МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра програмної інженерії

КВАЛІФІКАЦІЙНА РОБОТА МАГІСТРА

на тему: **«РОЗРОБКА МОБІЛЬНОГО ДОДАТКУ КЕРУВАННЯ ФІНАНСАМИ ЗАСОБАМИ EXPRESS JS ТА REACT NATIVE»**

Виконав: студент 2 курсу, групи 8.1211-1іпз

спеціальності 121 інженерія програмного забезпечення
(шифр і назва спеціальності)

освітньої програми інженерія програмного забезпечення
(назва освітньої програми)

Ю.О. Луцик
(ініціали та прізвище)

Керівник доцент кафедри програмної інженерії, доцент, к.ф.-м.н. Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Рецензент доцент кафедри фундаментальної та прикладної математики, доцент, к.ф.-м.н. Панасенко Є.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя – 2022

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра програмної інженерії

Рівень вищої освіти магістр

Спеціальність 121 інженерія програмного забезпечення

Освітня програма інженерія програмного забезпечення

ЗАТВЕРДЖУЮ

Завідувач кафедри програмної інженерії, к.ф.-м.н., доцент

_____ Лісняк А.О.

(підпис)

« ____ » _____ 2022 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Луцику Юрію Олександровичу

(прізвище, ім'я та по–батькові)

1. Тема роботи Розробка мобільного додатку керування фінансами засобами Express JS та React Native

керівник роботи Кудін Олексій Володимирович, к.ф.-м.н., доцент

(прізвище, ім'я та по–батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 04 » травня 2022 р. № 500-с

2. Строк подання студентом роботи 01.12.2022

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково–пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі, аналіз предметної області.
2. Проєктування програмного доповнення.
3. Реалізація та тестування програмного доповнення.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) _____
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 04.05.2022

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Розробка плану виконання кваліфікаційної роботи магістра.	01.06.2022	
2.	Збір вихідних даних та аналіз предметної області.	08.07.2022	
3.	Обробка методичних та теоретичних джерел.	12.08.2022	
4.	Специфікація вимог до системи. Робота над першим розділом.	21.09.2022	
5.	Проєктування системи. Робота над другим розділом.	09.10.2022	
6.	Реалізація та тестування системи. Робота над третім розділом.	13.11.2022	
7.	Оформлення та нормоконтроль кваліфікаційної роботи магістра.	01.12.2022	
8.	Захист кваліфікаційної роботи магістра.	15.12.2022	

Студент _____ Ю.О. Луцик
(підпис) (ініціали та прізвище)

Керівник роботи _____ О.В. Кудін
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ А.В. Столярова
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота магістра “Розробка мобільного додатку керування фінансами засобами Express JS та React Native”: 68 с., 20 рис., 13 джерел, 2 додатки.

EXPRESS JS, NODE JS, REACT NATIVE, REDUX UML.

Об’єкт дослідження – процес розробки мобільного додатку керування фінансами, засобами “React Native” та “Express JS”.

Мета роботи – розробити мобільний додаток керування фінансами за допомогою “React Native” та “Express JS”.

Методи дослідження – моделювання, проєктування, програмний, аналітичний.

У наш час неможливо уявити будь-яку розвинену торгівлю без Інтернет-ресурсів чи мобільних застосунків. Зрештою, торгівля через мобільні застосунки дуже важлива. Зараз існує велика кількість служб та додатків, які допомагають у створенні мобільних додатків. Одним із прикладів таких ресурсів є фреймворк “React Native”, за допомогою якого було створено мобільний додаток «FreeShop» та серверною частиною “Express JS”.

Таким чином, за результатами роботи створено зручний та ефективний мобільний додаток продукту програми FreeShop з однією з найпоширеніших серверних систем “Express JS”.

SUMMARY

Master's qualifying paper “Development of the Finance Management Mobile Application using Express JS and React Native”: 68 pages, 20 figures, 13 references, 2 supplements.

EXPRESS JS, NODE JS, REACT NATIVE, REDUX UML.

The object of the research is the process of developing a mobile financial management application using React Native and Express JS tools.

The goal of the work is to develop a mobile finance management application using React Native and Express JS.

Research methods are modeling, design, software, and analytical.

Nowadays, it is impossible to imagine any developed trade without Internet resources or mobile applications. After all, mobile commerce is very important. There are now a large number of services and applications that help in creating mobile applications. One example of such resources is the “React Native” framework, which was used to create the “FreeShop” mobile application and the “Express JS” backend.

Thus, based on the results of the work, a convenient and effective mobile product application of the FreeShop program was created with one of the most common server systems “Express JS”.

ЗМІСТ

ВСТУП

У наш час створено безліч мобільних додатків, котрі пропонують клієнту різноманітні товари. Адже так клієнту буде простіше ознайомитися з направленням вашого бізнесу та потрібним списком послуг чи товарів. Для таких сервісів були створені допоміжні ресурси та спеціалізовані бібліотеки, котрі значно полегшують створення мобільних додатків та надають значно потужніші можливості. Сьогодні кількість таких ресурсів дуже велика, тому розробник має можливість вибрати для себе найзручніший. На мою думку, найбільш розвиненим та зручним на сьогодні є “React Native” для розробки кросс-платформних додатків. Прикладом мобільного додатку з цим фреймворком буде виступати фінансовий додаток FreeShop та серверною частиною на “Express JS”.

Великий функціонал фреймворка дозволяє легко виконувати різноманітний функціонал. Його поєднання з іншими технологіями дозволяє виконувати задачі в найшвидші терміни, що в свою чергу знижує вартість створення програм.

Актуальність дослідження: тема кваліфікаційної роботи зумовлена популярністю та простотою фреймворка “React Native” та “Express JS”.

Об’єкт дослідження – процес розробки мобільного додатку, засобами “React Native” та “Express JS”.

З огляду на це, можна виділити головні аспекти розробки програмного продукту фінансового додатку.

Мета – розробити мобільний додаток FreeShop з серверною частиною на “Express JS”.

Предмет дослідження: створення мобільного додатку, що дозволяє знаходити знижки на товари.

Методи дослідження: моделювання, аналітичний, проєктування.

У першому розділі демонструється аналіз та пошук, збір інформації фреймворка, ознайомлення “React Native”, “Express JS”.

У другому розділі розробляються план проєктування програми, та виконується розбір основних класів, які будуть задіяні в системі.

У третьому розділі демонструється розробка мобільного додатку, після успішної розробки програма тестується і створюється керівництво по користуванню.

1 ТЕХНІЧНЕ ЗАВДАННЯ

1.1 Терміни та визначення

Система – мобільний додаток на основі “React Native”.

Node JS – програмна платформа для переведення JavaScript коду в мову спільного призначення.

Express JS – фреймворк для Node JS.

Redux – бібліотека, для управління станом програми.

Користувач – людина, яка заінтересована в покупці товару.

Адміністратор – користувач, який має права на редагування програми.

БД – база даних, місце збереження інформації.

Кешбек – різновид бонусної програми, для залучення клієнтів.

Фреймворк – програмна платформа, яка визначає структуру програмної системи.

1.2 Функціональні вимоги

Функціональне призначення системи – реалізувати мобільний додаток для здійснення пошуку потрібних знижок за допомогою фреймворка “React Native”.

Цілі розробки системи – створення та реалізація мобільного додатку для пошуку знижок за допомогою фреймворка “React Native”.

Учасники системи: системою користується клієнт, адміністратор.

Система має надавати адміністраторам такі можливості:

– додавання/редагування/перегляд/видалення знижок;

– додавання/редагування/перегляд/видалення категорій знижок.

Система має надавати клієнтам такі можливості:

– вибір знижки;

– пошук знижки.

Для ефективного та зручного використання системи програма має надавати адміністраторам системи можливість конфігурації, а саме:

– зміна назви знижки;

– включення/вимикання режиму розробника;

– вибір валюти;

– збереження налаштувань.

1.3 Опис предметної області

Предметною областю є розробка мобільного додатку на основі фреймворка “React Native”. Дана система повинна забезпечити можливість розповсюдження та використання знижок в інтернеті.

Даний мобільний додаток побудований та спроєктований таким чином, щоб власник магазину за певним умов, якщо кваліфікація його магазину зміниться, то досить швидко можна перекваліфікувати сайт. Процес запуску та подальшої роботи має наступний вигляд:

– адміністратор магазину після повного аналізу предметної області повинен скласти зробити запити до найбільш популярних представників товарів;

– після того, як буду ухвалені всі умови, адміністратор додає всі знижки, а потім їх згрупувати;

– адміністратор додає всі потрібні знижки до додатку через панель адміністратора;

– після попередніх кроків додаток готовий до експлуатації;

– коли клієнт заходить до додатку виникає запрос на отримання даних, які згодом він отримує з бази даних;

– після того, як клієнт зацікавився певним товаром, він зберегти, або поділитися купоном на знижку, і потім його використати;

– потім клієнт при покупці зацікавленого товару може використати свій купон на знижку як у реальному магазині, так і онлайн додатку.

1.4 Опис системи

Онлайн торгівля стає невід'ємною складовою діяльністю сучасного бізнесу будь-якої організації. У зв'язку з цим велику актуальність набуває розробка систем, які дозволяють автоматизувати процеси компаній та окремих осіб, пов'язаних з торгівлею. Також досить важливою частиною є маркетинг, який і сприяє створенню додатків із переліком знижок

Фінансовий додаток являє собою спеціалізований мобільний додаток, який належить фірмі-посереднику, яка співпрацює з торгівельними фірмами, тощо та призначений для просування споживчих товарів на ринку, збільшення обсягів продажу, залучення нових покупців [1].

Характерними рисами кешбек додатків є те, що вони можуть активно залучати клієнтів до покупки товарів, а саме шляхом інформування клієнта про найактуальніші знижки, наприклад в чорну п'ятницю. Клієнту не потрібно шукати інформацію про різні товари окремо в інтернеті, він може зробити це в цьому додатку. Також завдяки використанню Internet-технологій є можливою персоналізація підходу до споживачів з урахуванням попередніх використань знижок та використання мобільного додатку як ефективного способу маркетингових досліджень (анкетування, конференції покупців і т.п.) [2].

Користування мобільними застосунками серед користувачів мобільних пристроїв стає все більш і більш популярним. Згідно з даними дослідження компанії AppAnnie за 2017 рік кількість завантажень застосунків зросла на

60 %, споживчі витрати зросли більш ніж удвічі, а час, витрачений на застосунок кожним користувачем складає близько 43 днів на рік

Компанія з дослідження ринку Gartner передбачила, що протягом 2013 року приблизна кількість завантажених мобільних застосунків складатиме 102 мільярди (91 % з яких будуть безплатними), що своєю чергою складатиме 26 мільярдів доларів США (а це на 44,4 % більше, у порівнянні з 18 мільярдами доларів США за 2012 рік). Згідно з даними, до 2 кварталу 2015 року, самі лише магазини Google Play і Apple Store зібрали 5 мільярдів доларів [3].

Мобільні додатки потребують зовсім не великих витрат на утримання та організацію роботи, завдяки швидкій розробці та широкому впливу, порівняно з тим, як би відбувалося розповсюдження знижок та інформування клієнта у реальних магазинах, оскільки у ньому значно обмеженіша матеріально-технічна база (будівлі, споруди, приміщення) та кількість обслуговуючого персоналу.

Проте мобільні додатки мають і недоліки. Основними є невизначеність реального існування товару та відповідність його основним параметрам якості, шахрайства при проведенні грошових транзакцій, проблеми з розбіжністю певних характеристик товарів [4].

Основними вимогами, які ставляться користувачами до мобільного додатку є:

– зрозумілий інтерфейс та зручна система навігації по додатку;

– зручна система посилань, що дозволяє оптимальним способом одержати необхідну користувачеві інформацію;

– мінімальна кількість дій користувача для здійснення покупки.

1.5 Програми аналоги

В наш час ринок фінансових додатків розвивається з шаленою швидкість. Що є тільки на користь користувачу, адже він може обрати найбільш підходяще для нього. Найбільш популярні додатки для українського ринку:

– Letyshops;

– Payback.

Розгнянемо детальніше кожен з них.

Letyshops – це міжнародний сервіс кешбеку, який працює з 2014 року. Він співпрацює з сотнями українських та міжнародних магазинів, які пропонують товари та послуги у десятках категорій. Крім звичайного кешбеку, портал надає бонуси за залучення друзів, а також збільшує відсоток коштів, що повертаються в рамках програми лояльності. Тісно співпрацюючи з багатьма великими компаніями, Letyshops дає великий кешбек у деяких торгових точках. Іноді він перевищує 20 %. Гроші, отримані з покупок, можна виводити на банківські картки та мобільні рахунки. Мінімальна сума для зняття складає 250 гривень, а максимальна – 7000. Кошти можна переказати на благодійні цілі.

Payback – український кешбек-сервіс, заснований у 2016 році. Серед його партнерів близько сотні місцевих та міжнародних компаній, серед яких AliExpress, ЄВА, MOYO, Banggood та ін. Відсоток коштів, що повертаються, тут трохи нижче, але програма лояльності теж присутня. Вона дозволяє збільшувати кешбек у межах накопичувальної системи. Чим більша загальна сума угод – тим вище і збільшення. Користуватися Payback.ua можна, переходячи до магазину для покупок із сайту сервісу або за допомогою фірмових розширень для браузерів. Вони доступні для Opera, Chrome, Firefox та Яндекс-браузера. Виведення коштів підтримується на карти банків, мобільні номери та Webmoney, але нижнього ліміту для зняття компанія не називає.

2 ПРОЄКТУВАННЯ МОБІЛЬНОГО ДОДАТКУ

2.1 Використання React та його можливості

React – це бібліотека для створення інтерфейсів користувача. React не є фреймворком – він навіть не розрахований виключно для web. Він використовується для візуалізації і в зв'язці з іншими бібліотеками. Наприклад, React Native можна використовувати для створення мобільних додатків, React можна використовувати для створення додатків віртуальної реальності, крім того є й інші варіанти [8].

Основна мета React – мінімізувати помилки, що виникають при розробці інтерфейсів користувача. Це досягається за рахунок використання компонентів – автономних логічних фрагментів коду, які описують частина призначеного для користувача інтерфейсу. А вже ці компоненти об'єднуються для створення повноцінного інтерфейсів користувача. React абстрагує більшу частину роботи по візуалізації, залишаючи вам можливість зосередитися на дизайні [8].

React не зобов'язує до суворих правил щодо угод про код або організації файлів. Це дозволяє командам домовлятися, що для них більш підходить, і структурувати React проєкт відповідним чином. React може відповідати за одну кнопку, кілька частин або ж весь призначений для користувача інтерфейс програми [9].

Крім того, такі переваги React-додатки, як написання інтерфейсів за допомогою JSX, вимагають процесу компіляції. Додавання на сайт компілятора Babel призводить до більш повільного виконання коду, тому такі інструменти зазвичай настроюються для процесу складання. Так, можливо, у React є серйозні вимоги до інструментарію, але його можна освоїти за досить не великий період часу.

Основні складові частини React:

– компоненти React;
– рендеринг ReactDOM;
– класові компоненти і функціональні компоненті;
– JSX;
– стан (state);
– обробка подій;
– асинхроний метод setState;
– параметри (props);
– посилання (refs).

Компонент це одна ізольована частина інтерфейсу. Для прикладу в звичайній головній сторінці блогу ви можете виявити компонент сайдбара і список компонентів для постів в блозі. Всі вони зібрані з компонентів, так що ви б мали список компонентів, кожен для кожного поста в блозі і кожен зі своїми специфічними властивостями [10].

Переваги:

– дуже швидка робота, завдяки реалізації React Virtual DOM і різним оптимізаціям рендеринга;
– підтримка рендеринга на стороні сервера, що робить його потужною платформою для контент-орієнтованих додатків;
– React реалізує концепції функціонального програмування, створюючи простий в тестуванні і багаторазово використовуваний код.

Недоліки:

– міксування шаблонів з логікою (JSX) може стати незрозумілим для деяких розробників React;
– React відходить від компонентів на основі класів, що може стати перешкодою для розробників, яким більш комфортно працювати з об'єктно-орієнтованим програмуванням (ООП).

2.2 Порівняння нативнив та крос–платформних мобільних програм

Для розробки сучасних мобільних додатків існує безліч способів. Фундаментальна відмінність між нативною та кросплатформною розробкою залежить від того, для якої операційної системи ви розробляєте. Нативна мобільна розробка дозволяє створювати програми для певної операційної системи – Android або iOS. На відміну від цього, кросплатформна розробка мобільних пристроїв дозволяє створювати програми для кількох операційних систем.

Нативні мобільні програми – це програми, розроблені для Android або iOS. Залежно від того, для якої операційної системи ви розробляєте, ваші програми зазвичай кодуються певною мовою програмування. Ви пишете рідні програми для Android за допомогою мов програмування Java або Kotlin. Для розробки нативних програм для iOS, ви використовуєте мови програмування Objective-C або Swift.

Плюси нативних мобільних додатків:

– краща продуктивність: ви створюєте та оптимізуєте власні програми для певної платформи. Це робить їх дуже швидкими, ефективнішими та чуйними на дії користувачів;

– висока безпека: нативні програми можуть підвищити безпеку даних вашого користувача. Вони мають доступ до вбудованих функцій безпеки для конкретної платформи;

– доступ до повного набору функцій: нативні мобільні програми можуть отримати доступ до всіх функцій і можливостей пристрою, забезпечуючи більш багатий і інтегрований досвід користувача.

Мінуси нативних мобільних додатків:

– висока собівартість: розробка нативних мобільних додатків зазвичай дорога, тому що необхідно окремі команди розробників для конкретних платформ;

– час розробки - окремим командам, які працюють над схожими програмами для різних платформ, потрібен час.

Відомі приклади нативних мобільних програм: Google Maps, Spotify, WhatsApp.

Відмінністю крос-платформних мобільних програм від нативних є те, що для різних платформ використовується одна кодова база. Метою кросплатформної розробки додатків є націлювання на різні операційні системи в одному проекті. Такі додатки створюються за допомогою міжплатформних фреймворків, які використовують SDK для певної платформи (SDK для Android та iOS SDK) з єдиного API. Це дає вам змогу легко отримати доступ до SDK і бібліотек різних платформ. Найбільш популярними крос-платформними фреймворками зараз є React Native та Flutter.

Плюси крос-платформних мобільних додатків:

– Низька собівартість - крос-платформна мобільна розробка потребує лише однієї команди розробників. Вони повинні бути обізнаними в обраній системі.

– Перевикористання коду - крос-платформні фреймворки дозволяють використовувати єдину кодову базу. Це забезпечує послідовність використовуваної бізнес-логіки.

– Швидка розробка - повторно використовуючи код і покращуючи продуктивність, розробка стає дуже швидкою.

Мінуси крос-платформних мобільних додатків:

– Складна інтеграція - кросплатформні фреймворки не інтегрують усі особливості платформи. Для деяких апаратних інтеграцій, як-от використання графічних процесорів, можуть знадобитися навички розробки нативних програм.

– Нижча продуктивність - крос-платформні фреймворки зазвичай включають спеціальне середовище виконання для запуску вашої програми.

Він відповідає за зв’язок із спеціальними службами платформи. Це додає ще один рівень обчислень, що призводить до зниження продуктивності.

Відомі приклади крос-платформних мобільних програм:

– Instagram, Skype, Walmart, Airbnb (React Native);

– Google Ads, My BMW App, eBay Motors, the New York Times (Flutter).

2.3 Використання React Native, його можливості, та порівняння з аналогами

React Native - це фреймворк, який дозволяє розробникам створювати власні мобільні програми за допомогою JS. Це дає змогу створити програму для кількох платформ, використовуючи ту саму кодову базу. Це може скоротити час розробки вашого проекту та знизити загальну вартість створення мобільного додатку. Він був розроблений Facebook і отримав велику популярність. Створити мобільну програму React Native, яка підтримує iOS і Android, приблизно на 30 % швидше, ніж розробляти нативні програми окремо. Ще одна перевага полягає в тому, що вам не потрібні дві окремі команди для нативної розробки, а скоріше послуги однієї команди React Native.

Розробка React Native чудово підходить для більшості програм, які значною мірою залежать від інтерфейсу користувача, тому що, не докладаючи зусиль, можливо змусити інтерфейс користувача працювати як на iOS, так і на Android. І, головне, є можливість поділитися бізнес-логікою. Крім цих переваг, React Native використовує flexbox для макета, який працює однаково на iOS, Android і в Інтернеті, тож ми можемо передавати свій досвід з Інтернету замість того, щоб вивчати нові механізми. З іншого боку, нативна програма чудова, коли ми розглядаємо можливість використання всіх функцій, які пропонує операційна система, включаючи такі модулі, як

обробка відео/аудіо або багатопотоковість. Оскільки React Native зосереджується лише на інтерфейсі користувача, він може бути менш ефективним для програм із багатьма нативними функціями.

2022 рік став ще одним хорошим роком для користувачів, оскільки він приніс React Native 0.68 із підтримкою нової архітектури React Native! Вирішальним моментом нової архітектури є впровадження нової системи візуалізації – Fabric, яка покращує взаємодію між хост-платформами та логікою React. Він рендерить логіку в C++, а ядро C++ спільно використовується між платформами, що дозволяє React Native синхронно рендерити інтерфейси React.

До цього зв’язок між JavaScript і нативним потоком був реалізований за допомогою моста, який надсилає дані лише після серіалізації як JSON. Цей ефект в асинхронному процесі. Ще одна важлива зміна в архітектурі New React Native - це нова нативна система TurboModule System - удосконалення Native Modules.Ця функція значно впливає на час запуску програми React Native, оскільки дозволяє коду JavaScript завантажувати модулі лише тоді, коли вони потрібні. У попередній моделі всі модулі ініціалізувалися під час запуску, що збільшувало час завантаження.

Нова архітектура має багато переваг, наприклад:

– перевірка типів забезпечується на JavaScript і хост-платформах;

– підвищений взаємозв’язок завдяки рендерингу, реалізованому на C++;

– краща сумісність платформ хостингу завдяки новій системі візуалізації;

– покращене отримання даних завдяки інтеграції з React Suspense;

– швидший запуск програми завдяки TurboModules.

Одним із головних аналогів React Native є Flutter. Flutter - це фреймворк з відкритим вихідним кодом від Google для створення красивих, нативно скомпільованих мультиплатформенних додатків із єдиної кодової бази.

Модель розвитку базується на об’єктах. У Flutter кожен об’єкт – кнопка чи шрифт – є віджетом. Віджети є основою концепції Flutter – їх використання дуже схоже на будівництво з блоків, і вони дозволяють легко створювати красиві програми. Мова, що лежить в основі Flutter, схожа на C++ і Java, зокрема в тому, як вона компілює вихідний код. Dart доступний навіть для початківців завдяки своїй недивній об’єктній орієнтації та синтаксису.

У той же час команда розробників програми Flutter отримує доступ до найнижчого рівня коду, де вони можуть створювати нові віджети, налаштовані відповідно до потреб. Розробники роблять це за допомогою тих самих інструментів, які використовувала команда Flutter для створення стандартних віджетів.

Flutter і React Native мають багато спільного. Перш за все, обидва вони є фреймворками з відкритим кодом, якими може користуватися кожен. Обидва вони розроблені провідними компаніями: Google створив Flutter, а React Native - Facebook. Щоб порівняти основну інформацію про Flutter і React Native, ви можете перевірити цю інформацію за допомогою NIX.

Flutter легше використовувати, оскільки він більш стійкий до оновлень системи. Це означає, що коли iOS або Android оновлять ОС, програма залишиться незмінною. Навпаки, React Native залежить від нативних елементів, тому після випуску оновлення можуть з’явитися деякі проблеми в програмі запуску. Наразі Flutter підтримує Android 4.1+ або iOS 8+, тоді як React Native підтримує Android 4.1+ та iOS 10+.

Найбільша різниця між React Native і Flutter полягає в тому, що у Flutter є плагіни, розроблені командою Google, як-от геолокація та відображення. Великою перевагою React Native є велика аудиторія розробників, так як база кода з React може бути частково перевикористана.

2.4 Принцип роботи кешбеку

Кешбек-сервіси в Україні мають наступний економічний принцип роботи:

– компанія оформляє угоду з інтернет-магазинами про те, що вони будуть ділитися відсотком із прибутку, принесеного кожним залученим клієнтом;

– клієнти здійснюють покупки в магазині, переходячи в нього через сайт або додаток кешбек-сервісу, і це фіксується у даних про замовлення;

– магазин фіксує кількість угод, скоєних за посередництва партнера, і переводить йому відрахування, що належать за договором, з отриманого за рахунок цього прибутку;

– кешбек-сервіс залишає частку з партнерських відрахувань собі як заробіток, а основну частину перекладає покупцям.

Подібна модель співробітництва є вигідною для всіх сторін. Кешбек-сервіс отримує прибуток у вигляді частки від партнерських відрахувань, магазин нарощує дохід рахунок збільшення кількості клієнтів і товарообігу, а покупець повертає частину витрачених раніше коштів.

Як правило, нарахування бонусів провадиться не відразу після покупки. Оскільки закон встановлює 14-денний термін повернення невикористаного товару, виплата здійснюється після закінчення цього (чи тривалішого) періоду.

2.5 Використання UML під час розробки модуля

Для розробки будь-якого серйозного додатку завжди використовуються різноманітні діаграми тому, що чим більше проєкт, тим складніше зрозуміти та розробляти всі аспекти.

Для розробки різних фінансових додатків використовуються різні методології. Моделі дозволяють нам більш детально зрозуміти структуру і роботу системи. Однією з найбільш поширених мов моделювання є мова моделювання UML (Unified Modeling Language), призначена для моделювання, уявлення, проєктування та документування програмних систем, організаційно-економічних систем, технічних систем та інших систем різної природи. Візуальні моделі дозволяють спростити розуміння між замовником та розробником [13].

Мова UML широко застосовується для моделювання бізнес-процесів, системного проєктування та відображення організаційних структур. UML дозволяє розробникам програмного забезпечення досягти угоди в графічних позначеннях для представлення загальних понять (таких як клас, компонент, узагальнення (generalization), об’єднання (aggregation) і поведінку) і більше сконцентруватися на проєктуванні та архітектури. Він спрощує складний процес проєктування ПО шляхом створення «креслення» для побудови системи. Мова UML не прив’язаний до якої-небудь окремої платформі або мови програмування, тому він добре підходить для з’єднання мереж різних систем. Він розроблявся з урахуванням гнучкості і тому здатний адаптуватися до виникаючих нових проблем.

UML містить наступний набір діаграм:

– діаграма класів – для моделювання структури класів системи і зв’язків між ними в системі;

– діаграми компонентів – для моделювання ієрархії всих компонентів в системі;

– діаграми розміщення – для моделювання фізичної архітектури системи;

– діаграми варіантів використання – для моделювання функціональних вимог до системи між користувачами;

– діаграми станів – для моделювання поведінки об’єкта в системі при зміні стану;

2.6 Діаграма класів

Діаграма класів використовується для опису типів об’єктів системи і різного роду статичні відносини, які існують між собою. Ці діаграми відображають також властивості класів, операції класів та обмеження, які накладаються на зв’язки між об’єктами. В цих діаграмах функціональність описує властивості і операції.

Діаграма класів представляє собою певний граф, вершинами якого є елементи типу “класифікатор”, які пов'язані різними типами структурних відносин. Ця діаграма може також містити інтерфейси, пакети, відносини і навіть окремі екземпляри, такі як об'єкти і зв'язки. Діаграма показує статичну структурну модель проєктованої системи. Тому діаграму класів прийнято вважати графічним представленням таких структурних взаємозв'язків логічної моделі системи, що не залежать.

Переваги діаграми класів:

– діаграма класів ілюструє моделі даних навіть для дуже складних інформаційних систем;

– забезпечує огляд того, як додаток структуровано перед створенням фактичного коду. Це може легко скоротити час обслуговування;

– допомагає краще зрозуміти загальні схеми програми;

– дозволяє малювати докладні діаграми, які виділяють код, необхідний для програмування.

Діаграма класів складається з елементів, які в сукупності відображають загальне бачення предметної області. Окремі компоненти цієї діаграми можуть утворювати пакети для представлення більш загальної моделі системи. Якщо діаграма класів є частиною деякого пакета, то її компоненти повинні відповідати елементам цього пакета, включаючи можливі посилання на елементи з інших пакетів.

Клас даних діаграмах служить для позначення безлічі об'єктів, які мають однакову структуру, поведінку і відносини з об'єктами з інших класів.

Клас може не мати екземплярів або об'єктів. У цьому випадку він називається абстрактним класом. Відносини між класами:

– залежність;
– асоціація;
– узагальнення.

Залежність означає відношення між двома або більше класами, в якому зміна одного може викликати зміни іншого. Це завжди буде створювати більш слабкі відносини. Залежність вказує, що один клас залежить від іншого.

Асоціація представляє статичний тип відносин між класами

Узагальнення допомагає зв'язати підклас з його суперкласом. Підклас успадковується від свого суперкласу. Ставлення узагальнення можна використовувати для моделювання реалізації інтерфейсу. Діаграма класів дозволяє успадковувати від декількох суперкласів.

Абстрактний клас – це клас з прототипом операції, але не з реалізацією. Також можливо мати абстрактний клас без будь-яких операцій, оголошених всередині нього. Анотація корисна для визначення функціональних можливостей класів. Нижче наводиться діаграма Класів.

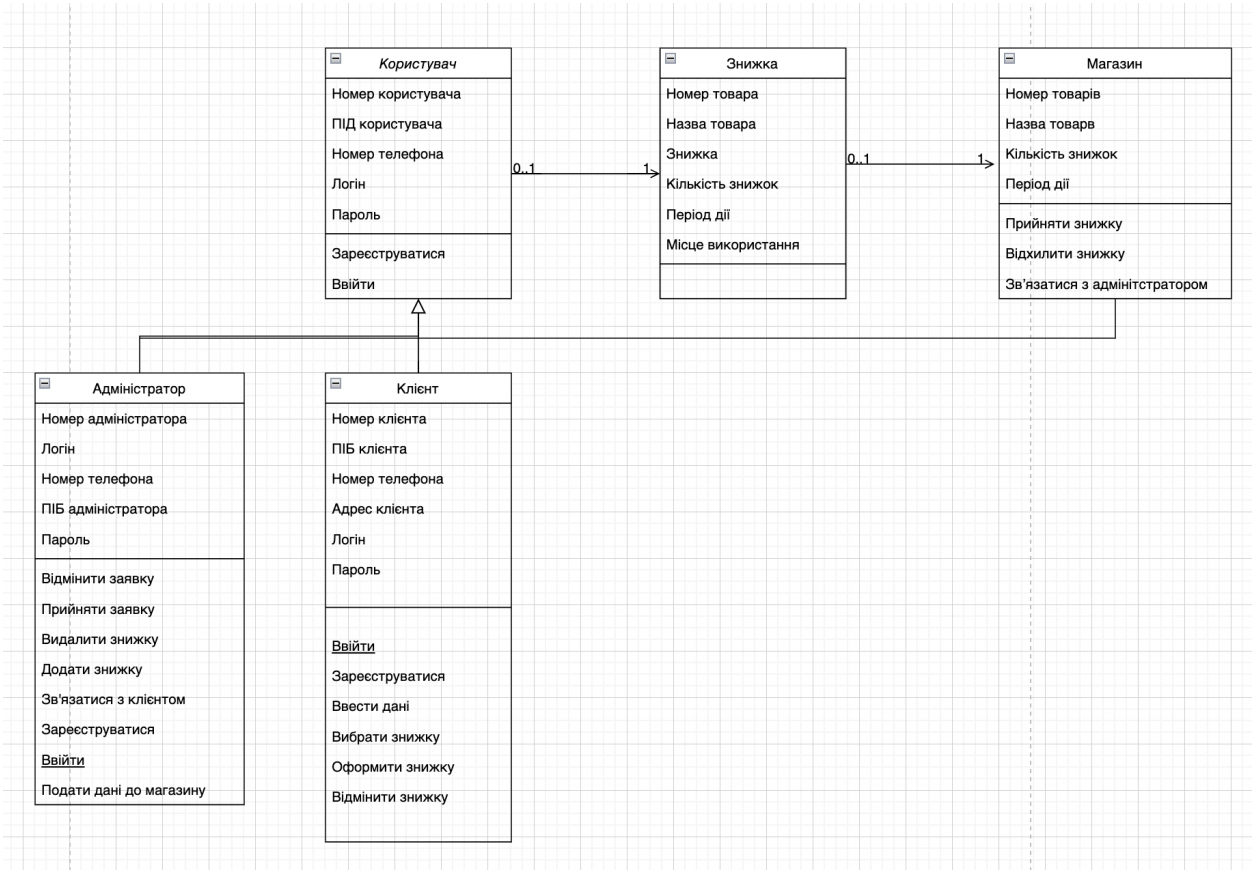

Рисунок 2.1 – Діаграма класів

На рисунку 2.1 представлена діаграма класів фінансового мобільного додатку. На діаграмі представлені: адміністратор, клієнт, який безпосередньо є користувачем системи з боку “клієнт”. Також представлений клас «Користувач», який є абстрактним та відображає спільні атрибути і методи.

Клієнт заходить в мобільний додаток, та потрапляє на головну сторінку, на якій відразу представлені деякі знижки на найпопулярніші товари, щоб відразу його зацікавити. Далі, якщо він не знайшов потрібну чи цікаву йому знижку, він може перейти до певної категорії, де представленні інші знижки на певні товари. Якщо клієнт зацікавлений знижкою, він може перейти на детальну сторінку знижки, та ознайомитися з деталями. Якщо знижка прийшлась до вподоби клієнту він може скопіювати промокод, та використати його при покупці товару. Промокод можна використовувати як в онлайн магазині, та і в оффлайн магазині.

2.6.1 Опис класу Користувач та його атрибутів і методів

Призначення: даний клас представляє собою батьківський клас для користувачів системи та є абстрактним та показує зв’язок з знижками.

Атрибути:

– номер користувача (id) – type (int), представляє унікальний ключ кожного користувача;

– ПІБ – type (string), представляє ім’я, прізвище, та по-батькові користувача;

– номер телефону – type (int), представляє унікальний номер телефону кожного користувача;

– логін – type (email), представляє унікальну электрону пошту кожного користувача;

– пароль – type (password), представляє пароль для входу на особистий аккаунт кожного користувача.

Метод - зареєструватися. Призначення: даний метод надає можливість користувачу зареєструватися в системі.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку “Зареєструватися”. Система відправляє запит до бази даних та якщо він успішний створює додає нового користувача до системи

Альтернативний потік: якщо заповнена користувачем форма реєстрації не є валідною, то система зверне увагу на поля, які було введено некоректно, та пропонує виконати знову реєстрацію.

Передумова: щоб метод було успішно виконано необхідно, заздалегідь зареєструвати електронну пошту на яку прийде підтвердження .

Виняткова ситуація 1: користувач ввів невалідні дані – система відображає відповідне повідомлення. Користувач може повторно ввести дані.

Виняткова ситуація 2: користувач раніше був зареєстрований, система пропонує ввійти, або надіслати дані для входу на електронну адресу.

Метод війти.

Призначення: даний метод надає можливість користувачу ввійти до системи.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку “Ввійти”. Система відправляє запит до бази даних та якщо він успішний відкриває клієнту його профіль.

Альтернативний потік: якщо заповнена користувачем форма реєстрації не є валідною, то система зверне увагу на поля, які було введено не коректно, та запропонує виконати знову.

Передумова: перед початком виконання даного варіанта використання користувач повинен бути зареєстрованим в системі.

Виняткова ситуація 1: користувач ввів невалідні дані – система відображає відповідне повідомлення. Користувач може повторно ввести дані.

2.6.2 Опис класу Клієнт та його атрибутів і методів

Призначення: даний клас представляє собою дочірній клас та демонструє можливості та роль клієнта в системі.

Атрибути:

– номер клієнта (id) – type (int), представляє унікальний ключ кожного клієнта;

– ПІБ – type (string), представляє ім’я, прізвище, та по-батькові клієнта;

– номер телефону – type (int), представляє унікальний номер телефону кожного клієнта;

– логін – type (email), представляє унікальну электрону пошту кожного клієнта;

– пароль – type (password), представляє пароль для входу на особистий аккаунт кожного клієнта.

Методи:

– зареєструватися;
– ввійти;
– вибрати знижку;
– використати знижку.

Призначення: даний метод надає можливість користувачу вибрати та використати знижку.

Основний потік подій: починає виконуватися, коли клієнт натискає на кнопку Використати. Система відправляє запит до бази даних та якщо він успішний видає користувачу необхідний промокод.

Виняткова ситуація 1: база даних не відповідає – система відображає відповідне повідомлення та пропонує користувачу повторити задачу пізніше.

Метод відмінити знижку.

Призначення: даний метод надає можливість користувачу відмінити знижку.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку Відмінити. Система відправляє запит до бази даних та якщо він успішний видаляє товар з корзини.

Передумова: перед початком виконання даного методу користувач повинен подати запит на знижку.

Виняткова ситуація 1: база даних не відповідає – система відображає відповідне повідомлення та пропонує користувачу повторити задачу пізніше.

Метод оформити знижку.

Призначення: даний метод надає можливість клієнту заповнити форму замовлення знижки.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку “Замовити”. Система відкриє форму замовлення.

Альтернативний потік: якщо заповнена користувачем форма замовлення не є коректною, то система зверне увагу на поля, які було введено некоректно, та пропонує виконати знову.

Виняткова ситуація 1: користувач ввів невалідні дані – система відображає відповідне повідомлення. Клієнт може повторно ввести дані.

2.6.3 Опис класу Адміністратор та його атрибутів і методів

Призначення: даний клас представляє собою дочірній клас та демонструє можливості та роль адміністратора в системі.

Атрибути:

– номер клієнта (id) – type (int), представляє унікальний ключ кожного клієнта;

– ПІБ – type (string), представляє ім’я, прізвище, та по-батькові клієнта;

– номер телефону – type (int), представляє унікальний номер телефону кожного клієнта;

– логін – type (email), представляє унікальну электрону пошту кожного клієнта;

– пароль – type (password), представляє пароль для входу на особистий аккаунт кожного клієнта.

Методи:

– зареєструватися;

– ввійти;

– додати знижку.

Призначення: даний метод надає можливість адміністратору додати знижку до мобільного додатку.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку «Додати знижку. Система відправляє запит до бази даних та якщо він успішний додає знижку до БД.

Альтернативний потік: якщо заповнена адміністратором форма знижки не є коректною, то система зверне увагу на поля, які було введено не коректно, та запропонує виконати знову.

Передумова: перед початком виконання даного варіанта використання користувач повинен бути ввійденим до системи.

Виняткова ситуація 1: адміністратор ввів некоректні дані – система відображає відповідне повідомлення. Адміністратор може повторно ввести дані.

Виняткова ситуація 2: адміністратор вже додану знижку – система відображає відповідне повідомлення. Адміністратор може повторно ввести дані.

Метод видалити знижку з мобільного додатку.

Призначення: даний метод надає можливість адміністратору видалити знижку з мобільного додатку.

Основний потік подій: починає виконуватися, коли користувач системи натискає на кнопку «Видалити знижку. Система відправляє запит до бази даних та якщо він успішний видаляє знижку з БД.

Альтернативний потік: якщо заповнена адміністратором форма видалення не є валідною, то система зверне увагу на поля, які було введено некоректно, та запропонує виконати знову.

Передумова: перед початком виконання даного варіанта використання користувач повинен бути ввійденим до системі.

Виняткова ситуація 1: користувач ввів невалідні дані – система відображає відповідне повідомлення. Адміністратор може повторно ввести дані.

Виняткова ситуація 1: користувач ввів неіснуючу знижку – система відображає відповідне повідомлення. Адміністратор може повторно виконати операцію.

Метод прийняти запит на знижку.

Призначення: даний метод надає можливість прийняти запит на знижку від користувача.

Основний потік подій: починає виконуватися, коли адміністратор отримує повідомлення, що клієнт замовив запит на знижку. Клієнт перевіряє можливість видати знижку.

Альтернативний потік: якщо заповнена клієнтом форма замовлення не є коректною, то адміністратор повідомляє про це клієнта.

Передумова: перед початком виконання даного варіанта використання адміністратор повинен бути ввійденим до системи та клієнт потрібен замовити товар.

Виняткова ситуація 1: дані товари відсутні – то адміністратор повідомляє про це клієнта.

Метод відмінити запит на знижку.

Призначення: даний метод надає можливість адміністратору відмінити запит на знижку.

Основний потік подій: починає виконуватися, коли адміністратор не приймає запит на знижку. через закінчення терміну дії знижки, чи ще не її початком , чи відмову клієнта.

Передумова: перед початком виконання даного варіанта використання адміністратор повинен бути ввійденим до системи та клієнт потрібен відмінити запит на знижку.

Метод зв’язатися з клієнтом.

Призначення: даний метод надає можливість адміністратору повідомити клієнта про успішне замовлення на знижку та уточнення нюансів.

Основний потік подій: починає виконуватися, коли адміністратор перевірить всі потрібні фактори для замовлення знижки.

Альтернативний потік: якщо клієнт не відповідає на телефонний дзвінок, то запит на знижку призупиняється на деякий час.

Передумова: перед початком виконання даного варіанта використання адміністратор повинен бути ввійденим до системи.

Виняткова ситуація 1: клієнт не відповідає – система відображає відповідне повідомлення в профілі користувача.

Метод передати всі аналітичні дані до магазину.

Призначення: даний метод надає можливість передавати аналітичні дані магазинам.

Основний потік подій: починає виконуватися, коли адміністратор передасть магазину всі аналітичні дані.

Передумова: перед початком виконання даного варіанта використання адміністратор повинен передати магазину всі аналітичні дані.

2.6.4 Опис класу Магазин та його атрибутів і методів

Призначення: даний клас представляє собою абстрактний клас та демонструє можливості та роль магазину в системі.

Атрибути:

– номер товарів (id) – type (int), представляє унікальний ключ кожного товару;

– назва товару – type (string), назву товару;

– кількість знижок – type (int), представляє доступну кількість знижок;

– термін дії знижки – type (email), представляє термін дії знижки.

Метод прийняти знижку

Призначення: даний метод надає можливість адміністратору магазину прийняти знижку.

Основний потік подій: починає виконуватися, коли клієнт використає промокод в магазині.

Альтернативний потік: якщо заповнена клієнтом форма знижки не є коректною, то магазин зверне увагу на поля, які було введено некоректно, та запропонує виконати знову.

Передумова: перед початком виконання даного варіанта використання користувач повинен взяти промокод з мобільного додатку.

Виняткова ситуація 1: клієнт ввів некоректні дані – магазин інформує клієнта про невірне використання.

Виняткова ситуація 2: клієнт ввів вже використану знижку – магазин інформує клієнта про невірне використання.

Метод відхилити на знижку.

Призначення: даний метод надає можливість магазину відхилити знижку.

Основний потік подій: починає виконуватися, коли магазин не приймає знижку. через закінчення терміну дії знижки, чи ще не її початком , чи відмову клієнта.

Передумова: перед початком виконання даного варіанта використання клієнт потрібен використати знижку.

Метод зв’язатися з адміністратором.

Призначення: даний метод надає можливість зв’язатися з адміністратором магазину для повідомлення про нові знижки або зміни умов в інших.

Основний потік подій: починає виконуватися, коли магазин звернеться до адміністратора.

Передумова: перед початком виконання даного варіанта використання магазин повинен звернутися до адміністратора.

2.6.5 Опис класу Знижки та його атрибутів і методів

Призначення: даний клас представляє собою абстрактний клас для програмної реалізації замовлення.

Атрибути:

– знижка – type (object), детальна інформація про знижку;

– назва товару – type (string), назва товару для якого діє знижка;

– номер товару (id) – type (int), представляє унікальний ключ кожного товару;

– кількість знижок – type (int), кількість доступних знижок;

– період дії – type (time), представляє період дії знижки;

– місце використання – type (string), представляє місце, де користувач може використати знижку.

2.7 Діаграма діяльності

При моделюванні поведінки проєктованої системи виникає необхідність в детальному розумінні всіх процесів зміни їх стану, але і деталізувати всі нюанси алгоритмічної і логічної реалізації операцій, які виконуються регулярно. Досить часто для таких цілей використовувалися блок-схеми або структурні схеми алгоритмів. Всі ці схеми звертають увагу та деталізують послідовність виконання певних дій або елементарних операцій, які в сумісному виконанні призводять до отримання необхідного результату.

Для моделювання процесу виконання операцій в мові UML використовуються діаграми діяльності. Всі стани на діаграмі діяльності відповідають виконанню деякої простої операції, а перехід в наступний до наступного стану буде виконано тільки по завершенні цієї операції в попередньому стані.

Діаграми створені, щоб давати зрозуміти важливу роль потоків управління в системі, яка моделюється. Звичайні блок-схеми алгоритмів, які використовуються для цієї мети, мають велику кількість обмежень в відображенні паралельних процесів і синхронізації між собою. Застосування доріжок і об’єктів відкриває необхідні можливості для детального уявлення процесів, дозволяючи специфікувати діяльність підрозділів організації. Ці

діаграми будуються для окремих класів, окремих операцій класу або цілої підсистеми.

Діяльністю називається поведінка, реалізована об’єктом, коли він знаходиться в відповідному стані. Діяльність – поведінка, яка може зупинятися. Вона може виконуватися до свого завершення, якщо об’єкт знаходиться в відповідному стані, або може бути перервана переходом об’єкта в інший стан.

Виконання діяльності є виконанням окремої діяльності, буквально, включно із виконанням тих дій, що містяться в діяльності. Кожна дія в діяльності може виконуватись один, два, або більше разів під час одного виконання діяльності. Щонайменше, дії мають отримувати дані, перетворювати їх та тестувати, деякі дії можуть вимагати певної послідовності. Специфікація діяльності (на вищих рівнях сумісності) може дозволяти виконання декількох (логічних) потоків, та існування механізмів синхронізації для гарантування виконання дій у правильному порядку.

Наведемо діаграму діяльності, що описує модель поведінки варіанти використання “Використання” в магазині. Діаграма представлена на рисунку 2.7.

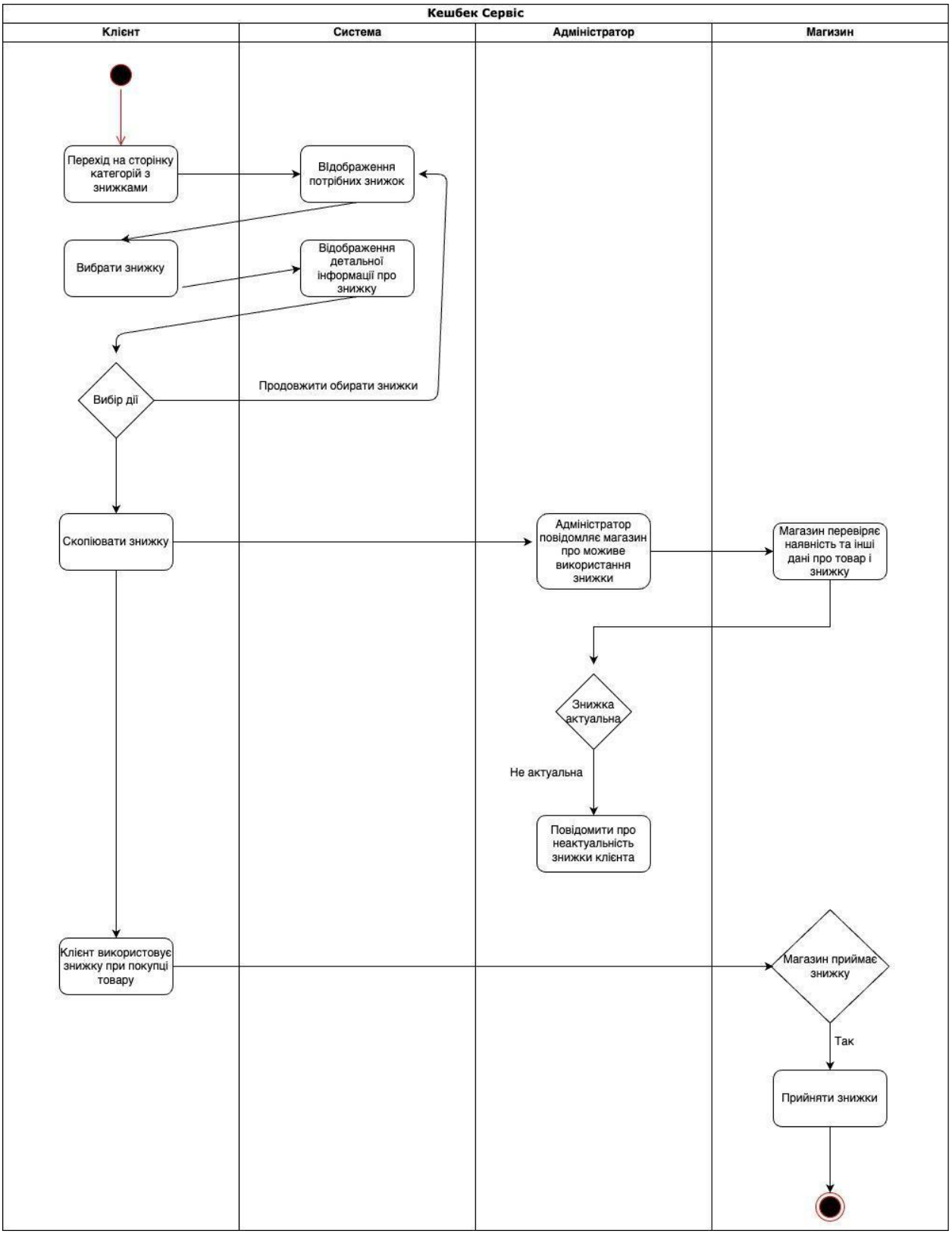

Рисунок 2.7 – Діаграма діяльності

2.8 Діаграма розгортання

На діаграмі розгортання відображаються всі задіяні ресурси та їх зв’язки між собою під час роботи програми, а також об’єкти, що виконуються на цих вузлах. Всі задіяні елементи відповідають представленню робочих екземплярів одиниць коду. Діаграма розгортання відображає робочі екземпляри компонентів (див. рис. 2.8).

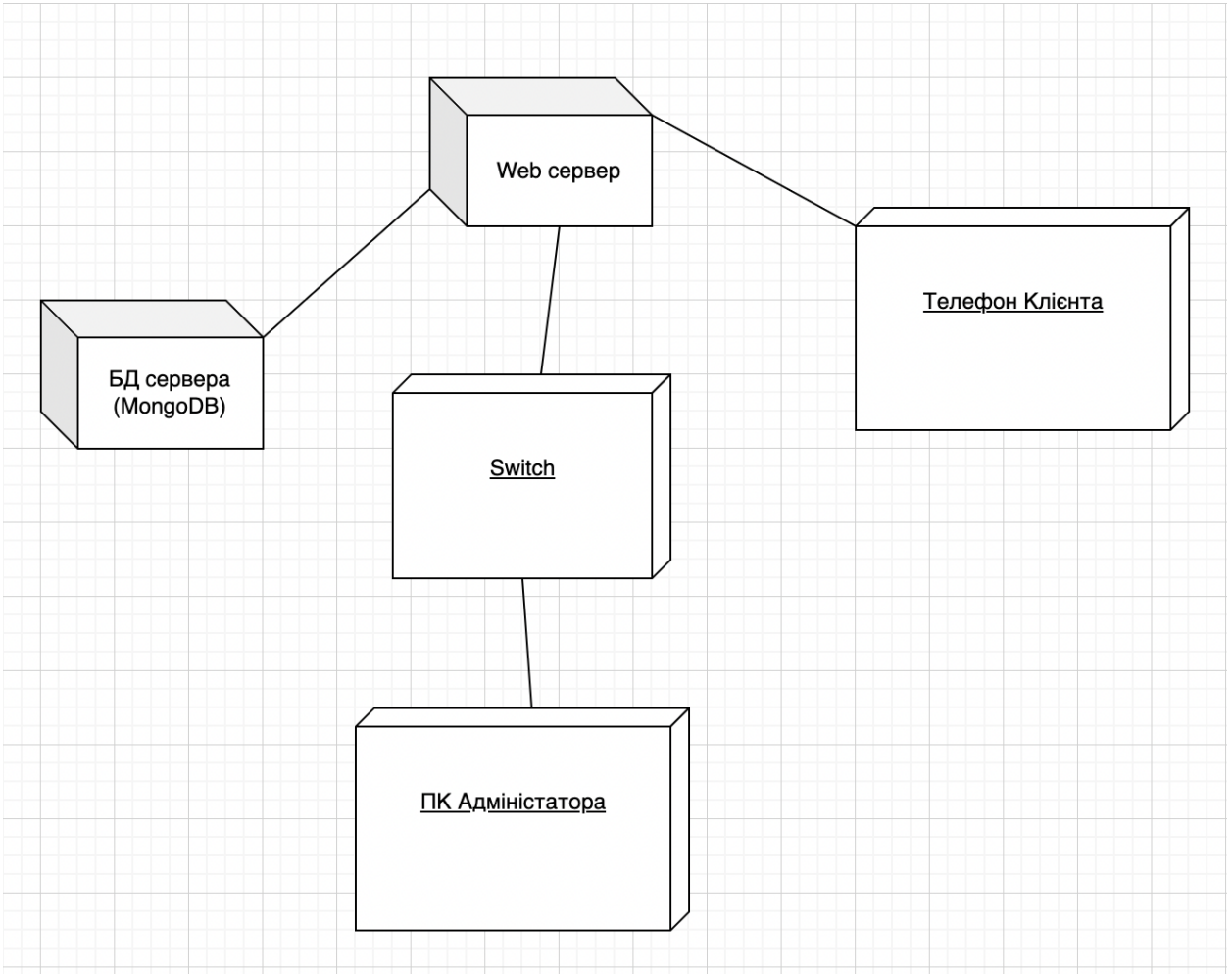

Рисунок 2.8 – Діаграма розгортання

3 РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ

3.1 Процес створення сервера

Для того, щоб створити проєкт потрібно перейти до потрібного каталогу де буде розміщуватися серверна частина проєкту. Переходимо до терміналу та виконуємо команду “npm init” для ініціалізації проєкту. Далі необхідно встановити декілька пакетів для успішної роботи сервера, а саме (npm install <пакет>):

- Mongoose – підключення до БД;
- Express – створення сервера;
- JsonWebToken – передача даних клієнта для авторизації;
- BcryptJs – хешування даних клієнта.

Наступним кроком створено створено роутери, які будуть відповідати за адресацію Routers:

- ‘auth_routes’ – для авторизаціі;
- ‘categories_routes’ – для категорій;
- ‘cash_routes’ – для товарів.

До кожного роутера відключається відповідна модель бази даних. В роутах підключаються пакети для шифрування, отримання та передачі данних з бази даних (require ('bcryptjs')).

Наступним короком було створення самого серверу в якому необхідно підкличити пакети для налаштування сервера та підключення бази даних (require (' mongoose')). Налаштовується сервер на порту(5000). Підключено створені раніше роутери (app.use ('/api/auth', require ('../routes/auth_routes'))).

3.2 Процес створення та налаштування MongoDB

Спочатку було створено акаунт до якого буде виконуватися підключення. Далі створюється база даних, в налаштування якої необхідно взяти адресу, необхідну для підключення ("mongoUrl": "mongodb+srv://yura:72uliton@cluster0.j0hwi.mongodb.net/myFirstDatabase?retryWrites=true&w=majority").

Далі в головному файлі сервера (app.js) підключається БД (mongoose.connect (config.get ('mongoUrl'), {

useNewUrlParser: true

useUnifiedTopology: true,

useCreateIndex: true.

})).

Наступним кроком було створені моделі, які відповідають необхідним таблицям в БД для творення колекцій. Models:

– ‘user’ – модель користувача;
– ‘category’ – модель категорії;
– ‘item’ – модель товару.

Нижче наведено діаграму «Сутність-Зв’язок» даних моделей.

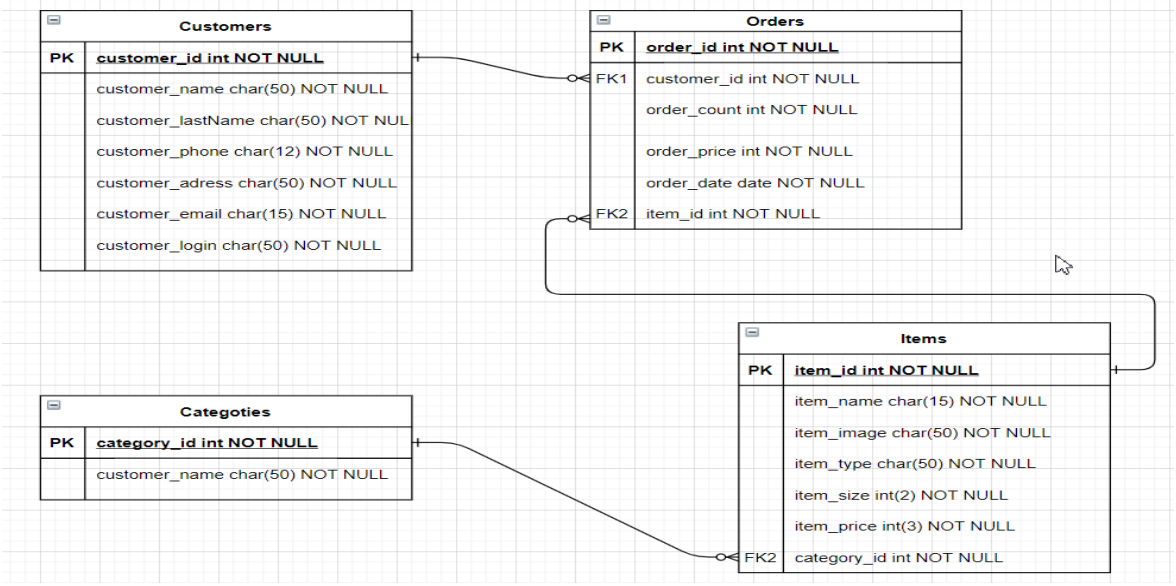

Рисунок 3.2 – Діаграма класів

3.3 Процес створення мобільного додатку React Native

Для того, щоб створити проєкт потрібно перейти до потрібного каталогу де буде розміщуватися серверна частина проєкту було отворено папку “client”. Переходимо до терміналу та виконуємо команду “npx create-react-native-app free-shop --template react-native-template-typescript”. Далі встановлено необхідні пакети (yarn add <пакет>):

– react-native-navigation – управління навігацією;
– React-redux – управління станом компонентів.
– react-native-fast-image – кешування картинок;
– react-native-reanimated – управління анімацією.

Далі в головному файлі App.tsx весь додаток обертається в роутери (NavigationProvider, ErrorBoundary, ReduxProvider) для того, щоб в додатку була доступна навігація, стан для управління додатком і уловлювач помилок.

Наступним кроком було створено компоненти, які будуть багаторазово використовуватися в різних сторінках. Components:

– ‘Button.tsx’ – компонент кнопки з різними станами;
– ‘Coupon.tsx’ – компонент купона кешбеку;
– ‘Header.tsx’ – компонент верхної частини екранів;
– ‘Carousel.tsx’ – компонент каруселі елементів;
– ‘Input.tsx’ – компонент для вводу текста.

Далі було створено сторінки сайту. Screens:

– ‘LoginScreen.tsx’ – екран авторизації та реєстрації;
– ‘CashbackScreen.tsx’ – екран детальної інформації про кешбек;
– ‘SalesScreen.tsx’ – екран з усіма доступними пропозиціями;
– ‘OnboardingScreen.tsx’ – екран демонстрації можливостей додатку.

Наступним кроком було створено свої хуки. Хуки – це функції, за допомогою яких можна працювати зі станом і методами життєвого циклу React з функціональних компонентів. CustomHooks:

– ‘Auth.hook.ts’ – хук, який відповідає за створення токену доступу користувача, та по сплину певного час просить виконати повторну автентифікацію;

– ‘Http.hook.ts’ – хук, який відповідає http запит з його заголовками.

Далі було створено стан компоненти (redux) для того, щоб в стані виконувати роботу з даними. Створено actions, які будуть викликатися в компонентах. Actions:

– ‘cart.js’ – подія, яка описує об’єкт, що відповідає за додаваня, видалення товарів з корзини;

– ‘categories.ts’ – подія, яка описує об’єкт, що відповідає за взяття категорій з БД;

– ‘filters.ts’ – подія, яка описує об’єкт, що відповідає за фільтрацію товарів;

– ‘cashbacks.ts’ – подія, яка описує об’єкт, що відповідає за взяття товарів з БД.

Для того, щоб передавати actions було створено reducers, які виконують певну роботу із даними. Для того, щоб викликати reducers передавати actions в dispatch. Reducers:

– ‘cart.js’ – оброблювач, який отримує подію і виконує редагування товарів в кошику, та передає в redux;

– ‘categories.ts’ – оброблювач, який отримує подію і додає категорії в redux;

– ‘filters.ts’ – оброблювач, який отримує подію і фільтрує за переданими пареметрами товари і передає в redux;

3.4 Тестування проєкту

Тестування займає важливу роль в процесі розробки і створення якісного програмного забезпечення. Необхідно серйозно ставитися до аналізу і проєктування структурованого процесу, який забезпечує своєчасний і успішний випуск проєкту. Тестування буває автоматизованим та ручним.
При ручному підході тест-кейси запускаються вручну без використання програмних засобів. При автоматизованому тестуванні запуск тест-кейсів здійснюється за допомогою спеціально підготовлених скриптів.

Чек-лист – це список, який містить ряд необхідних перевірок для будь-якої роботи. Відзначаючи пункти списку, можна дізнатися про стан або коректності виконання цієї роботи. Саме завдяки чек-листу можна деталізовано протестувати продукт. Чек-лист має велике значення в великих проєктах тому, що неможливо перевірити всі пункти для тестування без детального аналізу проєкту (рис. 3.1, 3.2).

	A	B	C
1	Platform	Android	IOS
2	Головна сторінка		
3	Наявність логотипу	Passed	Passed
4	Наявність логотипу меню	Passed	Passed
5	Наявність футера	Passed	Passed
6	Елементи сторінок		
7	Коректне відображення стилів	Passed	Passed
8	Відображення шрифта тексту	Passed	Passed
9	Коректне відображення кнопок і компонентів	Passed	Passed
10	Відображення кольорові гами усіх елементів	Passed	Passed
11	Граматична коректність текстів	Passed	Passed
12	Коректні підсказки для усіх елементів	Passed	Passed
13	Коректність скролу	Passed	Passed
14	Коректне розміщення головних елементів	Passed	Passed
15	Адаптивність додатку	Passed	Passed
16	Маштабування фото до контенту	Passed	Passed

Рисунок 3.1 – Чек-лист верстки

	A	B	C
1	Platform	Android	IOS
2	Регістрація й особистий профіль		
3	Регістрація	Passed	Passed
4	Редагування профіля	Passed	Passed
5	Форма зворотнього зв'язку		
6	Перевірка валідації полів	Passed	Passed
7	Перевірка відправлення смс повідомлень	Passed	Passed
8	Перевірка доставки смс повідомлень	Passed	Passed
9	Користувач		
10	Редагування профіля	Passed	Passed
11	Поділитися знижкою	Passed	Passed
12	Вибрати знижку	Passed	Passed
13	Фільтрувати знижку	Passed	Passed
14	Замовити знижку	Passed	Passed
15	Адміністратор		
16	Видалення користувача	Passed	Passed
17	Коректировка даних користувача	Passed	Passed
18	Додавання категорій	Passed	Passed
19	Видалення категорій	Passed	Passed
20	Додавання знижок	Passed	Passed
21	Видалення знижок	Passed	Passed
22	Перегляд замовлень	Passed	Passed
23	Перегляд замовлень	Passed	Passed

Рисунок 3.2 – Чек-лист функціоналу

Додаток містить unit і integrations тести для всіх компонентів, екранів, хуків, додатків. Далі наведено приклад одного з таких тестів та його роботи (рис. 3.3, 3.4).

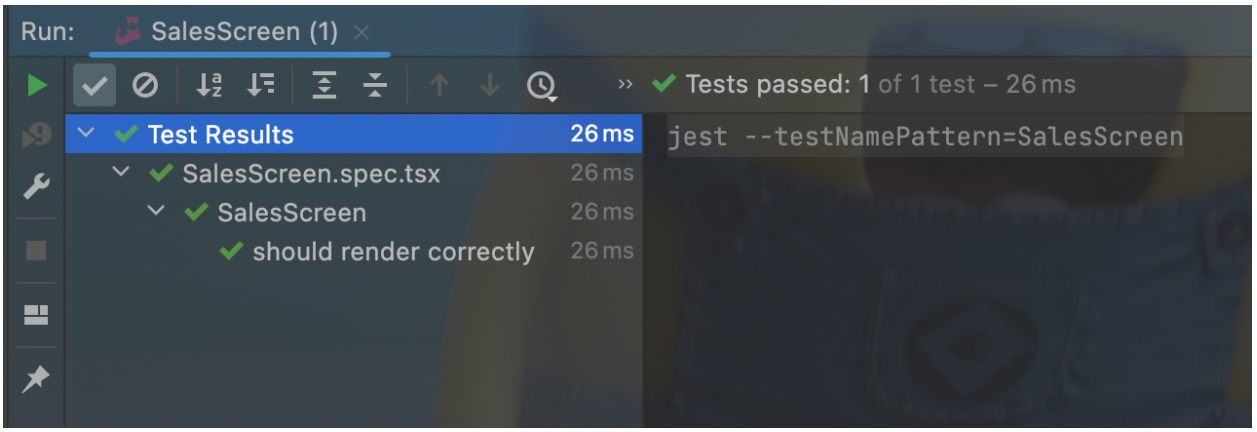

Рисунок 3.3 – Приклад роботи тесту

```
describe( name: 'SalesScreen', fn: () => {
  it( name: 'should render correctly', fn: () => {
    const { getByText } = render(<SalesScreen />);

    expect(getByText( predicate: 'sales.addCoupon')).toBeDefined();

    expect(getByText( predicate: 'coupon.title')).toBeDefined();
    expect(getByText( predicate: 'coupon.text')).toBeDefined();

    expect(setValuesSpy).toHaveBeenCalledWith( params: {
      ...startingFormikState,
      isLinkingCard: false,
    });

    expect(mockGoToPage).toHaveBeenCalled();

    fireEvent.press(getByText( predicate: 'coupon.title'));

    expect(setValuesSpy).toHaveBeenCalledWith( params: {
      ...startingFormikState,
      isLinkingCard: true,
    });

    expect(mockGoToPage).toHaveBeenCalled();
  });
});
```

Рисунок 3.4 – Приклад тесту

3.5 Керівництво користувача

3.5.1 Рівень кваліфікації користувача

Користувач сайту повинен володіти певною кваліфікацією.

Навички користувача для роботи з смартфоном, та навички роботи користувача в інтернеті.

Знайомство з Керівництвом користувача.

3.5.2 Підготовка до роботи

Запуск системи.

Доступ до фінансового додатку здійснюється через мережу Інтернет за допомогою мобільного додатку. Для коректної роботи клієнтської частини повинен використовуватися версія операційної системи android 9.0 та вище, ios 13.0 та вище.

При вході в додаток користувач потрапляє на сторінку авторизація після успішної реєстрації чи авторизації користувач потрапляє на головну сторінку в розділ «Home».

Мобільний додаток підтримує такі типи користувачів:

Анонімний користувач – приватна особа, не авторизована на сайті, що не має доступу до основного контенту.

Користувач – приватна особа, авторизований на сайті (далі Користувач), що має можливість редагування профілю та використання промокодів.

Адміністратор магазину – особа, що відповідає за обробку промокодів, та керування системою.

3.5.3 Склад та структура застосунку

Застосунок складається з двох функціональних частин (див. рис. 3.3):

Клієнтська – відповідає за процес вибору та замовлення кешбеку.

При вході в мобільний додаток користувач повинен зареєструватися, та пройти по необхідності підказки по можливостям додатку. При реєстрації користувач вибирає регіон, в якому він буде робити подальші замовлення. Потім зареєстрований користувач переходить на першу сторінку з всіма доступними знижками для його регіону.

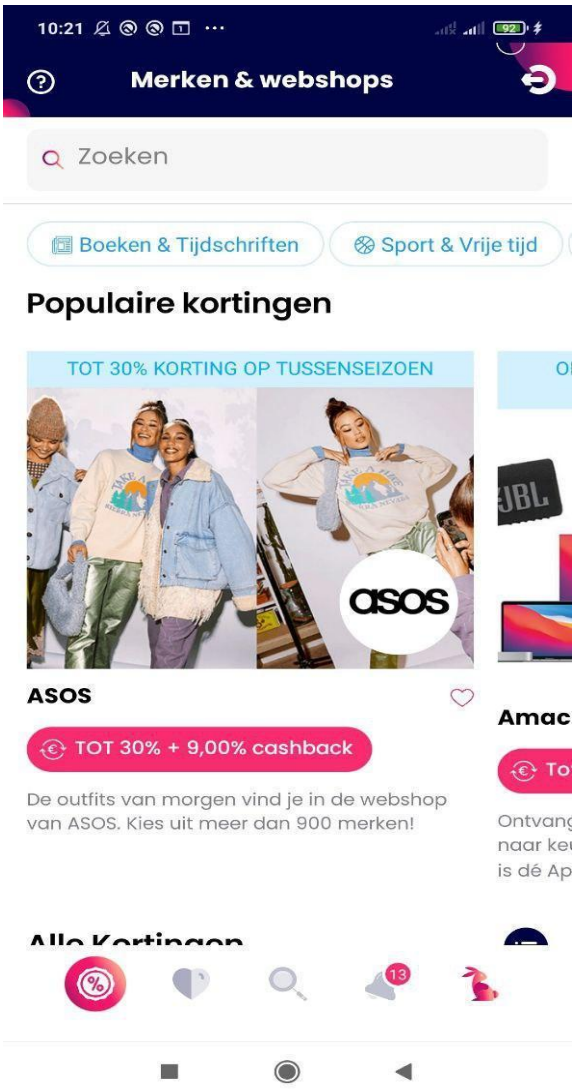

Рисунок 3.3 – Інтерфейс головного екрану

Далі користувач може пролистати до низу, та ознайомитися і можливо зацікавитись якимись знижками. Якщо користувач шукає знижку для певного товару, він може скористатися пошуком або вибрати конкретну категорію. Користувач може добавляти товари в улюблені (див. рис. 3.4, 3.5, 3.6).

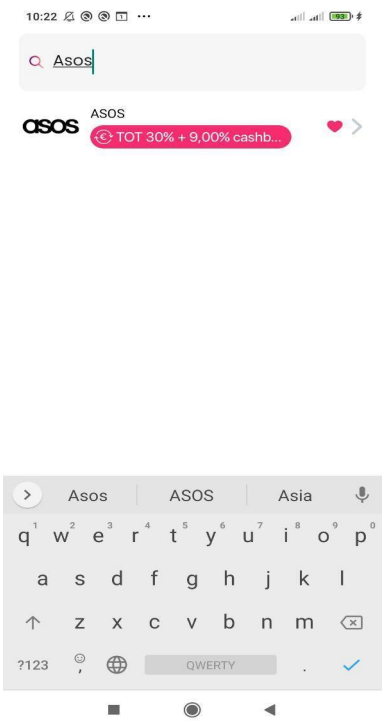

Рисунок 3.4 – Пошук знижки на певний товар

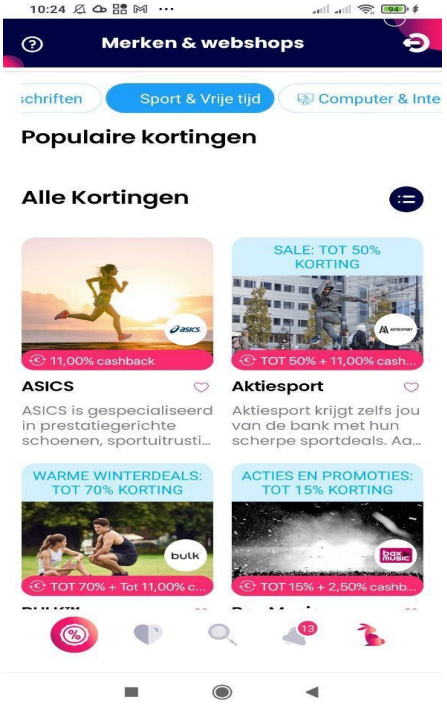

Рисунок 3.5 – Фільтрація певної категорії

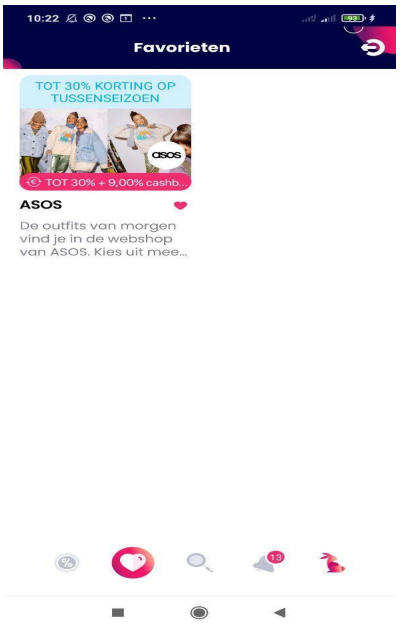

Рисунок 3.6 – Обрані знижки

3.5.4 Процес використання знижки (клієнт)

Коли клієнт знаходить необхідну знижку, то він може її зберегти, поділитися в соцмережі, чи на електронну пошту, або скопіювати, та зайти в магазин, в якому вона діє, та використати її (див. рис. 3.7, 3.8).

Рисунок 3.6 – Необхідна знижка

Якщо користувач обрав поділитися через електонну пошту, то він перейде в поштовий додаток, з текстом кешбеку

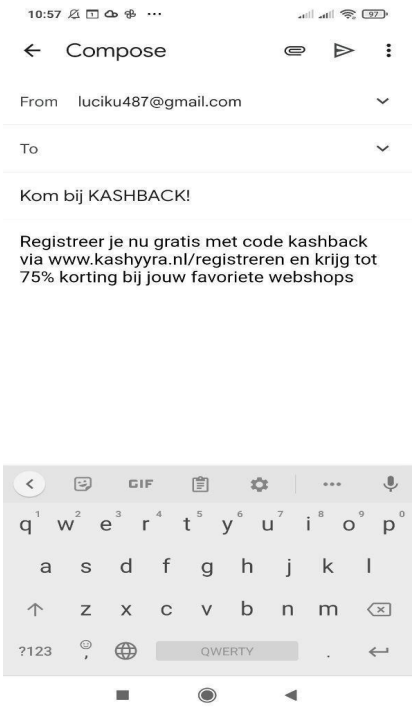

Рисунок 3.7 – Каталог товарів

ВИСНОВКИ

Результатом проведеної роботи було написано технічне завдання на розробку мобільного додатку засобами Express JS та React Native. Для створення цієї системи було обрано середу розробки Intellij IDEA, через її великі можливості та кількість встроєних додатків.

У відповідності з метою кваліфікаційної роботи був розроблений фінансовий мобільний додаток “FreeShop” на базі React Native та Express JS з наступними технологіями:

– React Native для реалізації front-end;

– Express JS для реалізації back-end.

У відповідності з поставленими задачами були виконані наступні етапи створення модуля:

– сформовані вимоги до веб застосунку (функціональні та нефункціональні (інтерфейс, кросбраузерність, безпека)). Також проведено огляд предметної області;

– реалізовано мобільний додаток “FreeShop” на базі React Native та ExpressJS (наведена інструкція по створенню модуля та структуру проєкту);

– протестована робота мобільного додатку.

ПЕРЕЛІК ПОСИЛАНЬ

1. Шиколенков Т. А. Розробка і робота мобільних додатків: навч. посіб. Київ: Видавничий центр КНУКіМ, 2014. 379 с.
2. Акіла М. Т. Початковий Node.js і Express.js: навч. посіб. Київ: Видавничий центр КНУКіМ, 2016. 334 с.
3. Мардан А. К. React швидко. Мобільні додатки на React Native, JSX, Redux: навч. посіб. Львів: “ОРілі”, 2019. 560 с.
4. Стефанов О. А. React Native швидкий старт: навч. посіб. Київ: Видавничий центр КНУКіМ, 2016. 478 с.
5. Ходоров Х. В. Розробка кросс-платформних додатків на React Native: навч. посіб. Харків: Видавничий центр ВНУТ, 2017. 491 с.
6. Копелан Р. М. Патерни розробки NodeJS: навч. посіб. Харків: Видавничий центр ВНУТ, 2014. 345 с.
7. Янг А. А., Кантелон М. Р. Node.js в дії: навч. посіб. Київ: ООО “Маннінг”, 2017. 378 с.
8. Хедрік Б. К. Вивчення React Native: навч. посіб. Дніпро: “ОРілі”, 2018. 251 с.
9. Бенкер К. О. MongoDB в дії: навч. посіб. Миколаїв: ООО “Ексмо”, 2014. 452 с.
10. Чиннатамбі В. Р. Практичне керівництво React Native: навч. посіб. Київ: ООО “Ексмо”, 2019. 368 с.
11. Багл Р. М. Вивчення React Hooks: навч. посіб. Харків: Видавничий центр ВНУТ, 2017. 308 с.
12. Бенкс А. К., Порселло Є. Ю. Вивчення React і Redux: навч. посіб. Київ: “ОРілі”, 2017. 408 с.
13. Холмс С. О. Технології стека React Native і Typescript: навч. посіб. Київ: ООО “Маннінг”, 2018. 457 с.

ДОДАТОК А

Основні функціональні компоненти та хуки

A.1 CashbackScreen.tsx

```
import React from 'react';
import {View, StyleSheet, Text, Image} from 'react-native';
import {useNavigation} from '../../utils/navigation';

import CommonView from '../../components/Views/CommonView';
import EmptyHeader from '../../components/Headers/EmptyHeader';
import {CommonButton} from '../../components/Buttons/CommonButton';
import StepByStep from '../../components/StepByStep';

import {calcFontSize, calcHeight, calcWidth} from '../../utils/dimensions';
import {stepByStepRequests} from '../../styles/stepByStep';
import {black, grey, white} from '../../utils/constants/colors';
import {FONTS} from '../../utils/fonts';

const Cashback = () => {
 const {navigate} = useNavigation();
 return (
  <CommonView style={styles.main}>
    <EmptyHeader />
    <View style={styles.margin}>
      <View
        style={{flex: 2.25, alignItems: 'center', justifyContent: 'center'}}
      >
        <Image // ok
          source={require('../../../assets/png/cashback-image.png')}
          style={{
            width: calcHeight(185),
            height: calcHeight(185),
```

```
          }}
        />
      </View>

      <View style={{flex: 1.7}}>
        <Text style={styles.header}>
          Ontvang automatisch een deel van je geld terug
        </Text>
        <Text style={styles.text}>
          Webshops betalen voor marketing, maar wij delen hun kosten graag met
          jou
        </Text>
      </View>
      <View style={{flex: 1}}>
        <View style={{flexDirection: 'row'}}>
          <CommonButton
            onPress={() => console.log('asdasdasdasdasdas')}
            text="Aan de slag"
            styleContainer={{flex: 1, marginRight: calcWidth(5)}}
            isBorderGradient={true}
          />
          <CommonButton
            onPress={() => console.log('asdasdasdasdasdas')}
            text="Bekijk video  ►"
            styleContainer={{flex: 1, marginLeft: calcWidth(5)}}
          />
        </View>
        <StepByStep steps={3} currentStep={2} style={stepByStepRequests} />
      </View>
    </View>
  </CommonView>
 );
};

const styles = StyleSheet.create({
```

```
 main: {
   flex: 1,
   backgroundColor: white,
 },
 margin: {
   marginHorizontal: calcWidth(20),
   flex: 1,
 },
 header: {
   color: black,
   fontSize: calcFontSize(24),
   fontFamily: FONTS.Poppins.Bold700,
   textAlign: 'center',
   width: '90%',
   alignSelf: 'center',
 },
 text: {
   color: grey,
   fontSize: calcFontSize(15),
   fontFamily: FONTS.Poppins.Regular400,
   textAlign: 'center',
   marginTop: calcHeight(5),
 },
});

export default Cashback;
```

A.2 CouponComponent.tsx

```
import React from 'react';
import {
 StyleSheet,
 Text,
 TouchableOpacity,
```

```
 View,
 Linking,
 Platform,
 Alert,
} from 'react-native';

import {calcFontSize, calcHeight, calcWidth} from '../../utils/dimensions';
import Clipboard from '@react-native-clipboard/clipboard';
import {black, linearGradient, red, white} from '../../utils/constants/colors';
import LinearGradient from 'react-native-linear-gradient';

import Kopier from '../../../assets/svg/Kopier.svg';
import WhatsappIcon from '../../../assets/svg/Wapsapp.svg';
import QuestionSvg from '../../../assets/svg/question-cicrle.svg';
import Email from '../../../assets/svg/email.svg';

interface ICouponComponentProps {
 copierText?: string;
 isShownIcon?: boolean;
 code?: string;
 couponText?: string;
 margin?: number;
}

const CouponComponent = ({
 code,
    // copierText = `Registreer je nu gratis met code ${code} via https://raboomapp.com/SigningScreen/${code} en raboomapp://SigningScreen/${code} krijg tot 75% korting bij jouw favoriete webshops`,
 copierText = `Registreer je nu gratis met code ${code} via www.raboom.nl/registreren en krijg tot 75% korting bij jouw favoriete webshops`,
 isShownIcon,
 couponText = 'Deel jouw code en krijg €2,50!',
 margin = 0,
}: ICouponComponentProps) => {
```

```
const shareWhatsappHandle = () => {
  console.log('Share')
};

const shareEmailHandle = () => {
        const emailBodyIOS = `Registreer je nu gratis met code ${code} via www.raboom.nl/registreren en krijg tot 75% korting bij jouw favoriete webshops`;
        const emailBodyAndroid = `Registreer je nu gratis met code ${code} via www.raboom.nl/registreren en krijg tot 75%25 korting bij jouw favoriete webshops`;
  const emailSubject = 'Kom bij RABOOM!';
  const handle = Platform.select({
    android: () => {
      Linking.openURL(
        `mailto:?subject=${emailSubject}&body=${emailBodyAndroid}`,
      ).catch(e => console.log('error', e));
    },
    ios: () => {
      Linking.openURL(
        `mailto:?subject=${emailSubject}&body=${emailBodyIOS}`,
      ).catch(e => console.log('error', e));
    },
  });
  if (handle) {
    handle();
  }
};

return (
  <View style={[styles.itemContainer, {marginHorizontal: margin}]}>
    <LinearGradient
      start={{x: 0, y: 1}}
      end={{x: 1, y: 1}}
      locations={linearGradient.main.locations}
      colors={linearGradient.main.colors}
      style={styles.backgroundGradient}
```

```
>
  {isShownIcon && (
   <TouchableOpacity
    onPress={() => {
     Linking.openURL('https://raboom.nl/account#deel').catch(e =>
      console.log('error', e),
     );
    }}
   >
    <QuestionSvg
     width={calcWidth(18)}
     height={calcWidth(18)}
     margin={calcWidth(8)}
    />
   </TouchableOpacity>
  )}
  <View style={styles.leftColumn}>
   <Text style={styles.couponTitle}>{couponText}</Text>
   <Text style={styles.couponNumber}>{code}</Text>
  </View>
  <View style={styles.rightColumn}>
   <View>
    <TouchableOpacity
     style={styles.button}
     onPress={() => Clipboard.setString(`${copierText}`)}
    >
     <View style={{flexDirection: 'row'}}>
      <Kopier
       width={calcWidth(14)}
       height={calcWidth(14)}
       alignSelf="center"
      />
      <Text style={styles.text}>Kopieer link</Text>
     </View>
    </TouchableOpacity>
```

```
          <TouchableOpacity
            style={styles.button}
            onPress={shareWhatsappHandle}
          >
            <View style={{flexDirection: 'row'}}>
              <WhatsappIcon
                width={calcWidth(14)}
                height={calcWidth(14)}
                alignSelf="center"
              />
              <Text style={styles.text}>Whatsapp</Text>
            </View>
          </TouchableOpacity>
          <TouchableOpacity style={styles.button} onPress={shareEmailHandle}>
            <View style={{flexDirection: 'row'}}>
              <Email
                width={calcWidth(14)}
                height={calcWidth(14)}
                alignSelf="center"
              />
              <Text style={styles.text}>E-mail</Text>
            </View>
          </TouchableOpacity>
        </View>
      </View>
    </LinearGradient>
  </View>
 );
};

const styles = StyleSheet.create({
 itemContainer: {
   flex: 1,
   paddingTop: calcHeight(10),
 },
```

```
backgroundGradient: {
  flexDirection: 'row',
  borderRadius: 20,
  width: '100%',
},
leftColumn: {
  flex: 3,
  paddingVertical: calcHeight(15),
  paddingRight: calcWidth(28),
  alignItems: 'center',
},
rightColumn: {
  flex: 2,
  paddingTop: calcHeight(10),
  paddingRight: calcWidth(20),
  marginBottom: calcHeight(20),
},
couponTitle: {
  fontSize: calcFontSize(14),
  color: white,
  textAlign: 'center',
},
couponNumber: {
  fontSize: calcFontSize(22),
  marginTop: calcHeight(18),
  fontWeight: 'bold',
  color: white,
},
button: {
  height: calcHeight(24),
  backgroundColor: white,
  color: red,
  justifyContent: 'center',
  borderRadius: 10,
  marginBottom: calcHeight(5),
```

```
  paddingLeft: calcWidth(10),
 },
 container: {
  height: calcHeight(24),
  marginBottom: 2,
 },
 text: {
  color: red,
  fontSize: calcFontSize(12),
  marginTop: calcHeight(-2),
  flex: 1,
  textAlign: 'center',
 },

 image: {
  width: 13,
  height: 13,
  borderWidth: 2,
  borderColor: black,
  marginLeft: 0,
 },
});

export default CouponComponent;
```

A.3 LoginScreen.tsx

```
import React, {useEffect, useState} from 'react';
import {View, StyleSheet, Text, TouchableOpacity} from 'react-native';
import LinearGradient from 'react-native-linear-gradient';
import {useNavigation} from '../../utils/navigation';

import {ErrorInput} from '../../components/Inputs/ErrorInput';
import {CommonButton} from '../../components/Buttons/CommonButton';
```

```
import CommonView from '../../components/Views/CommonView';
import {SimpleTextInput} from '../../components/Inputs/SimpleTextInput';

import {calcFontSize, calcHeight, calcWidth} from '../../utils/dimensions';
import {linearGradient, white} from '../../utils/constants/colors';
import {FONTS} from '../../utils/fonts';
import LogoIcon from '../../../assets/svg/raboom.svg';
import LabelIcon from '../../../assets/svg/label.svg';
import {
 fetchLogin,
 setEmail,
 setPassword,
} from '../../store/actions/signingActions';
import {useDispatch, useSelector} from 'react-redux';
import {IRootReducer} from '../../store/reducers';
import {translate} from '../../services/api/localization';

const LoginScreen = () => {
 const {email, password} = useSelector(
   (state: IRootReducer) => state.signingReducer,
 );
 const dispatch = useDispatch();
 const {navigate, reset} = useNavigation();
 const validatePass = (pass: string): boolean => {
   return pass.length < 6;
 };
 const validateEmail = (e: string): boolean => {
   const regex =

/^(([^<>()[\]\\.,;:\s@"]+(\.[^<>()[\]\\.,;:\s@"]+)*)|(".+"))@((\[[0-9]{1,3}\.[0-9]{1,3}\.[0-9]{1,3}\.
[0-9]{1,3}\])|(([a-zA-Z\-0-9]+\.)+[a-zA-Z]{2,}))$/;
   return !re.test(String(e).toLowerCase());
 };
 const [emailErrors, setEmailErrors] = useState<boolean | null>(
   validateEmail(email) || true,
```

```
);
const [isValidPass, setValidPass] = useState<boolean | null>(
  validatePass(password) || true,
);
useEffect(() => {
  setValidPass(validatePass(password));
  setEmailErrors(validateEmail(email));
}, []);

return (
  <LinearGradient
    start={{x: 0.4, y: 1}}
    end={{x: 1, y: 0}}
    locations={linearGradient.main.locations}
    colors={linearGradient.main.colors}
    style={styles.main}
  >
    <CommonView>
      <View style={styles.svgContainer}>
        <LogoIcon
          width={calcHeight(52.08)}
          height={calcHeight(51.88)}
          fill={white}
        />
        <View style={{width: calcHeight(10)}} />
        <LabelIcon
          width={calcHeight(162)}
          height={calcHeight(25)}
          fill={white}
        />
      </View>
      <View style={styles.inputMargin}>
        <SimpleTextInput
          value={email}
          onChangeText={text => {
```

```
      dispatch(setEmail(text));
      setEmailErrors(validateEmail(text));
    }}
    placeholder="E-mailadres"
   />
   {emailErrors && (
    <ErrorInput errorText="Vul een correct e-mailadres in" />
   )}
   <SimpleTextInput
    value={password}
    onChangeText={text => {
      dispatch(setPassword(text));
      setValidPass(validatePass(text));
    }}
    placeholder={translate('wachtwoord')}
    password
    withoutTogglePass
   />
   {isValidPass && (
    <ErrorInput
      errorText={translate(
        'wachtwoord moet uit minstens 6 tekens bestaan',
      )}
    />
   )}
  </View>
  <View style={styles.buttonContainer}>
   <CommonButton
    onPress={() => {
      dispatch(
        fetchLogin({
          grant_type: 'password',
          password: password,
          username: email,
        }),
```

```
            );
          }}
          text="Inloggen"
          disabled={!(!isValidPass && !emailErrors)}
        />
      </View>
      <TouchableOpacity onPress={() => navigate('ForgotPasswordScreen')}>
        <Text style={[styles.textStyle, {fontSize: calcFontSize(13)}]}>
          {translate('Login')}
        </Text>
      </TouchableOpacity>
    </CommonView>
  </LinearGradient>
 );
};

const styles = StyleSheet.create({
 main: {
  flex: 1,
  paddingHorizontal: calcWidth(20),
 },
 buttonContainer: {
  flex: 1,
  alignItems: 'center',
  marginTop: calcHeight(10),
  marginBottom: calcHeight(30),
 },
 inputMargin: {marginTop: calcHeight(50)},
 svgStyle: {
  fontSize: calcFontSize(40),
  textAlign: 'center',
  color: white,
  marginTop: calcHeight(100),
  backgroundColor: '#927792',
 },
```

```
 textStyle: {
   fontFamily: FONTS.Poppins.Medium500,
   marginTop: calcHeight(10),
   fontSize: calcFontSize(16),
   color: white,
   textAlign: 'center',
   marginBottom: calcHeight(15),
 },
 svgContainer: {
   flex: 1,
   flexDirection: 'row',
   alignItems: 'flex-end',
   justifyContent: 'center',
   marginTop: calcHeight(80),
 },
});

export default LoginScreen;
```

A.4 Http.hook.js

```
import {useState, useCallback} from 'react'
export const useHttp = () => {
  const [loading, setLoading] = useState(false)
  const [error, setError] = useState(null)
  const request = useCallback(async (url, method = 'GET', body = null, headers = {}) => {
    setLoading(true)
    try {
      if (body) {
        body = JSON.stringify(body)
        headers['Content-Type'] = 'application/json'
      }
```

```
    const response = await fetch(url, {method, body, headers})
    const data = await response.json()
    if (!response.ok) {
      throw new Error(data.message || 'Что-то пошло не так')
    }
    setLoading(false)
    return data
  } catch (e) {
    setLoading(false)
    setError(e.message)
    throw e
  }
}, [])
const clearError = useCallback(() => setError(null), [])
return { loading, request, error, clearError }
```

ДОДАТОК Б

Посилання на Git

https://github.com/PrOsToIOpA/diplom-2