Київський національний університет імені Тараса Шевченка

Факультет інформаційних технологій

Кафедра програмних систем і технологій

УДК 004.942

На правах рукопису

ВИПУСКНА КВАЛІФІКАЦІЙНА БАКАЛАВРСЬКА РОБОТА

Тема: “ Створення додатку для маркетплейсу домашнього харчування ”

Спеціальність – 121 “Інженерія програмного забезпечення”

ПОЯСНЮВАЛЬНА ЗАПИСКА

БР.ІПЗ - 22.00.00.000

Студент

ІПЗ-42 _____ /Максим ТЕОДОРОВИЧ/

Науковий керівник

к. ф.-м. н., доц._____ /Ольга СУПРУН/

Консультант
з питань нормоконтролю

_____ /Тамара ЧАПОВСЬКА/

Допускається до захисту
Завідувач кафедри

д.т.н., проф. _____ /Олексій БИЧКОВ/

Київ – 2021

Рішенням Екзаменаційної комісії
випускна кваліфікаційна робота студента

_____

захищена з оцінкою

_____

Голова Екзаменаційної комісії
д. т. н., проф. Віктор ВИШНІВСЬКИЙ

Київський національний університет імені Тараса Шевченка
Факультет інформаційних технологій
Кафедра програмних систем і технологій
Освітньо-кваліфікаційний рівень бакалавр
Спеціальність 121 “Інженерія програмного забезпечення”

ЗАТВЕРДЖЕНО
Зав. кафедри програмних систем і технологій
_____(Олексій БИЧКОВ)

ЗАВДАННЯ

НА ВИПУСКНУ КВАЛІФІКАЦІЙНУ БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Теодоровичу Максиму Сергійовичу

1. Тема бакалаврської роботи “Створення додатку для маркетплейсу домашнього харчування” , керівник роботи Ольга СУПРУН затверджені на засіданні кафедри програмних систем і технологій, протокол №6 від **“** 11 ” листопада 2020 р.

2. Строк подання студентом роботи 01 червня 2021 р.

3.Вихідні дані до роботи Аналітична інформація використання мобільних додатків у сфері приготуванні їжі завдяки третім особам

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

1. Оглаяд методів розробки серверної частини

2. Написання архітектури додатка

3. Розробка серверної частини

4. Розробка мобільного додатка на дві платформи

5. Перелік графічного матеріалу (з точним зазначенням обов’язкових креслень)

1.Схема роботи потоків на сервері Node.js

2.Схематичне зображення роботи Middleware

6. Консультанти з роботи із зазначенням розділів роботи, що їх стосуються

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв
РОЗДІЛ 1	Ольга СУПРУН		
РОЗДІЛ 2	Ольга СУПРУН		

РОЗДІЛ 3	Ольга СУПРУН		

Дата видачі завдання 11 листопада 2020 р.

Керівник _____ (Ольга СУПРУН)

Завдання прийняв до виконання _____ (Максим ТЕОДОРОВИЧ)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назви етапів бакалаврської роботи	Термін виконання етапів роботи	Відмітка про виконання
1.	Уточнення постановки задачі	25.02.2021-01.03.2021	Виконано
2.	Аналіз літератури	02.03.2021-03.03.2021	Виконано
3.	Аналіз існуючих методів вирішення завдання	04.03.2021-10.03.2021	Виконано
4.	Обґрунтування вибору рішення	11.03.2021-15.03.2021	Виконано
5.	Написання алогритмичної моделі	16.03.2021-26.03.2021	Виконано
6.	Опис розробленого алгоритму	27.03.2021-01.04.2021	Виконано
7.	Розроблення програмного забезпечення	02.04.2021-02.05.2021	Виконано
8.	Тестування розробленого програмного забезпечення	03.05.2021-07.05.2021	Виконано
9.	Оформлення і друк пояснювальної записки	08.05.2021-10.05.2021	Виконано
10.	Оформлення презентації	11.05.2021-18.05.2021	Виконано
11.	Отримання рецензії	01.06.2021	
12.	Затвердження пояснювальної записки роботи завідувачем кафедри	11.06.2021	
13.	Захист дипломної роботи	22.06.2021	

Студент – бакалавр _____(Максим ТЕОДОРОВИЧ)

Керівник роботи _____(Ольга СУПРУН)

АНОТАЦІЯ

Випускна кваліфікаційна бакалаврська робота: 63 с., 2 рис., 1 табл., 7 додат., 6 джерела.

Тема: Створення додатку для маркетплейсу домашнього харчування.

Об'єкт дослідження: методика розробки програмного забезпечення.

Мета роботи: надати можливість людям створити свій персональний ресторан в будь-якому місці з мінімальними вкладеннями, споживачам мінімізувати витрати на придбання якісної готової продукції з можливістю доставки в будь-який час і в будь-яке місце.

Предмет дослідження: додаток для маркетплейсу домашнього харчування.

Результати дослідження:

Досліджено можливості створення додатка, на мові програмування JavaScrip, для двох операційних систем: Android, IOS. З'ясовано методику підключення бази даних MongoDB до серверу.

Висновок:

В результаті досліджень було створено мобільний додаток на мові JavaScript, також створення серверна частина, яка працює на AWS EC2. Завдяки цьому додатку люди матимуть змогу зареєструвавшись у системі, продавати приготовлені власноруч їжу.

СИСТЕМА ДОМАШНЬОГО ХАРЧУВАННЯ, ДОДАТОК ДЛЯ ПРОДАЖУ ЇЖІ, МАРКЕТПЛЕЙС ДОМАШНЬОГО ХАРЧУВАННЯ

АННОТАЦИЯ

Выпускная квалификационная бакалаврская работа: 63 с., 2 рис., 1 табл., 7 доп., 6 источника.

Тема: Создание приложения для маркетплейсу домашнего питания.

Объект исследования: методика разработки программного обеспечения.

Цель работы: дать возможность людям создать свой персональный ресторан в любом месте с минимальными вложениями, потребителям минимизировать затраты на приобретение качественной готовой продукции с возможностью доставки в любое время и в любое место.

Предмет исследования: приложение для маркетплейсу домашнего питания.

Результаты исследования:

Исследованы возможности создания приложения на языке программирования JavaScript, для двух операционных систем: Android, IOS. Выяснено методику подключения базы данных MongoDB к серверу.

Вывод:

В результате исследований было создание мобильное приложение на языке JavaScript, также создание серверная часть, которая труда и на AWS EC2. Благодаря этому приложении люди смогут зарегестрировавшись в системе, продавать приготовленные собственноручно блюда.

СИСТЕМА ДОМАШНЕГО ПИТАНИЯ, ПРИЛОЖЕНИЕ ДЛЯ ПРОДАЖИ ЕДЫ, МАРКЕТПЛЕЙС ДОМАШНЕГО ПИТАНИЯ

SUMMARY

Graduation qualification bachelor's thesis: 63 p., 2 figures, 1 tables, 7 ext., 6 sources.

Subject: Creating an application for the marketplace of home catering.

Object of the research: software development methodology.

Purpose of work: to enable people to create their own personal restaurant in any place with minimal investment, consumers to minimize the cost of purchasing high-quality ready-made products with the possibility of delivery at any time and to any place.

Subject of the study: application for marketplace home catering.

Research Findings:

Explored the possibility of creating an application in the programming language JavaScrip, for two operating systems: Android, IOS. Explored the technique of connecting MongoDB database to the server.

Conclusion:

As a result of the research was the creation of a mobile application in JavaScript, also the creation of a server in chatsktii. Thanks to this application, people will be able to register in the system and sell cooked food.

HOMEWORKS SYSTEM, APP FOR SELLING FOOD, MARKETPLACE HOMEWORKS

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

JSON — JavaScript Object Notation - це безсхемне текстове представлення структурованих даних, яке базується на парах ключ-значення та упорядкованих списках.

ПЗ - сукупність програм системи обробки інформації і програмних документів, необхідних для експлуатації цих програм.

ОС — це базовий комплекс програм, що виконує керування апаратною складовою комп'ютера або віртуальної машини; забезпечує керування обчислювальним процесом і організовує взаємодію з користувачем.

IOS - мобільна операційна система, яку Apple використовує для своїх iPhone.

БД – база даних.

СУБД–система управління базами даних.

СЛС–структурно-логічна схема.

Backend - програмно-апаратна частина сервісу. Backend відповідає за здійснення функціонування внутрішньої частини веб-сайту.

API - це набір програмного коду, який забезпечує передачу даних між одним програмним продуктом та іншим. Він також містить умови цього обміну даними.

AJAX –Asynchronous JavaScript AndXML -підхід до побудови користувацьких веб інтерфейсів.

ECTS–European Credit Transfer and Accumulation System (Європейська система переведення і накопичення кредитів).

ANSI–American National Standards Institute (Американський національний інститут стандартів).

XML–eXtensible Markup Language (розширювана мова розмітки).

ВСТУП

Харчування, є неодмінною і першчерговою потребою будь-якої людини. Існують ресторани які наймають співробітників для готування їжі, доставлення цієї продукції споживачеві, витрачають гроші на оренду приміщення та інші витрати. З цієї причини ціна на продукцію така висока. Своєю чергою існує альтернатива у вигляді напівфабрикатів, які коштують недорого, але програють в загальній якості продукту. Основний клієнтський сегмент платформи ділиться на покупців і продавців. У кожен з під сегментів входять люди, які з найбільшою ймовірністю захочуть скористатися послугами платформи. Якщо проаналізувати споживача, можна виділити ключові потреби того, що йому потрібно від готової їжі:

- Якість інгредієнтів.
- Смакові характеристики.
- Зовнішній вигляд продукту.
- Ціна.

Люди, які хотіли б готувати в ресторанах маючи кулінарні таланти, сидять вдома або працюють в інших сферах. Також велика кількість кухарів були звільнені через карантин, низьких зарплат, або з інших причин. Вони були змушені піти на іншу роботу, або залишитися зовсім без заробітку. Своєю чергою концепція платформи повністю змінює ставлення споживачів до виробництва їжі. Головне завдання цієї платформи змінити ринок харчування, змінити ставлення споживачів до їжі яка була приготовлена в домашніх умовах, зменшити органічні відходи, збільшити інтерес до біо їжі та дати можливість заробити людям які вміють добре готувати.

В даний момент на українському ринку не існує подібної платформи, перевагою також є інноваційний алгоритм продажу готової їжі на замовлення, що не застосовувалась раніше.

Мета і задачі дослідження:

- Розробка архітектури системи.
- Розробка серверної системи, з API для додатку.
- Збереження даних користувачів у базі даних.
- Розробка мобільного додатку на дві операційні системи.

Для досягнення вищевказаних завдані, мені знадобилися AWS EC2 для безперебійної роботи серверу, та доступу з різних куточків світу. Для написання додатку знадобився React Native, а для збереження зображень користувачів AWS S3, це хмарне сховище.

За результатами досліджень та розробок, проведених у бакалаврській роботі, підготовлено доповідь для участі в 8-ій Східно-Європейський конференції «Математичні та програмні технології Internet of Everything», що відбулася в м. Києві 14 травня 2021 р. та опубліковано тези.

РОЗДІЛ 1
ПРИЗНАЧЕННЯ І ГАЛУЗЬ ЗАСТОСУВАННЯ

Програма призначена для людей, які мають можливість монетизувати свої кулінарні ідеї. В Україні понад 50 великих населених пунктів, в кожному з них проживає понад 100 тисяч осіб. Концепція побудована таким чином, що продукція яка приготовлена за всіма стандартами, має прийнятну вартість, знайде кінцевого споживача на нашій платформі і дасть приріст нових користувачів виробникам даної продукції.

Основний клієнтський сегмент для якого ми представляємо цінність ділиться на покупців і продавців. У кожен з під сегментів входять люди, які з найбільшою ймовірністю хочуть скористатися послугами платформи.

1.1. Галузь економіки та її перспективи

Програма відноситься до третинного сектору економіки. Платформа функціонує в сфері послуг. Надані послуги дозволяють зв'язати покупця і виробника з усіма умовами для здійснення угоди. За статистикою видно що 20% (7.458млн людина) населення з різною періодичністю харчується в ресторанах і кафе, а також до 2020 року 58% замовлень на продукти по всій країні буде здійснюватися через Інтернет. Для платформи це дає можливість залучити максимальну кількість людей з великим потенціалом для масштабування. Основна спрямованість даної платформи полягають в надання якісної екосистеми для користувачів і продавців, яка буде поєднувати в собі якість і легкість придбання.

1.2. Застосування додатку у реальному світі

Додаток є сервісом з продажу їжі. Унікальність ціого полягає в об'єднанні різних продуктів харчування від людей або компаній на одному майданчику. По суті послуги спрощують запуск невеликих приватних підприємств, а також дають можливість додаткового заробітку будь-якій людині.

Основні послуги додатку які були реалізованні:

- Реалізація готової їжі.
- Оформлення замовлень.
- Створення раціонів харчування.
- Доставка їжі.

Реалізація готової їжі та попередніх замовлень на страви:

Продавець отримує можливість реалізувати свою продукцію для максимальної аудиторії людей з мінімальними витратами. Це означає, що додаток бере всі незручності пов'язані з організацією, а продавець зможе зосередитися на якості своєї продукції. Послуга досить обширна і дозволить продати товари вже готові, або ж ті, які будуть зроблені на майбутнє за рахунок переміщення в різні категорії для зручного пошуку покупцем. З боку покупця функція дозволить заощадити гроші на харчуванні і отримати велику кількість пропозицій на ринку їжі, а також зручний інтерфейс, різноманітні фільтри для пошуку цікавлять страв. До всього покупець не обмежений лише їжею на передзамовлення, він може зробити замовлення на готову їжу поблизу в будь-який час.

Створення раціонів харчування:

Покупець сервісу може скласти собі раціон харчування за своїми уподобаннями, таким чином людина заздалегідь планує свої замовлення для зручності користування.

Доставка їжі:

Для покупців у платформі передбачений радіус замовлення готової їжі, тобто покупець побачить пропозиції поруч з ним і зможе їх замовити. Такий вид доставки дозволить додати можливість заробітку на швидких замовленнях знизити витрати, спростити схему і не втратити власну унікальність.

1.3. Маркетинг і збут продукції

Покупці це найчисленніша частина платформи. Основні покупці харчуються в кафе, ресторанах, фудкорти, замовляють комплексні обіди. Також люди які замовляють їжу через glovo, zakaz.ua, Bolt, Raket.

Продавець

Цільова аудиторія:

- Домохазяїни
- Кухарі які тимчасово не працюють
- Люди, які перебувають у декретній відпустці, або тимчасово не працюють.
- Приватні підприємства випускають крафтового продукцію.

Переваги:

- Можливість відкрити свій онлайн ресторан, без істотних інвестицій і тимчасових витрат.
- Зменшення витрат на оренду приміщення і на співробітників, зменшення собівартості продукту.
- Можливість швидкого зростання аудиторії.
- Легкий старт продажів.

Покупець

Цільова аудиторія:

- Люди які хотіли б платити менше за готову їжу
- Мілленіали
- Аудиторія яка любить домашню їжу
- Аудиторія яка воліє правильне харчування
- Люди у яких недостатньо часу щоб займатися цим удома
- Люди які бажають замовити складний в приготуванні продукт за меншою ціною Переваги:

Переваги:

- Можливість замовити готову їжу за оптимальними цінами
- Замовлення продукції по персональним вимогам (час, продукт, кількість, специфікації)
- Платформа надає легкий і зручний доступ для покупки.
- Великий вибір страв і продавців
- Широкий спектр фільтрів, налаштувань для пошуку товару, що цікавить або продавця, а також система рейтингового оцінювання.

РОЗДІЛ 2
ТЕХНІЧНІ ХАРАКТЕРИСТИКИ

2.1. Формулювання завдання на розробку системи

Розробити систему, яка могла б зареєструвати користувача у системі. Надати змогу використовувати всі послуги, якщо користувач має автомобіль. Послуги включають в себе:

- Систему авторизації та валідації користувача.
- Систему повідомлень.
- Систему обробки інформації.
- Систему математичного підрахунку рейтингу користувача.
- Система пошуку.
- Система створення їжі.
- Система підбору найкращих послуг.

Опис алгоритму

Для рішення поставленої задачі потрібно заздалегідь зібрати базу інформації про ціни на ринку домашнього харчування. Потім завдяки цьому створити модель для продавців. Після цього, можна приступати до розробки ПО. По перше потрібно розробити архітектуру ПЗ, завдяки цій архітектурі потрібно створити BackEnd, якій буде обробляти, та буде «головою» усього додатку. Для візуальної частину потрібно зробити мобільний додаток, який буде працювати з BackEnd.

2.2. Опис і обґрунтування вибору технічних та програмних засобів

BackEnd виконувався в Atom - це текстовий редактор з відкритим вихідним кодом, який надається GitHub, який зараз належить Microsoft. це альтернатива розробнику для використання більш простих текстових редакторів, таких як блокнот або блокнот плюсплюс. На відміну від notepad plus plus та Notepad, Adam надає додаткові переваги для розробників, такі як підсвічування синтаксису для певних блоків коду та можливість переглядати всю структуру вашого проекту,

щоб ви могли переходити між файлами набагато швидше, ніж якщо ви просто редагувати на блокноті в блокнот плюс плюс. Atom заснований на Electron (раніше відомий як Atom Shell) - фреймворку кросплатформної розробки з використанням Chromium і io.js. Редактор написаний на CoffeeScript і LESS. Версія 1.0 була випущена 25 червня 2015 р.

Для зберігання даних було використано MongoDB тому що NodeJS май дуже гарний зв'язок і дає легкий доступ до бази даних. MongoDB - це програма управління базами даних NoSQL з відкритим кодом. NoSQL використовується як альтернатива традиційним реляційним базам даних. Бази даних NoSQL досить корисні для роботи з великими наборами розподілених даних. MongoDB - це інструмент, який може керувати орієнтованою на документи інформацією, зберігати або отримувати інформацію. MongoDB підтримує різні формати даних. Це одна з багатьох реляційних технологій баз даних, що виникла в середині 2000-х під прапором NoSQL - як правило, для використання в додатках великих даних та інших робочих процесах, що включають дані, які не вкладаються в жорстку реляційну модель. Замість використання таблиць і рядків, як у реляційних базах даних, архітектура MongoDB складається з колекцій та документів. Підтримується пошук за регулярними виразами. Також можна налаштувати запит на повернення випадкового набору результатів. Існує підтримка індексів.

Для зберігання фотографій, було обрано aws s3. Головна перевага S3-сховища - можливість роботи з будь-якої географічної точки з файлами будь-якого типу і обсягу. Принцип роботи S3 простий: дані містяться в безліч контейнерів (папок). Вміст будь-якого контейнера можна переглядати, переміщати або видаляти. У кожного контейнера і об'єкта є адреса у вигляді унікального ідентифікатора. Він використовується, як ключ доступу до даних. Ці ключі можуть мати будь-який строкове значення. При необхідності, можна зробити так, щоб ключі містили важливу для роботи інформацію - наприклад, приналежність

об'єкта до якого-небудь проекту. Також об'єктів в кожному контейнері можна призначити теги. Кожен об'єкт може мати кілька тегів різного виду. Це стане в нагоді, якщо потрібно описати його максимально точно. Наприклад, можна створити аудіофайл з тегами імені виконавця, назв пісень, назви альбому та іншої інформації. Надалі ці метадані індексуються, що значно полегшує і в рази прискорює пошук потрібних об'єктів за заданими ознаками. Крім того, при роботі з ними немає заплутаною ієрархічної файлової системи зі складними і довгими адресами. Важливий момент: працюючи з S3-сховищем, ви знаєте, де і в якому вигляді зберігаються ваші дані, хто може отримувати до них доступ і які ресурси використовує ваша організація в будь-який момент часу. Засоби ідентифікації і контролю доступу в поєднанні з безперервним моніторингом даних гарантують безпеку.

Для написання мобільного додатку було обрано мову програмування JavaScript, та написання кросплатформенного додатку використовуючи React Native. Його основне завдання - забезпечити відображення на екрані того, що можна побачити на веб-сторінках. React дозволяє легко створювати інтерфейси, розділяючи кожну сторінку на невеликі фрагменти і компоненти. Він дуже зручний для створення веб-додатків і не вимагає великого порогу входження.

Мінімальні вимоги до складу технічних засобів

Для запуску BackEnd потрібно:

- Процесор – Pentium II с частотой 300 МГц
- Оперативна пам'ять – 128 Мб
- 10 Мб доступного повстанства на жёсткому диску
- Монітор – 15’’ Color VGA

Для запуску мобільного додатку потрібен любий Iphone з встановленим IOS-8 не менше, або телефон з операційною системою Android з версією 5+.

2.3. Використання бібліотек та менеджерів пакетів

Один з найважливіших модулів для програмування на NodeJs та React Native, npm - це менеджер пакетів, який входить до складу Node.js. Протягом багатьох років Node широко використовувався розробниками JavaScript для обміну інструментами, установки різних модулів і управління їх залежностями. Ось чому людям, які працюють з Node.js, дуже важливо зрозуміти, що таке npm.

Npm працює, виконуючи одну зі своїх двох ролей:

- Це широко використовуваний репозиторій для публікації проектів Node.js з відкритим вихідним кодом. Це означає, що це онлайн-платформа, де кожен може публікувати та ділитися інструментами, написаними на JavaScript.
- npm - це інструмент командного рядка, який допомагає взаємодіяти з онлайн-платформами, такими як браузери і сервери. Ця утиліта допомагає в установці і видаленні пакетів, управлінні версіями і залежностями, необхідними для запуску проекту.

Декілька слів про використання React Native. Інформація з офіційного сайту: «React Native дозволяє створювати мобільні додатки, використовуючи при цьому тільки JavaScript з такою ж структурою, що і у React. Це дає можливість складати багатофункціональний мобільний UI із застосуванням декларативних компонентів». Отже, виходить, що React Native - це фреймворк, в основі якого лежить React.js, що дозволяє розробляти Кросплатформені додатки як для Android, так і для iOS.

Переваги використання кроссплатформи:

1. Кроссплатформенная розробка

Основна мета розробників - надати клієнтам сервіси. Ніхто не хотів би, щоб його користувачі були обмежені лише однієї якоїсь платформою тільки тому, що розробник не може створювати додатки для інших платформ. Отже, і сам

розробник не повинен обмежувати свої здібності тільки тому, що йому або їй комфортно працювати з конкретним інструментом розробки. Фреймворк React Native є портативним, тобто його єдина кодова база, написана в JavaScript, створить модулі як для Android, так і для iOS.

2. Освоєння React

Освоївши React Native і JavaScript, ви відкриєте для себе новий світ front-end розробки стосовно, наприклад, до веб-сайтам. Фреймворк React Native заснований на тих же компонентах, що і React, тому отримані тут навички не обмежуються тільки розробкою мобільних додатків.

3. Час збірки швидше, ніж в Android Studio

Ви коли-небудь витрачали більше 2-3 хвилин на складання, щоб протестувати / пофиксить базову функцію, і при цьому багфикс розтягувався на довгі години? Рішенням проблеми стане React Native. З ним на збірку йде значно менше часу. З такою функцією, як «Гаряча перезавантаження» (Hot Reloading), розробка і тестування користувальницького інтерфейсу - це легко. Завдяки цій функції додаток перезавантажується кожен раз, коли JS-файл зберігається!

4. JavaScript зручний для передачі даних по мережі

У React Native виклик API, рендеринг зображень по URL і інші процеси дуже прості. Більше не потрібно використовувати Retrofit, OkHttp, Picasso і т. Д. Набагато менше часу витрачається на налаштування. Коли дані надходять з API на платформі Android, вони спочатку перетворюються в POJO-модель і лише потім використовуються в елементах UI. А ось дані JSON, отримані в React Native, зручні для JavaScript і можуть безпосередньо використовуватися для попереднього перегляду UI. Це дозволяє полегшити веб-інтерфейс для GET або POST-запитів від REST API.

5. Розробка UI

У React Native як розмітки UI виступає модуль flexbox, серйозний конкурент XML-розмітки на Android. Flexbox дуже популярний в співтоваристві веб-розробників. У React Native UI-елементи в основному повинні розроблятися з нуля, тоді як в нативної розробці для Android бібліотека підтримки Google Design Support Library вже підключена. Це дає розробнику свободу в плані інтерактивного і адаптивного дизайну.

Недоліки використання кроссплатформи:

1. Можливо, ви ненавидите JavaScript

Багато людей не люблять JavaScript просто за те, що ця мова не схожий на традиційні мови, такі як Java, C ++ та інші. Докладні негативні відгуки ви можете знайти тут і тут.

2. Не так вже й багато сторонніх бібліотек

Спільнота React Native як і раніше знаходиться в стадії становлення і підтримує сторонні бібліотеки, не такі популярні, як нативна бібліотека Android (до речі, оцініть слайд-шоу моєї бібліотеки для Android).

2.4. Використання об'єктно орієнтованого проектування

Під патернами проектування розуміється опис взаємодії об'єктів і класів, адаптованих для розв'язання загальної задачі проектування в конкретному контексті. Застосування патернів проектування дозволяє істотно знизити час на створення ієрархій класів розроблюваної системи, тому що патерни описують способи вирішення широкого класу задач. Приклад патернів програмування які я використовував у своїй курсовій роботі:

- Фабричний метод - це породжує патерн проектування, який визначає загальний інтерфейс для створення об'єктів в супер класі, дозволяючи подклассам змінювати тип створюваних об’єктів.

- Абстрактна фабрика - це породжує патерн проектування, який дозволяє створювати сімейства пов'язаних об'єктів, не прив'язуюсь до конкретних класах створюваних об’єктів.
- Міст - це структурний патерн проектування, який розділяє один або кілька класів на дві окремі ієрархії - абстракцію і реалізацію, дозволяючи змінювати їх незалежно один від одного.
- Відвідувач - це поведінковий патерн проектування, який дозволяє додавати в програму нові операції, не змінюючи класи об'єктів, над якими ці операції можуть виконуватися.

Поряд з цими існує чимала кількість і інших патернів проектування. Я опишу патерн проектування «singleton», ідея якого реалізована в розробленій програмі.

2.5. Процес аналітичної ієрархії

Процес аналітичної ієрархії (MAI) - це математичний інструмент вирішення проблем, який став популярним серед управлінського персоналу наприкінці 1990-х - на початку 2000-х років. Метод MAI був створений після розуміння структури проблеми та реальних перешкод, з якими стикаються менеджери при її вирішенні.

Логіка MAI

Метод MAI розглядає проблему у трьох частинах. Перша частина - це проблема, яка потребує вирішення, друга частина - альтернативні варіанти вирішення проблеми. Третя і найважливіша частина щодо методу MAI - це критерії, що використовуються для оцінки альтернативних рішень.

Метод MAI розуміє, що хоча існує кілька критеріїв, величина кожного критерію може бути не рівною. Наприклад, якщо вам доводиться вибирати між двома ресторанами, смак і час очікування - це два фактори, проте обидва вони можуть не мати однакового значення у вашому сприйнятті. Смак може бути набагато важливішим за час очікування тощо. Тому, якщо ви призначаєте вагу 2 за

смаком і 1 часу очікування, ви, швидше за все, приїдете до ресторану, який найкраще задовольнить ваші потреби.

Отже, під час оцінки альтернативних рішень, до критеріїв слід додати ваги, щоб забезпечити правильний висновок. Це може здатися очевидним. Однак до самого пізнього часу вчені управління стикалися з питаннями, як призначати ваги. У наведеному вище прикладі наше призначення ваг було довільним. Також приклад мав лише два критерії. Зі збільшенням кількості критеріїв (факторів) призначення стають все більш довільними.

Метод MAI має вбудовані механізми стримування та противаги. Таким чином, ці стримування та противаги забезпечують досягнення логічно послідовних рішень при порівнянні відносної важливості критеріїв у процесі присвоєння їм ваг.

2.6. Scrum

Для створення додатку було обрано Scrum підхід, мета якого якісний процес котролю додатку. Це різко контрастує з більш традиційним підходом у стилі водоспад, який заздалегідь фіксує обсяг проекту, вимагаючи широкого створення вимог, аналізу та проектної документації перед початком розробки. Затримки та перевитрата бюджету є загальним явищем, і неможливість встановити пріоритети набору функцій часто призводить до продуктів низької якості, які перевантажені функціями, які клієнт / користувач насправді не вимагає.

Як працює управління проектами scrum

Scrum-підхід до управління проектами дозволяє організаціям, що розробляють програмне забезпечення, визначити пріоритетну роботу, яка має найбільше значення, і розбити її на керовані частини. Scrum - це співпраця та спілкування як з людьми, які виконують роботу, так і з людьми, яким потрібна виконана робота. Йдеться про часті доставки та реагування на відгуки, підвищення цінності бізнесу шляхом забезпечення того, щоб клієнти отримували те, що насправді хочуть. Перехід від традиційних підходів до управління проектами до

управління проектами Scrum вимагає коригування з точки зору виконуваної діяльності, створених артефактів та ролі в команді проекту.

РОЗДІЛ 3
АЛГОРИТМ СИСТЕМИ

Частково алгоритм роботи комплексу вже був розглянутий в попередніх параграфах. Тому в цьому параграфі зупинимось тільки на деяких моментах. Раніше було зазначено, що система складається с трьох малих підсистем: BackEnd, додаток та веб версія для рекламодавців.

Інтерфейс є «зв'язком із зовнішнім світом». Він очікує в циклі приходу будь-якої команди, періодично перевіряючи її наявність. Така поведінка є стандартним для додатків, написаних з використанням API. Коли приходить команда, відбувається її розбір, тобто визначається тип і, відповідно, дія, яку потрібно виконати (тобто якого модулю передати управління).

Коли користувач обирає якусь команду, ініціюється запуск модуля. Програмна реалізація така, що для роботи модуля створюється окрема нитка (легкий процес). Під ниткою розуміється незалежна послідовність виконання програмного коду всередині окремого процесу. Нитки всередині процесу розподіляються і виконуються абсолютно незалежно, тобто якщо одна нитка очікує введення інформації, це ніяким чином не перериває виконання інших ниток. Зазвичай нитка виконується, поки не буде заблокована будь-яким запитом або поки не закінчиться відведений їй відрізок часу (квант часу). Таким чином, робота модуля, яка є досить ресурсномісткою, не буде «підвищувати» додаток. Крім цього, виконання нитки можна перервати в будь-який момент. Ці особливості нитки як не можна до речі підходять для реалізації модуля.

3.1. Архітектура програмної системи

Правильно закладена архітектура програмної системи є результатом успішно розробленого продукту. Оскільки мобільний додаток є лише частиною всієї системи, до якої входять API, яке надає методи створення, редагування та видалення об’єктів даних. Головним каркасом системи э бекенд. Я хотів би

зауважити, що майже скрізь додаток вузла розробляється поверх іншого фреймворку, тобто Express.js, який також є простим рішенням маршрутизації для http. Зараз важливе питання полягає в тому, як знайти кращий спосіб упорядкування файлів та даних у системі, таким чином, щоб проектом стало легко керувати, а також включати хороші принципи програмування.

1. Тип безпеки
2. Поділ турботи
3. Інкапсуляція функцій
4. Краще оброблення помилок
5. Краще управління реакцією
6. Краще управління обіцянками
7. Надійні одиничні випробування
8. Проста можливість розгортання

Отже бекенд-програма тримається незалежно від будь-якого інтерфейсу. Інтерфейс, як веб-сайти та мобільний додаток, може викликати API загальної серверної бази для служб. Це сховище проектів з відкритим кодом робить те саме. У багатопотокових установках обробки кожен сервер має у своєму розпорядженні обмежений пул потоків. Щоразу, коли сервер отримує запит від клієнта, він вибирає потік з пулу і призначає його запиту клієнта. Цей потік забезпечить всю обробку, пов’язану з цим запитом. Усередині цих потоків обробка носить послідовний та синхронний характер, тобто виконується одна операція за раз. Незалежно від того, коли новий одночасний запит надходить на сервер, він може забрати будь-який доступний потік з пулу і ввести його на чергування.

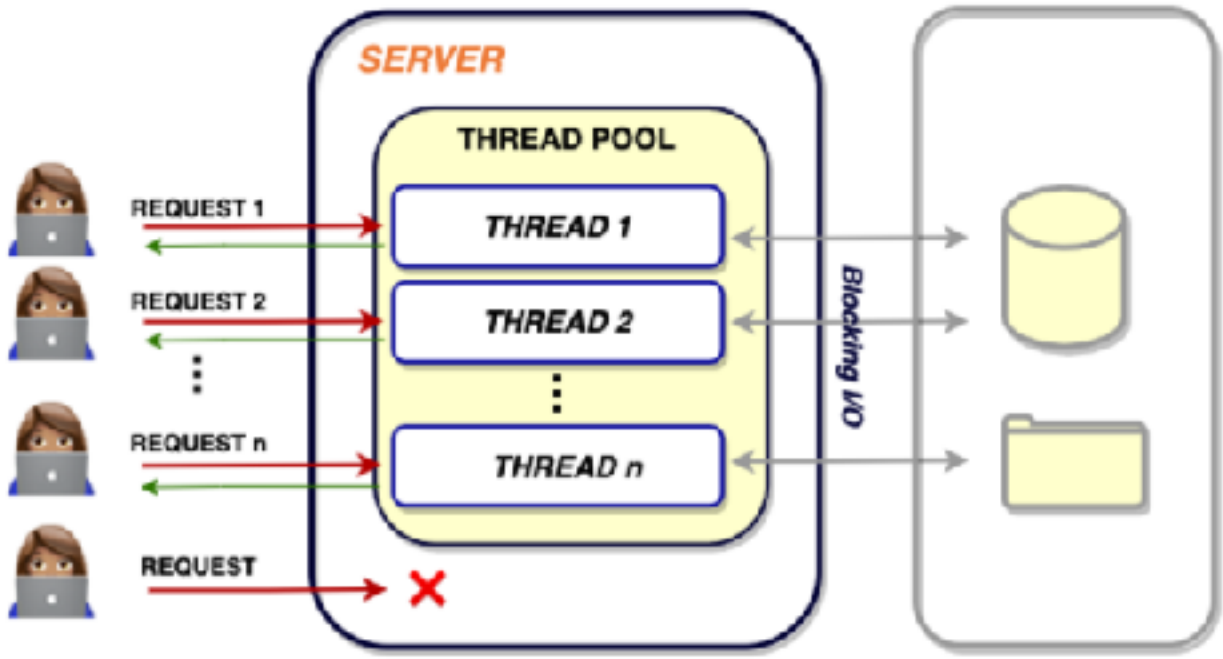


Рис. 3.1. Схема роботи потоків на сервері Node.js

Цикл подій може сам подбати про базову обробку, але для асинхронних операцій вводу-виводу, що включають такі модулі, як fs (I / O-heavy) та crypto (CPU-heavy), він може завантажити обробку до робочого пулу системне ядро. Робочий пул реалізований у libuv і може створювати та керувати кількома потоками відповідно до вимоги. Ці потоки можуть індивідуально запускати відповідні призначені завдання синхронно і повертати свою відповідь у цикл подій, коли вони будуть готові. Поки ці потоки працюють за призначеними їм операціями, цикл подій може продовжувати працювати в звичайному режимі, одночасно забезпечуючи інші запити. Коли потоки закінчать зі своїми завданнями, вони можуть повернути свої результати у цикл подій, які потім можуть повернути це назад у чергу виконання, яку потрібно виконати, або повернути назад клієнту.

Процес продумування такої архітектури можна пояснити тим, що при типових веб-навантаженнях один основний потік може працювати і масштабуватися набагато краще в порівнянні зі звичайними архітектурами “один

потік на запит". Як результат, Node.js - це варіант для багатьох, оскільки він має переваги щодо швидкості та масштабованості. Проте застереження тут полягає в тому, що продуктивність може постраждати від заздалегідь складних, інтенсивних операцій, таких як множення матриць для обробки зображень, науки про дані та програм машинного навчання. Вони можуть заблокувати єдиний основний потік, роблячи сервер невідповідним. Однак для таких випадків Node.js також представив робочі потоки, які розробники можуть використати для створення ефективних багатопоточних програм Node.js.

3.2. Алгоритму мобільного дотатку для IOS

1. Перевірка на існування даних о користувачі у системі, якщо такі існують, користувач входить у систему.
2. Якщо не існують, то користувач має вибір:
    1) Реєстрація (Ім'я, прізвище, номер телефону, емейл, пароль).
    2) Вхід (Емейл, пароль).
3. Блокування всіх вікон, якщо у нього не підтверджений аккаунт.
4. Користувач може обрати один з трьох пунктів меню:
    1) Дім.
    2) Створити товар.
        I. Створити «Горячу їжу»
        II. Створити їжу на замовлення
    3) Аккаунт.
5. У вкладки «аккаунт», із системи підгружається дані про користувача, якщо ні то користувач повинин заповнити форму і система відправть інформацію на перевірку.
6. Також на цьому вікні користувач повинен підтвердити свіоє містоположення та пройти систему веріфікації, якщо він уперше у системі.
7. Знизу у ціому вікні, можна редагувати меню своїх товарів.
8. Вкладка «Дім» - це вкладка у якій корисувач має змогу подивитися свої «Горячі товари».

9. Вибравши товар, він може его змінити зарузити нову фотографію, або видалити цій товар.

10.Вікно стрворення «Гоячої їжі»

1) По перше користувач повине обрати головне фото для товара.
2) Заповнити поля: опис, названння, інгрідієнти, ціну, вагу.
3) У горячої їжі є срок придатності, тому користувач вказує діапазон дат, та годин придатності його товару.
4) Заповняє зображеннями галерею товара.

2. Вікно стрворення їжі на замовлення

1) По перше користувач повине обрати головне фото для товара.
2) Заповнити поля: опис, названння, інгрідієнти, ціну, вагу.
3) Заповняє зображеннями галерею товара.

3.3. Алгоритм BackEnd

1. Запуск сервера.
2. Считування протоколів та правил, підключення усіх модулів.
3. Створюється веб-сервер.
4. Створюється новий експрес-додаток, а потім використовується для створення http-сервера через модуль http.
5. Метод прослуховування сервера використовуємо для прив'язки до зазначеного порту і запуску прослуховування вхідних запитів.
6. Перевірка системи на працездатність.
7. Створення підключення з MongoDB.
8. Перевірка бази даних
9. Отримуємо запит.
    1) Система перевіряє метод відправки запиту.
    2) Перевіряємо шлях за яким було отримано запит.
    3) Дивимось на дані які були відправлені, та зберігаємо їх.
    4) Обробляємо запит.

10.Відправляємо відповідь.

1) Якщо запит «PUT» відповідь не потрібна.

2) Створюємо єкземпляр класа у який заздалегіть кладемо дані для відповіді.

3) Відправляємо данні.

РОЗДІЛ 4
СЕРВЕРНА ЧАСТИНА

Бэк-энд (англ. back-end) — програмно-апаратна частина сервиса. Якщо порівняти сайт з кораблем, то бекенд - це машинне відділення. BackEnd складається з трьох частин:

1. Сервера.
2. Додатки.
3. База даних.

Також BackEnd відповідає за:

- за зберігання та організацію даних.
- взаємодіє з інтерфейсом.
- відправляє і отримує інформацію, що відображається у вигляді веб-сторінки.

Для того щоб розробити BackEnd обрно мову програмування JavaScript, також середовище Node.js

4.1. Обгрунтування вибору Node.js

Можливість використовувати JavaScript для написання інтерфейсу та бекенда лежить в основі багатьох переваг Node.js:

- Легка крива навчання. Знання JavaScript дає розробнику хороший старт з Node.js.
- Велика громада. Node.js, будучи проектом з відкритим кодом, заохочує підтримку та внесок, спрямований на вдосконалення та впровадження платформи. Це місія його Фонду, призначеного для постійного розвитку та вдосконалення Node.js.
- Надійність. Використання Node.js дозволяє організувати повнотекстову розробку JavaScript, що забезпечує швидкість і продуктивність програми.

- Масштабованість. Це справжня коштовність середовища розробки Node.js, оскільки дозволяє створювати додатки, які можуть легко розвиватися разом із вашим бізнесом. Node.js чудово працює в системах, що використовують архітектуру мікросервісів або контейнеризацію, де масштабованість та гнучкість можна досягти швидко та легко.
- Чудова екосистема. npm (менеджер пакунків Node.js) для 650 000 безкоштовних пакетів коду.

Однониточні процеси. За своєю природою Node.js може одночасно обробляти лише одну команду. Однак блокування інших процесів дозволяє уникнути реалізації асинхронного механізму, що дозволяє одночасно виконувати кілька простих завдань. Однак складний розрахунок, що вимагає багато обробних ресурсів, може перекрити потік і спричинити затримки. Низька якість інструментів з відкритим кодом. npm як перевагу Node.js, однак у нього є і зворотна сторона. Хоча кількість модулів та пакетів, доступних у реєстрі, велика, і на вибір є з чого вибрати, деякі пакети можуть бути неякісними або погано задокументованими. Це не має нічого спільного з якістю основного коду Node.js, хоча надані інструменти іноді можуть бути неякісними.

Node.js має успішні основні моменти, які можуть бути використані різними посередницькими адміністраціями з контрастами у часі реакції. Його також можна використовувати для передачі інформації з різних джерел. Продуктивність повинна розглядатися не як одновимірна особливість, а як багатовимірна. Можна подумати, що розробник є більш продуктивним в Java через помилки під час компіляції. Підприємства можуть об’єднати команди інтерфейсів та серверних систем у єдиний блок, завдяки чому програма забезпечує менший час із високою продуктивністю.

4.2. Використання Node.js

Node.js представляє собою платформу для написання JavaScript-додатків з використанням зовнішніх бібліотек.

Завдяки Node.js написаний для браузера код JavaScript отримує доступ до глобальних об'єктів, таким як document і window, поряд з іншими API і бібліотеками. За допомогою Node код звертається до жорсткого диска, баз даних і Мережі. Це робить можливим написання абсолютно будь-яких додатків: від утиліт командного рядка і відеоігор до повноцінних веб-серверів.

Найчастіше Node.js використовується при написанні веб-додатків з інтенсивним введенням-виведенням. Найпоширеніший приклад - це веб-сервери. Node.js популярний для створення додатків реального часу: чатів, комунікаційних програм та ігор. Багато додатків Node.js мають і серверну, і клієнтську частини.

4.3. Npm

NPM - або "Node Package Manager" - це менеджер пакунків за замовчуванням для середовища виконання Node.js. NPM складається з двох основних частин: інструмент CLI (інтерфейс командного рядка) для публікації та завантаження пакетів, і Інтернет-сховище, яке розміщує пакунки JavaScript. npmjs.com це центр виконання, який приймає пакети товарів від продавців (автори пакунків npm) і розподіляє ці товари серед покупців (користувачі пакета npm). Щоб полегшити цей процес, у центрі реалізації npmjs.com працює npm CLI, якф буде призначена як особистий асистент кожному окремому клієнту npmjs.com.

- V8 прискорює JavaScript, використовуючи C ++
- V8 - движок з відкритим вихідним кодом, написаний на C ++.
- JavaScript -> V8 (C ++) -> машинний код
- V8 реалізує сценарій ECMAScript, як зазначено в ECMA-262. ECMAScript був створений Ecma International для стандартизації JavaScript

4.4. Запуск HTTP-серверу на Node.js, обробки запитів

Для початку було створено HTTP-додаток в Node.js, вбудовані модулі http/ https головні модулі для роботи сервера. Було підключенно модуль http і прив'язан до порту 8080.

Потрібно зазначити важливі аспекти:

- requestHandler: ця функція буде викликатися кожен раз, коли на сервер прийде запит.
- if (err): обробка помилок: якщо порт вже зайнятий або є якісь інші причини, за якими сервер не може бути запущений, ми отримаємо повідомлення

```
const http = require('http')
const port = 8080
const requestHandler = (request, response) => {
  console.log(request.url)
}
const server = http.createServer(requestHandler)
server.listen(port, (err) => {
  if (err) {
    return console.log('something bad happened', err)
  }
  console.log(`server is listening on ${port}`)
})
```

про це.

Модуль http вкрай низькорівневий: створення складного веб-додатки з використанням вищенаведеного фрагмента коду дуже занадто багато роботи. Саме з цієї причини було обрано фреймворки для роботи над проектом:

- express.

- Helmet .
- hapi.
- Morgan.
- koa.
- restify.
- Mongoose.

Для того, щоб розробити високорівневий BackEnd, було використатно Express і преобразиован http-сервер:

```
const express = require ('express')
const app = express ()
const port = 8080
app.get ('/', (request, response) => {
    response.send ('Drivvy BackEnd!')
});
app.listen (port, (err) => {
    if (err) {
        return console.log ('something bad happened', err)
    }
    console.log (`server is listening on $ {port}`)
})
```

Найбільша відмінність полягає в тому, що Express за умовчанням дає роутер. Не потрібно вручну розбирати URL, щоб вирішити, що робити, замість цього потрібно визначити маршрутизацію програми за допомогою app.get, app.post,

app.put і так далі, а вони вже транслюються в відповідні HTTP-запити. Одна з найпотужніших концепцій, яку реалізує Express, - це патерн Middleware.

4.5. Middleware – проміжний обробник

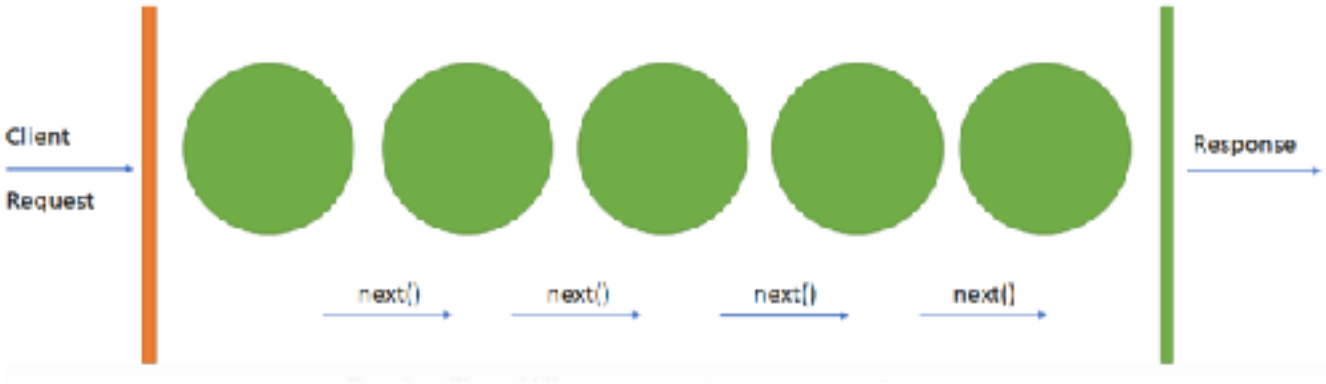


Рис. 4.1. Middleware

На рис.4.1 можна побачити, як запит йде через умовний Express-додаток. Він проходить через три проміжних обробники. Кожен обробник може змінити цей запит, а потім, грунтуючись на бізнес-логікою, третій middleware відправить відповідь, або запит потрапить в обробник відповідного роута. Слід зазначити:

- app.use: це те, як можна описати middleware. Цей метод приймає функцію з трьома параметрами, перший з яких є запитом, другий - відповіддю, а третій - коллбеком next. Виклик next сигналізує Express про те, що він може переходити до наступного проміжного оброблювачу.
- Перший проміжний обробник тільки логірує заголовки і миттєво викликає наступний.
- Другий додає додаткове властивість до запиту - це одна з найбільш потужних функцій шаблону middleware. Проміжні обробники можуть додавати додаткові дані до об'єкта запиту, який можуть зчитувати middleware.

4.6. Обробка помилок

Як і у всіх фреймворками, правильна обробка помилок має вирішальне значення. В Express потрібно створити спеціальний проміжний обробник - middleware з чотирма вхідними параметрами:

```
app.get('/', (request, response) => {
  throw new Error('oops')
})
app.use((err, request, response, next) => {
  console.log(err)
  response.status(500).send('Something broke!')
})
```

Оброблювач помилок повинен бути останньою функцією, доданої за допомогою app.use. Оброблювач помилок приймає коллбек next. Він може використовуватися для об'єднання декількох обробників помилок.

4.7. Підключення Node.js до MongoDB

Найбільш популярною системою управління базами даних для Node.js на даний момент є MongoDB. Для роботи з цією платформою перш за все необхідно встановити сам сервер MongoDB. Крім самого сервера Mongo для взаємодії з Node.js потрібен драйвер. При підключенні і взаємодії з бд в MongoDB можна виділити наступні етапи:

1. Підключення до сервера.
2. Отримання об'єкта бази даних на сервері.
3. Отримання об'єкта колекції в базі даних.
4. Взаємодія з колекцією (додавання, видалення, отримання, зміна даних).

Ключовим класом для роботи з MongoDB є клас MongoClient, через нього буде йти всі взаємодії зі сховищем даних. Щоб з'єднатися з сервером mongodb було застосовано метод *connect* ():

В кінці завершення роботи з бд за допомогою методу client.close ().

MongoDB зберігає дані в колекціях (collections), які повністю виправдовують свою назву. Дані які надходять до серверу хроняться у колекціях:

```
mongoose.connect(process.env.DB_CONNECT, { useNewUrlParser: true },
function (err) {
    if (err) {
        console.log("connection error:", err);
    } else {
        console.log("MongoDB connection successful");
    }
})
```

- Seller;
- Buyer;
- complaints
- dynamicpromos;
- menuproducts;
- payments
- products

4.8. Види запитів, які реальзовані на BackEnd

1. Реєстрація та логін рекламодавців та автомобілістів.
2. Перевірка email та номеру телефона на реальність.
3. Інформація об аккаунті.
4. Додавання нової інформації користувача.
5. Список продуктів у меню.
6. Зміненя продуктів у меню.
7. Видалення продуктів з меню.
8. Створення нового продукту до меню.
9. Створення нового «Готового продукту».
10. Оновлення фотографії у профілі користувача.

11. Оновлення, або додавання нового місце знаходженя користувач.
12. Список продуктів «Готової їжі».
13. Зміненя продуктів «Готової їжі».
14. Видалення продуктів «Готової їжі».
15. Додавання платіжної інформації користувача.

4.9. Опис функціонування серверної частини

BackEnd отримує запит на реєстрацію користувача, перевіряє MongoDB на наявність вже існуючий email або номеру телефона. Пароль користувача шифрується завдяки спеціальним протоколам та кодом. Потім при авторизації користувача перевіряється email та пароль з даними в базі даних. Також користувачу дається Токен, завдяки якому він може спілкуватися з BackEnd у наступних запитах. Запити у цих двух прикладах «POST». Коли драйвер зареєструвався у базі даних користувач май поле «status», цей статус інформує користувача щоб він заповнив інформацію. У рекламодавця є статус «email», при реєстрації користувача на почту відправляється підтвердження. Рекламодавець підтверджує почту статус «email» становиться «true», після цього він має можливість увійти у систему. З мобільного додатка сервер отримує багато запитів але їх можна класифікувати на 4:

1. Геопозиція.
2. Аккаун.
3. Замовлення.
4. Статуси.

4.10. Чат та сокети

Сокет - це стійкий двонаправлений зв’язок між двома комп’ютерами, як правило, один виконує роль клієнта, а інший - сервера (іншими словами: постачальника послуг та споживача).

З'єднання є постійним, що означає, що як тільки клієнт і сервер з'єднаються, кожне нове повідомлення, надіслане клієнтом, буде отримуватися точно тим же сервером. Це не стосується API REST, які мають бути без громадянства. Набір серверів REST із збалансованим навантаженням не вимагає (насправді, навіть не рекомендується) того самого сервера відповідати на запити того самого клієнта.

Зв'язок може розпочинатися сервером, що також є однією з переваг використання сокетів над REST. Це спрощує значну частину логістики, коли частину даних потрібно переміщати від сервера до клієнта, оскільки при відкритому сокеті немає інших необхідних умов, а дані просто перетікають з одного кінця в інший. Це також одна з функцій, яка робить сервери чату на основі сокетів таким простим і прямим випадком використання. Для використання REST або подібний протокол, знадобиться багато додаткового мережевого трафіку для запуску передачі даних між сторонами.

Після встановлення підключення до сокету клієнтські програми повинні вручну приєднатися до чату та певної кімнати всередині нього. Це означає, що клієнт надсилає ім’я користувача та назву кімнати як частину запиту, а сервер, крім усього іншого, веде облік пар ім’я користувача-сокет в об’єкті Map. Метод з’єднання екземпляра сокета гарантує, що конкретний сокет присвоєно правильній кімнаті. Роблячи це, ми можемо обмежити обсяг широкомовних повідомлень (тих, які потрібно надсилати кожному відповідному користувачеві). На щастя для нас, цей метод і вся логістика управління приміщеннями надаються Socket.io нестандартно, тому нам насправді не потрібно робити нічого, крім використання методів.

Метод дбає про налаштування обробника для кожного нового отриманого повідомлення. Це можна трактувати як еквівалент обробника маршруту для REST API за допомогою Express. Тепер, якщо сокети насправді не розуміють «повідомлень», натомість вони просто піклуються про події. І для цього модуля дозволяється лише дві різні назви подій, „нове повідомлення” та „нове вечора”, бути повідомленням, отриманим або надісланим подією, тому і сервер, і клієнт повинні переконатися, що вони використовують однакові назви подій. Це частина контракту, який повинен відбутися, як і те, як клієнти повинні знати кінцеві точки API, щоб використовувати їх, це повинно бути зазначено в документації вашого сервера. Що стосується загальних повідомлень, переконуємось, що назва кімнати, на яку націлено, насправді така, до якої користувач раніше приєднався. Це лише невеличка перевірка, яка запобігає проблемам під час надсилання повідомлень.

Для приватних повідомлень фіксуємо дві його частини: цільового користувача та власне повідомлення, використовуючи швидкий та простий регулярний вираз. Після цього створюється корисне навантаження JSON і передаємо його наданому зворотному виклику. Отже, в основному цей метод призначений для отримання повідомлення, перевірки його, синтаксичного аналізу та повернення. Яка б логіка не потрібна була після цього кроку, вона буде знаходитись у користувацькому зворотному виклику.

РОЗДІЛ 5
МОБІЛЬНИЙ ДОДАТОК

React - це інструмент для створення призначених для користувача інтерфейсів. Його основне завдання - забезпечити відображення на екрані того, що можна побачити на веб-сторінках. React дозволяє легко створювати інтерфейси, розділяючи кожну сторінку на невеликі фрагменти і компоненти. Він дуже зручний для створення веб-застосувань і не вимагає великого порогу входження. Так от, хлопці з Facebook подумали, що було б краще, якби React можна було б використати для створення кроссплатформенных мобільних застосувань, і в 2015 році React Native був представлений публіці і з'явився на GitHub, де вже кожен міг внести свій вклад. React популярний з кількох причин. Він компактний і відрізняється високою продуктивністю, особливо при роботі з даними, що швидко міняються. За рахунок своєї компонентної структури, React заохочує вас писати модульний і багаторазово використовуваний код.

Нативні застосування - треба вивчити два стеки технологій, дублювати функціональність і тестувати її двічі. Це дуже дорого и дуже довго. Кроссплатформенная розробка - існує декілька варіантів: Flutter, Xamarin, Native Script і React Native. Було обрано React Native, ось декілько плюсів використання:

- Кроссплатформенность.
- Швидко і дешево. Як наслідок першого пункту, не треба підтримувати дві платформи. Додається нова функціа на React Native під одну платформу, а 80% кода працює і на іншій.
- Велике співтовариство. У Telegram є співтовариство React Native на 3000 розробників.
- Багатий вибір доступних компонентів і готових рішень.

З React Native є два способи розробити додаток : використати Expo або React Native CLI. Expo - це шар абстракцій - набір інструментів, бібліотек і сервісів для

швидкого запуску. Це деякий API, який "з коробки" дає доступ до можливостей пристрою : до камери, геолокации, push- повідомленням. У Expo є інструменти відладки і тестування додатка. Алгоритм простий: встановлюємо на телефон додаток Expo Client, підключаємо телефон з комп'ютером в одну локальну мережу, заходимо в Expo на телефоні, відкриваємо застосування, що розробляється. Усі зміни в коде відображаються вмить. Це допомагає тестувати і расшаривать на команду. Зліт на Expo швидкий і стрімкий. З Expo не потрібні Xcode і Android Studio. Пишемо в нашому середовищі розробки, збираємо додаток за допомогою сервісу Expo, підписуємо і оновлюємо на льоту. Мінуси:

- Не можна додавати кастомные нативні модулі.
- Expo - шар абстракцій над React Native. Не можна оновитися на нову версію фреймворка, поки не оновиться Expo.

React Native CLI перевага в можливості кастомізації, додавання нативних модулів. Але для підпису і складання додатка(і публікації) потрібний Xcode або Android Studio. Щоб протестувати додаток, необхідно зробити складання і завантажити на телефон, наприклад, за допомогою TestFlight для iOS або прямої установки з ПК на телефон. Стандартна схема, але незручно і довго.

Особливості	React Native CLI	Expo
Нативні Модулі	+	-
Розмір додатку «Hello World!»	5mb	25mb
Не потрібен Xcode/Android Studio	-	+
Шрифти та іконки з коробки	-	+
JS API: камера, геопозиція …	-	+
Стороні бібліотеки	+	-
Просте тестування	Тяжке	Просте

5.1. Реєстрація і вхід в систему

Спливаючий екран / вступне вікно керування

Він з'явиться на 5 секунд і потім зникне. На тому ж екрані буде зображення і компонент ActivityIndicator. Для управління видимістю екрану буде використовуватися функцію setTimeout. Ця функція setTimeout викликатиме функцію через 5 секунд, яка буде перевіряти наявність сесії входу в AsyncStorage, і якщо буде знайдено значення user_id, то екран буде переключено на головний екран, а якщо буде знайдено пусте / нульове значення, то екран буде перенаправлений на екран входу. Використовується AsyncStorage для зберігання змінної сесії.

Екран входу в систему

Він має 2 полів для введення Email і пароля користувача (з базовою перевіркою порожнього поля) і кнопку для запуску служби аутентифікації.

Екран реєстрації

Має 5 полів для введення імені, Email, пароля, віку і адреси (з базовою перевіркою порожніх полів) і кнопку для запуску служби реєстрації.

Головний екран

Це початковий екран після входу в систему або безпосередньо з спливаючого вікна, якщо користувач вже увійшов в систему. Цей екран є частиною навігаційного ящика, в якому є опції Home, Setting і Logout.

Вихід

У головному навігаційному ящику існує опцію виходу, яка буде запитувати підтвердження виходу із системи, і якщо користувач натисне "Так", то він очистить сесію і перенаправляє на екран входу, а якщо натисне "Ні", то попередження про підтвердження зникне.

Навігатор стеків, який використовується для навігації між екранами авторизації (Вхід та Реєстрація), а також для внутрішньої навігації головним екраном або екраном налаштувань Навігатор шухляд, який використовується для налаштування цільового екрану, який має три варіанти: головний екран, екран налаштування та вихід.

App.js містить основну навігацію

1. SplashScreen.js для заставки
2. LoginScreen.js для екрана входу

3. RegisterScreen.js для екрана реєстрації

4. DrawerNavigationRoutes.js містить навігацію за ящиками для посадочних екранів

5. HomeScreen.js для головного екрана в панелі навігації

6. SettingsScreen.js для екрана налаштувань у панелі навігації

7. NavigationDrawerHeader.js, наш користувальницький компонент заголовка, що використовується для заголовка панелі навігації

8. Loader.js, компонент завантажувача, який повідомляє про завантаження під час виклику веб-API

9. CustomSidebarMenu.js, наш власний компонент бічної панелі / шухляди, який замінить бічну панель / шухляду за замовчуванням

10. Зображення в каталозі зображень будуть використовуватися на екрані заставки, екрані входу та реєстрації

Приклад запиту до серверу у вікні реєстрації:

```
router.post('/registerBuyer', createAccountLimiter, async (requestBuyer, resBuyer)=>{
    console.log(req.body.item)
    const { errorBuyer } = registerValidationBuyer(req.body);
    if (errorBuyer) return res.status(403).send({error: error.details[0].message});
    const saltBuyer = await bcrypt.genSalt(256);
    const hashedPassword = await bcrypt.hash(req.body.passwordBuyerAdd, saltBuyer);
    const buyerData = new BuyerData({
        name: req.body.nameBuyer,
        surname: req.body.surnameBuyer,
        email: req.body.emailBuyer,
        phone: req.body.phoneBuyer,
        password: hashedPasswordBuyer,
    });
    try{
        const savedBuyer = await buyer.save();
        console.log(savedBuyer._id)
        res.send({_id: savedBuyer._id})
    } catch(errBuyer){
        res.status(403).send({error: errBuyer});
        console.log(errBuyer)
    }
});
```

Приклад запиту до серверу для перевірки паролю та номеру телефона:

```
router.post('/checkPhone', createAccountLimiter, async (requestBuyer,
resBuyer)=> {
  const phoneExist = await Buyer.findOne({phoneBuyer:
req.body.phoneBuyer});
  if (phoneExist) return res.status(200).send('This phone already exist by another
Buyer!');
  res.send("OK")
})

router.post('/checkEmail', createAccountLimiter, async (requestBuyer,
resBuyer)=> {
  const emailExist = await Buyer.findOne({emailBuyer: req.body.emailBuyer});
  if (emailExist) return res.status(200).send('This email already exist by another
Buyer!');
  res.send("OK")
})
```

5.2. Колекція товарів

Списки є одним із поширених прокручуваних компонентів для відображення подібних типів об'єктів даних. Список схожий на вдосконалену версію компонента ScrollView для відображення даних. React Native надає компонент FlatList для створення списку. FlatList відображає лише елементи списку, які можуть відображатися на екрані. Крім того, FlatList пропонує багато вбудованих функцій, таких як вертикальна / горизонтальна прокрутка, подання верхнього / нижнього колонтитулів, роздільник, витягування для оновлення, ліниве завантаження тощо.

FlatList - це спеціалізована реалізація компонента VirtualizedList для відображення обмеженої кількості елементів, які можуть поміститися всередині поточного вікна. Решта елементів буде відображена за допомогою дії прокрутки списку. FlatList можна просто реалізувати за допомогою реквізитів data і renderItem для створення списку. Існує багато інших додаткових реквізитів, щоб додати більше функцій, тому реквізит можна класифікувати на основний та необов’язковий.

Первинний реквізит:

- data бере масив елементів будь-якого типу для заповнення елементів у списку.
- renderItem вимагає функції, яка приймає об'єкт елемента як вхід з джерела даних для побудови та повернення компонента елемента списку.

Завантаження даних списку

Для того, щоб отримати фіктивну відповідь, використовуйте API запиту пошуку TheCatsAPI, який не вимагає ніякої реєстрації або ключа API. Мережевий запит можна просто реалізувати, використовуючи запит отримання:

```
fetchCats() {
  fetch('https://api.thecatapi.com/v1/images/search?limit=10&page=1') // 1
    .then(res => res.json()) // 2
    .then(resJson => {
      this.setState({ data: resJson }); // 3
    }).catch(e => console.log(e));
}
```

Вищевказаний запит:

- Завантаження даних із обмеженням на десять зображень із першої сторінки.
- Перетворення відповіді на об’єкт JSON.
- Збереження перетворених об'єктів JSON відповіді в об'єкті стану як значення ключа даних за допомогою функції setState.

Реалізування функції конструктора елементів списку

Запит на отримання надасть дані одному з основних реквізитів FlatList. Тепер реалізується функція для повернення компонента елемента списку для підставки.

Ключовим полем у відповіді є URL-адреса, яка буде використана для завантаження та відображення зображення. Роботу функції renderItem Component:

```
renderItem:
  renderItemComponent = (itemData) =>
   <TouchableOpacity>
    <Image style={styles.image} source={{ uri: itemData.item.url }} />
   </TouchableOpacity>
```

- Бере об’єкт як параметр, який буде використовуватися компонентами інтерфейсу користувача.

- Компонент TouchableOpacity використовується для реалізації прослуховувача кліків, оскільки компонент Image не має пропсу onPress.

- Компонент Image бере URI як вихідні дані для відображення зображення. Він автоматично завантажить зображення.

5.3. Геолокація

По перше потрібно переконатися, що у файлі Info.plist є правильні описи використання. Повідомлення з’явиться у вікні сповіщення, коли програма запитує дозволи, і повідомляє користувачеві, чому ви просите про це дозвіл. Вони також є частиною процесу перевірки магазину програм.

Якщо запитується доступ до місцезнаходження лише тоді, коли додаток використовується, вам просто потрібно переконатися, що у вашому Plist є елемент NSLocationWhenInUseUsageDescription. Якщо ви запитуєте фоновий дозвіл, вам також потрібно буде додати NSLocationAlwaysAndWhenInUseUsageDescription та NSLocationAlwaysUsageDescription до файлу PList.

Наступним кроком є встановлення залежності постачальника послуг Google Fused Location. За замовчуванням використовується вбудований диспетчер

```
import React from "react";
import MapView from "react-native-maps";
  const App = () => {
   return (
    <MapView
     style={{ flex: 1 }}
     initialRegion={{
      latitude: 37.78825,
      longitude: -122.4324,
      latitudeDelta: 0.05,
      longitudeDelta: 0.05
     }}
    />
   );
  };
  export default App;
```

місцезнаходжень. Бібліотека response-native-location, може реалізувати логіку для отримання координат широти та довготи. За допомогою ES6 запит на асинхронізацію, який оброблятиме подія. Після запуску цього обробника подій користувач запитує дозвіл отримати доступ до свого місцезнаходження; після надання запиту місцеположення буде повернуто користувачеві.

RNLocation.configure - цей метод бере один об'єкт, distanceFilter. DistanceFilter - це мінімальна відстань у метрах, на яку зміниться місце розташування пристрою до виклику нового зворотного виклику оновлення місцезнаходження у зразку програми. Встановлено значення null, щоб змінюватись, коли ви пересуваєте пристрій.Спочатку оголошується змінна розташування (не призначена на даний момент) за допомогою ключового слова Let, щоб її можна було перепризначити в інший час. Далі перевіряється статус дозволу,

наданого пристроєм. Це отримується за допомогою методу requestPermission, щоб попросити користувача надати дозвіл додатку. Ця логіка реалізована в першому

```
let location;
if(!permission) {
  permission = await RNLocation.requestPermission({
    ios: "whenInUse",
    android: {
     detail: "coarse",
     rationale: {
      title: "We need to access your location",
      message: "We use your location to show where you are on the map",
      buttonPositive: "OK",
      buttonNegative: "Cancel"
     }
    }
  })
  console.log(permission)
  location = await RNLocation.getLatestLocation({timeout: 100})
  console.log(location, location.longitude, location.latitude,
      location.timestamp)
} else {
  console.log("Here 7")
  location = await RNLocation.getLatestLocation({timeout: 100})
  console.log(location, location.longitude, location.latitude,
        location.timestamp)
```

розділі конструкції IF. ЯКЩО дозвіл отримано від користувача, використовуючи метод getLatestLocation. Частина ELSE повертає метод getLatestLocation, якщо дозвіл був наданий раніше. Метод getLatestLocation закінчиться, якщо обіцянка не

буде вирішена за час, проаналізований. Об'єкт розташування реєструється на консолі з наведеною нижче інформацією у форматі JSON:

{

"accuracy": 2000, "altitude": 0, "altitudeAccuracy": 0, "course": 0, "courseAccuracy": 0,

"fromMockProvider": false, "latitude": 37.42342342342342, "longitude": -122.08395287867832,

"speed": 0, "speedAccuracy": 0, "timestamp": 1606414495665

}

5.4. Створення мапи

Наявність точної інформації про місцезнаходження користувачів - це чудовий спосіб покращити взаємодію з користувачем. response-native-maps - це складова система для карт, яка постачається з кодом власного платформи, який потрібно компілювати разом з React Native. initialRegion, який буде відображатися на карті, коли компонент кріплень. Значення InitialRegion не можна змінити після його ініціалізації. Значення {flex: 1} гарантує, що <MapView /> займає весь екран. Властивості latitudeDelta та longitudeDelta вказують, на скільки потрібно збільшити область на карті.

5.5. Аккаунт користувача

Після того, як користувач зареєструвався у системі, йому присвоєно статус «unverified». При цьому статусі користувачу доступен малий спектр дій у додатку, для того щоб отримати повний доступ до системи потрібно заповнити інформацію особі у вкладці «Акаунт».

Коли користувач переходить до вкладки «Акаунт», створюється GET запит для отримання даних користувача. Потім завдяки методу «JsonParse» додаток преображає рядок JSON у звичайний string. Отримані дані виводяться у:

- Name
- Email
- Phone number
- Model of car
- Km per day
- Total distance
- Wallet

Також існує кнопка «Finish Verification», після неї відкривається вікно у якому користувач повинен вписати назву свого автомобіля та кількість кілометрів яку він проїжджає за день. Отриману інформацію конфертуєм у формат Json і створюємо запит формату PUT. Оновлену інформацію о користувачі програма виводить на єкран та оновляє «ViewController». Також статус користувача з «unverified», становиться verified і доступ до всіх модулів додатку розблоковується. Приклад коду:

```
router.put('/sellerAdress', verify, jsonParser, function(requestSeller, resSeller) {
  Seller.findByIdAndUpdate({_id: req.user._idSeller},
    {addressSeller: req.body.addressSeller,
      latitudeOfAddressSeller: req.body.latitudeOfAddressSeller,
      longitudeOfAddressSeller: req.body.longitudeOfAddressSeller},
    {new: true, useFindAndModify: false},
    async (errSeller, userSeller) => {
      if (errSeller) return res.status(403).send(errSeller);

      ProductSeller.updateMany({userIdSeller: user._idSeller},
        {
          addressSeller: req.body.addressSeller,
          latitudeOfAddressSeller: req.body.latitudeOfAddressSeller,
          longitudeOfAddressSeller: req.body.longitudeOfAddressSeller
        },
        { new: true, useFindAndModify: false },
```

```
            async (errPSeller, userPSeller) => {
                if(errPSeller) return res.status(403).send(errPSeller);

                res.status(200).send("Address Seller: " + user.addressSeller)
            })
    })
})
```

ВИСНОВОК

Незважаючи на те, що існує велика кількість сервісів доставки їжі, ця система з високою ефективністю дає змогу приватним підприємцям, шеф-кухарям та іншим людям, готувати смачну та якісну їжу вдома, та продавати її іншим. Проаналізувавши виконану роботу, можна констатувати, що виникли невеликі проблеми при створенні мобільного додатку на React Native. А саме, використання бібліотек, тому що не існує іншого вибору написання робочого додатку, при отриманні відповіді від серверу на завжди виходило розібрати JSON. Було розроблено серверну частину на Node.js. Вона дозволила об’єднати мобільний додаток, базу даних та веб додаток в одну повноцінну систему для контролю, обробки запитів та даних. MongoDB своєю чергою зберігає усі дані для подальшого використання системою. Завдяки React Native було розроблено мобільний додаток для двох платформ. POST, GET, PUT запити були призначені для з'єднання додатка з BackEnd. Запити дали змогу змінювати дані у системі, реєструвати користувачів у системі, показувати дані про продані товари, створені товари та інше. Завдяки цій роботі отримав змогу розробити повноцінну систему з мобільним додатком. Також розробив структури програмного забезпечення для маркетплейсу домашнього харчування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Що таке Node.js

URL: https://nodejs.org/en/about/

2. Документація MongoDB

URL: https://docs.mongodb.com/

3. MongoDB в действии, Бэнкер Кайл

URL: https://www.ozon.ru/product/mongodb-v-deystvii-8688130/

4. Вирооп Нарла, Майбутнє світового ринку послуг з доставки продуктів харчування онлайн

URL: https://store.frost.com/future-of-global-online-food-delivery-services-market-forecast-to-2025.html

5. Що таке Node.js? Де, коли і як його використовувати

URL: https://www.simform.com/what-is-node-js/

6. JavaScript скрізь: Створення крос-платформних додатків за допомогою GraphQL, React, React Native та Electron

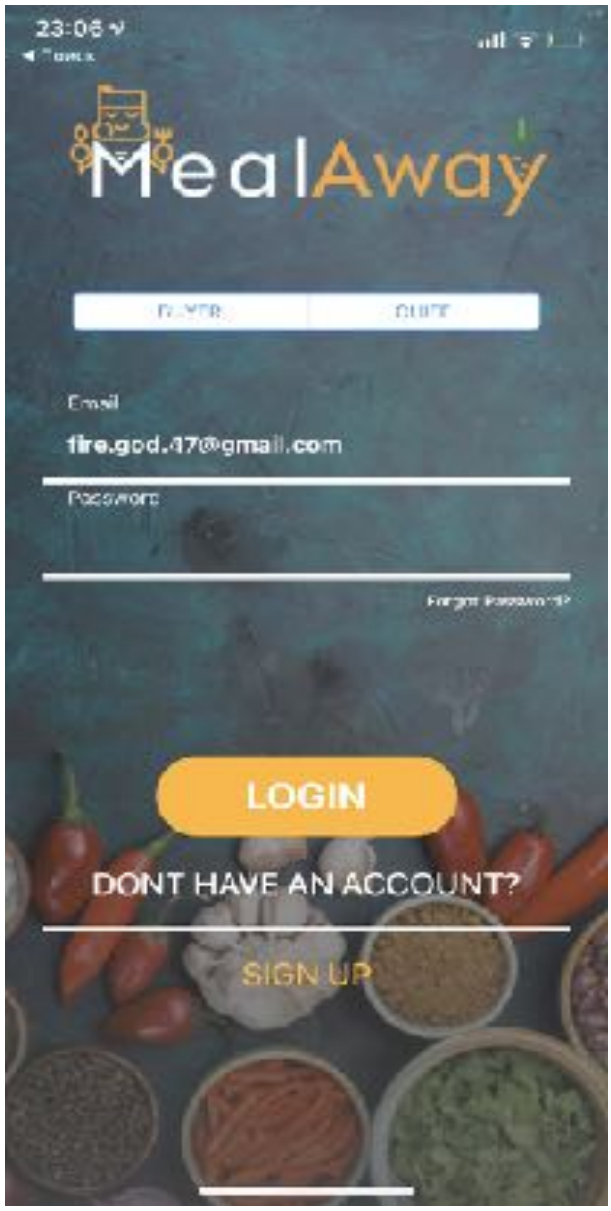
URL: https://www.amazon.com/gp/product/1492046981/ref=as_li_tl?ie=UTF8&camp=1789&creative=9325&creativeASIN=1492046981&linkCode=as2&tag=reacnati-20&linkId=cc0709e40f0d210790991b2c4482d6a4

7. Розуміння рідної архітектури React

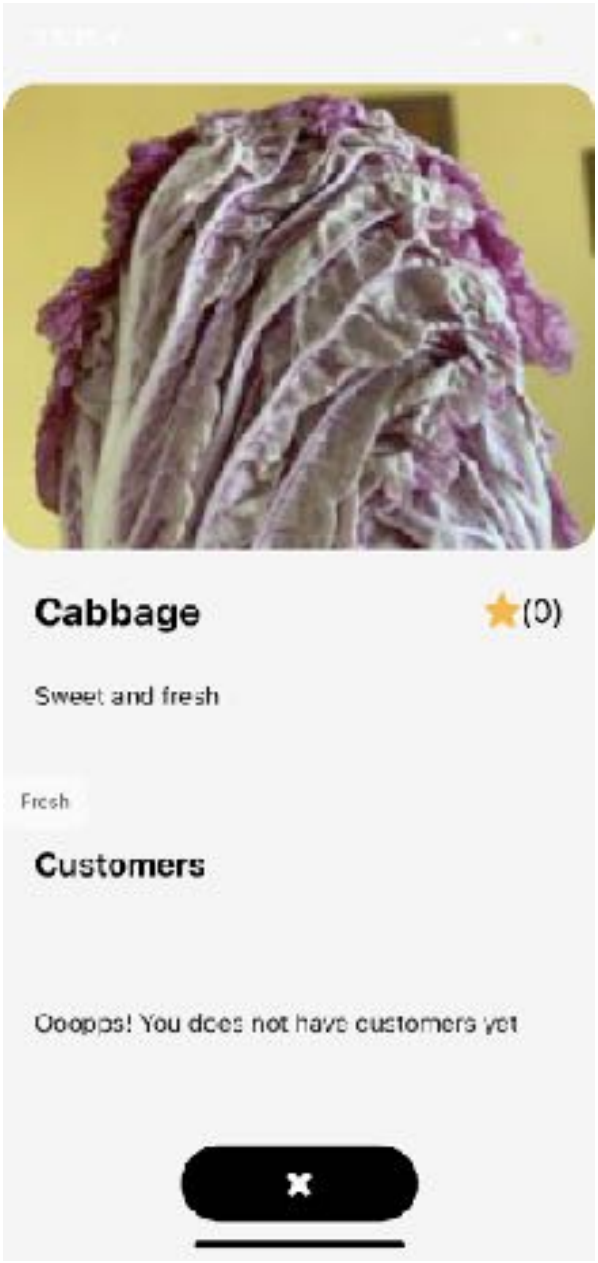
URL: https://dev.to/goodpic/understanding-react-native-architecture-22hh

ДОДАТКИ

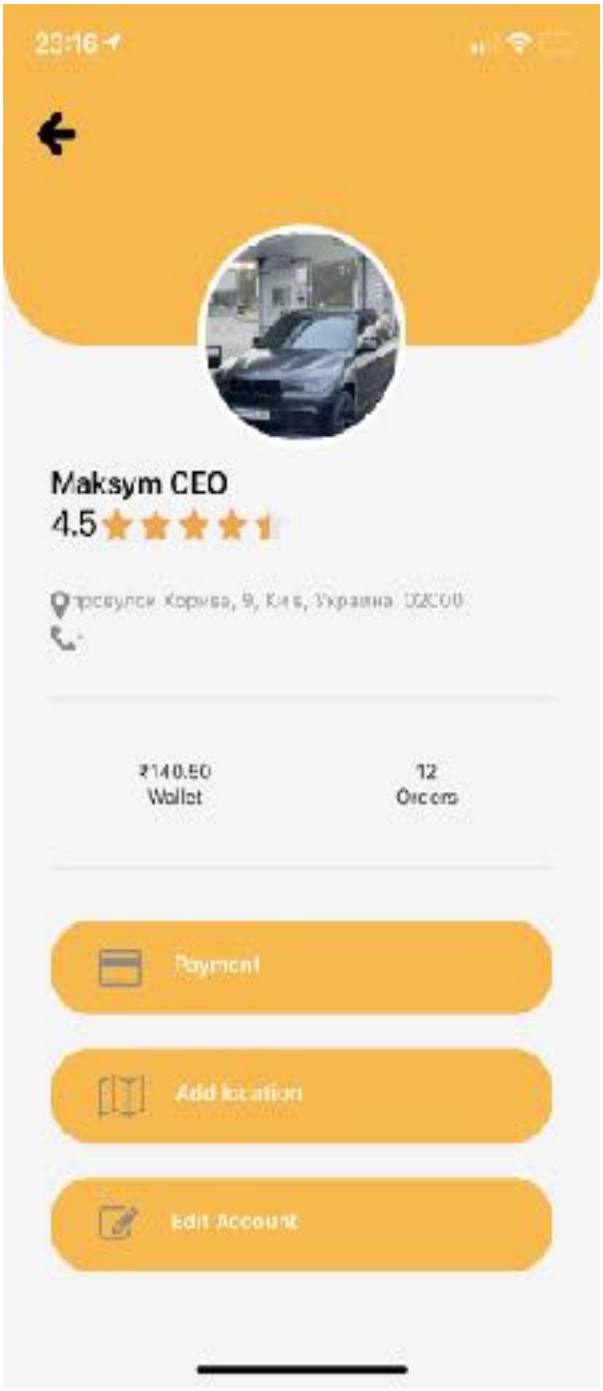
Додаток А. Вікно входу у мобільний додаток



Додаток Б. Вікно інформації про товар



Додаток В. Аккаунт користувача



Додаток Г. Інформація про товари, які зберігаються у базі даних

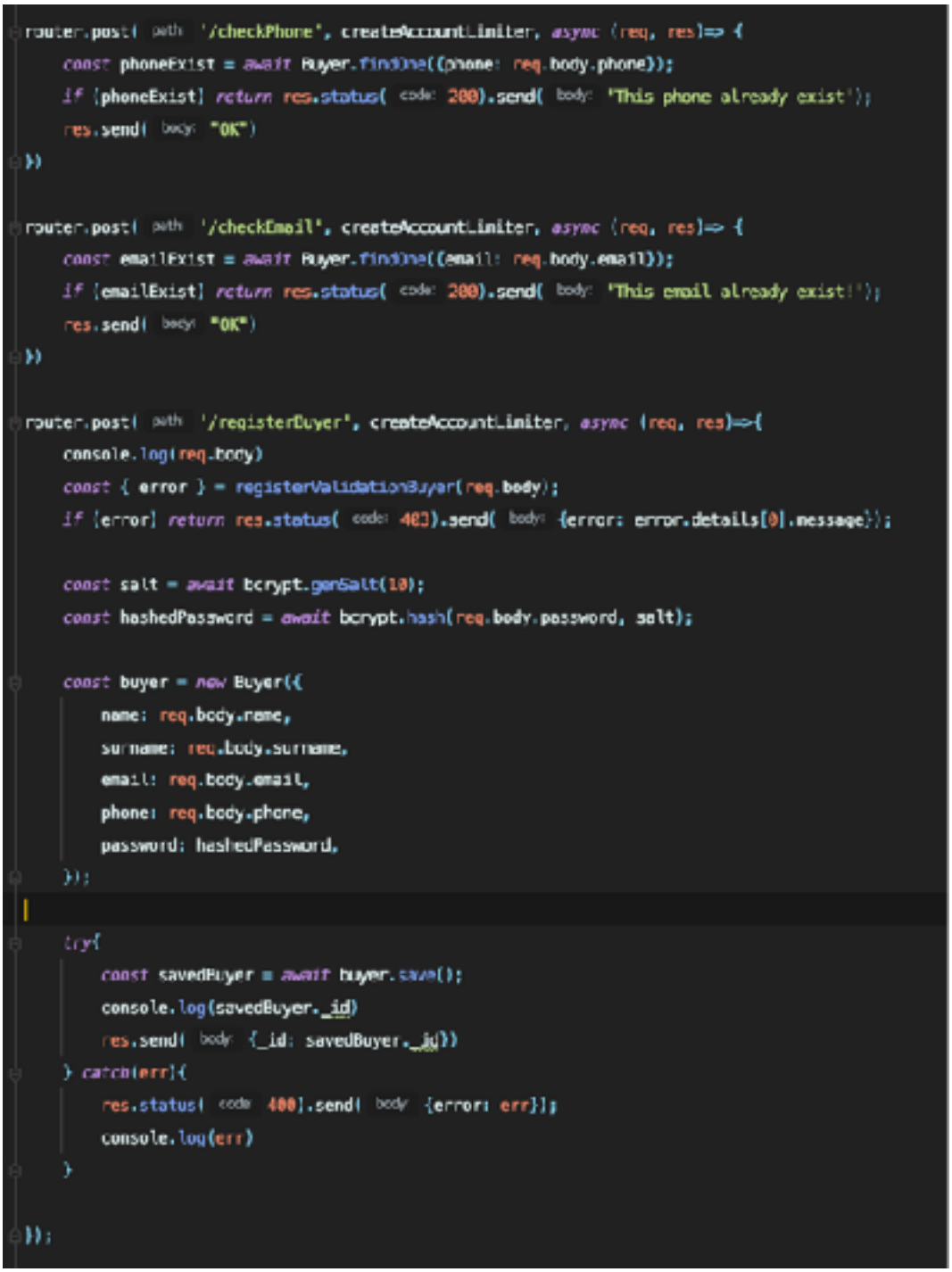
Додаток Д. Колекції даних

Collection Name	Documents	Documents Size	Documents Avg
buyers	11	21.13KB	1.92KB
complaints	10	2.61KB	267B
dynamicpromos	1	193B	193B
menuproducts	10	9.62KB	1016B
payments	65	140.77KB	2.17KB
products	12	19KB	1.58KB
sellers	9	9.92KB	1.1KB
users	1	185B	185B

Додаток Е. Усі існуючі запити до серверу

```
app.use(express.json());
app.use('/api/buyer', authRouteBuyer, deliveryProccessNotFood, eula);
app.use('/api/buyer/account', uploadImageBuyer, buyerInfo, notificationBuyer, buyerAddress, changeBuyerInfo );
app.use('/api/seller', authRouteSeller, checkStatus, getAllChiefs, getSellerInfo, deliveryProccess);
app.use('/api/seller/account', stripe, bankInformation, changeUserInfo, sellerAdress, userInfo, uploadImageSeller);
app.use('/api/products',
    searchByText,
    createProduct,
    creationProductsRoute,
    collectionOfProduct,
    getCustomers,
    getOrderedProduct,
    deleteProducts,
    changeProductInfo,
    createMenu,
    getCustomerInfo,
    getProductsMenu);
app.use('/api/buyer', buyerCollection, getDynamicPromo);
app.use('/api/payments', createPay, callBack, createPayTest, orderStatus, createPayStripe, paymentsInfo);
app.use('/api', fillAddressRoute);
app.use('/api/backOffice', createNewDynamicItem, getUsers);
```

Додаток Є. Приклад коду для реєстрації користувача у системі



```
router.post( path: '/checkPhone', createAccountLimiter, async (req, res)=> {
    const phoneExist = await Buyer.findOne({phone: req.body.phone});
    if (phoneExist) return res.status( code: 200).send( body: 'This phone already exist');
    res.send( body: "OK")
})

router.post( path: '/checkEmail', createAccountLimiter, async (req, res)=> {
    const emailExist = await Buyer.findOne({email: req.body.email});
    if (emailExist) return res.status( code: 200).send( body: 'This email already exist!');
    res.send( body: "OK")
})

router.post( path: '/registerBuyer', createAccountLimiter, async (req, res)=>{
    console.log(req.body)
    const { error } = registerValidationBuyer(req.body);
    if (error) return res.status( code: 403).send( body: {error: error.details[0].message});

    const salt = await bcrypt.genSalt(10);
    const hashedPassword = await bcrypt.hash(req.body.password, salt);

    const buyer = new Buyer({
        name: req.body.name,
        surname: req.body.surname,
        email: req.body.email,
        phone: req.body.phone,
        password: hashedPassword,
    });

    try{
        const savedBuyer = await buyer.save();
        console.log(savedBuyer._id)
        res.send( body: {_id: savedBuyer._id})
    } catch(err){
        res.status( code: 400).send( body: {error: err});
        console.log(err)
    }

});
```

Додаток Ж. Приклад коду для текстових повідомлень