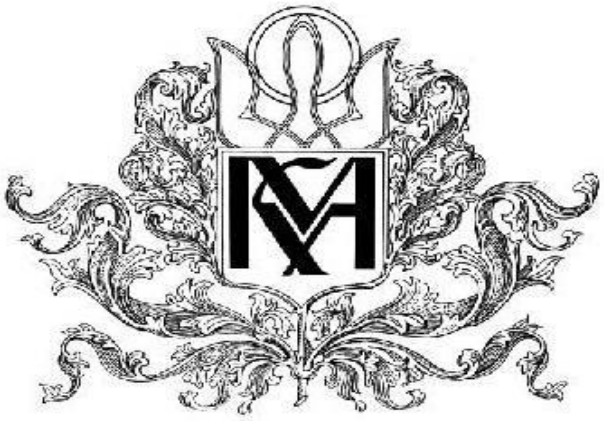
Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра Інформатики Факультету Інформатики



Розробка мобільного застосунку для управління фінансами

Керівник курсової роботи

Калітовський Б.В.

(підпис)

“____” _____

2021 р.

Виконав студент

Шевченко К. О.

“____” _____

2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ

Зав.кафедри мультимедійних систем,

доц., д.ф.-м.н.

_____О. П. Жежерун

„____”_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ

на курсову роботу

студенту Шевченку Кирилу Олександровичу 3-го курсу факультету інформатики

ТЕМА Розробка мобільного застосунку для управління фінансами

Зміст ТЧ до курсової роботи:

Індивідуальне завдання

Вступ

1 Аналіз предметної області

2 Огляд обраних технологій

3 Структура програми

Висновки

Список літератури

Додатки (за необхідністю)

Дата видачі „___” _____ 2021 р. Керівник Калітовський Б.В.

Завдання отримав _____

Тема: Розробка мобільного застосунку для управління фінансами

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	02.11.2020	
2.	Огляд технічної літератури за темою роботи.	25.11.2020	
3.	Знайомство та аналіз предметної області	Грудень 2020-Січень 2021	
3.	Проектування та розробка додатку	Лютий 2021-Квітень 2021	
4.	Написання текстової частини роботи	Квітень 2021	
5.	Створення презентації та доповіді	07.05.2021-10.05.2021	
6.	Захист курсової роботи	Травень 2021	

Студент Шевченко Кирило Олександрович

Керівник Калітовський Б.В.

“_____”_____

ЗМІСТ

Анотація

Курсова робота має на меті створення мобільного додатку для управління фінансами для iOS та Android. За основу розробки взято та розглянуто TypeScript, фреймворк React native, IDE PhPCode, допоміжні бібліотеки та Android Emulator. Проаналізовано декілька аналогічних програм та низку шаблонів проектування для застосунків на React Native. Реалізовано такі основні функції додатку : додавання чеку (витрати) до системи, список усіх доданих транзакцій, розрахунок витрат за певний період часу, відслідковування витрат за категоріями, панель налаштувань додатку.

Ключові слова: iOS, Android, React Native, TypeScript, Container/Component, управління фінансами, витрати, транзакції, мобільний додаток.

Вступ

Сьогодні, люди не можуть уявити своє життя без використання сучасних смартфонів. Мобільний пристрій став для людини справжнім помічником, замінивши багато різних пристроїв, тим самим включивши їх в себе. Це, і фотоапарат, і відеокамера, і музичний програвач, і мапа, і , звичайно, найзручніший спосіб зв’язку.

Можна наводити дуже багато прикладів того, що саме сучасний смартфон пропонує користувачу. Телефон – універсальний пристрій, який може бути як сховищем інформації, так і карткою, за допомогою якої можна здійснювати оплату чеків та рахунків в магазинах, ресторанах та кафе.

Ця курсова робота присвячена створенню додатку, який спростить ще одну проблему, яка дуже часто турбує сучасну людину – відслідковуванню витрат та доходів. Додаток на тему «Управління фінансами» допоможе у аналізі витрат за певний період часу, або ж за заданою категорією, що є дуже корисним задля кращого розподілу фінансів користувача. Створений додаток повинен бути простим у користуванні, мати приємний дизайн , та чітко та точно обраховувати усі дані, так як це дуже важливий аспект для користувачів.

Розділ 1 : Аналіз предметної області

1.1 Дослідження аналогів

Було проаналізовано декілька схожих додатків, які користуються попитом серед користувачів, під назвами «1money» та «Money Manager Expense & Budget» відповідно. Ці застосунки доступні у вільному доступі на платформі Google Play. В результаті аналізу додатків та відгуків користувачів щодо них, я дійшов висновку, що такі програми є майже ідеальними, проте навіть в них знайшлись деякі недоліки.

Під час аналізу даних програм все ж таки було виявлено такі проблеми:

- Недоліки у швидкодії інтерфейсу;
- Відсутність бажаних користувачу валют;
- Багато реклами у безкоштовній версії;
- Помилки при використанні унікальних для даної програми функцій;
- Некоректна робота Google authentication services;
- Незрозумілий інтерфейс;
- Дані додатки прив’язані лише до однієї платформи;
- Неефективні платні версії;
- Лімітована кількість дій для користувачів з безкоштовним обліковим записом;

Основними перевагами цих застосунків є:

- Багатий функціонал;
- Приємний інтерфейс;
- Можливість підлаштовувати під себе категорії та підкатегорії;

- Швидка конвертація різних валют;
- Можливість зберігання фінансів у різних валютах;
- Синхронізація даних;
- Наглядний показ транзакцій на графіку;
- Облік доходів та витрат;
- Фільтрація транзакцій по підкатегоріям та категоріям;
- Вибір дати транзакції;
- Доступ до акаунту з ПК (лише у Money Manager);
- Платна версія, яка дозволяє прибрати рекламу;

1.2 Характеристика предметної області

Отже, за основу береться створення додатку, який буде допомагати користувачу зберігати та оновлювати свої фінансові кроки. За допомогою такого додатку можна відслідковувати та аналізувати свої надходження до бюджету, або ж , навпаки , витрати, за певний період часу, що є дуже корисним для сучасної людини. Користувач, у даному додатку, може додавати як витрати, так і доходи, вибираючи при цьому бажану валюту.

В додатку повинні бути майже всі валюти світу, що зробить додаток функціональним для багатьох країн та надасть користувачу можливість конвертації. Також додаток має містити в собі панель вибору категорій. В кожній категорії є свої підкатегорії, які можна модифікувати : видаляти, редагувати та додавати нові. Повинно бути присутнім вікно вибору дати, а також додавання коментаря щодо витрати або ж доходу. Для ведення обліку необхідна панель, або ж сторінка, на якій буде доступна уся інформація про кожну з транзакцій

Також повинна бути сторінка, де користувач зможе аналізувати свої доходи та витрати за певний період часу, маючи при цьому перед собою таблицю/графік/зображення для кращого розуміння.

Додаток повинен бути простим у використанні, який легко підійде різним поколінням людей. Після аналізу аналогів та дослідження предметної області було визначено основні функціональні вимоги, які становлять базу для таких додатків.

1.3 Функціональні вимоги

Розглянемо основні функціональні вимоги, які були визначені після детального аналізу предметної області:

- Можливість вибирати категорії ;
- Можливість змінювати періоди відслідковування;
- Фільтрація витрат, або доходів за категорією;
- Вибір різних валют;
- Можливість робити замітки до транзакції;
- Вибирати дату відтворення певної транзакції;
- Адаптація додатку під обидві операційні системи : iOS та Android;
- Можливість додавати свої підкатегорії;
- Додавання акаунту з іншою основною валютою;
- Зміна теми додатку;
- Можливість переглядати інформацію про вже створені транзакції;
- Зберігання інформації у базі даних;
- Зрозумілий та швидкий інтерфейс;
- Конвертація валют;
- Можливість зміни назви категорії;

1.4 Висновок до розділу 1

У цьому розділі було проаналізовано аналоги, визначено основні функції для додатку, та головні проблеми таких застосунків. Можна зробити висновок, що такі програми хоч і є досить популярними та мають хороший функціонал, проте у деяких користувачів трапляються різні проблеми під час користування даними додатками, наприклад, дехто вважає такі застосунки незрозумілими у плані інтерфейсу, або ж багато з функцій просто їм непотрібні, що спонукає користувачів на видалення таких застосунків. То ж основною ціллю є створення простого та зручного в користуванні додатку, який буде працювати без підключення до мережі інтернет на обох зазначених операційних системах.

Розділ 2 : Огляд обраних технологій

2.1 Мова програмування JavaScript

JavaScript - це динамічна мова комп'ютерного програмування. Вона легка і найчастіше використовується для програмування веб-сторінок, реалізація яких дозволяє взаємодіяти з користувачем та створювати динамічні сторінки. Це інтерпретована мова програмування з об’єктно-орієнтованими можливостями.

Клієнтський JavaScript - найпоширеніша форма мови. Скрипт повинен бути включений у документ HTML або посилатися на нього, щоб код інтерпретувався браузером.

Це означає, що веб-сторінка не повинна бути статичним HTML, але може включати програми, які взаємодіють з користувачем, керують браузером та динамічно створюють вміст HTML.

JavaScript можна використовувати для захоплення ініційованих користувачем подій, таких як клацання кнопок, навігація посиланнями та інші дії, які користувач ініціює явно або неявно.

2.2 Середовище програмування

JetBrains PhpStorm —крос-платформове середовище розробки , яке розробляється компанією JetBrains і доступне як у безкоштовній версії ,так і в платній.

PhpStorm глибоко аналізує структуру коду і дійсно розуміє ваш код, підтримуючи всі можливості мови PHP як в нових, так і в legacy-проектах.

Редактор підтримує автодоповнення коду і рефакторинг, запобігає помилки на льоту. [5]

У PhpStorm ви можете працювати з найсучаснішими технологіями: HTML 5, CSS, Sass, Less, Stylus, CoffeeScript, TypeScript, Emmet і JavaScript. При цьому будуть доступні рефакторинг, налагодження та юніт-тестування. Завдяки функції Live Edit всі зміни можна тут же подивитися в браузері.

Одноманітні завдання зручно виконувати прямо в PhpStorm. IDE інтегрована з системами контролю версій, підтримує віддалене розгортання, бази даних і SQL, інструменти командного рядка, Docker, Composer, REST-клієнт і багато інших інструментів. Також PhpStorm включає в себе всю функціональність WebStorm, а також повноцінну підтримку PHP, баз даних і SQL.

PhpStorm подбає про якість вашого коду за допомогою сотень інспекцій, які перевіряють код на льоту і аналізують весь проект цілком. Підтримка PHPDoc, code (re) arranger, інструмент форматування, швидкі виправлення і інші можливості допомагають розробникам писати акуратний код, який легко підтримувати.

2.3 React Native framework

React Native - це платформа мобільних додатків з відкритим вихідним кодом, створена Facebook, Inc. Вона використовується для розробки додатків для Android, Android TV, iOS, macOS, tvOS, Web, Windows і UWP, дозволяючи розробникам використовувати структуру React разом з власної платформою. [1]

React Native працює у фоновому процесі (який інтерпретує написаний розробниками JavaScript) безпосередньо на кінцевому пристрої та взаємодіє з

власною платформою через серіалізовані дані через асинхронний та пакетний міст.

Компоненти React обгортають наявний Native код та взаємодіють із власними API через декларативну UI парадигму React та JavaScript. Це дозволяє розробляти Native програми для нових команд розробників і дозволяє існуючим Native командам працювати набагато швидше.

Завдяки потужності JavaScript, React Native дозволяє виконувати ітерації з блискавичною швидкістю. Більше не потрібно чекати закінчення власних збірок. Можна відслідковувати всі зміни у реальному часі.[3]

Хоча стиль React Native має подібний синтаксис до CSS, він не використовує HTML або CSS. Натомість повідомлення з потоку JavaScript використовуються для маніпулювання власними поданнями. React Native також дозволяє розробникам писати власний код такими мовами, як Java або Kotlin для Android та Objective-C або Swift для iOS, що робить його ще більш гнучким. React Native пропонує величезний вибір бібліотек та модулів, що спрощує написання коду, які дозволяють розробнику використовувати багатий вибір різних віджетів, анімацій та інших програмних рішень, які підходять як для Android, так і для iOS.

2.4 TypeScript

TypeScript - це мова з відкритим кодом, яка базується на JavaScript, одному з найбільш використовуваних у світі інструментів, додаючи визначення статичного типу. Типи дають спосіб описати форму об’єкта, забезпечуючи кращу документацію та дозволяючи TypeScript перевірити, чи правильно працює ваш код. [6]

Код TypeScript перетворюється в код JavaScript за допомогою компілятора TypeScript або Babel. Цей JavaScript - це чистий, простий код, який працює в будь-якому місці, де працює JavaScript: у браузері, на Node.JS або у програмах програмістів.

Багато текстових редакторів і середовищів розробки, наприклад, Visual Code Studio, Atom, Sublime, Visual Studio, Netbeans, WebStorm і інші, мають підтримку TypeScript на рівні плагінів, що дозволяє скористатися низкою переваг. TypeScript підтримує статичну типізацію, що дозволяє програмісту легше відлагоджувати програмний код, також можливе підключення різних модулів та бібліотек, підтримка повноцінних класів, що передує гарним принципам об’єктно орієнованого програмування.

2.5 Node package manager

Node package manager (npm) – пакетний менеджер, написаний на Javascript, який працює на платформі Node.js. Такий менеджер являє собою базу даних публічних та приватних пакунків які дозволяють розробнику спростити написання коду та використовувати прогресивні рішення у своєму додатку.

Для початку користування npm необхідно встановити Node.js. Node package manager має змогу встановлювати як локальні залежності для певного проекту, так і глобально інсталювати програмні інструменти.

Npm записує усі залежності у спеціальний JSON файл під назвою package.json. В цьому файлі знаходиться інформація про встановлений модуль та його версію а також інформація про проект. Такий файл дозволяє використовувати проект на різних машинах. Також npm генерує файл package-lock.json, який містить в собі дерево залежностей кожного з модулів. Для того, щоб запустити проект на іншому пристрої, необхідно запустити лише одну програмну команду – npm

install. Автоматично npm створить папку під назвою «node_modules» та завантажить туди усі необхідні модулі та пакунки. Node package manager є дуже корисним інструментом для розробників веб- та мобільних додатків, так як завдяки одній-двум командам можна встановити бажаний пакунок та продовжити роботу з ним.

2.6 Висновок до розділу 2

У цьому розділі було розглянуто низку програмних рішень для успішного програмування мобільних застосунків. Обрані технології для програмування мобільного додатку є одними з найкращих та найпопулярніших. Вони є зручними в користуванні, зрозумілими для програміста та пропонують багато різних рішень, які орієнтовані саме на мобільні пристрої та операційні системи iOS та Android. Завдяки React Native, можна, як було зазначено вище, програмувати додатки у режимі реального часу, зберігаючи купу часу для програміста, що робить цей фреймворк однією з найпопулярніших технологій на даний момент.

Розділ 3 : Структура програмного застосунку

3.1 Проектування та архітектура додатку

Головною метою було створення такого додатку, який би легко використовувався людьми різного віку на різних операційних системах. Тому треба було спроектувати легкий для розуміння та швидкий інтерфейс та реалізувати найнеобхідніші функції які потрібні для «Фінансового менеджера».

Для початку створення такого продукту, необхідно подбати про етап його проектування. Дуже важливо цей етап не пропускати, так як від нього залежить успішність застосунку : чи буде його складно розширювати, чи легко підтримувати додаток в робочому стані, чи зручно програмний код зрозуміти іншим програмістам.

Недотримання певних норм при проектуванні додатку може призвести до непоправних наслідків, таких як повна незрозумілість коду, дуже негативний вплив на кінцеву якість додатку. Саме тому етап проектування програмного засобу є дуже важливим – він робить розробку додатку набагато ефективнішою, програмний код читабельним для інших програмістів, а сам додаток гнучким та якісним.

Саме тому, в першу чергу я обирав шаблон проектування для додатку – правила та норми при розробці архітектури нашої програми. При проектуванні даного застосунку я обрав шаблон для React застосунків під назвою «Container/Component».

3.2 Опис шаблону проектування

На сьогоднішній день існує дуже багато різних шаблонів проектування які дуже спрощують життя програмістам. При створенні додатку на React Native я обрав шаблон проектування Container/Component , який є одним з найпоширеніших при розробці мобільних та веб- додатків.

Пишучи код у React або React Native, ми часто задаємось питанням, як структурувати наш код, щоб він значно полегшив нам життя при обробці змін стану, потоку даних та рендерингу тощо. Існує шаблон, який допомагає в організації React-додатків - розбиття компонентів в презентаційні та контейнери.

Презентаційні компоненти - це ті компоненти, єдиною роботою яких є рендерінг візуального представлення відповідно до стилю та переданих їм даних. По суті, вони не містять жодної бізнес логіки. Ось чому їх іноді ще називають «німими» компонентами (Dumb components). Це означає, що вони не мають прямого доступу до Redux чи інших сховищ даних. Дані передаються їм через властивості елементів. [11]

Презентаційні компоненти:

1. Піклуються про візуальне представлення
2. Мають власну розмітку та стилі.
3. Не мають залежностей від решти компонентів додатку, таких як Redux сховища.
4. Не вказують, як саме дані завантажуються або мутуються.
5. Отримують дані та виключно через властивості.
6. Рідко мають свій власний стан (Але коли вони мають, то це стан інтерфейсу).

Компоненти контейнера - це ті компоненти React, які мають доступ до сховища даних. Ці компоненти здійснюють виклики API, виконують обробку та містять бізнес-логіку програми. Компоненти контейнера не повинні мати логіку представлення в вигляді інтерфейсу або логіку презентації. Завдання компонентів контейнера полягає в обчисленні значень та передачі їх як реквізиту презентаційним компонентам. Ці компоненти іноді також називають розумними компонентами (Smart components).

Отже, компоненти контейнера:

- Відповідають за роботу різних компонентів;
- зазвичай не мають власної розмітки, крім деяких обгортаючих подань, і ніколи не мають жодних стилів;
- надають дані та поведінку презентаційним або іншим компонентам контейнера;
- Викликають Redux actions та надають їх як зворотні виклики презентаційним компонентам;
- Часто мають свій стан, оскільки вони, як правило, служать джерелами даних;
- зазвичай генеруються з використанням компонентів вищого порядку, таких як connect () від React Redux, createContainer () від Relay або Container.create () від Flux Utils, а не написані від руки;

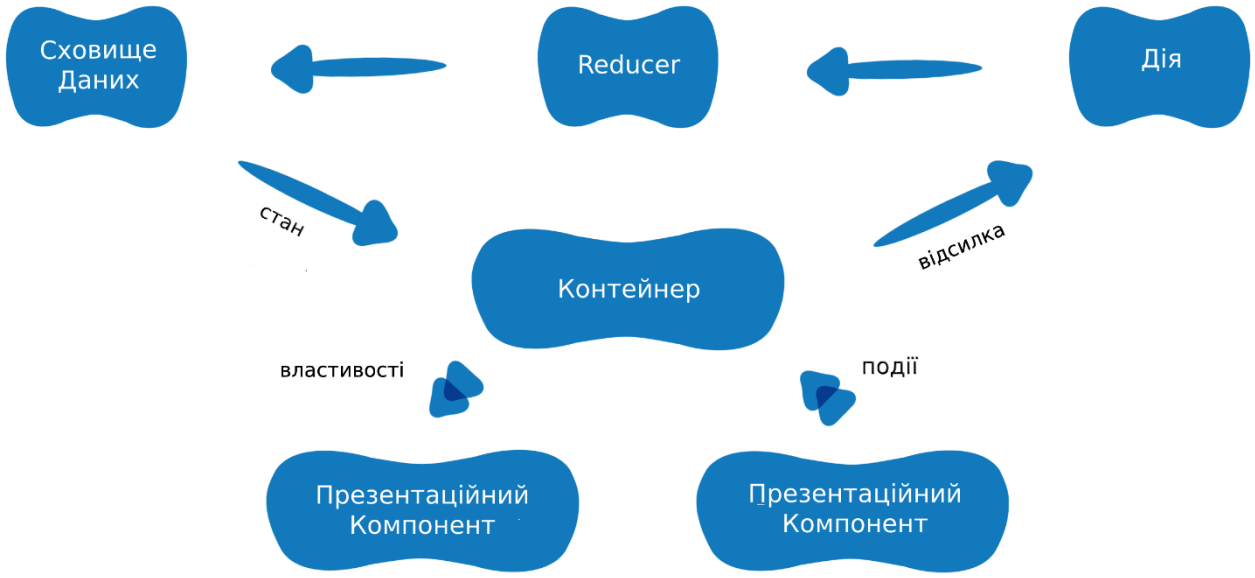
Компонент контейнера не виконує жодних реалізацій інтерфейсу чи стилів. Він лише керує бізнес-логікою. Це допомагає нам відокремити проблеми, що стосуються дизайну / макетування та Бізнес-логіки.

Container/Component шаблон дає нам багато переваг:

1. Менше дублювання коду. Оскільки ви змушені перемістити всі компоненти макета як окремі презентаційні компоненти, тепер ви

можете використовувати їх повторно, замість того, щоб копіювати код на кожній сторінці.

2. Презентаційні компоненти - це, по суті, рівень перегляду вашої програми. Отже, ви можете змінити стиль, не торкаючись логіки програми.
3. Краще розділення проблем. Ви краще розумієте свою програму та свій інтерфейс, написавши компоненти таким чином.
4. Краще багаторазове використання. Ви можете використовувати один і той же презентаційний компонент із абсолютно різними джерелами стану, і перетворити їх на окремі компоненти контейнера, які можна буде використовувати повторно.



3.3 Опис інтерфейсних рішень

Графічний інтерфейс є однією з найважливіших частин мобільного додатку. Правильно побудований інтерфейс є запорукою успішності програмного продукту, так як він є першим, на що користувач звертає увагу.

Під час проектування я намагався створити простий, зрозумілий та водночас лаконічний інтерфейс, який в змозі опанувати користувачі будь-якого віку на будь-якому девайсі.

При першому запуску програми користувач бачить перед собою екран який «вітає» користувача (див. рис. 3.3.1). В цьому вікні присутній девіз додатку, інструкція щодо подальших дій та невеличке фото.

Рисунок 3.3.1 Екран-привітання

Натиснувши на кнопку «Далі» користувач переходить у вікно вибору основної валюти(див. рис. 3.3.2). За замовчуванням основною валютою є гривня, але при

натиснувши на центральну кнопку відкривається модальне вікно, в якому користувач має можливість змінити валюту (див.рис.3.3.3).

Рисунок 3.3.2 Екран вибору валюти

Рисунок 3.3.3 Модальне вікно доступних валют

Після успішного вибору валюти та натисненні на кнопку «Далі» користувач переходить на екран, на якому додаток сповіщає про успішність його дій та базові інструкції для користування (див. рис. 3.3.4)

Рисунок 3.3.4 Екран зі сповіщенням

Після натиску на кнопку «Далі» перед користувачем з’являється основний екран, на якому якраз і створюються транзакції. В застосунку доступні дві теми : світла та темна (див.рис. 3.3.5). На основному екрані зображено 8 категорій, панель для введення суми транзакції, загальна сума транзакції, біля якої розташований код вибраної валюти.

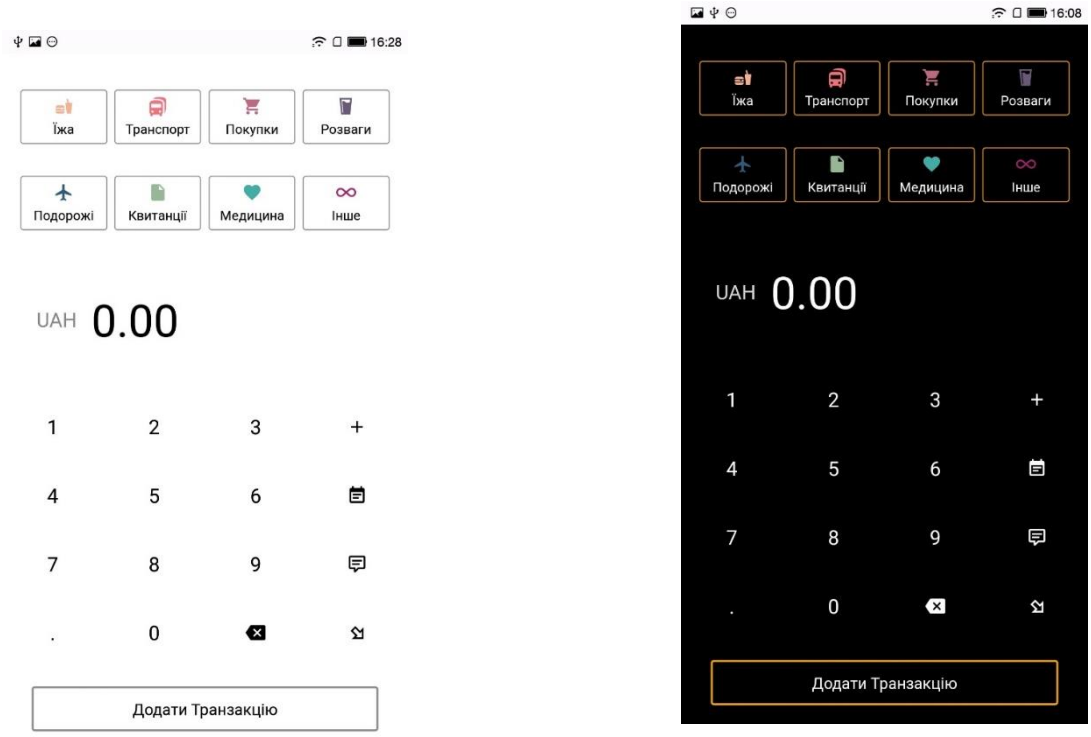


Рисунок 3.3.5 Основний екран (у світлій та темній темах)

На правій стороні панелі для вводу суми розташовані додаткові модальні вікна: вибір дати транзакції (див.рис. 3.3.6), додаткова інформація про транзакцію (коментар) та вибір типу транзакції(дохід чи витрата) (див.рис. 3.3.7)

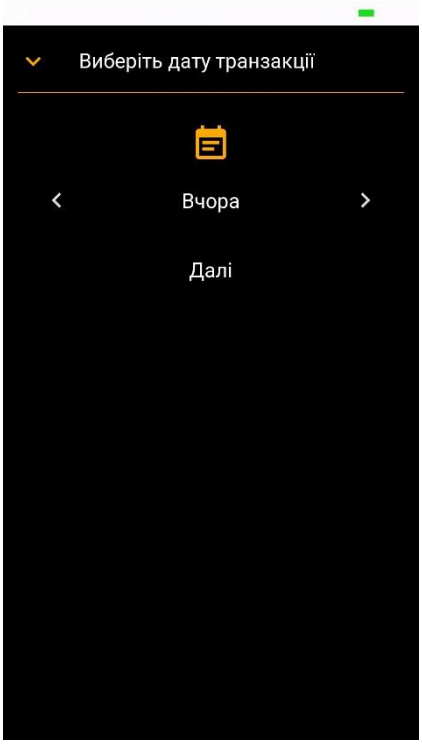


Рисунок 3.3.6 Вибір дати транзакції

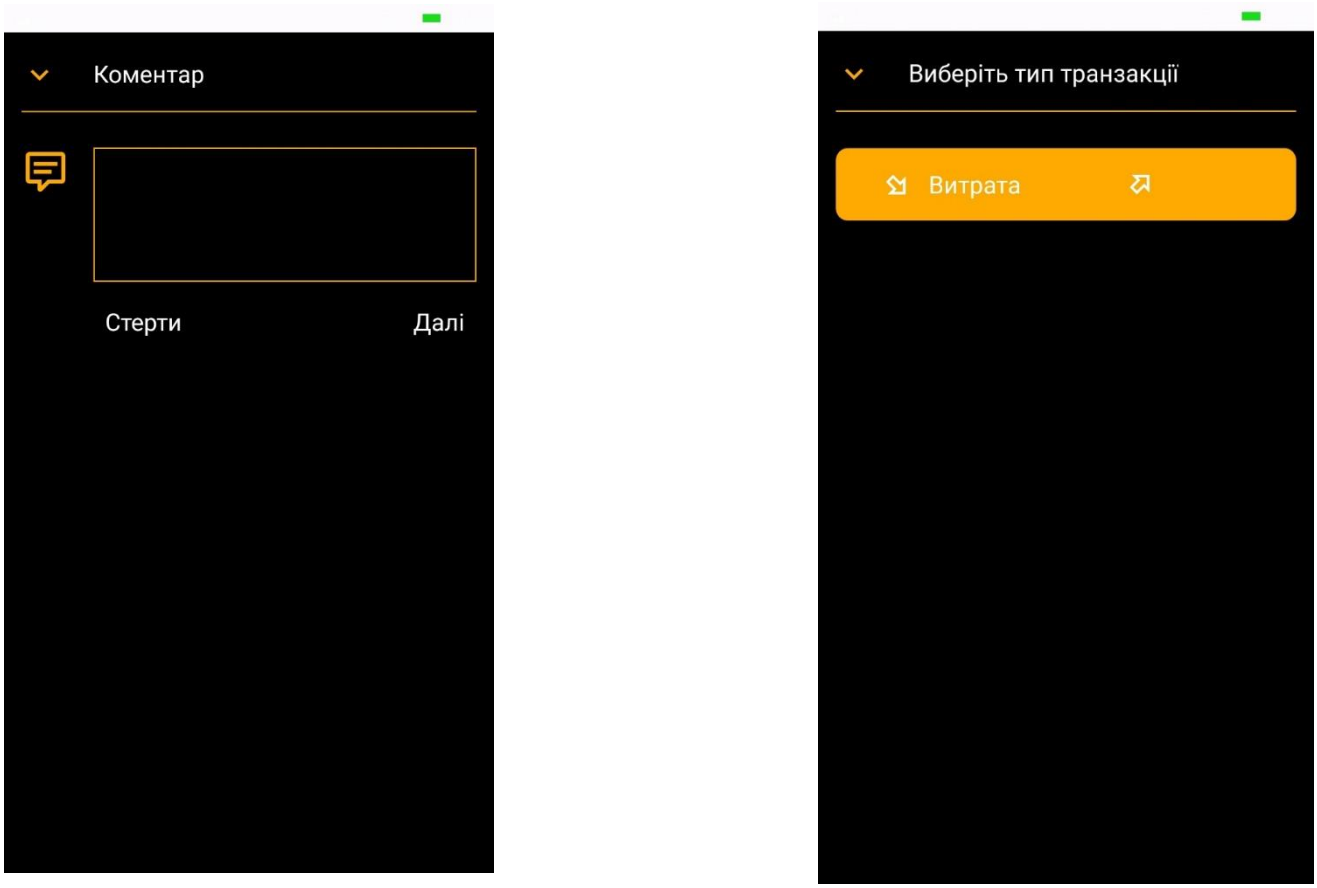


Рисунок 3.3.7 Модульні вікна «Коментар» та «Тип транзакції»

В залежності від типу транзакції у користувача будуть з’являтися різні категорії з різними підкатегоріями(див.рис.3.3.8).

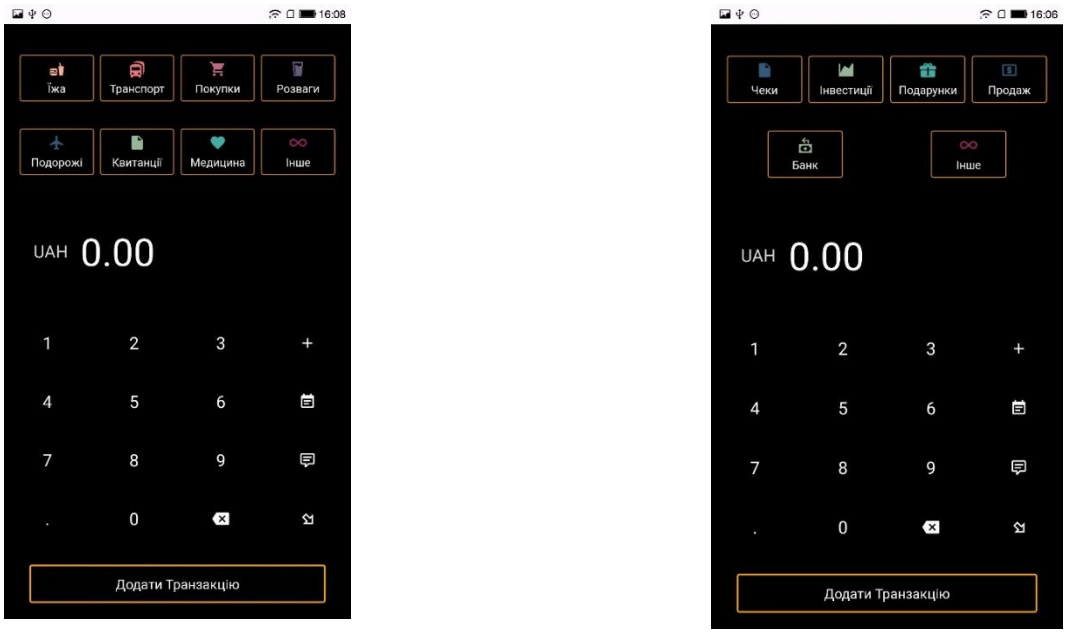
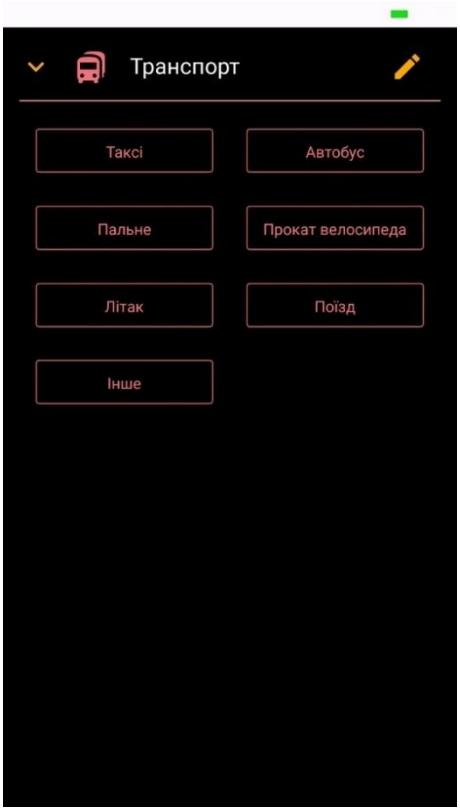


Рисунок 3.3.8 Типи панелі вибору категорій

При натисненні на певну категорію користувач перейде на вікно категорії, в якій міститься інформація про підкатегорії, а, також можливість редагувати назву категорії та додавати/видаляти підкатегорії при натисненні на іконку олівця(див.рис. 3.3.9). Варто зазначити, що без вибору категорії, користувач не зможе оформити запис транзакції. Також у користувача є можливість конвертації валют, для цього потрібно натиснути на код валюти біля загальної суми транзакції, та у модальному вікні вибрати бажану валюту для конвертації, перед користувачем модальне вікно зі списком валют(див.рис. 3.3.3). Функція конвертації доступна лише при підключенні девайсу до інтернету, так як конвертація виконується при підключенні до віддаленого серверу за допомогою GET запиту. Після конвертації користувач має право у спеціальному вікні погодитися або відмовитися на конвертацію(див.рис. 3.3.10).



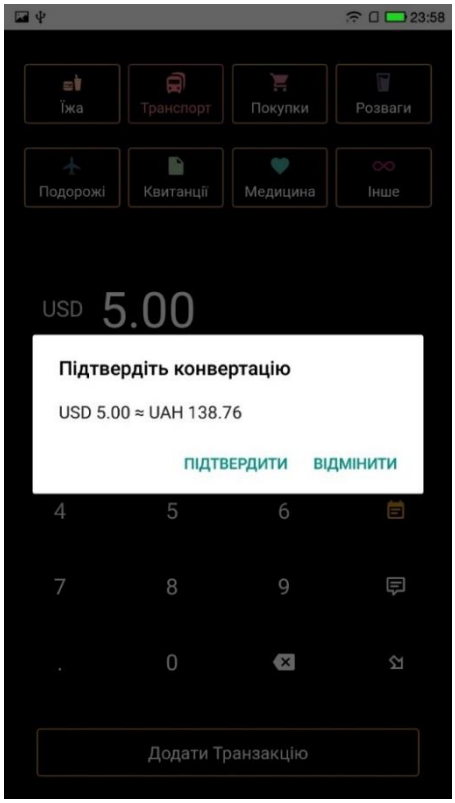


Рисунок 3.3.9 Екран категорії «Транспорт» Рисунок 3.3.10 Вікно підтвердження

При натисканні користувачем на іконку «+», він перейде на екран з обрахунком транзакцій, можна сказати, що це простий калькулятор(див.рис.3.3.11). На цій

панелі доступні такі функції як додавання, віднімання, множення та ділення, тобто найпростіші операції для обрахування доходів або ж витрат.

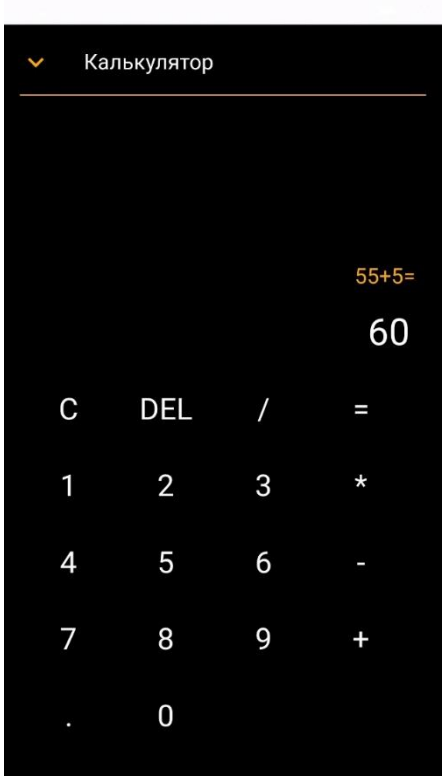


Рисунок 3.3.11 Калькулятор

При проведенні пальцем вліво, користувач перейде на екран з інформацією про всі транзакції(див.рис 3.3.12).

Витрати в темній темі позначаються білим кольором, доходи помаранчевим з символом «+» на початку, маючи при цьому вказані місця для відображення.

При натисненні на певну транзакцію користувач побачить перед собою модальне вікно з короткою інформацією про неї : категорія, підкатегорія, валюта та сума, коментар, якщо наявний.

Якщо користувач додав коментар до транзакції, то після назви підкатегорії перед ним з’явиться символ (К).

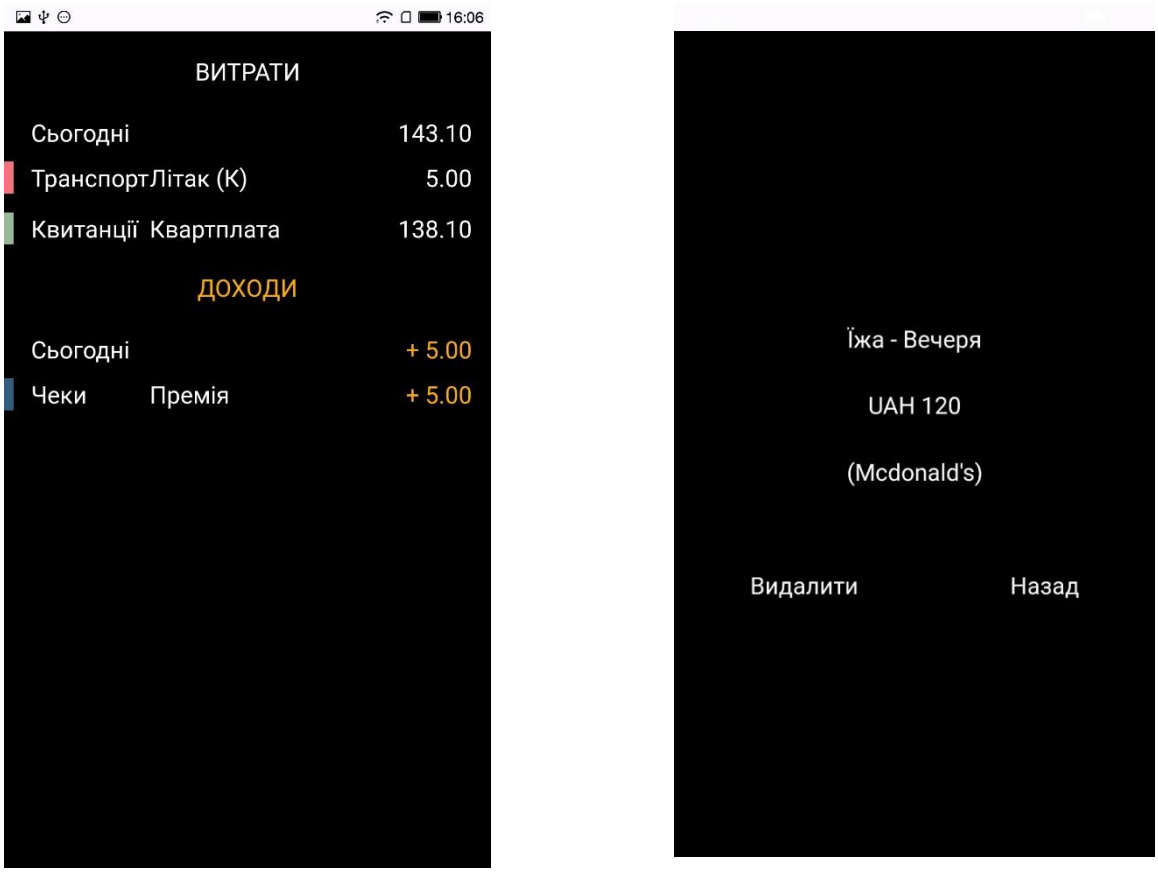


Рисунок 3.3.12 Екран транзакцій та вікно з інформацією про одну з них

При проведенні пальцем вліво користувач побачить перед собою екран з аналізом транзакцій за вибраний проміжок часу : день, тиждень, місяць та рік (див.рис. 3.3.13). На даному екрані присутня фільтрація по категоріям при натисненні на одну з них.

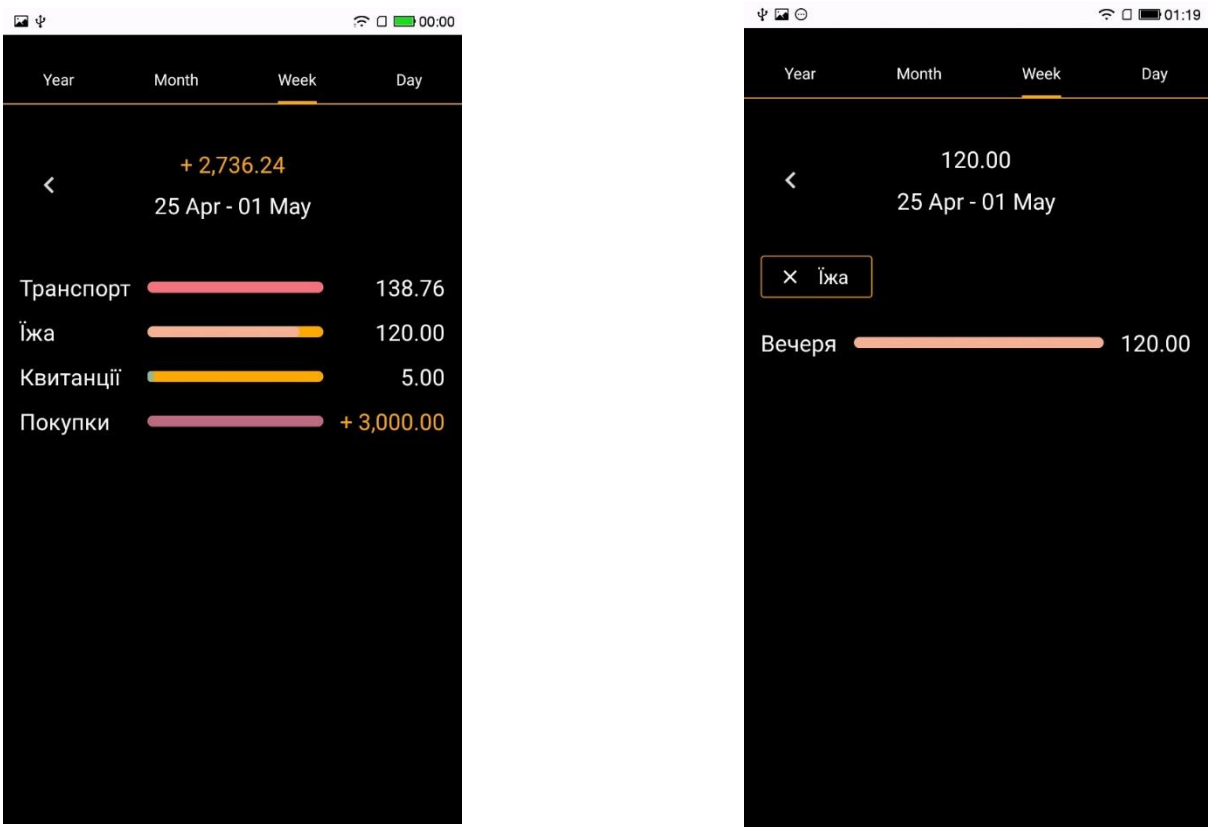


Рисунок 3.3.13 Екран зі статистикою та фільтрація за категорією

Далі користувач може так само провести пальцем вліво та побачити перед собою екран з налаштуваннями (див.рис.3.3.14). Користувач може скористатися такими функціями: вибір теми, реєстрація або видалення валютного акаунту. При реєстрації перед користувачем з’явиться модальне вікно як на рисунку 3.3.3.

Рисунок 3.3.14 Екран налаштувань

3.4 Оптимізація розміру розробленого додатку

Після розробки програмного застосунку було здійснено оптимізацію розміру додатку, його розмір було зменшено з 79MB до 60MB. Для цього було застосовано декілька рішень : стиснення Java Bytecode, розділення архітектури

armebi та x86 на два apk файли, заміна усіх картинок та іконок на векторні зображення, видалення деяких бібліотек з “node_modules”.

3.5 Висновок до розділу 3

У цьому розділі було описано архітектуру, шаблон проектування та графічний інтерфейс розробленого застосунку. При розробці цього додатку я дізнався багато нових ефективних рішень задля успішного проектування архітектури застосунку, а, також, рішень для візуального представлення.

Висновки

Результатом виконаної курсової роботи стала мобільна програма, яка дозволяє відслідковувати дії зі своїми фінансами. Завдяки React Native такий додаток є мультиплатформенним, тобто ним можна користуватися на iOS та Android. В даному застосунку реалізовані усі основні заплановані функції задля успішного користування.

В ході написання курсової було досліджено та опановано мною нові і ефективні підходи та методи розробки мобільних додатків. При створенні даного додатку , я опанував такі технології як фреймворк React Native,покращив свої знання з мови Typescript, попрацював з емуляторами систем iOS та Android.

Було створено зручний інтерфейс з гарною швидкодією який має усі функції для зберігання інформації про транзакції та їх аналізу. В додатку реалізовані усі базові функції для успішного використання.

Такий інтерфейс мають змогу опанувати користувачі різного віку, так як він є простим у розумінні, і, завдяки кросплатформенності, ним можуть користуватися користувачі різних смартфонів з вищезгаданими операційними системами. Також було застосовано певні дії задля зменшення ваги додатку.

Список використаної літератури

1. React Native overview [Електронний ресурс]. https://reactnative.dev/
2. React Native documentation [Електронний ресурс]. https://reactnative.dev/docs/getting-started
3. Akshat Paul, Abhishek Nalwaya, React Native for IOS development. – Apress,2016.
4. Habr – React Native [Електронний ресурс]. https://habr.com/ru/post/323214/
5. PhpStorm [Електронний ресурс]. https://www.jetbrains.com/ru-ru/phpstorm/
6. TypeScript overview [Електронний ресурс]. https://www.typescriptlang.org/
7. TypeScript documentation [Електронний ресурс]. https://www.typescriptlang.org/docs/
8. Persisting data in React Native [Електронний ресурс]. https://pusher.com/tutorials/persisting-data-react-native
9. Money Manager Expense & Budget [Електронний ресурс]. https://play.google.com/store/apps/details?id=com.realbyteapps.moneymanagerfree&hl=ru&gl=US
10. 1Money Google Play Market [Електронний ресурс]. https://play.google.com/store/apps/details?id=org.pixelrush.moneyiq&hl=ru&gl=US
11. React native Container/component pattern [Електронний ресурс]. https://www.reactnative.guide/9-redux/9.2-presentational-vs-containers.html
12. Modal view in React Native [Електронний ресурс]. https://reactnative.dev/docs/modal

13.React patterns [Електронний ресурс].

https://reactpatterns.com/#container-component

14. Using react native with TypeScript [Електронний ресурс].

https://habr.com/ru/company/otus/blog/456124/