МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

КАФЕДРА ЕЛЕКТРОНІКИ, ІНФОРМАЦІЙНИХ СИСТЕМ ТА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Кваліфікаційна робота

перший (бакалаврський)
(рівень вищої освіти)

на тему **Розробка мобільного кросплатформного додатку для забезпечення навчального середовища**

Виконав: студент 4 курсу, групи 6.1210-пзс
спеціальності 121 Інженерія програмного забезпечення
(код і назва спеціальності)
освітньої програми Програмне забезпечення систем
(код і назва освітньої програми)
Гнибіденко А.В.
(ініціали та прізвище)
Керівник доцент, к.ф.-м. н. В. І. Попівщий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)
Рецензент директор ТОВ «Дискус» Р.О. Лютий
(посада, вчене звання, науковий ступінь, підпис, ініціали та прізвище)

Запоріжжя
2024

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ІНЖЕНЕРНИЙ НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ім. Ю.М. Потебні
ЗАПОРІЗЬКОГО НАЦІОНАЛЬНОГО УНІВЕРСИТЕТУ

Кафедра електроніки, інформаційних систем та програмного забезпечення
Рівень вищої освіти _____перший (бакалаврський)_____
Спеціальність ___121_Інженерія програмного забезпечення_____
(код та назва)
Освітня програма _Програмне забезпечення систем
(код та назва)

ЗАТВЕРДЖУЮ
Завідувач кафедри_ Т.В. Критська
"_01_" __березня_____2024_року

З А В Д А Н Н Я
НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Гнибіденко Анні Вікторівні
(прізвище, ім'я, по батькові)

1.Тема роботи Розробка мобільного кросплатформного додатку для забезпечення навчального середовища
керівник роботи _____В.І. Попівщий, к.ф.-м.н., доцент.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом ЗНУ від _"26"_грудня_2023_року № 2215-с_____

2. Строк подання студентом кваліфікаційної роботи___07.06.2024____
3. Вихідні дані магістерської роботи
- комплект нормативних документів;
- технічне завдання до роботи.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
- огляд та збір літератури стосовно теми кваліфікаційної роботи;
- огляд та аналіз існуючих рішень та аналогів;
- дослідження проблеми створення кросплатформенних мобільних додатків;
- створення програмного продукту та його опис;
- перелік вимог для роботи програми;
- дослідження поставленої проблеми та розробка висновків та пропозицій.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)_____слайдів презентації

6. Консультанти розділів бакалаврської роботи

Розділ	Прізвище, ініціали та посада Консультанта	Підпис, дата Завдання прийняв

7. Дата видачі завдання_____01.03.2024____

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів магістерської роботи	Строк виконання етапів магістерської роботи	Примітка
1	Аналіз предметної області	15.03.24	виконано
2	Формулювання основної задачі дипломної роботи та узгодження її з науковим керівником	17.03.24	виконано
3	Аналіз існуючих методів рішення	20.03.24	виконано
4	Дослідження області кросплатформенних мобільних додатків	25.03.24	виконано
5	Узгодження подальших дій з науковим керівником	11.04.24	виконано
6	Аналіз теоретичних відомостей	15.04.24	виконано
7	Проектування інтерфейсу мобільних додатків	20.04.24	виконано
8	Узгодження інтерфейсу з науковим керівником	30.04.24	виконано
9	Реалізація функціоналу авторизації користувачів	14.05.24	виконано
10	Представлення отриманих результатів науковому керівнику і узгодження плану подальшого дослідження	17.05.24	виконано
11	Реалізація функціоналу хмарного сховища даних	20.05.24	виконано
12	Проведення аналізу можливостей розроблених програмних за стосунків	25.05.24	виконано
13	Оформлення звіту	27.05.24	виконано

Студент _____ _____Гнибіденко А.В._____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____Попівщий В.І._____
(підпис) (прізвище та ініціали)

Нормоконтроль пройдено

Нормоконтролер _____ _____І.А. Скрипник_____
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Сторінок: 66

Рисунків: 31

Джерел: 29

Гнибіденко А. В. Розробка мобільного кросплатформного додатку для забезпечення навчального середовища: кваліфікаційна робота бакалавра спеціальності 121 «Інженерія програмного забезпечення» / наук. керівник В.І. Попівщий. Запоріжжя : ЗНУ, 2024. 66 с.

Мета кваліфікаційної роботи полягає у створенні зручного мобільного кросплатформного застосунку для забезпечення навчального середовища. На основі аналізу наявних рішень було вирішено використати React Native через його переваги у розробці кросплатформних додатків, що дозволяє знизити витрати часу та ресурсів на розробку під різні операційні системи. Firebase було обрано як серверне рішення завдяки його потужним можливостям для аутентифікації користувачів, зберігання даних у реальному часі та можливості масштабування. Застосунки повинні бути з базовими функціональними можливостями.

У процесі розробки була розглянута проблема ефективної організації навчального процесу в умовах дистанційного навчання та необхідність створення інтерактивних навчальних матеріалів. У результаті був розроблений застосунок BEAM який має інтуїтивно зрозумілий інтерфейс для забезпечення легкого доступу до всіх функцій програми та систему управління навчальними матеріалами для зручного зберігання та організації лекційних матеріалів та завдань.

Ключові слова: *КРОСПЛАТФОРМЕННИЙ, JAVASCRIPT, TYPESCRIPT, JSX, REACT, REACT NATIVE, FIREBASE, FIREBASE AUTHENTICATION, FIREBASE CLOUD STORAGE, МОБІЛЬНИЙ ЗАСТОСУНОК, STUDY APP.*

SUMMARY

Pages: 66

Figures: 31

Sources: 29

Hnybidenko A. V. Development of a mobile cross-platform application for providing an educational environment: qualification work of a bachelor of specialty 121 "Software engineering" / science head V.I. Popivshchyi, Zaporizhzhia: ZNU, 2024. 66 p.

The goal of this qualification work is to create a convenient cross-platform mobile application to support the learning environment. Based on the analysis of existing solutions, it was decided to use React Native due to its advantages in developing cross-platform applications, which allows reducing time and resources required for development across different operating systems. Firebase was chosen as the server solution due to its robust capabilities for user authentication, real-time data storage, and scalability. The applications should have basic functional capabilities.

During the development process, the issue of effectively organizing the learning process in the context of remote learning and the need to create interactive learning materials was considered. As a result, the application BEAM was developed, which features an intuitive interface to ensure easy access to all application functions and a learning material management system for convenient storage and organization of lecture materials and assignments.

Keywords: *CROSSPLATFORM, JAVASCRIPT, TYPESCRIPT, JSX, REACT, REACT NATIVE, FIREBASE, FIREBASE AUTHENTICATION, FIREBASE CLOUD STORAGE, MOBILE APPLICATION, STUDY APP*.

ЗМІСТ

ВСТУП

Актуальність теми

З огляду на швидкий розвиток технологій, мобільний телефон став невід'ємною частиною нашого повсякденного життя і є основним джерелом, засобів необхідних для освіти, роботи, відпочинку та комунікації.

Враховуючи динаміку сучасного розвитку освітніх технологій та поширення мобільних пристроїв, розробка мобільного кросплатформного додатку для забезпечення навчального середовища набуває надзвичайної актуальності. Сучасне освітнє середовище вимагає інноваційних підходів, щоб забезпечити зручний та ефективний доступ до навчальних ресурсів для учнів та студентів. Це сприятиме підвищенню якості освіти, розширенню можливостей для інтерактивного навчання та забезпеченню студентів та викладачів необхідними інструментами для успішного навчання та викладання.

Кросплатформні додатки дозволяють користувачам отримувати якісний доступ до навчального контенту на різних пристроях, незалежно від їхньої операційної системи. Цей проект передбачає створення універсального додатку, який буде працювати на різних мобільних платформах, забезпечуючи студентам та викладачам зручний інструментарій для взаємодії та навчання.

Однією з проблем сучасного суспільства була та є відмінність операційних систем на різних пристроях. На даний момент існує достатньо стабільних засобів, які вже показали життєздатність на ринку, щоб створювати гідні застосунки.

У світлі цих викликів, присвячення кваліфікаційної роботи розробці мобільного кросплатформного додатку на створення зручного середовища для навчання студентів та учнів. Цей додаток дасть змогу поліпшити освітній процес, забезпечити доступ до необхідних ресурсів та сприяти активному навчанню і взаємодії.

Мета і завдання дослідження

Метою роботи є створення універсального додатку, який буде працювати на різних мобільних платформах, забезпечуючи студентам та викладачам зручний інструментарій для взаємодії та навчання на операційних системах Android та iOS.

Об’єкт дослідження

Об'єктом дослідження є кросплатформний застосунок для забезпечення навчального середовища.

Предмет дослідження

Предметом дослідження є розробка кросплатформного додатка для різних операційних систем та оцінка актуальності цієї теми.

Методи дослідження

Для розв’язання представлених завдань використовуються такі методи дослідження:

1. Аналіз джерел про технології створення кросплатформних застосунків.
2. Проведення аналогії з-поміж існуючих на ринку систем створення кросплатформних застосунків.
3. Синтез отриманих результатів досліджень.
4. Написання застосунку.
5. Апробація результатів.

Практичне значення одержаних результатів

Практичне значення отриманих результатів полягає у створенні мобільного кросплатформного додатка, який забезпечує навчальне середовище. Це сприятиме покращенню доступності та ефективності навчального процесу, надаючи студентам і викладачам зручні інструменти для

взаємодії та обміну інформацією незалежно від операційної системи, яку вони використовують.

Глосарій

Application Programming Interface (API) — це обчислювальний інтерфейс, який визначає взаємодію між кількома посередниками програмного забезпечення [4]. Він встановлює різновиди запитів, які можна здійснити, спосіб їх виконання, формати даних, які слід використовувати, угоди, яких слід дотримуватися і так далі.

Алгоритм — це набір інструкцій, які описують порядок дій виконавця з метою досягнення результату задачі за обмежену кількість кроків. Це система правил для виконання дискретного процесу, яка дозволяє досягти поставленої мети за скінченний час. Для візуалізації алгоритмів часто використовують блок-схеми. У випадку комп'ютерних програм, алгоритм є переліком детальних інструкцій, що реалізують процес обчислення. Цей процес починається з початкового стану і прогресує через послідовність логічних станів, завершуючись кінцевим станом..

Кросплатформенність (cross-platform) — це властивість програмного забезпечення працювати на більш ніж одній платформі, як програмній, так і апаратній; це також технології, що дозволяють досягти такої властивості. Кросплатформенність дозволяє значно зменшити витрати на розробку нового або адаптацію існуючого програмного забезпечення. Багато сучасних високорівневих мов програмування можна назвати кросплатформними.

Мобільний застосунок (mobile application) — це програмне забезпечення, яке розроблено для використання на смартфонах, планшетах та інших мобільних пристроях. Багато мобільних додатків встановлюються безпосередньо на пристрої або можуть бути завантажені з онлайн-магазинів додатків, таких як App Store, Google Play, Windows Phone Store і багатьох інших, як безкоштовно, так і за плату.

Back-end — програмно-адміністративна частина застосунку.

Front-end — це інтерфейс для взаємодії між користувачем і Back-end. Front-end та Back-end можуть бути розподілені між однією або кількома системами.

Клієнт-серверна архітектура — є одним із архітектурних шаблонів програмного забезпечення та є домінуючою концепцією у створенні розподілених мережних застосунків і передбачає взаємодію та обмін даними між ними [7].

Сервер або серверне програмне забезпечення в інформатиці (server) — це компонент програми обчислювальної системи, яка виконує функції обслуговування за запитом клієнта, надаючи йому доступ до певних ресурсів або послуг.

Callback — це передача виконуваного коду в якості одного з параметрів іншого коду. Callback дозволяє функції виконувати код, який задається як аргумент при виклику. Цей код може бути визначений в інших частинах програмного коду і недоступний для прямого виклику з цієї функції. Деякі алгоритмічні завдання мають не тільки числа або об'єкти як вхідні дані, але й дії (алгоритми), які природним чином задаються як зворотні виклики.

Cloud Storage — це модель зберігання даних у комп'ютері, де цифрові дані зберігаються у логічних пулах, а фізичне зберігання охоплює кілька серверів (зазвичай у кількох місцях). Фізичне середовище, як правило, належить хостинговим компаніям, які також керують цим середовищем. Постачальники хмарних систем зберігання даних відповідають за зберігання наявної інформації та доступ до неї, а також за роботу фізичного середовища. Користувачі можуть придбати у постачальників послуг хмарного сховища можливість зберігати свої дані там. *Нативний додаток* (native application) — це прикладна програма, яка була розроблена для використання на певній платформі або на певному пристрої [9].

Framework — інфраструктура програмних рішень, що полегшує розробку складних систем [3]. Інфраструктуру, можна розглядати як своєрідну

комплексну бібліотеку. Також використовується термін "каркас", а деякі автори вважають його основним.

Бібліотека в програмуванні (library) — збірка підпрограм або об’єктів, що використовуються для розробки програмного забезпечення.

Автентифікація (authentication) — це процедура встановлення належності користувача до інформації у системі на підставі його пред'явленого ідентифікатора. З точки зору інформаційної безпеки, аутентифікація є частиною процесу надання доступу до роботи в інформаційній системі. Вона виконується після ідентифікації користувача та передує авторизації [6].

Авторизація (authorization) — це процес керування рівнями та засобами доступу до захищених ресурсів. Вона може бути як фізичною (наприклад, доступ до кімнати готелю за допомогою картки), так і цифровою (наприклад, автоматизована система контролю доступу). Авторизація визначає, які ресурси системи користувач може використовувати на основі його ідентифікатора та пароля або надання певних повноважень особі чи програмі для виконання дій у системі обробки даних [6].

Ідентифікатор (identificator) — це назва об'єкта програми (змінної, масиву, структури, функції та ін.), що дозволяє звернутись до цього об'єкта; це властивість, яка повністю визначає сутність об'єкта в заданому просторі.

Клієнт в інформатиці (client) — апаратний або програмний компонент обчислювальної системи, який надсилає запити серверу.

Графічний інтерфейс користувача (GUI, Graphical user interface) — це тип інтерфейсу, що дозволяє користувачам взаємодіяти з електронними пристроями за допомогою графічних зображень та візуальних підказок, на відміну від текстових інтерфейсів, які базуються на використанні тексту, текстових команд та навігації [5].

Обліковий запис (account) у комп’ютерних системах — це сукупність наданої інформації про користувача, його ресурсів та прав відносно багатокористувацької системи. У такому обліковому записі зберігаються дані,

які ідентифікують користувача і визначають його доступ до різних функцій та можливостей системи. Інформація, що зберігається в обліковому записі, може включати ім'я користувача, пароль, електронну адресу, контактну інформацію та інші деталі.

Операційна система (operating system) — скорочено ОС — це основний комплекс програм, який керує апаратною складовою комп'ютера або віртуальної машини; забезпечує управління обчислювальним процесом і організовує взаємодію з користувачем. Операційна система зазвичай складається з ядра операційної системи та базового набору прикладних програм.

Шаблони проектування програмного забезпечення (*software design patterns*) — ефектні способи вирішення задач проектування програмного забезпечення [8]. Шаблон не є закінченим зразком, який можна безпосередньо транслювати в програмний код. Об'єктно-орієнтований шаблон найчастіше є зразком вирішення проблеми і відображає відношення між класами та об'єктами, без вказівки на те, як буде зрештою реалізоване це відношення [8].

Утиліта (*utility*) — сервісна програма, що допомагає керувати файлами, отримувати інформацію про комп'ютер, діагностувати й усувати проблеми, забезпечувати ефективну роботу системи. Утиліти розширюють можливості ОС [12].

РОЗДІЛ 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ ДЛЯ РЕАЛІЗАЦІЇ КРОСПЛАТФОРМНИХ НАВЧАЛЬНИХ СЕРЕДОВИЩ

1.1 Огляд літературних джерел

Дуже часто у сфері розробки програмного забезпечення ми стикаємося з проблемою кросплатформеності застосунків. Кожен застосунок повинен підтримувати всі операційні системи і, бажано, реалізовувати однаковий програмний (та стилістичний) інтерфейс. Незважаючи на те, що на даний момент існує безліч технологій, які дозволяють легко створювати кросплатформні додатки, багато програмних інженерів все ще використовують специфічні платформи, вважаючи, що вони втрачають в оптимізації або часом з новими технологіями.

Ця поширена помилка виникає не через будь-які недоліки технологій, а саме через консервативність самого програміста. Наразі існує достатня кількість кросплатформених технологій, які зайняли свою нішу на ринку і дозволяють не тільки скоротити час розробки програм, але й зберегти 100% продуктивність цільових мов.

Фреймворк (англ. Framework, каркас, платформа, структура, інфраструктура) — інфраструктура програмних рішень, що полегшує розробку складних систем. Спрощено дану інфраструктуру можна вважати своєрідною комплексною бібліотекою, але при цьому вона має ряд обмежень, що задають правила створення структури проекту та написання коду [10].

Кросплатформенність – властивість програмного забезпечення працювати більш ніж на одній програмній або апаратній платформі; технології; що дозволяють досягти такої властивості [11]. Вона дозволяє значно зменшити витрати на розробку нового або адаптацію існуючого програмного забезпечення. Кросплатформовими можна назвати більшість сучасних високорівневих мов програмування.

Приклад задач для вирішення яких використовується кросплатформенні технології:

1. Розробка кросплатформенних мобільних застосунків.
2. Розробка кросплатформенних десктопних застосунків.
3. Розробка кросплатформенних ігрових застосунків.
4. Розробка кросплатформенного програмного забезпечення з однаковими логікою та інтерфейсом у короткий проміжок часу зі збереженням рівня оптимізації.

Кросплатформовість є однією з найбільш важливих проблем у розробці програмного забезпечення. Завдання кросплатформовості має велике практичне значення. При створенні програмного забезпечення розробники часто мають за мету підтримку декількох операційних систем і використовують кілька спеціалізованих команд для розробки додатків для кожної операційної системи з мовою програмування, яка підтримується лише цією ОС. У таких випадках замовник витрачає більше коштів на підтримку розробки, яка стає складнішою через подальшу комунікацію між командами, дизайнерами та менеджерами. Часто такий підхід до розробки призводить до невдачі.

Коли програмний інженер обирає конкретну технологію, яка підтримує розробку для кількох операційних систем, на цьому етапі значно зменшується кількість людей, порівняно з використанням окремих команд для кожної ОС. Це дозволяє нам економити час та ресурси. Однак, кожна технологія має свої недоліки. Тому перед тим, як ставити завдання та використовувати неперевірену технологію, було б цілком розумно дослідити всі доступні кросплатформенні технології та їх процес створення додатків, щоб зменшити ймовірність невдач при розробці нового застосунку.

Наведемо деякі типові постановки задач створення кросплатформенних мобільних застосунків [18].

1. **Задача вибору технології**, яка полягає в тому, щоб обрати правильну кросплатформну технологію, яка повністю задовільнить потреби розробки застосунку.

2. **Розробка програмного забезпечення на мові технології**. Кожна кросплатформна технологія має основну мову програмування для подальшої інтерпретації її у нативні мови програмування.

3. **Аналіз застосунку на кожній операційній системі**, який полягає в тому, щоб макет замовника був повністю відтворений на усіх операційних системах однаково.

1.2 Аналіз програмного забезпечення

1.2.1 Аналіз React Native

React Native ідеально підходить для розробки освітніх додатків, основні переваги:

- Кросплатформність, яка забезпечує доступність додатка для більш широкої аудиторії, незалежно від операційної системи.
- Інтерактивність та швидкий розвиток, що дає можливість швидко вносити зміни і покращення в додаток у відповідь на зворотній зв'язок від користувачів.
- Інтеграція з іншими технологіями, легка інтеграція з різноманітними веб-сервісами та іншими технологіями, що використовуються в освіті (наприклад, платформи для онлайн-курсів, системи управління навчанням і т.д.).

1.2.2 Firebase

Firebase має наступні переваг для розробки освітніх додатків:

- Реальний час, що є важливим для створення інтерактивних навчальних середовищ, де зміни миттєво синхронізуються між студентами та викладачами.
- Автентифікація користувачів, проста інтеграція з популярними методами входу забезпечує безпеку та зручність для користувачів.

1.3 Результати аналізу програмного забезпечення

React Native є потужним інструментом для розробки мобільних додатків, особливо у випадках, коли необхідно швидко створити продукт, доступний на обох основних мобільних платформах. Він забезпечує баланс між продуктивністю, зручністю розробки та функціональністю, що робить його ідеальним вибором для створення навчальних середовищ.

Firebase є потужною платформою для розробки мобільних та веб-додатків, яка надає безліч корисних сервісів для створення інтерактивних та масштабованих освітніх середовищ. Його використання дозволяє значно скоротити час на розробку та забезпечує високу якість обслуговування користувачів.

РОЗДІЛ 2 ЗАСОБИ ТА ТЕХНОЛОГІЇ РОЗРОБКИ КРОСПЛАТФОРМНИХ ЗАСТОСУНКІВ

2.1 Аналіз сучасних технологій для розробки кросплатформенних мобільних застосунків

Основні кросплатформенні програмні технології, які вже мають достатню кількість застосунків на ринку – це **Xamarin, React Native** та **Flutter**.

2.1.1 Огляд Flutter

Flutter — це відкритий програмний каркас для розробки додатків для платформ Android та iOS, а також веб-додатків, який був створений компанією Google [21]. Перша версія Flutter, що носила назву "Sky", працювала лише на Android [24]. Вона була представлена у 2015 році на конференції розробників Dart з обіцянкою рендерингу 120 кадрів в секунду. 4 грудня 2018 року, під час заходу Flutter Live, було оголошено про випуск стабільної версії 1.0 [22].

Основні переваги [25, 26, 28]: можливість досягнення 120 кадрів в секунду на смартфонах з підтримкою цього функціоналу за допомогою компонентів Dart, розроблених програмістами мови Dart; підтримка декларативної архітектури, наприклад, вкладенні компоненти один в одного для отримання гнучкої структури; широкий набір інструментів для розробників, що дозволяє тестувати додаток та виявляти помилки без великих витрат часу; відкритий програмний каркас; підтримка операційної системи Google Fuchsia; унікальна мова програмування — Dart; проста структура; власний графічний двигун (не потрібно розробляти окремий інтерфейс для Android та iOS); низький поріг входу.

Основні недоліки [25, 27]: більший розмір кінцевого пакету, оскільки до нього додається віртуальна машина Dart; інтерфейс створюється за допомогою

коду, що призводить до меншої роздільності між логікою і дизайном; менше бібліотек (і інформації) порівняно з нативною розробкою; нестабільність.

2.1.2 Огляд React Native

React Native — це відкритий програмний фреймворк для створення додатків для різних платформ, таких як Android, iOS, Windows Phone, Linux, macOS або Windows [14]. У 2015 році компанія Facebook представила цей фреймворк, який використовує архітектуру React для розробки нативних додатків для IOS, Android та UWP. Основні принципи роботи React Native схожі на ReactJS, за винятком того, що він не працює з DOM через VirtualDom. Замість цього, він працює безпосередньо на пристрої у фоновому процесі, інтерпретуючи JavaScript код, написаний розробниками, спілкуючись з нативною платформою. Це означає, що Facebook виправив помилку, на яку Марк Цукерберг звертав увагу ще у 2012 році. React Native повністю заснований на JavaScript і не залежить від HTML. Вся логіка додатка написана на JavaScript та використовує нативні SDK [16].

Основні переваги [14-18]: досягнення 60 кадрів у секунду за допомогою розділення JavaScript та UI потоків; використовує одну з найпоширеніших мов програмування – JavaScript; має підтримку TypeScript; має багато бібліотек та внутрішніх кросплатформних компонентів; використовує React API; використовує JSX технологію замість HTML5; має нескладну структуру; середній вхідний поріг.

Основні недоліки [15, 19]: JavaScript UI потік не до кінця оптимізований при великій кількості даних.

2.1.3 Огляд Xamarin

Xamarin — американська компанія, що займається розробкою програмного забезпечення і підтримкою Mono та інструментів для розробки додатків мовою C# для iOS, Android, Windows та Mac. Вона була заснована в травні 2011 року Мігелем де Ікасою і Нетом Фрідманом (Nat Friedman) після

поділу активів компанії Novell [23]. У лютому 2016 року Microsoft придбала компанію. В рамках Microsoft робота Xamarin буде сфокусована на розвитку платформи для розробки мобільних додатків мовою C# з використанням технологій .NET.

Поєднання досягнень Xamarin з продуктами Visual Studio, Visual Studio Team Services і Azure створює рішення, яке охоплює всі необхідні аспекти для розробки, тестування та поширення мобільних додатків для будь-яких категорій пристроїв, включаючи пристрої на платформах Android і iOS [23]. Основні переваги [23, 29]: ідеальна сумісність з Інтернетом речей; оптимізованість для тестування; можливість створювати інтерфейси, що адаптуються для користувача, подібні до «рідних»; простота освоєння. Основні недоліки [23, 29]: застаріла платформа; затримки з оновленнями платформи; обмежений доступ до відкритих бібліотек; не підходить для додатків з високопродуктивною графікою; великий розмір додатків; складнощі з інтеграцією.

2.2 Аналіз вимог до апаратного та програмного забезпечення

Основним завданням кросплатформного клієнтського мобільного додатку є робота з колекціями курсів та завдань, з використанням хмарного сховища в мережі. Обробка даних відбувається на віддаленому сервері. Тому були визначені наступні вимоги до клієнтського додатку.

Мінімальні вимоги до апаратного забезпечення клієнту:

- Одноядерний процесор з базовою тактовою частотою 2.0 GHz.
- Оперативна пам’ять ємністю не менше 1 Гб.
- Інтернет з’єднання 5 мбіт.

Рекомендовані вимоги до апаратного забезпечення клієнту:

- Двох-ядерний процесор з базовою тактовою частотою 2.2 GHz.

- Оперативна пам’ять ємністю не менше 2 Гб.
- Інтернет з’єднання 20 мбіт.

Оскільки серверна частина додатоку контролюється безпосередньо хмарною платформою Firebase, вимоги до забезпечення сервера відсутні.

2.3 Результати аналізу програмного забезпечення

Згідно огляду існуючих технологій було вирішено обрати для дослідження найсучаснішу з них: React Native. React Native демонструє стабільність на ринку вже декілька років, постійно оновлюючись та розвиваючись як платформа.

РОЗДІЛ 3 ПРОЕКТ ПРОГРАМНОЇ ЧАСТИНИ КРОСПЛАТФОРМЕНОГО МОБІЛЬНОГО ЗАСТОСУНКУ

3.1 Архітектура системи

Клієнтська частина додатка повинна реалізовувати однаковий графічний інтерфейс для Android та iOS операційних систем.

Застосунок повинен реалізовувати зручну навігацію та інтерфейс для користувача, а також зв’язок з хмарним сховищем даним:

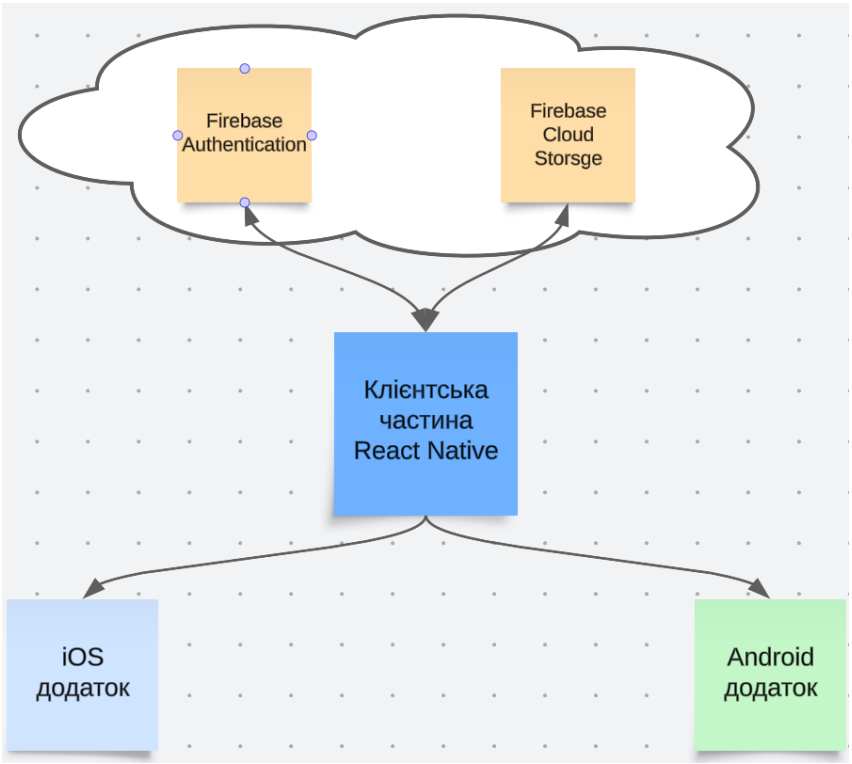

Рисунок 1 — Архітектура системи

3.1.1 Архітектура React Native застосунку

Структура React Native має два основних способа реалізації структури проекта:

1. З обгортанням у Expo-модуль, робота з нативними директоріями через спеціальну обгортку.

2. Без Expo-обгортки, робота напряму с нативними директоріями.

Expo-додаток

Обгортка Expo була створена у ранніх версіях фреймворка. Основною метою було спростити або повністю усунути роботу з нативною частиною. Структура React Native додатка в такому випадку виглядає наступним чином:

- одна директорія **.expo/**, в якій зібрана вся робота з нативною частиною, всі модулі інтеграції до фреймворка (робота з навігацією, аудіо, анімацією та іншим);
- одна директорія **source/**, всередині якої зібрані всі основні виконавчі файли JavaScript (або TypeScript);
- файли конфігурації різних доповнень (модулів та розширень) та гіт конфігурація до React Native додатка, які знаходяться у кореневому каталозі проекту;
- основний виконавчий файл, **index**, з якого починається виконання застосунку.

Для запуску Expo-додатка потрібно встановити емулятор мобільного пристрою потрібної платформи і почати запускати за допомогою команди **expo-cli**. Expo надає чітко визначений незмінний набір модулів, які можна використовувати як бібліотеки, для спрощення роботи та зменшення кількості коду. Місце розташування модулів: директорія **node_modules/**, яка з'являється та кешується після першої установки додатка. Будь-яка вихідна конфігурація є непередбачуваною, доступу до редагування та написання вихідних модулів немає.

На відміну від React Native додатку без Expo-обгортки, для запуску на реальному мобільному пристрої Expo запускає сервер, під'єднаний до поточного Wi-Fi роутера, використовуючи локальну IP адресу, та відображає QR-код. Після сканування цього коду програміст потрапляє до Expo-android або Expo-ios додатків і переходить до виконання коду. Важливою умовою є те,

що пристрій повинен бути підключений до тієї ж локальної мережі, що й джерело запуску. Якщо на пристрої відсутній Expo-додаток, користувач переходить до магазину додатків і шукає сторінку Expo-додатка для його подальшого встановлення.

Native-додаток

Структура React Native застосунку виглядає наступним чином:

- дві нативні директорії **android/** та **ios/**, для Android та iOS операційних систем відповідно, які відображають кінцевий стан інтерпретації головного коду (після оновлення основної директорії – нативні теж оновлюються головним сервером, якщо він запущений);
- одна директорія **source/**, всередині якої зібрані усі файли основні виконавчі файли JavaScript (або TypeScript);
- файли конфігурації різних доповнень (модулів та розширень) та гіт конфігурація до React Native застосунку, які знаходяться у корні проекту;
- основний виконавчий файл, **index**, з якого починається виконання програми.

Звичайний запуск на емуляторі мобільного пристрою відбувається за допомогою **react-native-cli** та пакет-менеджерів, наприклад NPM та Yarn, які запускають локальний сервер і здійснюють обробку головного JavaScript-джерела у нативні мови програмування. Дуже зручно оброблювати кожну з нативних директорій у відокремленій для цієї мети IDE: AndroidStudio та xCode. Там є можливість запустити пристрій-емулятор разом із сервером одним натиском миші. Для цього потрібно запустити проекти не з головної директорії, а з нативних, і натиснути кнопку з підписом/іконкою "Run".

На відміну від додатку з Expo-обгорткою, запуск виконавчого коду на реальному пристрої можливий лише при під'єднанні цього пристрою до комп'ютера, на якому встановлений проект. За допомогою спеціальних команд

потрібно запустити головний сервер та вказати підключений пристрій у параметрах команди.

Будь-яка нативна конфігурація підтримується, також є можливість написання додаткових модулів на нативних мовах програмування. На відміну від додатку з Expo-обгорткою, react-native не має модулів за замовчуванням і потребує детального встановлення та налаштування кожного з обраних для розробки NPM модулів.

3.1.2 Реалізація архітектури клієнтської частини

Для реалізації було обрано нативну конфігурацію для React Native яка забезпечує розробника більшою свободою та надає доступ до нативних директорій напряму.

Для розробки застосунку було обрано простий та нативно зрозумілий для користувача інтерфейс, з можливістю лише логіну користувачів та збереженням даним у хмарному сервісі. Користувач може потрапити з домашньої сторінки навігуючись за допомогою нижнього навігаційного бару , у свій профіль або на сторінку з доступними курсами.

3.2 Засоби реалізації

Під час розробки програмної системи, через низку переваг, було вирішено використовувати наступні засоби:

1. Мови програмування:
    a. Мова програмування JavaScript.
    b. Мова програмування TypeScript.
2. Середовища розробки:
    a. Visual Studio Code.
    b. Android Studio.
    c. Xcode.
3. Сервіси Firebase.

4. Сховище даних Redux.
5. Бібліотека Redux-Saga.
6. Менеджери пакетів Node Package Manager та Yarn.
7. Система контролю версій Git.

3.2.1 Мова програмування JavaScript

JavaScript — це динамічна, високорівнева, об'єктно-орієнтована прототипна мова програмування, що реалізує стандарт ECMAScript [15]. Вона використовується для створення скриптів на веб-сторінках, які надають можливість взаємодіяти з користувачем, керувати браузером та обмінюватися даними з сервером за допомогою функціоналу асинхронного програмування [6]. Мову програмування JavaScript класифікують як прототипну та скриптову мову програмування з динамічною типізацією [8].

JavaScript також частково підтримує інші парадигми програмування (імперативну та частково функціональну), а також властивості відповідної архітектури, такі як динамічна та слабка типізація, автоматичне керування пам'яттю, прототипне наслідування та функції як об'єкти першого класу.

На сьогодні JavaScript є однією з найпопулярніших мов програмування, але на початку багато програмістів ставилися до неї скептично [12]. З часом JavaScript вдосконалювалася, і для неї було розроблено багато бібліотек і фреймворків, а також поширилося використання JavaScript поза браузером. Синтаксис JavaScript схожий на мову C, але в порівнянні з C в ній є такі відмінності [11, 17, 20]:

- можливість динамічної зміни типу через механізм прототипів;
- функції як об'єкти першого класу;
- автоматичне приведення типів та автоматичне прибирання сміття;
- наявність анонімних функції.

JavaScript є основною мовою для розробки на платформі React Native.

3.2.2 Мова програмування TypeScript

TypeScript — мова програмування, представлена у 2012 компанією Microsoft; позиціонується як засіб розробки застосунків, що розширює можливості JavaScript [13]. Мова програмування TypeScript була створена Андерсом Гейлсбергом, розробником популярних мов програмування C#, Turbo Pascal і Delphi. TypeScript є зворотньо сумісним з JavaScript, та фактично TypeScript компілюється в стандартний JavaScript, тому він може працювати в будь-якому середовищі, де підтримується JavaScript.

Основний принцип TypeScript полягає в тому, що будь-який код на JavaScript сумісний з TypeScript. Це означає, що в програмах на TypeScript можна використовувати стандартні JavaScript-бібліотеки та раніше створені напрацювання. Більше того, існуючі JavaScript-проекти можна залишити в незмінному вигляді, а інформацію про типи додати у вигляді анотацій, розміщених в окремих файлах. Ці анотації не заважатимуть розробці та прямому використанню проекту, що є зручним підходом при розробці JavaScript-бібліотек.

Переваги над JavaScript

- можливість статичної типізації [13].
- можливість використання повноцінних класів [13].
- можливість підключення модулів [13].

3.2.3 Середовище розробки Visual Studio Code

Visual Studio Code — це потужне, безкоштовне та кросплатформне середовище розробки, створене компанією Microsoft. Воно користується популярністю серед програмістів завдяки своїм широким можливостям, високій продуктивності та активній спільноті користувачів.[16].

Редактор містить вбудовані інструменти для роботи з системою контролю версій Git, засоби рефакторингу та навігації, автодоповнення та контекстні підказки.

Підтримує розробку для таких мов і технологій: JavaScript, C++, C#, TypeScript, jade, PHP, Python, XML, Batch, F#, DockerFile, Coffee Script, Java, HandleBars, R, Objective-C, PowerShell, Luna, Visual Basic, Markdown, JSON, HTML, CSS, LESS і SASS, Haxe.

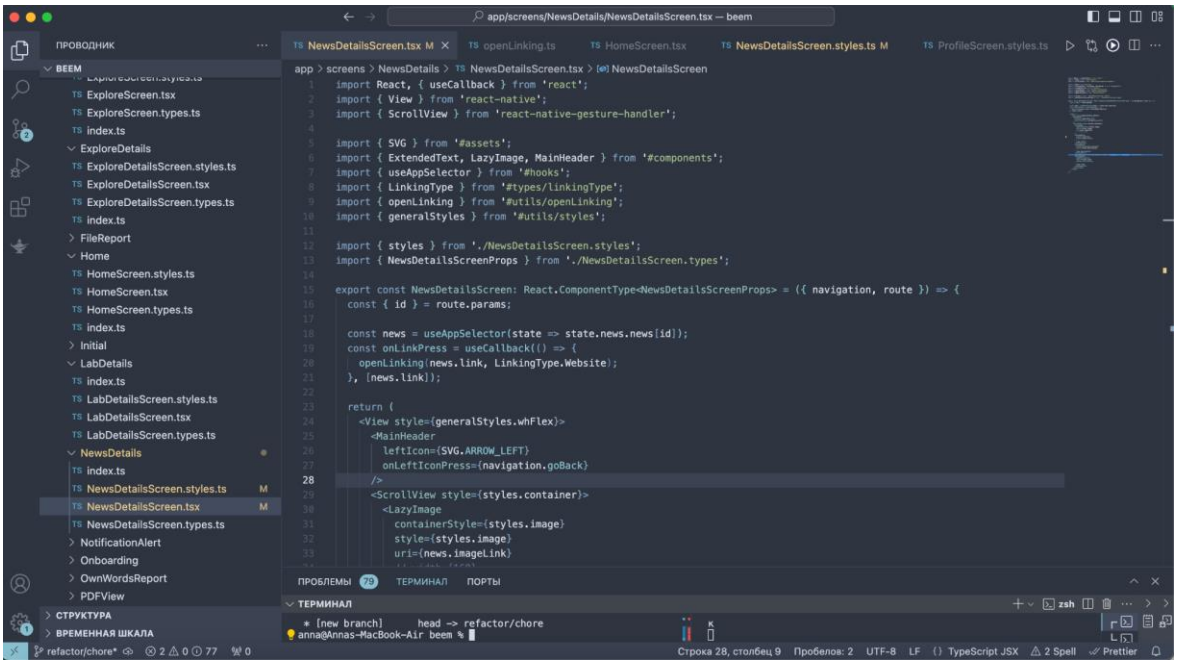

Рисунок *2 — Середовище розробки Visual Studio Code*

3.2.4 Середовище розробки Android Studio

Android Studio — це інтегроване середовище розробки (IDE) для платформи Android, представлене 16 травня 2013 року на конференції Google I/O [28]. Воно замінило плагін ADT для платформи Eclipse [29]. Це середовище розроблено на основі вихідного коду продукту IntelliJ IDEA Community Edition від компанії JetBrains.

Android Studio розвивається в рамках відкритої моделі розробки та поширюється під ліцензією Apache 2.0. Бінарні збірки доступні для Linux (тестування проводилося на Ubuntu), Mac OS X і Windows.

Це середовище надає інструменти для розробки додатків не лише для смартфонів і планшетів, а й для носимих пристроїв на базі Android Wear,

телевізорів (Android TV), окулярів Google Glass і автомобільних інформаційно-розважальних систем (Android Auto). Для додатків, спочатку розроблених з використанням Eclipse і плагіна ADT, існує інструмент для автоматичного імпорту існуючих проектів в Android Studio.

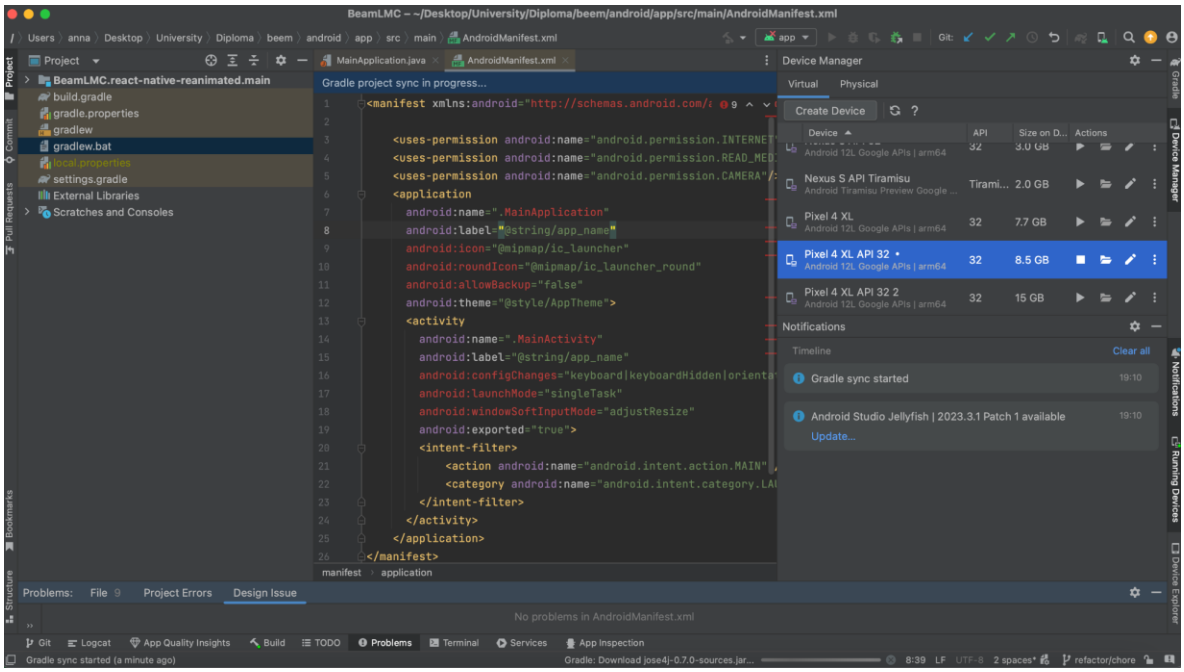

Рисунок *3 — Середовище розробки Android Studio*

3.2.5 Середовище розробки Xcode

Xcode — це інтегроване середовище розробки (ІРС) для macOS, яке включає набір інструментів для розробки програмного забезпечення, розроблених Apple для створення програмного забезпечення для macOS, iOS, iPadOS, watchOS та tvOS [17].

Це програмне забезпечення було випущено вперше у 2003 році; останній стабільний реліз — версія 12.1, яка була випущена 20 жовтня 2020 року, і доступна безкоштовно через Mac App Store для користувачів macOS Catalina. Зареєстровані розробники можуть завантажити попередні версії та бета-версії пакету через веб-сайт Apple Developer. Xcode також має інструменти командного рядка (Command Line Tools), які дозволяють розробляти UNIX-

стиль за допомогою програми Terminal на macOS. Їх також можна завантажити та встановити окремо від основного IPC [17].

Xcode підтримує вихідний код для мов програмування C, C++, Objective-C, Objective-C++, Java, AppleScript, Python, Ruby, ResEdit (Rez) та Swift, з різними моделями програмування, включаючи, але не обмежуючись, Cocoa, Carbon та Java. Треті сторони додали підтримку для GNU Pascal, Free Pascal, Ada, C#, Go, Perl та D [19]. Xcode може генерувати багатоархітектурні двійкові файли, які містять код для різних архітектур у форматі Mach-O.

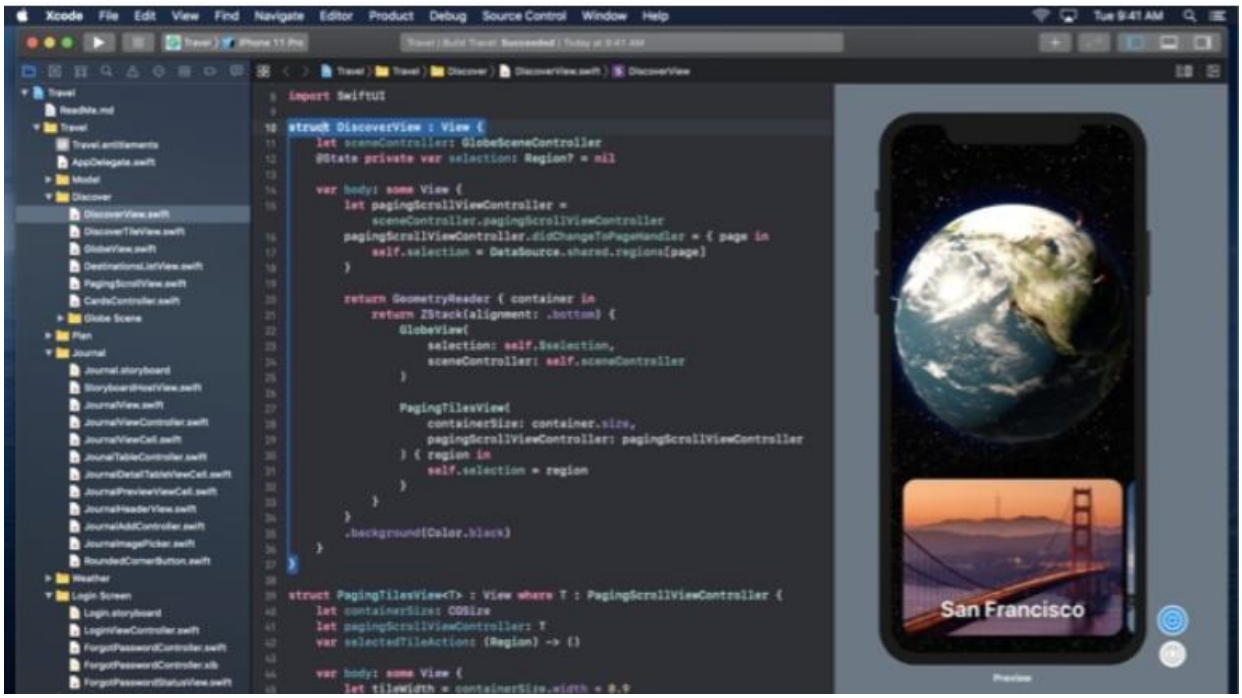

Рисунок *4 — Середовище розробки Xcode*

3.2.6 Хмарний сервіс Firebase

Firebase — це платформа, розроблена Google для створення мобільних та веб-додатків. Початково це було незалежна компанія, заснована у 2011 році. У 2014 році Google придбав платформу, і зараз вона є їх флагманською пропозицією для розробки додатків [15].

На платформі Firebase є 18 продуктів, які поділяються на три групи: "Develop", "Quality" та "Grow". Щоб почати використовувати продукти

Firebase, потрібно створити обліковий запис Firebase та додати ключову інформацію про додаток. Потім необхідно завантажити файли, які містять інформацію про з'єднання з платформою Firebase, налаштувати їх у Front-end додатку та використовувати надану документацію по бібліотеці, яка допоможе використовувати необхідні серверні функції для клієнтської сторони.

Firebase Authentication

Один з продуктів Firebase платформи — це сервіс серверної авторизації для клієнтських додатків, які працюють на різних платформах, включаючи мобільні, десктопні та веб-додатки. Firebase Authentication дозволяє клієнтській стороні кросплатформеного мобільного додатка не мати потреби використовувати серверний додаток для обробки токенів доступу або збереження інформації про користувачів.

Для налаштування Firebase Authentication необхідно додати розділ firebase/auth до бібліотеки Firebase у клієнтському додатку та налаштувати платформу для конфігурації на різних оперативних системах, таких як Android та iOS.

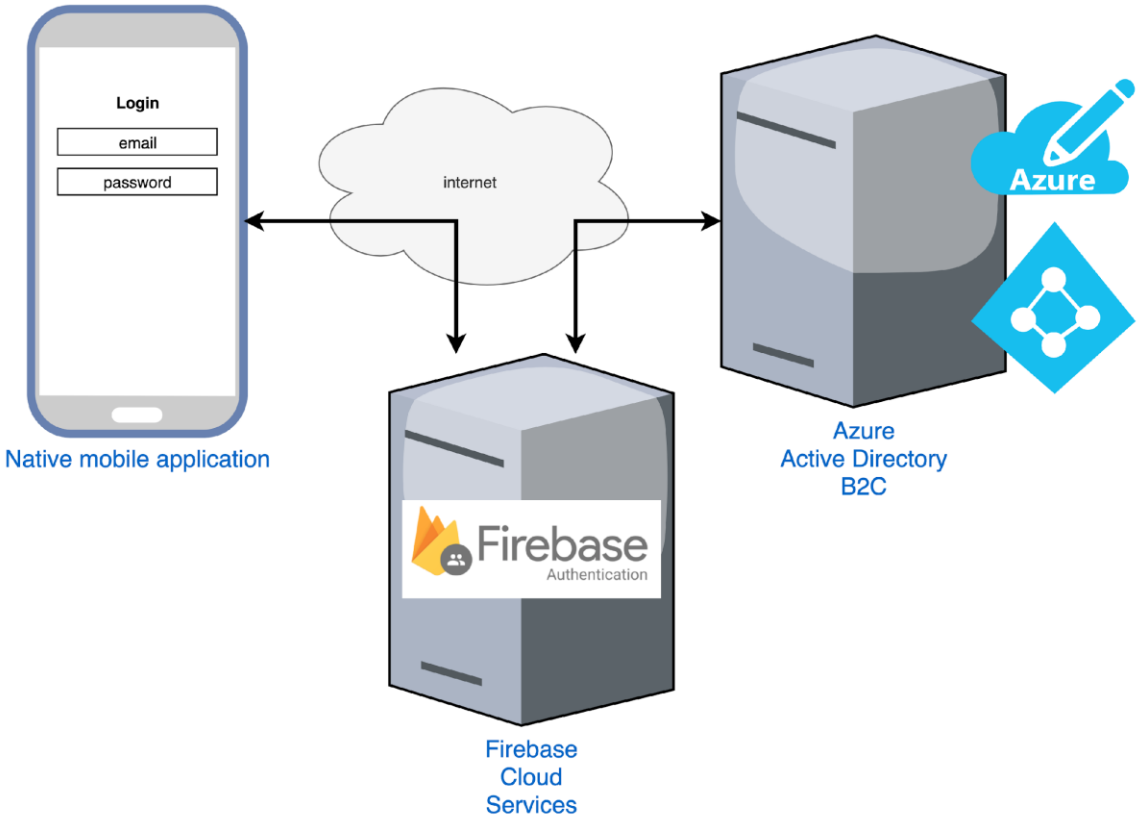

Рисунок 5 — *Схема роботи Firebase Authentication*

Firebase Cloud Storage

Інший продукт Firebase — це сервіс серверного сховища даних та доступу до них. За допомогою Firebase Cloud Storage клієнтська сторона кросплатформеного мобільного додатка може отримати доступ до збереження та роботи з даними без необхідності використовувати HTTP-запити до серверного додатку.

Для налаштування Firebase Cloud Storage також необхідно додати розділ firebase/storage до бібліотеки Firebase у клієнтському додатку та налаштувати платформу для конфігурації на різних оперативних системах, таких як Android та iOS.

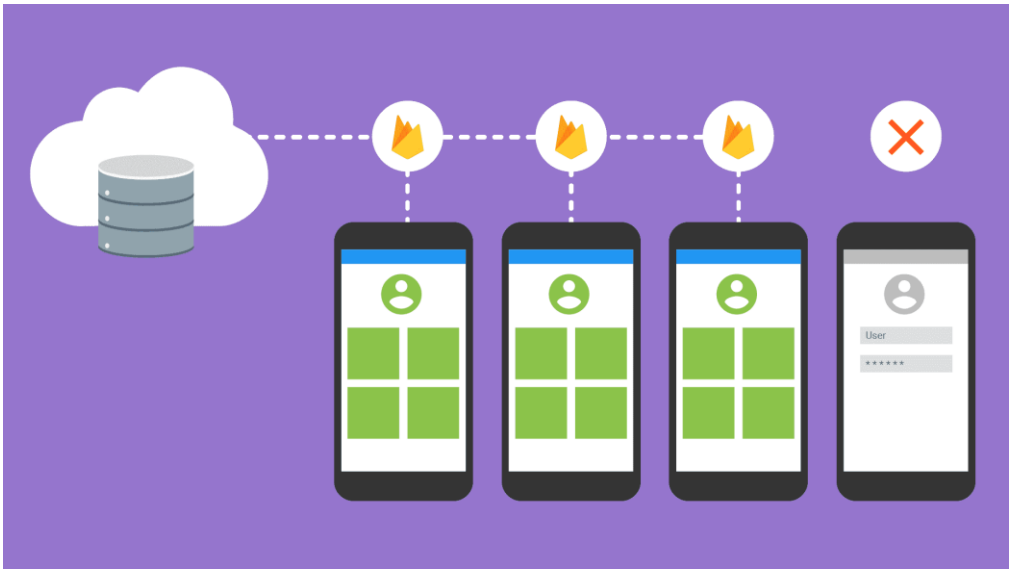

Рисунок 6 — *Зберігання даних до Firebase Cloud Storage*

3.2.7 Локальне сховище даних Redux

Redux — це відкрита бібліотека JavaScript для керування станом програми [14, 20, 26]. Його часто використовують разом з бібліотеками, такими як React або Angular, для створення користувацьких інтерфейсів. Redux був створений Даном Абрамовим і Ендрю Кларком на основі архітектури Facebook Flux.

Бібліотека має простий та обмежений API, який призначений для передбачуваного збереження стану програми. Вона працює на основі концепції функціонального програмування, схожої на функцію зменшення.

Архітектура Redux

Архітектура Redux працює наступним чином (див. рис. 7):

- Представлення може викликати будь-яку зазначену дію (Actions).
- Дії (Actions) генерують об'єкт з типом та даними, які передаються з представлення (View), а потім вони викликають диспетчера (Dispatcher).
- Редуктори (Reducers) обробляють отримані дії (Actions), зазвичай за допомогою конструкції switch-case. Редуктори реалізовані як чисті

функції, що означає, що при однакових вхідних даних результат виконання коду буде завжди однаковим.

- Редуктори зберігають результат у сховище (Store). При цьому сховище є повністю не змінним (Immutable), тому зміни здійснюються шляхом створення нових об'єктів даних, а не модифікації існуючих.

Це дозволяє забезпечити передбачуваність та незмінність стану програми, що сприяє полегшенню розробки та підтримки програмного продукту.

Redux також надає можливості для моніторингу та налагодження програми, таких як розширення Chrome Redux DevTools, яке дозволяє відстежувати стан програми та переглядати деталі змін.

У загальному, Redux є потужною бібліотекою для керування станом програми, яка допомагає забезпечити чистоту та передбачуваність коду, спрощує розробку та підтримку проектів і покращує продуктивність розробників.

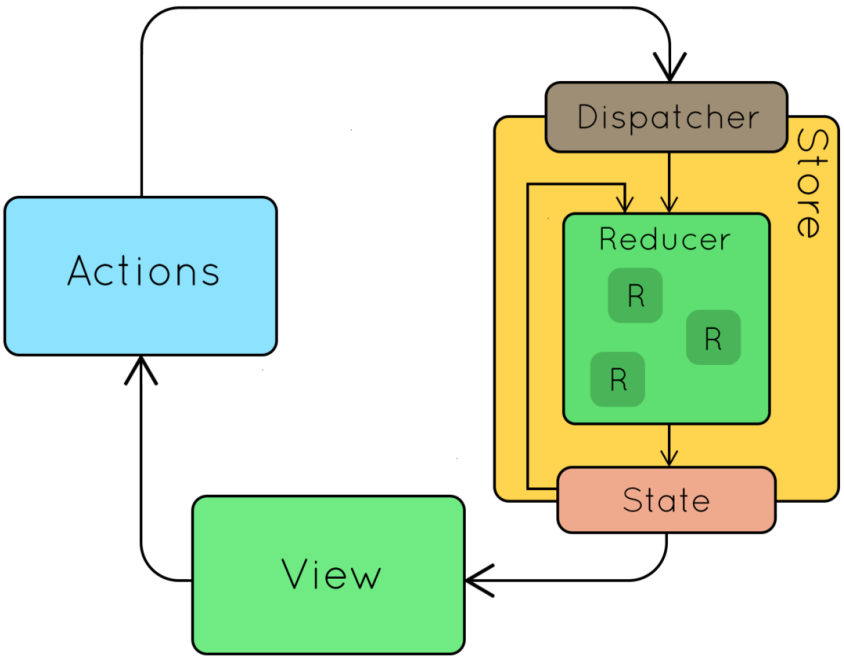

Рисунок *7 — Структура збереження даних Redux*

3.2.8 Бібліотека Redux-Saga

Redux-saga — це бібліотека, яка має на меті спростити управління побічними ефектами додатків, такими як асинхронні операції отримання даних та доступ до кешу браузера. Вона також покращує продуктивність, забезпечує легкість тестування та ефективну обробку помилок [14, 16].

Основна ідея полягає в тому, що сага представляє собою окремий потік в додатку, який відповідає за управління побічними ефектами. Redux-saga є middleware для Redux, що означає, що його можна запускати, призупиняти та скасовувати з основного програмного коду за допомогою звичайних Redux дій. Він також має доступ до повного стану Redux програми і може відправляти Redux дії [16].

Архітектура Redux-Saga

Саги працюють всередині контейнера Redux як middleware. Вони запускаються між діями (Actions) та Reducers, затримуючи поточну дію до тих пір, поки не буде виконана серверна робота або інші асинхронні операції. Після отримання даних з сервера або виконання інших асинхронних операцій, саги додають інформацію до об'єкта дії та передають його до Reducers (див. рис. 8).

Конструкція генераторів

Саги потребують спеціального написання функцій виконавця, які будуть взаємодіяти з серверними API. Зазвичай для цього у мовах програмування існують генератори функцій, які використовують ключове слово **yield** для призупинення основного потоку виконання, поки не отримають відповідь від сервера або не закінчиться встановлений час (timeout response) для виконання.

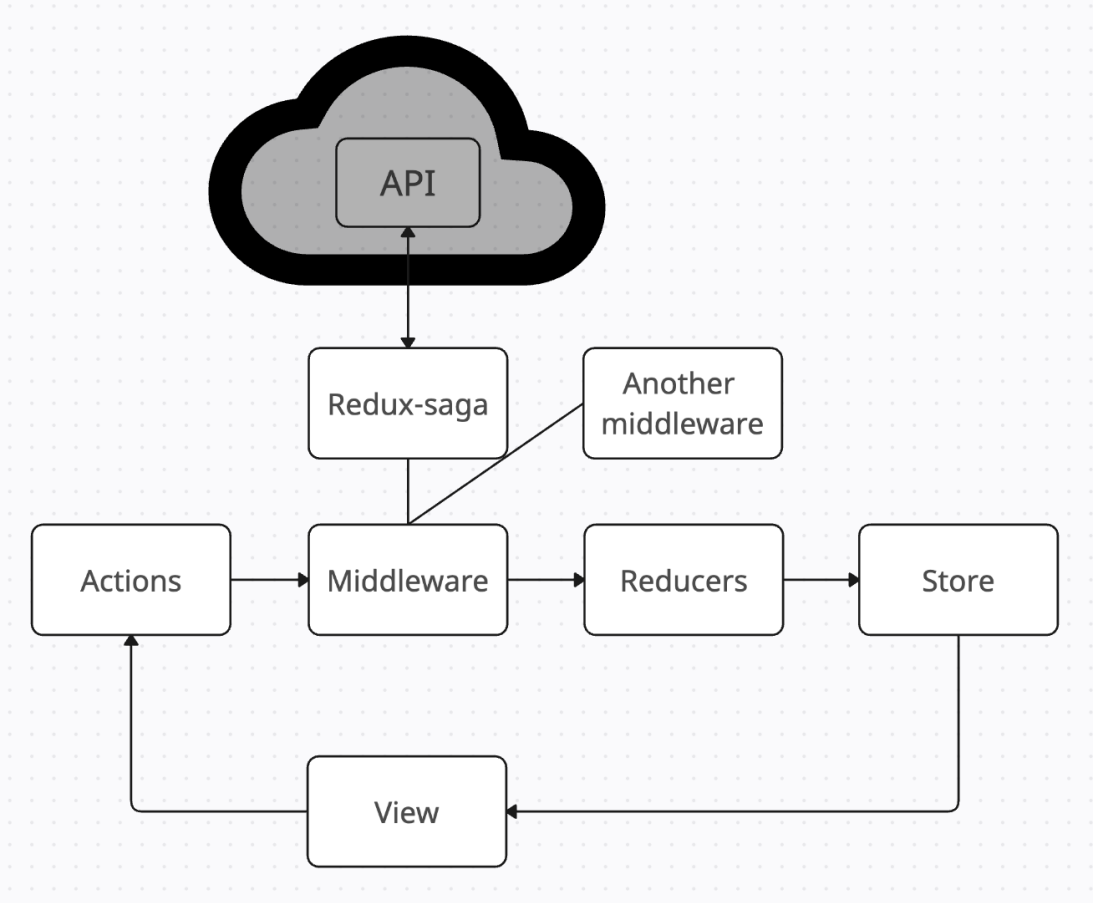

Рисунок *8 — Конструкція Redux-Saga у звичайному React додатку*

3.2.9 Менеджери пакетів Node Package Manager та Yarn

Обов'язковою умовою деяких платформ, які створюють кросплатформенні мобільні додатки, є тісне співпрацювання з менеджерами пакетів. Зазвичай ці менеджери відповідають за поширення окремих модулів від інших авторів, які зазначені у спеціальному файлі (наприклад, package.json) та використовуються для збереження місця на комп'ютері та в хмарному сховищі (GitHub, BitBucket, GitLab та інші). Це дуже корисно з точки зору розширюваності та модульності додатків.

Додатково можуть бути вказані інформація про версії модулів, їх назви, адреси та інші характеристики. Після встановлення модулів створюється інший файл (зазвичай називається package-lock.json), в якому вказані локально встановлені версії. Ця інформація може бути корисною і також може бути відправлена до сховища Git.

Node Package Manager

Самий відомий менеджер пакетів Node Package Manager (NPM) — це менеджер пакетів для мови програмування JavaScript, який є стандартним для середовища виконання Node.js. Він був випущений у 2010 році та став невід'ємною частиною екосистеми JavaScript, забезпечуючи розробникам простий спосіб встановлення, оновлення та керування бібліотеками та інструментами.

Після генерації package-json.lock файлу не оновлює його, навіть після повторного опрацювання модулів з package.json.

Yarn

Yarn — це менеджер пакетів для JavaScript, розроблений Facebook у співпраці з іншими компаніями, як альтернатива Node Package Manager (NPM). Випущений у 2016 році, Yarn надає швидкий, надійний і безпечний спосіб управління залежностями в проектах на JavaScript:

- більш швидке встановлення та редагування модулів завдяки паралельним процесам.
- генерує yarn-lock.json із іншою структурою.
- оновлення yarn-lock.json після кожної обробки модулів.
- більш структуровані консольні команди.

3.2.10 Система контролю версій Git

Git — це розподілена система контролю версій, яка служить для відстеження змін у будь-якому наборі файлів. Спочатку її створено з метою координації роботи між програмістами, які спільно працюють над вихідним кодом під час розробки програмного забезпечення [18]. Основними цілями Git є швидкість, цілісність даних та підтримка розподілених нелінійних робочих процесів.

За допомогою системи контролю версій Git можна організувати не тільки командну роботу, але й зручно працювати над проектами самостійно. Система гілок надає зручний інтерфейс для розподілення роботи та керування часом програміста (див. рис. 9). Також варто зауважити, що більше не потрібно зберігати безліч архівних копій на своєму комп'ютері, як це робилося раніше. За допомогою системи контролю версій можна зручно переміщуватися від однієї версії проекту до іншої.

Система контролю версій надає зручний спосіб отримання будь-якої збереженої версії за короткий проміжок часу. Проте, певні файли не повинні бути включені до Git. Це можуть бути файли, які створюються під час компіляції, кешування або модулів проекту. Тому використовується файл .gitignore, в якому вказуються типи файлів, які необхідно ігнорувати під час завантаження.

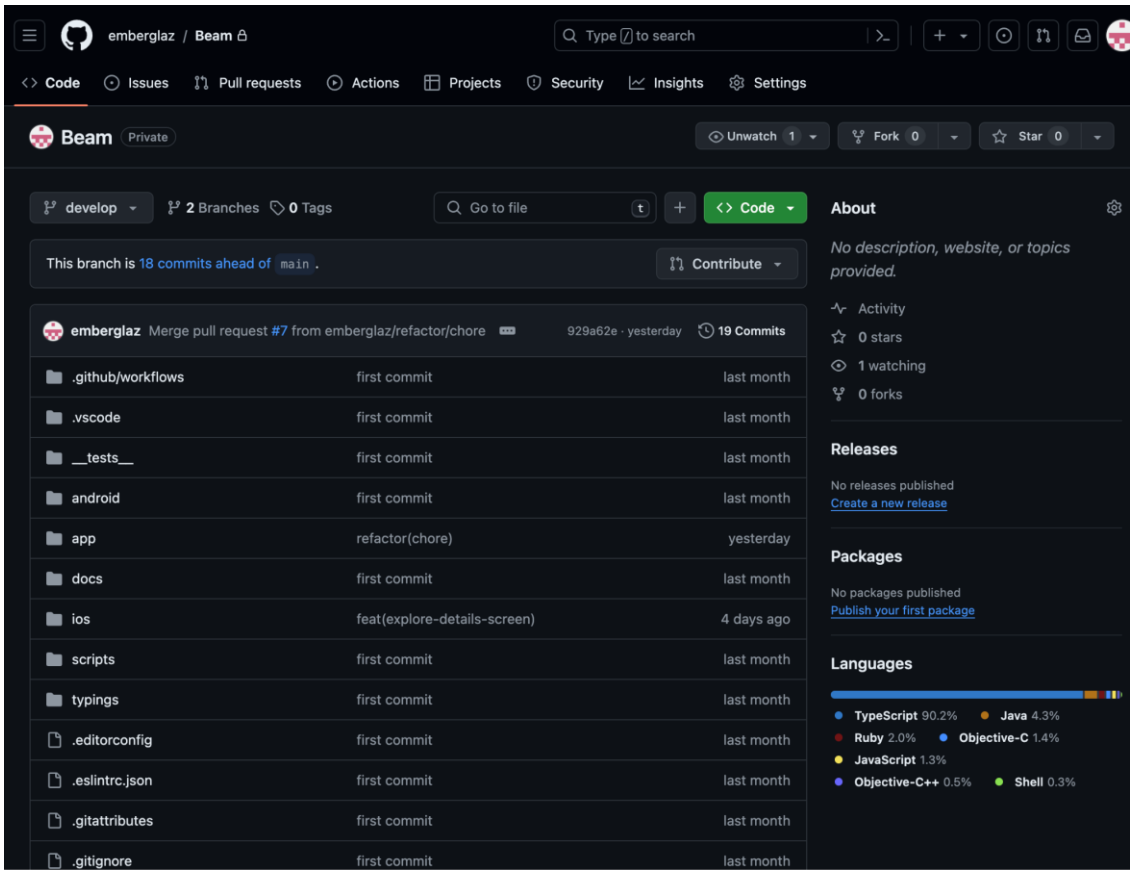

Рисунок 9 — *Система контролю версій Git на прикладі React Native застосунку*

3.3 Модулі і алгоритми

Під час розробки застосунку для забезпечення навчального середовища було розроблено наступні модулі:

1. Модуль автентифікації користувача Firebase.
2. Модуль Firebase Cloud Storage для хмарного сховища даних.
3. Модуль інтерфейсу застосунку.

3.3.1 Модуль автентифікації користувача Firebase

На рисунку 10 та лістингу 1 представлена архітектура модулю автентифікації Firebase:

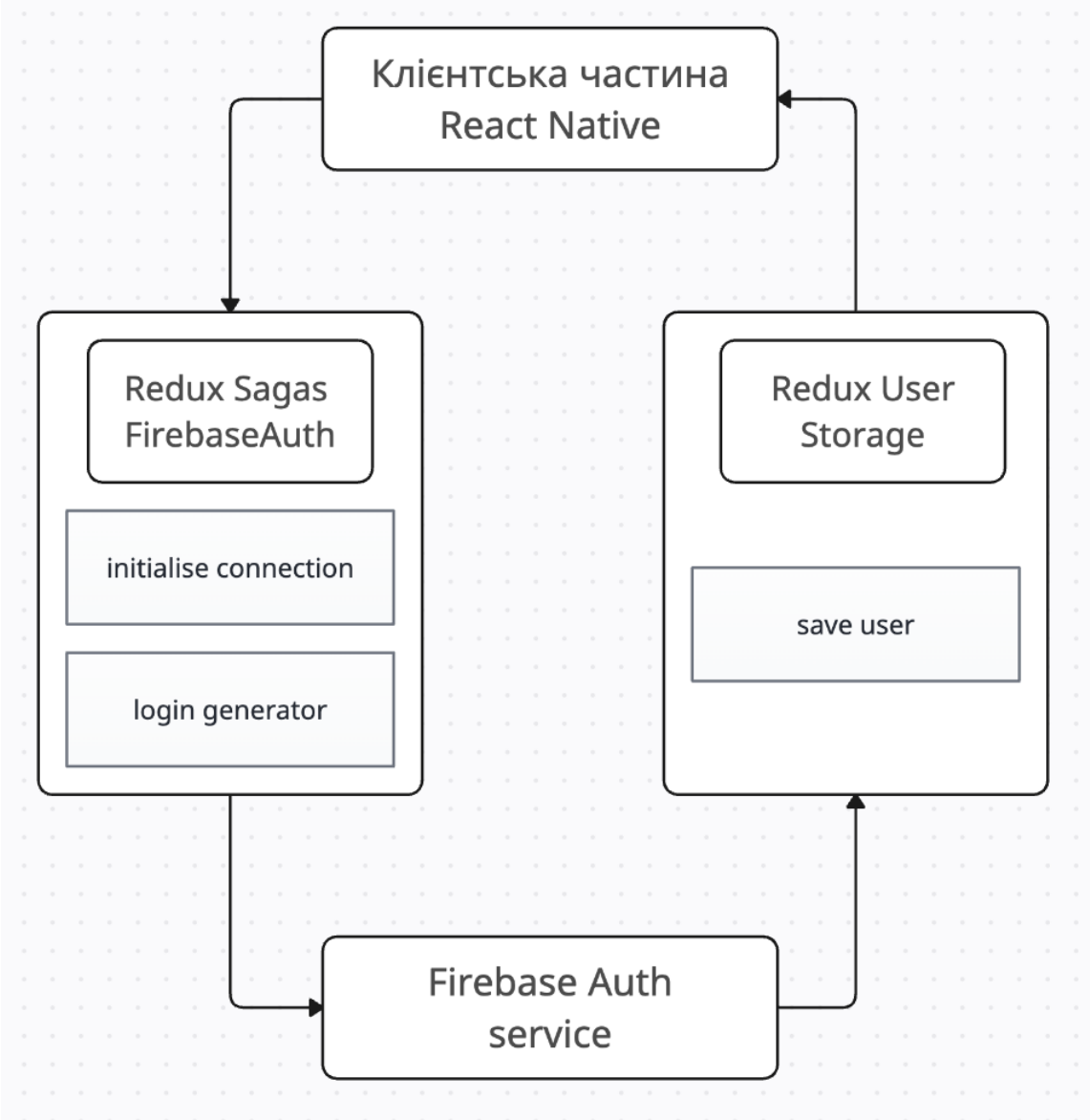

Рисунок *10 — Модуль автентифікації користувача через Firebase Auth*

Лістинг *1 Реалізація модулю автентифікації FirebaseAuth*

```
export const firebaseLoginUser = async (email: string,
password: string) => {
  const loginUserResponse: FirebaseAuthResponse = {
    user: null,
    error: null,
  };
  try {
```

```
    loginUserResponse.user = (await
auth().signInWithEmailAndPassword(email, password)).user;
  } catch (error) {
    loginUserResponse.error = (error as
FirebaseAuthError).message;
  }
  return loginUserResponse;
};

export const firestoreGetUser = async (email: string) => {
  const userId = getCurrentUser();
  const user = await firestore()
    .collection(FirestoreCollections.Users)
    // Filter results
    .where('email', '==', email)
    .get()
    .then(querySnapshot => querySnapshot._docs[0]._data);
  return { ...user, uid: userId };
};
```

Компонент React Native Redux Sagas FirebaseAuth

React Native реалізує структуру зв'язку з сервісом через Redux та Redux-Saga бібліотеки наступним чином:

1. Компонент у необхідний момент часу прокидує дію, що перехоплюється однією з саг, яка підписана на оновлення дій.
2. Сага очікує дію loginUserAction та реалізує авторизацію користувача.
3. Після вдалої авторизації сага викликає збереження користувацької інформації користувача, отриманої після виклику сервіса FirebaseAuth, до сховища Redux.
4. Далі ми викликаємо StaticNavigator, який перенаправляє користувача до головної сторінки.

5. В свою чергу Redux обробляє одержану інформацію та оновлення стану застосунку.

В лістингу 2 наведена головна реалізація обробки дій авторизації генераторами всередині саг.

Лістинг *2 Реалізація обробки дій автентифікації Redux Sagas FirebaseAuth*

```
function* loginUserWorker({ payload: { email, password } }:
LoginUserAction) {
  const loginUserResponse: FirebaseAuthResponse = yield
call(firebaseLoginUser, email, password);

  if (!loginUserResponse.error) {
    const user = yield call(firestoreGetUser, email);

    yield put(userSlice.actions.loginUserSuccess(user));

    yield call(StaticNavigator.replace,
RootStackRouteNames.MainTab);
  } else {
    yield
put(userSlice.actions.loginUserError(loginUserResponse.erro
r));
  }
}
```

Модуль FirebaseAuth Service

Щоб мати можливість використовувати сервіс автентифікації необхідно налаштовати платформу Firebase наступним чином:

1. Зареєструвати обліковий запис на сайті Firebase.
2. Додати та налаштувати по застосунку під кожну з операційних систем додатку.
3. Включити у вкладці налаштувань обробку автентифікації сервером.
4. Включити зберігання інформації користувачів окремо від основної бази даних.

На рисунках 11 та 12 наведено налаштований сервіс зберігання аккаунтів користувачів.

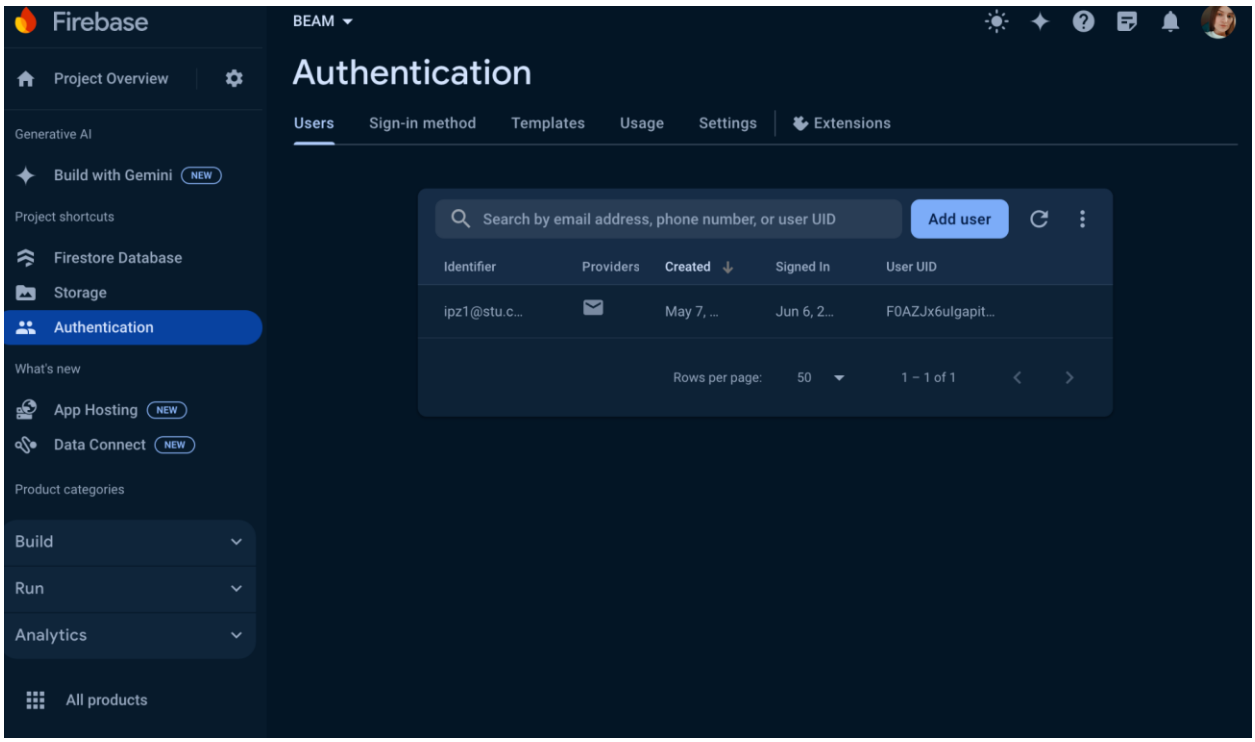

Рисунок *11 — Збережені користувачі платформою Firebase*

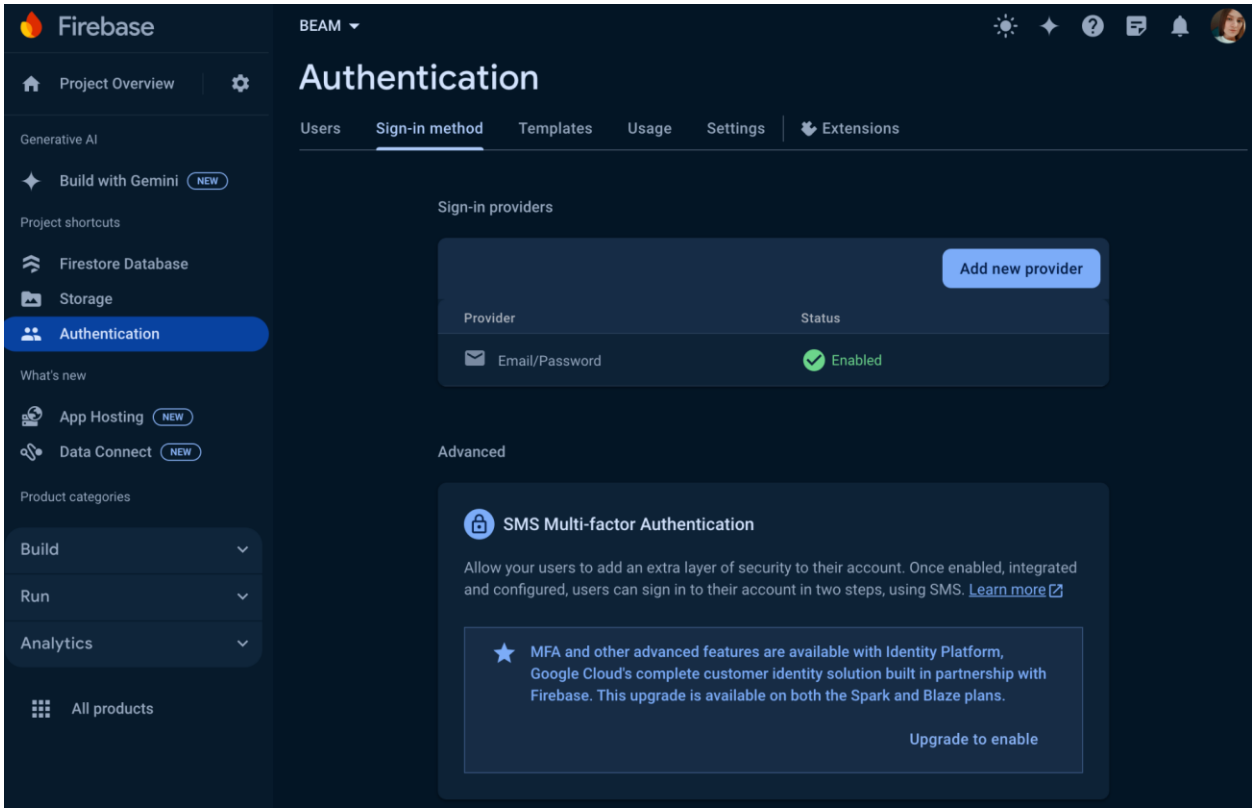

Рисунок *12 — Налаштовані методи авторизації Firebase*

3.3.2 Модуль хмарного сховища даних Firebase Cloud Storage

На рисунку 13 представлена архітектура модулю хмарного сховища даних Firebase Cloud Storage та зв'язок з клієнтською частиною React Native застосунку:

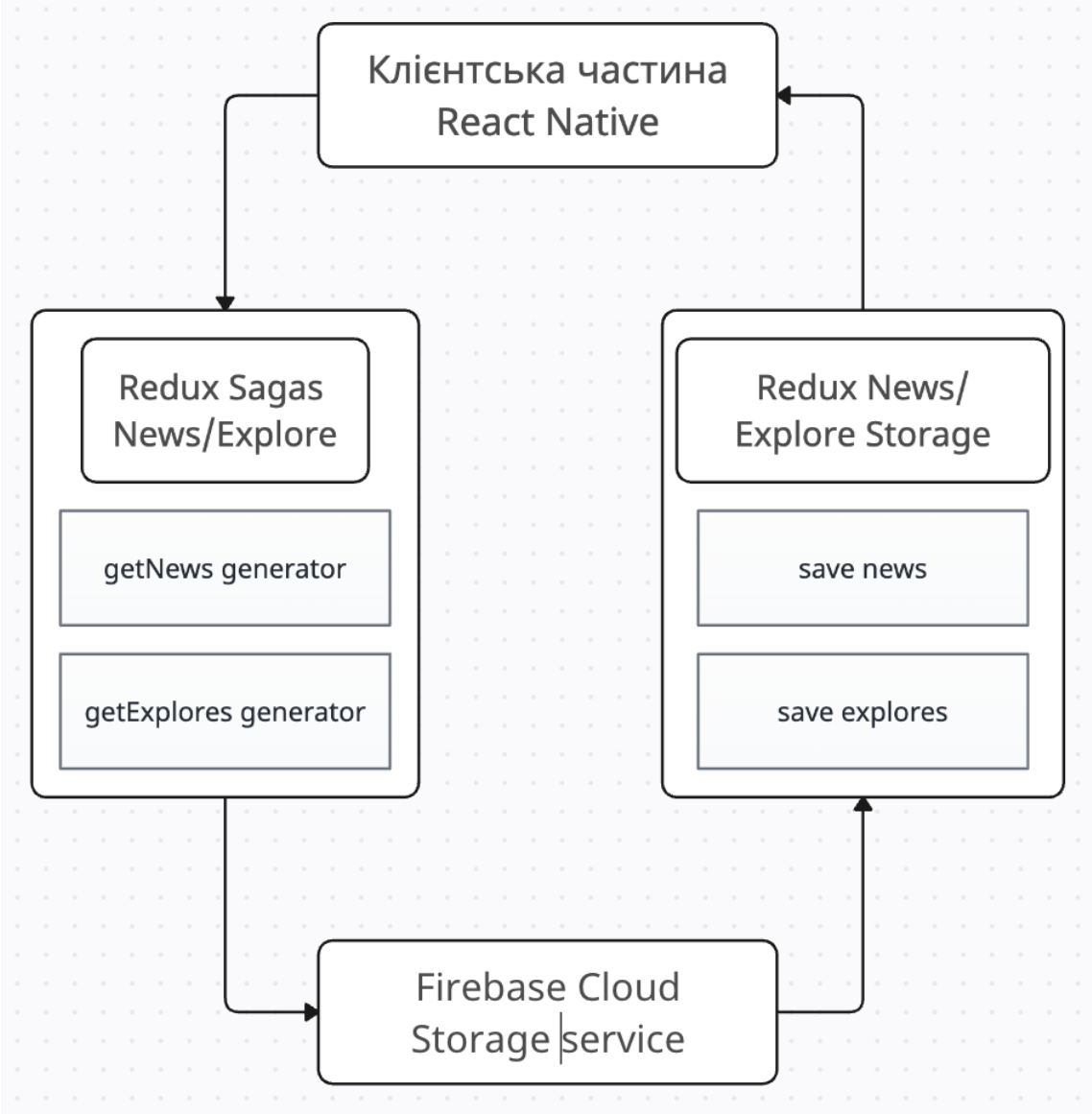

Рисунок *13 — Архітектура модуля хмарного сховища та з'єднання з ним*

Компонент React Native Redux Sagas News/Explore

Для клієнтської частини логіку роботи з хмарним сховищем Firebase Cloud Storage було розділено на дві частини: саги та redux-контейнер.

В лістингу 3 показані основні частини саги News та Explore, а в лістингу 4 наведено приклад отримання колекцій News та Explore з Firebase Cloud Storage.

Проект структурований таким чином, щоб забезпечити легкість підтримки та розширення. Логіка роботи з Firebase Cloud Storage винесена в

окремі модулі, що дозволяє легко змінювати та розширювати функціональність.

Лістинг *3 Саги зв'язку до Firebase Cloud Storage колекцій*

```
function* getNewsWorker() {
  try {
    const response: FirestoreGetNews = yield
call(firestoreGetNews);

    yield
put(newsSlice.actions.getNewsSuccess(convertFirestoreNews(r
esponse)));
  } catch {
    yield put(
      newsSlice.actions.getNewsError('Something went wrong!
Please check your internet connection and try again'),
    );
  }
}

function* getExploresWorker() {
  try {
    const response: FirestoreGetExplores = yield
call(firestoreGetExplores);

    yield
put(exploreSlice.actions.getExploresSuccess(convertFirestor
eExplores(response)));
  } catch {
    yield put(
      exploreSlice.actions.getExploresError(
        'Something went wrong! Please check your internet
connection and try again',
      ),
    );
  }
}
```

Лістинг *4 Реалізація модулю отримання колекцій Firebase Cloud Storage*

```
export const firestoreGetNews = async () => {
```

```
  const news = await
firestore().collection(FirestoreCollections.News).get();
  return news;
};

export const firestoreGetExplores = async () => {
  const explores = await
firestore().collection(FirestoreCollections.Explores).get()
;
  return explores;
};
```

Компонент Firebase Cloud Storage

Для використання сервісу хмарного сховища було налаштовано платформу Firebase наступним чином:

1. Зареєстровано обліковий запис користувача Firebase.
2. Додано два застосунку під кожну з операційних систем.
3. Створена база даних Cloud Storage.
4. Надано дозвіл на запис та читання усіх користувачів застосунка.

На рисунках 14 та 15 представлене налаштування сервісу Firebase Cloud Storage.

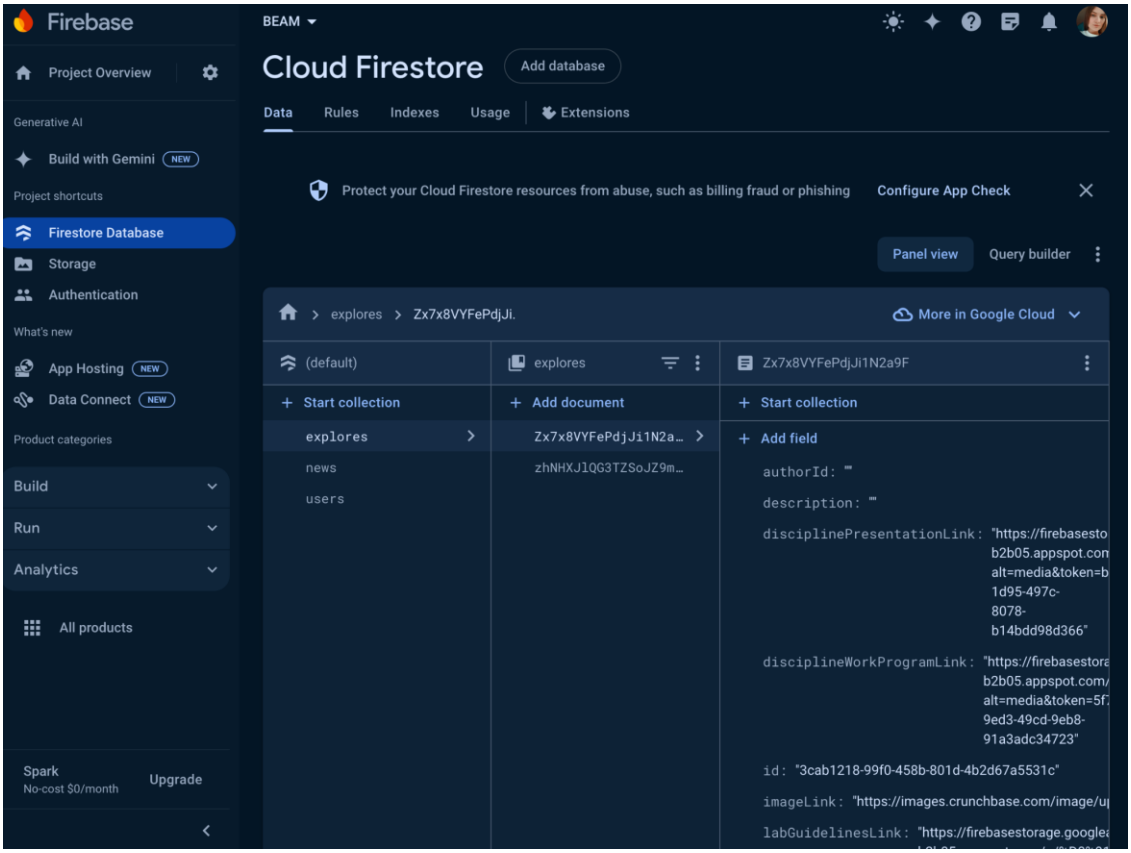

Рисунок *14 — Налаштування колекцій Firebase Cloud Storage*

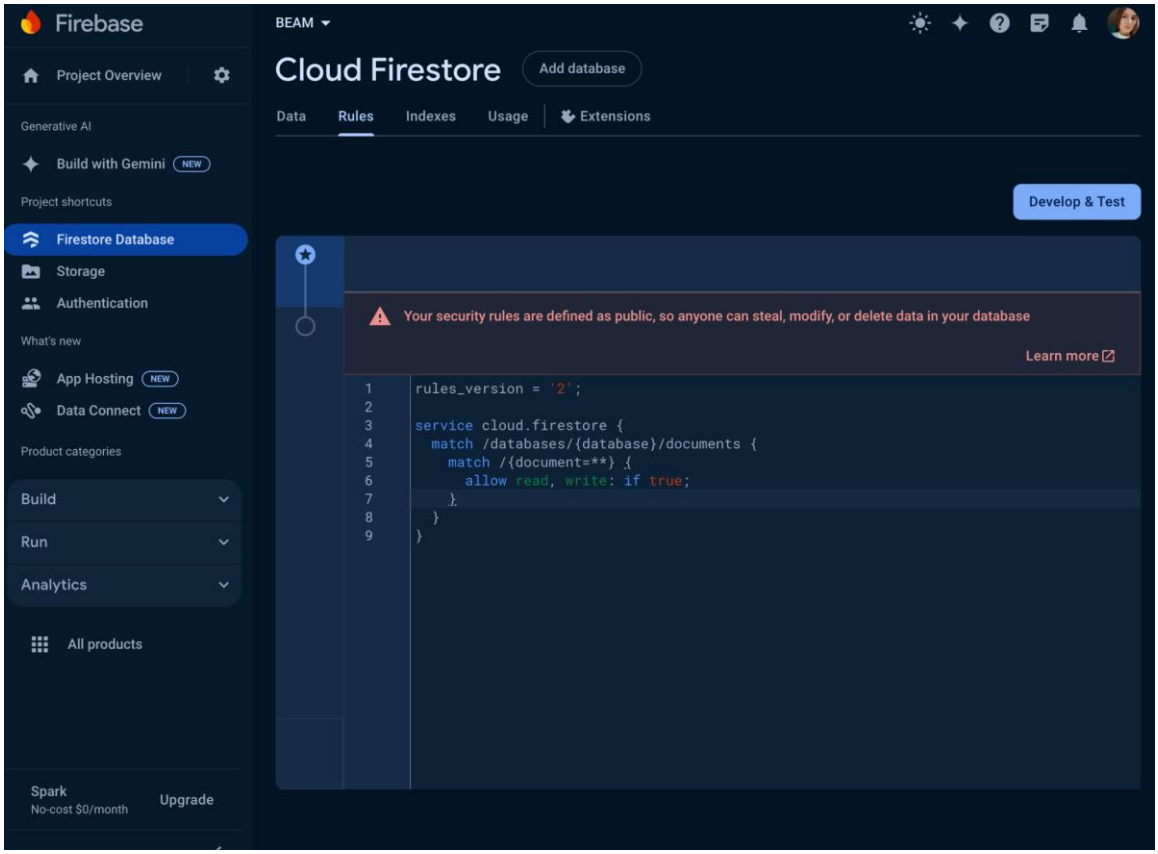

Рисунок *15* — *Налаштування доступу до колекцій*

3.3.3 Модуль інтерфейсу

Інтерфейс розроблений за допомогою патерну MVC (див. рис. 16).

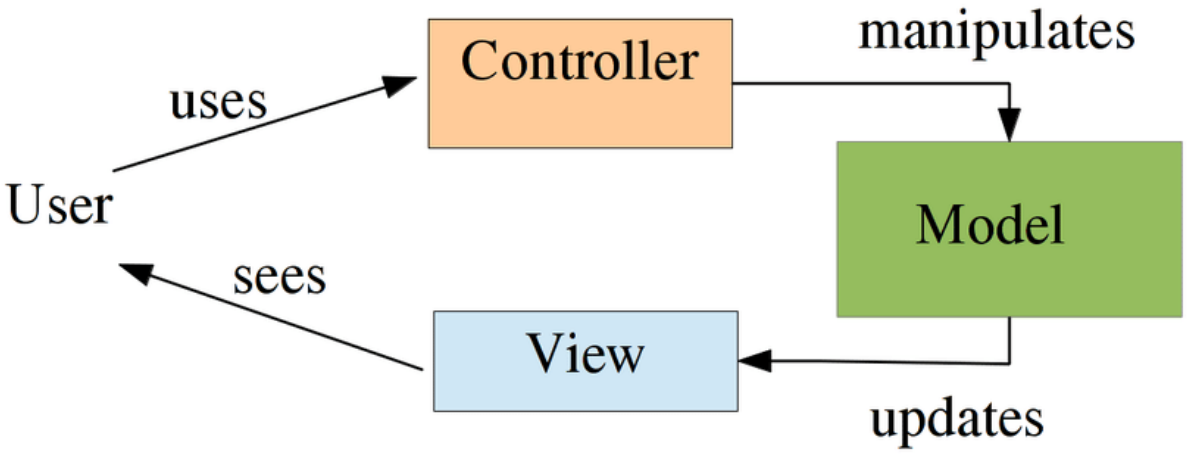

Рисунок *16* — *Паттерн проектування Model-View-Controller*

Модуль інтерфейсу React Native

Кожна сторінка та компонент мають свою реалізацію JSX-розмітки, а стилі для них винесені в окремі файли з суфіксом *.styles.ts. Типізація кожного компонента чи екрану також винесена в окремі файли з суфіксом *.types.ts. Основна логіка і використання styles та types знаходяться в файлі index.ts.

Для організації проекту створено дві директорії: components для компонентів та **screens** для сторінок. Окрема директорія створена для бібліотеки навігації, де розміщені файли, що відповідають за вкладеність та обробку стеків навігації.

Для уникнення дублювання коду, деякі сторінки приймають параметри навігації, що дозволяє їм рендерити та оброблювати логіку зі схожим інтерфейсом.

Директорії **services** та **utils** використовуються для реалізації деяких функціональностей, необхідних у різних частинах програми.

На рисунку 17 представлена структура файлів проекту, що відповідає за інтерфейс.

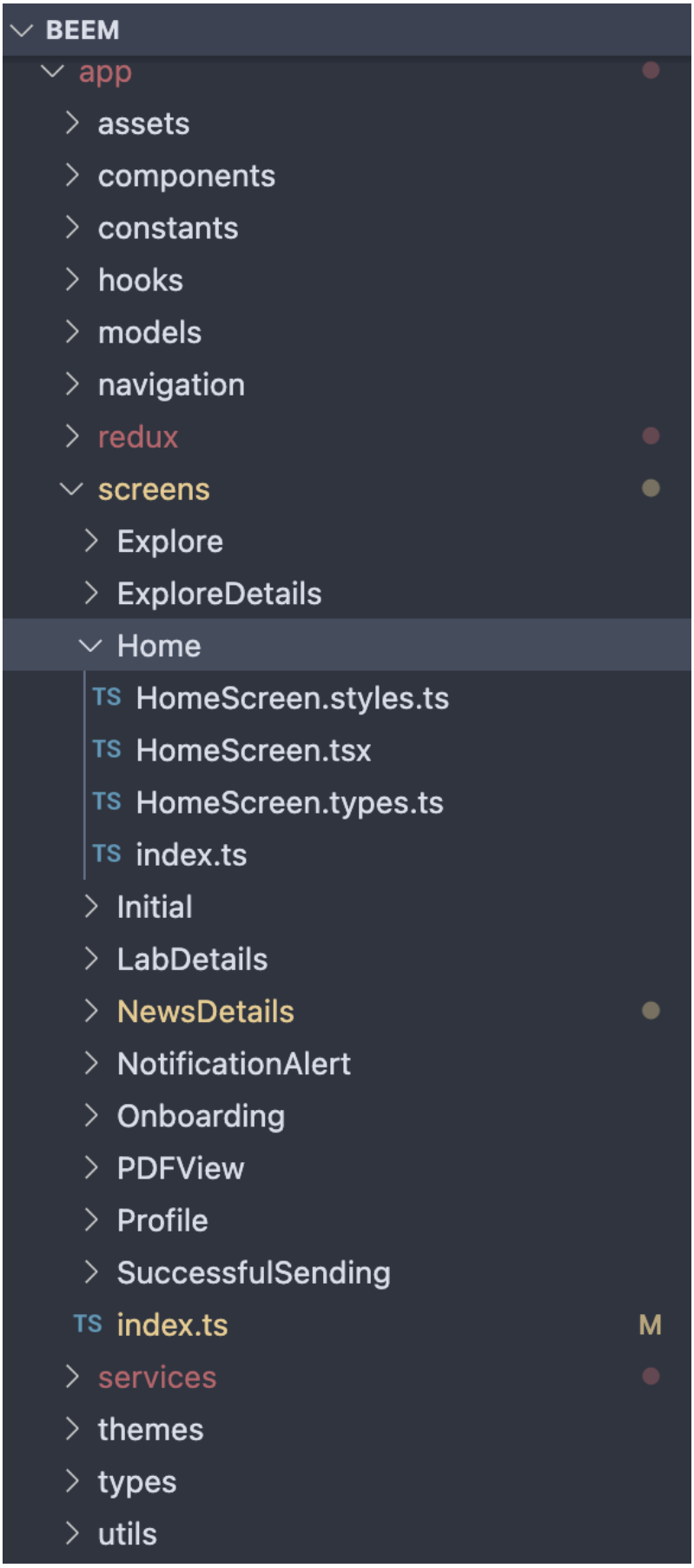

Рисунок *17 — Структура файлів інтерфейсу React Native застосунку*

В лістингу 5 наведена реалізація сторінки HomeScreen React Native застосунку.

Лістинг 5 *Реалізація React Native інтерфейсу HomeScreen*

```
import { values } from 'lodash';
import React, { useCallback } from 'react';
import { FlatList, ListRenderItem, TouchableOpacity, View }
from 'react-native';

import { MainHeader, NewsCard } from '#components';
import { useAppDispatch, useAppSelector, useMount,
useSpecificKeyExtractor } from '#hooks';
import { News } from '#models';
import { HomeStackRouteNames } from '#navigation';
import { newsSlice } from '#redux/slices';
import { generalStyles } from '#utils/styles';

import { styles } from './HomeScreen.styles';
import { HomeScreenProps } from './HomeScreen.types';

export const HomeScreen = ({ navigation }: HomeScreenProps)
=> {
  const dispatch = useAppDispatch();

  const newsCardKeyExtractor =
useSpecificKeyExtractor<News>('new-card-list-child-key',
'id');
  const newsList = useAppSelector(state =>
state.news.news);

  const onCardPress = useCallback(
    (id: string) => {
      navigation.navigate(HomeStackRouteNames.NewsDetails,
{ id });
      return id;
    },
    [navigation],
  );

  useMount(() => {
    dispatch(newsSlice.actions.getNews());
  });
```

```
  const renderItem: ListRenderItem<News> = useCallback(
    ({ item }) => (
      <TouchableOpacity
        style={generalStyles.flex}
        onPress={() => onCardPress(item.id)}
      >
        <NewsCard item={item} />
      </TouchableOpacity>
    ),
    [onCardPress],
  );

  return (
    <View style={generalStyles.whFlex}>
      <MainHeader title="News" />
      <View style={styles.container}>
        <FlatList
          contentContainerStyle={generalStyles.gap8}
          data={values(newsList)}
          keyExtractor={newsCardKeyExtractor}
          renderItem={renderItem}
          showsHorizontalScrollIndicator={false}
          showsVerticalScrollIndicator={false}
          style={generalStyles.flex}
        />
      </View>
    </View>
  );
};
```

Особливості реалізації:

1. Винесені кольори, стилі, константи, та ін. до окремих директорій.
2. Функціональний компонент.
3. Строга типізація за допомоги TypeScript.
4. Використання загальних стилів застосунку.
5. Використання пресетів.
6. Використання SVG зображень.

3.4 Проект інтерфейсу

3.4.1 Вимоги до інтерфейсу

Основні вимоги до інтерфейсу:

1. Інтерфейс повинен однаково добре працювати на різних платформах, таких як iOS та Android.
2. Інтерфейс має бути зрозумілим з першого погляду.
3. Користувачі повинні легко знаходити потрібні функції та опції.
4. Інтерфейс повинен бути адаптивним до різних розмірів екранів та роздільних здатностей.

Функціональні вимоги:

1. Лаконічний та інформативний екран з основними функціями та останніми оновленнями.
2. Легкий доступ до найважливіших розділів додатку.
3. Зручна навігація з мінімальною кількістю кліків для досягнення будь-якої функції.
4. Використання нижньої навігаційної панелі для швидкого доступу до основних розділів.
5. Зручна система для входу користувачів.
6. Можливість відновлення паролю користувача.
7. Профіль користувача
8. Доступ до курсів, лекцій та завдань.

Візуальні вимоги:

1. Сучасний та привабливий дизайн, який стимулює користувачів до взаємодії з додатком.
2. Використання брендових кольорів та стилів.

3. Використання спокійних кольорів, які не втомлюють очі під час тривалого використання.
4. Високий контраст для покращення читабельності тексту та елементів.
5. Чіткі та читабельні шрифти, зручні для читання на різних розмірах екранів.
6. Адекватні розміри шрифтів для заголовків, підзаголовків та основного тексту.
7. Зрозумілі та легко впізнавані іконки, що інтуїтивно пояснюють функції.
8. Використання стандартних іконок для спрощення розуміння інтерфейсу.

Технічні вимоги:

1. Швидка робота інтерфейсу без затримок та зависань.
2. Оптимізація графічних елементів для швидкого завантаження та плавної роботи.

3.4.1 Огляд інтерфейсу

Мобільний застосунок не надає можливість створення користувацького облікового запису всередині додатку, але дає можливість авторизуватися у додатку під корпоративною поштою (див. рис. 18-20).

На головному екрані застосунку (див. рис. 21) користувач може обрати яку новину переглянути детальніше (див. рис. 22) або перейти на іншу сторінку за допомогою нижньої панелі навігації.

Перейшовши на сторінку з доступними курсами (див. рис. 23) користувач може обрати зі списку курс та переглянути його детальніше (див. рис. 24).

Рисунок *18* — *Початкова сторінка для вибору шляху авторизації*

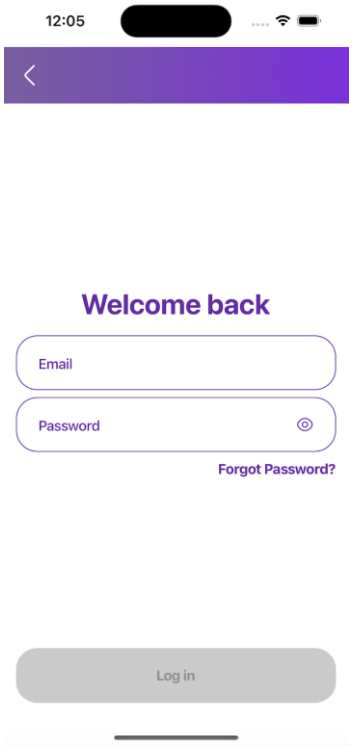

Рисунок *19* — *Сторінка логіну*

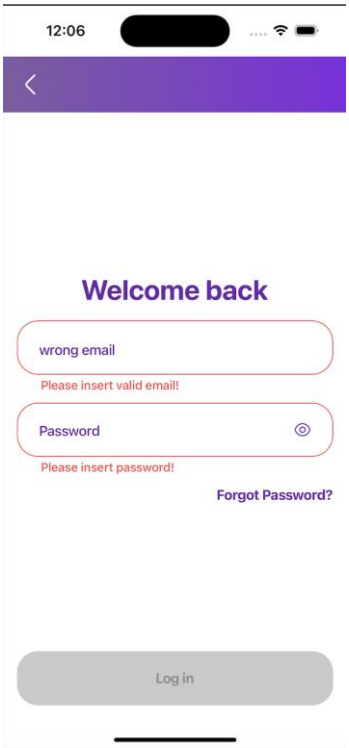

Рисунок *20 — Сторінка логіну з помилками валідації*

Рисунок *21 — Головна сторінка застосунку*

Рисунок *22* — *Сторінка NewsDetails*

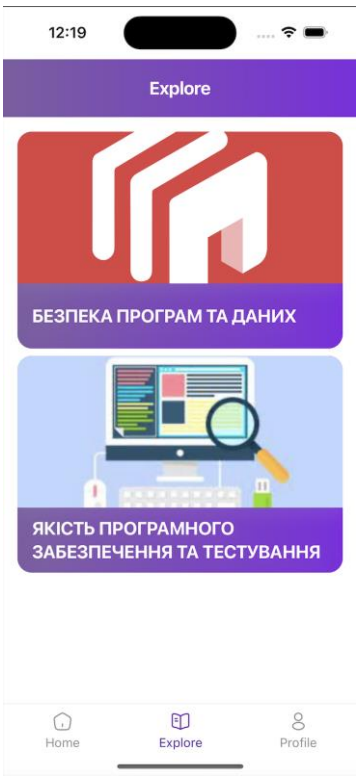

Рисунок *23* — *Сторінка ExploreScreen*

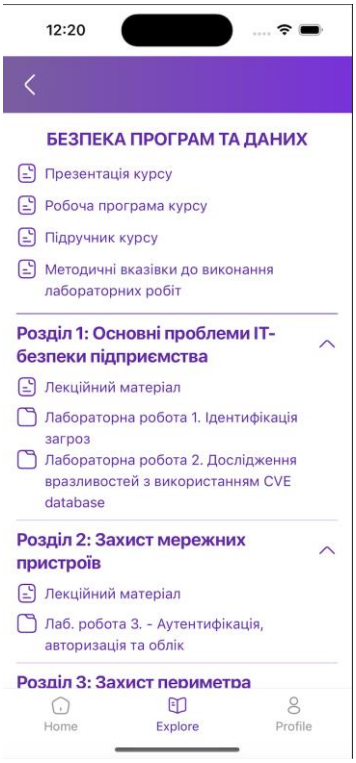

Рисунок *24 — Сторінка ExploreDetailsScreen*

На сторінці *ExploreDetailsScreen* користувач може прочитати матеріали курсу, переглянути файли натиснувши на відповідний файл, це навігує його на сторінку для перегляду файлів (див. рис. 25), або переглянути завдання з розділів (див. рис. 26-27). На сторінці для перегляду файлів можна збільшувати або зменшувати обраний файл подвійним тапом по экрану або за допомогою спеціальних нативних жестів (див. рис. 28).

Перейшовши на сторінку Profile (див. рис. 29) користувач може переглянути свої дані та вийти з свого облікового запису.

Користувач може скористатися опцією Забули пароль на сторінці з авторизацією, натиснувши на відповідну кнопку, для зміни пароля (див. рис. 30), якщо обліковий запис з такою поштою існує, то користувачу прийде лист з інструкцією, а додаток навігує на сторінку *SuccessfulSendingScreen* (див. рис. 31).

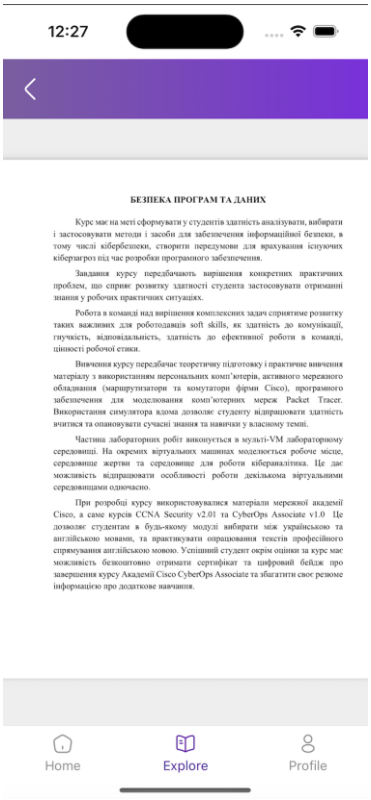

Рисунок *25* — *Сторінка PDFViewScreen*

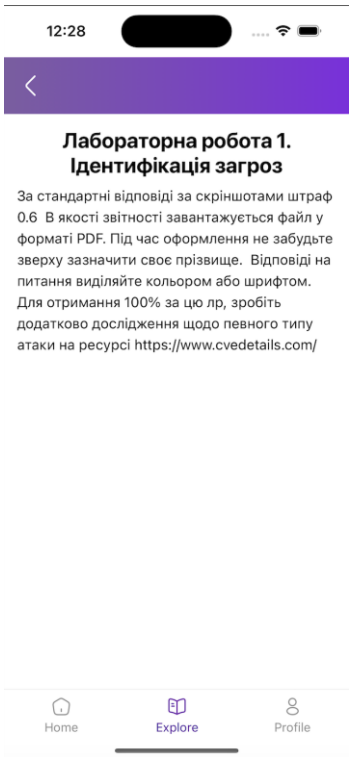

Рисунок *26* — *Сторінка LabDetailsScreen*

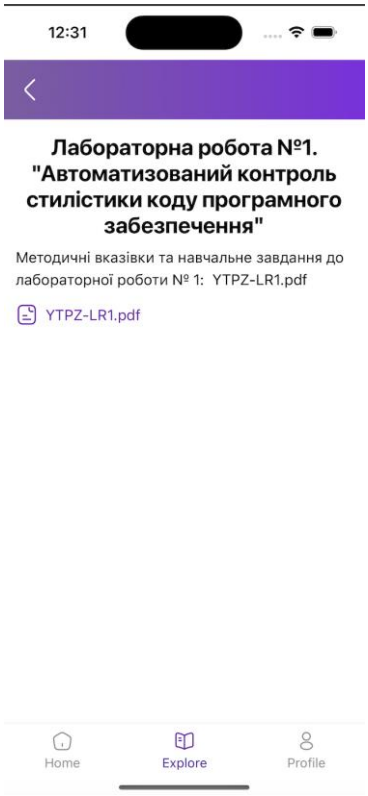

Рисунок *27 — Сторінка LabDetailsScreen з файлом*

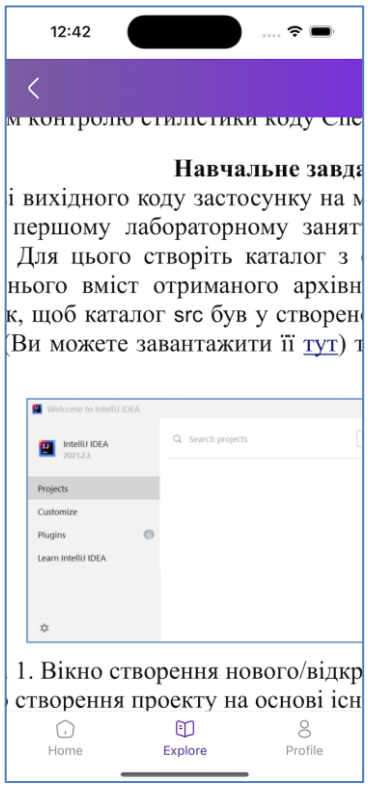

Рисунок *28 — Сторінка PDFViewScreen з збільшеним файлом*

Рисунок *29* — *Сторінка ProfileScreen*

Рисунок *30* — *Сторінка ResetPasswordScreen*

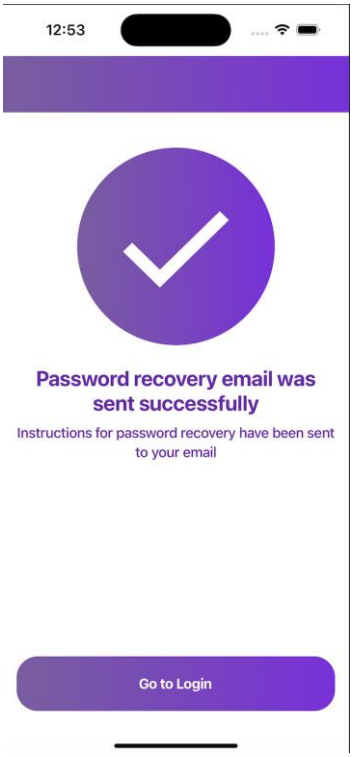

Рисунок *31 — Сторінка SuccessfulSendingScreen*

3.4.3 Тестування

Було виконане ретельне тестування інтерфейсу на різних пристроях та платформах для забезпечення стабільності та відсутності багів.

Використання емуляторів та реальних пристроїв для перевірки адаптивності та продуктивності.

ВИСНОВКИ

1. Розробка кросплатформних мобільних додатків є актуальною задачею. Під час аналізу технологій для створення таких додатків було складено список платформ, які відповідають найсучаснішим стандартам. Також було визначено набір технологій та інструментів розробки, їх продуктивність, рівень інтеграції та зручність використання.
2. Проведено дослідження та аналіз сучасних методів створення кросплатформних мобільних додатків.
3. Основними інструментами для реалізації кросплатформеного мобільного додатку було обрано мови програмування JavaScript та TypeScript. Вони надають можливість використовувати додаток на будь-якій платформі, перекладаючи вихідний код на нативні операційні системи.
4. Було налаштовано сервіс автентифікації та хмарного сховища даних Firebase. Та вивчена можливість використання хмарних сховищ Firebase у поєднанні з платформою React Native для створення кросплатформних мобільних додатків, зокрема управління колекціями та облік користувачів.
5. Було розроблено програмний застосунок для демонстрації можливостей технологій React Native для створення кросплатформних застосунків під основні мобільні операційні системи, такі як Android та iOS.
6. Розробка мобільного додатку для навчання є актуальним та перспективним напрямком розвитку в сфері освіти. Використання кросплатформних технологій дозволяє забезпечити доступність та універсальність додатку для всіх користувачів, що сприяє підвищенню якості освіти та розширенню можливостей для навчання.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Boduch A. , React and React Native: A complete hands-on guide to modern web and mobile development with React.js, 3rd Edition. 2020. 526 p.

2. Adam D. Scott, JavaScript Everywhere: Building Cross-Platform Applications with GraphQL, React, React Native, and Electron. 2020. 344 p.

3. Ward D., React Native Cookbook: Step-by-step recipes for solving common React Native development problems, 2nd Edition. 2019. 618 p.

4. DerksR., React.js Projects: Build 12 real-world applications from scratch using React, React Native, and React 360. 2020. 432 p.

5. Paul A., React Native for Mobile Development: Harness the Power of React Native to Create Stunning iOS and Android Applications, 2nd Edition. 2019. 237 p.

6. Dabit N., React Native in Action. 2019. 320 p.

7. Zammetti F., Practical React Native: Build Two Full Projects and One Full Game using React Native. 2018. 334 p.

8. Grzesiukiewicz M., Hands-On Design Patterns with React Native: Proven techniques and patterns for efficient native mobile development with JavaScript. 2018. 302 p.

9. Boduch A., React and React Native, 2nd Edition. 2018. 540 p.

10. Abbot D., Fullstack React Native: The Complete Guide to React Native. 2017. 552 p.

11. Rodriguez E. Martinez, React: Cross-Platform Application Development with React Native. 2018. 182 p.

12. Bush E., JavaScript Applications with Node.js, React, React Native and MongoDB. 2018. 392 p.

13. Eisenman B., Learning React Native: Building Native Mobile Apps with JavaScript, 2nd Edition. 2017. 244 p.

14. Lebensold J., React Native Cookbook: Bringing the Web to Native Platforms. 2018. 176 p.

15. Boduch A., React and React Native. 2017. 604 p.

16. Martinez E. R., React Native Blueprints. 2017. 346 p.

17. Novick V., React Native – Building Mobile Apps with JavaScript. 2017. 434 p.

18. Villa C., Bershadskiy S., React Native Cookbook. 2017. 437 p.

19. Burd B., Flutter For Dummies. 2020. 384 p.

20. Singh A., Mobile Deep Learning with TensorFlow Lite, ML Kit and Flutter: Build scalable real-world projects to implement end-to-end neural networks on Android and iOS. 2020. 380 p.

21. Alessandria S., Flutter Projects: A practical, project-based guide to building real-world cross-platform mobile applications and games. 2020. 490 p.

22. Zaccagnino C., Programming Flutter: Native, Cross-Platform Apps the Easy Way. 2020. 370 p.

23. Biessek A., Flutter for Beginners: An introductory guide to building cross-platform mobile applications with Flutter and Dart 2. 2019. 512 p.

24. Windmill E., Flutter in Action. 2020. 310 p.

25. Payne R., Beginning App Development with Flutter: Create Cross-Platform Mobile Apps. 2019. 309 p.

26. Cheng F., Flutter Recipes: Mobile Development Solutions for iOS and Android. 2019. 522 p.

27. Napoli M. L., Beginning Flutter: A Hands On Guide to App Development. 2019. 528 p.

28. Zammetti F., Practical Flutter: Improve your Mobile Development with Google’s Latest Open-Source SDK. 2019. 396 p.

29. Mainkar P., Google Flutter Mobile Development Quick Start Guide: Get up and running with iOS and Android mobile app development. 2019. 152 p.

Декларація
академічної доброчесності
здобувача ступеня вищої освіти ЗНУ

Я, Гнибіденко Анна Вікторівна, студентка _4_ курсу, форми навчання денної, Інженерного навчально-наукового інституту, спеціальність 121 Інженерія програмного забезпечення, адреса електронної пошти ipz20bd-103@stu.zsea.edu.ua, — підтверджую, що написана мною кваліфікаційна робота на тему «**Розробка мобільного кросплатформного додатку для забезпечення навчального середовища** » відповідає вимогам академічної доброчесності та не містить порушень, що визначені у ст.42 Закону України «Про освіту», зі змістом яких ознайомлена.

- заявляю, що надана мною для перевірки електронна версія роботи є ідентичною її друкованій версії;

згодна на перевірку моєї роботи на відповідність критеріям академічної доброчесності у будь-який спосіб, у тому числі за допомогою інтернет-систем, а також на архівування моєї роботи в базі даних цієї системи.

Дата 14.06.2024 Підпис _____ Гнибіденко Анна Вікторівна (студент)

Дата 14.06.2024 Підпис _____ Попівщий Василь Іванович (науковий керівник)