Національний університет «Полтавська політехніка імені Юрія Кондратюка»
(повне найменування вищого навчального закладу)

Навчально-науковий інститут інформаційних технологій і робототехніки
(повна назва факультету)

Кафедра комп‘ютерних та інформаційних технологій і систем
(повна назва кафедри)

Пояснювальна записка

до дипломного проекту (роботи)

магістра
(рівень вищої освіти)

на тему

Розроблення мультиплатформенного мобільного застосунку спрямованого навчити в’язати вузли

Виконав: студент 2 курсу, групи 601-ТН
спеціальності
122 Комп’ютерні науки
(шифр і назва спеціальності)
Харламов І.Є.
(прізвище та ініціали)

Керівник Капітон А. М.
(прізвище та ініціали)

Рецензент
(прізвище та ініціали)

Полтава – 2021 року

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «ПОЛТАВСЬКА ПОЛІТЕХНІКА ІМЕНІ ЮРІЯ КОНДРАТЮКА»

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І РОБОТОТЕХНІКИ

КАФЕДРА КОМП'ЮТЕРНИХ ТА ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ І СИСТЕМ

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

спеціальність 122 «Комп'ютерні науки»

на тему

«Розроблення мультиплатформенного мобільного застосунку спрямованого навчити в'язати вузли»

Студента групи 601-ТН Харламова Іллі Євгенійовича

Керівник роботи
доктор педагогічних наук,
доцент Капітон А.М.

Завідувач кафедри
кандидат технічних наук,
доцент Головко Г.В.

Полтава – 2021

РЕФЕРАТ

Пояснювальна записка містить: 77 сторінок, 44 рис., 41 джерело, додатки.

Об’єкт дослідження – розроблення мобільного застосунку спрямованого навчити в’язати вузли.

Предмет дослідження – інструментальні засоби розробки мобільних додатків.

Мета кваліфікаційної роботи – розроблення, тестування та введення в експлуатацію мобільного застосунку спрямованого навчити в’язати вузли.

Ключові слова: автоматизована інформаційна система, фреймворк, React, React Native, TypeScript, JavaScript, TS, JS, SQLite мобільний додаток, база даних, система керування базами даних, мобільний додаток, застосунок, IOS, Android, смартфон.

ABSTRACT

Explanatory note contains: 77 pages, 44 pictures, 41 references, additions.

Object of research – development of a mobile application aimed at learning to knit knots.

Subject of research – mobile application development tools.

The purpose of the qualification work – development, testing and commissioning of a mobile application aimed at learning to knit knots.

Keywords: automated information system, framework, React, React Native, TypeScript, JavaScript, TS, JS, SQLite mobile application, database, database management system, mobile application, application, iOS, Android, smartphone.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ ТА СКОРОЧЕНЬ

БД – база даних.

ОС – операційна система.

ПК – персональний комп’ютер.

СКБД – система керування базами даних.

JavaScript (JS) – динамічна, об'єктно-орієнтована прототипна мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується для створення сценаріїв веб-сторінок, що надає можливість на стороні клієнта (пристрої кінцевого користувача) взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки.

TypeScript(TS) – мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript.

Технічне завдання (ТЗ) – вихідний документ для проектування споруди чи промислового комплексу, конструювання технічного пристрою (приладу, машини, системи керування тощо), розробки автоматизованої системи, створення програмного продукту або проведення науково-дослідних робіт (НДР) у відповідності до якого проводиться виготовлення, приймання при введенні в дію та експлуатація відповідного об'єкту.

Інтерфейс користувача (ІК, UI) – сукупність засобів для обробки та відображення інформації, максимально пристосованих для зручності користувача.

Застосунок – користувацька комп'ютерна програма, що дає змогу вирішувати конкретні прикладні задачі користувача.

React – відкрита JavaScript бібліотека для створення інтерфейсів користувача, яка покликана вирішувати проблеми часткового оновлення вмісту веб-сторінки, з якими стикаються в розробці односторінкових застосунків. Розробляється Facebook, Instagram і спільнотою індивідуальних розробників.

React Native – React Native це JavaScript-фреймворк для написання реальних мобільних додатків з рендерингом для iOS і Android. Він заснований на React, JavaScript-бібліотеці Facebook для створення інтерфейсів користувача, але замість орієнтації на браузер він націлений на мобільні платформи. Іншими словами: веб-розробники тепер можуть писати мобільні програми, які виглядають і відчуваються по-справжньому «рідними», і все це з комфортом бібліотеки JavaScript, яку ми вже знаємо та любимо. Крім того, оскільки більшість написаного вами коду може використовуватися різними платформами, React Native спрощує одночасну розробку як для Android, так і для iOS.

ВСТУП

У сучасному світі, особливо в період карантину, дуже гострим є питання для кожного, де відпочити, куди піти на вихідних, як весело і цікаво провести час? Задаючись цим питанням, була знайдена відповідь, що в період ізоляції, найкращим вибором для відпочинку став туристичний похід, подалі від великої скупченості людей. Що взагалі таке – туристичний похід?

Туристичний похід – спланована подорож організованої групи туристів, або «соло» туриста, з використанням активних форм пересування за визначеним маршрутом, під час проходження якого можливе подолання природних перешкод: перевалів, порогів, печер тощо різних категорій та ступенів складності.

Група туристів самостійно згуртовується, назначає керівника, обирає район мандрівки, розробляє маршрут та визначає час проведення походу, складає харчовий раціон на час походу, готує спорядження тощо. Такі походи здійснюються тільки у вільний від роботи та навчання час — під час відпустки чи канікул і переважно коштом самих туристів.

Зав'язування вузлів може виявитися напрочуд цінною навичкою в поході. Від встановлення намету до першої допомоги - ніколи не знаєш, коли може знадобитися зав'язати міцний вузол. Зараз існують сотні типів вузлів, кожен з яких корисний для різних ситуацій та занять, знання цих шести основних вузлів – добрий початок для відпочиваючих та всіх, хто вирушає у дику природу. Отримайте перевагу перед своїми знаннями в області зав'язування вузлів, і ви будете готові до багатьох ситуацій. Більшість вузлів можуть бути знайдені в «Книзі вузлів Ешлі».

«Книга вузлів Ешлі» або аннлійською «The Ashley Book of Knots» (далі використовуватиметься абривіатура ABOK) – фундаментальна праця з вив'язування вузлів різного призначення[1]. Перше видання книги вийшло в 1944 р. Залишається однією з найбільш повних, знаменитих і популярних книг про вузли. З 1944 по 1993 рік тільки в США і Великій Британії була перевидана

12 разів. Написана американським художником, письменником і моряком Кліффордом Ешлі, що працював над книгою понад 11 років.

Саме тому метою роботи є розробка та створення мобільного застосунку «Knots3D», який зможе навчити користувачів зав’язувати більшість з відомих на сьогодні вузлів.

РОЗДІЛ 1
АНАЛІТИЧНИЙ ОГЛЯД ФРЕЙМВОРКУ REACT-NATIVE ДЛЯ РОЗРОБЛЕННЯ МУЛЬТИПЛАТФОРМЕННИХ МОБІЛЬНИХ ЗАСТОСУНКІВ

1.1 Опис предметної області

Однією з найбільш інтригуючих галузей розробки програмного забезпечення завжди була мобільна розробка, оскільки вона дає досить унікальну можливість для команди з однієї особи створити за порівняно короткий проміжок часу сучасну, корисну та значущу програму.

Мобільний розвиток також являє собою підприємницьку можливість, яка є в межах досяжності більшості програмістів. Не можна сказати, що амбітний розробник програмного забезпечення не міг самостійно створити веб-додаток або комп'ютерний додаток, але розробка мобільних пристроїв набагато доступніша, оскільки очікується, що мобільні додатки будуть невеликими та унікальними.

Однак мобільні розробки представляють для сольного розробника більше, ніж просто можливість створити власний проект — це, мабуть, майбутнє розвитку, оскільки мобільні пристрої стають все більшими і більшими частинами нашого життя.

1.2 Основні платформи мобільного розвитку

На сьогоднішній день існує немало мобільних операційних систем, але дві з найбільш широко прийнятних та популярних – це Apple iOS [2] та Google Android [3]. Ці дві мобільні системи беруть різні підходи до мобільної операційної системи.

Apple розповсюджує єдині прилади, що охоплюють підтримують IOS, і це називають підходом «Wanted Garden», в якому Apple регулює всі мобільні

програми та послуги, які можуть працювати на пристроях IOS. Apple розробила iOS, щоб запустити додатки на власному ядрі XNU. Apple також випустила кілька пристроїв, специфічних мобільних операційних систем, таких як WatchOS для Apple Watch та iPad OS для планшетів iPad.

Рисунок 1.1 – Apple iOS 15

Google приймає інший підхід з Android, який є відкритим джерелом. Це означає, що мобільний пристрій OEMS (виробники оригінального обладнання) можуть змінити вихідний код Android та налаштувати його, щоб відповідати своїм пристроям. Android працює на ядрі Linux.

Рисунок 1.2 – Clear Google Android

Існують інші мобільні операційні системи, але їхні ставки усиновлення знаходяться нижче, ніж iOS та Android. Ці інші операційні системи включають Kaios, Sailfish OS та Huawei Harmony OS.

KaiOS[4], заснований на припиненому Mozilla Firefox OS, запускається переважно на «німих» телефонах, також відомих як функціональні телефони. Ці мобільні пристрої мають дуже обмежену обчислювальну потужність, але найновіша версія KaiOS може принести більший функціонал, так як там можуть бути встановлені як App Store, так помічник Google.

Sailfish ОС[5] базується на декількох проектах з відкритим вихідним кодом та запускаються переважно на смартфонах та планшетних комп'ютерах з Jolla, яка розробила операційну систему та мобільні пристрої Sony. Sailfish ОС та KaiOS обидва запускаються на ядрі Linux.

Harmony OS[6], спочатку випущена в серпні 2019 року, також може працювати на пристроях LoT. Пристрої Huawei зазвичай запускають операційну систему Android, але Harmony OS може замінити Android на ці пристрої в майбутньому. Harmony OS працює на мікроядрах, що розробив Huawei.

1.3 Основні типи мобільної розробки

Будь-хто, хто планує побудувати додаток для свого бізнесу, неминуче доведе відповідати на питання: який тип мобільного додатка ми будуємо?

Існує три основних типи мобільних додатків[7] (рис. 1.3), якщо класифікувати їх за допомогою технології, яка використовується для їх коду:

- native програми створюються для однієї конкретної платформи або операційної системи;
- веб-програми є чутливими версіями веб-сайтів, які можуть працювати на будь-якому мобільному пристрої або ОС, оскільки вони доставляються за допомогою мобільного браузера;

- гібридні додатки є комбінаціями як рідних, так і веб-додатків, але загорнуті до рідного додатка, надаючи йому можливість мати свій значок або завантажувати з App Store.

1. Native програми

Native програми побудовані спеціально для операційної системи мобільного пристрою (ОС). Таким чином, ви можете мати внутрішні програми для мобільних пристроїв Android або IS-програми IOS, не кажучи вже про всі інші платформи та пристрої. Оскільки вони побудовані для однієї платформи, ви не можете змішувати та відповідати - скажімо, скористайтеся програмою BlackBerry на телефоні Android або використовуйте додаток iOS на телефоні Windows.

Використовувана технологія: нативні програми кодуються за допомогою різноманітних мов програмування. Деякі приклади включають: Java, Kotlin, Python, Swift, об'єктивний-C, C ++ та React Native.

2. Веб-програми

Веб-програми поводяться аналогічно новими програмами, але доступні через веб-браузер на своєму мобільному пристрої. Вони не окремі програми в сенсі, щоб завантажити та встановити код у своєму пристрої. Вони насправді являються веб-сайтами, які адаптують свій інтерфейс користувача до пристрою. Фактично, коли ви зіткнулися з опцією «Встановити» Веб-додаток, він часто просто закладає URL-адресу веб-сайту на своєму пристрої.

Один вид веб-додатків — це progressive web-application (PWA[8]), яка в основному є власним додатком, що працює всередині браузера.

3. Гібридні додатки

Останнім типом, про який піде мова в цьому підрозділі є гібридні програми. Це веб-додатки, які виглядають і відчувають себе як нативні програми. Вони можуть мати значок програми домашнього екрана, чуйний дизайн, швидка продуктивність, навіть зможе працювати в автономному режимі, але вони дійсно веб-програми, зроблені, щоб виглядати рідним.

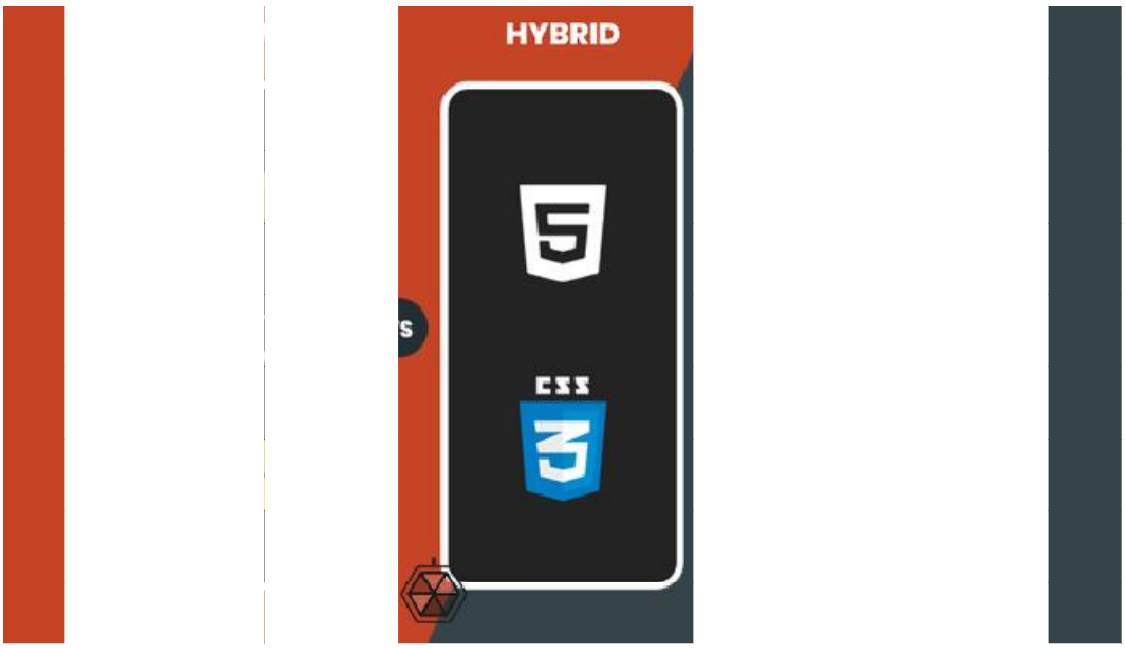

Рисунок 1.3 – Типи мобільних застосунків

1.4 Що таке React-Native?

React Native – це JavaScript Framework для написання реальних, нативно відрендерених мобільних застосунків для IOS та Android. Він заснований на React Library JavaScript Facebook для побудови користувацьких інтерфейсів, але замість того, щоб орієнтувати веб-переглядач, він спрямований на мобільні платформи. Іншими словами: веб-розробники тепер можуть писати мобільні програми, які виглядають і відчувають себе справді «рідними», все за допомогою комфорту бібліотеки JavaScript, які всі вже знають і люблять. Крім того, оскільки більша частина коду, який пишеться, можна розділити між платформами, React Native[9] полегшує мультиплатформенну розробку як для Anadroid, так і для iOS.

Подібно до ReactJS, React Native додатки, використовуються з використанням суміші JavaScript та XML-esque, відомий як JSX. Потім, під капотом, React Native як «міст», викликає нативний рендеринг API у Object-C (для iOS) або Java (для Android). Таким чином, JS-код буде відображатися за допомогою реальних компонентів мобільного інтерфейсу, а не html атрибутів, і буде виглядати та відчуватися, як будь-яка інша мобільна програма. React

Native також відкриває інтерфейси JavaScript для API платформи, тому React Native додатки можуть отримати доступ до функцій платформи, таких як камера телефону, або місце розташування користувача, мікрофон та інше.

React Native в даний час підтримує як iOS, так і Android, і має також потенціал для розширення майбутніх платформ. Переважна більшість коду, яку ми пише, буде перехресною платформою.

Переваги React Native

Той факт, що React Native фактично візуалізується за допомогою стандартних API візуалізації своєї хост-платформи, дозволяє йому виділятися з більшості існуючих методів розробки кросплатформенних додатків (рис. 1.4), таких як Cordova або Ionic. Існуючі методи написання мобільних додатків з використанням комбінацій JavaScript, HTML та CSS зазвичай відображаються за допомогою веб-переглядів[10]. Хоча цей підхід може працювати, він також має недоліки, особливо щодо продуктивності. Крім того, вони зазвичай не мають доступу до набору власних елементів інтерфейсу платформи хосту. Коли ці фреймворки дійсно намагаються імітувати рідні елементи інтерфейсу, результати зазвичай «відчуваються» лише трохи; зворотна інженерія всіх дрібних деталей таких речей, як анімація, вимагає величезних зусиль, і вони можуть швидко застаріти.

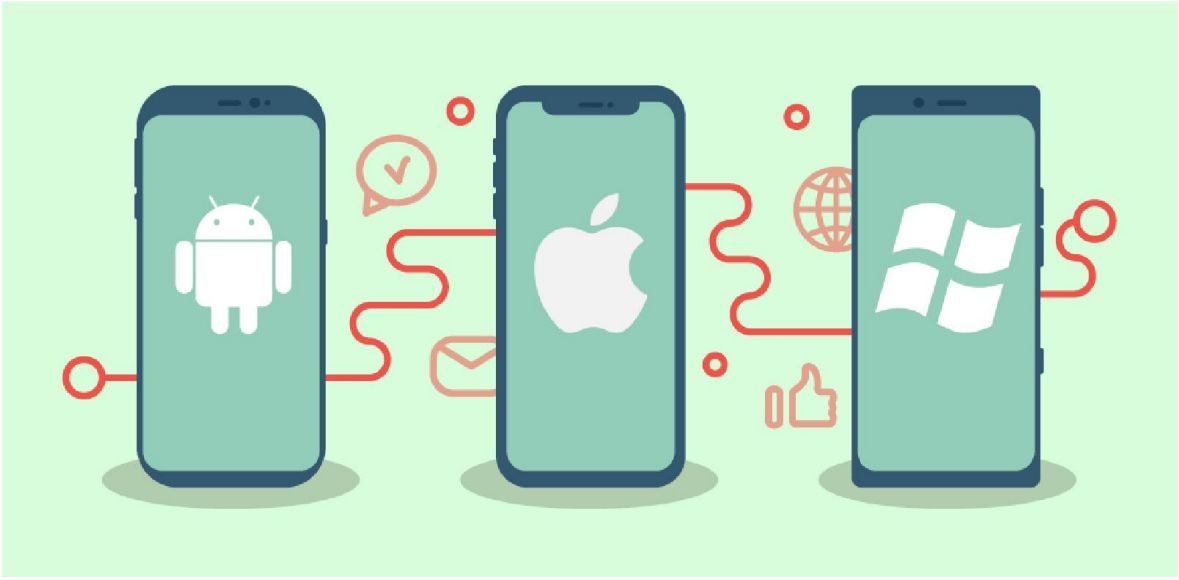

Рисунок 1.4 – Кросплатформенний Native-додаток

На відміну від цього, React Native фактично переводить вашу розмітку на реальні, рідні елементи інтерфейсу користувача, використовуючи існуючі засоби візуалізації переглядів на будь-якій платформі, з якою ви працюєте. Крім того, React працює окремо від основного потоку інтерфейсу користувача, тому ваша програма може підтримувати високу продуктивність без шкоди для можливостей. Цикл оновлення в React Native такий самий, як і в React: при зміні пропсів або станів життєвого циклу повторно відтворює рендер. Основна відмінність React Native від React у браузері полягає в тому, що перший робить це, використовуючи бібліотеки інтерфейсу своєї хост-платформи, а не використовуючи розмітку HTML та CSS.

Для розробників, які звикли працювати в Інтернеті з React, це означає, що вони можуть писати мобільні програми з продуктивністю та зовнішнім виглядом рідної програми, використовуючи знайомі інструменти. React Native також є покращенням у порівнянні з нормальною мобільною розробкою у двох інших областях: досвід розробників та потенціал розвитку між платформами.

Оскільки React Native – це «просто» JavaScript, вам не потрібно перебудовувати додаток, щоб побачити, як зміни відображаються; натомість ви можете натиснути Command+R, щоб оновити програму так само, як і будь-яку іншу веб-сторінку. Всі ці хвилини, витрачені на очікування створення вашої програми, дійсно можуть скластись, і, навпаки, швидкий ітераційний цикл React Native здається знахідкою.

Крім того, React Native дозволяє вам скористатися перевагами інтелектуальних засобів налагодження та повідомлення про помилки. Якщо ви знайомі з інструментами розробника Chrome або Safari (рис. 1.5), вам буде приємно дізнатися, що ви також можете використовувати їх для мобільної розробки. Аналогічно, ви можете використовувати будь-який текстовий редактор, який вам більше подобається для редагування JavaScript: React Native не змушує вас працювати в Xcode[11] для розробки для iOS або Android Studio для розробки Android.

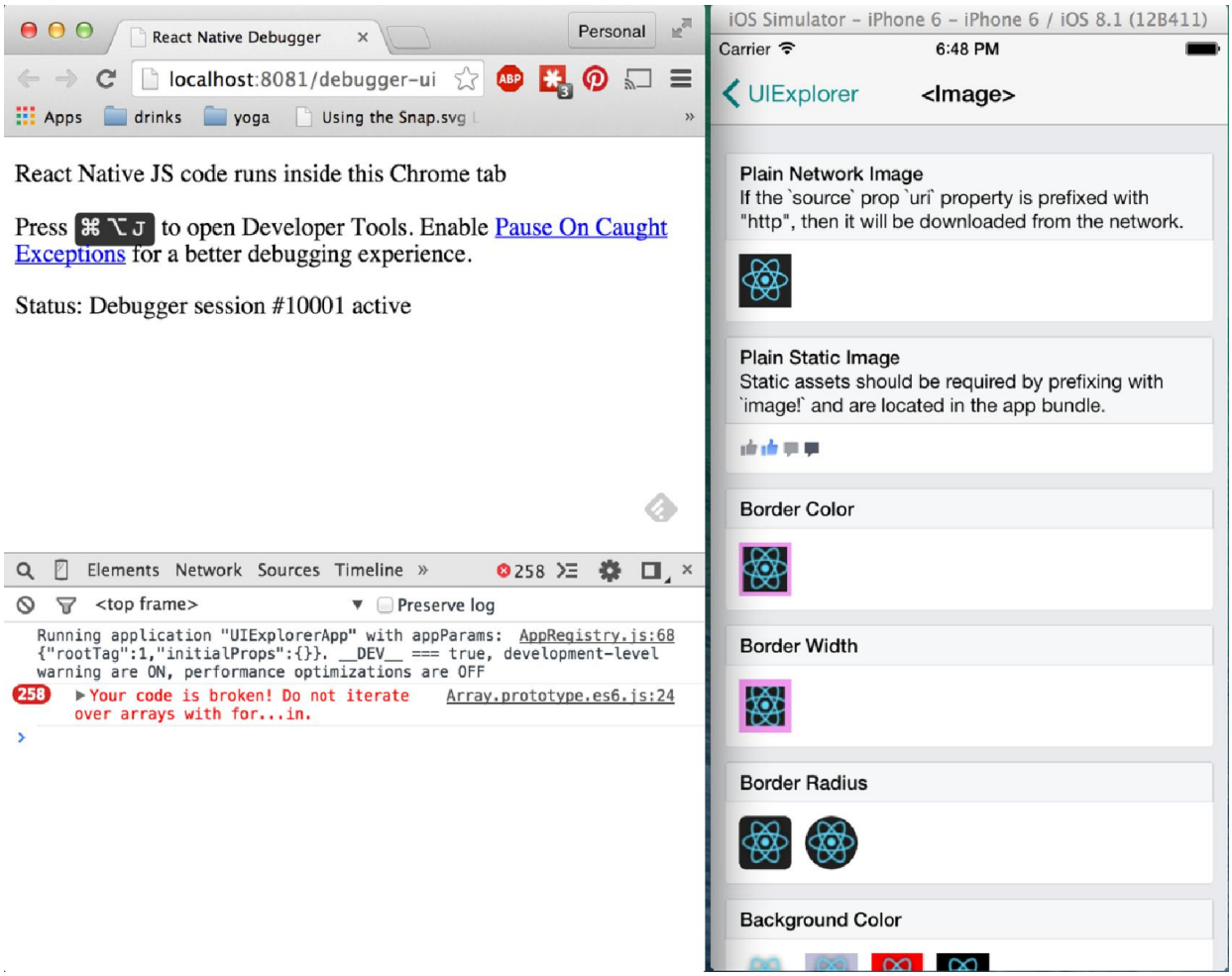

Рисунок 1.5 – Використання хром-дебагера з React-Native

Ризики та недоліки

Як і будь-що, використання React Native не позбавлене недоліків, і те, чи підходить React Native для вашої команди чи ні, дійсно залежить від індивідуальної ситуації.

Ймовірно, найбільшим ризиком є зрілість React Native, оскільки проект ще досить молодий. Підтримка iOS була випущена у березні 2015 року, а підтримка Android — у вересні 2015 року. Документація, безумовно, має місце для вдосконалення та продовжує розвиватися. Деякі функції на iOS та Android досі не підтримуються, а спільнота все ще знаходить найкращі практики. Доброю новиною є те, що в переважній більшості випадків ви можете самостійно реалізувати підтримку відсутніх API.

Оскільки React Native вводить у ваш проект ще один рівень, він також може зробити налагодження ще більш волосистим, особливо на перетині React і хост-платформи. React Native ще молодий, і тут застосовуються звичайні застереження, пов'язані з роботою з новими технологіями. Проте, загалом, я думаю, ви побачите, що користь перевищує ризики.

Підсумок

React Native — це захоплюючий фреймворк, який дозволяє веб-розробникам створювати надійні мобільні програми, використовуючи наявні знання JavaScript. Він пропонує більш швидку розробку для мобільних пристроїв та більш ефективний обмін кодом у iOS, Android та Інтернеті, не жертвуючи досвідом або якістю програм кінцевого користувача. Компроміс полягає в тому, що він новий і все ще триває. Якщо ваша команда може впоратися з невизначеністю, пов'язаною з роботою з новою технологією, і хоче розробити мобільні додатки для більш ніж однієї платформи, вам слід подивитися на React Native.

1.5 Огляд популярних проектів на React-Native

Зважаючи на всі плюси та мінуси, не дивно, що багато компаній вибирають React Native для розробки своїх мобільних додатків. Ось перелік прикладів тих, хто використав цю структуру у виробничому середовищі:

- Facebook;
- Facebookads;
- Walmart;
- Bloomberg;
- Instagram;
- SoundCloud Pulse;
- Townske;
- Gyroscope;

- Delivery.com;
- Wix.

Чому ці компанії вибрали React Native і які в них результати.

1. Facebook і React Native

React Native розпочався як проект хакатону Facebook, розроблений у відповідь на потреби компанії. Facebook хотів донести до мобільних пристроїв усі переваги веб-розробки, такі як:

- швидкі ітерації;
- наявність єдиної команди для розробки всього продукту.

Так React Native був реалізований і використано для розробки мобільних додатків як для iOS, так і для Android. Спочатку Facebook розробляв React Native лише для підтримки iOS. Однак, завдяки останній підтримці операційної системи Android, бібліотека тепер може надавати мобільні інтерфейси для обох платформ. Facebook використав React Native для розробки власного додатка Ads Manager, створивши версію для iOS та Android. Обидві версії були створені однією командою розробників.

Те, що побачив Facebook після написання додатку за допомогою фреймворку – це значне покращення продуктивності. Тепер запуск інформаційної панелі подій відбувається вдвічі швидше.

Більшість досягнень зроблено на рівні фреймворку, що визначає, що ваш додаток React Native автоматично виграє при переході на останню версію програми React Native[12].

2. Facebook Ads і React Native

Ads Manager – це інструмент Facebook, який дає змогу створювати рекламу у Facebook і керувати нею. Тут ви можете переглядати всі свої кампанії, набори реклами й окремі оголошення у Facebook, вносити до них зміни й дивитися результати[13].

За допомогою Ads Manager можна виконувати такі дії:

- створювати рекламні кампанії. В Ads Manager можна використовувати покроковий процес створення реклами для розробки рекламних оголошень. Під час створення реклами вам потрібно вибрати маркетингову ціль, людей, яких ви хочете охопити, місця показу реклами та формат реклами. Докладніше про створення реклами;
- керувати кількома оголошеннями одночасно. В Ads Manager можна редагувати налаштування, як-от аудиторію, бюджет і плейсменти, для багатьох оголошень і створювати копії оголошень, дублюючи їх. Докладніше про керування рекламою в Ads Manager;
- переглядати дані про ефективність реклами. Дивіться актуальні дані щодо ефективності своєї реклами та плануйте звіти. Ви можете переглядати результати як на рівні оголошень, так і на вищому рівні, щоб бачити дані про ефективність своїх кампаній. Також можна застосовувати розподіл, щоб переглядати метрики, які вас цікавлять, і створювати або планувати звіти про рекламу.

Платформа соціальних мереж – не єдиний додаток React Native, який випускався під дахом Facebook. Facebook Ads був першим додатком React Native для Android і першим кросплатформенним додатком на основі React Native, створеним у компанії.

Чому React Native?

Фреймворк здавався придатним для багатьох складних бізнес-логік, необхідних для точного вирішення відмінностей у форматах оголошень, часових поясах, валютах, форматах дат, валютних конвенціях тощо, зокрема, що значна частина його вже була складена в JavaScript.

З точки зору дизайну, інтерфейс чистий з інтуїтивно зрозумілим UI та простою навігацією. Анімації та переходи ідеальні; вони ні в якому разі не відчувають себе неприродними або глючними.

Загальний досвід чудовий, і якщо ваша маркетингова група не використовує додаток, Facebook настійно рекомендує їм почати.

Результати створення мобільного додатку React Native

Перше, що ви помітите, це те, що програма працює блискавично, незалежно від операцій, які ви хочете виконати; від перевірки статусу поточної кампанії до створення нової - для переходу на наступний рівень або легкого доступу до даних потрібна лише секунда або дві.

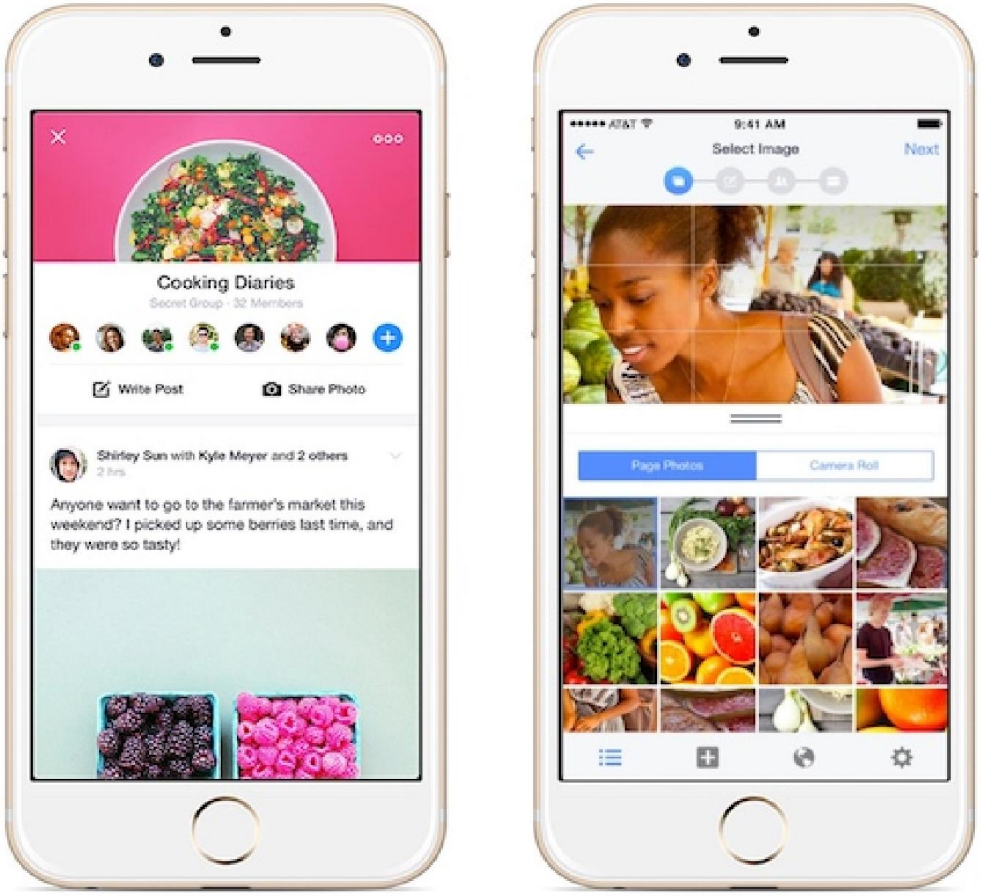

Рисунок 1.6 – Facebook Ads Manager

3. Walmart і React Native

Walmart вже довів свою новаторську позицію, представивши Node.js у свій стек. Через кілька років вони також переписали свій мобільний додаток на React Native. Одним із напрямків роботи Walmart у сфері мобільних застосунків є створення інструментів самообслуговування, які надають покупцям доступ до інформації, яку вони шукають, приймаючи рішення про покупку. Легкий доступ до такої інформації особливо важливий для великих магазинів, таких як Walmart, де покупці мають великий вибір[14].

Чому React Native?

Walmart має дуже високі цілі, прагнучи стати найбільшим у світі інтернет-магазином. Маючи такі великі цілі, компанії необхідно було зробити сміливі кроки, які передбачали більший ризик, щоб отримати конкурентну перевагу. Ось чому вони завжди шукають шляхи покращення досвіду клієнтів, випробовуючи нові технології.

React Native забезпечує чудову продуктивність, майже ідентичну оригінальним програмам, і надзвичайно плавну анімацію.

Результати створення програми React Native

Walmart вдалося покращити продуктивність програми як на iOS, так і на Android, використовуючи менші ресурси та за коротший проміжок часу.

96% кодової бази були розподілені між платформами, тоді як навички та досвід розробників використовувалися у всій організації.

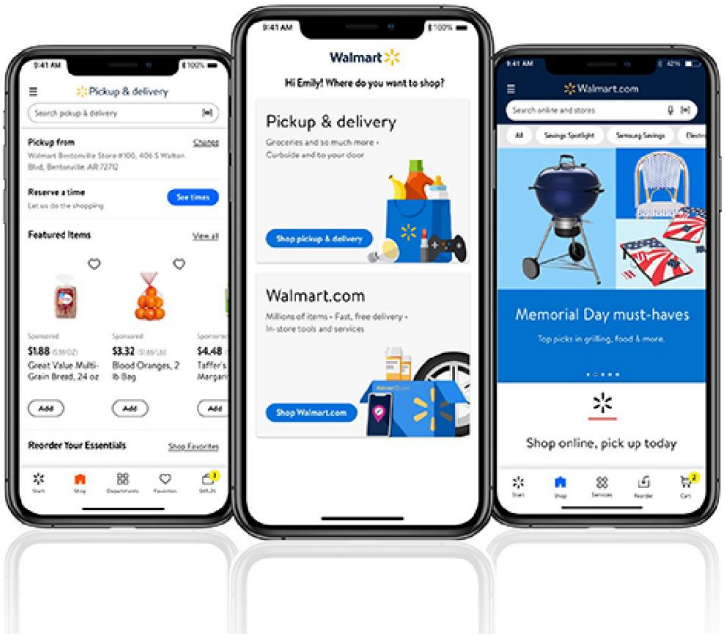

Рисунок 1.7 – Walmart додаток

4. Bloomberg & React Native

Новий споживчий мобільний додаток Bloomberg для iOS та Android надає клієнтам спрощений, інтерактивний досвід із простим у доступі

персоналізованим контентом, відео та прямими ефірами, розміщеними в засобах масової інформації Bloomberg.

Інженерна команда штаб-квартири Bloomberg у Нью-Йорці створила додаток за допомогою технології додатків React Native, основного інструменту, який насправді забезпечує кросплатформенність власних додатків[15].

Чому React Native?

«Мобільний додаток для споживачів вартував нам величезних зусилль, тому що нам довелося перевести всю нашу організацію на React Native» – каже Габріель Лью, старший інженер Bloomberg, який очолював роботу команди розробників.

Ще однією перевагою React Native є те, що вона автоматизує оновлення коду, прискорюючи випуск нових функцій продукту. Замість перекомпіляції ваш додаток миттєво перезавантажується.

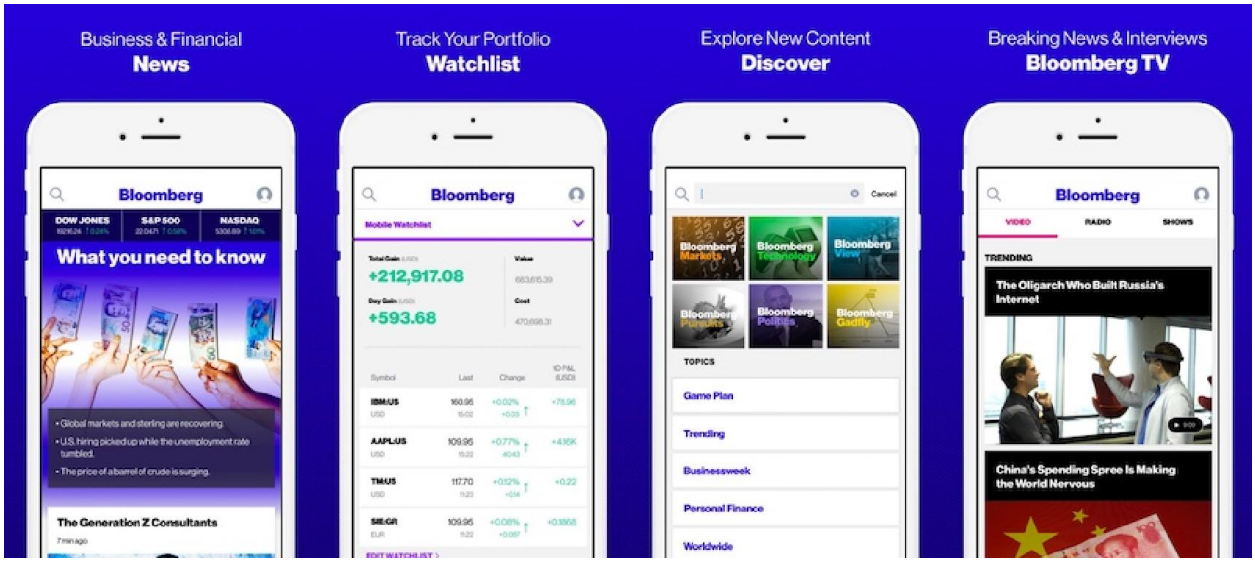

Рисунок 1.8 – Bloomberg додаток

5. Instagram і React Native

Instagram — це безкоштовний сервіс соціальних мереж, створений для обміну фотографіями та відео. Він був запущений у жовтні 2010 року на iPhone, а став доступний на Android у квітні 2012 р. Facebook придбав послугу у квітні 2012 р. І відтоді володіє нею.

Як і більшість програм у соціальних мережах, Instagram дозволяє стежити за користувачами, які вас цікавлять. Це створює канал на вашій домашній сторінці, де відображаються останні публікації від усіх, кого ви стежите. Ви можете ставити лайки, коментувати їх та ділитися ними з іншими людьми.

Окрім публікації звичайних фотографій та відео, які залишаються на вашій сторінці постійно, Instagram також підтримує історії. Якщо ви користувалися Snapchat або більшістю інших платформ соціальних медіа, вам це буде відомо. Історії дозволяють публікувати багато фотографій та відеокліпів у серії. Будь-хто може переглядати їх протягом 24 годин, після чого вони закінчуються.

Instagram прийняв виклик інтеграції React Native у існуючий рідний додаток. Вони почали з найпростішого вигляду, який ви можете собі уявити, - вивід Push Notification, які в основному було реалізовано як WebView. Це не вимагало створення навігаційної інфраструктури, оскільки інтерфейс був досить простим[16].

Результати створення мобільного додатку React Native

Команда розробників Instagram зіткнулася з деякими проблемами, але React Native значно покращив швидкість розробки. Від 85% до 99% коду ділиться між додатками Android та iOS, залежно від продуктів. Таким чином, команда змогла поставити додаток набагато швидше, ніж це було б із власним рішенням.

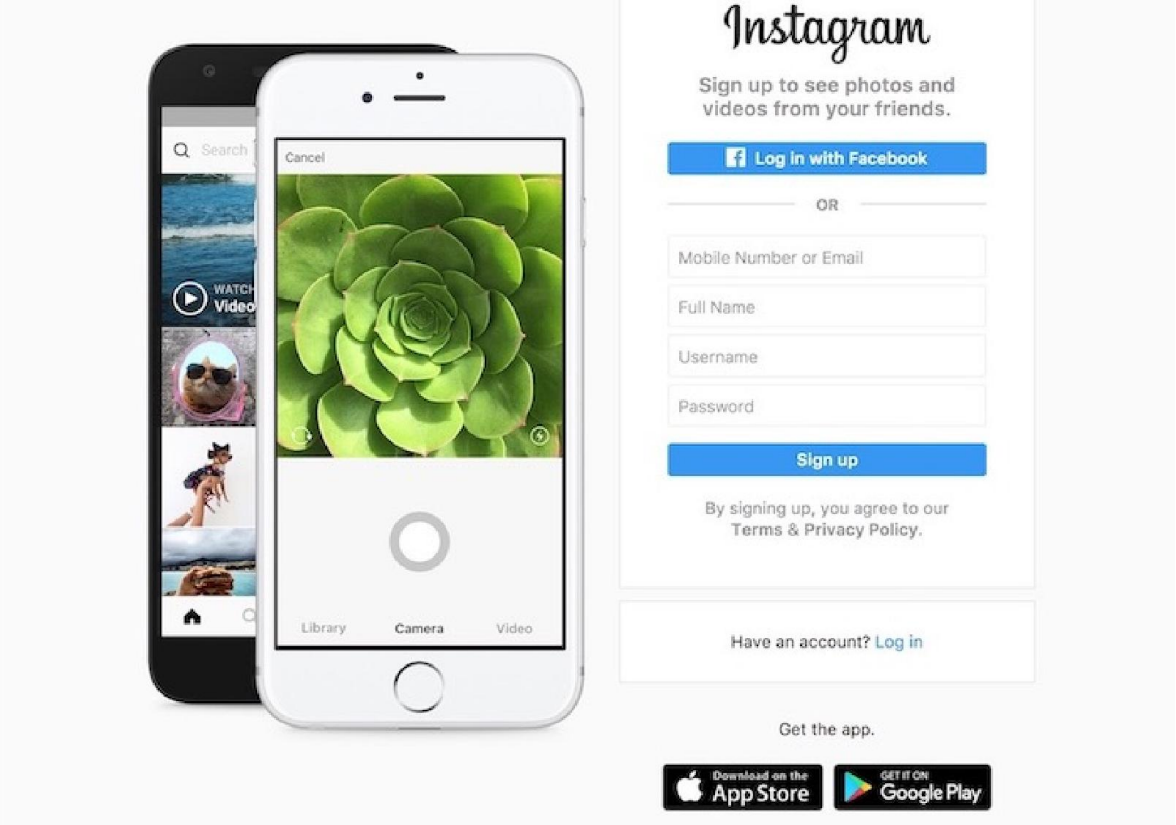

Рисунок 1.9 – Instagram додаток

6. SoundCloud Pulse & React Native

SoundCloud є одним з топ-5 музичних додатків як для iOS, так і для Android, сьогодні випустив новий мобільний додаток, призначений спеціально для художників та творців. Додаток під назвою SoundCloud Pulse дозволяє своїм користувачам публічно та приватно ділитися своїми звуками, відстежувати статистику та продуктивність, відповідати на коментарі та стежити за іншими користувачами. Компанія заявляє, що вже працює над розширеним набором функцій, який буде запущений разом із випуском iOS у майбутньому.

SoundCloud Pulse — це перший випадок, коли компанія запустила мобільний інструмент, спрямований на творців, які підживлюють його спільноту[17].

Чому React Native?

Коли компанія почала розробку другого набору власних програм, вони зіткнулися з кількома перешкодами. Розробників iOS було неможливо знайти, і вони не хотіли мати величезний розрив між випусками iOS та Android. Тому незалежна дослідницька група почала проводити сесії тестування користувачів із прототипами програм React Native.

Результати створення програми React Native

Їх досвід роботи з рамками був загалом позитивним. Розробникам було легше працювати над додатком на основі React Native, ніж над нативним додатком. Більш того, вони змогли самостійно створити додаток без частого входу спеціалізованих мобільних розробників.

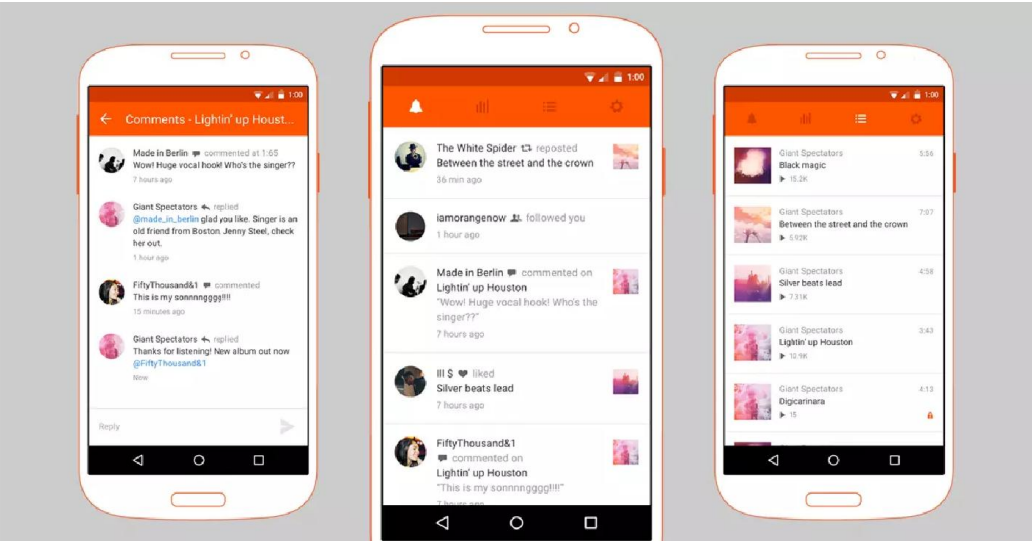

Рисунок 1.10 – SoundCloud Pulse додаток

7. Townske & React Native

Townske прагне стати вашим путівником міста під час наступної подорожі. Додаток пов'язує вас з місцевими жителями, щоб отримати список їхніх улюблених місць, а також створює кураторський список місць для вивчення вражень, якими діляться місцеві жителі. Користувачі не мають обов’язкового облікового запису, що чудово, оскільки це дозволяє швидко знайти наступне місце, яке ви хочете відвідати. Уявіть, що у вас низький рівень підключення до Wi-Fi або у вас розряджений акумулятор — у цих випадках це

чудовий додаток, адже він може працювати в оффлайн режимі та зможе допомогти вам з вибором місця, яке ви можете відвідати[18].

Результати створення мобільного додатку React Native

React Native зосереджений виключно на створенні мобільного інтерфейсу. Порівняно з фреймворками JavaScript, такими як AngularJS або MeteorJS, React Native орієнтований на користувацький інтерфейс, що робить його більше схожим на бібліотеку JavaScript, ніж на фреймворк.

Отриманий користувальницький інтерфейс дуже чуйний і веде себе плавно завдяки асинхронній взаємодії JavaScript з рідним середовищем. Це означає, що додаток швидше і має кращий час завантаження, ніж типовий гібридний додаток, і відчуває себе більш плавно.

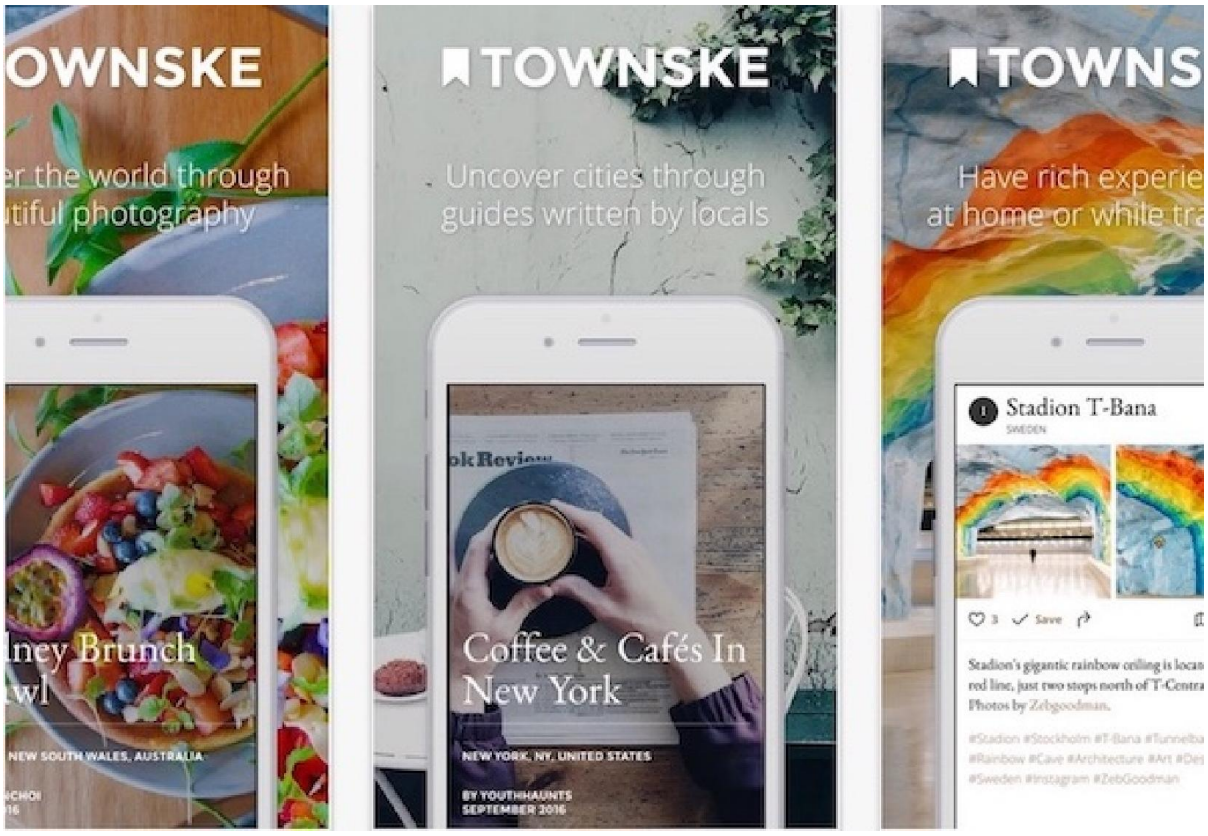

Рисунок 1.11 – Townske додаток

8. Gyroscope і React Native

Gyroscope дозволяє побачити повну історію вашого життя. Це додаток про здоров'я на стероїдах. Ви не тільки можете відстежувати кроки, тренування

або частоту серцевих скорочень, але й за допомогою десятків інтеграцій ви також можете відстежувати такі дії, як продуктивність на комп'ютері, або використовувати трекер сну та автоматичний штучний інтилект, щоб переконатися, що ви висипаєтесь[19].

Результати створення мобільного додатку React Native

Завдяки React Native дані відображаються у двох привабливих добре продуманих представленнях: простий та картковий. Усі відстежувані дані об'єднуються у щоденних/тижневих/місячних звітах, і ви також можете легко глибоко зануритися в них і вибрати, на чому далі хочете зосередитися.

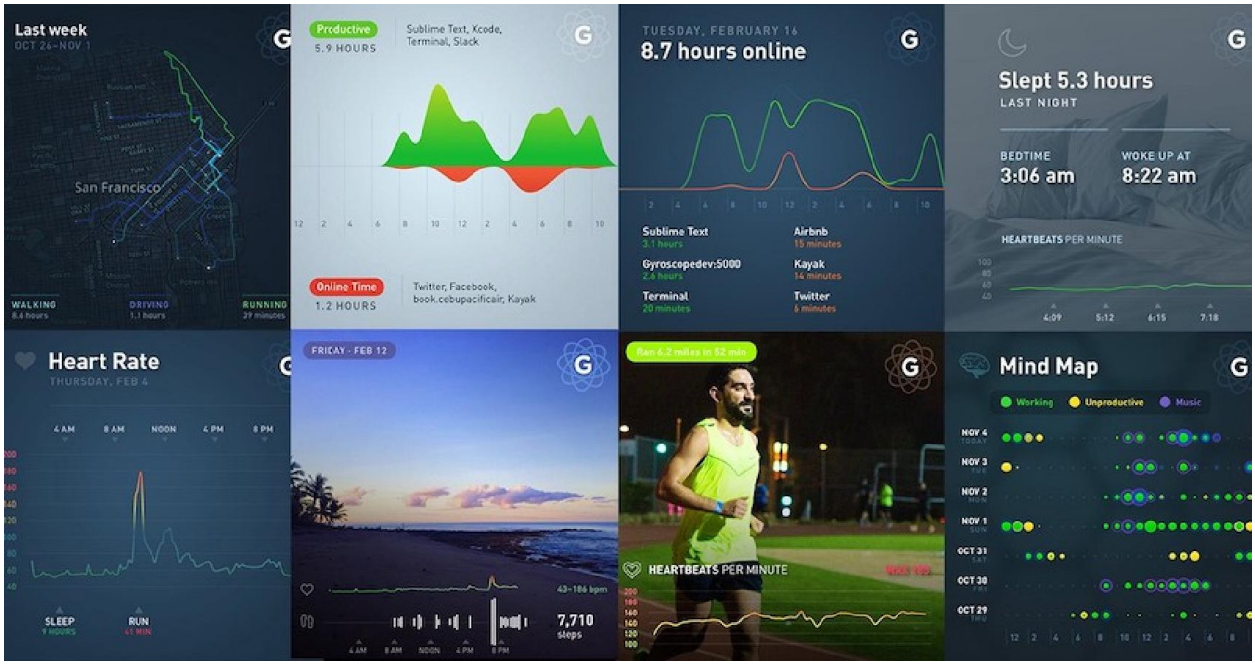

Рисунок 1.12 – Gyroscope додаток

9. Wix і React Native

Створена в 2006 році, Wix — це онлайн-компанія, яка надає послуги веб-хостингу та дизайну веб-сайтів. Користувачі можуть створювати та створювати свої сайти в HTML5/CSS, а також мобільні веб-сайти за допомогою утиліт перетягування та редагування.

Дві популярні особливості Wix полягають у тому, що розробники можуть створювати власні веб-програми, щоб продавати їх іншим користувачам, і що користувачам не потрібно знати кодування для створення веб-сайту[20].

Результати створення програми React Native

Хоча розробка власних додатків зазвичай пов'язана з неефективністю, нижчою продуктивністю та більш тривалим часом на розгортання, React Native забезпечила Wix швидкість та спритність розробки веб -додатків у гібридному просторі, і все це з власними результатами.

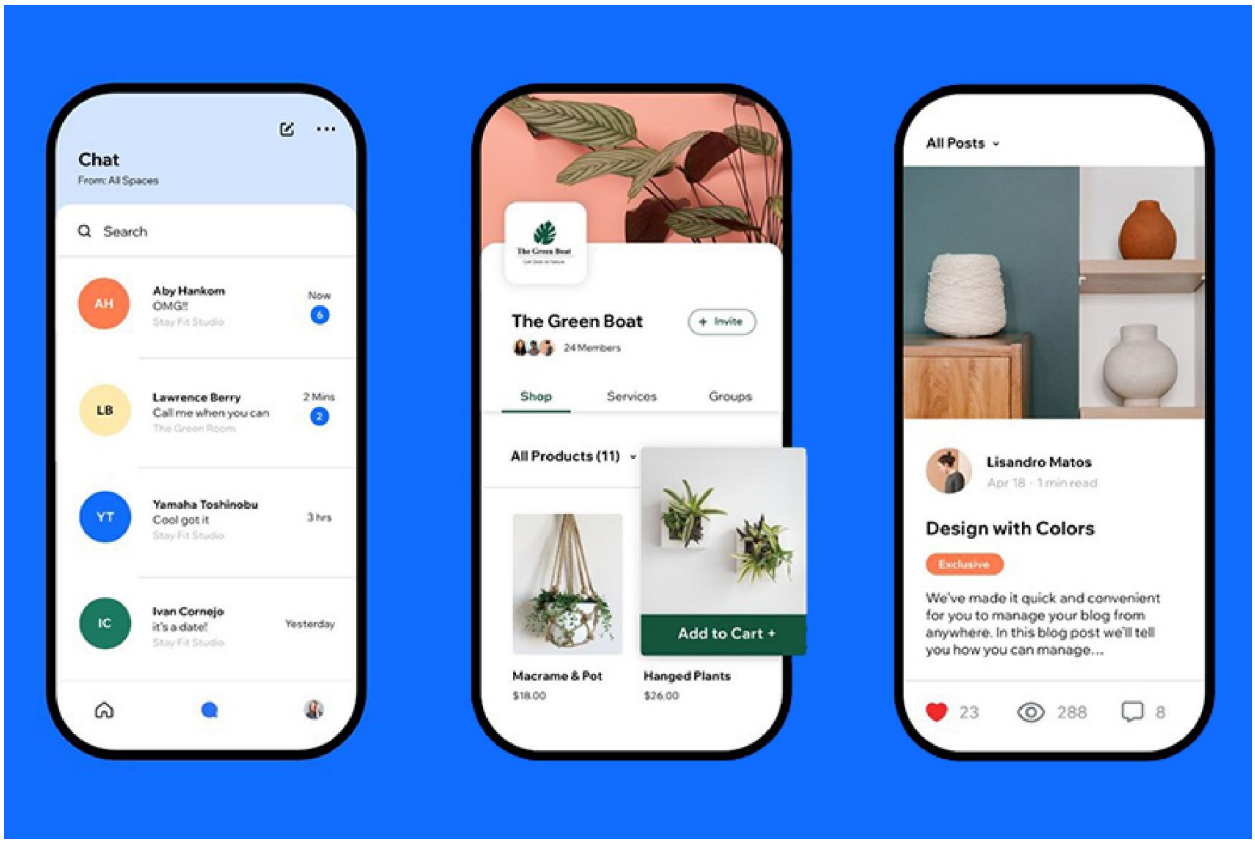

Рисунок 1.13 – Wix mobile app

10. Delivery.com і React Native

Delivery.com розширює можливості економіки району, дозволяючи клієнтам замовляти в Інтернеті свої улюблені блюда з ресторанів, товари з продуктових магазинів, магазинів вина та міцних алкогольних напоїв, а також у пральнях та хімчистках.

Щодня більше мільйона клієнтів delivery.com досліджують свої території та замовляють у більш ніж 10 000 регіональних компаній, перебуваючи вдома, на роботі чи в дорозі. Зі штаб-квартирою в Нью-Йорку та розширенням присутності по всій території США, delivery.com пропонує електронну комерцію невід'ємну частину місцевого повсякденного життя, допомагаючи клієнтам робити покупки, рости та розвиватись квартали[21].

Результати створення мобільного додатку React Native

React Native дозволив компанії підключити карту до функцій пристрою, таких як обертання, масштабування та компас, одночасно використовуючи менше пам'яті та пришвидшуючи завантаження. Якщо додаток підтримує старі операційні системи (і старі пристрої), це може допомогти забезпечити безперебійну роботу програми.

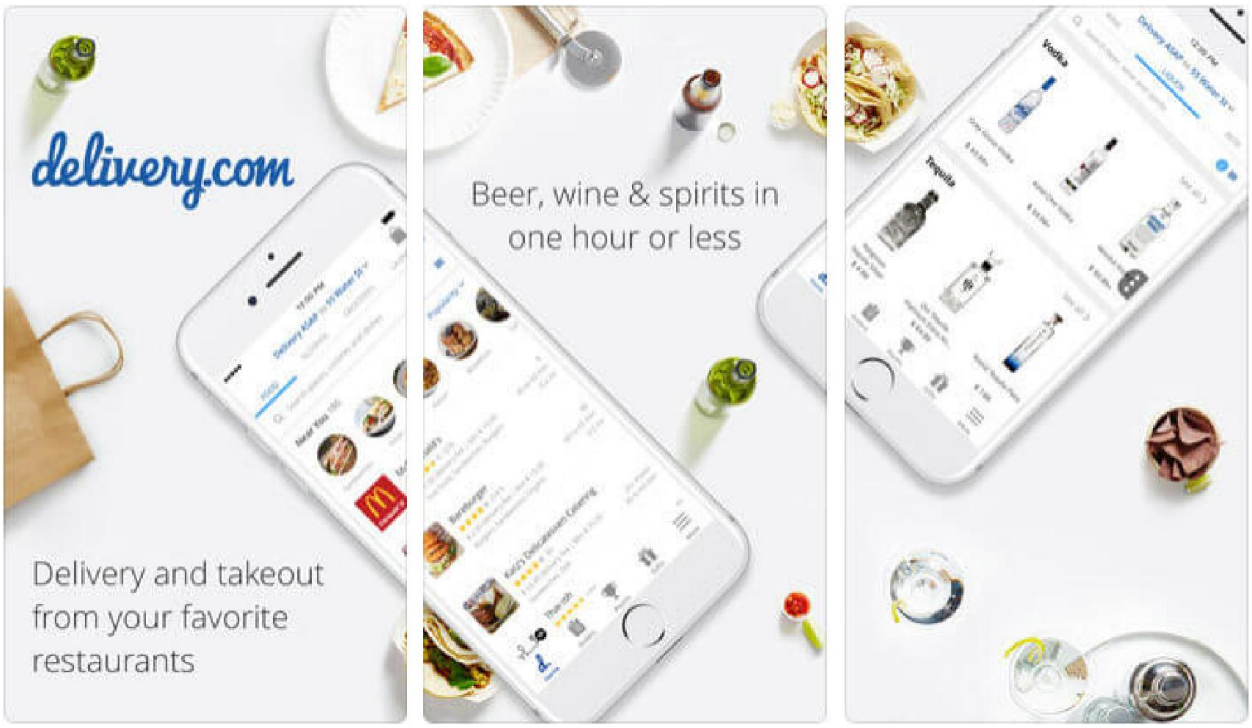

Рисунок 1.14 – Delivery.com mobile app

1.6 Постановка задачі

Завданням кваліфікаційної роботи є розробка структури застосунку, реалізація цієї структури на практиці за допомогою React Native, а також тестування і його введення у експлуатацію.

Застосунок забезпечує такі функції:

1. Надає користувачам всю необхідну інформацію про вузли(опис, складність, АВОК, застереження, можливі назви).
2. Додавання вузлів до обраного та їх видалення звідти.
3. Користувачі можуть виконувати пошук серед списку вузлів.
4. Є можливість зміни мови додатку.
5. Користувачі можуть переглядати анімацію зав’язування вузла.
6. Користувачі можуть змінювати швидкість програвання анімації.
7. Можна керувати анімацією зав’язування вузла, а саме програвати її вперед або назад.
8. Можна переключити анімацію зав’язування вузла на анімацію перегляду вузла в 360°.
9. Можна керувати анімацією відображення вузла в 360°, а саме програвати її вперед або назад.
10. Користувачі можуть віддзеркалювати вузли стосовно осі ординат.
11. Користувачі можуть поширювати застосунок будь-яким зручним для них способом.

РОЗДІЛ 2
ТЕОРЕТИЧНІ ОСНОВИ РОЗРОБЛЕННЯ МУЛЬТИПЛАТФОРМЕННОГО МОБІЛЬНОГО ЗАСТОСУНКУ СПРЯМОВАНОГО НАВЧИТИ В'ЯЗАТИ ВУЗЛИ

2.1 Моделювання застосунку за допомогою uml-діаграм

UML, скорочення від Unified Modeling Language[22], є стандартизованою мовою моделювання, що складається з інтегрованого набору діаграм, розроблених, щоб допомогти розробникам систем та програмного забезпечення з визначенням, візуалізацією, побудовою та документуванням артефактів програмних систем, а також для бізнес-моделювання інших непрограмних системи. UML є набором передових інженерних практик, які довели свою ефективність при моделюванні великих і складних систем. UML – дуже важлива частина розробки об'єктно-орієнтованого програмного забезпечуюча стабільність процесу розробки програмного забезпечення. UML використовує переважно графічні позначення для вираження дизайну програмних проектів. Використання UML допомагає проектним групам спілкуватися, досліджувати потенційні проекти та перевіряти архітектурний дизайн програмного забезпечення.

Розробниками мови UML було також створено і RUP (Rational Unified Process) – раціональний уніфікований процес. Причому, при застосуванні мови UML не висувається вимога одночасного використання RUP, оскільки вони є абсолютно незалежними. Процес RUP може використовуватись для розробки проекту в залежності від типу останнього та вимог замовника [23].

Діаграми варіантів використання описують взаємини і залежності між групами варіантів використання і дійових осіб, які беруть участь в процесі.

Важливо розуміти, що діаграми варіантів використання не призначені для відображення проекту і не можуть описувати внутрішній устрій системи. Діаграми варіантів використання призначені для спрощення взаємодії з

майбутніми користувачами системи, з клієнтами, і особливо знадобляться для визначення необхідних характеристик системи. Іншими словами, діаграми варіантів використання говорять про те, що система повинна робити, не вказуючи самі застосовувані методи.

На рисунку 2.1 зображено діаграму варіантів використання для майбутнього додатку [24].

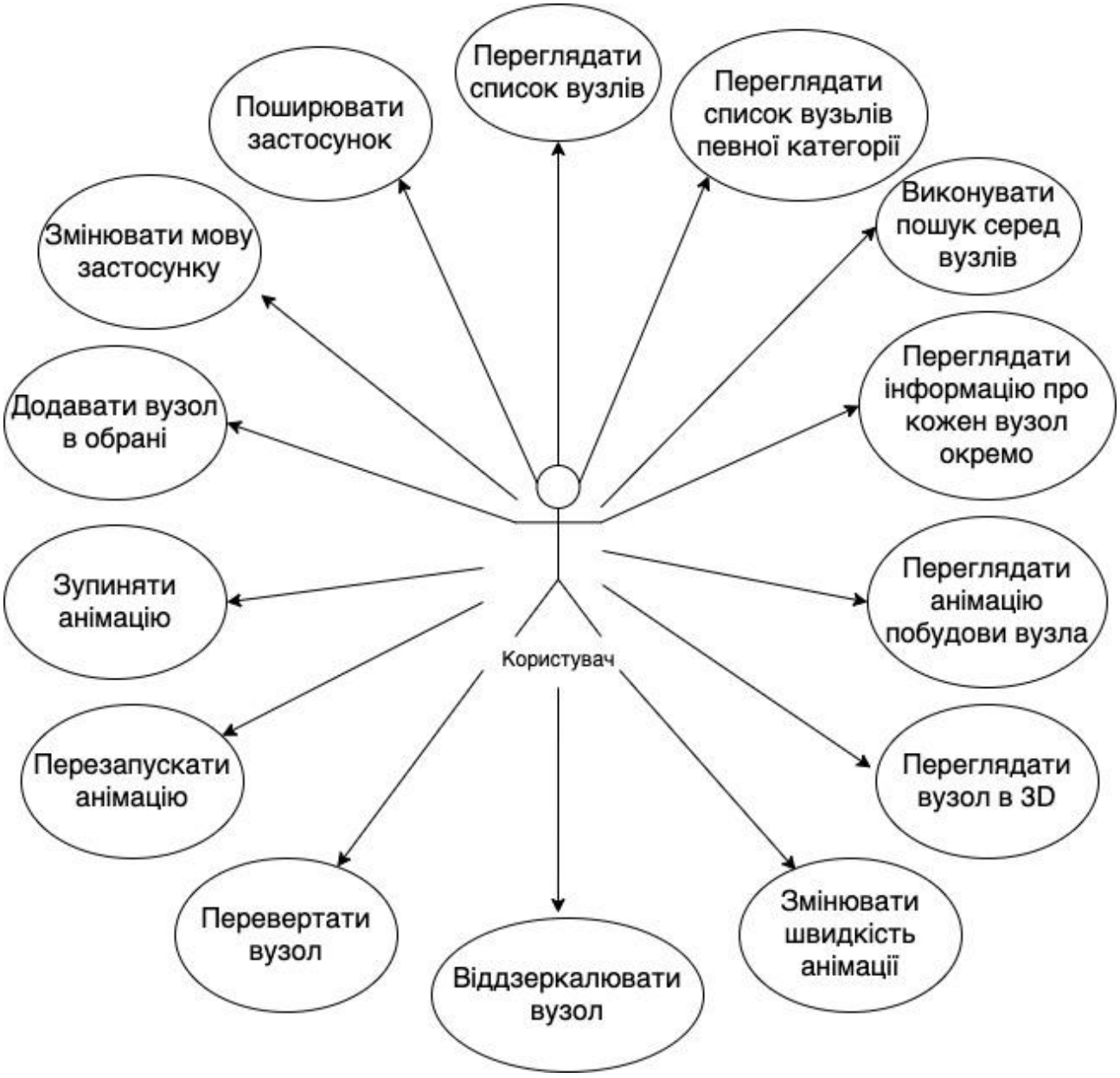

Рисунок 2.1 – Діаграма варіантів використання мобільного застосунку

В розробленому застосунку передбачений один тип користувачів– звичайний користувач. Він має змогу переглядати список категорій вузлів, переглядати список вузлів певної категорії, виконувати пошук серед вузлів, переглядати інформацію про кожен вузол окремо, переглядати анімацію побудови вузла, переглядати вузол в 3D, змінювати швидкість анімації, віддзеркалювати вузол, перевертати вузол, перезапускати анімацію, зупиняти анімацію, додавати вузол в обрані, змінювати мову застосунку, поширювати застосунок.

Діаграма станів є графом спеціального виду, який представляє певний автомат. Вершинами графа є можливі стани автомата, зображувані відповідними графічними символами, а дуги позначають його переходи зі стану в стан. Діаграми станів можуть бути вкладені одна в одну для більш детального представлення окремих елементів моделі.

Діаграма станів використовується для представлення стану системи або частини системи у момент часу. Це діаграма поведінки, яка представляє поведінку із застосуванням кінцевих переходів стану. Діаграми станів також називаються state-machine або statechart діаграма. Ці терміни часто використовуються як синоніми. Отже, діаграма станів використовується для моделювання динамічної поведінки класу у відповідь на час та зміна зовнішніх стимулів. Можна сказати, що кожен клас має стан, але не кожен клас моделюється за допомогою діаграм станів [25].

Використання діаграми станів:

- для позначення подій, відповідальних за зміну стану;
- для моделювання динамічної поведінки системи;
- щоб зрозуміти реакцію предметів/класів на внутрішні чи зовнішні подразники.

На рисунку 2.2 зображено діаграму станів для звичайного користувача застосунку Knots3D.

Рисунок 2.2 – Діаграма станів для користувача мобільного застосунку

Діаграма дій – ще одна важлива діаграма поведінки в UML діаграмі, що описує динамічні аспекти системи. Діаграма дій – це, насправді, розширена версія блок-схеми, яка моделює перехід від однієї дії до іншого.

Діаграми дій описують, як дії координуються для надання послуги, яка може знаходитись на різних рівнях абстракції. Як правило, подія повинна бути досягнута деякими операціями, особливо коли операція призначена для досягнення низки різних речей, що потребують координації, або як події в одному варіанті використання пов'язані один з одним, зокрема, варіанти використання, в яких дії можуть перекриватися та вимагати узгодження. Вона

також підходить для моделювання того, як набір варіантів використання координується для представлення бізнес-процесів [26].

Рисунок 2.3 – Діаграма діяльності для користувачів мобільного додатку

Дана діаграма зображує в загальному вигляді основний User Flow передбачений для юзера. Відкривши додаток користувач бачить список всіх категорій вузлів, може змінити мову, або одразу переглянути окрему категорію, і після вибрати один вузол і побачити інформацію про нього.

2.2Структура застосунку

Кожен додаток зазвичай має головну сторінку, яка відкривається першою. Верхня частина містить хедер, що є незмінним для будь-якої сторінки, він в додатках слугує також в якості навігаційної панелі.

У процесі створення будь-якого мобільного або веб-програми ви повинні переконатися, що кожен компонент добре побудований. Навіть найменші проблеми, які можуть виникнути в процесі створення архітектури мобільного додатка, можуть підірвати якість кінцевого результату. Як кажуть, хочеш щось робити – роби добре. Ось чому кожна популярна програма для Android і iOS має високонадійну архітектуру мобільного додатка і успішно перемагає своїх користувачів. Що таке архітектура мобільного додатку? Це набір структурних елементів та його інтерфейсів, у тому числі складається система, і навіть їх поведінка у межах всіх структурних елементів. Можна сказати, що це скелет програми, і вся робота мобільного додатку визначається її якістю. Упускаючи важливий елемент у створенні архітектури мобільної програми, ви ставите під загрозу успіх свого проекту. Складність побудови якісної архітектури залежить від обсягу застосування. Правильна архітектура дозволить заощадити багато часу, енергії та коштів у майбутньому.

Для проектування додатку мобільного кросплатформенного застосунку спрямованого навчити в’язати вузли Knots3D було обрано лінійну структуру, оскільки даний ресурс має невелику кількість сторінок і просте у розумінні та навігації меню. Цю структуру зображено на рисунку 2.8.

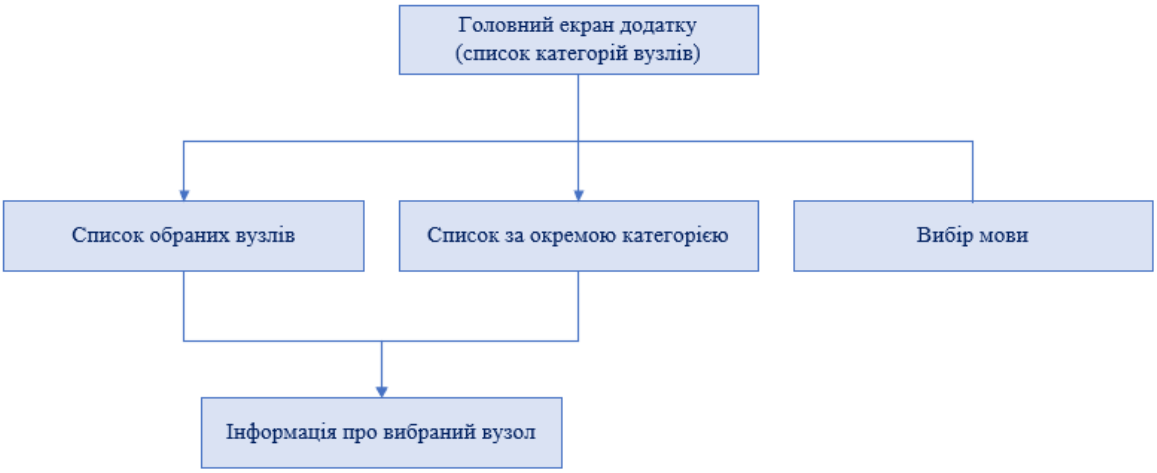

Рисунок 2.6 – Структура додатку Knots3D

2.3Макет сторінок

Макет був розроблений з огляду на навчальні додатки схожого спрямування.

Відповідно до розробленої структури, було спроектовано макети головної сторінки додатку (Рисунок 2.7).

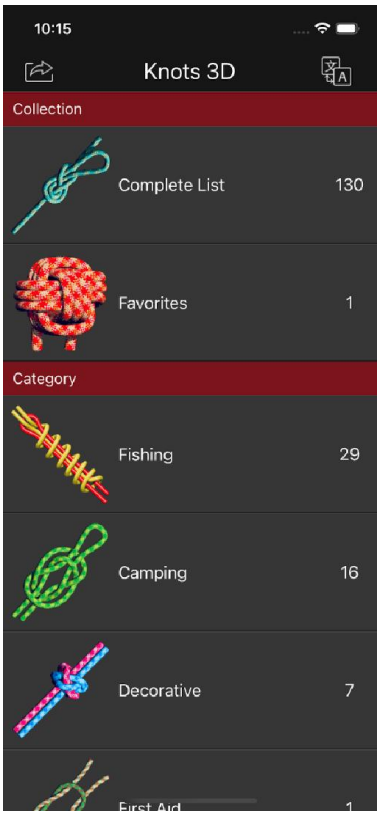

Рисунок 2.7 – Макет головної сторінки додатку

Дизайн і підбір колірної гами – те, що відіграє не останню роль у створенні застосунку. Вибір, зроблений вірно, важливий, адже палітра впливає на настрій людини і в якійсь мірі на його поведінку. Кольором однозначно можна привернути увагу або відштовхнути. Щоб не втратити потенційних клієнтів, власник ресурсу повинен враховувати думку професійного дизайнера. Але і самому корисно мати уявлення про підбір гармонійної колірної гами для додатків. Як підбирати відтінки, які сервіси допоможуть «підглянути» кольору для продукту і їх поєднання, як працюють колірні схеми. Що ще потрібно врахувати при виборі палітри для застосунку [27].

Правила підбору Головний інструмент, який застосовують для комбінування відтінків, – це колірний круг. З його допомогою можна знайти ефектні поєднання, які гармонійно виглядають разом. Але спочатку треба зрозуміти принципи роботи з ним.

Гармонійні комбінації народжуються з:

- 3 кольорів, які в колі утворюють трикутник. Класична тріада – коли кожен елемент трійки віддалений від іншого на рівну відстань. Один колір панує, два інших доповнюють його. Ще є аналогові тріади – поєднання відтінків, що лежать поруч один з одним на колірному колі;
- 4 кольорів, що утворюють прямокутник. Складна схема, при якій 1 колір – головний, 2 – додаткових, а ще один – підкреслює акценти. Всі 4 кольори не можна використовувати як основні: з’явиться дисбаланс, дизайн буде, що називається, різати око;
- контрастів, тобто поєднань 2 кольорів, розташованих в колі протилежно.

Значення відтінків Багатьох при виборі кольору цікавить його значення. Коротко, як сприймаються кольори в дизайні:

Чорний: сила, лаконічність, влада.

- білий: свіжість, ясність, чистота;
- бежевий: скромність, нейтралітет, ніжність;
- відтінок слонової кістки: стиль, елегантність, комфорт;

- сірий: строгість, професіоналізм, контроль;
- фіолетовий: еліта, розкіш, романтика;
- синій: спокій, стабільність, безпека;
- зелений: зростання, розвиток, природність;
- помаранчевий: молодість, дружба, бадьорість;
- жовтий: веселощі, ентузіазм, енергія;
- червоний: пристрасть, дія, активність.

Для простоти вибору кольорів можна використати коло кольорів, що зображене на рисунку 2.8.

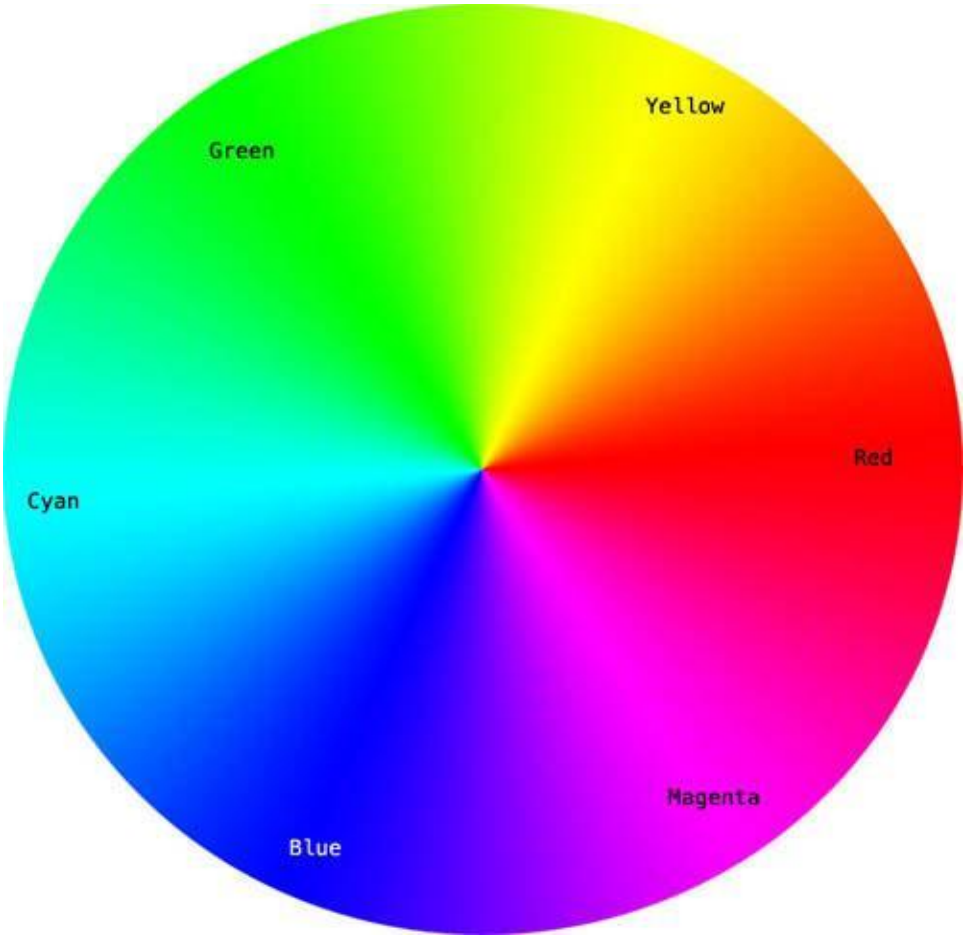

Рисунок 2.8 – Коло кольорів

Щоб знайти правильну кольорову схему, необхідно використовувати будь-які два кольори один навпроти одного, будь-які три кольори на рівній відстані при формуванні трикутника або будь-який з чотирьох кольорів, що утворюють прямокутник (дві пари кольору один навпроти одного). Схеми

залишаються правильними незалежно від кута повороту.

Для дизайну застосунку було обрано винний та сірий кольори, оскільки:

1. Зараз дуже популярною є темна тема, яка добре підходить до тематики застосунку.

2. Їх легко сприймає людське око і тому воно не втомлюється, дивлячись на них.

Також була розроблена іконка додатку у вигляді зав’язаного вузла, що зображено на рисунку 2.9.

Рисунок 2.9 – Іконка додатку «Knots3D»

РОЗДІЛ 3
ПРАКТИЧНА ЧАСТИНА РОЗРОБЛЕННЯ МУЛЬТИПЛАТФОРМЕННОГО МОБІЛЬНОГО ЗАСТОСУНКУ СПРЯМОВАНОГО НАВЧИТИ В'ЯЗАТИ ВУЗЛИ

3.1 Вибір та обґрунтування веб-технологій для розроблення програмного забезпечення

3.1.1. Вибір та обґрунтування використання SQLite у якості системи керування базами даних програмного забезпечення.SQLite – це внутрішньо процесорна бібліотека, яка реалізує автономний, безсерверний, транзакційний механізм бази даних SQL із нульовою конфігурацією. Код SQLite знаходиться у відкритому доступіі, таким чином, може використовуватися в будь-яких цілях, комерційних або приватних [28].

SQLite – це найбільш поширена база даних у світі з більшою кількістю додатків, ніж ми можемо порахувати, включаючи декілька гучних проектів.

SQLite – це вбудований механізм бази даних SQL. На відміну від більшості інших баз даних SQL, SQLite не має окремого серверного процесу. Вона читає та записує безпосередньо у звичайні файли на диску. Повна база даних SQL з кількома таблицями, індексами, тригерами та уявленнями міститься в одному файлі на диску. Формат файлу бази даних є кросплатформним – ви можете вільно копіювати базу даних між 32-бітними та 64-бітовими системами або між архітектурами з прямим порядком байтів та зворотним порядком байтів. Ці функції роблять SQLite популярним як формат файлів програми. Файли бази даних SQLite – це рекомендований формат зберігання Бібліотеки Конгресу США. Думайте про SQLite не як про заміну Oracle, а як про заміну fopen().

SQLite – компактна бібліотека. При включенні всіх функцій розмір бібліотеки може бути меншим за 750 КБ, залежно від цільової платформи та налаштувань оптимізації компілятора. (64-бітний код більше. І деякі

оптимізації компілятора, такі як агресивне вбудовування функцій та розгортання циклу, можуть призвести до того, що об'єктний код буде набагато більшим.) Існує компроміс між використанням пам'яті та швидкістю. SQLite зазвичай працює швидше, ніж більше пам'яті ви даєте. Тим не менш, продуктивність зазвичай непогана навіть у середовищах з низьким обсягом пам'яті. Залежно від того, як він використовується, SQLite може бути швидше, ніж пряме введення-виведення файлової системи[29].

База даних SQLite містить лише один файл на диску, що робить їх переносимими до будь-якої іншої бази даних. Деякі організації використовують більше однієї бази даних у своїх додатках, тому дійсно важливо, щоб база даних була переносимою. SQLite також можна використовувати для нативних та кросплатформових додатків. Отже, якщо ви розробляєте додаток на React Native або Java, ви можете використовувати SQLite.

Нижче наведено ще кілька причин, чому вам слід використовувати SQLite:

- SQLite використовує SQL, тому має всі функції стандартної бази даних SQL;
- деяким розробникам потрібні бази даних, які можуть масштабуватися та забезпечувати підтримку паралелізму. SQLite, що має широкі можливості, може бути пов'язаний з будь-яким виробничим додатком.
- часто розробникам буває важко проводити тестування, коли база даних програми безладна. SQLite дуже хороший для тестування;
- відсутність конфігурації: SQLite не вимагає складного налаштування для зберігання даних. Коли ви створюєте власні програми Java, вони інтегруються з платформою;
- розробники називають SQLite безсерверною базою даних, і це справді виправдовує очікування. Вам не потрібно налаштовувати будь-яку API або встановлювати якусь бібліотеку для доступу до даних із SQLite;
- SQLite є кросплатформним, що означає, що його можна використовувати в додатку Android, побудованому на Java, а також в

кросплатформному додатку, побудованому на React Native.

3.1.2. Вибір та обґрунтування використання мови JavaScript для розроблення програмного забезпечення.Сучасний JavaScript – це «безпечна» мова програмування загального призначення. Вона не надає низькорівневих засобів роботи з пам'яттю, процесором, так як спочатку була орієнтована на браузери, в яких це не потрібно.

Що ж стосується інших можливостей – вони залежать від оточення, в якому запущений JavaScript. У браузері JavaScript вміє робити все, що відноситься до маніпуляції зі сторінкою, взаємодії з відвідувачем і, в якійсь мірі, з сервером:

- створювати нові HTML-теги, видаляти існуючі, змінювати стилі елементів, ховати, показувати елементи і т.п.;
- реагувати на дії відвідувача, обробляти кліки миші, переміщення курсора, натискання на клавіатуру і т.п.;
- посилати запити на сервер і завантажувати дані без перезавантаження сторінки (ця технологія називається «AJAX»);
- отримувати і встановлювати cookie, запитувати дані, виводити повідомлення і т.д.

JavaScript – швидка і потужна мова, але браузер накладає на її виконання деякі обмеження.

Це зроблено для безпеки користувачів, щоб зловмисник не міг за допомогою JavaScript отримати особисті дані або якось нашкодити комп'ютеру користувача [30].

Цих обмежень немає там, де JavaScript використовується поза браузером, наприклад на сервері. Крім того, сучасні браузери надають свої механізми по установці плагінів і розширень, які володіють розширеними можливостями, але вимагають спеціальних дій по установці від користувача.

JavaScript не може читати / записувати довільні файли на жорсткий диск, копіювати їх або викликати програми. Він не має прямого доступу до

операційної системи.

JavaScript, що працює в одній вкладці, не може спілкуватися з іншими вкладками і вікнами, за винятком випадку, коли він сам відкрив це вікно або декілька вкладок з одного джерела (однаковий домен, порт, протокол).

3.1.3. Вибір та обґрунтування використання мови TypeScript для розроблення програмного забезпечення. TypeScript – мова програмування, представлена Microsoft восени 2012; позиціонується як засіб розробки веб-застосунків, що розширює можливості JavaScript [31].

Переваги TypeSctipt над JavaScript:

- статична типізація. Завдяки статичній типізації код на TypeScript більш передбачуваний і тому його легко дебажити;
- гарна підтримка ООП. Завдяки гарній підтримці ООП, модулів, інтерфейсів та просторів імен, проект навіть при великому обсязі буде впорядкованим;
- компілятор на стадії збирання знаходить багато помилок, ще до того, як вони потраплять в рантайм і зможуть щось зламати.

3.1.4. Вибір та обґрунтування використання фреймворку React Native для розроблення програмного забезпечення. React Native - це середовище програмування, розроблене Facebook, яке дає розробникам можливість створювати повноцінні нативні мобільні програми для iOS та Android з використанням універсальної мови програмування, що називається JavaScript. Відповідно до сайту Facebook React Native: «React Native використовує ті ж фундаментальні будівельні блоки, що й звичайні програми для iOS та Android. Ви просто поєднуєте ці будівельні блоки за допомогою JavaScript».

React Native покликаний допомогти розробникам повторно використовувати код в інтернеті та на мобільних пристроях. З React Native розробникам додатків не потрібно створювати один і той же додаток для iOS та Android з нуля. Натомість вони можуть повторно використовувати той самий

код у кожній операційній системі. У React Native чудово те, що між програмою, створеною на власному коді пристрою (Objective-C або Java), і програмою, створеною з використанням React Native, дуже мало відмінностей [32].

Фреймворк має відкритий вихідний код і керується спільнотою, і його підтримує велика кількість розробників, які готові поділитися своїми знаннями, досвідом та думками для покращення та підтримки фреймворку.

Переваги React Native для розробки мобільних додатків.React Native пропонує кілька переваг для розробки мобільних додатків.

1. Економія часу та грошей

Оскільки 95% або більше коду є кросплатформним, тобто він сумісний як з Android, так і з iOS, розробникам потрібно створити тільки одну програму, і в результаті створюються дві програми. Це заощаджує час на розробку програми, що дозволяє заощадити багато грошей, які були б вкладені у створення окремих додатків. З React Native підприємства можуть мати обидві програми одночасно за трохи більше половини вартості створення однієї версії. Компаніям більше не потрібно вибирати, яку версію створювати та запускати першою через витрати на створення двох окремих додатків. Крім того, обслуговування та оновлення виконуються в обох додатках одночасно, що дозволяє заощадити на майбутніх витратах після створення та запуску додатків.

2. Відмінна продуктивність

Програми React Native працюють майже так само, як нативні програми, створені на конкретній платформі iOS або Android. Вони також швидкі, тому що мова програмування оптимізована для мобільних пристроїв. Замість того, щоб переважно використовувати центральний процесор (ЦП), програми React Native використовують переваги графічного процесора (ГП). Це робить їх набагато швидше, ніж кросплатформні гібридні технології.

3. Підвищена гнучкість

Тип інтерфейсу, що використовується в React Native, дозволяє різним розробникам у команді легко перейти до місця, де хтось зупинився, і продовжити розробку. Це збільшує гнучкість команди та спрощує оновлення та

оновлення мобільного додатка. Це також забезпечує гнучкість для випробувачів, які можуть набагато простіше створювати сценарії тестування. Ці переваги також сприяють економії часу та грошей.

4. Легко транспортувати

Якщо в майбутньому виникає бажання або необхідність перенесення програми в інше середовище розробки, розробникам програм не потрібно починати заново. Вони можуть експортувати програму з React Native, перемістити її в Android Studio або Xcode і продовжити звідти. Це величезна перевага використання React Native для розробки мобільних програм і додає йому гнучкості.

5. Негайно переглянути зміни

React Native пропонує так зване «оперативне перезавантаження» або «гаряче перезавантаження», яке дозволяє розробникам негайно переглядати зміни, які вони внесли в код, в іншому вікні попереднього перегляду в реальному часі. Це дає розробникам велику перевагу завдяки зворотному зв'язку у режимі реального часу.

6. Публікуйте оновлення для своїх програм швидше

Раніше публікація оновлень для вашої програми займала набагато більше часу, і розробникам доводилося заново виконувати процес збирання для кожної програми окремо. З React Native цей процес спростився. Мало того, що обидві програми можуть бути оновлені одночасно, але і весь процес набагато простіше і може бути виконаний набагато швидше. У міру того, як ви створюєте покращення та оновлення для своїх користувачів, розробники впроваджують їх через бездротові (OTA) оновлення, які реалізуються навіть тоді, коли користувачі використовують програму. Потім, коли програма відкриється наступного разу, оновлення буде готове для користувача. Більше немає необхідності оновлювати програму через магазини програм вручну і погоджувати їх з Apple або Android, що економить час і робить процес більш оптимальним.

7. Доповнення існуючого додатку

У вас вже є програма, але ви хочете її розширити з мінімальними витратами? Ми можемо вставляти компоненти інтерфейсу користувача React Native в існуючу програму, не переписуючи його. Це може бути благом, коли ви просто хочете доповнити існуючу програму, не переписуючи її повністю.

3.1.5. Вибір та обґрунтування використання Visual Studio Code в якості IDE для розробки. Visual Studio Code – це «безкоштовний редактор, який допомагає програмісту писати код, допомагає у налагодженні та виправляє код за допомогою методу intelli-sense». Зазвичай, це допомагає користувачам легко писати код. Будь-яка програма/програмне забезпечення, яке ми бачимо або використовуємо, працює з кодом, який виконується у фоновому режимі. Традиційно кодування використовувалося в традиційних редакторах або навіть основних редакторах, таких як блокнот. Ці редактори раніше забезпечували базову підтримку кодувальникам [33].

Visual Studio Code має декілька унікальних функцій. Вони перераховані нижче:

- підтримка кількох мов програмування: підтримує кілька мов програмування. Раніше програмістам була потрібна веб-підтримка: інший редактор для різних мов, але з вбудованою багатомовною підтримкою. Це також означає, що він легко виявляє, чи є якісь помилки або посилання різними мовами, він зможе легко їх виявити;
- Intelli-Sense: він може визначити, чи залишився якийсь фрагмент коду незавершеним. Крім того, загальні синтаксиси змінних та оголошення змінних виконуються автоматично. Приклад: якщо програма використовує певну змінну, і користувач забув її оголосити, intelli-sense оголосить її для користувача;
- крос-платформна підтримка: традиційно редактори підтримували Windows, Linux або Mac. Але Visual Studio Code кроссплатформенний. Таким чином він може працювати на всіх трьох платформах. Також код працює на всіх трьох платформах; в іншому випадку коди програмного забезпечення з

відкритим вихідним кодом та пропрієтарного програмного забезпечення були різними;

- розширення та підтримка: Зазвичай підтримує всі мови програмування, але якщо користувач / програміст хоче використовувати мову програмування, яка не підтримується, він може завантажити розширення та використовувати його. А з погляду продуктивності розширення не уповільнює роботу редактора, оскільки це інший процес;
- репозиторій: з постійно зростаючим попитом на код безпечне та своєчасне зберігання не менш важливе. Він пов'язаний з Git або може бути пов'язаний з будь-яким іншим репозиторієм для отримання або збереження екземплярів;
- ієрархічна структура: файли коду розташовані у файлах та папках. У необхідних файлах коду є також деякі файли, які можуть знадобитися для інших складних проектів. Ці файли можна видалити для зручності;
- покращення коду: деякі фрагменти коду можуть бути оголошені трохи інакше, що може допомогти користувачеві в коді. Ця функція пропонує користувачеві, де необхідно, змінити його на запропонований варіант;
- підтримка терміналу: у багатьох випадках користувачеві необхідно почати з кореня каталогу, щоб почати з певної дії, вбудований термінал або консоль забезпечує підтримку користувача, щоб не перемикатися між двома екранами для того самого;
- мультипроекти: одночасно можна відкривати кілька проектів, що містять кілька файлів/папок. Ці проекти/папки можуть бути пов'язані один з одним, а можуть і не бути;
- підтримка Git: ресурси можна отримати з репозиторію Github Repo в Інтернеті та навпаки; економія також може бути зроблена. Вилучення ресурсів також означає клонування коду, доступного в Інтернеті. Цей код можна пізніше змінити та зберегти;
- коментування: звичайна функція, але деякі мови її не підтримують. Коментування коду допомагає користувачеві згадати чи відстежити відповідно

до бажаної послідовності.

3.2 Схема даних БД

Дизайн бази даних – це набір процесів, що полегшують проектування, розробку, впровадження та обслуговування систем управління корпоративними даними. Правильно спроектована база даних проста в обслуговуванні, покращує узгодженість даних та економічна з погляду дискового простору для зберігання. Розробник бази даних вирішує, як елементи даних співвідносяться та які дані мають бути збережені.

Основними завданнями проектування баз даних у СУБД є створення логічних та фізичних моделей дизайну запропонованої системи баз даних.

Логічна модель концентрується на вимогах до даних та даних, які мають зберігатися незалежно від фізичних міркувань. Її не хвилює, як дані зберігатимуться або де вони зберігатимуться фізично.

Основними критеріями, яким повинна задовольняти спроектована структура баз даних, є забезпечення функціональних вимог додатків і висока продуктивність системи. Погано спроектована база даних може призвести до структурного конфлікту, що істотно утруднить програмування прикладних задач. Проектування бази даних повинне забезпечити цілісність (виключення випадкових втрат чи перекручування даних) і погодженість відновлення даних, захист даних від несанкціонованого доступу. База даних повинна мати здатність адаптації до умов, що змінюються, її використання.

Діаграма відносин сутностей (ER) - це тип блок-схеми, яка ілюструє, як «сутності», такі як люди, об'єкти або концепції, пов'язані один з одним у системі. ER-діаграми найчастіше використовуються для розробки або налагодження реляційних баз даних у галузі розробки програмного забезпечення, інформаційних систем для бізнесу, освіти та досліджень. Також відомі як ERD або ER-моделі, вони використовують певний набір символів, таких як прямокутники, ромби, овали та сполучні лінії для зображення

взаємопов'язаності сутностей, відносин та їх атрибутів. Вони відбивають граматичну структуру, де сутності є іменниками, а відносини - дієсловами [34].

У ER-моделі сутності зображуються позначеними прямокутниками, асоціації (зв'язки) – позначеними ромбами або шестикутниками, атрибути – позначеними овалами, а зв'язки між ними – ненаправленими ребрами, над якими може проставлятися ступінь зв'язку (1 або буква, що заміняє слово «багато») і необхідне пояснення.

На рисунку 3.1 зображена схема бази застосунку.

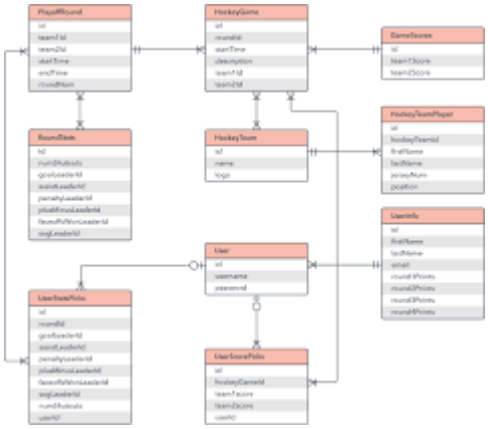

Рисунок 3.1 – ERD діаграма застосунку

Для додатку була розроблена база даних, яка має наступну структуру:

3.3 Захист інформації

В даний час неможливо створити програмне забезпечення, яке на 100% захищене. Імовірність стати жертвою атаки або зазнати вразливості системи безпеки пропорційна зусиллям, які розробники докладають для захисту програм, створених у React Native, від будь-яких атак.

Створення мобільного додатка – це не лише створення екранів, створення химерної анімації та обробка даних із API. Одна з найбільших проблем у процесі розробки – це безпека. Крім того, це один з найбільш часто упущених з уваги аспектів при створенні програмного забезпечення.

Коли ви хочете створити безпечну програму React Native, вам потрібно звернути увагу на безліч факторів. Це може вимагати деякого розуміння власних платформ iOS та Android та загальних правил безпеки з погляду програмування.

Що можна зробити, щоб зробити вашу програму React Native більш безпечним?

1. **Змінні навколишнього середовища**

Змінні середовища дозволяють відокремити секрети від вихідного коду. Це дуже корисно, коли у вас є ключі API або інші облікові дані, якими ви, можливо, не захочете поділитися з іншими людьми. Якщо ви створюєте проект з відкритим кодом, ви можете ділитися вихідним кодом, дозволяючи іншим людям створювати свої файли .env. Це також дуже корисно, якщо ви хочете динамічно налаштовувати програму без зміни вихідного коду. Наприклад, ви можете встановити різні URL-адреси бази даних для кожного середовища. Але все це має сенс тільки в тому випадку, якщо ви не забудете додати файли .env до свого .gitignore! В іншому випадку ви можете отримати несанкціонований доступ до багатьох служб, на які спирається ваш додаток. Якщо ви трохи скористаєтеся Google, ви знайдете безліч історій про людей, які втрачають гроші або дані через секрети, розкриті в репозиторії.

Як правильно керувати змінними оточення в React Native? Під час створення Knots3D було використано одну з найпопулярніших бібліотек, наприклад, react-native-dotenv. Її дуже легко інтегрувати у свій проект та убезпечити свої секрети.

2. **Зберігання sensitive data**

Для постійних даних користувача вам необхідно вибрати правильний тип сховища залежно від його чутливості. Можливо, вам потрібно використовувати ці дані для підтримки в автономному режимі або зберегти токени доступу користувачів, щоб їм не доводилося повторно автентифікуватися щоразу, коли вони використовують програму. Одним із найпопулярніших модулів для зберігання даних у React Native є Async Storage, який було обрано для Knots3D.

Питання в тому, чи достатньо воно безпечне?

Асинхронне сховище - це незашифроване та постійне сховище значень ключів, яке є у всьому додатку. Воно не розділяється між програмами - кожен додаток має своє середовище пісочниці і не має доступу до даних з інших програм. Більше того, це гарна ідея для зберігання неконфіденційних даних у додатку. Це може бути стан Redux, або GraphQL, або деякі глобальні змінні програми. З іншого боку, ви не повинні використовувати це для зберігання токенів та секретів, оскільки сховище жодним чином не зашифроване.

3.4 Інтерфейс

Відповідно до розробленої структури та макету, було реалізовано головну сторінка додатку (Рисунок 3.2). Вона містить всі категорії вузлів, які представлені в базі даних, кнопка для зміни мови, кнопка поділитись додатком, категорія обраних вузлів.

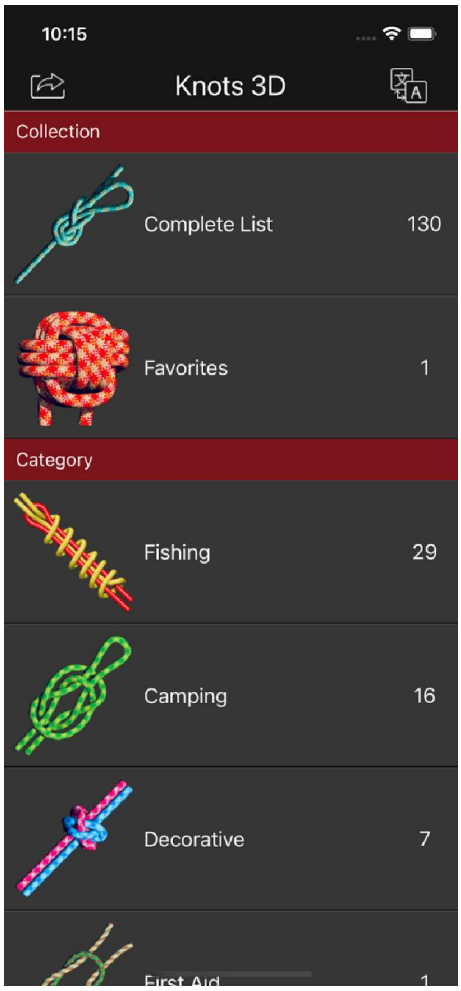

Рисунок 3.2 – Головна сторінка додатку в IOS

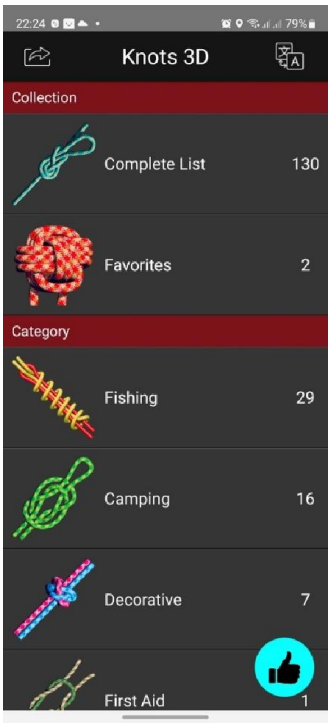

Рисунок 3.3 – Головна сторінка додатку в Android

На рисунку 3.4 зображена сторінка списку вузлів за обраною категорією, яка містить прев’ю кожного вузла, його назву і можливі підназви. Також в шапці сторінки є назва категорії і дві кнопки: повернутися на попередню сторінку і кнопка пошуку.

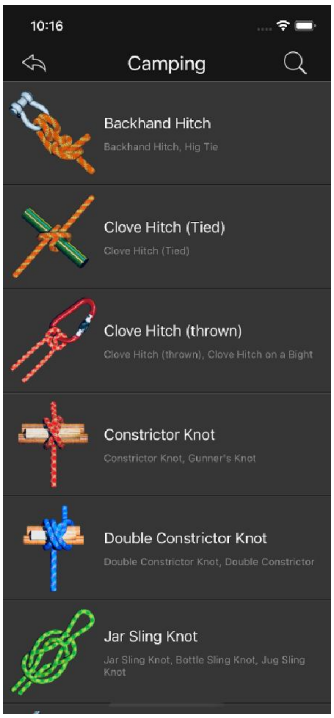

Рисунок 3.4 – Сторінка списку вузлів групи «Camping»

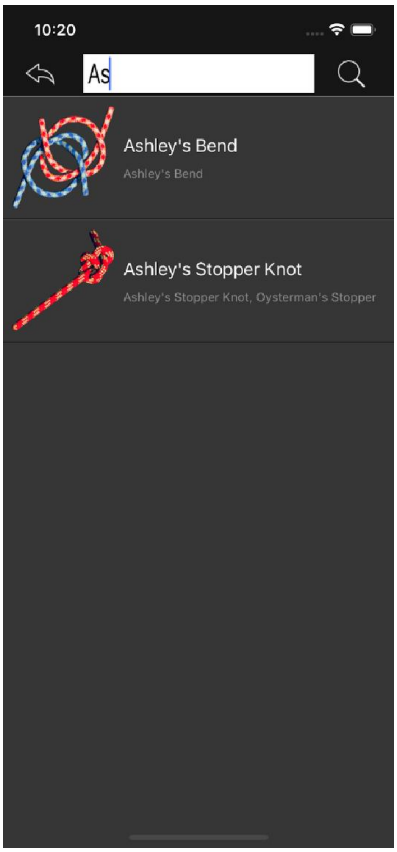

Рисунок 3.5 – Сторінка списку вузлів групи «Camping»

На рисунку 3.6 зображена сторінка вибору мови, яка містить вибір наступних мов:

- англійська;
- німецька;
- іспанська;
- російська;
- французька;
- італійська;
- турецька;
- українська.

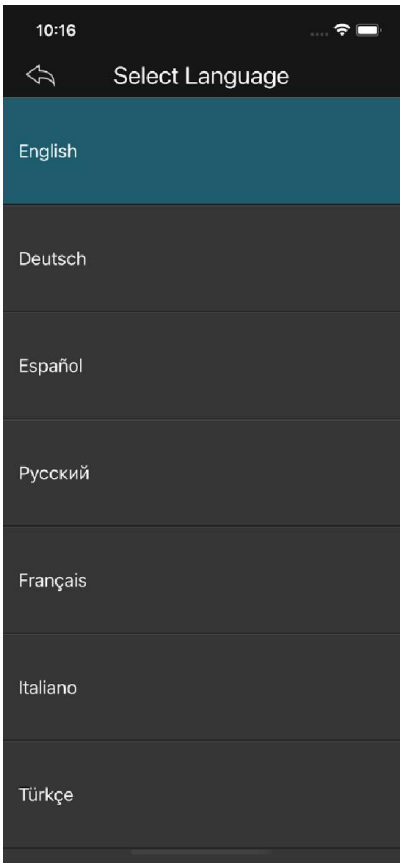

Рисунок 3.6 – Сторінка вибору мови

На рисунку 3.7 зображено сторінку інформації про вузол. На ній можна побачити сам вузол, опис вузла, його АВОК, інші можливі назви вузла, його міцність на розрив та категорії, в які входить обраний вузол. В шапці сторінки є назва вузла, кнопка повернення на попередню сторінку і кнопка додавання вузла в «Обране». В підвалі сторінки ж є наступні елементи: кнопка віддзеркалення вузла відносно осі ординат, кнопка перевертання вузла відносно осі абсцис, кнопка перезапуску анімації, якщо вона завершена, або плей/пауза, якщо вона в процесі, кнопка вибору режиму анімації (360° огляд або анімація зав’язування вузла). З правої сторони є слайдер зміни швидкості анімації. Також, рухаючи палець вправо-вліво по вузлу, можна пролистувати анімацію власноруч.

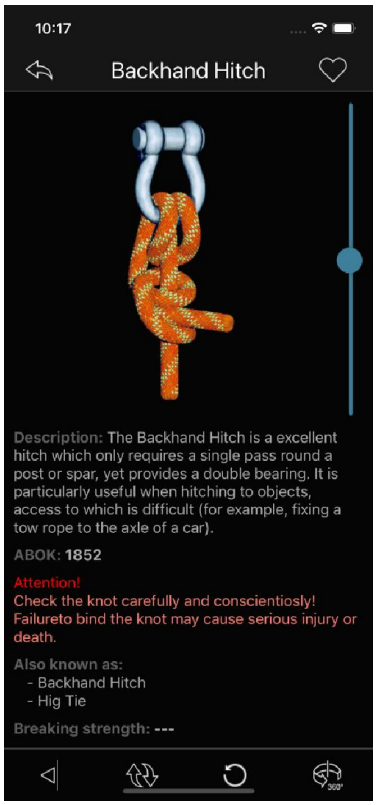

Рисунок 3.7 – Сторінка інформації про вузол «Backhand Hitch»

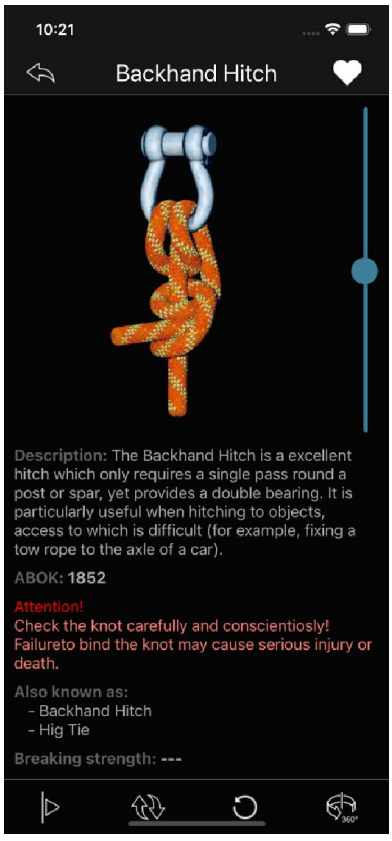

Рисунок 3.8 – Віддзеркалений вигляд вузла «Backhand Hitch»

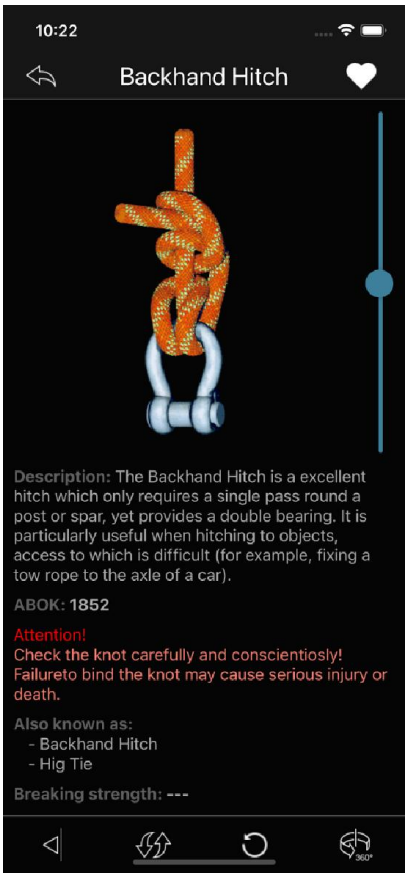

Рисунок 3.9 – Перевернутий вигляд вузла «Backhand Hitch»

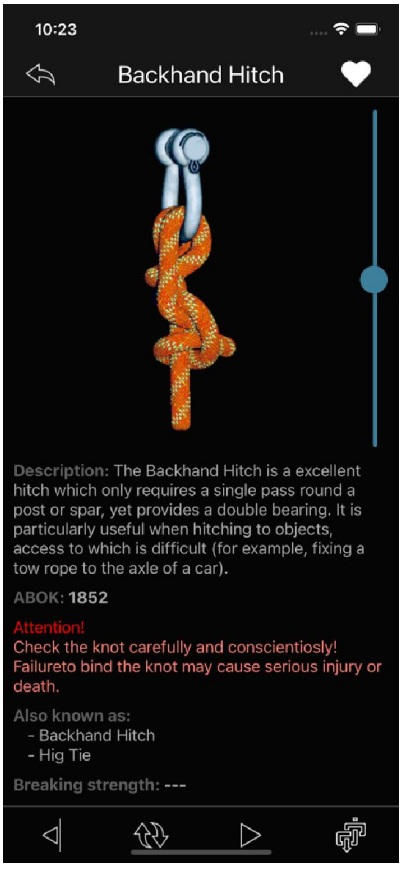

Рисунок 3.10 – Вид збоку «Backhand Hitch» під час перегляду в режимі 360

На рисунку 3.11 зображена головна сторінка після зміни мови додатку на німецьку.

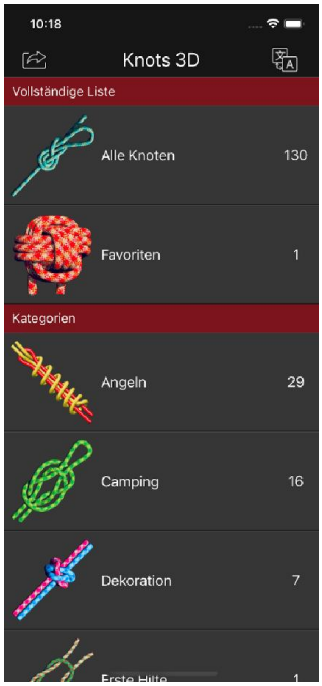

Рисунок 3.11 – Головна сторінка додатку німецькою мовою

На рисунку 3.12 зображений вигляд додатку після натиснення кнопки «Поділитися»

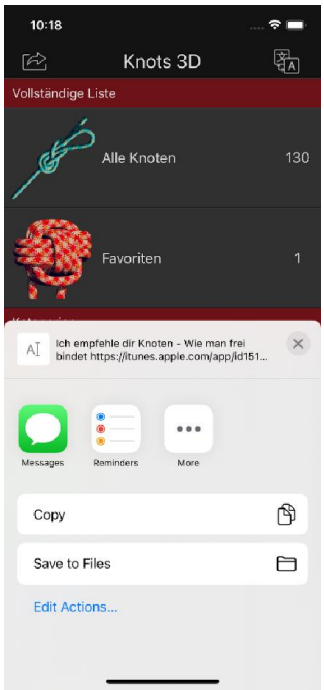

Рисунок 3.12 – Поділитися додатком

На рисунках 3.13 –3.16 представлено адаптацію додатку під планшет:

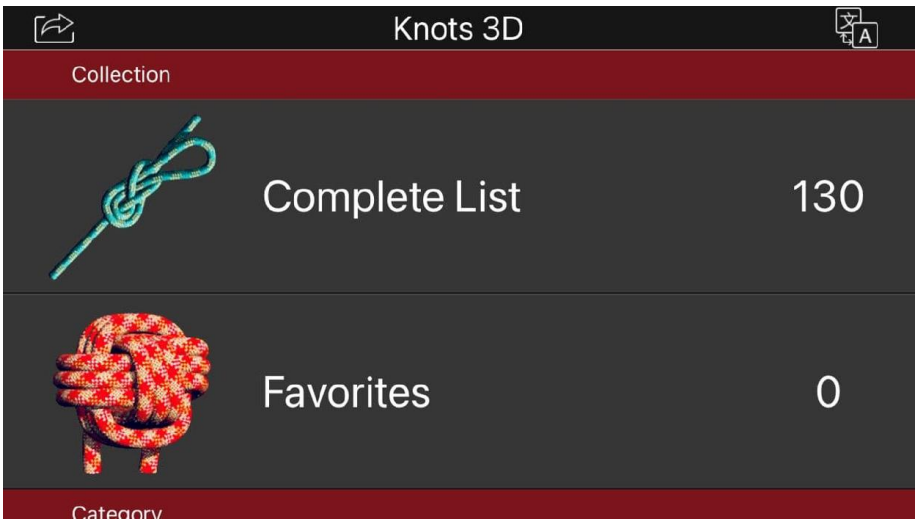

Рисунок 3.13 – Iphone se portrait (портретний режим)

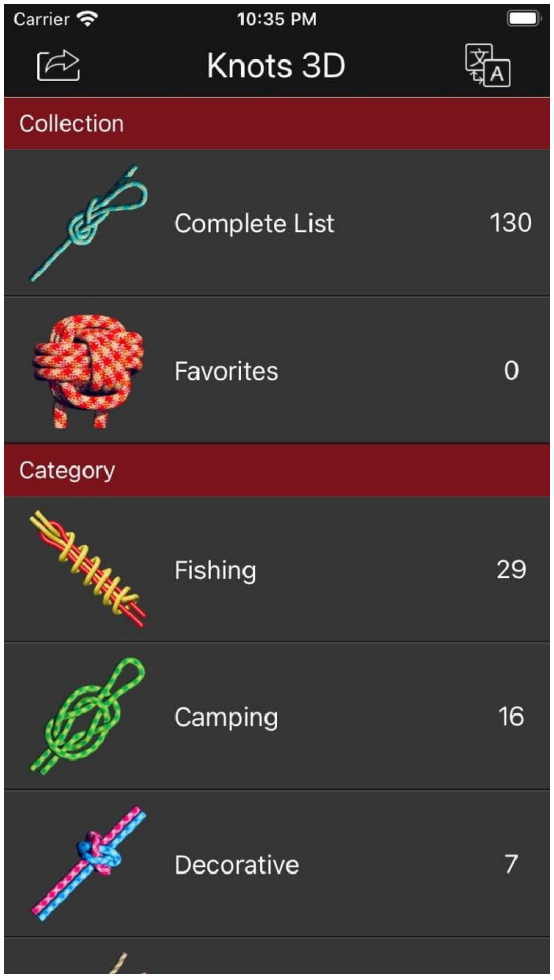

Рисунок 3.14 – Iphone se (список категорій)

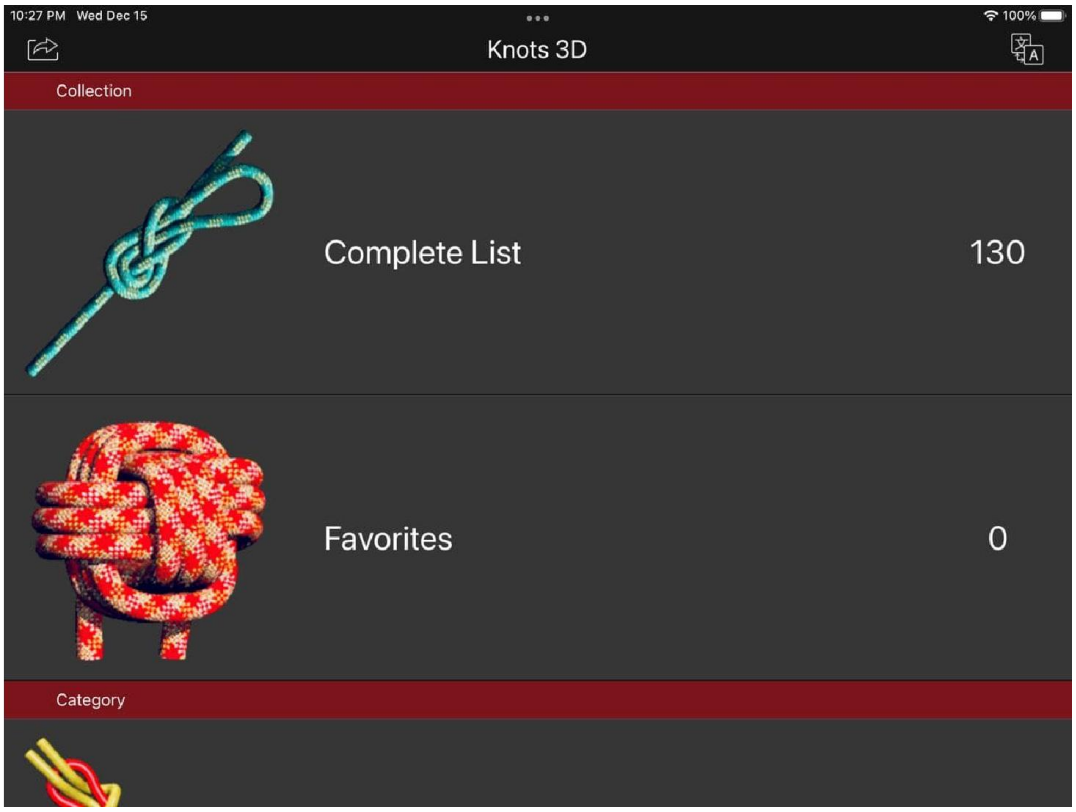

Рисунок 3.15 –Планшет портретний

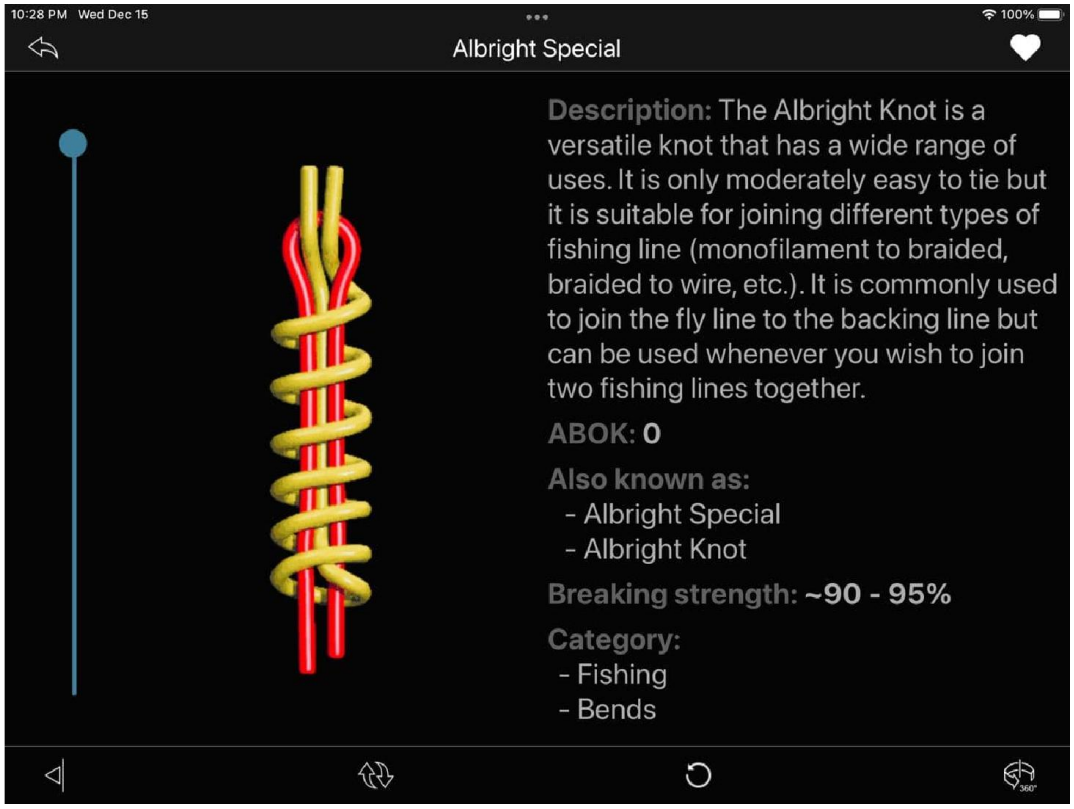

Рисунок 3.16 –Інформація про вузол в портретному режимі

3.5 Структура модулів

Для більшої зручності програмної реалізації, а також для спрощення читання програмного коду було обрано наступну структуру модулів.

```
├── ios/
│   ├── knots3D/ - dir for builded app
│   └── Podfile - file to collect all necessary modules for ios bundle
├── android/
│   ├── app/ - dir for builded app
│   └── graddleFiles - files to collect all necessary modules for ios bundle
├── src/
│   ├── assets/ - dir for a static images, svgs, audio, etc
│   │   ├── img/ - dir for png/jpeg images
│   │   └── favicon.ico - favicon file
│   │
│   ├── actions/
│   │   ├── app_part.ts - exports action creators (function)
│   │   ├── ...
│   │   └── index.ts - includes exports of all actions
│   │
│   ├── constants/
│   │   ├── actions.ts - exports constants with action names
│   │   ├── ...
│   │   └── index.ts - includes exports of all constants
│   │
│   ├── components/ - dir for presentative components
│   │   └── component_name/ - example of component directory
│   │       ├── styles/
│   │       │   └── index.sass
│   │       └── index.tsx
│   │
│   ├── controllers/ - dir for complex pages with logic
│   │   └── api_controller.ts - example controller
│   │
│   ├── containers/ - dir for complex pages with logic
│   │   └── container_name/ - example of container directory
│   │       ├── styles/
│   │       │   └── index.sass
│   │       └── index.ts
│   │
│   ├── sagas/
│   │   ├── api.ts - example middleware
│   │   ├── ...
│   │   └── index.ts - includes main yield
│   │
│   ├── reducers/
│   │   ├── app.ts - example reducer
│   │   ├── ...
│   │   └── index.ts - main reducer that include combining of all other reducers
│   │
│   ├── db.tsx - file to take control of database
│   ├── store.tsx - file store creation
│   └── index.tsx - main file
```

РОЗДІЛ 4
ТЕСТУВАННЯ ТА ВВЕДЕННЯ В ЕКСПЛУАТАЦІЮ

4.1 Опис методів тестування

Що таке автоматизоване тестування?

Автоматизоване тестування – це програмне забезпечення для автоматизації ручного процесу перевірки та перевірки програмного продукту, керованого людиною. Більшість сучасних проектів програмного забезпечення включають автоматичне тестування із самого початку. Однак, щоб повною мірою оцінити цінність автоматизованого тестування, необхідно зрозуміти, яким було життя до того, як воно стало широко поширеним [35].

Раніше коли ручне тестування було нормою, софтверні компанії зазвичай наймали команду QA на повний робочий день. Ця група розробить набір «планів тестування» або покрокових контрольних списків, які підтверджують, що функція програмного проекту поводиться так, як очікувалося. Команда QA потім вручну виконувала ці контрольні списки щоразу, коли в проект програмного забезпечення додавалися нове оновлення або змінна, а потім повертала результати планів тестування групі інженерів для перевірки і подальшої розробки для вирішення проблем.

Цей процес був повільним, дорогим і схильним до помилок. Автоматичне тестування дає величезні переваги для ефективності роботи команди та рентабельності інвестицій груп із забезпечення якості.

Завдяки автоматичному тестуванню відповідальність за право власності переходить до рук інженерної групи. Плани тестування розробляються паралельно з регулярною розробкою функцій дорожньої карти, а потім виконуються автоматично за допомогою безперервної інтеграції програмного забезпечення. Автоматичне тестування сприяє скороченню розміру команди QA і дозволяє групі QA зосередитися на найважливіших функціях.

Різновиди автоматизованого тестування.

1. End-to-End тести

Мабуть, найцінніші для реалізації тести – це наскрізні (E2E) тести. E2E-тести імітують взаємодію на рівні користувача з повним стеком програмного продукту. Плани тестування E2E зазвичай охоплюють такі історії на рівні користувача, як: «користувач може увійти в систему», «користувач може категоризувати обраний тип продуктів», «користувач може змінити налаштування електронної пошти».

Інструменти тестування E2E фіксують та відтворюють дії користувача, тому плани тестування E2E потім стають записами основних потоків взаємодії з користувачем. Якщо програмному продукту не вистачає будь-якого покриття для автоматизованого тестування, він отримає найбільшу користь від застосування E2E-тестів для найважливіших бізнес-потоків. Тести E2E можуть бути дорогими заздалегідь, щоб захопити і записати послідовність дій користувача. Якщо програмний продукт не випускає швидких щоденних випусків, може бути більш економічним доручити групі людей вручну виконати плани тестування E2E [36].

2. Модульні тести

Як випливає з назви, модульні випробування охоплюють окремі одиниці коду. Одиниці коду найкраще вимірювати у визначеннях функцій. Модульний тест охоплюватиме окрему функцію. Модульні тести підтвердять, що очікуване введення функції відповідає очікуваному результату. Код, у якому використовуються конфіденційні обчислення (наприклад, у сфері фінансів, охорони здоров'я чи авіакосмічної галузі), найкраще покривається модульними тестами. Модульні тести недорогі, швидко впроваджуються та забезпечують високу окупність інвестицій[37].

3. Інтеграційні тести

Часто блок коду виконує зовнішній виклик сторонньої служби. Основна тестована кодова база не матиме доступу до коду цієї сторонньої утиліти.

Інтеграційні тести мають справу з імітацією цих сторонніх залежностей та твердженням, що код, що взаємодіє з ними, поводиться належним чином [38].

Інтеграційні тести схожі на модульні тести за способом написання та інструментарієм. Інтеграційні тести можуть бути недорогою альтернативою E2E-тестам, проте окупність інвестицій залишається спірною, коли вже існує комбінація модульних тестів та E2E.

4. Тести продуктивності

При використанні в контексті розробки програмного забезпечення продуктивність використовується для опису швидкості та оперативності реагування програмного проекту. Ось деякі приклади показників продуктивності: час завантаження сторінки, час до першої обробки, час відгуку результатів пошуку. Тести продуктивності створюють вимірювання та затвердження для цих прикладів. Автоматичні тести продуктивності запускатимуть тестові приклади за цими показниками, а потім попереджатимуть команду про будь-які регреси або втрату швидкості.

В розробці додатку було обрано використовувати модульне тестування. Для його реалізації використано тестовий ранер Jest.

Jest – це середовище тестування з відкритим вихідним кодом, побудоване на JavaScript, розроблене в основному для роботи з веб-програмами на основі React або React Native. Часто модульні тестування не дуже корисні при запуску в інтерфейсі будь-якого програмного забезпечення. В основному це пов'язано з тим, що модульні тести для зовнішнього інтерфейсу вимагають великої та трудомісткої настройки. Цю складність можна значно зменшити за допомогою фреймворку Jest [39].

Більше того, Jest можна використовувати для перевірки майже всього, що пов'язане з JavaScript, особливо візуалізації веб-додатків браузером. Jest також широко використовується для автоматичного тестування браузерів, що робить його одним із найпопулярніших фреймворків для тестування Javascript.

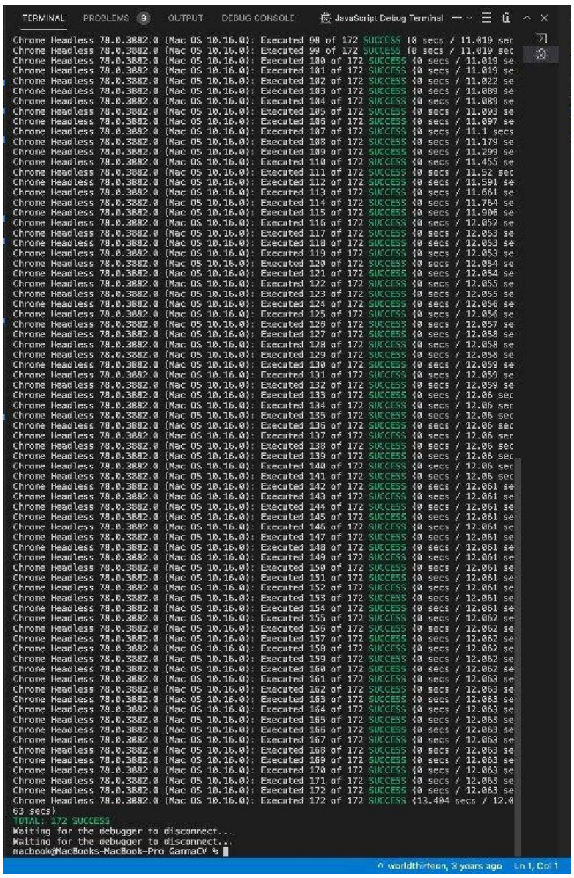

Рисунок 4.1 – Тестування додатку

4.2 Введення в експлуатацію

Мобільний додаток має бути доступний саме там, де проживає найбільша потенційна аудиторія. В даний час цими місцями для Android та iOS є, відповідно, Google Play та Apple App Store.

Процес публікації програми дозволяє зробити вашу програму доступною для мільйонів потенційних користувачів.

У наші дні в магазинах програм випускаються тисячі програм. Через це треба виділятися з натовпу. Успішна демонстрація програми вимагає деякого вміння у маркетингу та презентації. На щастя, досвідчений партнер неодмінно проведе вас через весь процес. Зрештою, публікація програми – життєво важлива частина життєвого циклу розробки програми.

Офіційний посібник від Google і посібник від Apple – відмінні відправні точки для того, щоб опублікувати створений застосунок.

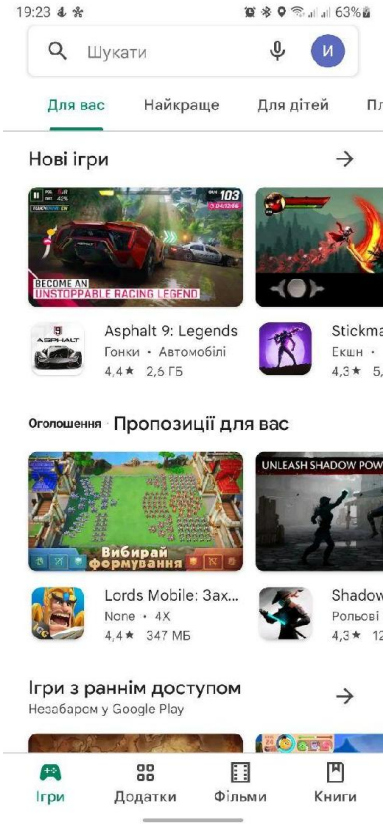

Рисунок 4.2 – Головна сторінка Google Play Store

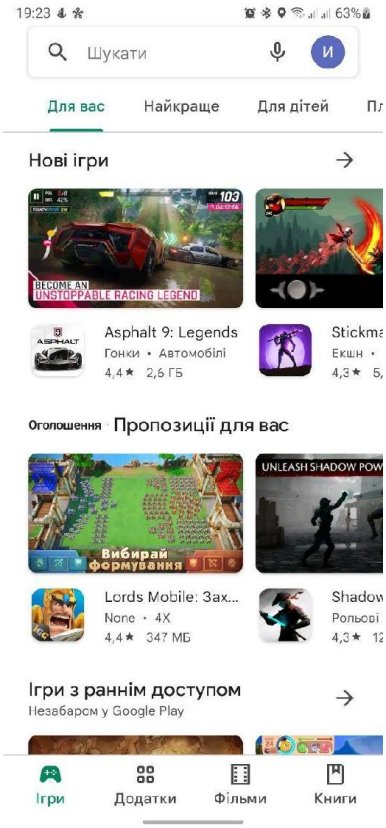

Рисунок 4.3 – Головна сторінка App Store

Публікація програми в Google Play

Перш ніж почати публікацію програми, треба створити свій аккаунт в Google Play Console, для цього потрібно оплатити підписку в розмірі 25$. Після реєстрації і початком публікації рекомендується мати під рукою кілька речей. У магазинах додатків є структурований спосіб представлення додатків. Ось чому можна наперед дізнатися, що потрібно для запуску програми. Google створив посібник, де докладно описується кожна частина процесу завантаження програми. Ось список усіх необхідних умов:

- ім'я програми. Ім'я, яке буде показано користувачам Google Play. Назва пакета програми. Назва пакета програми має бути унікальною для всього Google Play. Більше того, його не можна буде змінити у майбутньому. Кожна програма Android має унікальний ідентифікатор програми, який виглядає як повернена адреса www, наприклад com.example.myapp. Цей ідентифікатор однозначно ідентифікує програму на пристрої та в Google Play. Якщо ви хочете завантажити нову версію своєї програми, ідентифікатор програми (і сертифікат, яким ви його підписуєте) повинні збігатися з вихідним APK * - якщо ви змініте ідентифікатор програми, Google Play буде розглядати APK як зовсім іншу програму;
- Android Package Kit (скорочено APK) – це формат файлу пакета, що використовується операційною системою Android для розповсюдження та встановлення мобільних додатків. Так само, як системи Windows (ПК) використовують файл .exe для встановлення програмного забезпечення, APK робить те саме для Android;
- короткий опис програми. Це короткий опис функціональності програми. Це 80-символьне поле, яке відображається користувачеві у списках, а потім його можна розгорнути, щоб побачити повний опис;
- повний опис програми. Тут перераховано всі найважливіші функції та короткі інструкції для користувачів. Це один із найважливіших розділів на сторінці презентації програми. Ідеальний опис – короткий інформативний абзац. У цьому полі Google Play можна розмістити до 4000 символів;

- графічні ресурси. Сюди входять усі графічні ресурси для презентації програми. Це значок програми, відео, скріншоти та тематична графіка. Ці ресурси призначені для демонстрації всіх найважливіших функцій. Тут Google детально описує потрібні активи;
- тип та категорії програми. Google Play вимагає, щоб ви вказали категорію своєї програми. Призначення правильних категорій допоможе вам зробити програму більш доступною для користувачів. Тут це детально описано;
- рейтинг контенту Ви повинні дотримуватися політики рейтингу контенту Google і правдиво вказувати правильний віковий рейтинг і те, який контент ви представляєте користувачам. Повний посібник із присвоєння рейтингу контенту є тут;
- контактна інформація для власника програми;
- URL політики конфіденційності. Політика конфіденційності тепер є обов'язковою, якщо програма Google Play запитує користувачів конфіденційні дозволи. Потрібно дотримуватися GDPR та повідомляти користувачам, для чого вам потрібні їх дані, а також як вони зберігатимуться та оброблятимуться;
- країни та локалізація. Можна вибрати країни, в яких буде доступна програма, а також забезпечити локалізацію всього завантаженого контенту [40].

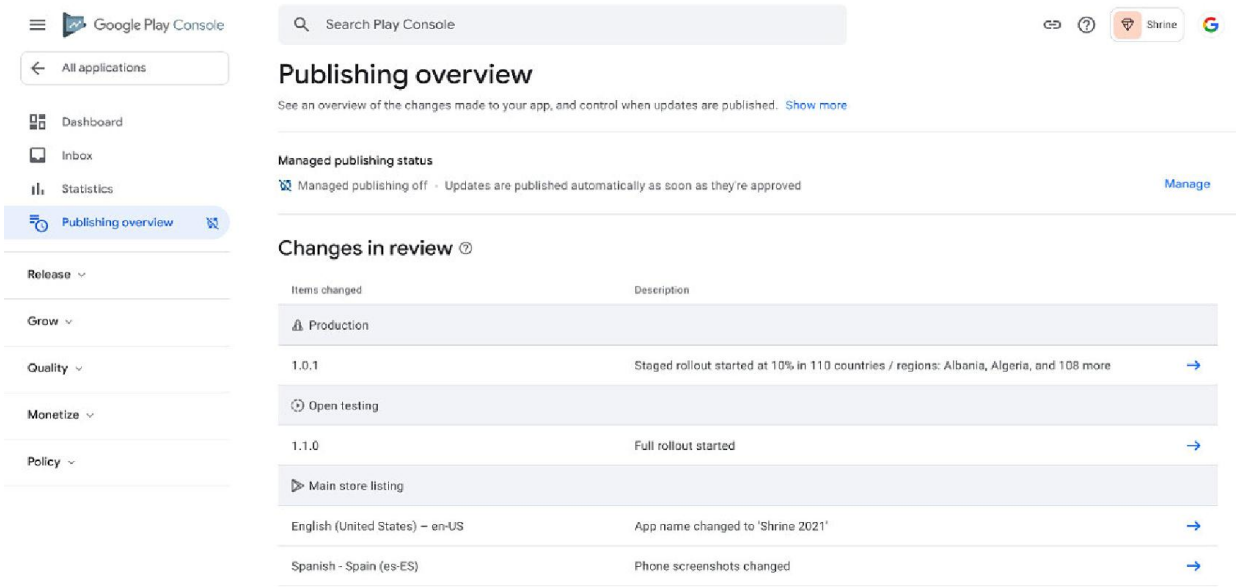

Рисунок 4.4 – Сторінка overview Google Play Console

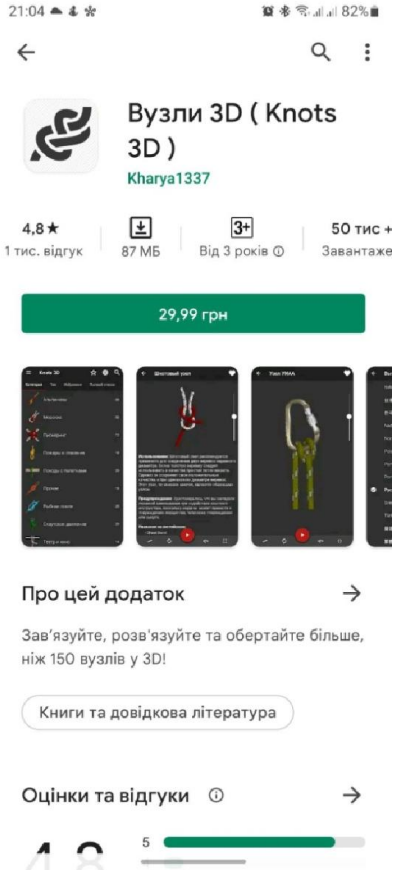

Рисунок 4.5 – Сторінка додатку в Google Play Store

Як можна побачити з скріншоту, додаток було завантажено користувачами більше 50 тисяч разів та добре оцінений, судячи з відгуків.

Як підготуватися до публікації програми в App Store

Як і в Google Play, в App Store треба зареєструватись і оформити підписку в розмірі 99$. Після реєстрації треба мати список речей, необхідних для завантаження програми. Це можна і потрібно зробити до того, як ви приступите до процесу завантаження програми. Apple створила офіційний посібник з описом цих частин фінальної сторінки презентації програми. Для успішного запуску необхідно підготувати наступний список матеріалів-вітрин:

- ім'я програми;
- значок. Піктограма програми – це перше, що бачить користувач магазину програм. Попрацюйте з графічним дизайнером, щоб створити простий і відомий значок;

- підзаголовок. Це короткий 30-символьний опис програми, що відображається під назвою програми;
- попередні перегляди програм. У розділі прев'ю додатків представлені основні функції програми. Програма автоматично відтворює попередній перегляд із вимкненим звуком, коли користувачі переглядають сторінку продукту, тому переконайтеся, що перші кілька секунд відео виглядають привабливо;
- скріншоти. Знімки екрана підкреслюють суть програми. Ви можете розмістити до 10 зображень на сторінці в App Store. Залежно від того, які типи пристроїв підтримує програму, вам необхідно додати скріншоти для різних розмірів екрана, наприклад, окремі приклади для великих пристроїв iPad Pro та iPhone Pro Max. Якщо програма підтримує темний режим, подумайте про те, щоб увімкнути хоча б один знімок екрана, який показує, як це виглядає для користувачів;
- опис. Ідеальний опис – це короткий інформативний абзац, за яким слідує короткий список [41].

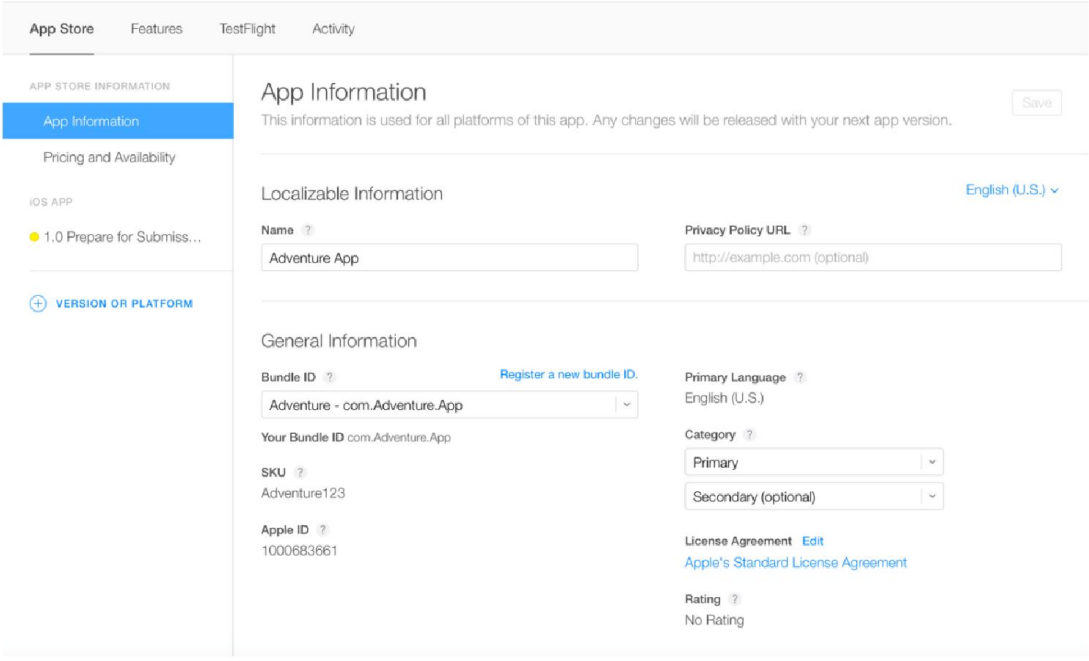

Рисунок 4.6 – Сторінка App Information XCode

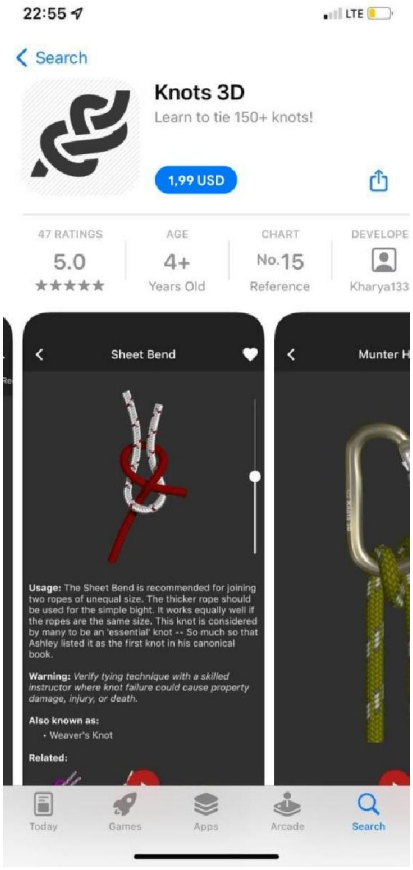

Рисунок 4.7 – Сторінка App Information XCode

Як можна побачити з скріншоту, додаток було потрапив в чарти на 15 місце та був оцінений користувачами на 5 зірок.

ВИСНОВКИ

У дипломній роботі магістра розроблено структуру, реалізовано її на практиці, а також виконано комплексне тестування мультиплатформенного мобільного застосунку спрямованого навчити в’язати вузли.

При розробці додатку були вибрані та обґрунтовані сучасні мови програмування та фреймворки, такі як JavaScript, TypeScript, React Native, Jest, SQLite що дозволяють створювати інтерактивні мобільні додатки швидко та якісно.

Застосунок забезпечує такі функції:

1. Надає користувачам всю необхідну інформацію про вузли(опис, складність, АВОК, застереження, можливі назви).
2. Додавання вузлів до обраного та їх видалення звідти.
3. Користувачі можуть виконувати пошук серед списку вузлів.
4. Є можливість зміни мови додатку.
5. Користувачі можуть переглядати анімацію зав’язування вузла.
6. Користувачі можуть змінювати швидкість програвання анімації.
7. Можна керувати анімацією зав’язування вузла, а саме програвати її вперед або назад.
8. Можна переключити анімацію зав’язування вузла на анімацію перегляду вузла в 360°.
9. Можна керувати анімацією відображення вузла в 360°, а саме програвати її вперед або назад.
10. Користувачі можуть віддзеркалювати вузли стосовно осі ординат.
11. Користувачі можуть поширювати застосунок будь-яким зручним для них способом.

Отже розроблений застосунок відповідає всім вимогам, поставленим на етапі постановки завдання.

Також застосунок було опубліковано на сучасних маркетплейсах мобільних додатків, таких як Google Play Store і Apple App Store, і почав користуватись популярністю серед користувачів, про що свідчать високі оцінки та кількості завантажень.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Клиффорд В. Эшли. The Ashley Book of Knots – Doubleday; 1st edition, 1993 – 619с.

2. Apple iOS [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.apple.com/ru/ios/ios-15/

3. Що таке Android [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.android.com/intl/ru_ru/what-is-android/

4. KaiOS [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.kaiostech.com/

5. SalifishOS [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://sailfishos.org/

6. Harmony OS [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.harmonyos.com/en/

7. Types of mobile app [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://clevertap.com/blog/types-of-mobile-apps/

8. PWA — это просто [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://habr.com/ru/post/418923/

9. React Native [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://reactnative.dev/

10. Why Use React Native for Your Mobile App? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.stxnext.com/blog/why-use-react-native-your-mobile-app

11. Xcode [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://developer.apple.com/xcode/

12. Facebook [Електронний ресурс]. – Режим доступу: https://www.facebook.com

13. Facebook Ads[Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.facebook.adsmanager&hl=uk&gl=US

14. Walmart [Електронний ресурс]. – Режим доступу: https://www.walmart.com

15. Bloomberg [Електронний ресурс]. – Режим доступу: https://www.bloomberg.com/

16. Instagram [Електронний ресурс]. – Режим доступу: https://www.instagram.com

17. SoundCloud [Електронний ресурс]. – Режим доступу: https://soundcloud.com

18. Townske [Електронний ресурс]. – Режим доступу: https://play.google.com/store/apps/details?id=com.townske.android&hl=ru&gl=US

19. Gyroscope [Електронний ресурс]. – Режим доступу: https://www.gyroscope.com

20. Wix [Електронний ресурс]. – Режим доступу: https://ru.wix.com

21. Delivery.com [Електронний ресурс]. – Режим доступу: https://www.delivery.com

22. Мартін Фовлер. UML Distilled: A Brief Guide to the Standard Object Modeling Language – Еддісон-Уеслі Професіонал; 3-тє видання, 2003 – 208с.

23. Інформаційні технології та моделювання бізнес-процесів : навч. посіб. / О. М. Томашевський, Г. Г. Цегелик, М. Б.Вітер, В. І. Дубук. – К. : ЦУЛ, 2012. - 296 с.

24. UML – діаграма варіантів використанні [Електронний ресурс]. – Режим доступу: https://habr.com/post/47940/

25. Діаграма станів [Електронний ресурс]. – Режим доступу:https://lubbook.org/book_746_glava_8_Tema_8._Derzhavna_reєstraci.html

26. Калянов Г. Н. Моделювання, аналіз, реорганізація та автоматизація бізнес-процесів: навч. посібник. М.: Фінанси та статистика, 2006. 240 с.

27. Шон Адамс, Пітер Доусон, Джон Фостер, Тоні Седдон – Graphic Design Rules: 365 Essential Dos and Don'ts – Princeton Architectural Press – 384с.

28. About SQLite [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.sqlite.org/about.html

29. Введение в JavaScript [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://learn.javascript.ru/intro

30. What is TypeScript [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.typescriptlang.org/

31. React Native. Learn once, write anywhere. [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://reactnative.dev/

32. Дерек Аустин. How to Set Up VS Code Like a Pro in Just 5 Minutes [Электронный ресурс]: [Стаття]. – BetterProgramming, 2020 – https://betterprogramming.pub/how-to-set-up-vs-code-like-a-pro-in-just-5-minutes-65aaa5788c0d

33. Що таке ER-діаграма, і як її створити? [Електронний ресурс]: [Веб-сайт]. – Електронні дані. – Режим доступу: https://www.lucidchart.com/pages/ru/erd-диаграмма

34. What is a test automation? [Електронний ресурс]: [Блог]. – Електронні дані. – Режим доступу: https://www.testim.io/blog/what-is-test-automation/

35. Что такое end-to-end тестирование? [Електронний ресурс]: [Блог] – Електронні дані. – Режим доступу: https://qna.habr.com/q/401848

36. Unit Testing Tutorial [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://www.guru99.com/unit-testing-guide.html

37. Integration Testing: What is, types, Top Down and Bottom Up, example [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://www.guru99.com/integration-testing.html

38. Performance Testing: What is, types, Top Down and Bottom Up, example [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://www.guru99.com/performance-testing.html

39. Jest [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://jestjs.io/

40. How to upload an app to Google Play Store [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://appinventiv.com/blog/how-to-submit-app-to-google-play-store/

41. How to submit your app to App Store [Електронний ресурс]: [Веб-сайт] – Електронні дані. – Режим доступу: https://codewithchris.com/submit-your-app-to-the-app-store/

ДОДАТОК А
ПРОГРАМНИЙ КОД ДОДАТКУ

Animations.js

```
import React from 'react';
import { Animated, Easing, View, ViewPropTypes } from 'react-native';
import { PanGestureHandler, State } from 'react-native-gesture-handler';
import PropTypes from 'prop-types';

export default class Animation extends React.Component {
  constructor(props) {
    super(props);

this.animationValue = new Animated.Value(0);
this.animationValue.addListener(({ value }) => {
      const { frameCount } = this.props;

      if (value >= 0 && value <= frameCount) {
this.setState({ value });
      }
    });
this.interpolationRange = {};

this.state = this.calculateImageFramesSize();

this.generateInterpolationRanges();
  }

  calculateImageFramesSize = () => {
    const { columns, rows, width, height } = this.props;

    let { frameHeight, frameWidth } = this.props;

    let ratio = 1;

    let imageHeight = frameHeight * rows;

    let imageWidth = frameWidth * columns;

    if (width) {
      ratio = (width * columns) / imageWidth;
      frameHeight = (imageHeight / rows) * ratio;
      imageHeight *= ratio;
      imageWidth = width * columns;
      frameWidth = width;
    } else if (height) {
      ratio = (height * rows) / imageHeight;
      imageHeight = height * rows;
      frameWidth = (imageWidth / columns) * ratio;
      imageWidth *= ratio;
      frameHeight = height;
    }

    return {
      imageHeight,
      imageWidth,
      frameHeight,
      frameWidth,
    };
  };
```

```
  changeView = async (callBack) => {
callBack();
    await this.setState(this.calculateImageFramesSize());
    await this.generateInterpolationRanges();
  };

  getFrameCoords = (i) => {
    const { columns } = this.props;
    const { frameHeight, frameWidth } = this.state;
    const currentColumn = i % columns;
    const x = -currentColumn * frameWidth;
    const y = -((i - currentColumn) / columns) * frameHeight;

    return {
      x,
      y,
    };
  };

  generateInterpolationRanges = () => {
    const { frameCount } = this.props;
    const frameCoords = Array.from({ length: frameCount }, (_, i) =>
this.getFrameCoords(i)
    );
    const input = [].concat(
      ...Array.from({ length: frameCount }, (_, i) => [i, i + 1])
    );
    const outputY = [].concat(...frameCoords.map(({ y }) => [y, y]));
    const outputX = [].concat(...frameCoords.map(({ x }) => [x, x]));

this.interpolationRange = {
      translateY: {
        in: input,
        out: outputY,
      },
      translateX: {
        in: input,
        out: outputX,
      },
    };
  };

  stop = () => {
this.animationValue.stopAnimation();
  };

  reset = () => {
this.animationValue.setValue(0);
  };

  start = (speed = 50) => {
    const { frameCount, onEndAnimation, loop } = this.props;

    const animation = Animated.timing(this.animationValue, {
      toValue: frameCount,
      duration: (frameCount / (speed / 2)) * 1000,
      easing: Easing.linear,
    });

    if (loop) {
this.loopAnimation(animation);
    } else {
```

```
animation.start(({ finished }) => {
        if (finished) {
onEndAnimation();
        }
      });
    }
  };

  loopAnimation = (animation) => {
animation.start(({ finished }) => {
      if (finished) {
this.reset();
this.loopAnimation(animation);
      }
    });
  };

  onSwipe = ({ nativeEvent }) => {
    let { value } = this.state;

    const { translationX } = this.state;
    const { loop, frameCount } = this.props;
    const currentX = nativeEvent.x;
    const direction = currentX <= translationX ? 'left' : 'right';

this.setState({
      translationX: currentX,
    });

    if (Math.abs(currentX - translationX) > 1) {
      if (direction === 'left') {
        value -= 1;
      } else {
        value += 1;
      }

      if (loop) {
        if (value > frameCount) {
          value = 0;
        } else if (value < 0) {
          value = frameCount;
        }
      }
this.animationValue.setValue(value);
    }
  };

  onSwipeStateChanged = ({ nativeEvent }) => {
    const { onEndAnimation, onSwipe, frameCount } = this.props;
    const { value } = this.state;
    switch (nativeEvent.state) {
      case State.BEGAN: {
onSwipe();
this.setState({
          translationX: nativeEvent.x,
        });
        break;
      }
      case State.END: {
        if (value >= frameCount - 1) {
onEndAnimation();
        }
        break;
```

```
      }
    }
  };

render() {
    const { imageHeight, imageWidth, frameHeight, frameWidth } = this.state;
    const { style, source, onLoad } = this.props;

    const {
      translateY = { in: [0, 0], out: [0, 0] },
      translateX = { in: [0, 0], out: [0, 0] },
    } = this.interpolationRange || {};

    return (
<PanGestureHandler
        onGestureEvent={this.onSwipe}
        onHandlerStateChange={this.onSwipeStateChanged}
>
<View
          style={[
            style,
            {
              justifyContent: 'center',
              alignItems: 'center',
            },
          ]}
>
<View
            style={{
              height: frameHeight,
              width: frameWidth,
              overflow: 'hidden',
            }}
>
<Animated.Image
              source={source}
              onLoad={onLoad}
              style={{
                height: imageHeight,
                width: imageWidth,
                transform: [
                  {
                    translateX: this.animationValue.interpolate({
                      inputRange: translateX.in,
                      outputRange: translateX.out,
                    }),
                  },
                  {
                    translateY: this.animationValue.interpolate({
                      inputRange: translateY.in,
                      outputRange: translateY.out,
                    }),
                  },
                ],
              }}
            />
</View>
</View>
</PanGestureHandler>
    );
  }
}
```

```
Animation.propTypes = {
  source: PropTypes.number.isRequired,
  columns: PropTypes.number.isRequired,
  rows: PropTypes.number.isRequired,
  frameHeight: PropTypes.number.isRequired,
  frameWidth: PropTypes.number.isRequired,
  frameCount: PropTypes.number.isRequired,
  style: ViewPropTypes.style,
  onEndAnimation: PropTypes.func,
  onSwipe: PropTypes.func,
  onLoad: PropTypes.func,
  width: PropTypes.number,
  height: PropTypes.number,
  loop: PropTypes.bool,
};

Animation.defaultProps = {
  style: {},
  onLoad: null,
  onEndAnimation: null,
  onSwipe: null,
  width: null,
  height: null,
  loop: false,
};
```

Knots.jsx

```
import React from 'react';
import PropTypes from 'prop-types';
import { connect } from 'react-redux';
import { View, InteractionManager, StyleSheet } from 'react-native';
import { OptimizedFlatList } from 'react-native-optimized-flatlist';
import { KnotCategory } from '../components/Category';
import { knotSet } from '../actions/knots';
import { SearchIcon } from '../components/Icons';
import HeaderTitle from '../components/HeaderTitle';
import SearchBox from '../components/SearchBox';
// import AdBanner from '../components/AdMob';
import { knotPropType } from './Categories';
import knotPreviews from '../assets/knots';
import { results, search } from '../assets/staticLocalisation.json';

class Knots extends React.PureComponent {
  constructor(props) {
    super(props);

this.state = {
      isMounted: false,
      showSearchField: false,
      searchParam: '',
    };
  }

componentDidMount() {
this.setState({ isMounted: true });
    InteractionManager.runAfterInteractions(() => {
      const { navigation } = this.props;

navigation.setOptions({
        headerRight: () => (
<View style={{ marginRight: 15 }}>
<SearchIcon onPress={this.onSearch} />
</View>
```

```
        ),
      });
    });
  }

  onSearch = () => {
    const { navigation, langCode } = this.props;

this.setState({
      showSearchField: true,
    });
navigation.setOptions({
      headerTitle: () => (
<HeaderTitle title={results[`name_${langCode}`]}>
          {/* eslint-disable-next-line react/destructuring-assignment */}
          {this.state.showSearchField && (
<SearchBox
              changeSearchParam={this.changeSearchParam}
              endEditing={() => this.setState({ showSearchField: false })}
              placeholder={search[`name_${langCode}`]}
            />
          )}
</HeaderTitle>
      ),
    });
  };

  renderItem = ({ item }) => {
    const {
      navigation: { navigate },
      setKnot,
      langCode,
      isPortrait,
      dHeight,
      dWidth,
    } = this.props;
    const name = item[`knotenname_${langCode}`];
    const preview = knotPreviews.find(
({ knotennummer }) => knotennummer === +item.knotennummer
    );
    const { imagePreview } = preview;

    return (
<KnotCategory
        data={{
          name,
          image: imagePreview,
        }}
        onPress={() => {
navigate('Knot', {
            title: name.split('_')[0],
            reverse: item.favorite,
          });
          setKnot(item);
        }}
        isPortrait={isPortrait}
        height={dHeight}
        width={dWidth}
      />
    );
  };

  changeSearchParam = (searchParam) => {
```

```
this.setState({ searchParam });
  };

  filterKnots = () => {
    const { knots, langCode } = this.props;
    const { searchParam } = this.state;

    return knots.filter(
      (knot) => knot[`knotenname_${langCode}`].includes(searchParam) && knot
    );
  };

render() {
    const { isMounted } = this.state;

    if (!isMounted) {
      return null;
    }

    return (
<View style={styles.container}>
<OptimizedFlatList
          data={this.filterKnots()}
          legacyImplementation
          windowSize={15}
          initialNumToRender={10}
          keyExtractor={({ knotennummer }) => `${knotennummer}`}
          renderItem={this.renderItem}
        />
        {/* <AdBanner /> */}
</View>
    );
  }
}

const styles = StyleSheet.create({
  container: {
    flex: 1,
    backgroundColor: '#363636',
  },
  searchBox: {
    position: 'absolute',
    width: 200,
    backgroundColor: '#fff',
    left: '50%',
    transform: [{ translateX: -100 }, { scaleY: 1.4 }],
    zIndex: 100,
    paddingHorizontal: 4,
    paddingVertical: 2,
    fontSize: 20,
  },
});

Knots.propTypes = {
  knots: PropTypes.arrayOf(knotPropType).isRequired,
  navigation: PropTypes.shape({
    navigate: PropTypes.func.isRequired,
    setOptions: PropTypes.func.isRequired,
    setParams: PropTypes.func.isRequired,
  }).isRequired,
  route: PropTypes.shape({
    params: PropTypes.shape({
      title: PropTypes.string.isRequired,
```

```
    }).isRequired,
  }).isRequired,
  setKnot: PropTypes.func.isRequired,
  langCode: PropTypes.string.isRequired,
  isPortrait: PropTypes.bool.isRequired,
  dHeight: PropTypes.number.isRequired,
  dWidth: PropTypes.number.isRequired,
};

const mapStateToProps = ({
  knots: { filteredKnots },
  language: { langCode },
  dimensions: { isPortrait, height, width },
}) => ({
  knots: filteredKnots,
  langCode,
  isPortrait,
  dHeight: height,
  dWidth: width,
});

const mapDispatchToProps = (dispatch) => ({
  setKnot: (knot) => dispatch(knotSet(knot)),
});

export default connect(mapStateToProps, mapDispatchToProps)(Knots);
```

dimentions.js

```
import { Dimensions } from 'react-native';
import { CHANGE_WIDTH_HEIGHT } from '../constants/dimensions';

const getDimWidthHeight = () => {
  const { width, height } = Dimensions.get('window');
  const isPortrait = width < height;

  return { width, height, isPortrait };
};

const initialState = getDimWidthHeight();

export default (state = initialState, action) => {
  if (action.type === CHANGE_WIDTH_HEIGHT) {
    return getDimWidthHeight();
  }

  return state;
};
```

android/build.gradle

```
apply plugin: "com.android.application"

import com.android.build.OutputFile

project.ext.react = [
    entryFile: "index.js",
    enableHermes: false
]

apply from: '../../node_modules/react-native-unimodules/gradle.groovy'
apply from: "../../node_modules/react-native/react.gradle"

def enableSeparateBuildPerCPUArchitecture = false
```

```
def enableProguardInReleaseBuilds = false

android {
    compileSdkVersion rootProject.ext.compileSdkVersion
    buildToolsVersion rootProject.ext.buildToolsVersion

    defaultConfig {
        applicationId "com.knots3d"
        minSdkVersion rootProject.ext.minSdkVersion
        targetSdkVersion rootProject.ext.targetSdkVersion
        versionCode 7
        versionName "1.6"
    }
    splits {
        abi {
reset()
            enable enableSeparateBuildPerCPUArchitecture
            universalApk false  // If true, also generate a universal APK
            include "armeabi-v7a", "x86", "arm64-v8a", "x86_64"
        }
    }
    signingConfigs {
        release {
            storeFile file(RELEASE_STORE_FILE)
            storePassword RELEASE_STORE_PASSWORD
            keyAlias RELEASE_KEY_ALIAS
            keyPassword RELEASE_KEY_PASSWORD
        }
    }
    buildTypes {
        release {
            minifyEnabled enableProguardInReleaseBuilds
            proguardFiles getDefaultProguardFile("proguard-android.txt"),
"proguard-rules.pro"
            signingConfig signingConfigs.release
        }
    }
    // applicationVariants are e.g. debug, release
    applicationVariants.all { variant ->
variant.outputs.each { output ->
            // For each separate APK per architecture, set a unique version code
as described here:
            // <http://tools.android.com/tech-docs/new-build-system/user-
guide/apk-splits>
            def versionCodes = ["armeabi-v7a":1, "x86":2, "arm64-v8a": 3,
"x86_64": 4]
            def abi = output.getFilter(OutputFile.ABI)
            if (abi != null) {  // null for the universal-debug, universal-
release variants
output.versionCodeOverride =
                        versionCodes.get(abi) * 1048576 +
defaultConfig.versionCode
            }
        }
    }
    compileOptions {
        sourceCompatibility JavaVersion.VERSION_1_8
        targetCompatibility JavaVersion.VERSION_1_8
    }
}

def jscFlavor = 'org.webkit:android-jsc:+'
```

```
def enableHermes = project.ext.react.get("enableHermes", false);

dependencies {
    implementation fileTree(dir: "libs", include: ["*.jar"])
    implementation "com.android.support:appcompat-
v7:${rootProject.ext.supportLibVersion}"
    implementation "com.facebook.react:react-native:+"  // From node_modules
addUnimodulesDependencies()

    if (enableHermes) {
        def hermesPath = "../../node_modules/hermes-engine/android/";
        debugImplementation files(hermesPath + "hermes-debug.aar")
        releaseImplementation files(hermesPath + "hermes-release.aar")
    } else {
        implementation jscFlavor
    }
}

// Run this once to be able to run the application with BUCK
// puts all compile dependencies into folder libs for BUCK to use
task copyDownloadableDepsToLibs(type: Copy) {
    from configurations.compile
    into 'libs'
}

apply from: file("../../node_modules/@react-native-community/cli-platform-
android/native_modules.gradle");
applyNativeModulesAppBuildGradle(project)
```

ios/info.plist

```
<?xml version="1.0" encoding="UTF-8"?>
<!DOCTYPE plist PUBLIC "-//Apple//DTD PLIST 1.0//EN"
"<http://www.apple.com/DTDs/PropertyList-1.0.dtd>">
<plist version="1.0">
<dict>
        <key>CFBundleDevelopmentRegion</key>
        <string>en</string>
        <key>CFBundleDisplayName</key>
        <string>$(PRODUCT_NAME)</string>
        <key>CFBundleExecutable</key>
        <string>$(EXECUTABLE_NAME)</string>
        <key>CFBundleIdentifier</key>
        <string>$(PRODUCT_BUNDLE_IDENTIFIER)</string>
        <key>CFBundleInfoDictionaryVersion</key>
        <string>6.0</string>
        <key>CFBundleName</key>
        <string>$(PRODUCT_NAME)</string>
        <key>CFBundlePackageType</key>
        <string>APPL</string>
        <key>CFBundleShortVersionString</key>
        <string>$(MARKETING_VERSION)</string>
        <key>CFBundleSignature</key>
        <string>????</string>
        <key>CFBundleVersion</key>
        <string>$(CURRENT_PROJECT_VERSION)</string>
        <key>GADApplicationIdentifier</key>
        <string>ca-app-pub-8323348147242911~8682987108</string>
        <key>ITSAppUsesNonExemptEncryption</key>
        <false/>
        <key>LSRequiresIPhoneOS</key>
        <true/>
        <key>NSAppTransportSecurity</key>
```

```
	<dict>
		<key>NSAllowsArbitraryLoads</key>
		<true/>
		<key>NSExceptionDomains</key>
		<dict>
			<key>localhost</key>
			<dict>
				<key>NSExceptionAllowsInsecureHTTPLoads</key>
				<true/>
			</dict>
		</dict>
	</dict>
	<key>NSMotionUsageDescription</key>
	<string>Allow Knots 3D to access your device's accelerometer</string>
	<key>UILaunchStoryboardName</key>
	<string>LaunchScreen</string>
	<key>UIRequiredDeviceCapabilities</key>
	<array>
		<string>armv7</string>
	</array>
	<key>UISupportedInterfaceOrientations</key>
	<array>
		<string>UIInterfaceOrientationPortrait</string>
		<string>UIInterfaceOrientationLandscapeLeft</string>
		<string>UIInterfaceOrientationLandscapeRight</string>
		<string>UIInterfaceOrientationPortraitUpsideDown</string>
	</array>
	<key>UIViewControllerBasedStatusBarAppearance</key>
	<false/>
</dict>
</plist>
```