МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДОНЕЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ВАСИЛЯ СТУСА

ПРИСІЧ АРТЕМ ВОЛОДИМИРОВИЧ

Допускається до захисту:
завідувач кафедри
інформаційних технологій,
д-р техн. наук, доцент
_____ Тетяна НЕСКОРОДЄВА
«_____» _____ 2022р.

РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АНАЛІЗУ ВИКОРИСТАННЯ МОБІЛЬНОГО ДОДАТКУ

Спеціальність 122 «Комп'ютерні науки»

Кваліфікаційна (магістерська) робота

(відповідно до стандарту спеціальності та ОП)

Науковий керівник:
Римар П. В., старший викладач
кафедри інформаційних технологій
Науковий консультант:
Нескородєва Т.В.,
д-р техн. наук, доцент

_____
(підпис)

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

Вінниця 2022

АНОТАЦІЯ

Присіч А.В. Розробка інформаційної системи для аналізу використання мобільного додатку. Спеціальність 122 «Комп’ютерні науки», освітня програма «Сучасні інформаційні технології та програмування», Донецький національний університет імені Василя Стуса, Вінниця, 2022.

У кваліфікаційній (магістерській) роботі досліджена потреба створення даної системи та дослідження методів аналізу даних. Розроблена система для аналізу використання мобільного додатку. Створено функціональну та у майбутньому швидко розширювану систему для розробників та клієнтів що значно спростить та полегшить їх роботу у майбутньому. Встановлені актуальні напрямки та перспективи розширення функціональності додатку.

Ключові слова: Android, IOS, аналіз даних, плагін, система, мова програмування Dart, Flutter SDK, Firebase, Android Studio, час на сторінці.

ANNOTATION

Prysich A.V. Development of an information system for analyzing the use of a mobile application. Specialty 122 "Computer Science". Vasyl Stus Donetsk National University, Vinnytsia, 2022.

In the qualification (master's) thesis, the need for creating this system and researching data analysis methods was investigated. A system for analyzing the use of a mobile application has been developed. A functional and future-expandable system has been created for developers and customers which will greatly simplify and facilitate their work in the future. Current directions and prospects for expanding the functionality of the application have been established.

Keywords: Android, IOS, data analysis, plugin, system, Dart programming language, Flutter SDK, Firebase, Android Studio, time on page

ЗМІСТ

ВСТУП

В умовах сучастних реалій компанії намагаються скоротити витрати бізнесу, пришвидшити вихід на ринок та посилити присутність в онлайні за допомогою додатків. Зі швидких мобільних рішень Surf рекомендує Flutter.

Кросплатформні програми — давня мрія будь-якого бізнесу, оскільки окремі (нативні) програми для iOS та Android набагато дорожчі у розробці та підтримці.

У 2018 році команда Surf серйозно взялася за розвиток кроссплатформного спрямування. До найкращих практик ми рухаємося всі 9 років, але саме в сучасних реаліях найкращім рішенням для рітейлу, фінтеху та ecommerce став Flutter.

Кросплатформа буває різною

Говорячи простою мовою, кросплатформа - це набір інструментів (фреймворк), який дає змогу розробити додатки, що одразу будуть підходити для iOS та Android. На перший погляд звучить чудово. Насправді свої можливості та обмеження є у кожного фреймворку. Їх зараз є велика кількість: React Native, Xamarin, PhoneGap, Titanium, Ionic, Flutter є найпопулярнішими.

Краща среед усіх кросплатформа повинна відповідати двом вимогам:

бути економічною з погляду розробки;

забезпечувати найбільш якісний досвід для користувачів.

Економ. Переваги полягають не тільки у створенні одного додатка замість двох.

Добре було б зловити мінімум проблем при адаптації на кожну з платформ. З цим все гаразд у більшості фреймворків, якщо розглядати їх у контексті простих додатків.

Проблема починається, коли програма обростає функціями та неминуче ускладнюється (це відбувається з будь-яким продуктом на стадії розробки).

Користувальницький досвід. Важливо, щоб користувачі сприймали кросплатформну програму як нативну — тобто начебто вона написана спеціально для цієї платформи. Як мінімум, там повинні бути плавні анімації,

характерні для даної ОС елементи інтерфейсу, робота з жестами. І ось із цим погано у всіх, крім Flutter.

Часто кросплатформа виглядає як більш просунута мобільна версія сайту в обгортці програми. Анімації гальмують, використовуються незвичні елементи інтерфейсу з непередбачуваною поведінкою.

Все це погіршує досвід користувача, результатом є нижчі оцінки та утримання (швидкість, з якою користувачі повертаються до вашої програми) у магазині. У Flutter такої проблеми немає, рівень анімації є перевагою, і ви можете без проблем використовувати знайомі та зручні нативні елементи.

Актуальність теми дослідження. Мобільні програми - найгарячіша тенденція на сучасному IT-ринку. Сучасні мобільні програми пропонують кілька вирішальних переваг. Швидкість та зручність для користувача.

Для прискорення процесу розробки та зручності використання створити систему для аналізу використання мобільного додатку що значно пришвидшить розробку через імплементацію вже готового функціоналу. Також кожен плагін має тенденцію до розвитку та розширення через спільноту розробників що їм користуються . Через велику кількість Flutter розробників ймовірність розширення функціоналу та підтримка плагіну з подальшим доданням до нього різноманітного коду , що б довести його до максимальної зручності дуже велика.

Мета Розробка системи для аналізу використання мобільного додатку через створення плагіну з можливістю для подальшого розширення та вдосконалення роботи. Основною функцією плагіну є збирання даних про користування користувачем мобільним додатком, а саме скільки часу користувач провів на кожному із екранів додатку , скільки часу всього часу користувались додатком, та відображення даних на окремому екрані що імпліментовано у сам плагін. Розробник може оперувати цими даними на свій розсуд , створити звіт або надіслати їх на сервер для подальшого детального аналізу.

Завдання дослідження.

1. Вивчити та з'ясувати Dart, Flutter SDK, Git, GitLab, GitLab CI/CD Android Studio, Android Emulator, Flutter Plugin, Swift, iOS Simulator, Xcode,
2. З'ясувати як розроблюються плаліни для Flutter із взаємодією за платформами Android та IOS.
3. **Об’єкт дослідження**. Збір та аналіз даних через фреймворк Flutter для ОС Android та IOS.

Предмет дослідження. Популярний фреймворк, для створення плагіну одразу під платформи Android та IOS.

Для досягнення поставленої мети необхідно вирішити такі дослідницькі завдання.**:**

4. Розглянути поняття плагіну.
5. Обрати інструменти для створення системи аналізу (плагіну).
6. Обрати спосіб збору та аналізу даних.
7. Створити систему у вигляді плагіну для аналізу мобільного додатку

Робота складається з 3 розділів, 15 пунктів, 7 рисунку та списку літератури, що містить 13 джерел, загальний обсяг роботи 61 сторінки.

РОЗДІЛ 1
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБКИ МУЛЬТИПЛАТФОРМНИХ ДОДАТКІВ

1.1 Плагіни на мові Flutter

Слово «плагін» складено з двох слів: plug in — дослівно «підключати». Плагін — модуль, який можливо підключити до програми і надати їй додаткові розширенні можливості. Для прикладу можемо взяти програму, яка дає користувачеві малювати зображення. Припустимо, програміст написав плагін, який розширює можливості програми — наприклад, дозволяє застосувати до зображення якісь особливі ефекти. В такому випадку ми повинні підключити плагін до програми, після чого його можна викликати прямо з програми.[1]

Нативні програми (написані рідними мовами ОС - Swift і Kotlin, для iOS і Android відповідно) майже не мають мінусів, але вимагають значно більших витрат на розробку та підтримку. Потрібно буде створювати бізнес-логіку, верстку та інтерфейс з поглядом на особливості кожної платформи. Все це позначиться на бюджеті проекту.

Кросплатформна програма на Flutter має переваги нативної розробки і дозволяє перевикористовувати велику частину коду між різними платформами, що дає змогу заощадити, за нашими підрахунками, до 50% бюджету.

Шлях до розробки кросплатформних додатків останнім часом був забитий, головною причиною чого була його приваблива природа єдиної кодової бази, де розробникам доводиться писати код один раз, і цей код можна використовувати для розробки додатків для різних платформ. Я знаю! Це весело, як,

1. Єдина кодова база
2. Скорочений час розробки
3. Знижена вартість розробки

і, звичайно, є недоліки, безсумнівно.

Flutter — це найновіший крок у розробці кросплатформних додатків, розроблений компанією Google, і з кожним днем він набирає популярності,

залишаючись сильним у порівнянні з іншими конкурентами завдяки певним факторам:

1. Час розробки
2. Продуктивність під час виконання
3. Гарна підтримка інструментів

Однак, окрім різноманітних переваг, встановлених ним, є кілька недоліків, одним із яких є обмежений набір бібліотек.

Незважаючи на те, що Flutter підтримується Google, але будучи відносно новим фреймворком, йому не вистачає підтримки фреймворків сторонніх розробників, і будь-яке міжплатформне середовище залежить від нативних функцій платформи, таких як камера, Bluetooth, місцезнаходження тощо, до яких можна отримати доступ за допомогою плагінів.

Плагіни є оболонкою рідного коду, як-от android (java або kotlin) та iOS (swift або Objective C). Плагіни написані в коді конкретної платформи для доступу до функцій конкретної платформи.

Flutter підтримує використання пакетів і плагінів, наданих іншими розробниками для його екосистеми. Звідси ви можете отримати доступ до пакетів і плагінів Flutter.

Як все працює під капотом. API для конкретної платформи Flutter залежить від техніки передачі повідомлень, це відбувається в два етапи. По-перше, Flutter використовує канал платформи як засіб зв’язку між програмою Flutter і хостом, причому хост є частиною програми для iOS або Android. По-друге, хост отримує повідомлення через канал платформи, а потім викликає специфічні для платформи API, використовуючи рідну мову програмування платформи, і асинхронно надсилає зворотний зв’язок до програми Flutter.

Зокрема, на стороні клієнта MethodChannel використовується для надсилання повідомлень, а на стороні хоста MethodChannel на Android і FlutterMethodChannel на iOS використовуються для отримання та надсилання повідомлень. Усе це робиться для того, щоб забезпечити використання найменшої кількості шаблонного коду.

Команда розробників FlutterDevs для створення високоякісних і функціонально багатих програм. Наймiть розробника flutter для вашого крос-платформного проекту мобільного додатку Flutter на погодинну або повну ставку відповідно до ваших вимог! Ви можете зв’язатися з нами на Facebook, GitHub і Twitter для будь-яких запитів, пов’язаних із flutter.

1.2 Що таке крос-плафторма

Крос-платформенне це термін, використовуваний для позначення частини програмного забезпечення, яке є сумісним з більш ніж однією системою. Наприклад, популярний медіа-плеєр VLC сумісний з трьома основними настільних операційних систем: Microsoft, Mac OS і Linux. Крос-платформна підтримка може також поширюватися на мобільні пристрої, з великою кількістю додатків, доступних як на Apple App Store і тому Магазин Google Play .

Залежно від того, як кодуються конкретне програмне забезпечення, воно може зажадати від розробника повторного коду та перепакування програмного забезпечення повністю. Деякі структури дозволяють розробникам забезпечити підтримку платформ на їх програмне забезпечення більш гладко.

Є багато причин, щоб гарантувати, що ви використовуєте крос-платформену частина програмного забезпечення. Наприклад, один з найбільш важливих міркувань при використанні частини програмного забезпечення є сумісність файлів. Якщо ви використовуєте формат файлу, який доступний тільки на одній операційній системі, то ви могли б розглянути питання про перетворення його в формат з читачем, який працює за кількома системами.

Перехресна сумісність є великою темою, коли справа доходить до ігор. Ще в розпал консольних воєн з 1980-х до початку 2010-х років, крос-платформної сумісності була велика проблема. Основні виробники ігор, таких як Sony, Microsoft і Nintendo все мали титули, отримані і фінансуються. Ці ігри були розроблені часто їх в будинку студій або ігрові студії, які вони мали стратегічне партнерство с.

Тому багато ігор були «ексклюзиви», доступні тільки купити і грати на одному пристрої. Коли споживачі вирішують, яка консоль купити, вони повинні

були б розглянути, які ігри доступні для конкретних консолей. Halo завжди буде на Xbox, Uncharted буде на Playstation, і Mapio буде на Nintendo консолей.

У той час як Nintendo, як правило прилип до пропонуючи ексклюзив, Microsoft і Sony почали пропонувати більш крос-платформної підтримки. Через зростання комп'ютерних ігор в останнє десятиліття, багато великих ексклюзивні титули, почали пробиратися до ПК і консолі. Через більш високу стелю продуктивності на робочих столах, розробники також почали проектувати свої ігри з геймерами у вигляді і зниженні продуктивності при переході на консолі.

Крім того, здатність грати багато різних назв з одного пристрою, один з найбільших міркувань для тих, хто наполягає на сумісності між платформами є крос-платформної гри, часто звані крос-гри. Це відноситься до функції, яка дозволяє людям грати в ту ж гру з різними пристроями, щоб грати разом з іншими користувачами режимах. Наприклад, люди, які відіграють Fortnite на Playstation 5 буде мати можливість грати проти і в тандемі з гравцями користувачів за допомогою Windows PC.

Як гри були отримувати менше ексклюзивними, поштовх для кросу-гри зростає протягом останніх кількох років. Хоча є ще деякі незгодні, як Nintendo, більшість виробників і розробники ігор почали відкриватися до його реалізації. Люди в одних і тих же груп друзів, які володіють різними платформами, але як і раніше хочуть грати один з одним користь сильно від гри з цими функціями.

Перехресна гра також значно збільшує доступну користувачів системи конкретної гри, що добре для обох гравців і розробників. Активне, велика спільнота скорочує час очікування, збільшує діапазон рівнів, доступних для гри, і робить його більш обнадійливим для нових гравців, щоб почати гру.[2]

Мобільні програми стали незмінним супутником нашого життя. З їхньою допомогою ми можемо не тільки розважатися та спрощувати своє життя, здійснювати покупки або замовляти ті чи інші послуги онлайн, але й просувати свій бізнес, збільшувати клієнтську базу, а отже, і множити прибуток. І якщо в необхідності створення програми для своєї справи сумнівів ні в кого виникнути не може, то з вибором типу mobile-app можуть з'явитися деякі складнощі.

Всі сучасні програми для мобільних пристроїв можна розділити на нативні та кросплатформні, і кожна з цих двох груп має як свої сильні сторони, так і свої недоліки.

Нативними є програми, розроблені безпосередньо під певну платформу відповідною мовою програмування. Так, під час створення програми під Android використовується Java, а IOS-додатків – Objective-c чи Swift. Під час створення таких проектів фахівці враховують усі особливості платформ, особливу увагу приділяючи UI/UX дизайну, вимогам/рекомендаціям розробників операційних систем, а також останнім тенденціям у mobile-індустрії. Один фахівець не зможе повною мірою володіти всіма зазначеними вище мовами, тому для розробки одного нативного продукту під різні платформи необхідно підключати різних розробників, а це додаткові витрати, та й час розробки буде значним. Але при цьому програми будуть «заточені» під певну платформу, отримають доступ до внутрішніх ресурсів та функцій пристрою та працюватимуть максимально ефективно.

Незважаючи на чималий список переваг нативних розробок, клієнти не завжди хочуть витрачати час та гроші на їх розробку, підключаючи до процесу створення кількох майстрів. Оптимальним варіантом у таких випадках є кроссплатформенна розробка, що дозволяє створювати програми під будь-які платформи з використанням стандартних веб-технологій. При цьому розробкою може займатися одна людина, яка має необхідні знання та досвід роботи з HTML5, JavaScript і CSS3. Кросплатформні розробки можуть бути скомпільовані у файл.apk для Android та в.ipa для IOS. Таким чином, на основі однієї розробки можна отримати дві програми під популярні операційні системи, витративши на це менше часу та грошей. Однак подібні розробки мають свої недоліки, тому до кожного конкретного випадку вкрай бажано підходити індивідуально і вибирати найбільш підходящий варіант - нативна або кросплатформова розробка.

Більшість серйозних програм має свою клієнтську частину, яку часто називають frontend, і серверну – backend. Фронтенд відповідає за те, що ви побачите на екрані свого мобільного пристрою, тобто все візуальне подання

програми, включаючи дизайн, розміри та розташування вікон, меню, кнопок, стрілок та будь-яких інших елементів. Також фронтенд відповідає за реакцію програми на ті чи інші дії користувача, спрямовані на перехід у різні розділи АППА, виклик нових меню тощо.

Бекенд є серверною частиною програми і розташований на віддаленому сервері, який може знаходитися де-завгодно і керуватися за допомогою найрізноманітніших програмних засобів Взаємозв'язок між клієнтською та серверною частиною здійснюється завдяки API (інтерфейсу для програмування додатків). Іншими словами, API – якийсь посередник між frontend та backend, який передає запити від клієнтської частини до сервера, повертаючи необхідні користувачеві дані.

Клієнтська частина програми вкрай важлива, оскільки саме з нею матиме справу сам користувач і від зручності frontend залежатиме його загальне уявлення про роботу АППА. Її можна розробляти як вручну, але для цього необхідно добре розумітися на HTML5, CSS3 і java-script, так і за допомогою так званих фреймворків. У першому випадку часто використовується середовище розробки Apache Cordova, яке також широко відоме під назвою PhoneGap. Використовуючи це середовище, можна створювати програми для будь-яких платформ, застосовуючи web-технології, які Cordova перетворює на зрозумілий для конкретної платформи код. Cordova відкриває фактично необмежені можливості для web-розробників, яким зовсім не обов'язково вивчати Objective-C або Swift, Java або Kotlin для створення програм під певні операційні системи.

У той час, як Cordova не має обмежень у UI та логіці, фреймворки пропонують готові шаблонні рішення. З одного боку, це суттєво прискорює та спрощує процес розробки, оскільки фахівець може використовувати вже готові кнопки, списки, поля введення, картки та інші UI-елементи. З іншого боку, фахівець може використовувати для розробки лише інструменти та елементи, які доступні у вибраному фреймворку. Найпопулярнішим з них вважається Ionic, що дозволяє створювати кросплатформні програми на будь-який смак. У цьому фреймворку є велика вбудована колекція стандартних елементів, які візуально імітують

нативні програми, але їх дизайн у разі потреби можна змінити. При цьому розробник може підключати безліч додаткових плагінів, що розширюють можливості ionic framework, причому створений на цьому фреймворку проект можна запускати прямо у вікні браузера і оцінювати, як виглядатиме і працюватиме створювана програма без необхідності використання емулятора або установки на смартфон.

У той час, як клієнтською частиною займаються дизайнери та розробники, які мають знання в HTML, CSS, JS та фреймворках, бекендом займаються програмісти іншого профілю. Для налаштування серверів можуть використовуватися різні мови програмування та інструменти, головне, належним чином налаштувати їхню роботу та взаємозв'язок із клієнтською частиною. Тут необхідно використати відповідні системиуправління БД (базами даних). Це може бути традиційний MySQL, Redis, PostgreSQL або будь-яка інша база даних (наприклад, MongoDB), яка підходить для реалізації конкретного проекту і в якій бекенд-розробник добре розуміється. Для створення серверної частини програми, розробники можуть використовувати PHP, NodeJS, C#, Ruby, Python, Java та інші мови програмування.

Фахівці mobile-студії KitApp підходять до питання розробки frontend та backend частин комплексно та максимально відповідально. Наші розробники створять для вас кросплатформовий додаток будь-якої складності та спрямованості максимально швидко та якісно! Зв'яжіться з нами, і наші фахівці оперативно проконсультують вас з усіх питань!

ринку мобільних додатків вже більше десяти років, проте він і досі бурхливо розвивається. Попит з боку компаній постійно зростає і він все ще помітно перевищує пропозицію, що призводить до постійного подорожчання розробки. Одне з рішень у здешевленні цього процесу — кроссплатформенна розробка, коли той самий код використовується на всіх платформах.[3]

1.3 Чому саме Flutter

Мультиплатформність - те, чим Flutter підкуповує насамперед. Менеджери всіх часів дивувалися: навіщо писати те саме під різні платформи

(iOS, Android), а потім ще й для Web і, можливо, для десктопу? Тому з давніх-давен світ був наповнений різними засобами розробки, що допомагають об'єднувати розробку для всіх платформ в одному проекті — на одній кодовій базі і навіть одній мові програмування. Серед них були навіть залишаються FireMonkey, Qt, Xamarin, React Native і той самий Flutter, який багато в чому випереджає своїх попередників: на ньому можна писати непогані програми одночасно і під iOS, і під Android. Так, і крім мобілок були спроби затягнути в наш проект та Web на Flutter, але безуспішні.

Open source. Говорячи про популярність Flutter, варто відзначити, що він відкритий і безкоштовний: якщо раптом Google покине проект, вихідний код залишиться доступним для всіх — і ви зможете його підтримувати разом із ком'юніті.

Декларативний підхід. Найбільшої популярності Flutter знайшов серед розробників завдяки декларативному підходу у реалізації інтерфейсів, і цей підхід значно прискорює розробку.

Сумісність з Native. Останньою перевагою Flutter можна виділити можливість затягнути в проект код на Native. Тобто, наприклад, специфічне визначення геолокації, яка працює по регіонах, ви не знайдете з коробки середовища, оскільки воно є тільки на iOS і не було сенсу затягувати його у Flutter. Але не варто засмучуватися - Flutter дозволяє впроваджувати код, написаний на Swift, у вигляді плагіна і використовувати цю можливість, специфічну для iOS і тільки для iOS. Така ж можливість є і в інших платформах при роботі з Flutter, тому тим, хто вважає, що "Flutter не вміє все" можна відповісти: "А що не вміє - навчимо на Native".[4]

Dart так собі мова. Тут усі чекатимуть якесь обґрунтування, але я просто зібрав відгуки смакових уподобань у розробників, які не побажали залишатися на Flutter. Перенавантажений зайвими конструкціями синтаксис, крапки з комою і так далі - просто виснаження Java. А Android-розробники начебто на Kotlin з Java побігли від хорошого життя. А чому, власне, KMM може випередити Flutter? Ось, наприклад, погляньмо на Dart: коли в нього завезли null safety? Правильно,

лише навесні 2021 року, адже це базовий механізм статичної типізації. Скільки років змінюється Dart – і тільки зараз у ньому з'явилося те, що було доступне у перших версіях Swift та Kotlin сім років тому. Зате є подібність coroutines, що чудово для екс-бекенд-мови.

Спільнота платформи. Звичайно, чудово, що Flutter можна подивитися у вихідниках, як і побачити, скільки issues зараз відкрито в офіційному репозиторії: їхня кількість тільки зростає - це говорить про те, що команда поки не справляється з потоком помилок. Наприклад, тільки зовсім нещодавно зробили, щоб анімації не гальмували на iOS.

Фреймворки поки що теж сируваті: при найменшому відхиленні програми від звичайного документообігу у бік специфіки заліза писати вам і так і так доведеться на Native. Наприклад, з чим зіткнулася моя команда: камера (сканер кодів – це основний функціонал, дуже критичний компонент для нашої програми) гальмує та глючить із підсвічуванням. Коли ми починали робити цей компонент, Flutter ще не було готового рішення (зараз воно існує, але за якість не ручаюся). Ще для збереження стану програми використовувався сторонній компонент Hive і в якийсь момент оновлення бібліотеки стан програми обнулився: виявилося, що в компоненті вже відомі кілька критичних багів, і чомусь їх ніхто не намагається виправити. В іншому, після того як у Flutter, нарешті, зафіксували проблеми з анімацією, здається, можна «жити», якщо ваш додаток містить просту функціональність.

Власний UI, характерний для обох мобільних платформ. Раніше я згадував, що Flutter має свій власний рендеринг — це означає, що поведінка візуальних компонентів (курсора в полі введення, контекстного меню, анімації, навігації екранів тощо) відрізняється від прийнятого на кожній із платформ. Все це може трохи спантеличити користувача вашої програми.

Інфраструктура. У пошуках інфраструктурних рішень як нативний розробник iOS я натикався найчастіше на Stack Overflow, присвячений проблемам з Flutter: збою роботи CocoaPods або fastlane при складанні проекту.

Якщо ви захочете завести автотести, у вас можуть виникнути перешкоди. В Ozon прийнято використовувати Appium: коли намагалися завести його через спеціальний драйвер, то при тестуванні програми він постійно відвалювався. Якщо у вас є якийсь позитивний приклад автотестування, поділіться в коментарях, який інструмент використовували і з чим стикалися.

І все-таки кому потрібен Flutter? Сама технологія "гібридної" розробки з'явилася не сьогодні. Ідея кроссплатформенного рішення, яке дозволить написати код один раз і запустити на двох платформах одночасно, існувала давно, проте мало хто вирішувався на її реалізацію. Офіційний реліз Flutter у 2018 році сколихнув індустрію мобільної розробки.

Приклади компаній, які вже використовують Flutter, можна знайти, зокрема, на сайті фреймворку:

1. BMW - додаток My BMW для керування та перевірки стану автомобіля;
2. eBay - сервіс eBay Motors для продажу автомобілів;
3. Alibaba - додаток Alibaba Xianyu для реалізації вживаних товарів;
4. Яндекс - Таксометр;
5. KFC – аналітична BPM-система для менеджерів та співробітників ресторанів.

Створення кросплатформових додатків «два в одному» стабільно цікаве бізнесу:

по-перше, як швидке рішення,

по-друге, як можливість заощадити бюджет.

Переваги:

Швидкість. Ви реалізуєте програму одночасно для двох платформ, перевикористовуючи частину коду та скорочуючи time-to-market (TTM) - термін виходу на ринок.

1. **Економія.** Flutter у середньому заощаджує від 5 до 20% часу розробки, хоча для кожного проекту цей показник буде унікальним, залежно від складності програми, його UI та функцій.

2. **Усереднена «формула»:** 1 Flutter-розробник найчастіше може виконувати завдання 2 нативних розробників (iOS, Android) у ті самі терміни. У свою чергу, якщо в додатку багато нативних функцій, зручно поєднувати нативну розробку (наприклад, для бібліотек) і Flutter (для створення єдиної логіки та UI).
3. **Одинаковий UI (інтерфейс користувача) Android-і iOS-платформах**.Двигун Skia від Google допомагає правильно намалювати той чи інший елемент для кожної версії, а в результаті ви отримуєте однаковість у додатках.
4. **Менше ризик помилок у бізнес-логіці оскільки iOS- та Android-версії мають єдину основу.**
5. **Можливість компіляції програми під desktop-платформи у подальшій перспективі.**

Обмеження.

Високий рівень складності

Якщо у вашому додатку є складні обчислення, велика кількість товарів, потрібен доступ до апаратних можливостей пристрою (камера, файли) – фреймворк потрібно вибирати разом із командою, яка займеться розробкою програми.

Дефіцит кадрів

У галузі поки що мало досвідчених кросплатформових розробників, що ускладнює підбір IT-команди, хоча ком'юніті постійно зростає.

Особливості роботи з графікою

У деяких гібридних додатках можуть «підгальмовувати» інтерфейс та графіка, однак, безпосередньо для Flutter ця проблема нетипова.

Підтримка старих версій

Кросплатформні програми можуть вимагати додаткових витрат, якщо потрібна підтримка старих версій iOS і Android.

Швидкість доставки фіч

Допустимо, ви оцінили авторизацію по відбитку пальця (fingerprint) в умовні 10 годин. Якщо в операційній системі відбудуться будь-які зміни, в нативному проекті термін розробки найчастіше залишиться незмінним — 10 годин, тоді як у кросплатформній реалізація може тривати 10-15 годин. При цьому практика показує, що комьюніті досить оперативно знаходить способи скорочення цього терміну.

1.5. Головне для аналізу мобільного додатку

Грамотна аналітика дає інформацію для прийняття рішень щодо подальших кроків. Але як зрозуміти, які метрики важливі для мобільного додатка? Чи є якийсь універсальний набір або потрібно підбирати ключові показники в залежності від сфери діяльності? У цій статті ми розповімо про 10 важливих метриках для мобільного софта, з яких ви зможете підібрати оптимальні для своєї сфери діяльності.

Як зрозуміти, що продукт цікавий цільової аудиторії? Як визначити, що він росте і розвивається? Для цього необхідно відстежувати ряд важливих метрик і на основі отриманих даних приймати рішення щодо подальших дій.

Retention Rate

Retention Rate – показник утримання клієнтів. Це показує, скільки користувачів повернулися в мобільний додаток. Різні експерти по-різному думають про коефіцієнти, залежно від інформації, яку вони хочуть отримати.

Найпростіший і найпоширеніший варіант — підрахувати відсоток користувачів, які повертаються у вашу програму через якийсь час (зазвичай вважається один місяць).

Наприклад, порівняння певного сегмента клієнтів, які використовували вашу програму в серпні, з тим же сегментом клієнтів, які взаємодіяли з вашим додатком у липні, покаже вам, скільки людей залишилося у вашій компанії.

У липні додаток завантажили 2000 людей. У серпні програмою продовжували користуватися 500 людей. RR = (500/2000) x 100% = 25%. Індикатор легко розраховується, але надає цінну інформацію. Якщо ваш

коефіцієнт утримання низький, вам слід шукати причини, через які ваші користувачі не повертаються.

Але що вважати високим показником? Експерти давно сперечаються про це питання, але так і не дійшли єдиного сенсу. Але маю думку. Все залежить від сфери діяльності та специфіки вашого бізнесу, але що вище, то краще.

Churn Rate

Показником, протилежним коефіцієнту утримання, є коефіцієнт відтоку клієнтів (показник відтоку). Показує відсоток клієнтів, які перестали використовувати вашу програму.

Щоб розрахувати значення, спочатку визначте рівень утримання, а потім використовуйте формулу:

Daily Sessions per DAU

При розрахунку DAU (щодня активних користувачів) кожен користувач враховується лише один раз. Але в деяких компаніях (наприклад, в соціальних мережах) може бути задана планка: користувачі повинні заходити в додаток кілька разів за день. Тому розраховують середню кількість сесій на одного споживача.

Stickiness

Stickiness – липкість, завязиваемость або лояльність. Коефіцієнт показує, як часто аудиторія повертається в додаток. Для розрахунку необхідно знати метрики "Активні користувачі за день" (DAU) та "Активні користувачі за місяць" (MAU).Щоб зрозуміти лояльність вашої аудиторії, потрібно знайти співвідношення щоденних активних користувачів до щомісячних активних користувачів.

Чим вище підсумковий відсоток, тим частіше користуються вашим додатком. Нормальна ситуація, коли поступово Stickiness збільшується. Приходять нові користувачі, вони стають лояльними і рекомендують програму свого оточення.

Якщо коефіцієнт почав знижуватися, отже, ваш продукт чимось перестав влаштовувати аудиторію. Можливо, він більше не вирішує проблему так, як це бачать споживачі. Іншими словами, програма не відповідає product market fit.

Як і у випадку з попередніми метриками, нормативного значення у Stickiness немає. Але експерти сходяться на думці, що треба прагнути до 100%.

Cost per acquisition (CPA)

Ця метрика потрібна для визначення ефективності рекламних кампаній. Не всі канали збуту дають однакову ефективність. Багато в чому це залежить від сфери діяльності та правильності настройки реклами. Наприклад, якийсь додаток отримує 90% нових користувачів з контекстної реклами, а в інше з цього каналу приходить тільки 20%.

І як зрозуміти, які канали збуту ефективні, а які ні? Це ніби наперед визначають аналітики, але для швидкої оцінки досить розрахувати Cost per acquisition (CPA):

Якщо з контекстної реклами ви отримуєте 20% нових користувачів, а з орієнтування 80%, є сенс відмовитися від першого каналу збуту і зосередити зусилля і доступні ресурси на другому. Це рішення дозволить залучати більше нової аудиторії без збільшення витрат на рекламу.

Lifetime Value (LTV)

Lifetime Value (LTV) – довічна цінність клієнта, що показує загальну суму доходу від користувача за весь час, поки він був клієнтом (користувачем додатки).

Цей показник дозволяє зрозуміти, яку цінність можна отримати з одного середньостатистичного користувача продукту за весь час. Під цінністю зазвичай розуміють прибуток або дохід.

Щоб розрахувати LTV, необхідно знати:

- Як часто користувачі здійснюють цільові дії.
- Чому дорівнює вартість цільового дії в грошовому еквіваленті.
- Протягом якого періоду користувачі здійснюють цільові дії. Враховують постійних клієнтів і тих, які пішли відразу, так як підраховується середнє значення [5]

1.6. Переваги нативних платформ.

Моментальний доступ до нових функцій «рідної» платформи.

Будь-які оновлення Google або Apple можна надати клієнтам одразу, буквально наступного дня після виходу чергової версії ОС. На Flutter та інших кросплатформових фреймворках може знадобитися більше часу для того, щоб розпочати роботу з новими функціями.

Наприклад:

- темна тема бета-версії iOS-13 на Flutter вийшла трохи пізніше за основний реліз;
- Віртуальна реальність VR у Flutter на момент написання статті представлена тільки на базовому рівні: можна вивести відео або зображення, а ось прогулятися по 3D-локації навряд чи вийде.

Доступна взаємодія зі специфічними функціями пристрою.

Наприклад:

- Робота з високонавантаженими процесами;
- Обробка відео;
- Гіроскоп, компас, модуль розпізнавання відбитка пальця;

Функції шифрування даних у банківських технологіях.

При цьому нативні модулі іноді можна підключати до гібридних програм, припустимо, відеочат — на кросплатформі, а SDK для прискорення роботи відеочату на C++. Наприклад, за допомогою бібліотеки dart:ffi можна інтегрувати взаємодію з кодом C++, а для простих навантажувальних елементів можна використовувати ізоляти, які дозволяють виконувати високонавантажені дії в новому процесі, уникаючи проблем з навантаженням в основному ізоляті.

Нативне середовище дозволяє досягти максимальної продуктивності.

Нативний код використовує менше оперативної пам'яті, слабше навантажує процесор під час роботи з досить важкою анімацією, оскільки відсутні різні перетворення показу зображення, які є у Flutter.

Запит на швидкість розробки з'являється у бізнесу під час змін на ринку. Приклад - тренд на загальну цифровізацію в ритейлі та фудтеху під час пандемії.

1.4. Недоліки нативних платформ

Нативна технологія може коштувати дорожче.

Один Flutter-розробник часто може виконувати завдання двох нативних фахівців (iOS, Android), як зазначено вище. Однак, для реалізації окремих нативних функцій, для яких немає готових плагінів, розробнику потрібно буде знати і iOS, і Android, щоб адаптувати нативні плагіни до Flutter або створити їх з нуля.

Окремі випробування для кожної платформи.

Як і при нативній розробці, потрібно передбачити тестування програми для Android та iOS. При цьому можливі відмінності в логіці фічі в різних версіях.

Для анімацій переважна кросплатформова технологія.

При створенні різних складних анімацій може знадобитися свій підхід до кожної платформи, що збільшить витрати праці. У свою чергу, кросплатформовий Flutter не просто надає розробнику прошарок між кодом і системними віджетами, але й допомагає в їхньому малюванні, забезпечуючи високу швидкість дій і чуйність інтерфейсу.

1.5. Мобільні додатки (Android, iOS).

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

Спочатку мобільний додаток використовувався для швидкої перевірки електронної пошти, але його високий попит призвів до його використання в інших сферах, таких як мобільні ігри, GPS, спілкування, перегляд відео та користування Інтернетом.

Загалом додатки переслідують як мінімум дві чіткі цілі: не нудьгувати власникам і полегшити життя, іншими словами, марні та корисні мобільні додатки.

Потреба в мобільних додатках навіть не вписується в дискусію, тому що поширення доступних мобільних пристроїв, зручних мультисенсорних екранів, безкоштовного, іноді безкоштовного, доступу до Інтернету, легкого обміну інформацією – все це значно збільшило потенціал Клієнтська аудиторія для будь-якого бізнесу.

Велика кількість мобільних пристроїв продається вже з набором встановлених мобільних програм, таких як веб-браузери, клієнти електронної пошти, календарі (наприклад, Google Calendar), програми для покупки та прослуховування музики тощо. Користувачі можуть видалити деякі попередньо встановлені програми зі свого мобільного пристрою за допомогою звичайного процесу видалення, звільняючи більше місця для зберігання інших (бажаних) програм.

Програми, які не встановлюються відразу на мобільний пристрій, можна завантажити та встановити через платформу розповсюдження. Такі платформи називаються магазинами додатків. Вони почали з’являтися в 2008 році і зазвичай управляються компаніями-власниками мобільних операційних систем: Apple App Store, Google Play, Windows Phone Store і Black Berry App World. Однак існують також різні незалежні магазини програм, такі як Cydia, Get Jar або F-Droid. Велику кількість мобільних додатків можна безкоштовно встановити та використовувати, але вони є платні. Усі програми зазвичай завантажуються безпосередньо з платформи на цільовий пристрій, але іноді їх можна завантажити на ноутбук або настільний комп’ютер. Зазвичай 20..30% вартості платної програми йде компанії-посереднику (наприклад, iTunes), а решта – виробнику. Таким чином, один і той самий додаток може мати різні тарифи залежно від мобільної платформи.

З мобільними додатками все набагато простіше, ніж з веб-додатками. Вони такі ж, як комп'ютерні програми, тільки для смартфонів і планшетів, і

завантажити їх можна не тільки в фірмових магазинах Apple і Android, але і з будь-якого безкоштовного джерела. Існує багато технологій, які можна використовувати для створення мобільних додатків, але вони діляться на дві великі категорії:

1. Рідні жителі. Ключ до розуміння того, як працює цей метод, — у самій назві. По суті, це рідна мова пристрою. Отже, рідна програма – це програма, створена за допомогою певної мови програмування для певної платформи. Програми написані рідною мовою мобільної платформи - Java та Kotlin для Android та Swift та objective-C для iOS/iPadOS. Легко отримуйте доступ до технічних частин вашого смартфона, таких як мікрофон і камера. Вони гарантують хороший, знайомий досвід користувача.

2. Мультиплатформенність. Ці програми також іноді називають кросплатформними. Універсальні додатки, що не залежать від платформи, розробляються з використанням нативних і веб-технологій. Якщо порівнювати їх із нативними додатками, то гібриди будуть дешевшими, а тим більш часті – зручнішими. Вони дозволяють заощадити гроші замість того, щоб платити за розробку як для iOS, так і для Android, але в той же час вони не підходять для дуже складних проектів і забезпечують поганий користувацький досвід. Зазвичай це робиться на таких платформах розробки, як React Native, Flutter, Ionic тощо.

Переваги мобільних додатків:

Комплексна взаємодія з користувачами. На відміну від мобільної версії сайту, у випадку з мобільними додатками є можливість відправляти push-повідомлення. Наприклад, якщо ви опублікуєте новий вміст на своєму сайті, користувачі дізнаються про нього, лише коли відвідають сайт. З мобільним додатком просто відправляєте усім активним користувачам, повідомлення, обмеження по символам, як в SMS-компаніях не має;

Локальні сповіщення працюють майже так само, за винятком того, що вони встановлені на самому пристрої. Якщо користувач виконує якусь дію в додатку, яка очікує його реакції через певний час, програма нагадає йому про це. [6]

Також з'являються ширші можливості для зворотного зв'язку - користувачі можуть залишати коментарі, відгуки в магазині додатків, особисто в соціальних мережах через додаток - вони будуть автоматично транслюватися на всі ресурси;

Кращий інтерфейс. Всі елементи управління: кнопки, текстові поля, посилання повинні легко натискатися пальцем на мобільних пристроях, а не курсором мишки. Екрани пристроїв відрізняються за розміром і пікселями. Роздільна здатність екрану мобільного телефону майже збігається з роздільною здатністю ноутбука, але в той же час фізичний розмір менший. Тому невідомо, як буде виглядати сайт на тому чи іншому пристрої, чи зручно це, чи навіть реально ним користуватися. Мобільні операційні системи для цієї ситуації мають власну, спеціально навчену логіку, щоб розрізняти пристрій, який вони використовують, щоб протидіяти наслідкам натискання пальцями маленьких кнопок. Але ця логіка працює по-різному на різних платформах і пристроях;

Якість інтерфейсу також відображається в навігації. Кожна мобільна операційна система має власну логіку перемикання між робочими екранами програми. В Android це кнопка «Назад», в IOS — свайп від лівого краю екрана. Користувачі кожної операційної системи звикли до однакової поведінки в кожній програмі. У кожного сайту свій спосіб навігації, при переході на інший сайт потрібно шукати такі кнопки, як «ОК», «Назад», «Скасувати», яких може взагалі не бути;

Дуже особисте. Мобільні телефони дуже добре знають своїх власників, і використання цієї інформації для покращення рівня обслуговування є однією з ключових причин швидкого зростання мобільних додатків.

Мобільний додаток завжди здатний запам'ятати всі дані користувача та змінити інтерфейс відповідно до його потреб. Якщо користувач уже ввів деякі дані (наприклад, свою домашню адресу в програмі доставки), то йому не потрібно буде вводити цю адресу знову. Навіть на різних пристроях із увімкненою синхронізацією користувачі бачитимуть програми, заповнені особистими даними. IOS запам'ятовує основні проміжки часу дій власника пристрою і в такому порядку іконки мобільних додатків на екрані;

Те ж саме стосується пристроїв - в залежності від типу програми, дозволів підтягуються дані з камери, акселерометра, компаса, барометра і т.д.; Локалізація - наприклад, якщо програма повинна відобразити список ресторанів, вона, швидше за все, дізнається його поточне місцезнаходження з урахуванням відстані від користувача. Логіка роботи мобільного сайту завжди відносно проста і може не враховувати незліченну кількість даних, які можуть надати мобільні пристрої;

Робота в офлайні. Інтернет є усюди, але не завжди. Навіть при наявності мобільного інтернет-з'єднання його якість в середньому гірша ніж якість домашніх й офісних інтернет-ліній. Якщо поставлена ціль, щоб користувачі не втрачали зв'язок з продуктом відразу, як тільки обірвався інтернет-зв'язок, розробка мобільного додатку – єдине можливе рішення;

Також ознакою престижу є те, що компанія розробляє мобільні додатки. Це ознака того, що компанія піклується про своїх клієнтів, надаючи їм вільний вибір ресурсів і вищий рівень комфорту, особливо коли вам потрібно побудувати тривалі, якісні та продуктивніші відносини з клієнтами. Наприклад, заявка - клієнта обслуговує персональний консультант. Крім того, власний мобільний додаток – це додатковий рекламний канал або, навпаки, привід переконатися в його перевагах за рахунок відсутності нав'язливої реклами. Мінуси мобільного додатку: Якщо ви розмістите QR-коди або прив'язки на своєму веб-сайті та в додатку, ваш веб-сайт завжди отримуватиме більше відвідувань, ніж установок вашого мобільного додатка. Деякі користувачі не встановили програму після того, як натиснули посилання на програму в магазині;

Кросплатформенність – мобільний додаток доступний не на всіх пристроях, на відміну від веб-сайту, який доступний на всіх пристроях;

Миттєві оновлення - для оновлень програми в магазині, на відміну від веб-сайту, завжди потрібен час, щоб перевірити їх. Оновлений сайт доступний користувачам практично відразу; Як правило, вартість розробки та підтримки мобільних додатків завжди вище відповідних показників для сайту;

Просування мобільного додатку однозначно дорожче, ніж веб-сайт. Ціна користувача безпосередньо залежить від його порогу входу, чим вищий поріг, тим більшу ціну він залучає. Для того, щоб користувачі заходили на веб-сайт, на рекламному блоці встановлюється певне значення кліку, і відвідувачі приймаються відразу після кліку. Те ж саме має бути вірним і для мобільних додатків, просто це неправда, що користувач заходить у магазин додатків і встановлює їх. Тому ціна інсталяції завжди набагато вища за ціну кліка.

Програми також можуть бути встановлені користувачем вручну, наприклад, запущений пакет програм Android на пристрої з операційною системою Android.

Спочатку мобільні програми пропонувалися як інструменти для контролю та управління звичайними інформаційними потоками, включаючи електронну пошту, календар, контакти, фондовий ринок і інформацію про погоду.

Однак попит і доступність інструментів розробниками призвели до швидкого поширення додатків для інших категорій електронних пристроїв за допомогою настільних додатків. Як і з іншим програмним забезпеченням, різке зростання кількості та різноманітності мобільних додатків призвело до появи великої кількості ресурсів знань про них, включаючи відгуки, відгуки та рецензії, тобто: блоги, журнали та спеціалізовані служби пошуку додатків в Інтернеті.

Використання мобільних додатків користувачами мобільних пристроїв стає все більш популярним. Згідно з даними дослідження App Annie, у 2017 році кількість завантажень додатків зросла на 60%, споживчі витрати зросли більш ніж удвічі, а кожен користувач проводив у додатках близько 43 днів на рік. Загальна кількість завантажень додатків зросла до 115 мільярдів у 2019 році: 31 мільярд з App Store і 84 мільярди з Google Play. Дослідники виявили, що використання мобільного додатку суттєво корелювало з його вмістом і залежало від часу доби та місця розташування користувача.

Висновок до розділу 1.

В даному розділі загальне поняття про плагінів та сучасних мультиплатформних застосунків. Визначено актуальність використання плагінів

в сучасних реаліях та розробку її за допомоги фреймворка Flutter, висвітлено усіх їх актувльні переваги та недоліки.

РОЗДІЛ 2
ПОСТАНОВКА ЗАДАЧІ ТА ІНСТРУМЕНТИ

2.1. Постановка задачі.

Розробити інформаційної систему (плагін) для аналізу використання мобільного додатку

додаток для таких ОС: Android, iOS

Етапи створення мобільних додатків:

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

Завданням данної роботи є розробка плагіну для аналізу та накопичування даних про користування мобільним додатком. Отже, система (плагін) буде являти собою готове рішення для розробників, що дозволить без власного написання коду збирати та аналізувати дані мобільного додатку а саме час перебування на кожному екрані додатку, аналіз у відсотках на яких екранах користувач провів більше всього часу , та скільки всього часу провів користувач у додатку.

Таким чином, основний функціонал для користувачів:

- Можливість отримати данні про кількість часу на кожному з екранів
- Можливість оперувати зібраними даними у різних форматах (DateTime, int milliseconds)

- Можливість отримати данні про весь час користування додатком
- Можливість отримати аналіз даних у відсотках про перебування користувача на екрані

2.2. Огляд і дослідження можливостей мови програмування Dart.

Сполучна ланка, яка будує місток між розробником і додатком, — це мова програмування. Існує багато мов, які дозволяють розробляти мобільні додатки. Сьогодні неможливо працювати в сфері розробки і знати лише одну мову програмування. Нові мови з'являються щороку, і всі вони створені для досягнення максимальної ефективності для певних завдань. Значна частина попиту розширюється та зростає.

Рисунок 2.1 – Логотип мови Dart

Dart — це мова програмування, створена Google. Dart позиціонується як заміна/альтернатива JavaScript. Один із розробників мови, Марк С. Міллер, написав, що JavaScript «має фундаментальні недоліки» («Javascript has fundamental flaws...»), які неможливо виправити. Ось чому був створений Dart. Перша публічна інформація про цю мову програмування з'явилася на конференції розробників Goto 12 вересня 2011 року. 10 жовтня 2011 року відбувся офіційний запуск мови Google Dart.

Завдання перед розробниками мови:

• Створити структуровану, але гнучку мову для веб-програмування.

• Зробити мову схожою на існуючі, щоб полегшити навчання.

• Висока продуктивність прийомних програм у браузерах та інших середовищах, від смартфонів до серверів.

• Спочатку було запропоновано два методи запуску програм Dart. Один із способів – через віртуальну машину (у браузерах, що підтримують мову), а інший – через проміжний переклад javascript (універсальний).

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

4 липня 2014 року ECMA затвердила першу версію мовного стандарту, і стандарт отримав назву ECMA-408 [7].

Dart - це оптимізована для клієнта мова, яка використовується для розробки швидких програм на будь-якій платформі. Мета Dart - надати найбільш продуктивну мову програмування для кроссплатформенной розробки у поєднанні з гнучкою платформою часу виконання серед додатків.

Мова визначається його технічною оболонкою, виборами, зробленими під час розробки, які визначають особливості та сильні сторони мови. Dart розроблений для технічної оболонки, яка особливо добре підходить для розробки на стороні клієнта, забезпечуючи як розробку (гаряче перезавантаження до секунди), так і високоякісний виробничий досвід для широкого спектру компіляцій (веб, мобільні пристрої та настільні комп'ютери). , Вважаю за краще обидва.

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

Мова Dart є типобезпечною. Використовуйте статичну перевірку типів, щоб гарантувати, що значення змінної майже завжди відповідає статичному типу змінної. Винятками є динамічні універсальні типи. Зазвичай це називається набором звуків. Типи обов'язкові, але інструкції типів необов'язкові через визначення типів.

Система типів Dart також є гнучкою, дозволяючи вам поєднувати динамічний набір із перевіркою під час виконання, що корисно під час експериментів або коли вам потрібен особливо динамічний код.

Dart надає надійну нулову безпеку (null safety), тобто значення не можуть бути нуловими (null), якщо не вказати, що вони можуть бути нуловими (<T?>). Завдяки надійній нуловій безпеці Dart може захистити вас від нулових ситуацій під час виконання за допомогою статичного аналізу коду. У порівнянні із багатьма іншими нул-безпечними мовами, коли Dart визначає, що змінна не допускає нул, ця змінна завжди не допускає нул. Якщо перевірити запущений код у налагоджувачі, виявиться, що відсутність нулових значень зберігається під час виконання, отже, надійна нулова безпека.

Рисунок 2.2 – Логотип Flutter SDK

2.3 Огляд та вивчення функцій Flutter SDK, плагінів, пакетів та бібліотек

Flutter — це кросплатформна мобільна платформа, яка дозволяє розповсюджувати вашу мобільну програму як на мобільних телефонах Android, так і на iOS. За допомогою мосту Mapp Intelligence Flutter можна відстежувати поведінку користувачів у вашій програмі. На жаль, неможливо відстежувати програми для watchOS, tvOS і iPadOS за допомогою пакета SDK Mapp Intelligence Flutter.

Збирайте важливі дані про те, як використовуються ваші додатки, відстежуйте, як ваші користувачі взаємодіють із вашим додатком, як вони переглядають певні сторінки та спеціальні події. На основі даних відстеження з додатків можна виміряти різні показники, які вже відомі з веб-аналітики,

наприклад покази сторінок, події, розмір екрана, операційна система, відстеження електронної комерції тощо. Особливо дані про покази сторінок або події, а також кількість переходів під час відстеження електронної комерції є корисною для того, щоб зробити висновки для оптимізації вашого бізнесу.

Ця документація допоможе вам налаштувати програму, надаючи корисні приклади та пояснювальні примітки, щоб краще зрозуміти структуру SDK.

Однією з основних переваг платформи Dart є «гарячий перезапуск». Це коли зміни у вихідному коді застосовуються одразу у запущеному додатку без необхідності перезапуску. [8]

Flutter – це платформа з відкритим вихідним кодом, розроблена Google, яка дозволяє легко та швидко створювати мобільні програми для iOS та Android.

При цьому Flutter взагалі не використовує нативних компонентів. Натомість всі елементи інтерфейсу користувача всередині фреймворку створюються з використанням власного графічного движка. Flutter дозволяє створювати всі елементи інтерфейсу користувача вашої програми з готових віджетів. У цьому відношенні Flutter схожий на інші фреймворки (React або Vue), але також має багато відмінностей. Так, без використання мови програмування JavaScript. Натомість Flutter використовує мову Dart[9].

Ідеально підходить для запуску MVP (мінімально життєздатний продукт)

Якщо вам потрібно якомога швидше представити свій продукт інвесторам, ви можете використовувати Flutter!

4 основні причини використовувати Flutter для вашого MVP:

- Розробка мобільного додатка з Flutter дешевше, тому що вам не потрібно створювати і підтримувати два мобільних додатки (по одному для iOS і Android).
- Одного розробника достатньо для створення MVP.
- Це ефективно; ви не бачите різниці між рідним додатком і додатком Flutter.

- Це красиво; Ви можете легко використовувати віджети, надані Flutter, і персоналізувати їх, щоб створити оригінальний користувальницький досвід для ваших клієнтів.[10]

Flutter DevTools – це набір інструментів для підвищення продуктивності та налагодження для Dart та Flutter.

Функції, доступні в DevTools:

• Перегляд макету інтерфейсу користувача та стану вашої програми Flutter.

• Діагностика проблем з продуктивністю інтерфейсу користувача в Flutter.

• Профільування ЦП для Flutter або Dart.

• Профілювання мережі Flutter.

• Налагодження на рівні вихідного коду Flutter або Dart.

• Виправлення неполадок із пам'яттю у програмах командного рядка Flutter або Dart.

• Перегляд загальної журнальної та діагностичної інформації про запуск програми командного рядка Flutter або Dart.

• Аналіз розміру коду та програми.

Slow development. Повільний цикл розробки через необхідність щоразу компілювати програму при внесенні змін. Додатковою незручністю є скидання стану програми. Припустимо, проблема знаходиться на третьому етапі майстра. При спробі виправити її, потрібно на кожну зміну в коді або знову проходити всі попередні кроки, або за допомогою хитрощів зробити так, щоб програма запускалася з потрібного місця.

Cross-platform look and feel. При розробці бувають випадки, коли програма має виглядати однаково на обох платформах. Враховуючи особливості платформ, іноді цього досить складно досягти.

Platform diversity. Різноманітність вендорів, які роблять пристрої на платформі Android, буває, позначається зненацька на тому, як виглядає і працює мобільний додаток.

Дві команди. При розробці окремих програм для iOS та Android, доводиться містити дві команди, які розробляють одні й ті самі функції. А також є

необхідність синхронізувати релізи при введенні в дію мажорних оновлень, щоб користувачі однієї з платформ не були обмежені у функціоналі.[11]

GetX (get) – дуже легке та потужне рішення для Flutter. Він поєднує в собі високоефективне управління станом, інтелектуальне впровадження залежностей та швидке та практичне управління маршрутами.

Рисунок 2.3 – Логотип GetX

GetX має три основні принципи. Це означає, що продуктивність, ефективність та організація є пріоритетами для всіх бібліотечних ресурсів.

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

Ефективність: GetX простий та зручний у використанні, оскільки має дуже зручний для розробників синтаксис. Що б вам не потрібно було зробити, GetX завжди має просте рішення. Це заощаджує час розробки та забезпечує максимальну продуктивність програми.

Загалом розробники повинні подбати про видалення контролерів з пам'яті. За замовчуванням ресурси, що не використовуються, видаляються з пам'яті, тому у GetX є можливість не дбати про це. Якщо потрібно щось зберегти в пам'яті, ви можете явно оголосити «permanent: true» у своїй залежності. Таким чином ви не тільки заощаджуєте час, а й знижуєте ризик непотрібних залежностей пам'яті. Завантаження залежностей також відкладається за умовчанням.

Організація: GetX дозволяє чітко розділити уявлення, логіку уявлення, бізнес-логіку, реалізацію залежностей та навігацію. Це рішення не залежить від

дерева віджетів (рендерингу), оскільки для переходу між корінням не потрібно ніякого контексту. Доступ до контролерів/блоків через внесенийВиджет не вимагає контексту, тому логіка подання та бізнес-логіка повністю відокремлені від рівня рендерингу. Немає потреби впроваджувати класи контролера/моделі/блоку у дерево віджетів через мультипровайдер. Для цього GetX використовує власну функцію застосування залежностей, щоб повністю відокремити DI від подання.

З GetX розробники точно знають, де знаходиться вся функціональність їхньої програми, і за умовчанням бачать чистий код. GetX не тільки спрощує розробку, а й дозволяє спільно використовувати модулі, що раніше було неможливо з Flutter. BLoC - це відправна точка для організації коду Flutter, що відокремлює бізнес-логіку від рендерингу. GetX — це природна еволюція цього, що відокремлює як бізнес-логіку, а й логіку уявлення. Додаткові реалізації залежностей і маршрутів також відокремлені один від одного, і шар даних знаходиться поза його межами. Ви знаєте, де все знаходиться і все це досить легко.

GetX - це найпростіший, практичніший і масштабованіший спосіб створення високопродуктивних додатків за допомогою Flutter SDK. Він має велику екосистему, яка добре працює разом, що спрощує роботу як для новачків, так і для експертів. Пропонує величезний набір [12]

GetX має безліч функцій, які ви можете почати програмувати в готовому вигляді, але кожен з цих методів знаходиться в окремому контейнері і виконується тільки при використанні. Якщо використовується лише керування станом, компілюється лише керування станом. Нічого, пов'язане з керуванням станом, не компілюється, якщо використовується лише root.

GetX має величезну екосистему, велику спільноту, безліч учасників і підтримуватиме Flutter, поки він існує. GetX може запускати той самий код на Android, iOS, в Інтернеті, Mac, Linux, Windows і навіть на серверах. Get Server робить код, написаний у вашій програмі, повністю доступним на сервері.

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

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

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

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

GetX спрощує розробку, не переймаючись продуктивністю. Продуктивність Flutter вже вражає, але бувають ситуації, коли розробники використовують менеджери станів та локатори для виділення блоків, сховищ, контролерів тощо. У цьому випадку ви повинні створити вручну виняток для цієї залежності, коли вона вам більше не потрібна. GetX використовує лише контролер, і контролер просто видаляється з пам'яті, коли він більше не використовується. Завдяки

SmartManagement, все, що не використовується, видаляється з пам'яті, тому вам не потрібно турбуватися ні про що, крім програмування. Розробники завжди повинні намагатися споживати якнайменше ресурсів без написання логіки.

Real Separation - Концепція «відділення презентації від бізнес-логіки». Це не особливість BLoC, MVC, MVVM, інші стандарти ринку мають цю концепцію. Однак цю концепцію часто можна пом'якшити у Flutter за рахунок використання контекстів. Якщо вам потрібний контекст для пошуку InheritedWidget, він знадобиться або у вашому поданні, або вам потрібно буде передати контекст у параметрі. Це рішення не підходить, оскільки завжди є залежність від бізнес-логіки View. GetX не повністю забороняє використання StatefulWidgets, InitState і т. д. Контролер має життєвий цикл і не залежить ні від чого у поданні, наприклад, коли йому потрібно зробити запит REST API. Коли ви використовуєте onInit для ініціації HTTP-дзвінка, змінні заповнюються в міру надходження даних. GetX є повністю реактивним (працює в потоці), тому при завантаженні елемента всі віджети, що використовують цю змінну, автоматично оновляться у своєму поданні. Це дає вам свободу самостійно писати і тестувати свою бізнес-логіку, оскільки вам не потрібно відправляти в неї нічого, крім користувацьких подій (таких як натискання кнопок) і ви можете працювати тільки з інтерфейсом користувача.

GetX постійно оновлюється, додаючи нові функції.

Overlay Support (overlay_support) — плагін Flutter, який підтримує накладення, спрощуючи програмне створення тостів та сповіщень. [13]

Carousel Slider (carousel_slider) — це пакет Flutter, який створює віджет слайдера каруселі з підтримкою нескінченного прокручування та вашими власними дочірніми віджетами. [14]

Cupertino Icons (cupertino_icons) – це бібліотека Flutter, що містить стандартний набір значків, які використовуються у віджетах Flutter Cupertino. [15]

Syncfusion Flutter Charts (syncfusion_flutter_charts) — бібліотека трепетних діаграм, яка включає віджети візуалізації даних, такі як декартові діаграми і

кругові діаграми, для створення інтерактивних високопродуктивних анімованих діаграм в реальному часі. [16]

MIME (mime) – пакет Flutter для керування визначеннями типів MIME та обробки мультимедійних потоків типів MIME. Ви можете використовувати клас MimeTypeResolver для визначення MIME-типу файлу. Він підтримує як використання розширень імен файлів, і байтове представлення файлів. Існує вбудований екземпляр MimeTypeResolver, доступний через функцію верхнього рівня lookupMimeType. Цей вбудований екземпляр має найпоширеніші розширення імен файлів та зареєстровані байти. Перетворювачі користувача можуть бути створені шляхом створення екземпляра MimeTypeResolver і додавання розширень імен файлів і байтів за допомогою addExtension і addMagicNumber. [17]

Logger (Логгер) – невеликий, простий у використанні пакет ведення журналів Flutter, який можна розширювати та налаштовувати під будь-які потреби. Цей реєстратор дуже нагадує реєстратор Android. [18]

File Provider (file_picker) — це плагін Flutter, який дозволяє вибирати один або кілька файлів за допомогою вбудованого провідника з підтримкою фільтрації та зміни розширень. [19]

Url Strategy (url_strategy) — це пакет для веб-застосунків Flutter, який дозволяє вам встановити веб-URL. [20]

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

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

Android Path Provider (android_path_provider) – це плагін Flutter для отримання каталогів Android. (Завантаження, відео, зображення). Цей плагін працює лише на платформі Android. [двадцять три]

Package Info Plus (package_info_plus) — це плагін Flutter для запиту інформації про пакет вашої програми, такий як CFBundleVersion на iOS та versionCode на Android.[24]

Flutter Local Notifications (flutter_local_notifications) — це кросплатформовий плагін Flutter для відображення та планування локальних повідомлень для програм Flutter, які можна налаштувати для кожної платформи. [двадцять п'ять]

Flutter Lints (flutter_lints) — цей пакет Flutter містить набір рекомендованих lints для додатків, пакетів та плагінів Flutter для просування хороших методів кодування. Цей пакет базується на рекомендованому Dart наборі lints, знайденому в package:lints. Лінти виявляються синтаксичним аналізатором дротика, який статично перевіряє код дротика. У IDE з підтримкою Dart проблеми, виявлені аналізаторами, відображаються в інтерфейсі користувача. Крім того, ви можете запустити аналіз Flutter та викликати аналізатор вручну. [26]

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

Віджет Flutter Focus — це віджет, який може фокусуватися і розмивати на основі FocusNode.[27].

Flutter Build Runner - пакет build_runner надає особливий спосіб створення файлів з використанням коду Dart поза такими інструментами, як pub. На відміну від публікації/складання, файли завжди генеруються безпосередньо на диску, а перескладання є інкрементальним на основі таких інструментів, як Bazel. [28]

Vibration - Плагін для обробки Vibration API на iOS, Android і в Інтернеті. Документи API.[29]

Flutter Shared Preferences - Обгортає постійне сховище для певної платформи для простих даних (NSUserDefaults на iOS і macOS, SharedPreferences на Android тощо). Дані можуть бути збережені на диску асинхронно, і немає гарантії, що записи будуть збережені на диску після повернення, тому цей плагін не можна використовувати для зберігання критичних даних.[30]

Local Auth - Цей плагін Flutter надає засоби для локальної автентифікації користувача на пристрої.

На підтримуваних пристроях це включає автентифікацію за допомогою біометричних даних, таких як відбиток пальця або розпізнавання обличчя.[31]

2.3. Огляді і дослідження аналогів

Firebase — американська компанія, постачальник хмарних сервісів, заснована в 2011 році Ендрю Лі і Джеймсом Темпліном і поглинена корпорацією Google у 2014 році.

У травні 2012 року компанія отримала 1,4 мільйона доларів від Flybridge Capital Partners, Greylock Partners і NEA, а в червні 2013 року залучила 5,6 мільйона доларів від Union Square Ventures і Flybridge Capital Partners.

Рисунок 2.4 – логотип Firebase

Основним сервісом є хмарна СУБД класу NoSQL, яка дозволяє розробникам додатків зберігати та синхронізувати дані між кількома клієнтами. Він підтримує інтеграцію з програмами для операційних систем Android і iOS і реалізує API для програм JavaScript, Java, Objective-C і Node.js. Також можна працювати безпосередньо з базами даних у стилі REST із низки фреймворків JavaScript. ,

AngularJS, ReactJS, Vue.js, Ember.js, Backbone.js. Для шифрування даних надаються API.

Серед інших послуг, які компанія пропонує, хостинг для зберігання статичних файлів (CSS, HTML, JavaScript тощо), запущений 13 травня 2014 року, доставка через CDN і надає послугу автентифікації клієнта, яка використовує лише код і підтримує вхід. Facebook, GitHub, Twitter, Google (FirebaseSimpleLogin).

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

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

Продукти Firebase добре працюють як окремо, так і разом, прекрасно обмінюючись даними.

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

Firebase допомагає створювати якісні програми, розширювати базу користувачів і заробляти більше грошей. Кожна функція працює окремо, але разом вони працюють ще краще.[32]

Що таке Firebase Analytics?

Як зрозуміло з назви, Firebase Analytics - це інструмент, який допоможе вам дізнатися більше про продуктивність вашої програми. Він надасть повну інформацію про те, як користувачі операційних систем iOS та Android взаємодіють із цією програмою.

Як тільки ви налаштуєте взаємодію з програмою, вона автоматично почне відстежувати її залежно від заданих подій. Це означає, що користувачі зможуть отримувати інформацію про продуктивність своєї програми з початку роботи.

Основою аналітики Firebase є Google Analytics, яка є необмеженим та безкоштовним аналітичним рішенням. У всіх сервісах Firebase буде інтегрована

аналітика, яка здатна надавати своїм користувачам необмежену кількість звітів щодо 500 подій, які визначають користувачі Firebase SDK.

Це допоможе вам зрозуміти поведінку аудиторії, що в свою чергу допоможе приймати правильні рішення про продуктивність програми та подальше її просування.

FirebaseAnalytics та Google Analytics

Більшість людей думають, що Google Analytics і Firebase Analytics - це те саме. Проте дуже важливо зрозуміти, що вони мають відмінності, які є в обох платформ. Тут у нас є кілька суттєвих відмінностей, про які вам потрібно знати.

Зазвичай Firebase Analytics також називають Google Analytics, тому що основою Firebase є Google Analytics. Він чудово підходить для всіх типів мобільних програм.

Google Analytics має повну підтримку WEB-застосунків. Однак не варто плутати його з Google Analytics, розробленою для роботи з інтернет-ресурсами.

Призначення обох платформ по-різному. Firebase орієнтована на мобільні пристрої, а Google Analytics – це специфічна аналітична платформа. Google Analytics працює для мобільних програм, але основне її призначення, це робота з інтернет ресурсами з метою моніторингу та реклами.

Переваги Firebase Analytics

Коли є різні типи аналітичних інструментів, людям цікаво знати, чому їм варто зупинитися на Firebase Analytics. Існує багато переваг даної платформи, які ви не знайдете в інших інструментах. Тут ми згадали основні, які, на нашу думку, вам потрібно знати.

1. **Деталізовані звіти**

Важливо надати клієнтам персоналізований досвід роботи з мобільним додатком. Щоб зробити це можливим, розробникам додатків необхідно знати про поведінку та моделі використання клієнтів. Firebase допоможе розробникам програм отримувати повні звіти про залучення та продуктивність користувачів.

Таким чином, це допоможе розробникам внести необхідні покращення та оновлення, щоб гарантувати, що клієнти матимуть кращий досвід роботи з цим

додатком. За допомогою цієї аналітики ти зможеш керувати всім. Вона також дає змогу краще оцінити ситуацію.

2. **Інтеграція з Google Analytics**

Firebase інтегрується з Google Analytics, що є однією з найбільших переваг цієї платформи. Це рішення, інтегроване з сервісом Google, дозволяє розробникам краще розвивати свої програми, розуміючи, що потрібно клієнтам залежно від їх рівня залучення.

Це дасть вам всі необхідні інструменти для розуміння поведінки користувачів додатків і краще зрозуміти, у якому напрямку вам варто працювати, щоб підвищити задоволеність клієнтів. Це створить більш персоналізоване та індивідуальне відчуття від використання вашої програми.

3. **Підвищення лояльності**

Знання про залучення користувачів дуже важливо при розробці програм. Firebase надасть повну інформацію про залучення клієнтів, щоб ви могли розвивати можливості та функції, які користувачі хочуть знайти у розробленому вами додатку. Це підвищить задоволеність та лояльність клієнтів.

Таким чином, розробники зможуть надати контент, необхідний аудиторії у потрібний час, щоб усі потреби клієнтів були професійно задоволені вашим продуктом.

Підтримує стійке зростання.

Не буде помилкою сказати, що Firebase Analytics стане чудовою та корисною функцією для розробників додатків, оскільки вона допоможе розвивати стійке зростання бізнесу та керувати потребами клієнтів та зможе краще налагодити зворотний зв'язок. Це надійна платформа для зростання каналів розробки програм.

Можливості Firebase Analytics

Firebase Analytics постачається з великою кількістю відмінних функцій для розробників. Тут перелічені основні їх.

Firebase Analytics надає аналітику для мобільних додатків у зручному та зрозумілому вигляді, який можна налаштувати під свої цілі. Тобто. ви можете налаштувати всі необхідні параметри та події відповідно до ваших уподобань.

Дані Firebase Analytics надаються відповідно до поведінки користувачів у додатках і можуть допомогти прийняти ефективніші рішення для покращення маркетингових стратегій.

Панель моніторингу Firebase Analytics - одна з кращих Огляд Android Studio

Flurry Analytics – безкоштовна платформа для збору та аналізу статистики мобільних додатків iOS та Android. Сервіс отримує дані про користувачів, їх дії та події. Одночасно можна відстежувати до п'яти програм. Для початку роботи потрібне підключення SDK. Інструмент доступний як у браузері, так і в програмах, які можна завантажити з AppStore та Google Play.

Можливість відстежувати події та дії відвідувачів буде корисною для веб-розробників. За допомогою сервісу маркетологи зможуть вивчити аудиторію – географію, тип пристроїв, інтереси. Відсутність плати робить платформу доступною для бізнесу-початківця.

Рисунок 2.5 – логотип Flurry Analytics

Flurry Analytics збирає відомості про події та подає їх у вигляді діаграм. При аналізі можна сегментуватися за версіями програми, датою встановлення, демографічними даними власника телефону, за мовою та каналом залучення відвідувача. Інструмент дозволяє простежити стежку користувача - як він

взаємодіяв із програмою протягом певного періоду. Є функція прорахунку вирв продаж, за допомогою якої можна дізнатися, наприклад, на якому етапі клієнти кидають покупку. Завдяки платформі можна порівнювати поведінку аудиторії однієї програми для двох різних операційних систем. Система надає інформацію про інтереси користувачів. Існує автоматичний розрахунок середнього щоденного доходу.[33]

AppMetrica. Безкоштовна платформа від Яндекса, призначена для мобільного аналітики, атрибуції установок, збору звітів про баги і відправки push-повідомлень. Підходить для трекінгу мобільних додатків на Android, iOS, Windows, Xamarin, Unity і Cordova.

Рисунок 2.6 – логотип AppMetrica

Ключові особливості:

- підтримує багато платформ;
- повністю безкоштовно (ніяких обмежень і демо-режимів);
- є функції когортного аналізу та сегментації аудиторії;
- формує автоматичні звіти про помилки додатка;
- відстежує налаштовані подій.

На відміну від Google Analytics, ця система аналітики орієнтована на мобільні додатки і дозволяє відстежувати статистику завантажень і отримувати повідомлення про нові помилки. Маркетологи оцінять функції когортного аналізу, контролю атрибуції і сегментації аудиторії.[34]

AppsFlyer

AppsFlyer – сервіс мобільного атрибуції для вивчення ефективності як онлнай-, так і оффлайн-каналів збуту. Вбудовані інструменти аналітики мобільних додатків на iOS і Android дозволяють вивчати поведінку користувачів,

розраховувати ROI маркетингових витрат і багато іншого. Працювати з платформою можна на робочому столі і смартфоні.

Рисунок 2.7 – логотип AppsFlyer

Ключові особливості:

- аналізує ефективність каналів залучення користувачів;
- збирає детальну статистику про додатки на iOS і Android;
- збирає інформацію про поведінку користувачів;
- відстежує прибутковість банерів всередині програми.

Ця система аналітики більше підходить для роботи маркетологів, так як її основний орієнтир – оцінка ефективності рекламних кампаній (онлайн і оффлайн). Фіксованої оплати за сервіс немає, вона визначається на основі кількості прорахованих конверсій.

Налаштування аналітики мобільного додатка – важливий етап у розвитку продукту. Без інформації про поведінку користувачів, ефективності рекламних кампаній і т.п. неможливо приймати об'єктивні рішення щодо подальших дій. Ви будете працювати «наосліп» і не розуміти, що робите.

Простий установки системи аналітики теж недостатньо. Ви повинні розуміти, за якими метриками стежити. Ключові показники визначаються на стадії планування нового мобільного застосування в залежності від його призначення, сфери діяльності та інших особливостей бізнесу.

Про грамотної аналітики замислюйтесь на ранніх стадіях розробки. Для настройки і тестування недостатньо тижні. Це об'ємний процес, але він виправдовує витрачені кошти, тому не забудьте приділити йому робочий час і підготувати все до запуску додатка.[35]

AppAnnie. За допомогою програмного забезпечення для аналітики мобільних програм організації можуть відстежувати активність сеансів та поведінку користувачів App Annie Review. Ці продукти вимірюють відкриття та завантаження мобільних програм, виявляють точки виходу користувачів, відстежують час, що витрачається програмою, та створюють шляхи для відстеження активності користувачів App Annie Review.

Рисунок 2.8 – логотип AppAnnie

App Annie програмне забезпечення може створювати звіти про дані різними способами, включаючи використання в реальному часі, часові тенденції, географічну сегментацію та атрибути пристрою або операційної системи. Маркетингові групи можуть використовувати ці дані для аналізу популярності у регіонах чи демографічного аналізу.

Бізнес-відділи та інші відділи можуть використовувати дані, надані за допомогою продуктів для аналізу мобільних додатків. Хоча деякі аналітики мобільних додатків охоплюють лише аналіз додатків, інші можна використовувати як додатків, так і для чисельного аналізу.

"App Annie - це платформа для класифікації додатків, аналізу та аналізу ринку, особливо ігор та додатків".

App Annie надає користувачам повний огляд аналітики додатків, порівняльного аналізу конкурентів та тенденцій у галузі додатків. Це потужне рішення для моніторингу програм. Додаток Annie доступний у двох тарифних планах: Free та Premium.

Безкоштовна версія включає основні функції, такі як діаграми, рейтинги, рейтинги, огляди та інформацію про додатки, в той час як тарифний план преміум-класу включає параметри оптимізації App Store, оцінки доходів і

завантаження, оцінки використання і реклами, рейтинги аудиторії та багато іншого. Статистика доступна для магазинів Google Play, Apple, Amazon та Windows Phone.

Ключовими інструментами цього набору є інструмент індексації, який відображає області додатків на основі завантажень і доходів, і інструмент реклами, який реєструє доходи та рекламу монетизації. За допомогою App Annie Intelligence користувачі можуть керувати всім життєвим циклом програми, визначати можливості зростання та покращувати свої знання про конкурентів.

Огляд продукту App Annie

1. Статистика діаграми App Store
2. Порівняйте ранги у країнах та категоріях
3. Рекламна аналітика
4. Аналітика продуктивності додатків
5. Завантаження та оцінка доходів
6. Рейтинг програм та найкращі програми
7. Порівняльний аналіз конкурентів
8. Прогнози галузевих тенденцій

App Annie надає унікальну платформу для вимірювання аналітики ринку додатків та оптимізації магазину додатків:

- Статистика магазину - відстежуйте екосистему програми та її конкурентів. Переглядайте основні діаграми щогодини або дні, а також історію рейтингу, ключові слова для пошуку та вибрані місця розміщення для мільйонів програм.
- Аналітика - Потужні візуалізації для аналізу кінцевого результату у пам'яті вашої програми. Відстежуйте завантаження, доходи, огляди, ключові слова та багато іншого у своєму додатку «портфель», легко організувати у вигляді панелі та без використання SDK.
- Рекламна аналітика – легко додавайте всі свої рекламні дані з кількох платформ. Більше ніякого ручного агрегування! Автоматично

отримуйте свою рекламу, дохід, CPI, eCPM та заповнюваність із кількох рекламних платформ на одній панелі.

- Enterprise Intelligence – незамінний інструмент, який використовується керівниками у всьому світі для планування стратегії для своїх додатків. Скачайте завантаження та оцінки доходів для всіх додатків, видавців, категорій, країн та платформ. Найрозумніша інтелектуальна платформа для додатків, що використовують дев'ять із десяти найкращих редакторів додатків.

app_usage 2.0.0 . Статистика використання програми лише для Android. Зауважте, що статистика точна лише за день. На жаль, це обмеження впровадження Google. Це плагін що теж збирає інформацію але має ряд обмежень. Він збирає статистику лише для Android платформи.[36]

2.4. Огляд Android Studio

Android Studio, інтегроване середовище розробки (IDE) для роботи з платформою Android, було анонсовано 16 травня 2013 року на конференції Google I/O.

IDE знаходиться у відкритому доступі з версії 0.1, випущеної у травні 2013 року, і перебуває на стадії бета-тестування з версії 0.8, випущеної у червні 2014 року. Перша стабільна версія, 1.0, була випущена в грудні 2014 року, коли ми оголосили застарілим модуль Android Development Tools (ADT) для Eclipse.

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

Нові функції з'являються з кожною версією AndroidStudio. В даний час доступні такі функції:

8. Розширений редактор макетів: можливість маніпулювати компонентами інтерфейсу користувача за допомогою WYSIWYG, перетягування, можливість попереднього перегляду макета в конфігураціях з кількома екранами.

9. Колекція програм на основі Gradle.

10. Створення різних типів збірок та кількох файлів .apk

11. Рефакторинг коду

12. Аналізатор статичного коду (Lints), який може знайти проблеми з продуктивністю, несумісність версій тощо.

13. Вбудований ProGuard та утиліта для підпису додатків.

14. Базове компонування Android та шаблони компонентів.

15. Підтримка розробки програм для Android Wear та Android TV.

16. Вбудована підтримка Google Cloud Platform, включаючи інтеграцію із сервісами Google Cloud Messaging та AppEngine.

17. Android Studio 2.1 підтримує Android N Preview SDK. Це означає, що розробники можуть почати писати програми для нової програмної платформи.

18. Нова версія AndroidStudio 2.1 працює з оновленим компілятором Jack, покращеною підтримкою Java 8 та покращеною функцією InstantRun.

19. Platform-tools 23.1.0 або новіше для Linux, лише 64-розрядна версія.

20. AndroidStudio 3.0 за замовчуванням включає мовні інструменти Kotlin та заснований на JetBrains IDE.[37]

2.5. Огляд Git та GitLab

Git – це безкоштовна розподілена система керування версіями з відкритим вихідним кодом, призначена для швидкого та ефективного керування будь-яким проектом, маленьким чи великим.

Це невеликий бізнес із дуже низькими обсягами виробництва. Розширіть інструменти SCM, такі як Subversion, CVS, Perforce та ClearCase, за допомогою таких функцій, як оптимізоване локальне поширення та міграція у промислові роботизовані процеси.

GitLub - це компанія, яка пропонує аналогічну послугу користувачам GitHub з додатковими перевагами, такими як приватні репозиторії для

безкоштовних передплатників. Можливість розгорнути систему на сторонньому сервері також є великою перевагою.

Програмне забезпечення доступне в системі керування пакетами Omnibus.

Цю програму створили Дмитро Запорожець та Валерій Сізов з України. Муніципальне управління Сіце-Сібранді знаходиться в Утрехті. Написано Рубі.

Станом на грудень 2016 року в компанії працює понад 80 співробітників і понад 1000 співробітників підписалися на відкритий вихідний код. Систему використовують понад 100 000 організацій, включаючи НАСА, ЦЕРН, Alibaba, Invincea, O'Reilly Media, Leibniz-Rechenzentrum (LRZ) та Дослідницький центр Юліха.

Кількість співробітників збільшилася з 9 у березні 2015 року до 150 у грудні 2016 року. Фінансування проекту збільшилось до 26 мільйонів доларів після отримання 20 мільйонів доларів від August Capital у вересні 2016 року.

На сайті є сервіс pastebin gist.github.com для швидкого опублікування фрагментів коду. [38]

GitLab пропонує кілька ключових переваг для розробників. Ось деякі з його найважливіших переваг:

• Надзвичайно просте налаштування

• Зручний інтерфейс та інструменти

Дозволяє необмежене безкоштовне особисте сховище

• Інтеграція з багатьма API та сторонніми сервісами.

• Працює надійно та безвідмовно

Звичайно, як і будь-який інший інструмент, доступний сьогодні, у GitLab є свої недоліки:

• Інтерфейс користувача може бути трохи складним для навігації

• У інструменті є помилки, і він може трохи заплутатися. [39]

Висновок до розділу 2

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

РОЗДІЛ 3
РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ АЛАЛІЗУ ВИКОРИСТАННЯ МОБІЛЬНОГО ДОДАТКУ

3.1. Створення структури плагіну

З курсу, щоб розробити plug-in з природним підтримкою, вам потрібно знати один або більше програмних мов, щоб отримати природний код поточно ви можете використовувати Java або Kotlin для Android і Objective-C або Swift для iOS розвитку.

Способи створення проекту плагіна

Отже, у вас є два способи зробити це:

- CLI (інтерфейс командного рядка)
- Android Studio

У нас є три різні варіанти створення проекту плагіна:

Підключення

Ця опція створює проект Flutter із підпроектами iOS та Android. Ви можете розробити власні функції та відкрити API, написаний на Dart.

Пакет

Цю опцію можна використати, якщо ви хочете розробити функціональні можливості просто за допомогою Dart без вбудованої підтримки, наприклад, це може бути віджет, бібліотека віджетів або деякі службові функції, які ви плануєте використовувати в кількох проектах тощо.

Модуль

Ця опція створює рішення, яке можна інтегрувати в існуючий проект Android або iOS.

Плагін

Клієнт просить мене інтегрувати рідну бібліотеку в наш проект, після деяких досліджень я вирішив розробити плагін flutter, тож давайте подивимося, як це зробити.

Визначення платформ, які плагін підтримує. У файл pubspec.yaml ми можемо додати конкретну інформацію про пакет.

```
flutter:
 plagin:
  platforms:
   android:
    package: com.example.fluter_screen_timehandler
    pluginClass: FlutterScreenTimehandlerPlugin
  ios:
    pluginClass: FlutterScreenTimehandlerPlugin
```

Плагін містить у собі модуль example в якому розробник плагіну має показати яким чином інші можуть з ним взаємодіяти. Зазвичай не весь функціонал показується розробником у даному модулі , але найважливіший.

Основна структура плагіну сладається з 6 файлів :

1. time_in_app.dart
2. time_traking_model.dart
3. widget_extencion.dart
4. flutter_screen_timehandler.dart
5. flutter_screen_timehandler_method_channel.dart
6. flutter_screen_timehandler_platform_interface.dart

Також присутній модуль example для демонстрації роботи плагіну. У pub dev він буде відображатись у вкладинці Example що б кожен розробник міг подивитися основну можливість плагіну просто скопіювавши код та додавши у свій проєкт. Це допомогає заощадити велику кількість часу . Замість того що б самостійно аналізувати та робирати код плагіну , розробник може просто скопіювати приклад його використаня що дає змогу швидше зрозуміти як він працює.

3.2 Розробка основного функіоналу

Розробка основного функціоналу плагіну починається з визначення що саме має робити плагін. У моєму випадку він має збирати дані про перебування користувача на екрані(скріні) додатку окремо для кожного.

Все це реалізовано у классі

```
import 'package:flutter_screen_timehandler/time_traking_model.dart';
import 'package:plugin_platform_interface/plugin_platform_interface.dart';

import 'flutter_screen_timehandler_method_channel.dart';

abstract class FlutterScreenTimehandlerPlatform extends PlatformInterface {
 FlutterScreenTimehandlerPlatform() : super(token: _token);

 static final Object _token = Object();

 static FlutterScreenTimehandlerPlatform _instance =
    MethodChannelFlutterScreenTimehandler();

 static FlutterScreenTimehandlerPlatform get instance => _instance;

 static set instance(FlutterScreenTimehandlerPlatform instance) {
  PlatformInterface.verifyToken(instance, _token);
  _instance = instance;
 }

 Future<String?> getPlatformVersion() {
  throw UnimplementedError('platformVersion() has not been implemented.');
 }

 static int timeOnScreen = 0;
```

```
static late Timer _timer;

static num allTimeInApp = 0;

static void startTimer() {
  _timer = Timer.periodic(const Duration(milliseconds: 1), (timer) {
    timeOnScreen = _timer.tick;
  });
}

static Map<String, dynamic> listOfScreens = {};

static List<TimeTracingModel> listOfScreensModels = [];

static void cancelTimer(String widgetName) async {
  if (listOfScreens.containsKey(widgetName)) {
    listOfScreens.update(widgetName, (value) => value += timeOnScreen);
    var oldElement = listOfScreensModels
        .where((element) => element.name == widgetName)
        .first;
    var index = listOfScreensModels.indexOf(listOfScreensModels
        .where((element) => element.name == widgetName)
        .first);
    listOfScreensModels.removeAt(index);
    listOfScreensModels.insert(index,
        TimeTracingModel(oldElement.name, timeOnScreen += oldElement.time));
  } else {
    listOfScreens.addAll({widgetName: timeOnScreen});
```

```
      listOfScreensModels.add(TimeTracingModel(widgetName, timeOnScreen));
    }
    timeOnScreen = 0;
    allTimeInApp = 0;
    for (var element in listOfScreens.values) {
      allTimeInApp += element;
    }

    _timer.cancel();
  }

  static String millisTiTime(num value) {
    var SS = ((value / 1000) % 60).round();
    var MM = ((value / (1000 * 60)) % 60);
    var HH = ((value / (1000 * 60 * 60)) % 24);

    var SSstring = SS < 10 ? "0${SS}" : "$SS";
    var MMstring = MM < 10
        ? "0${MM.toString().substring(0, 1)}"
        : "${MM.toString().substring(0, 1)}";
    var HHstring = HH < 10
        ? "0${HH.toString().substring(0, 1)}"
        : "${HH.toString().substring(0, 1)}";
    return "$HHstring:$MMstring:$SSstring";
  }

}
```

У классі є декілка методів , а саме :

1. startTimer()

2. cancelTimer()
3. millisToTime()

startTimer – при переході користувачем на екран , одразу починає відстежувати яку ділянку часу він провів саме на цьому екрані. Цей метод потрібен лише для запуску таймеру та щоб змінити значення змінної яка потім буде використовуватись для запису даних до map та list .

cancelTimer – більша частина функціоналу зосереджена саме у цьому методі.Він використовується одразу як користувач вийшов з екрану , або зробив навігацію на інший. Даний метод зберігає останнє значеня , що зберіг таймер, до моделі данних TimeTrakingModel , та записує його у лист данних.

TimeTracingModel – модель даних з 2 полями , name та time де name -це назва екрану , на якому знаходився користувач , time – кількість часу що він провів на ньому до навігації на інший екран.

```
class TimeTracingModel {
 final String name;
 final int time;

 TimeTracingModel(this.name, this.time,);
}
```

millisToTime – це метод що дозволяє користовачу перевести значення типу num до типу DateTime(HH/MM/SS) . Він розроблений для спрощення відображення данних.

Файли программи flutter_screen_timehandler.dart та flutter_screen_timehandler_method_channel.dart використовуються для спілкування за нативними платформами . Як приклад вони можуть отримати версію девайсу:

flutter_screen_timehandler_method_channel.dart

```
import 'package:flutter/foundation.dart';
import 'package:flutter/services.dart';
```

```
import 'flutter_screen_timehandler_platform_interface.dart';

/// An implementation of [FlutterScreenTimehandlerPlatform] that uses method channels.
class MethodChannelFlutterScreenTimehandler extends FlutterScreenTimehandlerPlatform {
 /// The method channel used to interact with the native platform.
 @visibleForTesting
 final methodChannel = const MethodChannel('flutter_screen_timehandler');

 @override
 Future<String?> getPlatformVersion() async {
  final version = await methodChannel.invokeMethod<String>('getPlatformVersion');
  return version;
 }
}
```

flutter_screen_timehandler.dart

```
import 'flutter_screen_timehandler_platform_interface.dart';

class FlutterScreenTimehandler {
 Future<String?> getPlatformVersion() {
  return FlutterScreenTimehandlerPlatform.instance.getPlatformVersion();
 }
}
```

Найбільша частина логіки створена у файлі flutter_screen_timehandler_platform_interface.dart що дозволяє відстежувати дані користувача про перебування на екранах додатку. Для підключення плагіну у файлі pabspec.yaml у залежності потрібно додати плагін.

```
dependencies:
 flutter:
    sdk:
 flutter_screen_timehandler:
```

3.3. Розробка допоміжних віджетів для користувачів

Розробка допіжних віджетів вкрай потрібна для розробників . Загалом їх у плагіні є 2:

1. StatefulWidgetWrapper
2. TimeInApp

StatefulWidgetWrapper – необхідний для спрощеня роботи з плагіном. Розробнику потрібно обернути свій єкран у цей віджет для того, що б він міг збирати інформацію за нього. Ця обгортка надає змогу плагіну одразу при переході користувача на екран почати відлік його перебування на данному екрані та зупинити відлік коли користувач перейде на інший екран. У якості параметрів цей класс приймає widget , тобто екран , який розробник хоче обернути у цю обгортку для подальшого його відслідковування, та runtimeTypeWidget – це значеня що є у кожного віджета у Flutter , та як мало хто знає цей параметер зберігає у собі назву самого віджета . Тобто якщо розробник назве віджет MyFirsWidget цей параметер збереже саме дане значення та передасть його для збереження та , у подальшому , ідентифікації окремого скріна.

Код файлу widget_extension.dart:

```
import 'package:flutter/widgets.dart';
import
'package:flutter_screen_timehandler/flutter_screen_timehandler_platform_interface.dart';
import 'package:focus_detector/focus_detector.dart';

class StatefulWidgetWrapper extends StatefulWidget {
  final Widget child;
```

```
  final String runtimeTypeWidget;

  const StatefulWidgetWrapper({
    Key? key,
    required this.child,
    required this.runtimeTypeWidget,
  }) : super(key: key);

  @override
  State<StatefulWidgetWrapper> createState() => _StatefulWidgetWrapper();
}

class _StatefulWidgetWrapper extends State<StatefulWidgetWrapper> {
  @override
  Widget build(BuildContext context) {
    return FocusDetector(
        onFocusGained: () {
          FlutterScreenTimehandlerPlatform.startTimer();
        },
        onFocusLost: () async {

FlutterScreenTimehandlerPlatform.cancelTimer(widget.runtimeTypeWidget);
        },
        child: widget.child);
  }
}
```

TimeInApp – це допоміжний віджет який відображає у собі усі зібрані данні у форматі списку. Він створений для швидкої інтеграції до додатку та відображення користувачу (якщо так вирішить розробник) інформації про його перебування на всіх екранах та загальний час , що він провів у додатку за

останню сесію. Розробнику не потрібно розробляти власноруч екран для відображення всіх даних , що значно пришвидшує роботу.Все що потрібно це зробити навігацію на цей екран і дозволити користувачу побачити всі дані.

Приклад вигляду екрану.

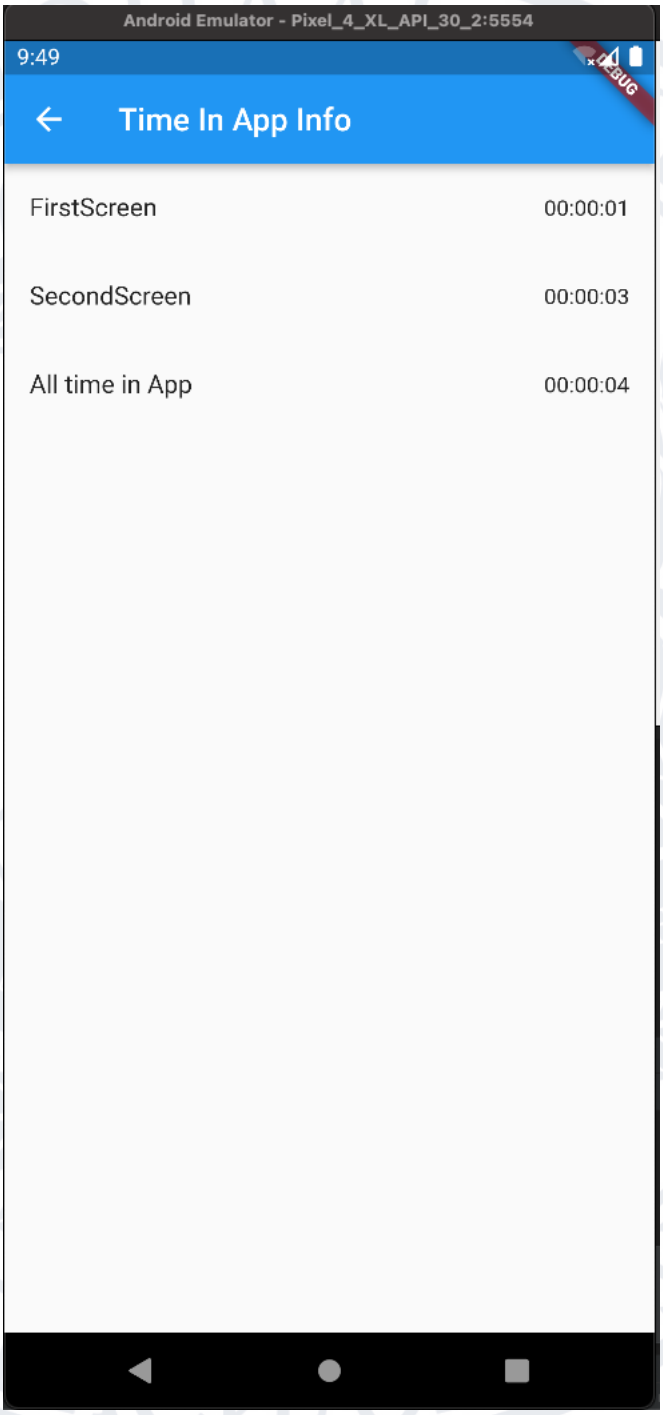

Рисунок 3.1 – Екран з відображенням результату роботи плагіну

Дані відображаються у оформаті DateTime(HH:MM:SS), для зручного користування та відображення (якщо потрібно користувачеві).

Код файлу time_in_app.dart:

```
import 'package:flutter/material.dart';
import
'package:flutter_screen_timehandler/flutter_screen_timehandler_platform_interface.d
art';

class TimeInApp extends StatefulWidget {
  const TimeInApp({Key? key}) : super(key: key);

  @override
  State<TimeInApp> createState() => _TimeInAppState();
}

class _TimeInAppState extends State<TimeInApp> {

  @override

  Widget build(BuildContext context) {
    return Scaffold(
      appBar: AppBar(
        title: Text("Time In App Info"),
      ),
      body: SingleChildScrollView(
        child: Column(
          children: [
            ...FlutterScreenTimehandlerPlatform.listOfScreensModels
                .map((e) => ListTile(
                      title: Text(e.name),
                      trailing: Text(FlutterScreenTimehandlerPlatform.millisTiTime(e.time)),
                    )),
            ListTile(
```

```
            title: Text("All time in App"),
            trailing: Text(FlutterScreenTimehandlerPlatform.millisTiTime(
              FlutterScreenTimehandlerPlatform.allTimeInApp)),
          )
        ],
      ),
    ),
  );
 }
}
```

Далі базовий віджет, створений для використання розробником, що дає змогу плагіну робити всю роботу власноруч .

3.4. Деплой плагіну для публічного використання

API плагіна підтримує об’єднані плагіни, які дозволяють відокремлювати різні реалізації платформи. Тепер ви також можете вказати, які платформи підтримує плагін, наприклад веб і macOS.

Згодом старі API плагінів стануть застарілими. У короткостроковій перспективі ви побачите попередження, коли фреймворк виявить, що ви використовуєте плагін старого стилю. Інформацію про те, як оновити плагін, див. у розділі Підтримка нових API плагінів Android.

Об'єднані плагіни. Об’єднані плагіни – це спосіб розділити підтримку різних платформ на окремі пакети. Отже, об’єднаний плагін може використовувати один пакет для iOS, інший для Android, інший для Інтернету та ще один для автомобіля (як приклад пристрою IoT). Серед інших переваг цей підхід дозволяє експерту домену розширити наявний плагін для роботи на платформі, яку він знає найкраще.

Для об’єднаного плагіна потрібні такі пакети:

app-facing package:

Пакет, від якого залежить використання плагіна користувачами. Цей пакет визначає API, який використовується програмою Flutter.

platform package(s):

Один або кілька пакетів, які містять специфічний для платформи код реалізації. Пакет, спрямований на програму, звертається до цих пакетів — вони не включені в програму, якщо вони не містять функціональні можливості певної платформи, доступні для кінцевого користувача.

platform interface package:

Пакет, який приклеює упаковку, спрямовану до програми, до пакета(ів) платформи. Цей пакет оголошує інтерфейс, який має реалізувати будь-який пакет платформи для підтримки пакета, спрямованого на додаток. Наявність єдиного пакета, який визначає цей інтерфейс, гарантує, що всі пакети платформи реалізують ту саму функціональність єдиним чином.

Схвалений об’єднаний плагін. В ідеалі, додаючи реалізацію платформи до об’єднаного плагіна, ви погоджуєтеся з автором пакету, щоб включити свою реалізацію. Таким чином оригінальний автор підтримує вашу реалізацію.

Наприклад, скажімо, ви пишете реалізацію foobar_windows для (уявного) плагіна foobar. У схвалений плагін оригінальний автор foobar додає вашу реалізацію Windows як залежність у pubspec для пакета, що звертається до програми. Потім, коли розробник включає плагін foobar у свою програму Flutter, реалізація Windows, а також інші схвалені реалізації автоматично стають доступними для програми.

Несхвалений об’єднаний плагін. Якщо ви не можете з будь-якої причини отримати свою реалізацію від оригінального автора плагіна, тоді ваш плагін не схвалено. Розробник все ще може використовувати вашу реалізацію, але повинен вручну додати плагін до файлу pubspec програми. Отже, розробник повинен включити як залежність foobar, так і залежність foobar_windows, щоб досягти повної функціональності.

Додавання ліцензій до файлу LICENSE. Окремі ліцензії всередині кожного файлу LICENSE мають бути розділені 80 дефісами на рядку.

Якщо файл LICENSE містить більше однієї ліцензії на компонент, тоді кожна ліцензія на компонент має починатися з назв пакетів, до яких застосовується ліцензія на компонент, з назвою кожного пакета в окремому рядку, а список імен пакетів відокремлений від фактичної ліцензії. текст порожнім рядком. (Назви пакетів не обов'язково збігаються з пакетом pub. Наприклад, сам пакет може містити код із кількох сторонніх джерел і, можливо, потребуватиме ліцензії на кожен із них.). [15]

Приклад ліцензії що використовується для плагіну:

Copyright 2013 The Flutter Authors. All rights reserved.

Redistribution and use in source and binary forms, with or without modification, are permitted provided that the following conditions are met:

* Redistributions of source code must retain the above copyright notice, this list of conditions and the following disclaimer.
* Redistributions in binary form must reproduce the above copyright notice, this list of conditions and the following disclaimer in the documentation and/or other materials provided with the distribution.
* Neither the name of Google Inc. nor the names of its contributors may be used to endorse or promote products derived from this software without specific prior written permission.

THIS SOFTWARE IS PROVIDED BY THE COPYRIGHT HOLDERS AND CONTRIBUTORS "AS IS" AND

ANY EXPRESS OR IMPLIED WARRANTIES, INCLUDING, BUT NOT LIMITED TO, THE IMPLIED

WARRANTIES OF MERCHANTABILITY AND FITNESS FOR A PARTICULAR PURPOSE ARE

DISCLAIMED. IN NO EVENT SHALL THE COPYRIGHT OWNER OR CONTRIBUTORS BE LIABLE FOR

ANY DIRECT, INDIRECT, INCIDENTAL, SPECIAL, EXEMPLARY, OR CONSEQUENTIAL DAMAGES

(INCLUDING, BUT NOT LIMITED TO, PROCUREMENT OF SUBSTITUTE GOODS OR SERVICES;

LOSS OF USE, DATA, OR PROFITS; OR BUSINESS INTERRUPTION) HOWEVER CAUSED AND ON

ANY THEORY OF LIABILITY, WHETHER IN CONTRACT, STRICT LIABILITY, OR TORT

(INCLUDING NEGLIGENCE OR OTHERWISE) ARISING IN ANY WAY OUT OF THE USE OF THIS

SOFTWARE, EVEN IF ADVISED OF THE POSSIBILITY OF SUCH DAMAGE.

Після того, як ви реалізували пакет, ви можете опублікувати його на pub.dev, щоб інші розробники могли легко використовувати його.

Перед публікацією обов’язково перегляньте файли pubspec.yaml, README.md і CHANGELOG.md, щоб переконатися, що їх вміст повний і правильний. Крім того, щоб покращити якість і зручність використання вашого пакета (а також збільшити ймовірність отримати статус Flutter Favorite), подумайте про додавання таких елементів:

- Різноманітні приклади використання коду
- Скріншоти, анімовані gif-файли або відео
- Посилання на відповідне сховище коду

Далі запустіть команду public у режимі сухого запуску, щоб перевірити, чи все проходить аналіз:

flutter pub publish --dry-run

Наступним кроком є публікація в pub.dev, але будьте впевнені, що ви готові, оскільки публікація назавжди:

flutter pub publish

Після успішного завантаженя свого плагіну до pub dev на платформі з’явиться ваш плагін після чого розробники зможуть встановлювати його для користування у своїх проєктах. Також кожен може відкривати так званні issues на github або gitlab в залежності куди був завантажений плагін . Це допоможе з розвитком плагіну та як і будь якого розробника можуть бути недоробки або як баги , що інші розробники знайшли у коді і буде можливість виправити їх та завантажити фікс з новою версією.[40]

ВИСНОВКИ

Від початку проробленої роботи було поставлено мету: розробити інформаційну систему для аналізу мобілного додатку , що спростить розробку для інших розробників та надасть можливість відстежувати кілкість часу що провів користувач на екрані та загалом у додатку.

Актуальність цього полягає в тому, що Flutter має дуже велике комьюніті та дуже стрімко рухається вперед. І зазвичай розробникам потрібно писати самостійно функціонал, що вже реалізований в цьому плагіні.

Для того , щоб розпочати роботу над плагіном було проаналізовано аналоги, актуальність питання. Плагін буде корисним для великої кількості розробників для пришвидшення розробки, та аналітикам, які зможуть аналізувати дані що зібрав плагін та використовувати іх на власний россуд.

Для плагіну було досліджено мову програмування Dart та фреймворк Flutter на більш низькому рівні аніж його використовують зазвичай. Зроблено деплой у публічний доступ для всіх бажіючих спробувати його у своєму додатку .

СПИСОК ЛІТЕРАТУРИ

1. Що таке плагін. URL: https://a2os.org.ua/12532/що-таке-плагін/
2. Чому крос-платформа для мобільних додатків URL: https://androidas.ru/nativ-ili-krossplatforma-chto-vybrat-nachinayushchemu-mobilnomu/
3. Що таке крос-платформа URL : https://www.thefastcode.com/uk-uah/article/what-does-cross-platform-mean-for-gaming-and-other-apps-?
4. Плагін на мові Flutter. URL: https://vc.ru/dev/286928-mobilnoe-prilozhenie-na-flutter-plyusy-i-minusy-dlya-biznesa
5. Головне для аналізу мобільного додатку URL : https://www.marketing-ua.com/article/10-golovnih-metrik-dlya-analitiki-mobilnogo-dodatka/
6. Мобільні та Веб додатки URL: https://qalight.ua/baza-znaniy/mobilnij-ta-veb-dodatok-u-chomu-rizniczya/
7. Dart language. URL: https://dart.dev/
8. Про Флаттер URL : https://avada-media.ua/ua/services/flutter/
9. Flutter SDK. URL: https://documentation.mapp.com/1.0/en/flutter-sdk-19109838.html
10. Ідеальний для MVP URL : https://ukrfire.com/what-is-flutter-and-why-it-is-worth-learning/
11. DOU про Flutter URL : https://dou.ua/lenta/articles/flutter-for-mobile-apps/
12. Get X. URL: https://pub.dev/packages/get
13. OvarlaySupport URL : https://pub.dev/packages/overlay_support
14. CarouselSlider URL : https://pub.dev/packages/carousel_slider
15. SyncfusionFlutterCharts URL : https://pub.dev/packages/syncfusion_flutter_charts
16. MIME URL: https://pub.dev/packages/mime
17. Logger URL : https://pub.dev/packages/logger
18. File Picker URL : https://pub.dev/packages/file_picker
19. Url Strategy URL : https://pub.dev/packages/url_strategy
20. Url Launcher URL : https://pub.dev/packages/url_launcher

21. Path Povider URL : https://pub.dev/packages/path_provider
22. Android Path Povider URL : https://pub.dev/packages/android_path_provider
23. Package Info Plus URL : https://pub.dev/packages/package_info_plus
24. Flutter Local Notifications URL : https://pub.dev/packages/flutter_local_notifications
25. Flutter Lints URL : https://pub.dev/packages/flutter_lints
26. Flutter Launcher Icons Maker URL : https://pub.dev/packages/flutter_launcher_icons_maker
27. Flutter Focus Widget URL : https://pub.dev/packages/focus_widget
28. Flutter Build Runner URL : https://pub.dev/packages/build_runner#usage
29. Flutter Vibration URL : https://pub.dev/packages/vibration
30. Flutter Shared Preferences URL : https://pub.dev/packages/shared_preferences
31. Local Auth URL : https://pub.dev/packages/local_auth
32. Firebase. URL: https://firebase.google.com/
33. Що таке Flurry Analytics.URL: https://coba.tools/flurry-analytics
34. Що таке AppMetrica URL : https://www.slideshare.net/alexandersibrikov/appmetrica
35. Що таке AppFlyer URL : https://www.appsflyer.com/company/about/
36. Що таке App Annie. URL: https://www.bloggersideas.com/ru/app-annie-review/
37. Android Studio. URL: https://developer.android.com/studio
38. GitLab Docs. URL: https://about.gitlab.com/
39. Переваги GitLab URL : https://production-ready.dev/2022/07/gitlab-tse/
40. Створення та деплой плагіну. URL: https://docs.flutter.dev/development/packages-and-plugins/developing-packages