ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

КВАЛІФІКАЦІЙНА РОБОТА

на тему: « Розробка мобільного додатку для інтернет магазину на основі Flutter »

на здобуття освітнього ступеня бакалавра (магістра)
зі спеціальності 122 Комп'ютерні науки
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні науки
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Владислав ДАВИДЕНКО
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр. КНД-41
Владислав ДАВИДЕНКО
Ім'я, ПРІЗВИЩЕ

Керівник: Віктор ВИШНІВСЬКИЙ
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
д.т.н. професор

Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2024

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Комп'ютерних наук

Ступінь вищої освіти Бакалавр

Спеціальність 122 Комп'ютерні науки

Освітньо-професійна програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедрою Віктор ВИШНІВСЬКИЙ
_____ Ім'я, ПРІЗВИЩЕ
«____»_____2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Давиденко Владислав Олександрович
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: Розробка мобільного додатку для інтернет магазину на основі Flutter

керівник кваліфікаційної роботи ВИШНІВСЬКИЙ Віктор Вікторович,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)

затверджені наказом Державного університету інформаційно-комунікаційних технологійвід «27» лютого 2024 р. №36

2. Строк подання кваліфікаційної роботи «31» травня 2024 р.

3. Вихідні дані до кваліфікаційної роботи: Огляд існуючих мобільних додатків;
розробка мобільного додатку для інтернет-магазину на базі Flutter;
тестування та вдосконалення додатку з метою забезпечення;
аналіз результатів та висновки щодо ефективності використання Flutter.
Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Вступне пояснення теми та актуальності дослідження.
2. Формулювання цілей та завдань дослідження, а також опис методів та підходів, використаних у дослідженні.
3. Представлення отриманих результатів та їх аналіз. Узагальнення висновків та пропозиції щодо подальших досліджень.

4. Перелік ілюстративного матеріалу: *презентація*

5. Дата видачі завдання «23» лютого 2024 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Проведення літературного огляду та аналізу сучасного стану ринку.	15.04.2024 р.	
2	Формулювання цілей та завдань дослідження, визначення методів та підходів.	20.04.2024 р.	
3	Збір та аналіз даних, розробка та тестування прототипу додатку.	25.04.2024 р.	
4	Виконання експериментів, обробка результатів та аналіз отриманих даних.	10.05.2024 р.	
5	Написання тексту розрахунково-пояснювальної записки та підготовка презентації результатів дослідження.	15.05.2024 р.	
6	Оформлення та завершення роботи над кваліфікаційною роботою.	17.05.2024 р.	
7	Підготовка доповіді до захисту.	20.05.2024 р.	

Здобувач(ка) вищої освіти _____ Владислав ДАВИДЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник
кваліфікаційної роботи _____ Віктор ВИШНІВСЬКИЙ
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи: 51 сторінку, 16 рисунків, 31 джерело.

Метою дослідження є підвищення ефективності розробки сучасних методів підвищення надійності кросплатформної розробки з використанням Flutter.

Об'єкт дослідження є процес розробки удосконалених методів підвищення надійності кросплатформної розробки з використанням Flutter.

Предмет дослідження є розробка та оцінка сучасних методів підвищення надійності крос-платформної розробки з використанням Flutter.

Розробка мобільних додатків стала стратегічно важливим кроком для багатьох компаній, які бажають забезпечити зручний доступ до своїх товарів та послуг. Особливо актуальною стала ця тема у зв'язку зі стрімким розвитком онлайн-торгівлі та зміною у способах, якими користувачі взаємодіють з магазинами. Мобільні додатки на основі Flutter відкривають нові можливості для підприємств у залученні клієнтів та підвищенні їхнього рівня задоволеності від обслуговування.

Зокрема, Flutter дозволяє розробникам створювати кросплатформенні додатки з високою продуктивністю та стабільністю. Це дозволяє ефективно використовувати ресурси для розробки та підтримки додатку, що є важливим аспектом в умовах постійних змін у сфері технологій та вимог користувачів. Таким чином, розробка мобільного додатку для інтернет-магазину на основі Flutter може стати ключовим конкурентним перевагою для бізнесу в сучасному цифровому середовищі.

КЛЮЧОВІ СЛОВА: FLUTTER ПЕРЕДОВІ МЕТОДИ, МІЖПЛАТФОРМНА РОЗРОБКА

ЗМІСТ

ВСТУП

Актуальність дослідження. Розробка мобільних додатків стала стратегічно важливим кроком для багатьох компаній, які бажають забезпечити зручний доступ до своїх товарів та послуг. Особливо актуальною стала ця тема у зв'язку зі стрімким розвитком онлайн-торгівлі та зміною у способах, якими користувачі взаємодіють з магазинами. Мобільні додатки на основі Flutter відкривають нові можливості для підприємств у залученні клієнтів та підвищенні їхнього рівня задоволеності від обслуговування.

Зокрема, Flutter дозволяє розробникам створювати кросплатформенні додатки з високою продуктивністю та стабільністю. Це дозволяє ефективно використовувати ресурси для розробки та підтримки додатку, що є важливим аспектом в умовах постійних змін у сфері технологій та вимог користувачів. Таким чином, розробка мобільного додатку для інтернет-магазину на основі Flutter може стати ключовим конкурентним перевагою для бізнесу в сучасному цифровому середовищі.

Вибрані теми є новаторськими у створенні високоякісних, конкурентоспроможних мобільних додатків і важливі як для розробників, так і для бізнесу, оскільки використання Flutter є ключовим фактором підвищення ефективності розробки, зниження витрат і прискорення запуску продуктів.

Це ключовий фактор підвищення ефективності розробки, зниження витрат і прискорення запуску продуктів.

Об’єкт дослідження є процес розробки удосконалених методів підвищення надійності кросплатформної розробки з використанням Flutter.

Метою дослідження є підвищення ефективності розробки сучасних методів підвищення надійності кросплатформної розробки з використанням Flutter.

Предмет дослідження є розробка та оцінка сучасних методів підвищення надійності крос-платформної розробки з використанням Flutter.

Наукові завдання:

1. Проаналізувати ключові особливості та переваги Flutter як кросплатформного фреймворку, включаючи його архітектуру, компоненти та інструменти розробки.

2. Дослідити кращі практики та методології в розробці мобільних додатків за допомогою Flutter, з акцентом на ефективність та продуктивність.

3. Розробити та проаналізувати практичний приклад мобільного додатку, створеного з використанням Flutter, для оцінки його функціональності, продуктивності та користувацького досвіду.

Методи дослідження – опрацювання літератури за даною темою, аналіз експлуатаційної документації, міжнародних стандартів та їх порівняння.

1 КОНЦЕПЦІЇ FLUTTER ДЛЯ НАДІЙНОЇ КРОСПЛАТФОРМНОЇ РОЗРОБКИ

1.1 Огляд фреймворку Flutter

Розроблений компанією Google, Flutter – це новітній фреймворк з відкритим вихідним кодом, який революціонізував світ мобільної розробки. Завдяки своїй унікальній функціональності та гнучкості, Flutter швидко завоював популярність серед розробників (рис. 1.1). Його унікальна візуальна мова та вбудована підтримка крос-платформної розробки дозволяє створювати високоякісні нативні додатки для різних операційних систем, включаючи Android та iOS, з єдиної кодової бази [1].

Рисунок 1.1 - Підтримка пристроїв Flutter

Історія Flutter бере свій початок у 2017 році, коли Google вперше представив фреймворк на світовому ринку. З тих пір він постійно розвивається і додає нові функції. Однією з ключових особливостей Flutter є використання мови програмування Dart, яка спеціально пристосована для створення інтерактивних та високопродуктивних мобільних додатків, що надає переваги та значно спрощує і прискорює процес розробки [1].

З часом Flutter довів свою ефективність у вирішенні поширених проблем, з якими стикаються розробники мобільних додатків. До них відносяться проблеми крос-платформної сумісності та необхідність розробки додатків для декількох систем одночасно.

Flutter ефективно вирішує ці проблеми, надаючи єдину кодову базу, яка забезпечує узгодженість і якість кінцевого продукту.

Розробка Flutter фокусується на широкій підтримці різних платформ, що дозволяє розробникам створювати додатки, які працюють на різних пристроях, використовуючи єдину кодову базу. Це не тільки спрощує процес розробки, але й покращує узгодженість та якість кінцевого продукту - Flutter постійно оновлюється та вдосконалюється, що свідчить про активне заохочення відкритої спільноти розробників, яка сприяє розвитку цієї технології [2].

З моменту свого створення Flutter зумів вирости з простої ідеї в повноцінний інструмент, який використовується великими компаніями і незалежними розробниками по всьому світу для створення інноваційних та ефективних додатків. Постійний розвиток та адаптація до мінливих потреб ринку та розробників зробили Flutter однією з ключових технологій крос-платформної розробки на сьогоднішній день.

Незважаючи на те, що Flutter є відносним новачком на ринку технологій, він зміг швидко адаптуватися до вимог сучасних розробників. Його головна перевага полягає в тому, що він пропонує швидкий та інтуїтивно зрозумілий процес розробки, що дозволяє розробникам зосередитися на створенні якісних продуктів, а не на вирішенні питань технічної сумісності між різними платформами. Це відбувається завдяки широкому спектру готових до використання віджетів, які дозволяють легко створювати складні та візуально привабливі інтерфейси [2].

Ще один важливий аспект, на який варто звернути увагу, - це спільнота Flutter. З моменту свого запуску Flutter створив активну спільноту розробників, яка активно співпрацює. Ця спільнота не лише ділиться знаннями та досвідом, але й активно сприяє розвитку фреймворку, пропонуючи нові функції, виправлення помилок та вдосконалення. Це сприяє постійному розвитку Flutter як відкритої платформи, що підтримується не тільки Google, але й широкою спільнотою ентузіастів.

Завдяки своїй модульності та гнучкості Flutter пропонує широкий спектр можливостей для індивідуальної та інноваційної розробки. Розробники можуть

створювати додатки, які не тільки виглядають нативно на кожній платформі, але й відповідають конкретним вимогам і побажанням своїх клієнтів. Такий підхід робить Flutter особливо привабливим для розробників, які хочуть створювати унікальні, високоякісні продукти [3].

Загалом, Flutter продовжує зміцнювати свої позиції на ринку крос-платформної розробки. Завдяки своїй високій продуктивності, гнучкості та підтримці спільноти, він став незамінним інструментом для розробників, які хочуть створювати інноваційні та ефективні мобільні додатки. Оскільки все більше компаній і розробників відкривають для себе переваги Flutter, його вплив на ринку крос-платформної розробки, безсумнівно, буде продовжувати зростати. Flutter - це не тільки платформа для створення додатків, але і платформа, яка заохочує інновації, спрощуючи процес розробки. Це ціла екосистема, яка відкриває нові горизонти для заохочення інновацій, спрощення процесу розробки та створення унікального користувацького досвіду. З цих причин Flutter є однією з найперспективніших та найефективніших технологій у світі крос-платформної розробки на сьогоднішній день [3].

1.2 Архітектура Flutter та її вплив на кросплатформну розробку

Ядро Flutter, що відповідає за інтерпретацію та виконання коду Dart, рендеринг елементів користувацького інтерфейсу та взаємодію з базовою платформою (Android, iOS тощо). Містить бібліотеки для графіки, анімації, роботи з мережею тощо.

Основні бібліотеки надають основні будівельні блоки для створення користувацьких інтерфейсів, включаючи віджети, макети та управління станом. Вони не залежать від платформи і працюють однаково на всіх пристроях.

Будівельні блоки користувацького інтерфейсу Flutter. Віджети можна повторно використовувати, поширювати, зберігати і зберігати для створення складних і динамічних інтерфейсів. Специфічні для платформи віджети, які надають оригінальні елементи користувацького інтерфейсу, такі як кнопки, текстові поля та панелі навігації. Вони адаптуються до мови дизайну платформи

при використанні рушія Flutter для рендерингу; макет цих компонентів платформи Flutter показано на рисунку 1.2

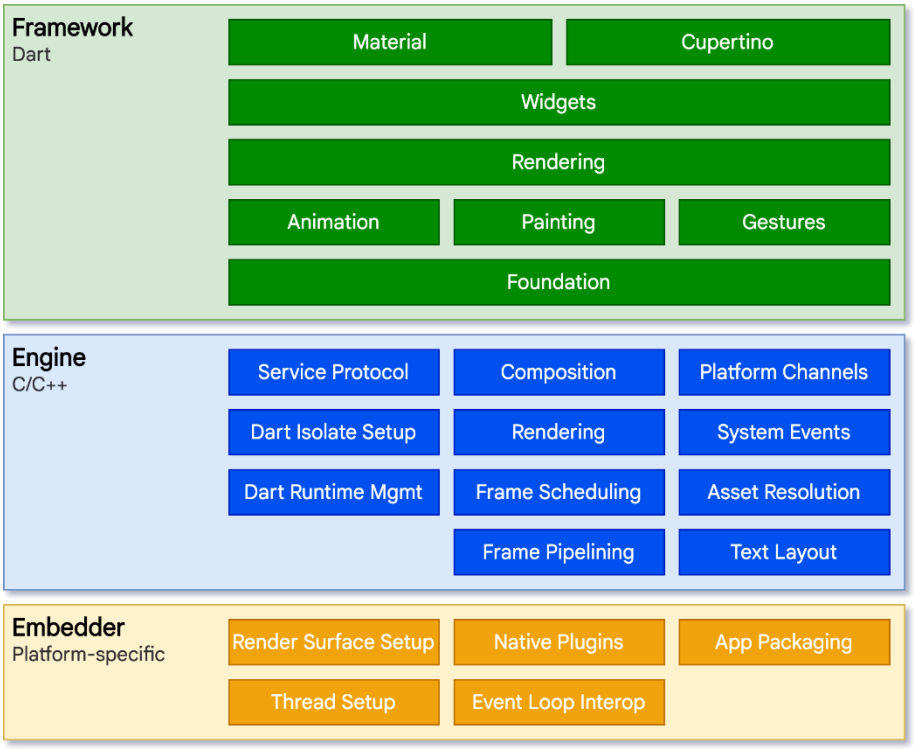

Рисунок 1.2 - Схема Flutter платформи

Повторне використання коду: Більша частина коду знаходиться в незалежних від платформи базовому та віджетному рівнях. Це означає, що основну логіку потрібно написати лише один раз і вона працює на різних платформах, що значно скорочує час і зусилля на розробку [4].

Конфігурація для конкретної платформи: спеціально розроблені віджети дозволяють вам налаштувати зовнішній вигляд вашого додатку для кожної платформи без необхідності переписувати основний функціонал. Це забезпечує нативний досвід для користувачів на всіх пристроях.

Теплий перезапуск: Flutter має унікальну перевагу теплого перезавантаження. Це гарантує, що зміни коду миттєво відображаються на пристрої, навіть під час роботи програми. Це прискорює розробку і полегшує крос-платформну налагодження.

Flutter надає доступ до нативної функціональності платформи через плагіни. Інтегруючись із специфічними для платформи API та сервісами, плагіни

розширюють функціональність вашого додатку і ще більше збагачують крос-платформний досвід.

Переваги використання архітектури Flutter:

— повторне використання коду і гаряче перезавантаження значно прискорюють процес розробки, дозволяючи створювати крос-платформні додатки швидше і ефективніше;

— завдяки структурованій архітектурі код легше розуміти, підтримувати та оновлювати, що призводить до більш стабільного процесу розробки;

— специфічні для платформи віджети та гаряче перезавантаження роблять ваш додаток нативним на кожній платформі, забезпечуючи послідовний та приємний користувацький досвід [4].

Розуміючи і застосовуючи принципи архітектури Flutter, можна використовувати можливості цього фреймворку для ефективного створення високоякісних крос-платформних додатків і забезпечення безперебійної роботи користувачів на кожному пристрої [4].

1.3 Порівняння Flutter з іншими кросплатформними рішеннями

Flutter – це інструментарій для розробки користувацьких інтерфейсів з відкритим вихідним кодом, розроблений компанією Google, який дозволяє розробникам створювати кастомні додатки для мобільних, веб та настільних комп'ютерів з єдиної кодової бази. Він надає мову програмування Dart та багатий набір готових віджетів для створення адаптивних та візуально привабливих користувацьких інтерфейсів. Статистика використання різних крос-платформних рішень показана на рисунку 1.3 [5].

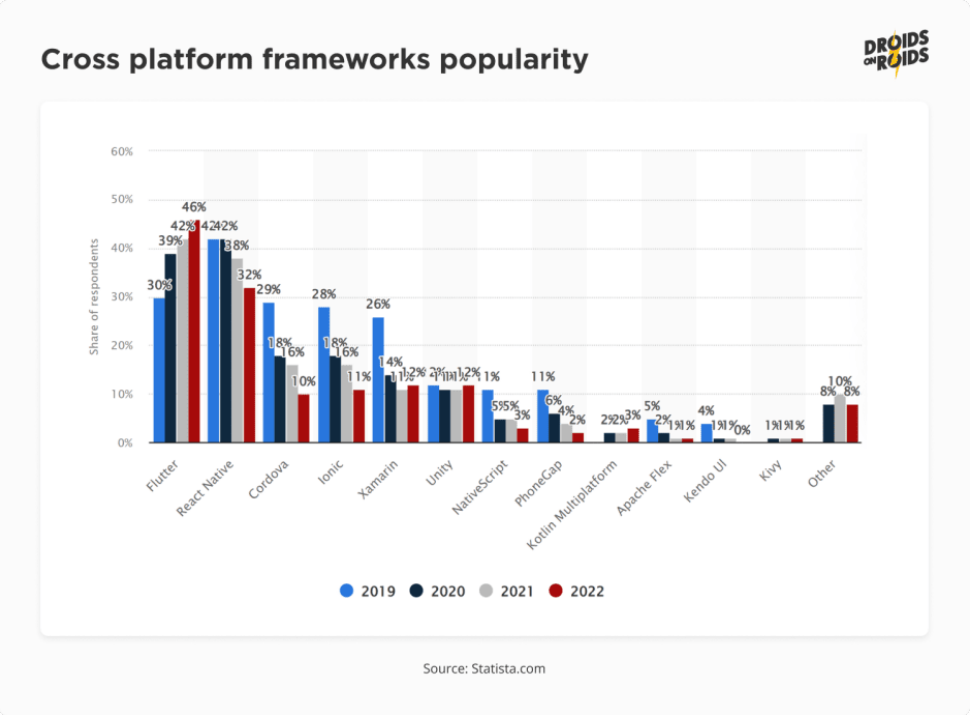

Рисунок 1.3 - Статистика використання крос-платформних рішень

Швидка розробка з гарячим перезавантаженням, що дозволяє оновлювати програму в режимі реального часу без втрати її стану. Великий набір готових віджетів та широкий спектр можливостей кастомізації; Високопродуктивні додатки з нативною продуктивністю завдяки збірці AOT (Advanced Out-of-the-Box). Потужна підтримка від Google та зростаюча спільнота.

Відносно нова і невелика екосистема в порівнянні з іншими фреймворками; Dart менш популярний, ніж JavaScript, тому розробникам може знадобитися вивчити нову мову.

React Native Розроблений Facebook, React Native – це фреймворк з відкритим вихідним кодом, який дозволяє розробникам створювати мобільні додатки за допомогою JavaScript та React. Він використовує нативні компоненти для рендерингу та надає додатку нативний вигляд і відчуття.

Плюси: Існує велика і зріла екосистема з численними бібліотеками та плагінами. Використовує JavaScript, популярну і широко використовувану мову програмування. Сильна підтримка спільноти та Facebook; спільна кодова база для React (веб-) та React Native (мобільних) проектів [5].

Мінуси: Продуктивність може бути нижчою, ніж у нативних додатків, особливо для складних додатків. Для деяких функцій можуть знадобитися кастомні модулі, що може ускладнити розробку.

Xamarin: Зараз Xamarin є частиною Microsoft, це крос-платформне середовище розробки додатків, яке дозволяє розробникам створювати нативні додатки для Android, iOS та Windows, використовуючи C# та фреймворк .NET.

Переваги: Використовує C#, популярну та потужну мову програмування. Потужна підтримка Microsoft та інтеграція з Visual Studio. Спільна кодова база для різних платформ, до 90% коду можна використовувати повторно; доступ до нативних API та оптимізація продуктивності за допомогою Xamarin.Forms та Xamarin.Native.

Мінуси: більший розмір додатків у порівнянні з нативними додатками; менша спільнота та екосистема у порівнянні з React Native та Flutter.

Ionic – це фреймворк з відкритим вихідним кодом для створення крос-платформних мобільних і веб-додатків з використанням веб-технологій, таких як HTML, CSS і JavaScript; з використанням Angular, React або Vue.js для побудови логіки додатків і наданням бібліотеки готових UI-компонентів, які емулюють користувацькі елементи інтерфейсу [5].

Кожен з цих фреймворків має свої переваги та недоліки, і вибір між ними часто залежить від конкретних вимог проекту.

1.4 Роль Flutter у сучасній розробці мобільних додатків

У сучасному динамічному мобільному середовищі Flutter став сильним конкурентом у світі розробки мобільних додатків. Його вплив на те, як створюються та працюють додатки, незаперечний, а популярність постійно зростає. Щоб зрозуміти його важливість, давайте глибше розглянемо його специфічні функції:

Крос-платформна розробка: Ключовою перевагою Flutter є його крос-платформені можливості: На відміну від традиційних фреймворків, які вимагають окремих кодових баз для iOS і Android, Flutter дозволяє писати додатки для обох

платформ з єдиної кодової бази для обох платформ з єдиної кодової бази. Це означає, що

– Скорочення часу та витрат на розробку: Розробники можуть зосередитися на написанні коду один раз, заощаджуючи цінний час і ресурси порівняно з підтримкою окремих кодових баз.

– Швидший вихід на ринок: Завдяки єдиній кодовій базі додатки можна розгортати на обох платформах одночасно, швидше охоплюючи більшу кількість користувачів.

– Послідовний користувацький досвід: Ваш додаток Flutter виглядає і відчувається як нативний додаток як на iOS, так і на Android, забезпечуючи безперебійну роботу користувачів незалежно від пристрою.

Високопродуктивні додатки: Незважаючи на кросплатформеність, додатки Flutter мають таку ж продуктивність, як і нативні додатки. Це досягається завдяки

– Dart, ефективна мова програмування: Flutter компілює код Dart безпосередньо у власний машинний код, усуваючи потребу в проміжних інтерпретаторах і створюючи плавні, чуйні додатки.

– Skia, високоякісний рушій рендерингу: Skia сприяє користувацькому досвіду світового класу, забезпечуючи плавну анімацію та візуально приголомшливі елементи користувацького інтерфейсу.

Швидка розробка та ітерації: Flutter дозволяє розробникам створювати та ітерації своїх додатків швидше та ефективніше завдяки таким функціям, як

– Гаряче перезавантаження: Ця функція дозволяє розробникам вносити зміни в код "на льоту" на екрані без необхідності перекомпілювати весь додаток. Це значно скорочує цикли розробки та сприяє швидкому створенню прототипів.

– Велика бібліотека віджетів: Flutter надає повний набір готових віджетів для поширених елементів інтерфейсу, що зменшує потребу розробників у написанні коду з нуля та пришвидшує розробку [6].

Виразний та гнучкий користувацький інтерфейс: Flutter пропонує неперевершену гнучкість та контроль над дизайном інтерфейсу користувача. Розробники можуть

– Створювати кастомні віджети: створюйте унікальні елементи інтерфейсу, які ідеально відповідають їхньому баченню та потребам програми.

– Використовуйте можливості Dart: Використовуйте виразну та об'єктно-орієнтовану природу Dart для створення динамічних та інтерактивних користувацьких інтерфейсів.

– Інтеграція з нативними платформами: безперешкодний доступ до функціональності нативної платформи у ваших додатках Flutter для паритету функцій та оптимальної продуктивності.

Поза межами мобільних пристроїв: Flutter блищить у мобільній розробці, але на цьому його можливості не обмежуються - за допомогою Flutter розробники можуть створювати красиві та продуктивні додатки:

– Веб: Створюйте інтерактивні веб-додатки, використовуючи ту саму кодову базу, що й мобільні додатки.

– Для настільних комп'ютерів: Перенесіть свої мобільні додатки на настільні комп'ютери з мінімальними зусиллями і розширюйте свою аудиторію.

– Вбудовані системи: Легкість та ефективність Flutter робить його ідеальним інструментом для розробки додатків для пристроїв з обмеженими ресурсами.

Спільнота та адаптація: Flutter має сильну та енергійну спільноту розробників. Ця мережа підтримки заспокоює:

– Обширна документація та навчальні посібники: Доступні численні ресурси, які допоможуть вам вивчити Flutter і подолати труднощі розробки.

– Бібліотеки та плагіни з відкритим кодом: Існує велика колекція готових до використання рішень, які пришвидшують розробку та відповідають конкретним потребам.

– Постійні оновлення та покращення: Команда Flutter активно вдосконалює фреймворк, забезпечуючи стабільність і додаючи нові функції на регулярній основі [7].

Таким чином, Flutter відіграє трансформаційну роль у сучасній розробці мобільних додатків. Його кросплатформені можливості, фокус на продуктивності

та інструменти швидкої розробки дозволяють розробникам створювати красиві, продуктивні та багатофункціональні додатки швидше та ефективніше. Сфера його застосування виходить за межі мобільних пристроїв, а процвітаюча спільнота забезпечує постійний ріст і розвиток. Оскільки мобільний ландшафт продовжує розвиватися, вплив Flutter зростатиме, і він буде цінним інструментом для розробників на довгі роки [7].

Висновок до розділу 1

Flutter – це потужний інструмент для розробки кросплатформних мобільних додатків, який надає розробникам можливість швидко і ефективно створювати високоякісні додатки для Android та iOS. Огляд фреймворку підкреслив його простоту використання та широкі можливості для розробників. Архітектура Flutter розкриває його переваги у вигляді швидкості розробки та легкої масштабованості. Порівняння з іншими кросплатформними рішеннями підкреслює переваги Flutter у вигляді продуктивності та надійності.

Flutter відіграє важливу роль у сучасній розробці мобільних додатків, дозволяючи розробникам швидко реалізувати свої ідеї та пропозиції на ринку. Цей фреймворк відкриває шлях до створення якісних та високопродуктивних додатків, що задовольняють вимоги сучасних користувачів. Завдяки великій кількості доступних бібліотек та інструментів, Flutter стає невід'ємною частиною робочого процесу для багатьох розробників, які прагнуть досягти успіху у своїй сфері.

Концепції Flutter для надійної кросплатформної розробки виявляються дуже привабливими для розробників, оскільки цей фреймворк дозволяє створювати додатки для різних платформ швидко і з мінімальними зусиллями.

Огляд фреймворку Flutter демонструє його потужність та гнучкість. Flutter володіє великою кількістю вбудованих компонентів та можливостей для створення відмінних користувацьких інтерфейсів.

Архітектура Flutter є однією з його ключових переваг. Вона дозволяє розробникам створювати додатки, які працюють швидко і ефективно на різних платформах, забезпечуючи їм гнучкість у розробці.

Порівняння Flutter з іншими кросплатформними рішеннями підкреслює його переваги, такі як швидкість розробки та висока продуктивність.

Роль Flutter у сучасній розробці мобільних додатків важлива, оскільки він дозволяє розробникам створювати високоякісні та ефективні додатки, які задовольняють потреби сучасних користувачів.

2 ПРОГРЕСИВНІ МОЖЛИВОСТІ FLUTTER ДЛЯ МАСШТАБУВАННЯ ДОДАТКІВ

2.1 Управління станом за допомогою Flutter

Управління станом в Flutter є ключовою концепцією для створення динамічних та реактивних додатків. Цей підхід передбачає визначення стану як інформації, яку можна читати синхронно при побудові віджета і може змінюватися під час роботи програми. Ефективне управління станом дозволяє розробникам створювати більш надійні та підтримувані додатки.

У Flutter існують різні підходи до управління станом, кожен з яких має свої переваги та сфери застосування. Основні методи включають:

– Локальне управління станом (Local State Management) в Flutter використовується для керування станом додатка в межах конкретного віджета або невеликої групи віджетів. Цей підхід дозволяє зберігати та оновлювати дані, які відображаються в інтерфейсі користувача, без впливу на інші частини додатка. Кожен віджет має свій власний стан, і зміни в цьому стані відбуваються локально, в межах самого віджета або його батьківського віджета. Локальне управління станом дозволяє забезпечити швидку та ефективну роботу додатка, особливо в невеликих або середніх за розміром проектах [8].

– Підняття стану вгору (Lifting State Up) - це підхід управління станом в Flutter, при якому стан віджета піднімається до батьківського віджета для спільного використання стану між багатьма віджетами. Зазвичай цей підхід використовується, коли декілька віджетів потребують доступу до одного та того ж стану або коли зміна стану одного віджета повинна відображатися у інших віджетах. Підняття стану вгору дозволяє створювати більш складні та взаємозалежні додатки, де стан може бути змінений в одному місці і автоматично оновлений у всіх віджетах, які використовують цей стан. Цей підхід допомагає уникнути дублювання стану та забезпечити більшу узгодженість даних у додатку [9].

– Provider Package - це пакет у Flutter, який надає простий та зручний спосіб управління станом додатку. Він базується на концепції інверсії керування, де віджети не вибирають або не зберігають свій власний стан, а замість цього отримують стан від свого батьківського або родинного віджета. Provider Package дозволяє легко передавати та оновлювати стан між віджетами, що робить процес управління станом більш простим та ефективним. Він також надає можливість використання різних видів моделей стану, таких як ChangeNotifier та StreamProvider, щоб відповідати потребам конкретного додатку. Provider Package став популярним серед розробників Flutter завдяки своїй простоті використання та ефективності у управлінні станом додатку [10].

– Bloc Pattern (Business Logic Component) - це підхід до управління станом у Flutter, який допомагає відокремити бізнес-логіку від інтерфейсу користувача. За допомогою Bloc Pattern створюються спеціальні об'єкти (блоки), які відповідають за обробку подій та вирішення бізнес-логіки додатку. Bloc Pattern базується на використанні різних типів потоків даних, таких як Stream у Dart, для передачі та оновлення стану додатку. Це дозволяє створювати додатки, які реагують на вхідні події та оновлюють свій стан відповідно до цих подій.

Основні переваги Bloc Pattern уключають у собі покращене управління станом додатку, відокремлення бізнес-логіки від UI, спрощення тестування та підвищення його надійності. Водночас, використання Bloc Pattern може бути складнішим у порівнянні з іншими підходами до управління станом, такими як Provider Package.

Управління станом у Flutter є важливим аспектом розробки додатків, оскільки дозволяє зберігати та оновлювати дані відповідно до вимог додатку. Вірне управління станом допомагає забезпечити правильну роботу додатку, зменшити його складність та зробити його більш надійним і швидким. Ось деякі причини, чому управління станом є важливим у Flutter:

– Управління станом дозволяє додатку реагувати на дії користувача та змінювати відображення в реальному часі. Наприклад, відразу оновлювати дані на екрані після зміни вхідних даних.

– Правильне управління станом дозволяє відокремити бізнес-логіку додатку від його інтерфейсу. Це робить код більш структурованим, зрозумілим та легким для тестування.

– Управління станом полегшує тестування додатку, оскільки окремі частини програми можуть бути протестовані незалежно від інших. Це дозволяє виявляти та виправляти помилки більш ефективно.

– Правильне управління станом дозволяє зменшити кількість звернень до сервера та збільшити швидкість роботи додатку, оскільки дані можуть зберігатися та оновлюватися локально.

– При розвитку додатку можливо збільшення кількості функцій і даних. Управління станом допомагає підтримувати масштабованість додатку, забезпечуючи його стабільність і продуктивність навіть при зростанні обсягів даних та функцій [11].

2.2 Ключові аспекти при розробці додатків на базі рішення Flutter

Оптимізація продуктивності є одним з ключових завдань при створенні додатків на Flutter. Один з способів покращити продуктивність – це ефективне використання асинхронних операцій. Завдяки архітектурним особливостям Flutter, використання Future та Stream може допомогти краще управляти асинхронним кодом та підвищити реактивність додатку. Працюючи з асинхронним кодом в Flutter, слід пам'ятати про основні інструменти для роботи з асинхронними операціями: «Future» та «Stream». «Future» використовується для представлення одноразового результату асинхронної операції, тоді як «Stream» дозволяє отримувати послідовні значення з асинхронного джерела (рис. 2.1). При правильному використанні ці інструменти дозволяють збільшити реактивність додатку та покращити його продуктивність [12].

```dart
Future<void> fetchData() async {
  // Асинхронний запит до серверу
  // ...
}
Stream<int> countDown(int from, int to) async* {
  for (int i = from; i >= to; i--) {
    await Future.delayed(Duration(seconds: 1));
    yield i;
  }
}
```

Рисунок 2.1 - Приклад використання асинхронних операцій

Мемоізація та кешування є важливими стратегіями для оптимізації продуктивності додатків Flutter. Мемоізація полягає в збереженні результатів обчислень для певних вхідних даних, щоб уникнути повторного обчислення в майбутньому. Це особливо корисно для складних або довгих обчислень, які можуть виконуватися багато разів (рис. 2.2).

Кешування використовується для зберігання та управління кешем даних, щоб зменшити час доступу до них у майбутньому. Це може бути корисно для збереження результатів запитів до бази даних або результатів мережевих запитів, щоб уникнути повторних запитів на сервер [13].

Ці підходи дозволяють покращити продуктивність додатків Flutter, зменшуючи час виконання операцій та збільшуючи швидкодію додатку.

```dart
final Map<String, dynamic> _cache = {};
dynamic getData(String key) {
  if (_cache.containsKey(key)) {
    return _cache[key];
  } else {
    // Логіка отримання даних
    // ...
    _cache[key] = fetchedData;
    return fetchedData;
}
}
```

Рисунок 2.2 - Приклад мемоізації

Для оптимізації зображень у Flutter можна використовувати такі підходи: Зменшення розміру зображень за допомогою стиснення без втрати якості. Для цього можна використовувати спеціальні інструменти або бібліотеки, такі як

flutter_image_compress або використовувати стиснуті формати зображень WebP (рис. 2.3).

Зображення можна завантажувати з низькою роздільною здатністю та замінювати їх на вище роздільність під час відображення на екрані. Це дозволяє зменшити час завантаження та використання пам'яті [14].

Кешування зображень дозволяє зберігати завантажені зображення на диску або в пам'яті для подальшого використання, що зменшує кількість запитів до сервера та підвищує швидкодію додатку.

```
Image.network(
  'https://example.com/image.webp',
  fit: BoxFit.cover,
)
```

Рисунок 2.3 - Оптимізація зображення за допомогою стиснутого формату

Змінна const використовується для створення об'єктів, значення яких відомо на етапі компіляції і не може змінюватись під час виконання програми. Це дозволяє заощадити пам'ять, оскільки компілятор може оптимізувати використання констант.

Змінна final (рис. 2.4)вказує на те, що значення змінної може бути призначено лише один раз і не може бути змінено після цього. Це дозволяє покращити читабельність коду та запобігти непередбачуваним змінам значень змінних.

```
final String appName = 'MyApp';
const double piValue = 3.14;
```

Рисунок 2.4 - Використання змінних

2.3 Стратегії тестування у розробці на Flutter

Стратегії тестування грають важливу роль у розробці додатків на Flutter, дозволяючи впевнитися в якості та надійності програмного забезпечення. Ось декілька ключових стратегій тестування, які можна використовувати у розробці на Flutter:

1. Юніт-тестування (Unit Testing): Цей вид тестування спрямований на перевірку окремих функцій, методів або класів для переконливості у їхній правильності та працездатності (рис. 2.5). В Flutter для юніт-тестування часто використовують пакет flutter_test.

```dart
int add(int a, int b) {
  return a + b;
}
void main() {
  test('Тест додавання', () {
    expect(add(2, 3), equals(5));
  });
}
```

Рисунок 2.5 - Приклад Unit Testing для функції вирахування суми

2. Інтеграційне тестування (Integration Testing): Цей тип тестування спрямований на перевірку взаємодії різних компонентів додатку. У Flutter можна використовувати пакет integration_test для написання та виконання інтеграційних тестів (рис. 2.6).

```dart
void main() {
  testWidgets('Тест екрану входу', (WidgetTester tester) async {
    await tester.pumpWidget(MyApp());

    // Симулювання введення даних та натискання кнопки входу
    await tester.enterText(find.byKey(Key('username')), 'user123');
    await tester.enterText(find.byKey(Key('password')), 'pass123');
await tester.tap(find.byType(ElevatedButton));

    await tester.pump();
    // Перевірка, чи користувач ввійшов у систему
    expect(find.text('Вітаємо, user123!'), findsOneWidget);
  });
}
```

Рисунок 2.6 - Приклад Integration Testing для екрану входу

3. Віджет-тестування (Widget Testing): Це тестування компонентів користувацького інтерфейсу, таких як віджети (рис. 2.7). У Flutter для віджет-тестування використовується клас WidgetTester.

```dart
void main() {
  testWidgets('Тест кнопки', (WidgetTester tester) async {
    // Збудування віджету кнопки
    await tester.pumpWidget(ElevatedButton(
      onPressed: () {},
      child: Text('Натискання'),
    ));

    // Перевірка, чи відображається текст на кнопці
    expect(find.text('Натискання'), findsOneWidget);

    // Симулювання натискання на кнопку
    await tester.tap(find.byType(ElevatedButton));
    await tester.pump();

    // Перевірка, чи змінився стан після натискання
    // (це залежить від логіки додатку)
    // expect(...);
  });
}
```

Рисунок 2.7 - Приклад Widget Testing для віджету кнопки

4. Поведінкове тестування (Behavioral Testing): Цей вид тестування спрямований на перевірку відповідності додатку до визначених специфікацій та очікувань щодо його функціонування. У Flutter можна використовувати фреймворки, такі як flutter_driver, для поведінкового тестування [15].

5. Тестування вручну (Manual Testing): Незважаючи на автоматизовані тести, важливо також проводити ручне тестування, оскільки воно дозволяє виявити проблеми, які можуть залишитися невиявленими автоматизованими тестами.

2.4 Приклади використання Flutter в корпоративних рішеннях

У сучасному стані мобільного розвитку Flutter набирає популярності серед корпоративних рішень завдяки своїм крос-платформним можливостям та швидкості розробки. Успіхи Flutter у корпоративному секторі:

1. міжнародний гігант електронної комерції Alibaba, один з найбільших світових гігантів електронної комерції, використовує Flutter для розробки своїх мобільних додатків, а завдяки крос-платформним можливостям Flutter, Alibaba може швидко розгортати оновлення та нові функції для мільйонів користувачів на різних платформах, зберігаючи при цьому узгоджений дизайн та функціональність. Розгортайте швидко [15].

2. Команда Google Ads успішно використовує Flutter для покращення користувацького інтерфейсу платформи. Це дозволяє їм швидше впроваджувати нові функції та зберігати зовнішній вигляд програми на різних пристроях.

3. Reflectly, додаток для ведення щоденника та самодопомоги, також створений за допомогою Flutter. Завдяки цьому фреймворку, Reflectly пропонує плавний і приємний користувацький досвід та швидкі, безперебійні оновлення.

4. BMW використовує Flutter для розробки інноваційних мобільних додатків. Це дозволяє автовиробнику швидко взаємодіяти з клієнтами, пропонуючи унікальні та корисні функції.

5. Телекомунікаційна компанія Tencent використовує Flutter у власних проектах. Tencent, провідна компанія в секторі телекомунікацій та технологій, використовує Flutter для розробки своїх мобільних проектів. Це є свідченням гнучкості та ефективності Flutter у великих корпоративних середовищах.

6. DHL logistics (DHL), одна з провідних світових логістичних компаній, використовує Flutter для покращення мобільності своїх послуг та систем відстеження. Flutter спрощує відстеження відправлень та оптимізує робочі процеси.

7. IT-консалтингова компанія Accenture Enterprise Vehicle Development використовує Flutter для створення корпоративних інструментів та бізнес-додатків для своїх клієнтів. Це включає інтеграцію з внутрішніми системами, аналітику даних та інші корпоративні функції; Flutter допомагає Accenture швидко розгортати високоякісні рішення в різних відділах та галузях.

8. Фінансова компанія Revolut, що надає фінансові послуги, використовує Flutter для створення мобільних додатків; гнучкість і швидкість розробки Flutter

дозволяє Revolut швидко впроваджувати нові фінтех-функції та вдосконалення для задоволення зростаючих потреб клієнтів.

9. Провідна освітня платформа Coursera використовує Flutter для створення мобільних додатків, які надають користувачам легкий доступ до курсів і матеріалів курсів Flutter допомагає забезпечити узгодженість дизайну і функціональності на різних платформах і забезпечує високий рівень користувацького досвіду Coursera також використовує Flutter для забезпечення високого рівня користувацького досвіду [16].

Висновок до розділу 2

Управління станом є ключовою концепцією для створення динамічних та реактивних додатків у Flutter. Розділ детально розглядає різні методи управління станом, такі як локальне управління станом, підняття стану вгору, використання пакету Provider та патерну Bloc. Ці методи дозволяють ефективно управляти станом додатку та підтримувати його реактивність .

Перечислено ключові аспекти при розробці додатків на Flutter, такі як оптимізація продуктивності, що включає в себе використання асинхронних операцій та стратегій кешування, а також стратегії тестування для підтримки якості додатку.

Розглянуто стратегії тестування для забезпечення якості додатків на Flutter. Зокрема, висвітлюються підходи до тестування інтерфейсу користувача, функціонального та інтеграційного тестування, які допомагають забезпечити стабільність та надійність додатків.

Наведені приклади успішного використання Flutter у корпоративних рішеннях, таких як створення внутрішніх інструментів та додатків для співробітників, корпоративних платформ електронної комерції та систем управління відносинами з клієнтами.

Продемонстровано широкі можливості Flutter у створенні високоякісних, ефективних та масштабованих додатків, що підтримують якість та надійність у корпоративному середовищі.

3 РОЗРОБКА МАСШТАБНОГО МОБІЛЬНОГО ДОДАТКУ «ТОРТОР» НА БАЗІ ФРЕЙМВОРКУ FLUTTER

3.1 Створення архітектури додатку

Коли ми говоримо про створення мобільного додатку на кшталт ТорТор, одним з перших і найважливіших етапів є розробка архітектури проекту. Архітектура додатку - це своєрідний каркас, який визначає, як компоненти додатку будуть взаємодіяти і спілкуватися із зовнішніми системами і сервісами [17].

У випадку з ТорТор головним завданням було створити архітектуру, яка б не тільки забезпечувала гнучкість і масштабованість, а й полегшувала майбутні зміни і додавання нових функцій. З огляду на це, було вирішено використовувати комбінацію чистої архітектури (рис. 3.1) і моделі BLoC для управління кейсами. У цьому підході логіка додатку розділена на окремі шари, і кожен шар відповідає за свою частину логіки.

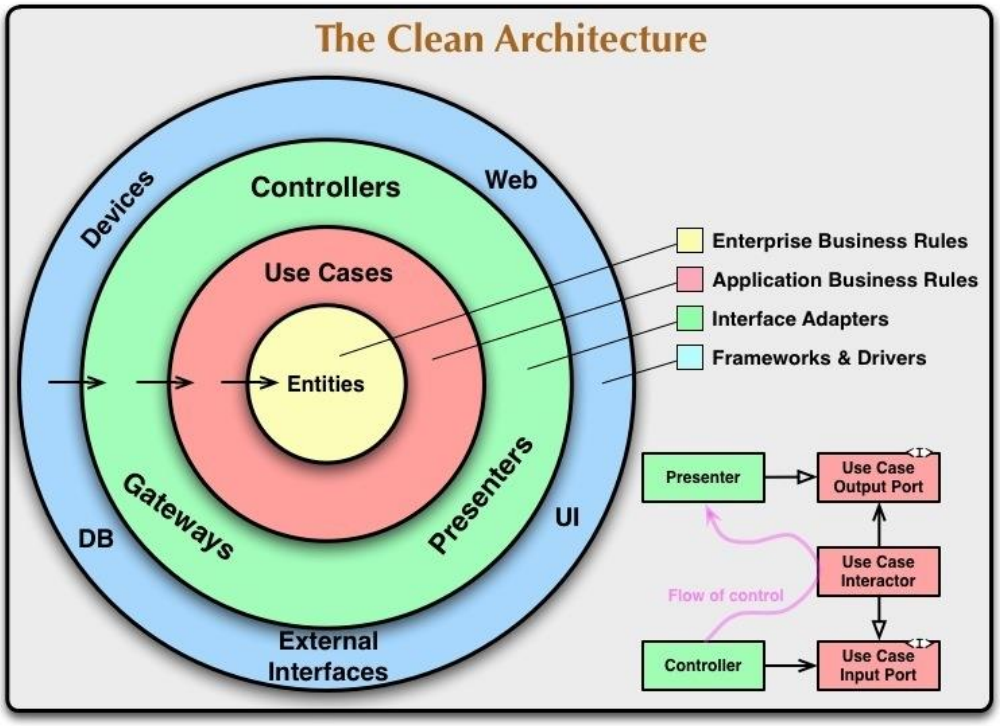

Рисунок 3.1 - Зображення Clean Architecture

Основна перевага чистої архітектури полягає в тому, що код стає більш організованим і його легше підтримувати. Оскільки він розробляється за

принципом спільної відповідальності, це дозволяє уникнути "спагеті-коду", де логіка змішується і переплітається. Такий підхід допомагає впорядкувати код і полегшує пошук та виправлення помилок [18].

Використання компонента бізнес-логіки (Business Logic Component, BLoC, (рис. 3.2) як шаблону проектування допомагає відокремити бізнес-логіку від користувацького інтерфейсу. Це означає, що всі рішення щодо логіки обробки даних, взаємодії з бекендом та зміни стану приймаються в окремому шарі, не завантажуючи код користувацького інтерфейсу. Такий підхід не тільки робить код більш читабельним і зрозумілим, але й спрощує процес тестування, оскільки бізнес-логіку можна тестувати окремо від користувацького інтерфейсу [19].

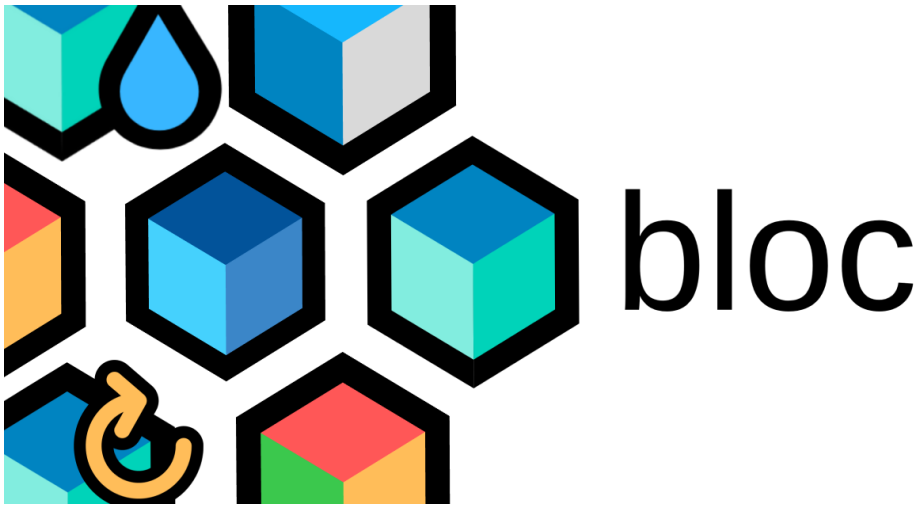

Рисунок 3.2 - Bloc pattern

Проект TopTop досягає високого рівня масштабованості та гнучкості завдяки використанню чистої архітектури та моделі BLoC. Це означає, що при додаванні нового функціоналу або внесенні змін не потрібно переписувати великі частини коду, а лише додавати або змінювати певні компоненти. Таким чином, додатки стають стабільнішими і простішими в обслуговуванні [20].

Таким чином, ретельно продумана архітектура TopTop є основою для успішної та ефективної розробки мобільних додатків. Вона забезпечує необхідну основу для розширення та адаптації додатків відповідно до мінливих потреб і вимог

ринку, а також допомагає забезпечити високу якість і надійність кінцевого продукту [21].

3.2 Розроблення інтерфейсу користувача

Інтерфейс користувача (UI) відіграє ключову роль у прийнятті та успіху мобільного додатку Розробка UI в TopTop (рис. 3.3) зосереджена на створенні простого, але привабливого та інтуїтивно зрозумілого дизайну Основними принципами розробки UI в TopTop є наступні:

Простота і чистота: Основною метою було зробити користувальницький інтерфейс максимально зрозумілим і зручним для навігації. Чітка структура, прості іконки та зрозумілі меню були використані для того, щоб полегшити користувачам пошук потрібних функцій та інформації.

Реагування на дії користувача: Важливим аспектом було забезпечення реагування на кожну дію користувача. Це означає, що кожна взаємодія з елементом інтерфейсу, наприклад, дотик або свайп, повинна супроводжуватися візуальним або звуковим зворотним зв'язком.

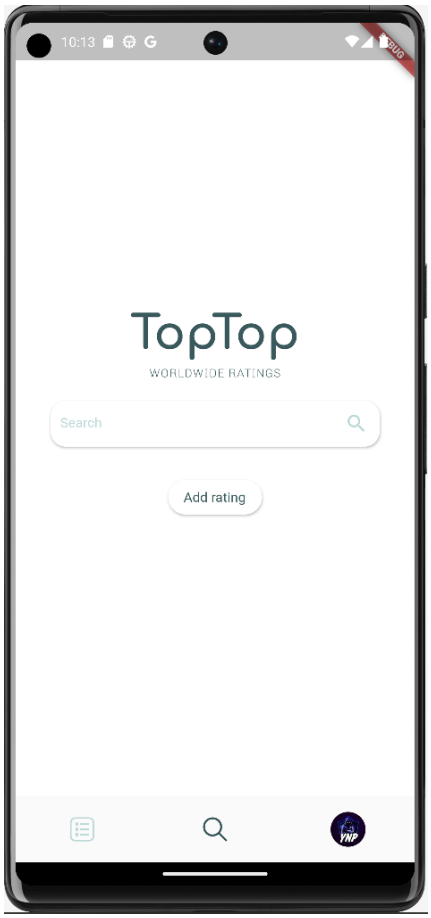

Рисунок 3.3 - Демонстрація головного екрану додатку

Адаптивність: Враховуючи різноманітність пристроїв, що використовують додаток, велику увагу було приділено адаптивності дизайну.

Великий акцент був зроблений на дизайні. Це означає, що інтерфейс повинен коректно відображатися і функціонувати на різних розмірах і орієнтаціях екрану, а завдяки гнучкій верстці і медіа-запитам Flutter ми змогли оптимізувати інтерфейс для різних пристроїв, від смартфонів до планшетів.

Анімація та переходи: Плавні анімації та переходи були інтегровані, щоб забезпечити більш динамічний та цікавий користувацький досвід. Це не тільки покращило естетичний вигляд додатку, але й зробило взаємодію з ним більш приємною та інтуїтивно зрозумілою.

Послідовність бренду: При створенні дизайну ТорТор було важливо дотримуватися фірмового стилю та палітри бренду. Це допомогло створити миттєво впізнаваний, професійний вигляд і відчуття, що є ключем до побудови довіри та лояльності користувачів.

Інтуїтивно зрозумілий користувацький досвід: Окрім візуального аспекту, багато уваги було приділено інтуїтивно зрозумілому та зручному користувацькому досвіду (UX). Це включає логічну організацію контенту, чітке розміщення елементів керування та легкий доступ до основних функцій. Особливий акцент було зроблено на тому, щоб нові користувачі могли легко орієнтуватися в додатку з самого початку, що має вирішальне значення для залучення та утримання аудиторії [22].

Розробка користувацького інтерфейсу ТорТор на Flutter дозволила нам ефективно реалізувати ці ключові принципи. Завдяки широкому спектру готових віджетів та можливостей кастомізації, команда розробників змогла розробити дизайн і створити унікальний і простий у використанні інтерфейс.

Під час доопрацювання користувальницького інтерфейсу ТорТор було зроблено кілька додаткових кроків для оптимізації користувацького досвіду

Використання адаптивних віджетів: Адаптивні віджети були використані для того, щоб забезпечити однакове відображення користувацького інтерфейсу на

різних пристроях з різними розмірами та орієнтацією екрану. Ці віджети автоматично адаптуються до розміру екрану, гарантуючи, що інтерфейс завжди буде привабливим і простим у використанні на будь-якому пристрої.

Забезпечення доступності Ми також приділили увагу питанням доступності. Це означає, що інтерфейс ТорТор розроблений з урахуванням потреб людей з різними обмеженими можливостями. Використання великих кнопок, чітких іконок, контрастних кольорів та підтримка зчитування з екрану роблять додаток доступним для широкого кола користувачів.

Тестування кінцевого користувача: Додаток був протестований реальними користувачами перед його остаточним запуском. Це не лише допомогло виявити та виправити недоліки в дизайні інтерфейсу, але й забезпечило цінний зворотній зв'язок щодо функціональності додатку та загального досвіду користування ним. Залучення користувачів на ранній стадії допомогло вдосконалити дизайн та забезпечити відповідність кінцевого продукту потребам та очікуванням користувачів [23].

Постійне вдосконалення: Після запуску додатку команда продовжила працювати над вдосконаленням інтерфейсу, враховуючи відгуки та пропозиції користувачів. Цей процес безперервного вдосконалення гарантує, що додаток залишається актуальним, простим у використанні та цікавим для користувачів.

В результаті цих зусиль користувальницький інтерфейс ТорТор не тільки відповідає технічним вимогам, але й створює позитивний, цікавий та емоційно насичений досвід для користувачів. Це не тільки приваблює нових користувачів, але й заохочує їх продовжувати користуватися ТорТор, що є важливим фактором успіху в сучасному цифровому світі [24].

3.3 Інтеграція сервісів

Інтеграція бекенд-сервісів є важливим кроком у розробці мобільних додатків ТорТор, оскільки вона впливає на багато аспектів програми, від обробки даних до функціональності та безпеки. Ось деякі ключові аспекти інтеграції бекенд-сервісів, на яких ми зосереджуємося:

Вибір правильної хмарної платформи: Вибір правильної хмарної платформи був дуже важливим. Після ретельного аналізу ми обрали Firebase від Google, оскільки вона пропонує широкий спектр послуг, які найкраще відповідали нашим потребам: Firebase пропонує не тільки базу даних в режимі реального часу, але й автентифікацію, аналітику, зберігання файлів та багато іншого. Розробка бази даних [25].

Розробка бази даних Структура бази даних була ретельно спланована, щоб забезпечити ефективне зберігання та обробку інформації про користувачів, рейтинги, голоси та відгуки. База даних Firebase NoSQL була обрана як гнучке та масштабоване рішення, яке може легко адаптуватися до збільшення обсягів даних та зміни вимог.

Аутентифікація користувачів: забезпечення безпечної та зручної аутентифікації було одним з наших пріоритетів. Ми інтегрували кілька методів автентифікації, включаючи електронну пошту та пароль, соціальні мережі та анонімний вхід, що дозволило користувачам обрати найбільш зручний метод.

Оптимізація обробки запитів та продуктивності: важливим аспектом була оптимізація реакції сервера на запити користувачів. Це передбачало мінімізацію затримок, оптимізацію запитів до бази даних та використання кешу, де це можливо, для покращення загальної продуктивності додатку.

Синхронізація даних: Оскільки TopTop вимагає актуальності даних в режимі реального часу, особливу увагу було приділено механізму синхронізації даних між клієнтським додатком і сервером. Це гарантує, що всі користувачі можуть бачити актуальну інформацію, де б вони не знаходилися.

Безпека даних: забезпечення безпеки даних користувачів є одним з наших головних пріоритетів. Численні заходи безпеки, такі як шифрування даних, безпечна передача даних і регулярні перевірки безпеки гарантують, що інформація захищена від несанкціонованого доступу та інших загроз.

Інтеграція API: Ми інтегрували різні зовнішні API для розширення функціональності TopTop. Це включає інтеграцію з соціальними мережами для

публікації рейтингів та опитувань, а також з аналітичними сервісами для відстеження взаємодії користувачів з додатком [26].

Масштабованість: Враховуючи потенційне зростання користувацької бази TopTop, велика увага була приділена масштабованості бекенду. Це означає, що система може впоратися зі збільшенням кількості користувачів і навантаження без шкоди для продуктивності.

В результаті цих зусиль бекенд TopTop був розроблений не тільки для задоволення поточних вимог додатку, але й для розміщення майбутніх розширень та оновлень. Це забезпечує стабільну основу для надійної та ефективної роботи додатку, сприяє високому рівню задоволеності користувачів і підтримує сталий розвиток TopTop у майбутньому.

Загалом, успішна інтеграція бекенд-сервісів TopTop не тільки забезпечила необхідну функціональність додатку, але й створила міцний фундамент для подальшого розвитку та розширення. З огляду на складність і важливість цього процесу, значний час і ресурси були витрачені на планування, розробку і тестування для забезпечення надійності та ефективності бекенду [27].

3.4 Реалізація функціоналу голосування

Основна функціональність додатку TopTop зосереджена на системі голосування та оцінювання (рис. 3.4). Ця функція вимагала ретельної розробки, щоб забезпечити як простоту використання для кінцевого користувача, так і надійність та точність обробки даних.

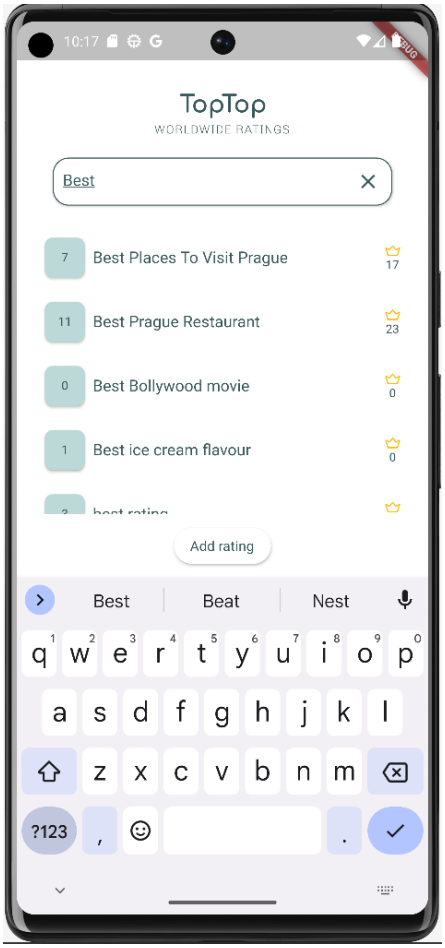

Рисунок 3.4 - Демонстрація екрану пошуку рейтингу

По-перше, було важливо, щоб процес створення опитування був інтуїтивно зрозумілим і простим. Ми розробили інтерфейс, який дозволяє користувачам легко створювати опитування, вводячи запитання та надаючи кілька варіантів відповідей. Користувачі також можуть налаштовувати параметри опитування, такі як тривалість опитування та видимість результатів.

Далі ми взялися за обробку та зберігання голосів, що передбачало розробку алгоритмів для ефективної обробки даних, введених користувачами, та оновлення результатів голосування в режимі реального часу. Важливо було забезпечити надійну та безпомилкову систему, щоб гарантувати, що всі голоси будуть точно підраховані.

Особливу увагу також було приділено захисту системи від можливих маніпуляцій та забезпеченню чесності процесу голосування. Впровадження заходів

безпеки, таких як аутентифікація користувачів та обмеження кількості голосів, запобігло можливим спробам втручання в результати.

Нарешті, результати голосування були представлені користувачам у зрозумілій та візуально привабливій формі. Графіки, діаграми та інші візуальні елементи дозволяють користувачам швидко оцінити результати та проаналізувати тенденції у відповідях.

Отже, реалізація функції голосування в ТорТор виявилася складним, але дуже важливим завданням. Потрібно було не тільки забезпечити надійність і точність підрахунку голосів, а й створити систему, яка б забезпечувала безперебійний і цікавий досвід голосування для користувачів. Ця система стала однією з ключових особливостей ТорТор і сприяє залученню та утриманню користувачів, надаючи їм можливість активно взаємодіяти з контентом додатку [28].

Крім того, успішна інтеграція цієї системи голосування відкрила шлях для подальших інновацій та розвитку додатку. Наприклад, у майбутньому можливе впровадження функції, яка дозволить користувачам створювати власні опитування з індивідуальними налаштуваннями та використовувати дані опитувань для отримання цінної інформації та тенденцій.

Завдяки планомірному підходу до розробки та впровадження функціоналу голосування, ТорТор стає не просто додатком для голосування, а повноцінною платформою для спілкування та обміну ідеями серед широкого кола користувачів.

3.5 Забезпечення взаємодії з користувачами

Створення успішного мобільного додатку – це не лише реалізація функціональності, але й забезпечення активної взаємодії з користувачами - цьому ТорТор приділив особливу увагу (рис. 3.5). Це пов'язано з тим, що активність користувачів є важливим фактором у підтримці та розвитку платформи.

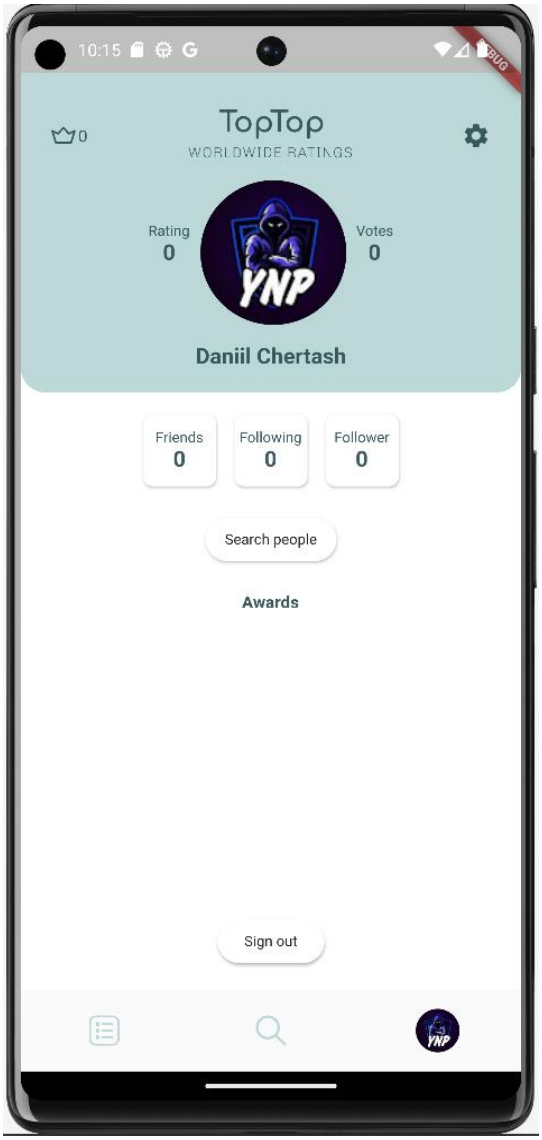

Рисунок 3.5 - Демонстрація екрану персонального профіля користувача

Залучення користувачів Наша стратегія залучення користувачів зосереджена на створенні цікавого контенту та заохоченні участі в голосуванні. Ми впровадили систему сповіщень для інформування користувачів про нові опитування, існуючі опитування та оновлення результатів. Це допомагає підтримувати інтерес та залученість користувачів.

Персоналізація користувацького досвіду Персоналізація користувацького досвіду TopTop є ще одним важливим аспектом. Аналізуючи поведінку та вподобання користувачів, ми надаємо індивідуальні рекомендації та персоналізовані опитування, щоб підвищити зацікавленість та залученість користувачів до додатку.

Соціальна взаємодія Оскільки в основі ТорТор лежить ідея спільноти та обміну ідеями, в ньому реалізовані функції, що полегшують соціальну взаємодію. Це включає можливість коментувати опитування, обговорювати результати і ділитися своїми голосами в соціальних мережах.

Зворотній зв'язок з користувачами Забезпечення зворотного зв'язку з користувачами є важливою частиною нашої стратегії. Ми створили просту, зручну систему зворотного зв'язку, яка дозволяє користувачам легко ділитися своїми думками та пропозиціями щодо покращення додатку.

Аналіз поведінки користувачів: Використовуючи аналітику, ми можемо зрозуміти, як користувачі взаємодіють з ТорТор, і визначити потенційні області для поліпшення. Аналізуючи такі дані, як частота використання додатку, опитування популярності та вподобання користувачів, ТорТор може бути адаптований до потреб користувачів.

Підтримка користувачів: Ми також зосереджені на наданні ефективної підтримки користувачів. Це включає в себе швидке реагування на запити про допомогу, вирішення технічних проблем і постійне оновлення FAQ з відповідями на найпоширеніші запитання. Надання якісної підтримки користувачів підвищує задоволеність і лояльність клієнтів.

Оновлення та вдосконалення: Нарешті, ТорТор постійно оновлюється і вдосконалюється на основі відгуків користувачів та аналізу поведінки. Це включає як покращення існуючих функцій, так і додавання нових, щоб зробити додаток більш привабливим та актуальним.

Як наслідок, активна взаємодія з користувачами є важливим фактором успіху ТорТор. Тому багато уваги було приділено не тільки технічній реалізації функцій, але й створенню корисного, цікавого та інтерактивного досвіду для користувачів.

3.6 Тестування та оптимізація

Тестування та оптимізація додатку ТорТор відіграли важливу роль у забезпеченні надійності та ефективності перед офіційним запуском. Цей етап

розробки був спрямований на виявлення та виправлення помилок і покращення загальної продуктивності додатку.

Процес тестування:

Тестування ТорТор охоплювало різні аспекти додатку, включаючи функціональність, користувацький інтерфейс, продуктивність, безпеку та взаємодію з бекендом.

Були проведені модульні тести, щоб перевірити кожен компонент додатку і переконатися, що він працює коректно.

Інтеграційне тестування допомогло забезпечити ефективну взаємодію різних частин програми.

Тестування користувацького інтерфейсу було важливим для того, щоб переконатися, що додаток є зручним та інтуїтивно зрозумілим.

Тестування продуктивності допомогло виявити та оптимізувати ділянки коду, які можуть сповільнювати роботу програми або використовувати надмірні ресурси.

Оптимізація:

Після тестування ми зосередилися на оптимізації ТорТор.

Було внесено зміни для підвищення ефективності запитів до сервера та зменшення часу завантаження даних.

Оптимізація зображень та мультимедійного контенту зменшила споживання даних та покращила швидкість відгуку додатку.

Реорганізація коду покращила читабельність і ефективність, зменшивши ризик майбутніх помилок і спростивши процес додавання нових функцій.

Безпека.

Безпека даних та операцій ТорТор була одним з головних пріоритетів. Захист від зовнішніх загроз і вразливостей був ретельно протестований. Це включало в себе

Використання сучасних методів шифрування для забезпечення безпеки даних користувачів.

Впровадження надійних механізмів автентифікації та авторизації для запобігання несанкціонованому доступу.

Проведення регулярних тестів на проникнення для виявлення та усунення потенційних слабких місць.

3.7 Запуск та моніторинг додатку

Після успішного завершення етапу розробки та тестування ми перейшли до фінальної фази - запуску та моніторингу додатку ТорТор. Цей етап є вирішальним, оскільки він визначає, як продукт буде сприйнятий на ринку і які дії необхідні для його підтримки та оптимізації.

Стратегія запуску:

Перед запуском була розроблена детальна маркетингова стратегія для забезпечення успішного просування ТорТор серед потенційних користувачів. Ця стратегія включала кампанію в соціальних мережах, залучення інфлюенсерів та промо-акції, щоб викликати інтерес та залучення.

Запуск додатку:

Додаток ТорТор був доступний на всіх основних платформах, як iOS, так і Android. Це забезпечило широке охоплення користувачів і доступ до додатку для різних груп користувачів. Під час запуску також була забезпечена підтримка клієнтів, щоб гарантувати, що будь-які питання або проблеми вирішувалися швидко.

Моніторинг та аналіз:

Після запуску ми активно відстежували роботу додатку за допомогою низки аналітичних інструментів, зокрема, чи правильно працює основна функціональність додатку - створення рейтингу та голосування (рис. 3.6). Це дозволило нам відстежувати ключові показники ефективності (KPI), такі як кількість завантажень, активність користувачів, час, проведений у додатку, та взаємодію з функціями. Аналіз цих даних надав цінну інформацію для подальшого вдосконалення та оптимізації додатку.

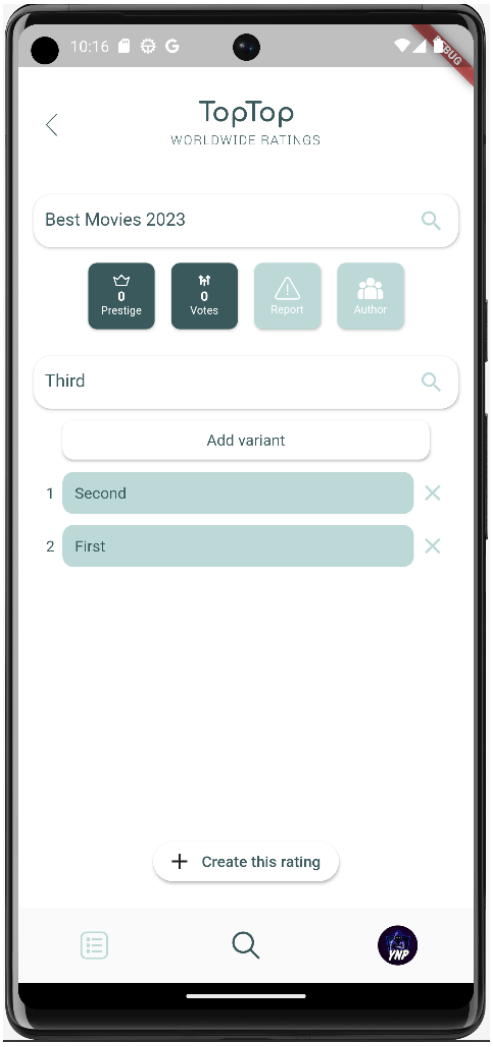

Рисунок 3.6 - Демонстрація можливості створення власних рейтингів

Відгуки користувачів

Отримання відгуків від користувачів було ще одним важливим аспектом процесу після запуску. Ми запропонували користувачам ділитися своїми думками та пропозиціями через різні канали зворотного зв'язку, включаючи додаток, соціальні мережі та електронну пошту, щоб ми могли швидко виявляти та вирішувати проблеми, а також збирати ідеї для нових функцій та вдосконалень.

Оновлення та вдосконалення

На основі аналізу та відгуків користувачів ми регулярно випускали оновлення TopTop. Вони включали як виправлення помилок, так і нові функції, щоб зробити додаток більш привабливим і корисним для користувачів. Наша команда також працювала над оптимізацією продуктивності та стабільності додатку, щоб забезпечити кращий користувацький досвід.

Загалом, запуск і моніторинг ТорТор дав нам можливість не тільки оцінити реакцію ринку на новий продукт, але й активно працювати над його вдосконаленням. Це був безперервний процес, який вимагав уваги до деталей, гнучкості та готовності швидко реагувати на зміни і запити користувачів. Постійний аналіз даних і зворотного зв'язку дозволив нам адаптуватися до ринкових тенденцій і потреб користувачів, що дало змогу ТорТор зберегти свою популярність і успіх на висококонкурентному ринку мобільних додатків.

Наш підхід до запуску та моніторингу ТорТор демонструє важливість гнучкості та здатності вносити швидкі зміни на основі фактичного використання продукту. Це не тільки допомагає утримати існуючу базу користувачів, але й сприяє залученню нових користувачів, надаючи їм покращений та оновлений досвід. Як результат, ТорТор - це не просто додаток, а динамічний продукт, що розвивається, який реагує на зростаючі потреби та інтереси своїх користувачів.

3.8 Оновлення та підтримка додатку

Після запуску додатку ТорТор наша команда продовжила працювати над оновленнями та підтримкою продукту. Це важливо для забезпечення довгострокової цінності та задоволеності користувачів [29].

Функціональні оновлення

Постійне оновлення та покращення функціональності ТорТор було головним пріоритетом. Це включало в себе

Додавання нових функцій, таких як нові типи опитувань та інтерактивні елементи для покращення користувацького досвіду.

Оптимізація існуючих функцій для підвищення продуктивності та зручності використання.

Покращення користувацького інтерфейсу, щоб зробити його більш інтуїтивно зрозумілим і цікавим.

Технічна підтримка та оновлення:

Надання постійної технічної підтримки для вирішення проблем або питань, з якими користувачі можуть зіткнутися під час використання додатку.

Регулярне оновлення додатку для виправлення помилок та забезпечення сумісності з останньою версією операційної системи.

Збір та аналіз відгуків

Постійно збирайте та аналізуйте відгуки користувачів, щоб зрозуміти, які частини додатку потребують вдосконалення або оновлення.

Цей аналіз допоможе вам у майбутньому розробляти оновлення та вдосконалення.

Розвиток спільноти:

Активно розвивайте спільноту ТорТор, організовуючи заходи, конкурси та ініціативи, які заохочують залучення та активність користувачів.

Постійне вдосконалення:

Постійне вдосконалення додатку на основі нових технологій та інновацій у сфері мобільних додатків, щоб ТорТор залишався конкурентоспроможним і відповідав сучасним тенденціям.

Нарешті, постійні оновлення та підтримка ТорТор допомагають гарантувати, що додатки не тільки відповідають поточним потребам користувачів, але й адаптуються до мінливих ринкових вимог і тенденцій. Це створює позитивний цикл взаємодії, де користувачі отримують покращені продукти, а ми отримуємо цінні відгуки та ідеї для подальшого розвитку. Наша мета - забезпечити, щоб ТорТор не тільки залишався актуальним і привабливим для користувачів, але й продовжував розвиватися як платформа для спілкування та самовираження [30].

Висновок до розділу 3

Створення архітектури додатку є важливим етапом у розробці мобільного додатку "ТОРТОР". На цьому етапі визначається структура програми, вибір паттернів проектування та архітектурних рішень. Це дозволяє побудувати додаток з урахуванням його масштабованості, надійності та зручності у розвитку.

Розробка інтерфейсу користувача є ключовим етапом у створенні привабливого та зручного у використанні додатку. На цьому етапі важливо врахувати вимоги до дизайну, взаємодії та навігації, щоб забезпечити позитивний досвід користувача.

Інтеграція різноманітних сервісів дозволяє забезпечити розширені можливості додатку та покращити його функціональність. На цьому етапі важливо правильно інтегрувати зовнішні сервіси, забезпечити їх взаємодію з додатком та забезпечити безпеку обміну даними.

Реалізація функціоналу голосування в додатку "ТОРТОР" є важливим етапом, оскільки ця функція може стати однією з ключових для користувачів. На цьому етапі важливо забезпечити зручність та безпеку голосування, а також правильну обробку результатів.

Забезпечення ефективної взаємодії з користувачами є важливим аспектом розробки додатку. На цьому етапі важливо враховувати вимоги до комунікації, відгуків користувачів та можливостей зворотного зв'язку для покращення якості обслуговування.

Тестування та оптимізація дозволяють перевірити роботу додатку на різних платформах та пристроях, виявити та виправити помилки, а також покращити його продуктивність та ефективність.

Після завершення розробки важливо правильно запустити додаток та забезпечити його стабільну роботу. Моніторинг додатку дозволяє вчасно виявляти та вирішувати проблеми, що можуть виникнути під час експлуатації.

ВИСНОВКИ

Flutter - це потужний крос-платформний інструмент розробки мобільних додатків, який дозволяє розробникам швидко та ефективно створювати високоякісні додатки для Android та iOS. Огляд фреймворку підкреслив його простоту використання та широкий спектр можливостей для розробників; архітектура Flutter демонструє його переваги з точки зору швидкості розробки та легкої масштабованості. Порівняння з іншими крос-платформними рішеннями підкреслює переваги Flutter у продуктивності та надійності.

Flutter відіграє важливу роль у розробці сучасних мобільних додатків, дозволяючи розробникам швидко виводити свої ідеї та пропозиції на ринок. Фреймворк прокладає шлях до створення високоякісних, високопродуктивних додатків, які відповідають вимогам сучасних користувачів. Завдяки великій кількості доступних бібліотек та інструментів, Flutter стає невід'ємною частиною робочого процесу для багатьох розробників, які хочуть досягти успіху у своїй галузі.

Концепція Flutter для надійної крос-платформної розробки виявилася дуже привабливою для розробників. Це пов'язано з тим, що фреймворк дозволяє швидко створювати додатки для різних платформ з мінімальними зусиллями.

Огляд фреймворку Flutter показує його силу та гнучкість: Flutter має багато компонентів і функцій, які роблять його чудовим користувацьким інтерфейсом.

Архітектура Flutter є однією з його найважливіших переваг. Вона забезпечує гнучку розробку, дозволяючи розробникам створювати додатки, які швидко та ефективно працюють на різних платформах.

Коли Flutter порівнюють з іншими крос-платформними рішеннями, підкреслюють його переваги, такі як швидкість розробки та висока продуктивність.

Роль Flutter у сучасній розробці мобільних додатків важлива тим, що він дозволяє розробникам створювати високоякісні, ефективні додатки, які відповідають потребам сучасних користувачів.

Керування станами є ключовою концепцією для створення динамічних і

чуйних додатків у Flutter. У цій главі ми розглянемо різні методи управління станами, включаючи локальне управління станами, витягування станів, використання пакетів провайдерів і модель блоків. Використання цих методів може допомогти вам ефективно керувати станом вашого додатку і підтримувати його адаптивність.

Обговорюються важливі аспекти розробки додатків Flutter, такі як оптимізація продуктивності, включаючи використання асинхронних транзакцій і стратегій кешування, а також стратегії тестування для підтримки якості програми.

Обговорюються стратегії тестування для забезпечення якості Flutter-додатків. Зокрема, підкреслюються підходи до тестування користувальницького інтерфейсу, функціонального тестування та інтеграційного тестування, які допомагають забезпечити стабільність і надійність роботи програми.

Представлені успішні приклади використання Flutter в корпоративних рішеннях, включаючи створення внутрішніх інструментів і додатків для співробітників, корпоративних платформ електронної комерції та систем управління взаємовідносинами з клієнтами.

Продемонстровано широкі можливості Flutter для створення високоякісних, ефективних і масштабованих додатків, які підтримують якість і надійність в корпоративному середовищі.

Створення архітектури додатку є критично важливим етапом у розробці мобільних додатків ТОРТОР. На цьому етапі визначається структура додатку, вибір патернів проектування та архітектурне рішення. Це гарантує, що додаток побудований з урахуванням масштабованості, надійності та простоти розробки.

Розробка користувацького інтерфейсу є важливим етапом у створенні привабливого і простого у використанні додатку. На цьому етапі важливо врахувати вимоги до дизайну, взаємодії та навігації, щоб забезпечити хороший користувацький досвід.

Інтеграція різних сервісів може забезпечити розширений функціонал і підвищити функціональність додатку. На цьому етапі важливо забезпечити

належну інтеграцію зовнішніх сервісів, гарантувати взаємодію з додатком і безпеку обміну даними. Реалізація функції голосування в додатку ТОРТОР є важливим кроком, оскільки це може бути однією з найважливіших функцій для користувачів. На цьому етапі важливо забезпечити зручність і безпеку голосування, а також коректну обробку результатів.

Забезпечення ефективної взаємодії з користувачем є важливим аспектом розробки додатку. На цьому етапі важливо врахувати вимоги до комунікації, відгуки користувачів і можливості зворотного зв'язку для покращення якості сервісу.

Тестування та оптимізація дозволяють перевірити працездатність додатку на різних платформах і пристроях, виявити і виправити помилки, підвищити продуктивність і ефективність.

Після завершення розробки важливо переконатися, що додаток коректно запускається і працює стабільно. Моніторинг додатків дозволяє своєчасно виявляти і вирішувати проблеми, які можуть виникнути під час роботи.

Після запуску важливо постійно оновлювати та підтримувати додаток, щоб він залишався актуальним і безпечним для користувачів.

ПЕРЕЛІК ПОСИЛАНЬ

1. Mobile App Legal and Compliance Issues [Electronic resource]. – URL: https://www.termsfeed.com/blog/legal-issues-mobile-apps/

2. Flutter State Management [Electronic resource]. – URL: https://flutter.dev/docs/development/data-and-backend/state-mgmt/intro

3. User Onboarding Best Practices [Electronic resource]. – URL: https://www.smashingmagazine.com/2018/08/best-practices-user-onboarding-mobile-apps/

4. Mobile App Usability Testing [Electronic resource]. – URL: https://www.nngroup.com/articles/mobile-app-usability/

5. Cross-Platform Mobile Development [Electronic resource]. – URL: https://www.oreilly.com/library/view/mobile-development-with/9781491989142/

6. Mobile App Accessibility Guidelines [Electronic resource]. – URL: https://www.w3.org/WAI/standards-guidelines/mobile/

7. Understanding Mobile User Experience [Electronic resource]. – URL: https://www.interaction-design.org/literature/topics/mobile-user-experience

8. Mobile App Security Best Practices [Electronic resource]. – URL: https://www.owasp.org/index.php/Mobile_Top_10_2016-Top_10

9. In-App Purchase Guidelines [Electronic resource]. – URL: https://developer.apple.com/in-app-purchase/

10. Google's Firebase Blog [Electronic resource]. – URL: https://firebase.googleblog.com/

11. User Interface Design Patterns [Electronic resource]. – URL: https://ui-patterns.com/

12. Mobile App Testing Automation [Electronic resource]. – URL: https://www.selenium.dev/documentation/en/mobile/

Mobile Application Development Lifecycle [Electronic resource]. – URL: https://www.sciencedirect.com/science/article/pii/S2351978917301912

13. Mobile App Marketing Insights [Electronic resource]. – URL: https://www.thinkwithgoogle.com/intl/en-154/marketing-strategies/app-and-mobile/mobile-app-marketing-insights/

14. Enhancing Mobile App User Engagement [Electronic resource]. – URL: https://www.forrester.com/report/Enhancing+Mobile+App+User+Engagement/-/E-RES117707

15. Cross-Platform UI Frameworks [Electronic resource]. – URL: https://www.codementor.io/blog/cross-platform-ui-frameworks-8n7doqb3z

16. Mobile App Development Frameworks Comparison [Electronic resource]. – URL: https://hackernoon.com/top-mobile-app-development-frameworks

17. Mobile App Development Frameworks Comparison [Electronic resource]. – URL: https://hackernoon.com/top-mobile-app-development-frameworks-in-2020-2d4d5fcebb10

18. Flutter vs React Native vs Xamarin [Electronic resource]. – URL: https://www.altexsoft.com/blog/engineering/the-good-and-the-bad-of-flutter-app-development/

19. Mobile App Development Trends 2023 [Electronic resource]. – URL: https://www.gartner.com/en/information-technology/insights/mobile-apps

20. Digital Marketing for Mobile Apps [Electronic resource]. – URL: https://digitalmarketinginstitute.com/blog/how-to-market-your-mobile-app

21. User Experience Research in Mobile Applications [Electronic resource]. – URL: https://www.springer.com/gp/book/9783319209459

22. Implementing Agile Methodology in Mobile App Development [Electronic resource]. – URL: https://www.agilenutshell.com/

23. Mobile App Analytics Platforms [Electronic resource]. – URL: https://mixpanel.com/focus/areas/mobile-app-analytics-platform/

Data-Driven Mobile App Design [Electronic resource]. – URL: https://hbr.org/2020/07/the-data-driven-approach-to-making-mobile-apps

24. Mobile App User Interface and User Experience Design [Electronic resource]. – URL: https://www.toptal.com/designers/mobile

25. Mobile App Legal Compliance [Electronic resource]. – URL: https://www.legalzoom.com/articles/developing-a-mobile-app-make-sure-its-compliant

26. Firebase Functions and Cloud Storage [Electronic resource]. – URL: https://firebase.google.com/docs/functions

27. Mobile App Privacy Policies [Electronic resource]. – URL: https://www.iubenda.com/en/help/11552-mobile-app-privacy-policy

28. Localization and Internationalization in Mobile Apps [Electronic resource]. – URL: https://phrase.com/blog/posts/guide-to-mobile-app-localization/

29. Performance Testing for Mobile Applications [Electronic resource]. – URL: https://www.loadrunner.com/tools/mobile

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК

Кваліфікаційна робота
на тему:

«Розробка мобільного додатку для інтернет-магазину на основі Flutter»

Виконав: Давиденко Владислав

Керівник: д.т.н., професор Віктор ВИШНІВСЬКИЙ

КИЇВ - 2024

Актуальність. Розробка мобільних додатків стала стратегічно важливим кроком для багатьох компаній, які бажають забезпечити зручний доступ до своїх товарів та послуг. Особливо актуальною стала ця тема у зв'язку зі стрімким розвитком онлайн-торгівлі та зміною у способах, якими користувачі взаємодіють з магазинами. Мобільні додатки на основі Flutter відкривають нові можливості для підприємств у залученні клієнтів та підвищенні їхнього рівня задоволеності від обслуговування .

Об'єкт дослідження – є процес розробки удосконалених методів підвищення надійності кросплатформної розробки з використанням Flutter.

Предмет дослідження – є розробка та оцінка сучасних методів підвищення надійності крос-платформної розробки з використанням Flutter.

Мета роботи – є підвищення ефективності розробки сучасних методів підвищення надійності кросплатформної розробки з використанням Flutter.

Наукові завдання:

1. Проаналізувати ключові особливості та переваги Flutter як кросплатформного фреймворку,включаючи його архітектуру, компоненти та інструменти розробки.
2. Дослідити кращі практики та методології в розробці мобільних додатків за допомогою Flutter, з акцентом на ефективність та продуктивність .
3. Розробити та проаналізувати практичний приклад мобільного додатку, створеного з використанням Flutter, для оцінки його функціональності, продуктивності та користувацького досвіду.

Огляд фреймворкуFlutter

Flutter бере свій початок у 2017 році коли Google вперше представив фреймворк на світовому ринку. З тих пір він постійно розвивається і додає нові функції. Однією з ключових особливостей Flutter є використання мови програмування Dart, яка спеціально пристосована для створення інтерактивних та високопродуктивних мобільних додатків що надає переваги та значно спрощує і прискорює процес розробки

Розробка Flutter фокусується на широкій підтримці різних платформ, що дозволяє розробникам створювати додатки, які працюють на різних пристроях, використовуючи єдину кодову базу. Це не тільки спрощує процес розробки, але й покращує узгодженість та якість кінцевого продукту - Flutter постійно оновлюється та вдосконалюється, що свідчить про активне заохочення відкритої спільноти розробників, яка сприяє розвитку цієї технології.

Завдяки своїй модульності та гнучкості Flutter пропонує широкий спектр можливостей для індивідуальної та інноваційної розробки. Розробники можуть створювати додатки, які не тільки виглядають нативно на кожній платформі, але й відповідають конкретним вимогам і побажанням своїх клієнтів.

З цих причин Flutter є однією з найперспективніших та найефективніших технологій у світі крос-платформної розробки на сьогоднішній день.

Архітектура Flutter та її вплив на кросплатформну розробку

Основні бібліотеки надають основні будівельні блоки для створення користувацьких інтерфейсів, включаючи віджети, макети та управління станом. Вони не залежать від платформи і працюють однаково на всіх пристроях.

Будівельні блоки користувацького інтерфейсу Flutter. Віджети можна повторно використовувати, поширювати, зберігати і зберігати для створення складних і динамічних інтерфейсів. Специфічні для платформи віджети, які надають оригінальні елементи користувацького інтерфейсу, такі як кнопки, текстові поля та панелі навігації. Вони адаптуються до мови дизайну платформи при використанні рушія Flutter для рендерингу; макет цих компонентів платформи Flutter показано на рисунку 1.1.

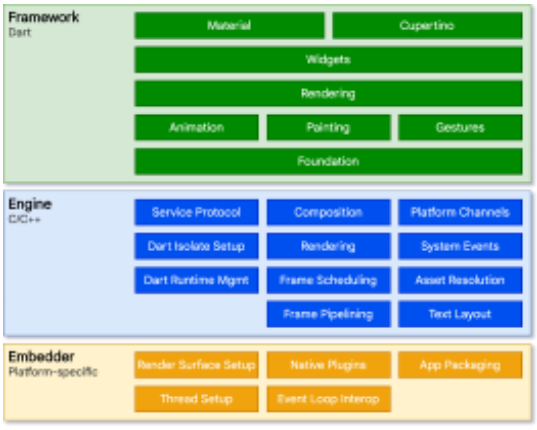

Рис 1.1 - Схема Flutter платформи

Переваги використання архітектури Flutter повторне використання коду і гаряче перезавантаження значно прискорюють процес розробки, дозволяючи створювати крос-платформні додатки швидше і ефективніше;

- завдяки структурованій архітектурі код легше розуміти, підтримувати та оновлювати, що призводить до більш стабільного процесу розробки;
- специфічні для платформи віджети та гаряче перезавантаження роблять ваш додаток нативним на кожній платформі, забезпечуючи послідовний та приємний користувацький досвід;
- Розуміючи і застосовуючи принципи архітектури Flutter, можна використовувати можливості цього фреймворку для ефективного створення високоякісних крос-платформних додатків і забезпечення безперебійної роботи користувачів на кожному пристрої.

Порівняння Flutter з іншими кросплатформними рішеннями

Швидка розробка з гарячим перезавантаженням, що дозволяє оновлювати програму в режимі реального часу без втрати її стану. Великий набір готових віджетів та широкий спектр можливостей кастомізації; Відносно нова і невелика екосистема в порівнянні з іншими фреймворками; Dart менш популярний, ніж JavaScript, тому розробникам може знадобитися вивчити нову мову.

React Native Розроблений Facebook, React Native – це фреймворк з відкритим вихідним кодом, який дозволяє розробникам створювати мобільні додатки за допомогою JavaScript та React. Він використовує нативні компоненти для рендерингу та надає додатку нативний вигляд і відчуття.

Плюси: Існує велика і зріла екосистема з численними бібліотеками та плагінами. Використовує JavaScript, популярну і широко використовувану мову програмування. Сильна підтримка спільноти та Facebook; спільна кодова база для React (веб-) та React Native (мобільних) проектів.

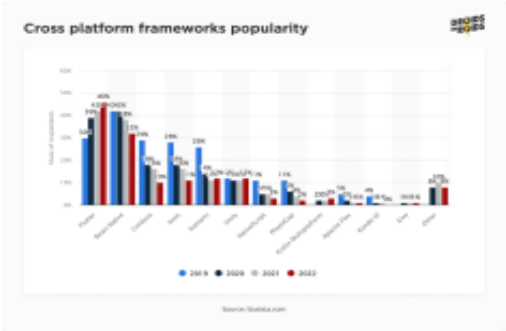

Рис. 1.2 - Статистика використання крос-платформних рішень

Ключові аспекти при розробці додатків на базі рішення Flutter

Оптимізація продуктивності є одним з ключових завдань при створенні додатків на Flutter. Один з способів покращити продуктивність – це ефективне використання асинхронних операцій. Працюючи з асинхронним кодом в Flutter, слід пам'ятати про основні інструменти для роботи з асинхронними операціями: «Future» та «Stream». «Future» використовується для представлення одноразового результату асинхронної операції, тоді як «Stream» дозволяє отримувати послідовні значення з асинхронного джерела (рис. 1.3). Для оптимізації зображень у Flutter можна використовувати такі підходи:

Зменшення розміру зображень за допомогою стиснення без втрати якості. Для цього можна використовувати спеціальні інструменти або бібліотеки, такі як flutter_image_compress або використовувати стиснуті формати зображень WebP (рис. 1.5). Змінна final вказує на те, що значення змінної може бути призначено лише один раз і не може бути змінено після цього. Це дозволяє покращити читабельність коду та запобігти непередбачуваним змінам значень змінних.

```dart
Image.network(
  'https://example.com/image.webp',
  fit: BoxFit.cover,
)
```

Рис. 1.5 - Оптимізація зображення за допомогою стиснутого формату

```dart
Future<void> fetchData() async {
  // Асинхронний запит до серверу
  // ...
}
Stream<int> countDown(int from, int to) async* {
  for (int i = from; i >= to; i--) {
    await Future.delayed(Duration(seconds: 1));
    yield i;
  }
}
```

Рис. 1.3 - Приклад використання асинхронних операцій

```dart
final Map<String, dynamic> _cache = {};
dynamic getData(String key) {
  if (_cache.containsKey(key)) {
    return _cache[key];
  } else {
    // Логіка отримання даних
    // ...
    _cache[key] = fetchedData;
    return fetchedData;
  }
}
```

Рис. 1.4 - Приклад мемоізації

```dart
final String appName = 'MyApp';
const double piValue = 3.14;
```

Рис. 1.6 - Використання змінних

Створенняархітектуридодатку

У випадку з TopTop головним завданням було створити архітектуру, яка б не тільки забезпечувала гнучкість і масштабованість, а й полегшувала майбутні зміни і додавання нових функцій. З огляду на це, було вирішено використовувати комбінацію чистої архітектури (рис. 1.7) і моделі BLoC для управління кейсами. У цьому підході логіка додатку розділена на окремі шари, і кожен шар відповідає за свою частину логіки. Використання компонента бізнес-логіки (Business Logic Component, BLoC, (рис. 1.8) як шаблону проектування допомагає відокремити бізнес-логіку від користувацького інтерфейсу. Це означає, що всі рішення щодо логіки обробки даних, взаємодії з бекендом та зміни стану приймаються в окремому шарі, не завантажуючи код користувацького інтерфейсу. Такий підхід не тільки робить код більш читабельним і зрозумілим, але й спрощує процес тестування, оскільки бізнес-логіку можна тестувати окремо від користувацького інтерфейсу.

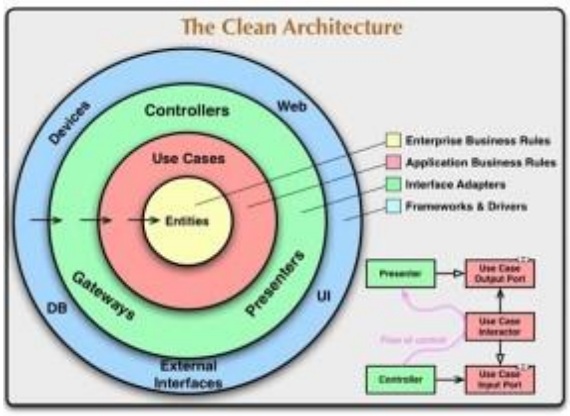

Рис. 1.7 - Зображення Clean Architecture

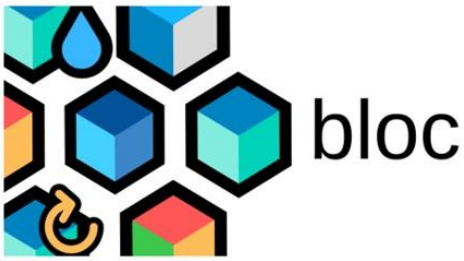

Рис. 1.8 – Bloc pattern

Розробленняінтерфейсукористувача

Інтерфейс користувача (UI) відіграє ключову роль у прийнятті та успіху мобільного додатку Розробка UI в TopTop (Малюнок 1.9) зосереджена на створенні простого, але привабливого та інтуїтивно зрозумілого дизайну Основними принципами розробки UI в TopTop є наступні:

Простота і чистота: Основною метою було зробити користувальницький інтерфейс максимально зрозумілим і зручним для навігації. Чітка структура, прості іконки та зрозумілі меню були використані для того, щоб полегшити користувачам пошук потрібних функцій та інформації.

Реагування на дії користувача: Важливим аспектом було забезпечення реагування на кожну дію користувача. Це означає, що кожна взаємодія з елементом інтерфейсу, наприклад, дотик або свайп, повинна супроводжуватися візуальним або звуковим зворотним зв'язком.

Адаптивність: Враховуючи різноманітність пристроїв, що використовують додаток, велику увагу було приділено адаптивності дизайну.

Великий акцент був зроблений на дизайні. Це означає, що інтерфейс повинен коректно відображатися і функціонувати на різних розмірах і орієнтаціях екрану, а завдяки гнучкій верстці і медіа-запитам Flutter ми змогли оптимізувати інтерфейс для різних пристроїв, від смартфонів до планшетів.

Анімація та переходи: Плавні анімації та переходи були інтегровані, щоб забезпечити більш динамічний та цікавий користувацький досвід. Це не тільки покращило естетичний вигляд додатку, але й зробило взаємодію з ним більш приємною та інтуїтивно зрозумілою.

Послідовність бренду: При створенні дизайну TopTop було важливо дотримуватися фірмового стилю та палітри бренду. Це допомогло створити миттєво впізнаваний, професійний вигляд і відчуття, що є ключем до побудови довіри та лояльності користувачів.

Рис. 1.9 - Демонстрація головного екрану додатку

Реалізація функціоналу голосування

Основна функціональність додатку TopTop зосереджена на системі голосування та оцінювання (рис. 3.4). Ця функція вимагала ретельної розробки, щоб забезпечити як простоту використання для кінцевого користувача, так і надійність та точність обробки даних. Особливу увагу також було приділено захисту системи від можливих маніпуляцій та забезпеченню чесності процесу голосування. Впровадження заходів безпеки, таких як аутентифікація користувачів та обмеження кількості голосів, запобігло можливим спробам втручання в результати.

Нарешті, результати голосування були представлені користувачам у зрозумілій та візуально привабливій формі. Графіки, діаграми та інші візуальні елементи дозволяють користувачам швидко оцінити результати та проаналізувати тенденції у відповідях. Отже, реалізація функції голосування в TopTop виявилася складним, але дуже важливим завданням. Потрібно було не тільки забезпечити надійність і точність підрахунку голосів, а й створити систему, яка б забезпечувала безперебійний і цікавий досвід голосування для користувачів. Ця система стала однією з ключових особливостей TopTop і сприяє залученню та утриманню користувачів, надаючи їм можливість активно взаємодіяти з контентом додатку.

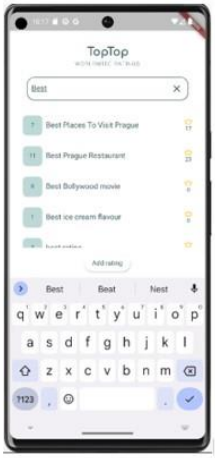

Рис. 1.10 - Демонстрація екрану пошуку рейтингу

Забезпечення взаємодії з користувачами

Створення успішного мобільного додатку – це не лише реалізація функціональності, але й забезпечення активної взаємодії з користувачами - цьому TopTop приділив особливу увагу (рис. 3.5). Це пов'язано з тим, що активність користувачів є важливим фактором у підтримці та розвитку платформи. Залучення користувачів Наша стратегія залучення користувачів зосереджена на створенні цікавого контенту та заохоченні участі в голосуванні. Персоналізація користувацького досвіду Персоналізація користувацького досвіду TopTop є ще одним важливим аспектом. Аналіз поведінки користувачів: Використовуючи аналітику, ми можемо зрозуміти, як користувачі взаємодіють з TopTop, і визначити потенційні області для поліпшення. Оновлення та вдосконалення: Нарешті, TopTop постійно оновлюється і вдосконалюється на основі відгуків користувачів та аналізу поведінки. Це включає як покращення існуючих функцій, так і додавання нових, щоб зробити додаток більш привабливим та актуальним.

Як наслідок, активна взаємодія з користувачами є важливим фактором успіху TopTop. Тому багато уваги було приділено не тільки технічній реалізації функцій, але й створенню корисного, цікавого та інтерактивного досвіду для користувачів.

Рис. 1.11 - Демонстрація екрану персонального профіля користувача

Запуск та моніторинг додатку

11

Після успішного завершення етапу розробки та тестування ми перейшли до фінальної фази - запуску та моніторингу додатку TopTop. Цей етап є вирішальним, оскільки він визначає, як продукт буде сприйнятий на ринку і які дії необхідні для його підтримки та оптимізації.

Стратегія запуску:

Перед запуском була розроблена детальна маркетингова стратегія для забезпечення успішного просування TopTop серед потенційних користувачів. Ця стратегія включала кампанію в соціальних мережах, залучення інфлюенсерів та промо-акції, щоб викликати інтерес та залучення.

Запуск додатку:

Додаток TopTop був доступний на всіх основних платформах, як iOS, так і Android. Це забезпечило широке охоплення користувачів і доступ до додатку для різних груп користувачів. Під час запуску також була забезпечена підтримка клієнтів, щоб гарантувати, що будь-які питання або проблеми вирішувалися швидко.

Моніторинг та аналіз:

Після запуску ми активно відстежували роботу додатку за допомогою низки аналітичних інструментів, зокрема, чи правильно працює основна функціональність додатку - створення рейтингу та голосування (рис. 3.6). Це дозволило нам відстежувати ключові показники ефективності (KPI), такі як кількість завантажень, активність користувачів, час, проведений у додатку, та взаємодію з функціями. Аналіз цих даних надав цінну інформацію для подальшого вдосконалення та оптимізації додатку.

Рис. 1.12 - Демонстрація можливості створення власних рейтингів

Відгуки користувачів

12

Отримання відгуків від користувачів було ще одним важливим аспектом процесу після запуску. Ми запропонували користувачам ділитися своїми думками та пропозиціями через різні канали зворотного зв'язку, включаючи додаток, соціальні мережі та електронну пошту, щоб ми могли швидко виявляти та вирішувати проблеми, а також збирати ідеї для нових функцій та вдосконалень.

Оновлення та вдосконалення

На основі аналізу та відгуків користувачів ми регулярно випускали оновлення TopTop. Вони включали як виправлення помилок, так і нові функції, щоб зробити додаток більш привабливим і корисним для користувачів. Наша команда також працювала над оптимізацією продуктивності та стабільності додатку, щоб забезпечити кращий користувацький досвід.

Загалом, запуск і моніторинг TopTop дав нам можливість не тільки оцінити реакцію ринку на новий продукт, але й активно працювати над його вдосконаленням. Це був безперервний процес, який вимагав уваги до деталей, гнучкості та готовності швидко реагувати на зміни і запити користувачів. Постійний аналіз даних і зворотного зв'язку дозволив нам адаптуватися до ринкових тенденцій і потреб користувачів, що дало змогу TopTop зберегти свою популярність і успіх на висококонкурентному ринку мобільних додатків.

ВИСНОВКИ

- Перечислено ключові аспекти при розробці додатків на Flutter, такі як оптимізація продуктивності, що включає в себе використання асинхронних операцій та стратегій кешування, а також стратегії тестування для підтримки якості додатку.
- Розглянуто стратегії тестування для забезпечення якості додатків на Flutter. Зокрема, висвітлюються підходи до тестування інтерфейсу користувача, функціонального та інтеграційного тестування, які допомагають забезпечити стабільність та надійність додатків.
- Наведені приклади успішного використання Flutter у корпоративних рішеннях, таких як створення внутрішніх інструментів та додатків для співробітників, корпоративних платформ електронної комерції та систем управління відносинами з клієнтами.
- Продемонстровано широкі можливості Flutter у створенні високоякісних, ефективних та масштабованих додатків, що підтримують якість та надійність у корпоративному середовищі.
- Створення архітектури додатку є важливим етапом у розробці мобільного додатку "ТОРТОР". На цьому етапі визначається структура програми, вибір паттернів проектування та архітектурних рішень. Це дозволяє побудувати додаток з урахуванням його масштабованості , надійності та зручності у розвитку.
- Розробка інтерфейсу користувача є ключовим етапом у створенні привабливого та зручного у використанні додатку. На цьому етапі важливо врахувати вимоги до дизайну, взаємодії та навігації, щоб забезпечити позитивний досвід користувача.
- Інтеграція різноманітних сервісів дозволяє забезпечити розширені можливості додатку та покращити його функціональність. На цьому етапі важливо правильно інтегрувати зовнішні сервіси, забезпечити їх взаємодію з додатком та забезпечити безпеку обміну даними.
- Реалізація функціоналу голосування в додатку "ТОРТОР"є важливим етапом, оскільки ця функція може стати однією з ключових для користувачів. На цьому етапі важливо забезпечити зручність та безпеку голосування, а також правильну обробку результатів.
- Забезпечення ефективної взаємодії з користувачами є важливим аспектом розробки додатку. На цьому етапі важливо враховувати вимоги до комунікації, відгуків користувачів та можливостей зворотного зв'язку для покращення якості обслуговування.
- Тестування та оптимізація дозволяють перевірити роботу додатку на різних платформах та пристроях, виявити та виправити помилки, а також покращити його продуктивність та ефективність .
- Після завершення розробки важливо правильно запустити додаток та забезпечити його стабільну роботу. Моніторинг додатку дозволяє вчасно виявляти та вирішувати проблеми, що можуть виникнути під час експлуатації .
- Після запуску важливо забезпечити постійне оновлення та підтримку додатку, щоб він залишався актуальним та безпечним для користувачів.