекті використовується GetX для зв'язку між рівнями:

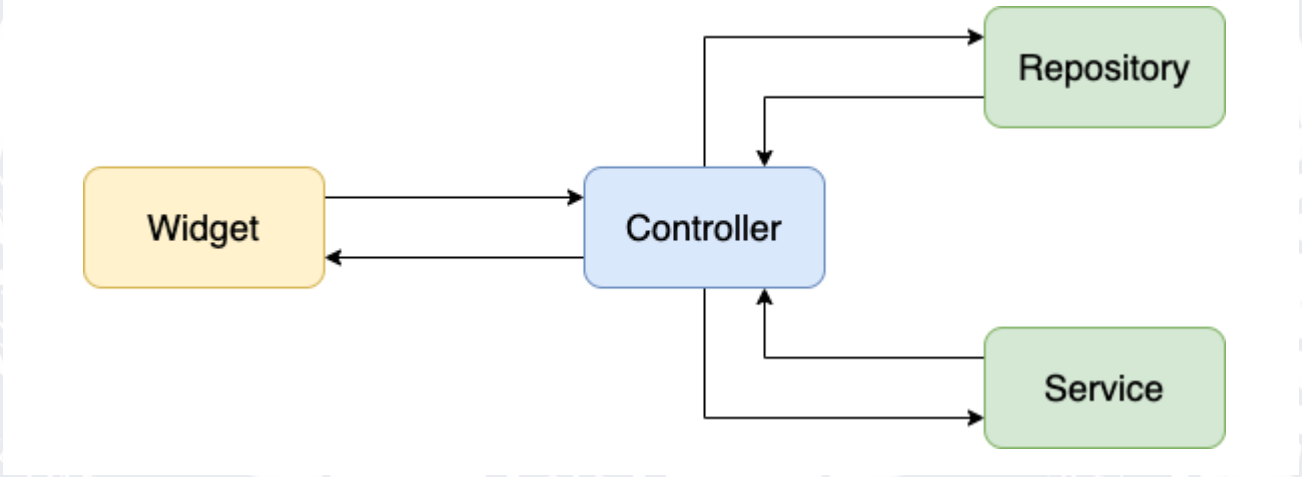

Рисунок 4.1 - Схема для зв'язку між рівнями

Структура додатку «Musicians Shop» має 4 рівні:

- data - рівень даних проекту, включає каталоги local, remote і cache.
- domain - містить абстракцію та бізнес-логіку проекту, включає models, requests, responses тощо.
- presentation - містить увесь інтерфейс користувача та підкаталог для кожного екрана.
- shared - містить утиліти або загальні функції та стилі програм.

main.dart - це початкова точка програми. У цьому файлі визначено всі конфігурації на рівні програми.

firebase_options.dart - у цьому файлі визначено всі конфігурації для Firebase.

Виконання програми Flutter розпочинається з основної функції (main). Для створення графічного інтерфейсу ця функція викликає іншу вбудовану функцію runApp (Widget App).Вона прикріплює певний віджет до екрану. По суті, це те, що ви бачите при запуску програми на екрані пристрою. Віджет, що прикріплюється передається в функцію runApp() в якості параметра. Коли користувач дивиться на свій пристрій або, точніше, на додаток, запущений на пристрої, то користувач бачить тільки екран. Насправді, все, що користувач бачить - це пікселі, які разом складають 2-мірне зображення, і коли ви торкаєтеся екрану пальцем, пристрій розпізнає тільки положення вашого пальця на склі.

Вся краса додатка написаного на Flutter (з візуальної точки зору) в більшості випадків полягає в оновленні цього зображення на основі наступних взаємодій:

- з екраном пристрою (наприклад, палець на склі)
- з мережею (наприклад, зв'язок з сервером)
- з переходами (наприклад, hero анімація)
- з іншими зовнішніми датчиками

Візуалізація зображення на екрані забезпечується апаратним забезпеченням (дисплеєм), яке регулярно (зазвичай 60 раз в секунду) оновлює дисплей. Ця називається "частотою оновлення" і виражається в Гц (Герцах). Дисплей отримує інформацію для відображення від GPU (Graphics Processing Unit), що представляє собою спеціалізовану електронну схему, оптимізовану і призначену для швидкого формування зображення з деяких даних (полігонів і текстур). Кількість разів в секунду, яке графічний процесор може генерувати "зображення" (= буфер кадрів) для відображення і відправки його на апаратне забезпечення, називається кадровою частотою (прим: framerate). Це вимірюється за допомогою блоку кадрів в секунду (наприклад, 60 кадрів в секунду або 60 fps).

Одна з головних цілей програми Flutter - це створити 2-мірне зображення і дати можливість взаємодіяти з ним. Також у Flutter, майже все обумовлено необхідністю оновлення екрану швидко і в потрібний момент.

Коли пишуть додаток Flutter, використовуючи Dart, то розробник залишається на рівні Flutter Framework. Flutter Framework взаємодіє з Flutter Engine через шар абстракції, званий Window. Цей рівень абстракції надає ряд API для непрямої взаємодії з пристроєм. Також через цей рівень абстракції Flutter Engine повідомляє Flutter Framework, коли:

- подія, що представляє інтерес, відбувається на рівні пристрою (зміна орієнтації, зміна налаштувань, проблема з пам'яттю, стан роботи програми);
- якась подія відбувається на рівні скла (жест);
- канал платформи відправляє деякі дані;
- коли Flutter Engine готовий до рендерингу нового кадру.

Коли ви розробляєте за допомогою Flutter, ви визначаєте структуру свого екрану (екранів), використовуючи віджети, які разом утворюють ієрархічну структуру. Ця структура виглядає, як дерево де перший віджет є його коренем. Також віджет сам по собі може бути агрегацією інших віджетів.

Формулювання «дерево віджетів» (Widget tree) існує тільки для полегшення розуміння, оскільки програмісти використовують віджети, але у Flutter немає дерева віджетів, правильніше буде сказати «дерево елементів» (tree of elements). Кожному віджету відповідає один елемент. Елементи пов'язані один з одним і утворюють дерево. Отже елемент є посиланням на щось в дереві.

Підсумки:

- У Flutter немає дерева віджетів, але є дерево елементів;
- Елементи створюються віджетами;
- Елемент посилається на віжет, який його створив;
- Елементи пов'язані разом з батьківськими співвідношеннями;
- У елемента може бути child або children;

- Елементи також можуть вказувати на Render Object.
- Елементи визначають, як відображаються частини блоків пов'язані один з одним

4.2. Створення логіки додатку та взаємодія з Firebase

Більшість пакетів SDK Firebase розроблені як бібліотеки з відкритим вихідним кодом в загальнодоступному репозиторії Firebase на GitHub. Firebase активно працює над тим, щоб найближчим часом перемістити бібліотеки Firebase, розроблені в приватному порядку, в загальнодоступний GitHub.

Firebase підтримує колекцію прикладів швидкого запуску для більшості API Firebase на Android. Ці короткі керівництва можна знайти в загальнодоступному репозиторії швидкого запуску Firebase на GitHub. [51]

Можна відкрити кожну інструкцію, як проект Android Studio, а потім запустити їх на мобільному пристрої або віртуальному пристрої (AVD). Або використовувати ці короткі керівництва як приклад коду для використання Firebase SDK. [47]

Деякі SDK Firebase для Android залежать від сервісів Google Play, що означає, що вони будуть працювати тільки на пристроях і емуляторах з встановленими сервісами Google Play. Ці SDK Firebase взаємодіють з фоновою службою сервісів Google Play на пристрої, щоб надати додатку безпечний, актуальний і легкий API. На деяких пристроях Android, таких як пристрої Amazon Kindle Fire або пристрої, що продаються в деяких регіонах, не встановлені сервіси Google Play. [48]

SDK Firebase можна розділити на три категорії:

1. Для цих SDK потрібні сервіси Google Play, в іншому випадку вони не працюють.
2. Рекомендуються сервіси Play - ці SDK вимагають, щоб сервіси GooglePlay мали повну функціональність, але вони як і раніше пропонують більшу частину функцій навіть без сервісів Google Play.
3. Сервіси Play не потрібні - ці SDKS не вимагають, щоб сервіси GooglePlay мали повну функціональність.

API REST управління Firebase дозволяє програмно знаходити і керувати проектами Firebase, включаючи ресурси Firebase та додаткові Firebase.

Загальний робочий процес додавання ресурсів та додатків Firebase у існуючому проекті Google Cloud, який на даний момент не використовує службу Firebase:

Основні кроки:

1. Додати сервіси Firebase в проект.
2. Додати додатки Firebase в проект Firebase.
3. Зв'язати проект Firebase з аккаунтам Google Analytics.
4. Завершити налаштування місцеположення проекту за замовчуванням.

Перш ніж виконувати будь-які дії, потрібно переконатися, що ввімкнені API.

Спочатку необхідно включити Management API для вашого проекту Google Cloud і згенерувати токен доступу.

Необхідно Увімкнути Management REST API для свого Google Cloud.

Якщо це не зроблено, необхідно включити Firebase Management API для використання з проектом Google Cloud:

1. Відкрити сторінку Firebase Management API в консолі Google API.
2. У відповідь на запит обрати проект Google Cloud.
3. Натиснути увімкнути на сторінці Firebase Management API.
4. Натиснути увімкнути на сторінці Firebase Management API.
5. Потім використовувати Firebase Admin SDK, щоб отримати токен доступу з облікових даних службового облікового запису.

Можна знайти проекти Google Cloud, які доступні для публікації сервісів Firebase. [52]

Тіло відповіді від запиту availableProjects.list містить список об'єктів Project Info. Якщо список проектів занадто довгий, тіло відповіді також містить nextPageToken, який можна використовувати в якості параметра запиту для отримання наступної сторінки проектів.

Також є можливість використовувати будь-яке значення project вказане у відповіді з availableProjects.list щоб додати служби Firebase або додатки в свій проект.

Проекти Google Cloud можуть скористатися послугами, пропонованими Firebase. Також можна додати служби Firebase в існуючий проект Google Cloud в консолі Firebase.

Результатом виклику projects.addFirebase є Operation. Перш ніж буде можливість викликати інші кінцеві точки, пов'язані з Firebase, для проекту, операція повинна бути успішною.

Щоб перевірити, чи успішна операція, можна викликати operations.get для операції до тих пір, поки значення done стане true а його response буде FirebaseProject тип FirebaseProject. У разі збою операції встановлюється errorgoogle.rpc.Status.

Оскільки done, true і тип response - FirebaseProject, проект Google Cloud тепер має служби Firebase. Відповідь також містить іншу корисну інформацію по вашому недавно створеному FirebaseProject, наприклад, про його projectNumber і його resources замовчуванням. Operation автоматично видаляється після завершення.

Firebase Project може використовувати безліч різних додатків, включаючи iOS, Android і веб-додатки. Потрібно звернути увагу, що також можна додати додатки Firebase в існуючий проект Firebase в консолі Firebase.

Також є можливість програмно зв'язати існуючий обліковий запис Google Analytics з існуючим Firebase Project. Потрібно звернути увагу, що також можна пов'язати існуючий проект Firebase з Google Analytics на вкладці «Інтеграції» в налаштуваннях проекту.

Для виклику projects.addGoogleAnalytics потрібно analytics_resource, який може бути або analyticsAccountId або analyticsPropertyId:

1. Треба вказати існуючий analyticsAccountId щоб підготувати новий ресурс GoogleAnalytics що зазначений в обліковому записі і зв'язати нову властивість з проектом Firebase.

2. Також треба вказати існуючий analyticsPropertyId щоб зв'язати ресурс GoogleAnalytics з вашим проектом Firebase.

Є можливість знайти analyticsAccountId і будь-який існуючий analyticsPropertyId на веб-сайті GoogleAnalytics.

Коли виконується запит projects.addGoogleAnalytics:

1. Перша перевірка визначає, чи відповідають існуючі потоки даних у властивості Google Analytics будь-яким існуючим програмам Firebase в Firebase Project (на основі packageName або bundleId пов'язаних з потоком даних). Потім, у разі необхідності, зв'язуються потоки даних і додатки. Ця автоматична прив'язка застосовується тільки до додатків Android і iOS.
2. Якщо відповідні потоки даних для ваших додатків Firebase не знайдені, нові потоки даних надаються у властивості Google Analytics для кожного з ваших додатків Firebase. Новий потік даних завжди готується для веб-додатку, навіть якщо він раніше був пов’язаний з потоком даних в вашому ресурсі Analytics.

Потрібно виконати запит projects.addGoogleAnalytics.

У тілі запиту project.addGoogleAnalytics вказуємо обліковий запис Google Analytics analytics Account Id. Цей виклик надасть нову властивість Google Analytics і зв’яже нову властивість з FirebaseProject.

Результатом виклику projects.addGoogleAnalytics є Operation. Перш ніж буде можливість викликати інші кінцеві точки, пов'язані з Firebase, для проекту, операція повинна бути успішною.

Щоб перевірити, чи успішна операція, можна викликати operations.get для операції, поки значення done стане true a response матиме тип analyticsDetails. У разі збою операції встановлюється errorgoogle.rpc.Status.

Оскільки done має значення true і тип response - analyticsDetails, FirebaseProject тепер пов'язаний з вказаною обліковим записом GoogleAnalytics. Operation автоматично видаляється після завершення.

Завершення налаштування місцеположення проекту за замовчуванням необов'язкове.

Якщо проект Firebase буде використовувати Cloud Firestore, Cloud Storage або додаток AppEngine, можна програмно вказати місце розташування ресурсу Google Cloud Platform (GCP) за замовчуванням для проекту. Також можна ввести місце в консолі Firebase.

Перед налаштуванням цього місця розташування треба ознайомитися з розділом Вибір місць розташування для вашого проекту, щоб дізнатися, яке місце розташування найкраще підходить для проекту. Також потрібно викликати projects.availableLocations щоб отримати список допустимих місць розташування для проекту, тому що, якщо проект є частиною організації Google Cloud, то політика організації може обмежувати допустимі розташування для проекту.

Виклик цього методу defaultLocation.finalize створює додаток AppEngine з defaultLocation.finalize хмарного сховища за замовчуванням, розташований в locationId який вказується в тілі запиту.

Розташування ресурсу GCP за замовчуванням могло бути вже зазначено, якщо в Project вже є додаток AppEngine або цей метод defaultLocation.finalize був раніше викликаний.

Потрібно projects.defaultLocation.finalize де locationId: місце, де зберігаються дані для сервісів GCP, що вимагають налаштування місцеположення, наприклад Cloud Firestore або Cloud Storage.

Результатом виклику projects.defaultLocation.finalize є Operation. Перш ніж буде можливість викликати інші кінцеві точки, пов'язані з Firebase, для вашого проекту, операція повинна бути успішною.

Щоб перевірити, чи успішна операція, можна викликати operations.get для операції, поки значення done стане true а його response буде типу google.protobuf.Empty. Якщо операція завершилася невдало, повідомлення про error тілі відповіді матиме тип google.rpc.Status. Operation автоматично видаляється після завершення.

Наслідуючи моделі даних NoSQL Cloud Firestore, дані зберігаються в документах з полями, зіставленими зі значеннями. Ці документи зберігаються у

колекціях. Колекції – це контейнери документів, які можна використовувати для організації даних та створення запитів. Документи підтримують широкий спектр типів даних, від простих рядків та чисел до складних вкладених об'єктів. Ви також можете створювати у своїх документах вкладені колекції для побудови ієрархічних структур даних, які масштабуються зі зростанням вашої бази даних. Модель даних Cloud Firestore підтримує будь-яку структуру даних, яка найкраще підходить для вашої програми.

Крім того, запити в Cloud Firestore є виразними, ефективними і гнучкими. Можна створювати неглибокі запити для отримання даних на рівні документа без необхідності отримувати всю колекцію або будь-які вкладені колекції. Можна додати сортування, фільтрацію і обмеження до своїх запитів або курсором, щоб розбивати результати на сторінці. Щоб дані у запитах додатка залишалися актуальними, не витягуючи всю базу даних при кожному оновленні, треба додати Listeners в реальному часі. Додавання Listeners в реальному часі в додаток дає можливість за допомогою моментального знімка даних щоразу повідомляти, коли дані змінюються, з моментальними змінами у клієнтських програмах, що прослуховують ці зміни, отримуючи тільки нові зміни.

Обов’язково треба захистити доступ до даних в Cloud Firestore використовуючи Firebase Authentication та правила безпеки Cloud Firestore для Android, JavaScript (Web) та iOS або Identity and Access Management (IAM) для серверних мов.

Шлях реалізації:

1. Потрібно інтегрувати SDK Cloud Firestore, та підключити клієнтів через Gradle, Cocoa Pods або сценарій include.
2. Потрібно захистити дані. Треба використовувати правила безпеки Cloud Firestore або управління ідентифікацією та доступом (IAM), щоб захистити дані для мобільних / веб-додатків і розробки серверів відповідно.
3. Додати дані. Потрібно створити документи і колекції в своїй базі даних.

4. Отримати дані. Створюйте запити або використовуйте Listeners в реальному часі для отримання даних з бази даних.

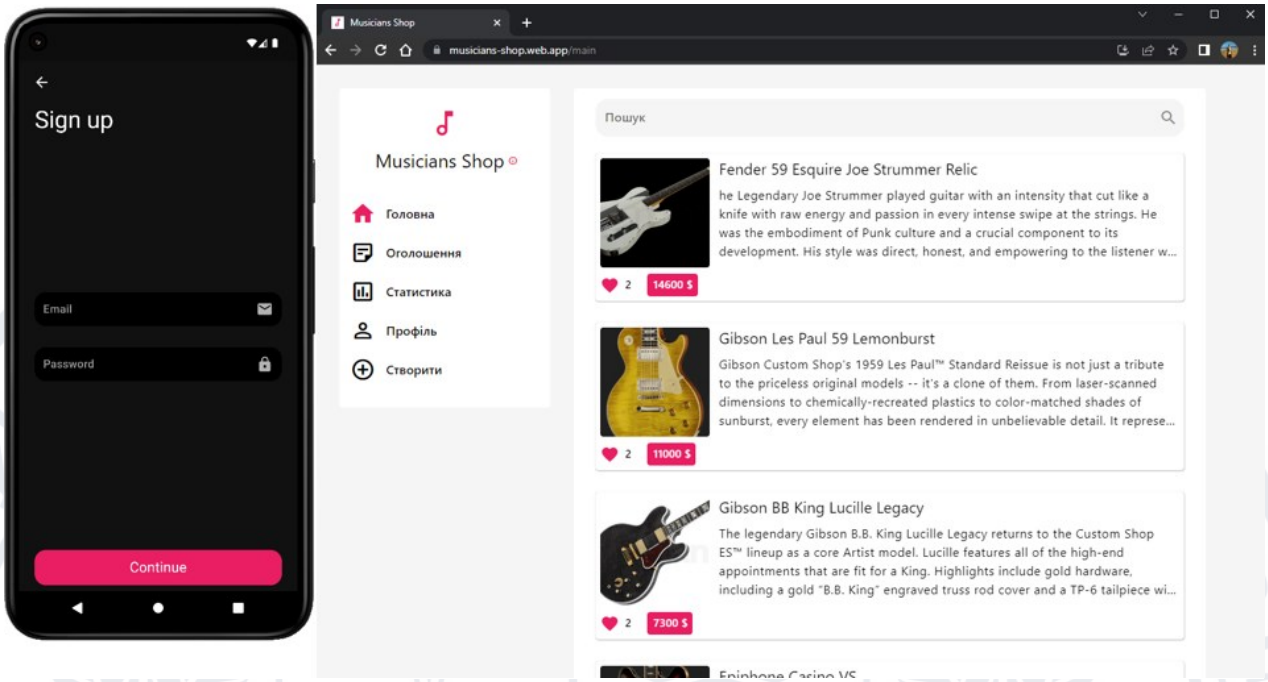

Рисунок 4.2 - скріншот додатку під Android (зліва) та Web (справа)

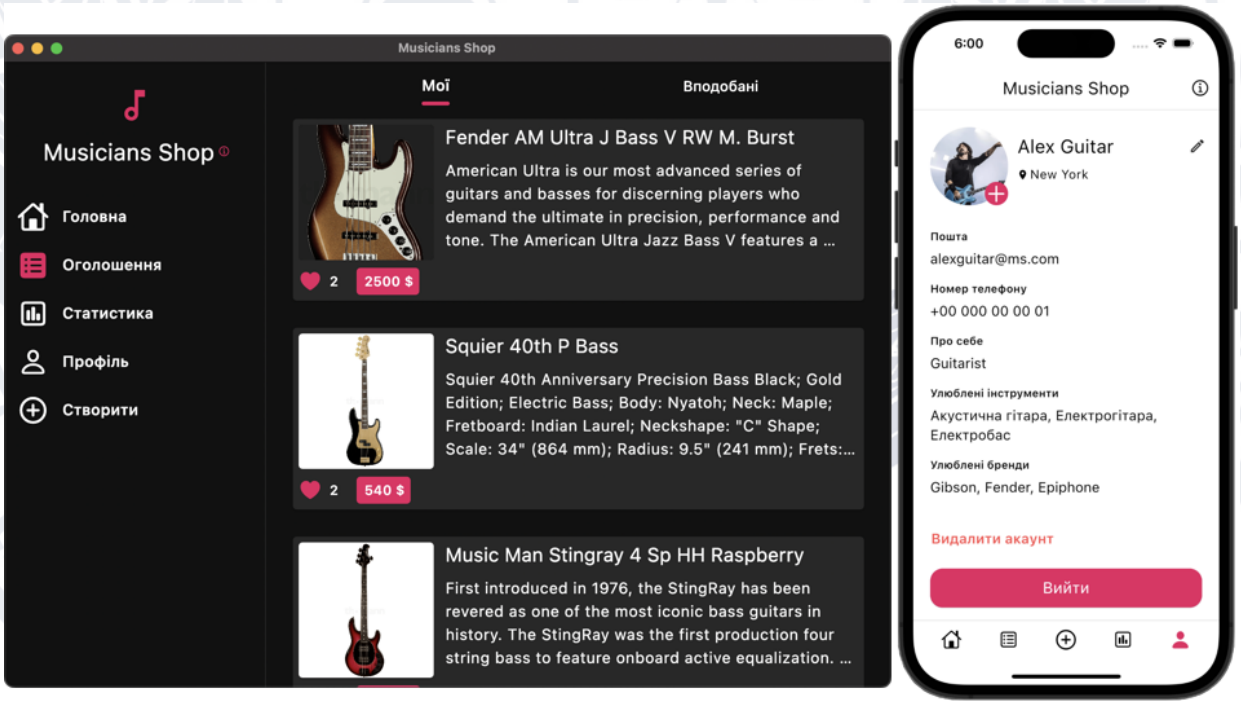

Рисунок 4.3 - скріншот додатку під macOS (зліва) та iOS (справа)

Після створення бази даних в Cloud Firestore треба створити базу даних Cloud Firestore, якщо це не було зроблено, далі треба створити проект Firebase: в консолі Firebase треба натиснути «Додати проект», а потім дотримуючись вказівок на екрані створити проект Firebase або додати служби Firebase в

існуючий проект GCP. Далі потрібно перейти в розділ Cloud Firestore в консолі Firebase. Там буде запропоновано вибрати існуючий проект Firebase. Потрібно дотримуватися робочого процесу створення бази даних. [57]

4.3. Робота з Git та GitHub

Під час розробки мультиплатформного додатку «Musicians Shop» усі зміни що вносяться в проект комітяться та пушаться в репозиторій на GitHub. Система контролю версій досить складна. Якщо один розробник працює над своїм проектом, то навіть не обов'язково реєструватися на сервері, можна просто встановити Git і працювати з системою контролю версій локально. Для локальної роботи над проектом не обов'язково навіть буде підключення до інтернету.

Репозиторій який створюється локально - це фактично такий же репозиторій як на сервері, тільки без можливості працювати над проектом розподіленої команди розробників. Система контролю версій - це, по суті, реалізація права програміста на помилку в своєму коді. «Людині властиво помилятися», помиляються навіть кращі програмісти і щоб повернутися до свого коду та виправити помилку, додати функціональність й існує така чудова система, як система контролю версій. Крім того використання системи контролю версій - це більш ефективний менеджмент. [58]

Висновок до розділу 4

Мобільний додаток «Musicians Shop» був розроблений за допомогою Flutter SDK, та найзручніших і найтехнологічніших інструментів, які доступні на сьогодні в pub.dev та Firebase. В основі цього мультиплатформного додатку використовується багато сучасних рішень певних задач, мультиплатформенний додаток зрозумілий як розробнику так і простому користувачеві. Цей додаток побудовано таким чином, що його легко модифікувати та змінювати і це відповідає сучасним тенденціям.

ВИСНОВКИ

На початку роботи було поставлено мету: дослідити процес розробки додатоку під Android, iOS, macOS та WEB з можливістю перегляду та створення оголошень, збору статистики додатку та адаптивного підбору оголошень під кожного користувача. Додаток має реалізовуватися з урахуванням усіх сучасних стандартів.

Актуальність цього полягає в тому, що мультиплатформеннні додатки та системи аналізу даних це найпопулярніша сучасна тенденція IT-ринку. Їх переваги: швидкість та зручність у користуванні, адаптивність до різних гаджетів користувачів, реалізація в різних варіаціях, актуальний підбір рекомендованих даних для кожного користувача окремо.

Було проаналізовано необхідність та актуальність розробки електронної дошки оголошень, визначені особливості роботи підсистем аналізу даних, мультиплатформенних додатків, у чому їх відмінність від нативних додатків та які є популярні тенденції по створенню різних додатків на сьогодні.

Було досліджено, вивчено і обрано набір інструментів для розробки сучасного мультиплатформного додатку, що надали змогу реалізувати всі необхідні функції. Було вирішено розробити мультиплатформений додаток під Android, iOS, macOS та WEB, застосовуючи SDK - Flutter, яка була визначена як оптимальна IT-технологій для реалізації необхідного функціоналу.

Для додатку «Musicians Shop», була обрана архітектура представлення GetX. Щоб реалізувати цю архітектуру у Flutter-додатку застосовується бібліотека get. В додатку також дотримані принципи clean architecture. Додаток «Musicians Shop» використовує технології Firebase, що надали можливість реалізувати авторизацію користувачів, створення хмарної бази даних, надсилання повідомлень, збір та аналіз інформації, відстеження збоїв системи та хостингу веб-додатку.

Реалізовано основний функціонал мультиплатформного додатку, а саме, реєстрація користувачів, можливість користувачів додатку створювати, редагувати та видаляти свої оголошення або переглядати оголошення інших

користувачів та інформацію про їх авторів. Реалізована система пошуку, система вподобань та система адаптивного підбору оголошень під кожного користувача окремо.

Викладений у цій роботі огляд інструментів для створення мультиплатформених додатків, огляд мови програмування та фреймворку, а також рекомендації та висновки можуть бути використані при вивчені мови програмування Dart, Flutter SDK та Firebase, а розроблений додаток може бути практично реалізований на ринку сучасних мультиплатформених додатків.

Отже, під час виконання поставлених задач, була створена сучасна дошка оголошень у вигляді мультиплатформного додатку написаного на Flutter SDK для Android, iOS, macOS та WEB. Реалізовано усі заплановані функції мультиплатформенного додатку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Mobile apps usage reached an all-time high amidst stay-at-home measures due to COVID-19 pandemic. URL: https://www.appannie.com/en/insights/market-data/mobile-app-usage-surged-40-during-covid-19-pandemic/
2. Dart language. URL:https://dart.dev/
3. Flutter SDK. URL:https://flutter.dev/
4. Android Studio. URL:https://developer.android.com/studio
5. Firebase. URL:https://firebase.google.com/
6. James G., Witten D., Hastie T., Tibshirani R. An Introduction to Statistical Learning with Applications in R – New York: Springer, 2013. – 441 p.
7. Hastie T., Tibshiriani R. Wainwright M. Statistical Learning with Sparsity . The Lasso and Generalizations – New York: Chapman and Hall/CRC, 2015. – 367 p.
8. Aggarwal C.C., Reddy C.K. Data Clustering: Algorithms and Applications – Boca Raton, FL: CRC Press, 2014. – 620 p.
9. Hahs-Vaughn D.L., Lomax R.G. Statistical concepts: a second course – New York, NY: Routledge, 2020. – 763 p.
10. Нескородєва Т.В., Федоров Є.Є., Січко Т.В., Нескородєва А.Р. «Експертні та рекомендаційні системи». Навч. посібник для здобувачів вищої освіти спеціальностей 122 Комп'ютерні науки, 113 Прикладна математика, 111 Математика - Вінниця. ДонНУ імені Василя Стуса. 2022. – 320с.
11. Крохмалюк В.В., Нескородєва Т.В. Аналіз даних про серцечні захворювання методами статистичного навчання // Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів та молодих вчених "Прикладні інформаційні технології" (29 квітня 2020 року) - Вінниця: ДонНУ імені Василя Стуса. С. 29-32. URL: https://r.donnu.edu.ua/handle/123456789/1491
12. Зінченко Б.В. Нескородєва Т.В. Аналіз даних про короновірус у світі методами статистичного навчання // Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів та молодих вчених

"Прикладні інформаційні технології" (29 квітня 2020 року) - Вінниця: ДонНУ імені Василя Стуса. С. 19-21. URL: https://r.donnu.edu.ua/handle/123456789/1491

13. Токар М.О. Нескородєва Т.В. Аналіз даних про мобільні ігри-стратегії засобами мови R. Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів та молодих вчених "Прикладні інформаційні технології" (29 квітня 2020 року) - Вінниця: ДонНУ імені Василя Стуса. С. 69-71. URL:
https://r.donnu.edu.ua/handle/123456789/1491
14. Сніжинський М.В. Нескородєва Т.В. Аналіз даних про рейтинг фільмів на платформі оцінювання IMDB засобами мови R. Матеріали всеукраїнської науково-практичної конференції для студентів, аспірантів та молодих вчених "Прикладні інформаційні технології" (29 квітня 2020 року) - Вінниця: ДонНУ імені Василя Стуса. С. 57-58. URL: https://r.donnu.edu.ua/handle/123456789/1491
15. Кучер М.О., Нескородєва Т.В. Аналіз факторів впливаючих на рівень щастя населення методами статистичного навчання Матеріали II всеукраїнської науково-практичної конференції «Комп'ютерні технології обробки даних» (м. Вінниця, 10 грудня 2021 р.) – Вінниця, ДонНУ імені Василя Стуса, 2021. – С. 39-42.
16. Огороднік М. О. Нескородєва Т.В. Аналіз даних про мобільні додатки APPLE IOS засобами мови R. Матеріали II всеукраїнської науково-практичної конференції «Комп'ютерні технології обробки даних» (м. Вінниця, 10 грудня 2021 р.) – Вінниця, ДонНУ імені Василя Стуса, 2021. – С. 42-44.
17. Deepak G. A., Ella H., Ashish K. Studies in Computational Intelligence. Advanced Computational Intelligence Techniques for Virtual Reality in Healthcare. vol. 875. ISBN 978-3-030-35252-3 (eBook) https://doi.org/10.1007/978-3-030-35252-3. Springer Nature Switzerland AG 2020.

18. Матеріали всеукраїнської науково-практичної конференції «Комп'ютерні технології обробки даних». Вінниця, Донецький національний університет імені Василя Стуса. Кафедра комп'ютерних наук та інформаційних технологій. 04.12.2020 р. URL: https://jktod.donnu.edu.ua/issue/view/369
19. GitHub Docs. URL: https://docs.github.com/en
20. У чому відмінність вебдодатків від мобільних додатків URL: https://brander.ua/blog/u-chomu-vidminnist-vebdodatkiv-vid-mobilnykh-dodatkiv
21. Що таке веб-додаток? Яка різниця між веб-додатком і сайтом, pwa та spa URL:https://webcase.com.ua/uk/blog/cho-takoe-web-prilozhenie-vse-vidy/
22. Кому вигідно замовити розробку десктопного додатка URL: https://meraldev.com.ua/desktop-apps/
23. Що таке десктопна програма URL: https://wezom.com.ua/ua/blog/desktop-prilozhenie
24. Приклади гібридних аплікацій, які змінять вашу думку про Native vs Hybrid URL: https://luxnet.io/uk/blog/hybrid-app-examples-that-change-your-opinion-on-native-vs-hybrid
25. Технологія GetX URL:https://pub.dev/packages/get
26. Overlay support URL:https://pub.dev/packages/overlay_support
27. Carousel Slider URL:https://pub.dev/packages/carousel_slider
28. Cupertino Icons URL:https://pub.dev/packages/cupertino_icons
29. Syncfusion Flutter Charts URL: https://pub.dev/packages/syncfusion_flutter_charts
30. Визначення типів mime URL:https://pub.dev/packages/mime
31. Логер URLhttps://pub.dev/packages/logger
32. File Picker URL:https://pub.dev/packages/file_picker
33. Url Strategy URL: https://pub.dev/packages/url_strategy
34. Url Launcher URL:https://pub.dev/packages/url_launcher
35. Path Provider URL:https://pub.dev/packages/path_provider

36. Android Path Provider URL:https://pub.dev/packages/android_path_provider
37. Package Info Plus URL:https://pub.dev/packages/package_info_plus
38. Flutter Local Notifications
URL:https://pub.dev/packages/flutter_local_notifications
39. Flutter Lints URL: https://pub.dev/packages/flutter_lints
40. Flutter Launcher Icons Maker URL:
https://pub.dev/packages/flutter_launcher_icons_maker
41. Бен Штраубі, Скотт Чакон «Pro Git» 2-ге видання (2014)
42. Ітан Маркотт - «Реагуючий веб-дизайн»
43. Аарон Уолтер - «Емоційний веб-дизайн»
44. Введення в адаптивність від Google URL:
https://developers.google.com/web/fundamentals/design-and-ux/responsive?pageId=11164180663038928762 2
45. Стаття Ітана Маркотта «Реагуючий веб-дизайн» URL:
https://alistapart.com/article/responsive-web-design/
46. What is Flutter? A brief introduction about flutter. URL:
https://medium.com/mobiosolutions/what-is-flutter-a-brief-introduction-about-flutter-40532f8af809
47. How to use Firebase with Flutter URL:
https://medium.com/47billion/how-to-use-firebase-with-flutter-e4a47a7470ce
48. Add Firebase to Flutter URL:
https://viveky259259.medium.com/add-firebase-to-flutter-6bc9e2755284
49. Google Sign In With Flutter URL:
https://medium.flutterdevs.com/google-sign-in-with-flutter-8960580dec96
50. Flutter Firebase Database Medium With Code Examples URL:
https://www.folkstalk.com/2022/10/flutter-firebase-database-medium-with-code-examples.html
51. How to deploy Flutter WebApp using Google Cloud Run URL:
https://medium.com/tide-engineering-team/deploy-flutter-webapp-using-gcp-cloud-run-4511c3d87e92

52. Cloud Computing, understanding the Google Cloud Platform - GCP URL: https://medium.datadriveninvestor.com/5-points-you-should-know-to-start-using-google-cloud-platform-faaaec31a421
53. Let's GetX in Flutter URL: https://medium.com/mindful-engineering/lets-getx-in-flutter-4eaff2826ac7
54. The Flutter GetX Ecosystem - State Management URL: https://medium.com/flutter-community/the-flutter-getx-ecosystem-state-management-881c7235511d
55. GetX State Management In Flutter URL: https://medium.flutterdevs.com/getx-state-management-in-flutter-a9710277b0bc
56. 3-Flutter update? Flutter problems! - The web coder path to Flutter URL: https://medium.com/flutter-community/3-flutter-update-flutter-problems-the-web-coder-path-to-flutter-c1522096ca0
57. Flutter App with Firebase Cloud Firestore URL: https://mercyjemosop.medium.com/flutter-app-with-firebase-cloud-firestore-17284f00ea68
58. How to integrate Android Studio with GitHub URL: https://medium.com/@justynagolawska/how-to-integrate-android-studio-with-github-8c5d6bec5d28
59. Dart (DartLang) Introduction: Getting started with Dart/Flutter URL: https://medium.com/run-dart/dart-dartlang-introduction-getting-started-with-dart-flutter-c5f59b8c456b
60. Flutter State Management: BLoC Pattern URL: https://medium.com/@aaron.chu/flutter-state-management-bloc-pattern-9cd6011c699