ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської роботи
на ступень вищої освіти бакалавр

на тему «**СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КОМУНІКАЦІЇ МІСЦЕВОЇ ВЛАДИ З МЕШКАНЦЯМИ МІСТА НА ФРЕЙМВОРКУ REACT NATIVE»**

Виконав: студент 4 курсу, групи ПД-42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності)
Власенко Іван Юрійович
(прізвище та ініціали)
Керівник Гаманюк І.М.
(прізвище та ініціали)
Рецензент
(прізвище та ініціали)

КИЇВ – 2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально–науковий інститут Інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти «Бакалавр»
Спеціальність підготовки 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____О.В.Негоденко
“____” _____ 2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУЮ РОБОТУ СТУДЕНТУ

Власенку Івану Юрійовичу
(прізвище, ім’я, по батькові)

1.Тема роботи: «Створення мобільного застосунку для комунікації місцевої влади з мешканцями міста на фреймворку React Native»

Керівник роботи Гаманюк Ігор Михайлович, с.в.
(прізвище, ім’я, по батькові, науковий ступінь, вчене звання)

затверджені наказом вищого навчального закладу від «12» березня 2021 року №65

2. Строк подання студентом роботи «1» червня 2021року

3. Вихідні дані до роботи: Мобільний застосунок для комунікації влади з мешканцями міста, методи та засоби проєктування та розробки програмного забезпечення, принципи побудови мобільних додатків.

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

4.1 Огляд предметної області
4.2 Вибір засобів реалізації
4.3Проєктування застосунку
4.4 Реалізація застосунку

5. Перелік графічного матеріалу (презентація)

1.
2.
3.
4.

6. Дата видачі завдання__1.06.2021__

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської Роботи	Строк виконання етапів роботи	Примітка
1	Отримання завдання на бакалаврську роботу	19.04.21	
2	Огляд предметної області	20.04.21	
3	Вибір інструментальних засобів реалізації	25.04.21	
4	Проєктування застосунку	28.04.21	
5	Реалізація веб-застосунку	30.04.21	
6	Тестування веб-застосунку	06.05.21	
7	Написання та оформлення пояснювальної записки	10.05.21	
8	Розробка графічних та презентаційних матеріалів	15.05.21	
9	Захист бакалаврської роботи	01.06.21	
10			

Студент _____ І.Ю. Власенко
(підпис) (прізвище та ініціали)

Керівник роботи _____ І.М. Гаманюк
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи: 52с., 30 рис, 1 дод., 1 табл., 22 джерела.

REACT NATIVE, EXPO, FLUX, GIT, FRAMEWORK, FIREBASE

Об’єкт дослідження – процес комунікації місцевої влади.

Предмет дослідження – комунікація влади з мешканцями міста за допомогою застосунку.

Мета роботи – розробка мобільного додатку та поліпшення комунікації влади з мешканцями міста.

Методи дослідження – уніфікований процес розробки програмного забезпечення.

У роботі було:

Проведено аналіз існуючих рішень розробки мобільних додатків, їх переваги та недоліки. Обраний оптимальний засіб розробки мобільного додатку, а саме фреймворк для Java Script React Native.

Були визначені вимоги та функціональні можливості мобільного додатку “Rozmova”.

Був проведений аналіз програмних засобів для React Native. Обраний оптимальний стек технологій, який забезпечує швидку розробку, та перевикористання коду.

Ініціалізація проєкту, установка та налаштування обраних бібліотек. Створення та налаштування бази даних на основі FireStore. Налаштування авторизації за допомогою FireBase. Розроблення діаграм діяльності, варіантів використання та пакетів. Проведено аналіз частин проєкту, та розроблена архітектура додатку.

Результат даної роботи полягає в можливості впровадження цього додатку в життя мешканців міста. Особливість даного сервісу це налагодження комунікації органів влади з громадськістю.

ЗМІСТ

ВТУП

Сучасне суспільство неможливо уявити без інформаційних технологій, які поступово проникають у всі сфери життя суспільства та держави. Засоби спілкування в Інтернет-просторі сьогодні особливо популярні, оскільки вони є найшвидшими та найзручнішими для створення, зберігання, передачі та обміну інформацією. Завдяки всім цим можливостям, взаємодії між суб’єктами стають все більш поширеними серед громадян, а традиційні засоби комунікації, такі як друк, радіо та навіть телебачення, відходять на другий план.

Важливо відзначити, що криза політичних інститутів призводить до того, що молодь все більше переорієнтовується на нові канали комунікації та форми активності, які не можна не брати до уваги. Тому органам влади важливо розвивати комунікацію з громадянами через Інтернет, створювати онлайн-майданчики з обговорення актуальних проблем, вести інформування і комунікацію через соціальні мережі, бо це дозволяє залучити молодь до суспільно-політичної активності.

Важливо відзначити, що Інтернет розглядається дослідниками як:

1. Сукупність технологій і інструментів, які полегшують взаємодію суб'єктів, дозволяє їм формувати мережі.
2. Середовище, в якому формуються нові форми участі, колективної дії.
3. Нове середовище, технологія, яка дозволяє забезпечити масову участь громадян в процесі прийняття рішень, висловленні своїх переваг і потреб.

У зв'язку з цим він одночасно є середовищем і інструментом здійснення комунікації. Цифрові технології прискорили комунікації і збільшили можливості охоплення аудиторії в просторі та часі.

Тому створення мобільного додатку для комунікації влади з людьми є актуальна задача, яка дає можливість на нову сучасну комунікацію з владою.

Об’єкт дослідження – процес комунікації місцевої влади.

Предмет дослідження – комунікація влади з мешканцями міста за допомогою застосунку.

Мета роботи – розробка мобільного додатку та поліпшення комунікації влади з мешканцями міста.

Методи дослідження – уніфікований процес розробки програмного забезпечення.

Наукова новизна даної роботи полягає у створенні інструмента комунікацій влади

1 МЕТОДИ РОЗРОБКИ МОБІЛЬНИХ ЗОСТАСУНКІВ

1.1 Мобільна розробка на сьогоднішній час

Маючи понад 3,2 мільярда користувачів смартфонів у всьому світі, не дивно, що індустрія мобільних додатків процвітає. Використання додатків та проникнення смартфонів все ще стабільно зростають, без будь-яких ознак уповільнення в осяжному майбутньому.

Дослідження показують, що середньостатистичний американець перевіряє свій телефон кожні 12 хвилин. 10% людей перевіряють свої телефони раз на чотири хвилини.

Ключова статистика щодо мобільних застосунків :

1) Очікується, що до 2023 року мобільні додатки принесуть понад 935 млрд доларів.

2) Apple App Store має 1,96 мільйона додатків, доступних для завантаження.

3) У магазині Google Play можна завантажити 2,87 мільйона додатків.

4) 21% Millennials відкривають додаток 50+ разів на день.

5) 49% людей відкривають додаток 11+ разів на день.

6) Середній власник смартфона використовує 10 додатків в день і 30 додатків кожен місяць.

Щороку ми спостерігаємо зростання кількості завантажень додатків рис 1.1.1. Ця тенденція збережеться і в найближчі роки. Минулого року було завантажено понад 204 мільярди додатків. Це приблизно на 6% більше, ніж у попередньому році. Як видно з графіку рис.1.1.1 , між 2016 і 2019 роками відбувся стрибок на 45%. Темпи зростання за рік не настільки різкі, але зростають.

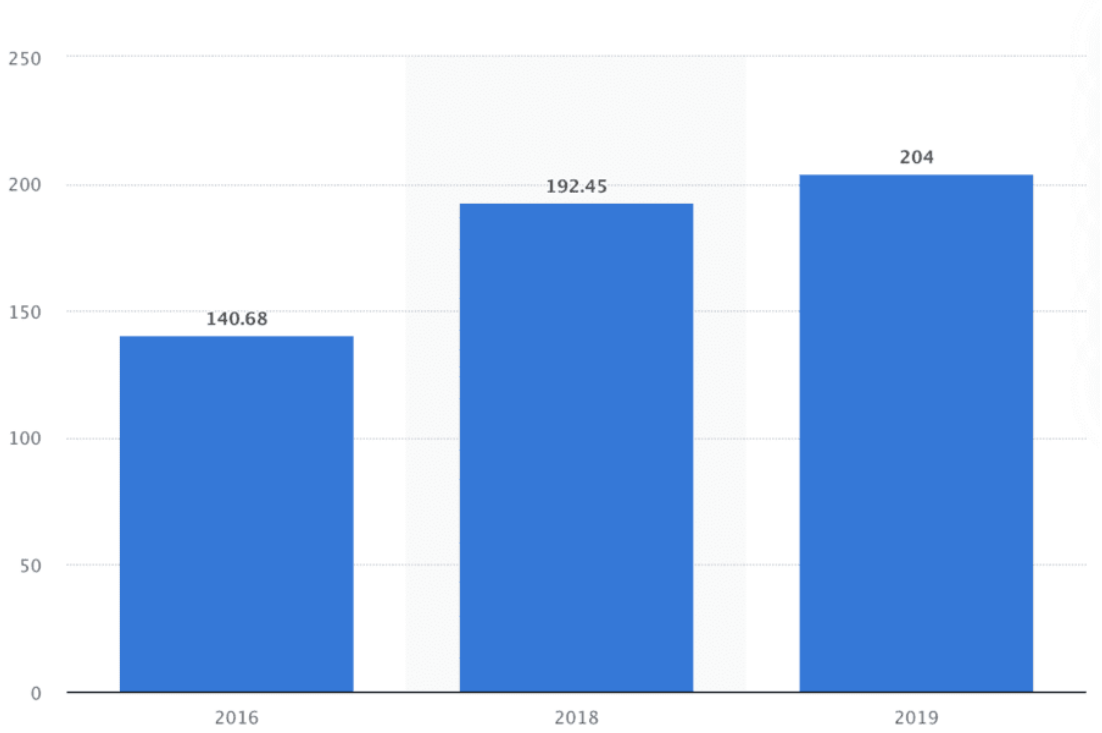


Рисунок 1.1.1 – Графік зростання завантажень

1.2 Види мобільної розробки

Для того, щоб додаток вирішував покладені на нього завдання, його слід грамотно розробити та реалізувати. На ринку мобільної розробки є різні пропозиції щодо створення мобільних додатків. Конструктори додатки, придатні для вирішення простих завдань, між платформні рішення та розробка в «нативному» середовищі. Давайте детальніше розглянемо кожен із методів створення мобільних додатків.

1.2.1 Конструктори додатків

Попри всі переваги, у будь-якого мобільного додатка є один істотний недолік - вартість його розробки. Самостійна розробка програмного забезпечення - все ще складний процес. До того ж такий товар дорожче, ніж сайт. Однак є інструменти,

що дозволяють створити мобільний додаток, не витрачаючи великих коштів. Їх називають конструкторами мобільних додатків[1].

Переваги:

1) Немає потреби в навичках програмування.
2) Немає потреби в навичках програмування.
3) Швидкість розробки додатків.
4) Готова архітектура.

Недоліки:

1) Дизайн шаблону.
2) Немає можливості редагувати структуру та код.
3) Прив'язка до певної платформи.
4) Товар ваш, але частково.
5) Труднощі з розширенням.

1.2.2 Нативна розробка

Нативні додатки - це мобільні додатки, розроблені спеціально для конкретної мобільної операційної системи[2].

Зараз на ринку IT-розробки провідні компанії пропонують створення мобільних додатків в нативних середовищ кожної платформи. Додатки, створені в нативному середовищі, являють собою продукти, що максимально відповідають вимогам кожної конкретної платформи, її рекомендованим стандартам. У таких додатках можна без обмежень використовувати всі функціональні можливості, що надаються iOS або Android. Такі програми не вступають в конфлікт на різних пристроях, відрізняються високою стабільністю роботи та високою якістю для користувача.

Нативні додатки легше допрацьовувати та функціонально розширювати при виході нових версій iOS або Android. Такі додатки легше просувати в Play Market і App Store, підтримувати та розвивати за бажанням власника. Зараз багато компаній віддають перевагу саме «нативної» розробки завдяки її перевагам щодо якості одержуваних додатків.

Переваги:

1) Продуктивність додатка. Оскільки нативні додатки розробляються спеціально для певної операційної системи, вони набагато швидше і досконаліше, ніж їх інші аналоги. Продуктивність робить Native Apps кращим вибором для високопродуктивних ігор і додатків.

2) UX. Користувачі вибирають Android або iOS в залежності від своїх індивідуальних характеристик. Отже, якщо хтось використовує Android протягом тривалого часу, у нього не виникне проблем з доступом до додатка, яке відповідає стандартам призначеного для користувача інтерфейсу для конкретної платформи. Таким чином, він став дуже зручним для користувача. Те ж саме стосується користувачів iOS.

3) Доступність додатків. Нативні додатки можуть легко залучити будь-які функції пристрою, такі як камера, мікрофон, календар, GPS, датчики друку фігур, датчик руху. Таким чином, доступність додатка дуже висока.

Недоліки:

1) Довший час розробки. Оскільки власний додаток орієнтоване на конкретну ОС, йому потрібно більше часу на розробку для кожної окремої платформи. Тому організатори повинні писати спеціальні коди для iOS, Android і т. Д. У результаті, щоб надати якісний продукт, загальний час розробки збільшується.

2) Дорого. Оскільки цикл розробки довше, нативні додатки трохи дорожче в порівнянні з іншими гібридними або вебдодатками.

3) Підтримка. Оскільки користувачі різних пристроїв можуть використовувати різні версії додатка, розробникам складно підтримувати.

1.2.3 Гібридні додатки

Гібридні додатки в основному являють собою комбінацію нативних і веб додатків. Ця програма, в якому один і той же додаток може бути розгорнуто в різних операційних системах, таких як Android, iOS, Windows і т. д., І навіть може працювати в різних браузерах (Chrome, Mozilla, IE, Safari тощо)[3].

Гібридні додатки є кросплатформені і можуть поширюватися між магазинами додатків відразу без кількох версій. Більшість гібридних додатків створено з використанням крос-сумісних веб технологій (HTML5, CSS3, jQuery, Titanium, PhoneGap, JavaScript тощо).

Переваги:

1) Витрати. Один з найбільших плюсів гібридного додатка - це найнижча вартість розробки. Оскільки ви не збираєтеся створювати кілька версій для різних магазинів додатків, цикл розробки також буде менше.

2) Легке масштабування. Гібридні додатки дуже легко масштабувати до різних платформ і ОС. Це пов'язано з тим, що веб технології майже на 100% схожі на різних платформах. Таким чином, код можна просто повторно використовувати без необхідності перебудовувати все додаток з нуля.

3) Швидкий вихід на ринок. Оскільки цикл розробки менше, і ми вже заощадили час, написавши лише один раз, потрібний проміжок часу запуску гібридних додатків менше. Додатки проходять перевірку якості, як стандартне програмне забезпечення або веб-сайт. Первісне тестування можна виконати з веб браузер. Поширення в магазині додатків і тестування платформи відбуваються швидко.

Недоліки:

1) UX. Це велика проблема для гібридних додатків. гібридний додаток ніколи не може надати користувачам повністю нативний досвід. Нативні додатки використовують компоненти системного інтерфейсу, які зручні для користувача,

2) можуть створювати значуще рішення і можуть допомогти підтримувати загальний робочий процес.

3) Рідні функції. Ви просто не можете розмістити всі бажані функції в гібридних додатках. І Android, і iOS мають кілька функцій, призначених виключно для їх ОС. Ви ніколи не зможете інтегрувати їх в гібридне мобільний додаток.

1.3 React Native, Vue Native

1.3.1 React Native

React Native - це середовище розробки мобільних додатків, яке дозволяє розробляти багато платформні додатки для Android і iOS з використанням власних елементів призначеного для користувача інтерфейсу[4]. Він заснований на середовищі виконання JavaScriptCore і трансформаторі Babel.

Цей знаменитий фреймворк для розробки мобільних додатків стартував влітку 2013 року як проєкт внутрішнього Хакатону Facebook. Його перша публічна версія була випущена в січні 2015 року на конференції Reactjs, а в березні 2015 року Facebook зробив React Native відкритим і доступним на GitHub.

З того часу він набув широкого поширення серед розробників і організацій через його здатності створювати власні програми та відмінні інтерфейси. На графіку нижче ви можете візуалізувати тенденцію зростання React Native рис. 1.3.1. Всього через 1,5 року після випуску він обігнав розробку для Android і iOS.

Рисунок 1.3.1 - Графік тенденції зростання React Native

Переваги:

1) Економія часу. Основна перевага цієї системи - економія часу. Фреймворк позбавляє розробників від перекомпіляції при кожній зміні, оскільки додаток перезавантажується відразу після редагування коду.

2) Просте виконання. Фреймворк дозволяє створювати єдину базу коду, яку можна змішувати для iOS і Android. Це дозволяє інженеру витрачати менше енергії на кодування.

3) Досвід використання. Мобільний додаток, створене на цій платформі, гарантує високу якість і оптимізований інтерфейс.

4) Єдина екосистема. Крім того, всебічно підготовлений JavaScript-інженер може розробити універсальний додаток, використовуючи цей фреймворк, не заглиблюючись в особливості екосистеми та мови кожної ОС.

5) Швидко. Це означає, що додаток буде швидше завантажуватися і працювати більш плавно, ніж додаток, створене з використанням нативної моделі.

Недоліки:

1) Налагодження. Це складна процедура налагодження програми, створеного на React Native. Вам потрібно буде вивчити спосіб, яким React Native створює код, і вирішити, як з ним діяти.

2) Конфігурація. Іноді для координації локальної бібліотеки всередині програми React Native потрібно безліч налаштувань. Наприклад, начерки Google Maps перетворюються в довгу роботу, в той час, як в додатку Android Native на це йде один момент.

1.3.2 Vue Native

Vue Native - це мобільна платформа для створення нативних мобільних додатків з використанням Vue.js. Він призначений для з'єднання React Native і Vue.js[5].

Vue - це прогресивна структура для побудови користувальницьких інтерфейсів. На відміну від інших монолітних каркасів, Vue розроблений з нуля, щоб його можна було поетапно застосовувати[6].

Переваги:

1) Синтаксис. Код, написаний на Vue Native, краток та ефективний - зрозуміло, багато чого можна досягти з меншим кількістю стрічок коду.

2) З моменту випуску у 2014 році Vue.js створив надійну бібліотеку інструментів. Використовуючи Vue Native, ви отримуєте доступ до всього хорошого, що приховується всередині.

3) Vuex - для управління сховищем додатку.

4) Своя навігація.

5) Інструменті для тестування.

Недоліки:

1) У документації Vue Native відсутня складна та загальна інформація - вам потрібно завантажитись у документацію React Native, щоб знайти відповіді.

2) Vue Native дозволяє використовувати бібліотеки, розроблені для React Native, іноді вам потрібно знати React, щоб адаптувати їх до потреб проєкту Vue Native.

3) Крім того, Vue Native є оболонкою для API-інтерфейсу React Native, вони розділяють всі мінуси.

1.4 Висновки

З урахуванням того, що React Native вирішує більшість відомих мінусів «гібридних» додатків, для розробки мобільного застосування буде використовувати саме цю технологію.

React Native об'єднує плюси «нативних» і «гібридних» додатків. А також дає нам можливість значною мірою перевикористати код при розробці аналогічного додатка для платформи Android.

2 ІДЕЯ ДОДАТКУ РОЗМОВА

Важливою складовою громадянського суспільства є інформатизація, яка зачепила всі сфери суспільства. Вона спрямована на максимальну відкритість і прозорість дій. Інновації в сфері IT дають додаткові можливості в розвитку громадянського суспільства, але правове регулювання в динамічній сфері відстає.

Внаслідок цього виникає протиріччя: з одного боку, в суспільстві спостерігається потреба у впровадженні та використанні інноваційних способів комунікації, але з іншого боку, є відставання організаційно-правового забезпечення цього процесу. У зв'язку з цим зростає значимість вивчення недоробок в цій сфері, а також аналізу позитивного зарубіжного досвіду.

Мобільний додаток Rozmova рис.2.1.1 поміщає владу міста Києва в вашу долоню. Користувачі додатка зможуть швидко і легко повідомляти про проблеми та турботи. Також отримувати актуальну інформацію від комунальних служб, та місцевого управляння.

Рисунок 2.1.1 – мобільний додаток Rozmova

2.1 Основні можливості мобільного-застосунку «ROZMOVA»

У кожного користувача додатку ROZMOVA є свій обліковий запис де є коротка інформація про нього. Це допоможе виконавцям зв'язатись з ним при виконанні його звернення. Також кожен користувач зможе бачити свої звернення перейшовши з профілю на спеціальну сторінку.

В додатку є сторінка з мапою. На цій мапі можливо побачити всі звернення мешканців. Є можливість перейти до звернення та побачити повний текст та фото. У кожного звернення є панель з коментарями. Там можливо надати вашу думку. Також є модальне вікно де можна побачити статус виконання цього звернення. Після того як воно буде виконане, в цьому вікні з’явиться звіт від виконавця. Також е можливість створити своє звернення. Для цього потребується описати проблему, та добивати пару фото. Ваше звернення з’явитися на мапі і всі інші користувачі зможуть це побачити

Головною сторінкою додатку є панель з новинами. Там будуть з’являтись новини від місцевої влади, комунальних служб, та звернення. Також ви зможете коментувати та оцінювати новини.

2.2 Висновки

Комунікація між владою та мешканцями дуже важлива. Саме так влада буде знати що турбує людей. Але зараз немає простого та зручного інструменту для цього. Rozmoma виступить як розв'язання цієї проблеми. Легкий та зрозумілий дизайн в купі зі всім потрібним функціоналом забезпечте розмову між місцевою владою та мешканцями міста.

3 ВИБІР СТЕКУ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ

3.1 Вибір середовища розробки

Для написання коду на React Native зазвичай використовують два середовища розробки. Це Visual Studio Code і Webstorm.

Visual Studio code - це безплатне середовище розробки. Самий популярний редактор коду у світі прямо зараз. Він розроблений Microsoft спільно з великою спільною розробників. Позиціонує себе як "легкий" редактор коду[7].

WebStorm - одна з найвідоміших і глибоких IDE для JavaScript. Він розроблений JetBrains - компанією з багаторічним досвідом створення потужних IDE для різних мов програмування. WebStorm заснований на IntelliJ IDEA, але є версією, спеціалізованої для JavaScript[8].

Visual Studio code володіє широкими можливостями налаштування і розширення, але при цьому створений за допомогою ресурсно міського Electron. Сам додаток досить плавний, працює без будь-яких проблем. Тільки після додавання декількох важких розширень помітно зниження продуктивності.

WebStorm має безліч функцій. Це змушує його працювати повільно і споживати багато ресурсів вашого пристрою. Але оскільки він використовує JVM, він, безумовно, трохи швидше та оптимізований, ніж будь-який додаток Electron.

Найбільша особливість Visual Studio code - це розширюваність. Додаток має 10K розширень. Без розширень vs code майже не відрізняється від простих редакторів коду таких як sibline, notepad ++.

WebStorm має майже все вбудоване. Практично всі популярні JavaScript-фреймворки, інструменти та бібліотеки мають своє місце в WebStorm.

Подивившись на самі популярні середовища розробки для React Native було вибрано WebStorm. Воно дає всі необхідні можливості для створення коду на java script.

3.2 Вибір фреймворку

В офіційній документації React Native ми дізнаємося про те, що у нас є два шляхи запуску мобільного додатка: expo і react-native cli табл. 3.2.1.

Expo - це фреймворк для React Native. Expo надає шар поверх команд API React Native, щоб полегшити їх використання та управління. Він також надає інструменти, які полегшують початок створення та тестування програм React Native. Також, він надає компоненти та послуги інтерфейсу, які зазвичай доступні лише при встановленні сторонніх компонентів React[9]. Усі вони доступні через Expo SDK.

Таблиця 3.2.1 – Порівняння React Native Init з Expo

Найменування	**react-native init**	**Expo**
Ви можете додати власні модулі, написані на Java / Objective-C	+	-
Вага стандартного додатка Hello World	5-мб	25-мб
Потрібно Android Studio і XCode для запуску проєкт	+	-
Шрифти необхідно імпортувати вручну в XCode	+	-
Спільне використання додатка (за допомогою QR-коду або посилання)	Важче	Легше
Якщо ви хочете поділитися цим додатком, вам потрібно відправити весь файл .apk / .ipa	+	-
Надає JS API з коробки, наприклад Push-Notifications, Asset Manager	-	+

Також Expo надає послуги зі збіркою додатку. Оскільки всі додатки Expo використовують один і той же власний код, Expo може легко створювати ці додатки для вас. Вони створили сервіс хмарної збірки.

Expo - це дійсно хороший спосіб розпочати роботу, оскільки він позбавляє від необхідності встановлювати багато програмного забезпечення. Це дозволяє легко розробляти і публікувати додатки

3.3 Вибір підходу програмної архітектури застосунку

3.3.1 Flux

Flux представляє архітектуру програм, що використовують React. Flux- це скоріше патерн, ніж конкретний фреймворк[10].

Додатки Flux складаються з трьох основних частин: dispatcher, store та view рис. 3.3.1.

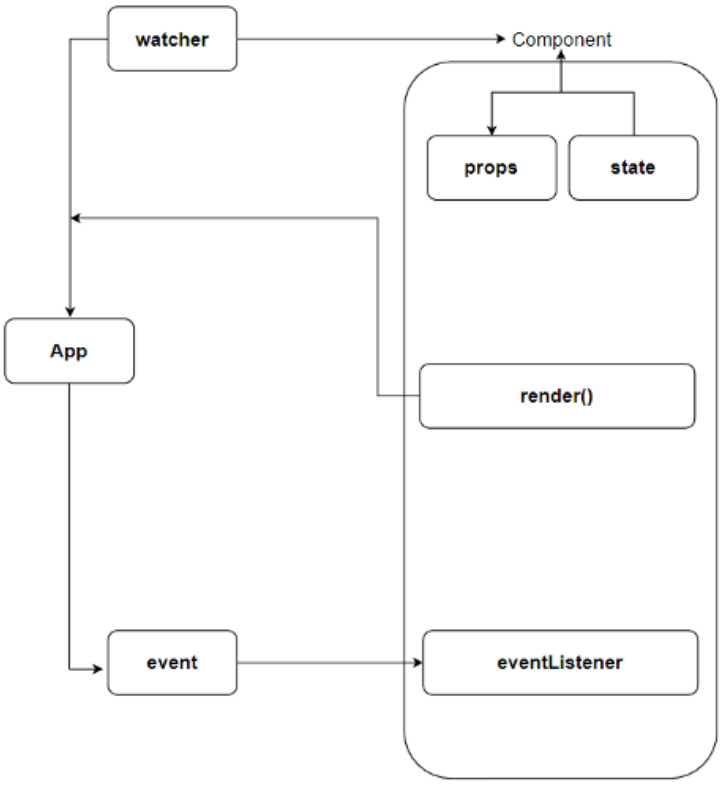


Рисунок 3.3.1 – Архітектура Flux

Dispatcher представляє головну точку у всій цій схемі, яка контролює потік даних у програмі Flux. Dispatcher реєструє stores та їх зворотні дзвінки. Коли диспетчер отримує якісь дії ззовні, диспетчер повідомляє ці сховища про вхідну дію через зворотні виклики сховищ.

Store містить стан програми та її логіку. Кожен окремий store управляє певною областю або доменом додатку.

Як описано вище, кожен магазин реєструється у диспетчера разом із його зворотними викликами. Коли диспетчер отримує дію, він викликає зворотний виклик, передаючи вхідну дію як параметр. Залежно від типу дії всередині store викликається той чи інший метод, при якому стан магазину оновлюється. Після оновлення магазину генерується подія, яка вказує на те, що магазин оновлено. І завдяки цій події представлення даних (тобто компоненти React) знають, що store оновлений, і вони самі оновлюють свій стан.

View прикрашають візуальну частину програми. Спеціальний вид подання - view-view представляє компонент верхнього рівня, який містить усі інші компоненти.

Controller-view прослуховує події, що надходять із store. Отримавши подію, controller-view передає дані, отримані з store, іншим компонентам[11].

Основні відмінні риси:

1) Синхронність: всі методи зворотного виклику, зареєстровані для кожної дії, синхронні у виконанні, саме ж дія може викликатися джерелом асинхронно.

2) Інверсія управління: потік управління передається відповідному сховищу і цільової функції зворотного виклику.

3) Семантичні дії: дія, що викликається джерелом, містить смислову інформацію, що дозволяє відповідному сховищу вибрати правильний метод виконання.

3.3.2 Redux

Зараз однією з найпопулярніших реалізацій архітектури Flux є бібліотека Redux. Крім того, Facebook пропонує програмний модуль під назвою Flux, який реалізує, серед іншого, диспетчер для використання з ReactJS.

На відміну від Flux, Redux не має концепції диспетчера. Це пов’язано з тим, що він покладається на чисті функції, а не на генератори подій. Чисті функції легко створювати. Вони не потребують додаткової сутності для управління ними. Flax часто можна описати як (state, action) => state. Таким чином, Redux наслідує архітектуру Flux[12], але робить її простішою завдяки чистим функціям рис. 3.3.2.

Інша важлива відмінність від Flux полягає в тому, що Redux припускає, що ви ніколи не змінюєте свої дані безпосередньо. Ви можете легко використовувати прості об’єкти та масиви для стану, але настійно рекомендується змінювати їх за допомогою редукторів. Ви завжди повинні повертати новий об'єкт, що простіше зробити за допомогою синтаксису розподілу об'єктів, запропонованого ES7 та реалізованого в Babel.

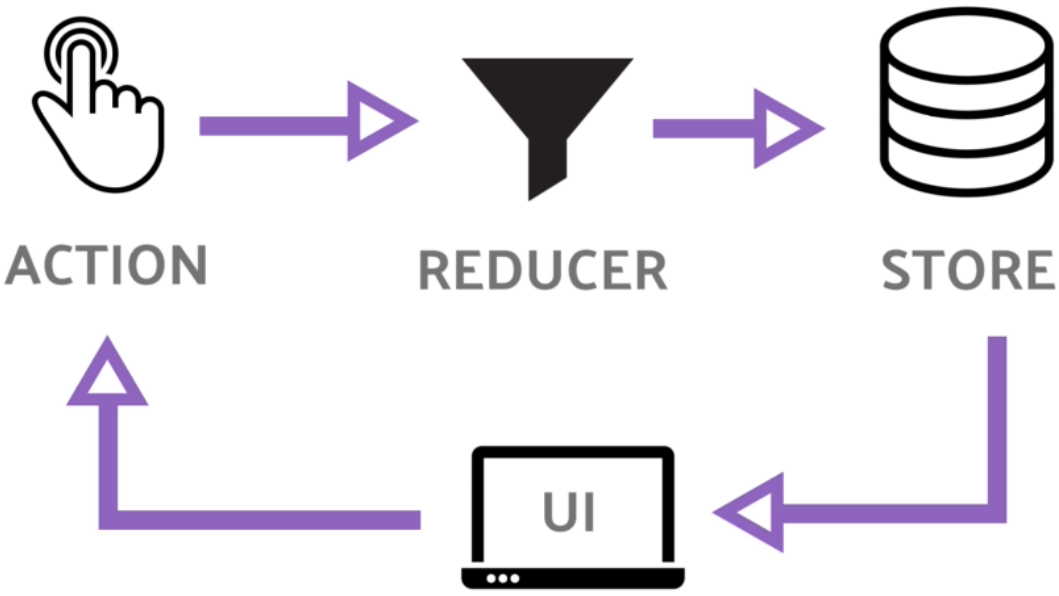


Рисунок 3.3.2 – Концепція Redux

Є 3 основних принципи Redux:

1) Єдине джерело істини. Ми повинні зберігати тільки один об'єкт стану в нашому додатку, ні більше, ні менше.

2) Стан доступно тільки для читання. Ми ніколи не повинні змінювати наш об'єкт стану на абсолютно новий. Замість цього ми повинні змінювати тільки ту інформацію, яку необхідно змінити.

3) Зміни над state робляться чистими функціями. Це означає, що функції редьюсера повинні бути в формі чистої функції, яка приймає поточний state і те, який action вона буде виконувати, і в результаті зробить дію над state.

3.3.3 Redux thunk

За замовчуванням дії Redux обробляються синхронно, що представляє проблему для додатків, яким потрібна взаємодіяти з зовнішніми API або використовувати побічні ефекти. Redux також дозволяє використовувати проміжне ПО між оброблюваних дією і дією, яке досягає редукторів.

Redux Thunk - це проміжне ПЗ, що дозволяє викликати action creator, які повертають функцію замість об'єкта дії. Ця функція отримує метод обробки store, який потім використовується для обробки регулярних синхронних дій всередині тіла функції після виконання асинхронних операцій[13].

3.4 Вибір node moduls

React native має довгий список сторонніх бібліотек, які роблять його ще більш зручним і прискорюють розробку. Для розробки мобільного додатку “ROZMOVA” я обрав такі інструменти:

Axios - це полегшений HTTP-клієнт для JavaScript, створений для надсилання асинхронних HTTP-запитів до кінцевих точок REST та виконання CRUD-операцій.

React-native-firebase - Це легкий шар над власними бібліотеками Firebase. React-native-firebase спрощує використання Firebase з React Native, будуючи міст JavaScript до власних наборів для розробки програмного забезпечення JavaScript.

Formik - це невелика бібліотека для створення форм у React. Formik сприяє побудові форм і дозволяє отримувати значення і поза станом форми, перевіряти та отримувати повідомлення про помилки та ефективно відправляти форми

react-native-vector-icons - ця бібліотека складається з великої кількості векторних картинок, призначених для проєкту у React Native. Кожен елемент повністю настроюється.

Moment - ця бібліотека була розроблена для роботи з різними форматами даних та має можливість аналізу, обробки та перевірки дат та часу в JavaScript

React Navigation - ця бібліотека допомагає легко налаштовувати екрани додатка. Це простий у використанні інструмент навігації на основі JavaScript. Бібліотека навігації React є повністю настроюваною та розширюваною.

react-native-maps - надає компонент Map, який використовує Карти Apple або Google Maps на iOS та Google Maps на Android.

3.5 Використання інструментів для контролю версій Git та GitLab

Система керування версіями - це категорія програмних засобів, яка допомагає реєструвати зміни, внесені у файли, відстежуючи зміни які були внесені в код.

Системи контролю версій дозволяють декільком розробникам, дизайнерам і членам команди працювати разом над одним проєктом. Ці системи мають вирішальне значення для забезпечення доступу кожного до останньої версії коду.

Git - найбільш часто використовувана система контролю версій. Git відстежує зміни, які ви вносите у файли, тому у вас є запис про те, що було зроблено, і ви можете повернутися до певних версіями, якщо вам коли-небудь знадобиться[14]

рис. 3.5.1. Git також спрощує спільну роботу, дозволяючи об'єднати зміни, внесені декількома людьми, в один джерело.

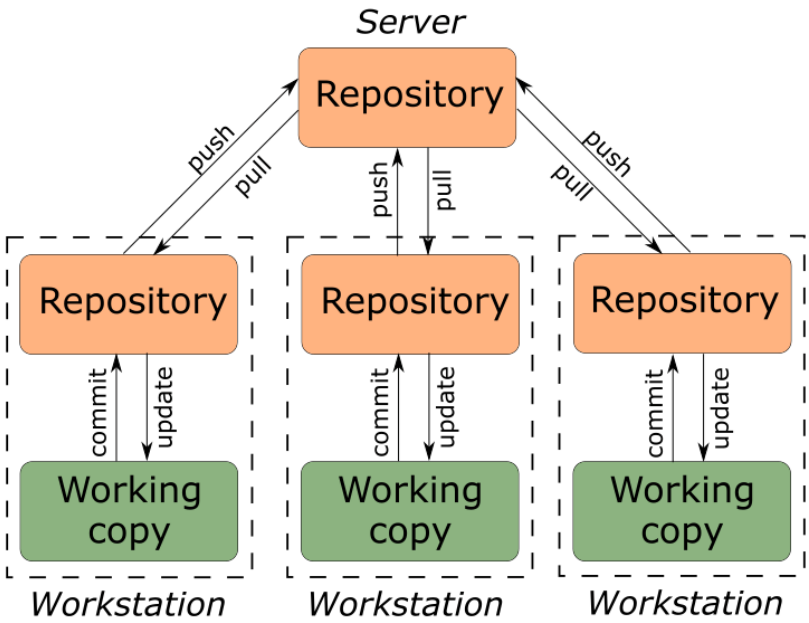


Рисунок 3.5.1 – Модель Git

GitLab - це сервіс, схожий на GitHub, який забезпечує внутрішнє веб управління DevOps репозиторіями Git. GitLab пропонує два варіанти: безплатну версію для спільноти та платну корпоративну версію. GitLab міститись безліч функцій, необхідних для успішного управління процесом розробки програмного забезпечення, таких як Jira integration, CI runner, release management, binary attachments, та багато іншого.

3.6 Створення дизайну при застосуванні Figma

Різниця між хорошим додатком і поганим додатком зазвичай полягає в якості його взаємодії з користувачем (UX). Хороший UX - це те, що відрізняє успішні програми від невдалих. Для проєктування гарного UX допомагає програма Figma

Figma - це програма для інтерфейсу користувача та інтерфейсу UX з чудовим інструментом проєктування, прототипування та генерації коду[15]. Зараз це провідний в галузі інструмент проєктування інтерфейсів із надійними функціями що працюють на кожному етапі процесу проєктування рис. 3.6.1.

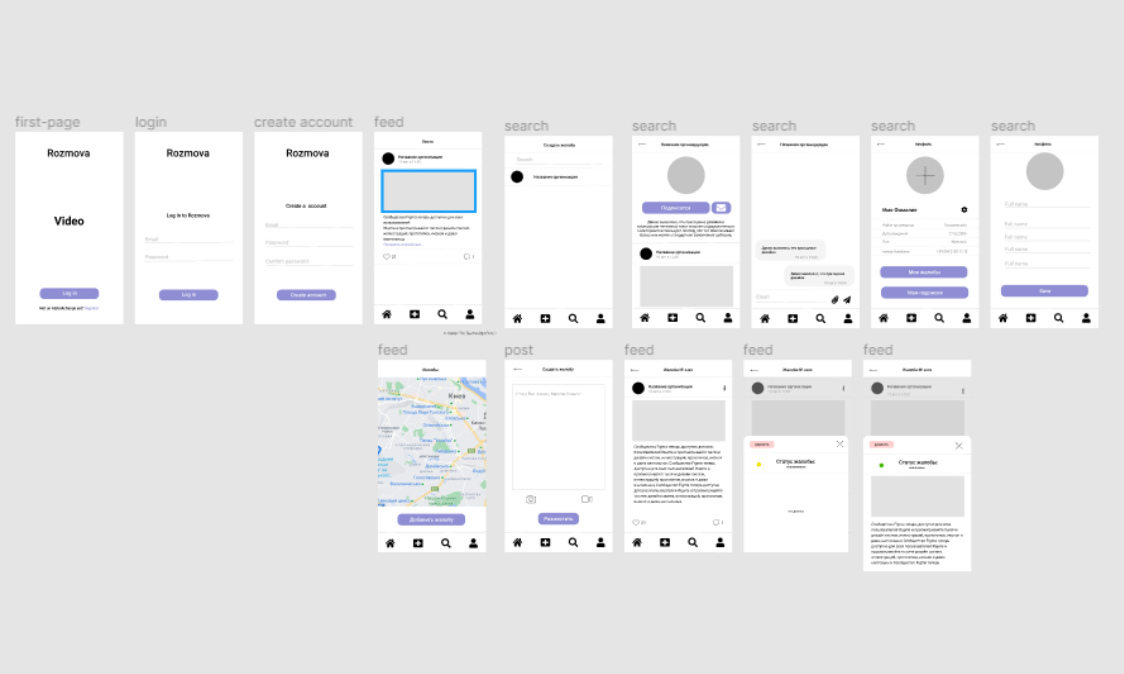


Рисунок 3.6.1 – Дизайн додатку у Figma

3.7 Висновки

В даному розділі проведений аналіз та обрані інструменти для роботи з React Native. Як засіб доступу до API буде використовуватись бібліотека Axios. Був обраний архітектурний підхід до реалізації призначеного для користувача інтерфейсу - Redux, та redux-thunk для реалізації асинхронних операцій в redux. Для управління навігацією буде застосована бібліотека React Navigation. Для створення, та управляння форм буде використана бібліотека Formik. Для реалізації карт у застосунку буде використана native-map. Для контролю версії, та зберігання коду буде використана GitLab. Дизайн розроблений у Figma.

На реалізацію цього додатку було вибрано оптимальний стек технологій, який забезпечує швидку розробку, та перевикористання коду.

4 РЕАЛІЗАЦІЯ

4.1 Ініціалізація додатку

4.1.1 Установка React Native (Expo)

Як вже було написано Expo - це фреймворк і платформа для універсальних програм React. Це набір інструментів і сервісів, створених на основі React Native і нативних платформ, які допоможуть вам розробляти, створювати, розгортати і швидко виконувати ітерацію в iOS, Android і вебдодатках з однієї і тієї ж кодової бази JavaScript / TypeScript.

Для того, щоб ініціалізувати наш проєкт за допомогою Expo, нам потрібно установити Expo CLI.

Expo CLI - це програма командного рядка, яка є основним інтерфейсом між розробником та інструментами Expo[16].

Для того зоб установити Expo CLI нам потрібно прописати в терміналі “yarn add expo-cli”. Далі прописуємо команду створення додатку “expo init rozmova” де "rozmova” це назва додатку.

Expo CLI пропонує свої шаблони рис. 4.1.1. :

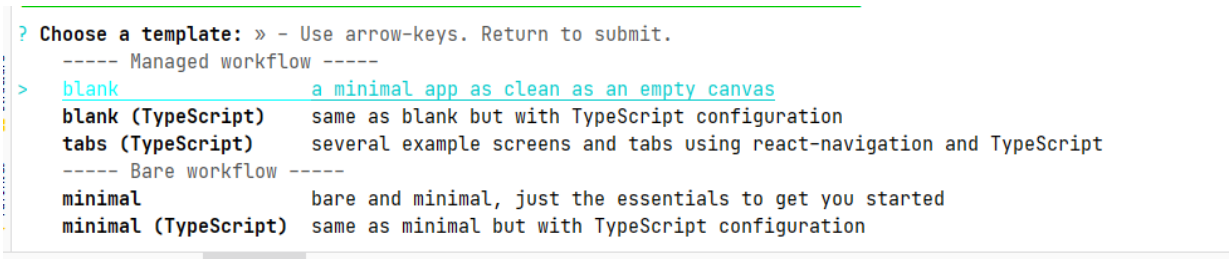


Рисунок 4.1.1 - Шаблони Expo Cli

1) blank - це пустий шаблон проєкту.

2) blank (Type Script) - це пустий шаблон проєкту який написаний на Type Script.

3) tabs (Type Script) - це проєкт написаний на Type Script з настроєною навігацією.

4) minimal - це мінімально настроєний проєкт.

5) minimal () -це мінімально настроєний проєкт написаний на Type Script.

Для свого додатку я обрав пустий проєкт(blank).

4.1.2 Expo developer tool

Проєкт створений за допомогою Expo може працювати у двох режимах: розробці або виробництві.

Режим виробництва мінімізує ваш код і краще відображає продуктивність, яку ваш додаток матиме на пристроях кінцевих користувачів.

Режим розробки включає корисні попередження та надає доступ до інструментів, що полегшують розробку та налагодження.

Для того, щоб запустити проєкт в режимі розробки нам потрібно в консолі написати “expo start”. В браузері відкриється веб додаток Expo develop tool рис. 4.1.2.

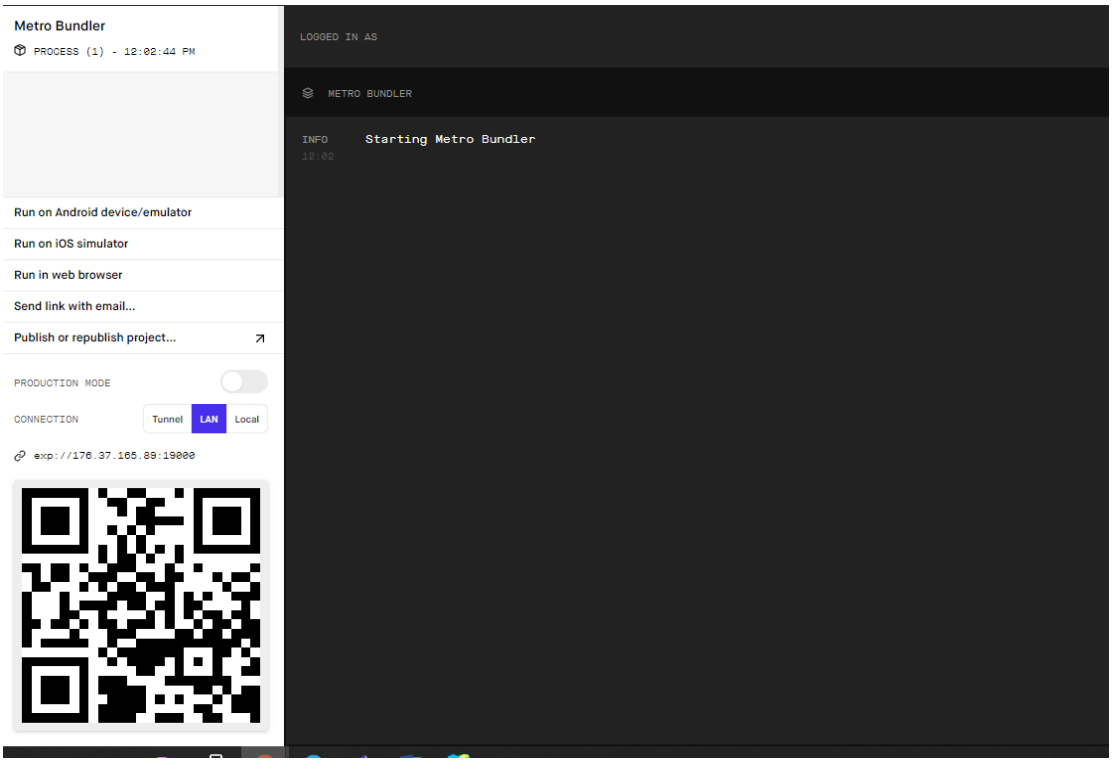


Рисунок 4.1.2 – Expo developer tool

Expo developer tool містити віддалену консоль в якій можливо побачити помилки які виникають процесі розробці. Також є панель вибору запуску проєкт.

За допомогою Expo CLI можливо запустити проєкт на : емуляторі операційної системи андроїд, на мобільному пристрою під управлінням операційної системи андроїд.

Для того, щоб запустити проєкт на мобільному пристрою треба установити додаток Expo Go. За допомогою цього додатку сканувати Qr код. Проєкт відкриється на мобільному додатку.

Також можливо запустити проєкт на емуляторі мобільного пристрою. Для цього я обрав Android Studio. Для того, щоб створити емулятор в Android Studio нам потрібно зайти на SDK Manengment. В цьому вікні нам потрібно установити Adnroid SDK. Далі ми заходимо в ADV Menengment рис. 4.1.3. та створюємо новий пристрій.

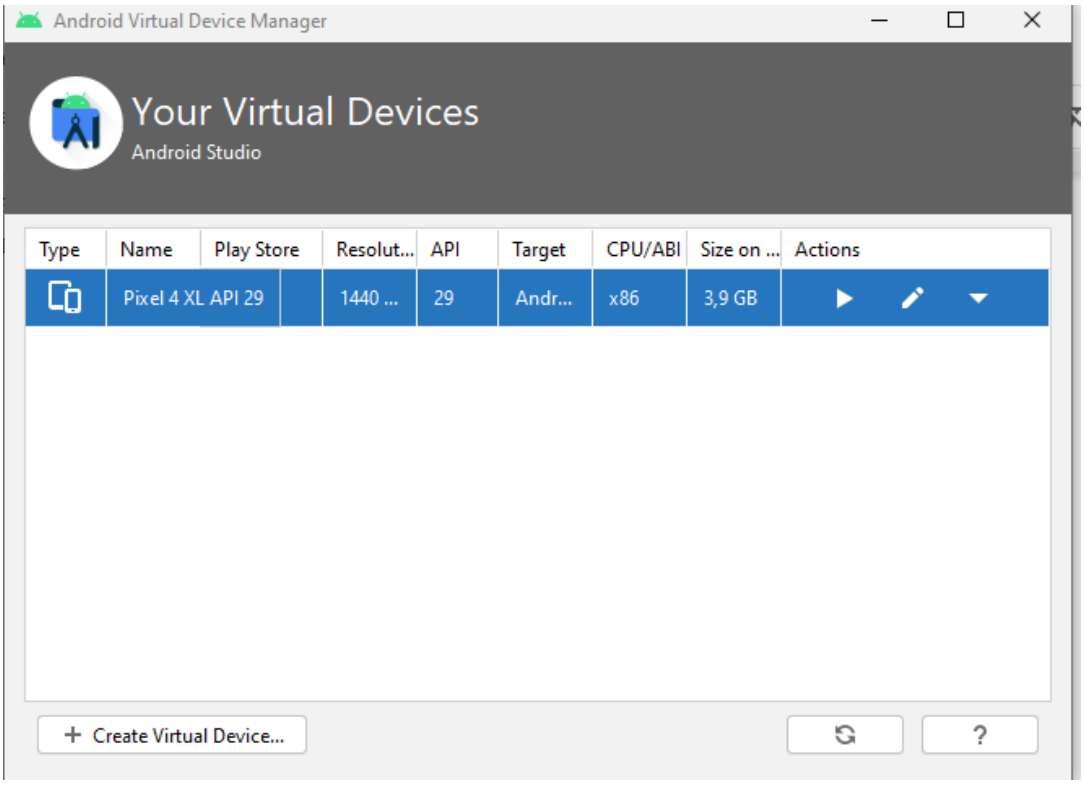


Рисунок 4.1.3 – ADB mamengment

Коли в нас є налаштований емулятор ми можемо запустити проєкт. Для цього запускаємо емулятор андроїд і в Expo developer tool вибираємо запуск за допомогою андроїд емулятора рис. 4.1.4.

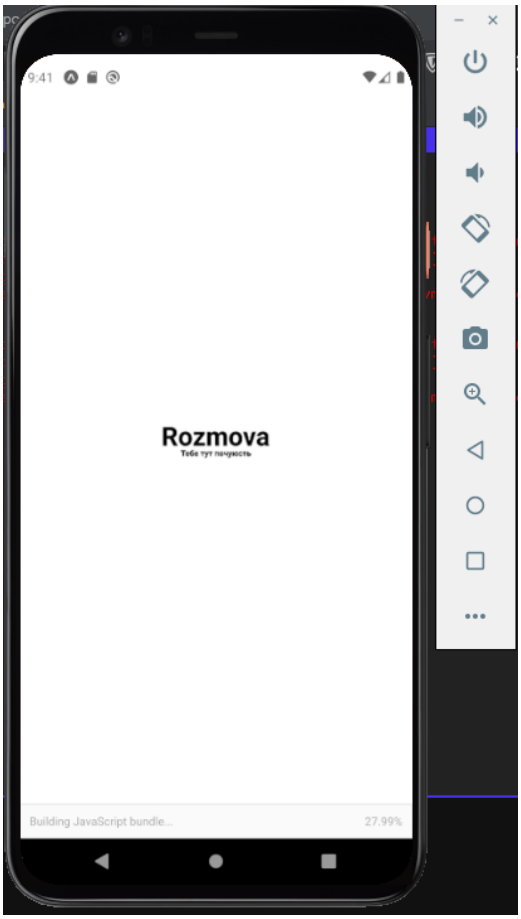


Рисунок 4.1.4 – Емулятор андроїд

4.1.3 Установка залежностей в проєкті

Тепер коли ми ініціалізували проєкт нам потрібно установити бібліотеки в залежності проєкту. Для цього ми будемо використовувати пакетний менеджер Yarn.

Yarn - це менеджер пакетів для коду. Він дозволяє використовувати та обмінюватися кодом з іншими розробниками з усього світу[17].

Для того, щоб установити пакет у проєкт нам потрібно в консолі прописати “yarn add package_name” де “package_name” - це назва пакету. Після установки пакету оновлюється файл package.json рис. 4.1.5. Після того як ми установили пакети які були вказані в 3 розділі ми можемо починати розробку проєкту

```
"dependencies": {
  "@react-native-community/masked-view": "^0.1.10",
  "@react-navigation/bottom-tabs": "^5.11.7",
  "@react-navigation/native": "^5.9.2",
  "@react-navigation/stack": "^5.14.2",
  "expo": "~40.0.0",
  "expo-font": "~8.4.0",
  "expo-image-picker": "~9.2.0",
  "expo-location": "~10.0.0",
  "expo-status-bar": "~1.0.3",
  "firebase": "^8.2.6",
  "formik": "^2.2.6",
  "key-mirror": "^1.0.1",
  "moment": "^2.29.1",
  "native-base": "^2.15.2",
  "react": "16.13.1",
  "react-dom": "16.13.1",
  "react-native": "https://github.com/expo/react-native/archive/sdk-40.0.1.tar.gz",
  "react-native-flash-message": "^0.1.22",
  "react-native-gesture-handler": "^1.9.0",
  "react-native-maps": "0.27.1",
  "react-native-reanimated": "^1.13.2",
  "react-native-safe-area-context": "^3.1.9",
  "react-native-screens": "^2.17.1",
  "react-native-web": "~0.13.12",
  "react-redux": "^7.2.2",
  "reanimated-bottom-sheet": "1.0.0-alpha.22",
  "redux": "^4.0.5",
  "redux-devtools-extension": "^2.13.8",
  "redux-thunk": "^2.3.0"
},
```

Рисунок 4.1.5 – Файл залежностей у проєкті

4.2 Діаграма використання

Для того, щоб було чітке розуміння для чого потрібен додаток, треба зробити діаграму варіантів використання рис. 4.2.1. В даній діаграмі буде відображений функціонал додатка який буде доступний кінцевому користувачу.

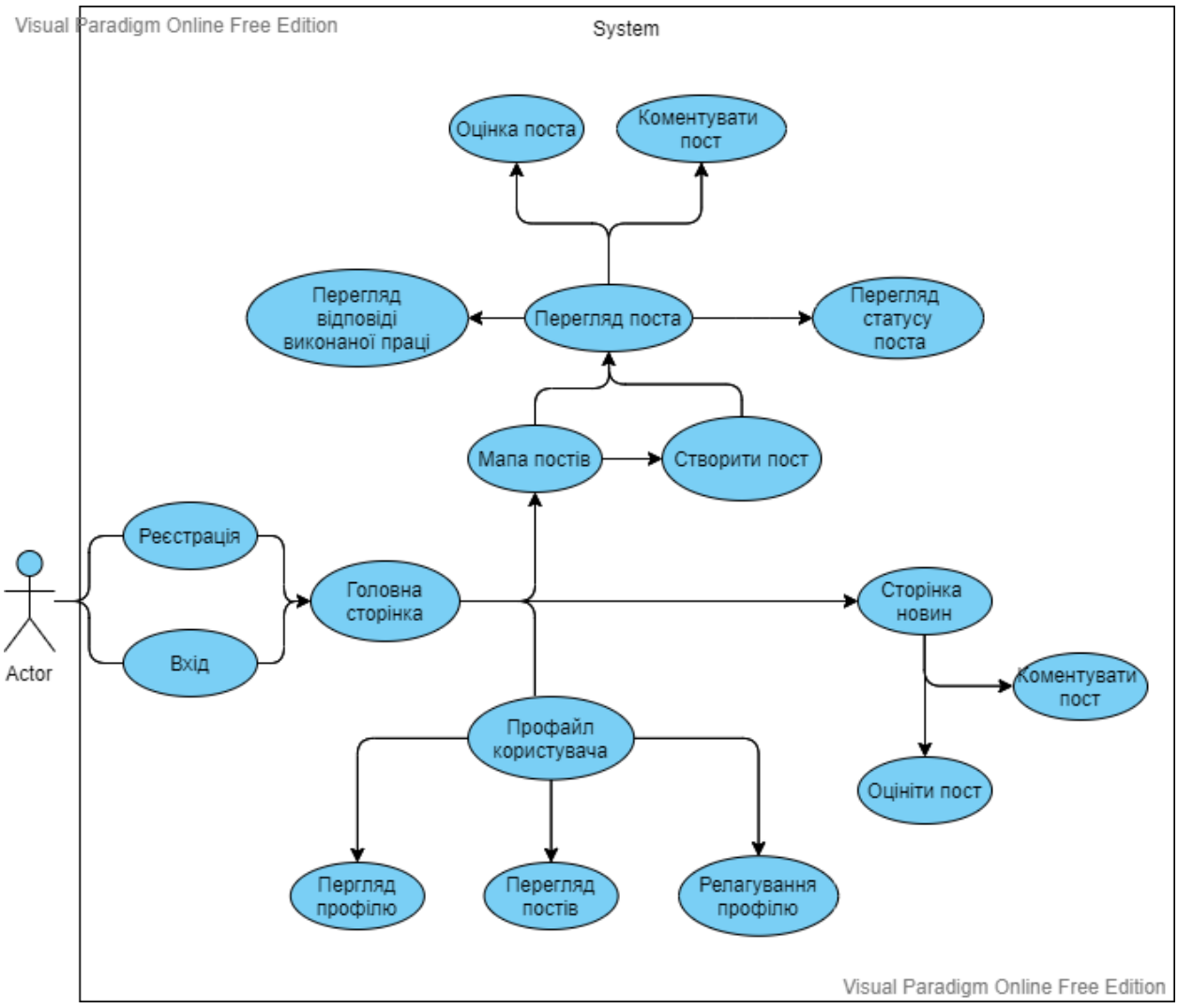


Рисунок 4.2.1 – Діаграма використання

4.3 Створення архітектури проєкту

Всі проєкти на React та React Native розробляються за допомогою компонентів. Архітектуру додатку за компонентами показано на діаграмі пакетів рис. 4.3.1.

Я розбив свої компоненти на контейнерні та презентаційні компоненти. Основна функція презентаційного компонента - відображення даних. Вони рідко обробляють стан і найкраще записуються як функціональні компоненти без

збереження стану. Компоненти контейнера - це компоненти, які визначають дані, які повинні відображати презентаційні компоненти.

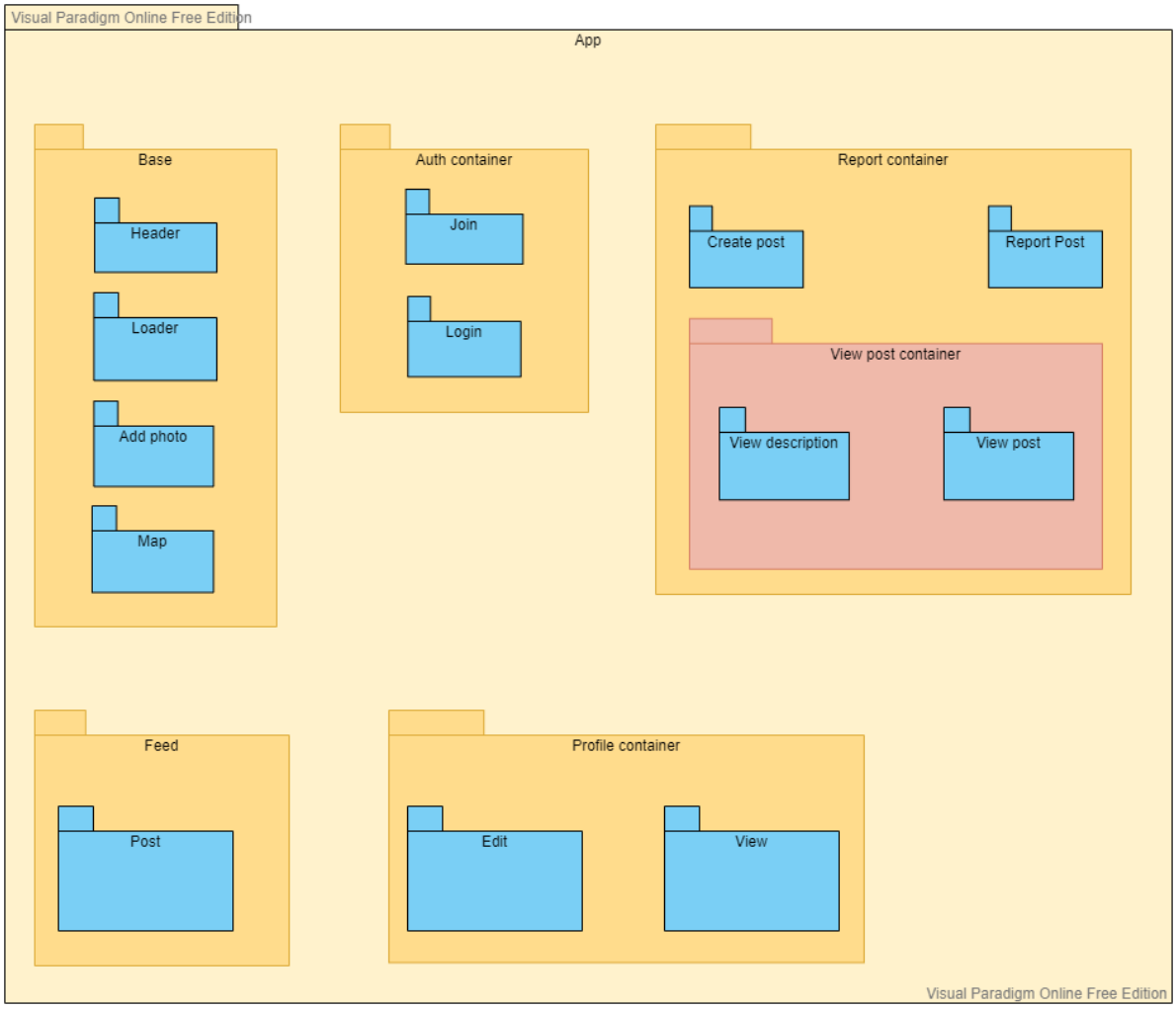


Рисунок 4.3.1 – Діаграма використання

Для навігації додатку було обрано React Navigation. Ця бібліотека дозволяє вашому додатку переходити між екранами та управляти історією навігації. Якщо ваш додаток використовує тільки один навігатор стека, то це концептуально схоже на те, як веб браузер обробляє стан навігації. Ключова відмінність між тим полягає в тому, що React Navigation надає жести і анімацію.

В проєкті використовується два типу навігації. Це переходи по кнопках, та на основі вкладок рис. 4.3.2.

Рисунок 4.3.2 – Панель навігації у додатку

Загальна структура навігації додатка виглядає як на діаграмі діяльності рис. 4.3.3.

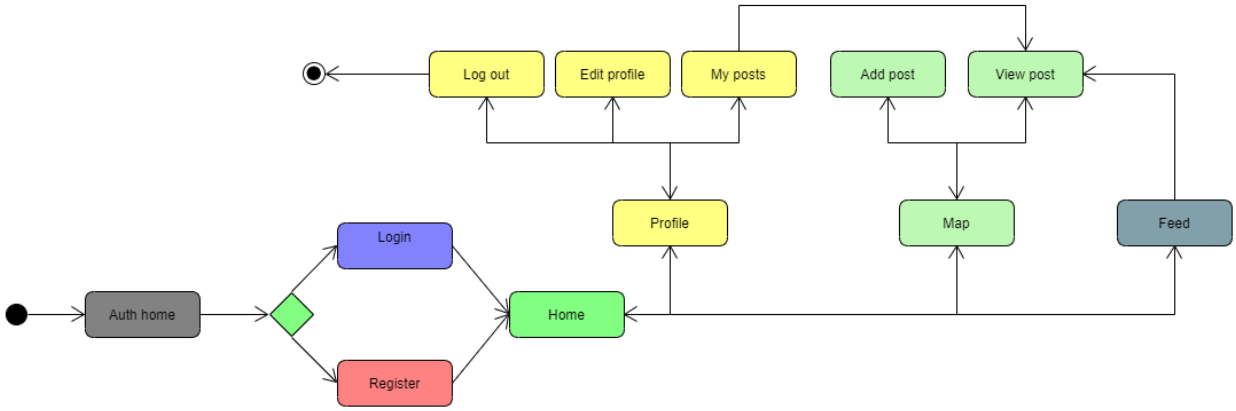


Рисунок 4.3.3 – Діаграма діяльності

В третій главі я обрав Flux архітектуру. Для реалізації в додатку підійде бібліотека Redux. Під'єднавши його до проєкту ми отримуємо єдине сховище. Завдяки цьому сховищем додатку легко управляти.

4.4 Використання Firebase

Firebase - це платформа для розробки мобільних та веб додатків при підтримці Google, яка допомагає розробникам створювати додатків[18]. Firebase керує власною інфраструктурою за допомогою інтернет-панелі інструментів.

Ці набори інструментів взаємо зв’язуються, масштабуються та інтегруються зі стороннім програмним забезпеченням для вирішення важких завдань за допомогою стандартних будівельних блоків.

4.4.1 Налаштування Firebase у проєкті

Для того, щоб використовувати FireBase нам потрібно створити в консолі Firebase проєкт. Коли ми створили проєкт в консолі нам потрібно додати його в наш додаток. Це можливо зробити за допомогою бібліотеки React-Native Firebase. Далі нам потрібно указати налаштування, який ми отримали при створюванні проєкту Firebase. Для цього я створив конфіг в якому зберігається налаштування для Firebase рис. 4.4.1.

```
export const firebaseConfig = {
    apiKey: "AIzaSyBHVqLQor9rzCvRJTquc3c_6C3h6oi-HuI",
    authDomain: "rozmova-1.firebaseapp.com",
    projectId: "rozmova-1",
    storageBucket: "rozmova-1.appspot.com",
    messagingSenderId: "343497335973",
    appId: "1:343497335973:web:81d54b35e4607c2094994c",
    measurementId: "G-ND5XP2JFGZ"
};
```

Рисунок 4.4.1 – Налаштування Firebase

Коли налаштування були задані, треба ініціалізувати Firebase рис 4.4.2. Ініціалізувати потрібно в головній компоненті. Для цього підійде компонента App.

```
if (!firebase.apps.length) {
    firebase.initializeApp(firebaseConfig);
}
```

Рисунок 4.4.2 – Ініціалізація Firebase

4.4.2 Створення FireStore в FireBase

Cloud Firestore - це гнучка база даних для розробки мобільних, веб додатків. Вона підтримує синхронізацію даних між клієнтськими додатками за допомогою слухачів в реальному часі і пропонує автономну підтримку для мобільних пристроїв, щоб створювати адаптивні додатки, які працюють незалежно від затримки в мережі або підключення до Інтернету[19].

Створюючи цю базу даних сервіс надає можливість обрати місцеположення серверу. Я обрав центральну Європу. Далі в налаштуваннях бази даних нам потрібно дозволити запис нових даних.

Загальна структура бази даних Firesore зображена на малюнку рис. 4.4.3

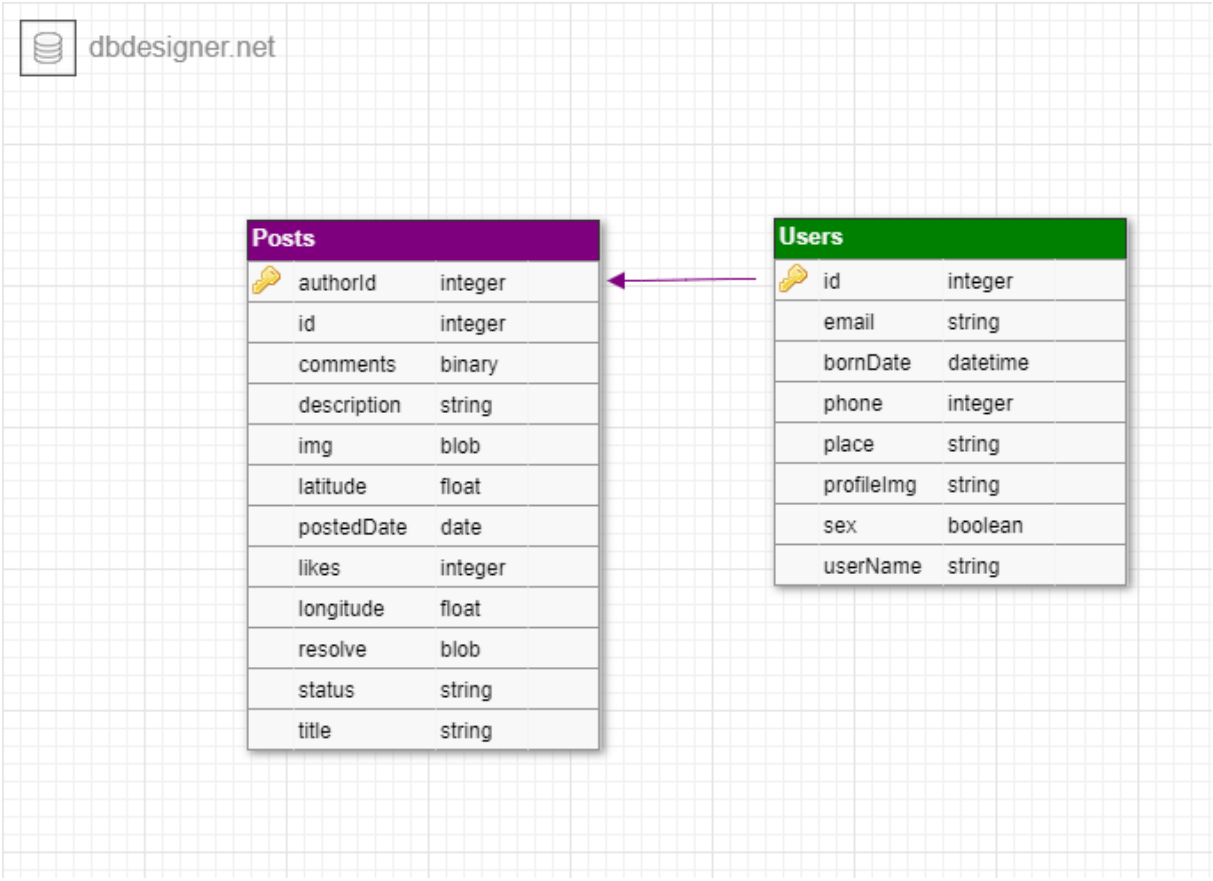


Рисунок 4.4.3 – Структура бази даних Firesore

4.4.3 Налаштування Firebase Authentication

Знання особистості користувача дозволяє додатку безпечно зберігати дані користувача в хмарі та забезпечувати однакову працю на всіх пристроях користувача.

Аутентифікація Firebase надає серверні сервіси, прості у використанні SDK та готові бібліотеки інтерфейсу для аутентифікації користувачів у додатку. Він підтримує автентифікацію за допомогою паролів, телефонних номерів, популярних постачальників об'єднаних ідентифікаційних даних, таких як Google, Facebook та Twitter тощо[20]. Ми будемо використовувати аутентифікацію за допомогою пошти та паролю.

Для реалізації аутентифікації у додатку нам потрібно перейти в налаштування аутентифікації та включити цей блок в консолі FireBase. На цій сторінці також можна подивитися всіх зареєстрованих користувачів та змінити листи які приходять на пошту.

4.5 Реалізація основних елементів додатка

Ознайомившись з backend частиною проєкту, перейдемо до опису основних функцій та компонентів додатку.

4.5.1 Авторізація

Додаток розділений на 2 частини. Частина авторизації та основна частина додатку. Для того, щоб відображати потрібні компоненти, в кореневому компоненті “MainMap” я роблю перевірку тернальним оператором рис. 4.5.1.

```
{isAuth ?
    <Tab.Navigator
        screenOptions={({route}) => ({
            tabBarIcon: ({focused, color, size}) => {
                if (route.name === 'Feed') {
                    return <Entypo name="home" size={20} color="black"/>
                } else if (route.name === 'Report') {
                    return <Entypo name="squared-plus" size={20} color="black"/>
                } else if (route.name === 'Search') {
                    return <FontAwesome name="search" size={20} color="black"/>
                } else if (route.name === 'Profile') {
                    return <FontAwesome name="user" size={20} color="black"/>
                }

            },
        })}
        tabBarOptions={{
            activeTintColor: 'black',
            inactiveTintColor: '#c9c9c9',

        }}
    >
        <Tab.Screen name="Feed" component={Feed}/>
        <Tab.Screen name="Report" component={Report}/>
        <Tab.Screen component={Search} name='Search'/>
        <Tab.Screen component={ProfileContainer} name='Profile'/>
    </Tab.Navigator>
    :
    <Stack.Navigator>
        <Stack.Screen
            component={Auth}
            name='Rozmova'
            options={{title: '', headerStyle: {elevation: null}, headerShown: false}}
        />
        <Stack.Screen component={Join} name='Join'/>
        <Stack.Screen component={Login} name='Login'/>
    </Stack.Navigator>
}
```

Рисунок 4.5.1 – Перевірка відображення компонентів

Перемінна isAuth за замовчуванням має значення “false”. Тобто перше що побачить користувач це сторінку авторизації.

Якщо ми детальніше розглянемо аутентифікацію, то побачимо таку структура файлів рис. 4.5.2

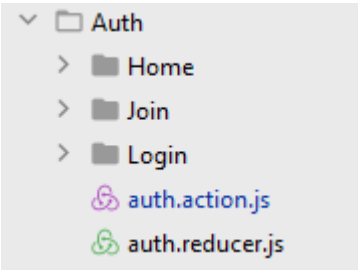


Рисунок 4.5.2 – Структура файлів авторізації

В компонентах Join та Login реалізована форма для авторизації користувача. Ця форма реалізована за допомогою бібліотеки Formik. Ця бібліотека самостійно зберігає дані які ввів користувач. Також вона дозволяє писати власну валідацію рис. 4.5.3. Для цього треба створити функцію стрілку і в параметри передати об'єкт помилок. Також для написання валідація використовуються регулярні вирази.

Регулярний вираз - це рядок тексту, яка дозволяє створювати шаблони, які допомагають зіставляти, знаходити та управляти текстом.

```
validate={values => {
    const errors = {};

    if (!values.nickName) {
        errors.email = 'Поле не може бути порожнім';
    }

    if (!values.email) {
        errors.email = 'Поле не може бути порожнім';
    } else if (
        !/^[A-Z0-9._%+-]+@[A-Z0-9.-]+\.[A-Z]{2,}$/i.test(values.email)
    ) {
        errors.email = 'Некоректний пароль';
    }

    if (!values.password) {
        errors.password = 'Поле не може бути порожнім';
    } else if (
        !passwordValid(values.password)
    ) {
        errors.password = 'Некоректний пароль';
    } else if( !passwordConfirm(values.password,values.c_password)){
        errors.password = 'Паролі повинні співпадати';
    }

    if (!values.c_password) {
        errors.c_password = 'Поле не може бути порожнім';
    } else if (
        !passwordValid(values.c_password)
    ) {
        errors.c_password = 'Некоректний пароль';
    }
```

Рисунок 4.5.3 – Валідація авторізації

Далі розглянемо файл auth.action.js. В цьому файлі реалізована вся асинхронна логіка компонентів авторизації рис. 4.5.3

```
const changeIsAuthAC = payload => ({type: types.SET_IS_AUTH, payload})
const changeAuthLoader = payload => ({type: types.SET_AUTH_LOADER, payload})
const setAuthProfileData = payload => ({type: types.SET_AUTH_PROFILE, payload})


export const handleLogin = (data) => async dispatch => {...}

export const singIn = (data) => async dispatch => {...}

export const logOut = () => dispatch => {...}
```

Рисунок 4.5.3 – Асинхронні функції авторизації

Три перші функції служать для того, щоб оновлювати сховище додатка. Далі йдуть асинхронні функції для зв'язку сервера з додатком. Розглянемо функцію handleLogin рис. 4.5.4, рис. 4.5.5. Всередині присутня конструкція try, catch, finaly. Вона потрібна для того, щоб якщо з'явиться помилка, можливо було б її опрацювати. В блоку try я роблю запит на сервер, і якщо він поверне помилку блок catch зможе її зловити та обробити її за умовами прописані.

```
try {
    dispatch(changeAuthLoader( payload: true))
    const createUser = await firebase.auth().createUserWithEmailAndPassword(data.email, data.password)
    await createUser.user.updateProfile(({displayName: data.nickName}))

    const newData = {
        userName: data.nickName,
        email: data.email,
        phone: '',
        profileImg: DEFAULT_IMG,
        id: createUser.user.uid,
        bornDate:'',
        sex:'',
        place:''

    }
    await firebase.firestore().collection( collectionPath: 'users').doc(createUser.user.uid).set({...newData})
    dispatch(changeIsAuthAC( payload: true))
```

Рисунок 4.5.4 – Функція handleLogin

```
} catch (e) {
    console.log(e.code)
    if (e.code === 'auth/email-already-in-use') {
        showMessage( options: {
            message: 'Данная почта уже занята',
            type: "danger",
        });
    } else {
        showMessage( options: {
            message: 'Чтото пошло не так!',
            type: "danger",
        });
    }
} finally {
    dispatch(changeAuthLoader( payload: false))
}
```

Рисунок 4.5.5 – Функція handleLogin

Файл auth.reducer відповідає за управляння сховищем. Сховище вже має початкове значення рис. 4.5.6. Також Сховище має конструкцію switch яке фільтрує запити на зміну. Після зміни сховища компоненти зможуть отримати нові дані, та оновити їх на сторінці.

```
const initial = {
    isAuth: false,
    isLoad: false,
    user:{}
}


const authReducer = (state {…} = initial, action) ⇒ {
    switch (action.type) {
        case types.SET_IS_AUTH:
            return {
                ...state,
                isAuth: action.payload
            }
        case types.SET_AUTH_LOADER:{
            return {
                ...state,
                isLoad: action.payload
            }
        }
        case types.SET_AUTH_PROFILE:{
            return {
                ...state,
                user:action.payload
            }
        }
        default:
            return state
    }
}

export default authReducer
```

Рисунок 4.5.6 – Сховище авторізації

В модулі авторизації повністю реалізована архітектура Flux, тобто в нас є “store” - це auth.reducer, “dispatcher” та “View” - це наші компоненти. Ця схема використовується у всьому проєкті.

4.5.2 Сторінка мапи

Головний задум проєкту це сторінка мапи. Для реалізації мапи у додатку, потрібні google map platform та бібліотека “react-native-maps”. Для того, щоб використовувати google map нам потрібен api key. Цей ключ ми можемо отримати

на сервісі Google Cloud API. Після того як ми отримали ключ, додаємо його в налаштування нашого додатку рис. 4.5.7.

```
"android": {
  "config": {
    "googleMaps": {
      "apiKey": "AIzaSyBvXI6FntCI3tlx71001Zeh4CDLEvz
    }
  },
```

Рисунок 4.5.7 – Налаштування мапи

Роботу сервісу мапи можливо побачити на діаграмі пакетів рис. 4.5.8

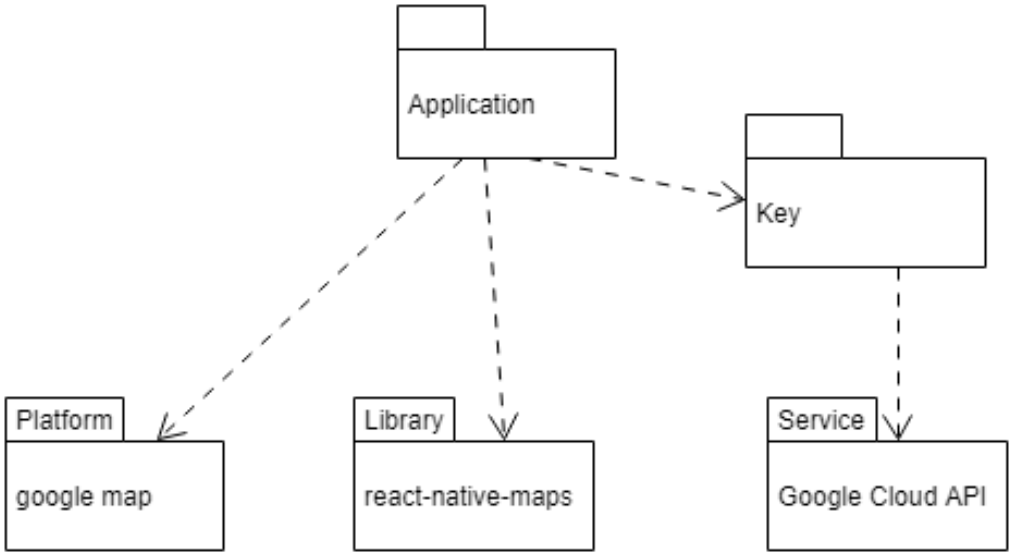


Рисунок 4.5.8 – Діаграма пакетів мапи

Створимо окрему компоненту с мапою. Перед тим як завантажитися сторінка нам потрібно ініціалізувати мапу рис. 4.5.9. Для цього в React Native є хук UseEffect. Він спрацьовує на створенні екземпляра компонента і його вставці в DOM.

useEffect - це хук, яка управляє побічними ефектами у функціональних компонентах[21].

DOM- об'єкт документа є кореневим вузлом документа HTML[22].

```
useEffect( effect: () => {
    (async () => {
        let {status} = await Location.requestPermissionsAsync();
        if (status !== 'granted') {
            setErrorMsg( value: 'Permission to access location was denied');
            return;
        }
        Location.getCurrentPositionAsync( options: {accuracy: Location.Accuracy.Highest})
            .then(location => {
                setLocation(location);
                {
                    cord ? handleChangePointCoordinate(location) : null
                }
            }).catch(e => setErrorMsg(e));
    })();
}, deps: []);
```

Рисунок 4.5.9 – Ініціалізація мапи

Після того як ініціалізували ключ. Створимо компоненту з мапою рис. 4.5.10 В параметри компоненти ми вказуємо стилі компоненту та елементи які відображаються на мапі.

```
<MapView style={styles.map}
         provider={PROVIDER_GOOGLE}
         showsUserLocation={true}
         followUserLocation={true}
         showsMyLocationButton={true}
         initialRegion={{
             latitude: location.coords.latitude,
             longitude: location.coords.longitude,
             latitudeDelta: 0.0036,
             longitudeDelta: 0.0121
         }}
>
```

Рисунок 4.5.10 – Компонент з мапою

4.5.3 Перевикористанні компоненті

Структура коду в React Native построєна таким чином що всі елементи розбиті на компоненти, тобто є можливість створювати нові елементи використовуючи вже написаний код. Зазвичай такі елементи додаються в папку Base рис. 4.5.12

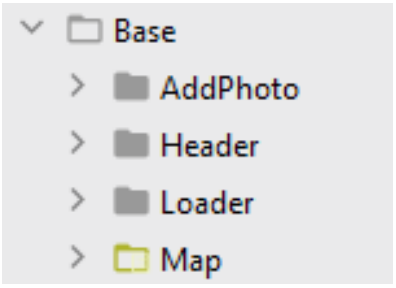


Рисунок 4.5.12 – Перевикористанні компоненти

4.6 Висновки

У даному розділі було розглянуто побудову мобільного додатку “ROZMOVA”. Був налаштований проєкт, встановлено все необхідні технології, створена база даних на основі Firebase. Також було розроблено фронтова частина додатка на React Native з фреймворком Expo.

ВИСНОВКИ

В результаті виконання даної роботи був розроблений мобільний додаток “Rozmova” на фреймворку React Native. Також були вершині такі задачі:

1) Проведено аналіз існуючих рішень розробки мобільних додатків, їх переваги та недоліки. На основі аналізу визначено що розробка на фреймворку React Native має багато переваг перед своїми конкурентами та є оптимальним рішенням для розробки додатку.

2) Визначено вимоги та функціональні можливості мобільного додатку “Rozmova”.

3) Був проведений аналіз програмних засобів для React Native. Був обраний оптимальний стек технологій, який забезпечує швидку розробку, та перевикористання коду.

4) Ініціалізація проєкту, установка та налаштування обраних бібліотек. Створення та налаштування бази даних на основі FireStore. Налаштування авторизації за допомогою FireBase. Розроблення діаграм діяльності, варіантів використання та пакетів. Проведено аналіз частин проєкту, та розроблена архітектура.

Перспективи проєкту. Надалі планується покращення працездатності розробленого додатка. Створення версії додатку для операційної систему IOS, та наповнення додатка новим функціоналом.

СПИСОК ВИКОРАСТИНОЇ ЛІТЕРАТУРИ

1) How to create mobile app [Електронний ресурс] – Режим доступу: https://buildfire.com/how-to-create-a-mobile-app/#section1 (дата звернення 29.04.2021). – Назва з екрана.

2) 5 Key of Native Mobile App Development [Електронний ресурс] – Режим доступу: https://clearbridgemobile.com/benefits-of-native-mobile-app-development/ (дата звернення 29.04.2021). – Назва з екрана.

3) Native, Cross-Platform, and Hybrid App Development: What to Choose [Електронний ресурс] – Режим доступу: https://forbytes.com/digital-transformation/native-cross-platform-and-hybrid-app-development-what-to-choose/ (дата звернення 29.04.2021). – Назва з екрана.

4) Core Components and Native Components [Електронний ресурс] – Режим доступу: https://reactnative.dev/docs/intro-react-native-components (дата звернення 29.04.2021). – Назва з екрана.

5) Vue documentation [Електронний ресурс] – Режим доступу: https://vue-native.io/docs/index.html (дата звернення 29.04.2021). – Назва з екрана.

6) Vue documentation [Електронний ресурс] – Режим доступу: https://vuejs.org/v2/guide/ (дата звернення 29.04.2021). – Назва з екрана.

7) Visual Studio code [Електронний ресурс] – Режим доступу: https://code.visualstudio.com/ (дата звернення 29.04.2021). – Назва з екрана.

8) Webstorm [Електронний ресурс] – Режим доступу: https://www.jetbrains.com/ru-ru/webstorm/ (дата звернення 29.04.2021). – Назва з екрана.

9) Expo documentation [Електронний ресурс] – Режим доступу: https://docs.expo.io/ (дата звернення 29.04.2021). – Назва з екрана.

10) Flux In-Depth Overview [Електронний ресурс] – Режим доступу: https://facebook.github.io/flux/docs/in-

depthoverview/#:~:text=Flux%20is%20the%20application%20architecture,a%20lot%20of%20new%20code. (дата звернення 29.04.2021). – Назва з екрана.

11) Model–view–controller [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Model%E2%80%93view%E2%80%93controller (дата звернення 29.04.2021). – Назва з екрана.

12) Getting Started with Redux [Електронний ресурс] – Режим доступу: https://redux.js.org/introduction/getting-started (дата звернення 29.04.2021). – Назва з екрана.

13) Redux Thunk [Електронний ресурс] – Режим доступу: https://github.com/reduxjs/redux-thunk (дата звернення 29.04.2021). – Назва з екрана.

14) Git [Електронний ресурс] – Режим доступу: https://en.wikipedia.org/wiki/Git (дата звернення 29.04.2021). – Назва з екрана.

15) What Is Figma? [Електронний ресурс] – Режим доступу: https://webdesign.tutsplus.com/articles/what-is-figma--cms-32272 (дата звернення 29.04.2021). – Назва з екрана.

16) Expo Cli [Електронний ресурс] – Режим доступу: https://docs.expo.io/workflow/expo-cli/ (дата звернення 29.04.2021). – Назва з екрана.

17) What is Yarn? [Електронний ресурс] – Режим доступу: https://yarnpkg.com/ (дата звернення 29.04.2021). – Назва з екрана.

18) The Good and the Bad of Firebase Backend Services [Електронний ресурс] – Режим доступу: https://www.altexsoft.com/blog/firebase-review-pros-cons-alternatives/ (дата звернення 29.04.2021). – Назва з екрана.

19) Cloud Firestore [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/firestore?authuser=0 (дата звернення 29.04.2021). – Назва з екрана.

20) Firebase Authentication [Електронний ресурс] – Режим доступу: https://firebase.google.com/docs/auth?authuser=0 (дата звернення 29.04.2021). – Назва з екрана.

21) Using the Effect Hook [Електронний ресурс] – Режим доступу: https://reactjs.org/docs/hooks-effect.html (дата звернення 29.04.2021). – Назва з екрана.

22) Введение Что такое Объектная Модель Документа (DOM)? [Електронний ресурс] – Режим доступу: https://developer.mozilla.org/ru/docs/Web/API/Document_Object_Model/Introduction (дата звернення 29.04.2021). – Назва з екрана.

ДОДАТОК А

Дипломна робота

“СТВОРЕННЯ МОБІЛЬНОГО ЗАСТОСУНКУ ДЛЯ КОМУНІКАЦІЇ МІСЦЕВОЇ ВЛАДИ З МЕШКАНЦЯМИ МІСТА НА ФРЕЙМВОРКУ REACT NATIVE”

Виконав студент:
Власенко І.Ю.
Керівник:
Гаманюк І.М.

Київ 2021

Основні характеристики роботи

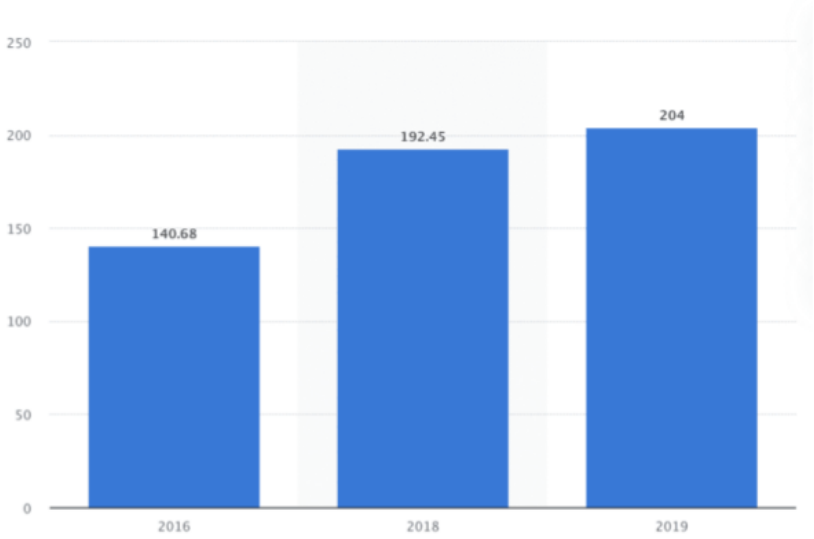
Об’єкт дослідження – процес комунікації місцевої влади.

Предмет дослідження – комунікація влади з мешканцями міста за допомогою застосунку.

Мета роботи – розробка мобільного додатку та поліпшення комунікації влади з мешканцями міста.

Методи дослідження – уніфікований процес розробки програмного забезпечення.

Мобільна розробка на сьогоднішній час



Види мобільних додатків

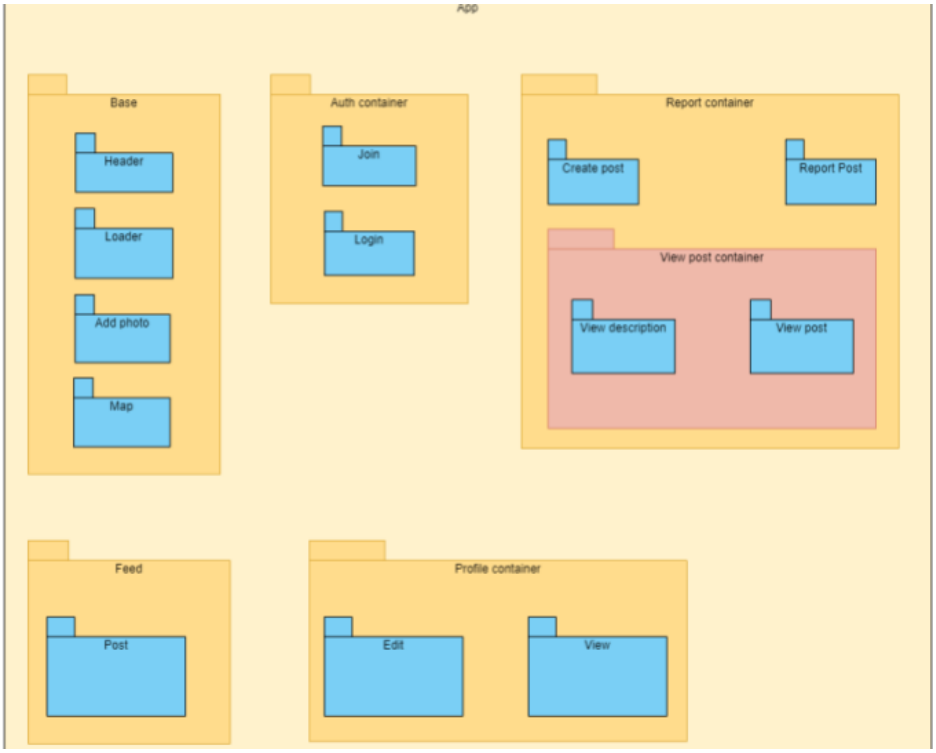
Вид мобільного додатки	Доступ до функціоналу пристрою	швидкість роботи	вартість розробки	поширення через магазин
Нативний	повний	дуже висока	висока	доступно
Гібридний	повний	дуже висока	середня	доступно
Веб додаток	частковий	висока	середня	не доступно

Графік зростання популярності React Native

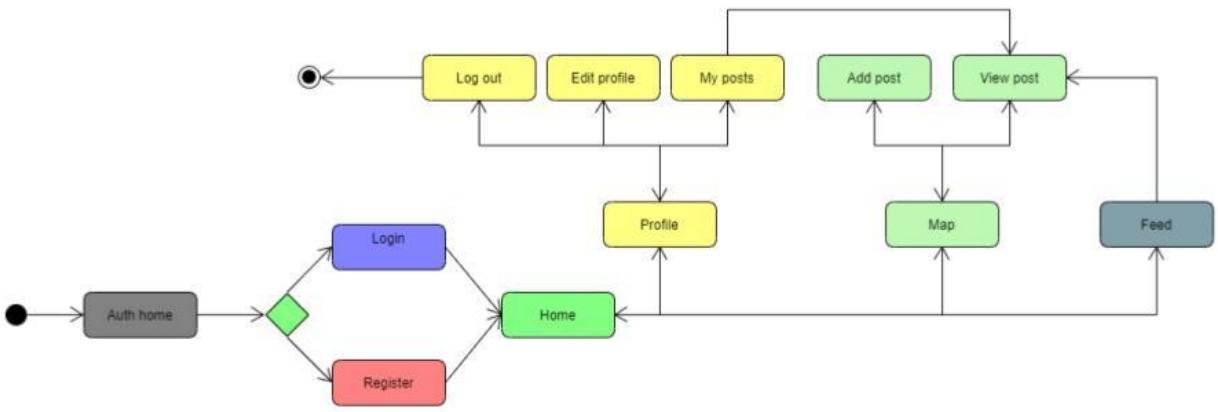
React Native init vs Expo

Найменування	**react-native init**	**Expo**
Ви можете додати власні модулі, написані на Java / Objective-C	+	-
Вага стандартного додатка Hello World	5-мб	25-мб
Потрібно Android Studio і XCode для запуску проєкт	+	-
Шрифти необхідно імпортувати вручну в XCode	+	-
Спільне використання додатка (за допомогою QR-коду або посилання)	Важче	Легше
Якщо ви хочете поділитися цим додатком, вам потрібно відправити весь файл .apk / .ipa	+	-
Надає JS API з коробки, наприклад Push-Notifications, Asset Manager	-	+

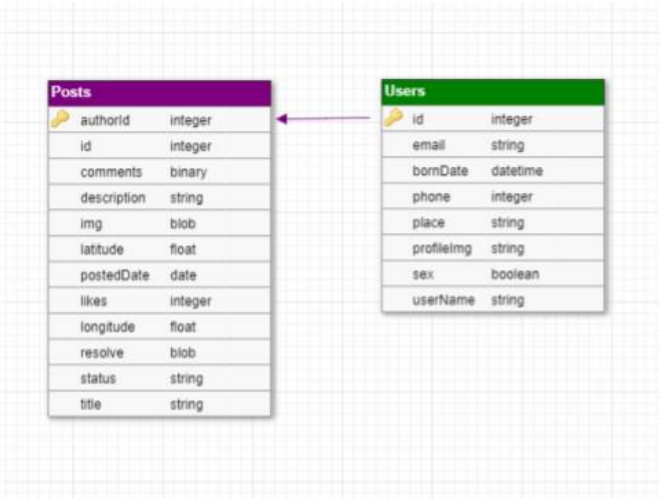
Архітектура проекту



Навігація проекту



Структура БД



Вхід та реєстрація

```
validate={values => {
    const errors = {};

    if (!values.nickName) {
        errors.email = 'Поле не може бути порожнім';
    }

    if (!values.email) {
        errors.email = 'Поле не може бути порожнім';
    } else if (
        !/^[A-Z0-9._%+-]+@[A-Z0-9-]+\.[A-Z]{2,}$/i.test(values.email)
    ) {
        errors.email = 'Некоректний пароль';
    }

    if (!values.password) {
        errors.password = 'Поле не може бути порожнім';
    } else if (
        !passwordValid(values.password)
    ) {
        errors.password = 'Некоректний пароль';
    } else if( !passwordConfirm(values.password,values.c_password)){
        errors.password = 'Паролі повинні співпадати';
    }
```

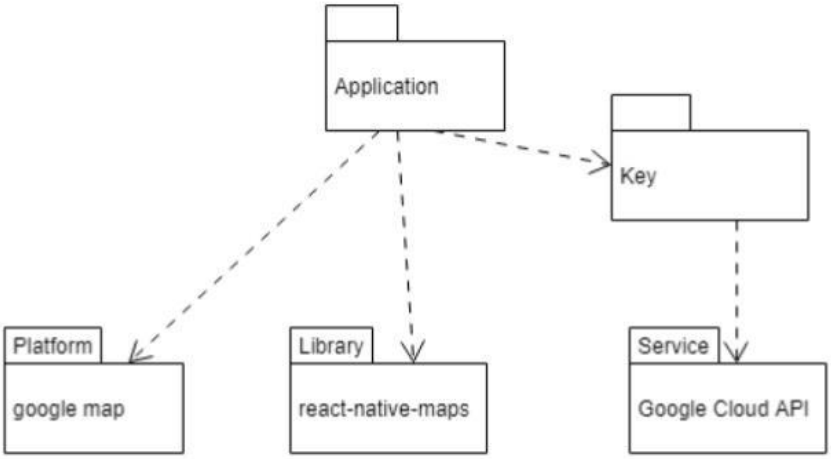
```
const changeIsAuthAC = payload => ({type: types.SET_IS_AUTH, payload})
const changeAuthLoader = payload => ({type: types.SET_AUTH_LOADER, payload})
const setAuthProfileData = payload => ({type: types.SET_AUTH_PROFILE, payload})


export const handleLogin = (data) => async dispatch => {...}

export const singIn = (data) => async dispatch => {...}

export const logOut = () => dispatch => {...}
```

Сервіс мапи



Висновки

В результаті виконання даної роботи був розроблений мобільний додаток “Rozmova” на фреймворку React Native. Також були вершині такі задачі:

1) Проведено аналіз існуючих рішень розробки мобільних додатків, їх переваги та недоліки. На основі аналізу визначено що розробка на фреймворку React Native має багато переваг перед своїми конкурентами та є оптимальним рішенням для розробки додатку.
2) Визначено вимоги та функціональні можливості мобільного додатку “Rozmova”.
3) Був проведений аналіз програмних засобів для React Native. Був обраний оптимальний стек технологій, який забезпечує швидку розробку, та перевикористання коду.
4) Ініціалізація проекту, установка та налаштування обраних бібліотек. Створення та налаштування бази даних на основі FireStore. Налаштування авторизації за допомогою FireBase. Розроблення діаграм діяльності, варіантів використання та пакетів. Проведено аналіз частин проекту, та розроблена архітектура.