ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра інженерії програмного забезпечення

Пояснювальна записка
до бакалаврської роботи
на ступінь вищої освіти бакалавр

на тему: «**РОЗРОБКА МОБІЛЬНОГО СЕРВІСУ ДЛЯ ЗАМОВЛЕННЯ ПРОДУКТІВ НА МОВІ ПРОГРАМУВАННЯ C#**»

Виконав: студент 4 курсу, групи ПД–42
спеціальності
121 Інженерія програмного забезпечення
(шифр і назва спеціальності/спеціалізації)

Шульчевський Є.О.
(прізвище та ініціали)

Керівник Жебка В.В.
(прізвище та ініціали)

Рецензент _____
(прізвище та ініціали)

Київ –2021

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

Навчально-науковий інститут інформаційних технологій

Кафедра Інженерії програмного забезпечення
Ступінь вищої освіти -«Бакалавр»
Спеціальність підготовки – 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
Інженерії програмного
забезпечення
_____ Негоденко О.В.
"____" _____2021 року

З А В Д А Н Н Я
НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТА

ШУЛЬЧЕВСЬКОМУ ЄВГЕНІЮ ОЛЕКСАНДРОВИЧУ

(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка мобільного сервісу для замовлення продуктів на мові програмування C#»

Керівник роботи: Жебка В.В., к.т.н., доцент кафедри ІПЗ
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом вищого навчального закладу від «12» березня 2021 року №65.

2. Строк подання студентом роботи «1» червня 2021 року

3. Вхідні дані до роботи
Методи роботи з реляційними базами даних;
Науково-технічна література з питань, пов'язаних з програмним забезпеченням та побудовою застосунків на мові програмування C#;
Програмні застосунки: Visual Studio 2019, Unity, Figma, Postman;

4. Зміст розрахунково-пояснювальної записки(перелік питань, які потрібно розробити).
4.1 Сервіси з замовлень продуктів харчування та їх можливості.
4.2 Вимоги та оцінка якості системи.
4.3 Опис проектування системи.

5. Перелік демонстраційного матеріалу (назва основних слайдів)

1. Актуальність проблеми
2. Об’єкт, предмет та мета дослідження
3. Існуюче програмне забезпечення та методи реалізації
4. Архітектура програмного додатку
5. Методи та класи програми
6. Архітектура бази даних

6. Дата видачі завдання «19» квітня 2021

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Підбір науково-технічної леітератури	19.04 – 22.04	
2	Аналіз тенденцій та ринку доставок продуктів харчування	23.04 – 26.04	
3	Аналіз існуючих застосунків для доставок продуктів	27.04 – 31.04	
4	Дослідження програмних засобів	01.04 – 05.04	
5	Моделювання об’єкту проектування	05.04 – 06.04	
6	Розробка функціоналу застосунку	07.04 – 13.04	
7	Вступ, висновки, реферат	14.04 – 15.04	
8	Розробка презентації застосунку	16.04 – 17.04	
9	Попередній захист роботи		

Студент ____ _____
(підпис) (прізвище та ініціали)

Керівник роботи _____ _____
(підпис) (прізвище та ініціали)

РЕФЕРАТ

Текстова частина бакалаврської роботи 44 с., 29 рис., 16 джерел.

Об'єкт дослідження – підвищення якості обслуговування клієнтів сфери доставок продуктів.

Предмет дослідження – додаток для системи замовлення та доставки продуктів клієнтам.

Мета роботи – розробка програмного забезпечення для клієнтів сфери доставок продуктів.

Методи дослідження – методи зберігання даних, методи розробки архітектури ПЗ, методи захисту передачі даних, методи тестування та валідації даних.

Наукова новизна даної роботи полягає в наступному:

- проведено аналітичне дослідження тенденцій розвитку ринку доставки продуктів харчування;
- проведено сегментування ринку доставки харчової продукції;
- встановлено, що оптимальним середовищем для розробки є Unity, а також була обрана мова програмування C#;
- на основі проведених досліджень було розроблено систему замовлення для різних груп споживачів.

Даний додаток допоможе користувачам зручно замовляти продукти не виходячи з дому.

В роботі проведено аналіз існуючих додатків на ринку доставки продуктів харчування. Було визначені основні переваги та недоліки додатків та встановлено основні функціональні можливості для користувачів.

Галузь використання – застосунок може використовувати будь-яка людина яка може завантажити додаток з Google Play Store якій необхідно замовити продукти дистанційно.

Ключові слова: ANDROID, UNITY, GIT, MYSQL, CLOUDFLARE, SSL, POSTMAN, PHPMYADMIN, C#, PHP.

ЗМІСТ

УМОВНІ ПОЗНАЧЕННЯ

ПЗ – програмне забезпечення

IDE – інтегроване середовище розробки

API – прикладний програмний інтерфейс

XML – розширювана мова розмітки

СКВ – система контролю версій

БД – база даних

UML – уніфікована мова моделювання

MySQL – реляційна база даних

ВСТУП

У світі сучасних технологій сервіси доставки стали більш необхідними. За допомогою сервісів доставок люди мають можливість заощадити свій час на походи в магазин або на приготування їжі.

Можливості сучасних доставок зросли на тлі активного розвитку технологій. Набагато зручніше замовити продукти або готову продукцію в онлайн режимі ніж йти в магазин. При такому сценарії, людина витратить набагато менше часу, ніж, відвідувати магазин наживо.

На ринку постійно з'являються все більше сервісів для доставки, цим самим підвищуючи конкуренцію між сервісами. Кожен сервіс намагається виділиться серед інших і надати користувачеві унікальні можливості.

Розробка мобільного додатку є найбільш ефективним способом для просування компанії.

Наукова новизна даної роботи полягає в наступному:

- проведено аналітичне дослідження тенденцій розвитку ринку доставки продуктів харчування;
- було проведено сегментування ринку доставки харчової продукції;
- встановлено що оптимальним середовищем для розробки є Unity, а також була обрана мова програмування C#.

Актуальність роботи обумовлено тим, що ринок доставки ще досить сирий. І аудиторія доставок є невеликою. Для популяризації та автоматизації бізнес-процесів планується розробка мобільного додатка для подальшого впровадження в роботу компанії.

Результати дослідження бакалаврської роботи апробовані:

- На всеукраїнській науково-технічній конференції: «Застосування програмного забезпечення в інфокомунікаційних технологіях»;
- На XI науково-технічній конференції студентів та молодих вчених: «Сучасні телекомунікаційні технології».

РОЗДІЛ 1. АНАЛІЗ РИНКУ ТА КОНКУРЕНТІВ ПО ЗАМОВЛЕНЮ ПРОДУКТІВ

1.1 Аналіз тенденцій розвитку ринку доставки продуктів харчування

Сегмент доставки наборів страв та продуктів із супермаркетів почав розвиватися відносно недавно і є частиною ринку доставки продуктів харчування, який з'явився в Україні на початку 2000-х. Щоб зрозуміти тенденції та виділити досліджуваний сегмент, спочатку слід розглянути ринок доставки їжі загалом.

Перші служби доставки їжі в Україні почали з'являтися в 2001 році і мали місцевий характер, основними клієнтами яких були середні та великі компанії, офісний персонал. Перші служби масової доставки продуктів харчування та продуктів почали з'являтися в 2008 році, тоді як основними замовниками були офісні працівники, які замовляли їжу в ресторанах та кафе. Деякі компанії, особливо у великих містах, пропонують обід як соціальний пакет для працівників.

Сучасний ринок сервісів доставки продуктів харчування в Україні є специфічним. За оцінками дослідницької компанії Statista ринок є відносно невеликим за обсягами маючи показник споживачів близько 1500000. Серед основних з постачальників продуктів харчування є сервіси такі як: Zakaz.ua, Glovo, Raketa, Royal Service. Значну роль у розвитку ринку поставок продуктів харчування зіграла пандемія COVID-19.

Новатором в сфері послуг по доставці продуктів та їжі в Україні стала компанія "Фуршет". В 2007-2008 роках послуги з доставки продуктів почали пропонувати такі компанії як: "Фоззи", "Экипаж-Сервис". В перші роки на ринок послуг з доставок продуктів приходилось близько 0,2% всього товарообороту в Україні.

На даний момент ринок сервісів доставок продуктів поділяється на дві категорії:

- Доставка готової їжі (піца, суші, ресторані блюда та інше);
- Доставка продуктів;

На основі результатів аналізу ми бачимо, що роздрібні мережі все частіше переходять з офлайн-режиму в режим онлайн. Частка гігантів ринку в рекламі мобільних додатків збільшується, і тому можна очікувати подальшого зростання мобільного ринку доставок.

Основними форматами рекламних мотивів є відеоформати та банери. Кожен бренд привертає увагу до себе найвигіднішими пропозиціями – постійні акції, переваги для клієнтів з точки зору терміну доставки, сезонні знижки, окремі місцеві акції, що збільшують попит. На основі власного досвіду та на основі даних досліджень ми можемо визначити, що шаблони в цьому форматі приваблюють велику кількість замовлень.

1.2 Сегментація ринку доставки продуктів харчування

В Україні існує безліч форматів доставки їжі. Серед них можна виділити не дуже популярні, як, наприклад доставка з кафе, тому її можна назвати нішевими.

Серед всіх форматів можна виділити такі види доставки:

- Доставка їжі з кафе / ресторанів. Цей сегмент є найпопулярнішим серед усіх інших видів доставки їжі. Багато з них мають власну службу доставки;
- Доставка продукції з роздрібних магазинів. Основними представниками цього сегменту є мережі Fozzy та Інтернет-супермаркети. Він також включає служби доставки, які доставляють товари з різних магазинів;
- Послуги доставки їжі. Найбільші служби доставки їжі в Україні: Rocket, Glovo, Food Express, Royal Service, Smachnogo.ua та zakaz.ua. Ці

послуги поєднують доставку з декількох магазинів чи закладів громадського харчування;

- Послуги доставки здорової їжі. Особливістю цих послуг із типових видів доставки (суші, гамбургери, піца) є те, що послуги орієнтовані на доставку всього комплексу готових страв (сніданок / обід / вечеря), переважно на тиждень. Також, особливості цих послуг включають наявність спеціального меню: безглютенового, для діабетиків, вегетаріанського та з можливістю створення меню для потреб замовника;
- Окремо слід виділити доставку продуктів з невеликих магазинів, в стилі "фермерських магазинів" та магазинів з "еко" продуктами. У загальній структурі поставок частка таких видів доставки невелика, оскільки ці товари переважно позиціонуються як вузькоспеціалізований сегмент;

1.3 Автоматизація процесів бізнес процесів

Щоб пропонувати конкурентоспроможні ціни і сервіс, ресторатори змушені знижувати витрати на оплату праці персоналу та скорочувати витрати. В цьому році ще більше рутинних завдань перейде у відання машин [1, с.32].

Програми автоматизації складу контролюють витрата продуктів, допомагають керувати собівартістю, правильно організувати закупівлі і боротися з такою напастю, як пересортиця – коли через помилку співробітника виникає надлишок і дефіцит декількох продуктів, що відносяться до однієї товарної групи.

Мобільні планшети для офіціантів миттєво передають прийняте замовлення на касу і кухню - це знижує ймовірність, що гість не отримає блюдо через забудькуватість офіціанта, і оптимізує чергу приготування замовлень, коли кухня сильно завантажена.

Чи не вся праця комп'ютерів в ресторанах прихований від очей гостей: кіоски самообслуговування взаємодіють безпосередньо з відвідувачами, приймаючи замовлення і рекламуючи десерт, який ідеально підійде до вибраного кави. Хоча електронна заміна касирам асоціюється в першу чергу з мережами фастфуд, завдяки зростаючим можливостям кастомізації кіоски самообслуговування можуть стати звичайним явищем і в інших форматах закладів.

1.4 Огляд аналогів

У наш час створено не мало мобільних додатків для оформлення онлайн доставки. Додатків даного напрямку не так багато, розглянемо основні з них. Проаналізуємо їх переваги та недоліки.

Glovo – додаток в якому для користування потрібна реєстрація. Вибір мови відсутній, підтримується лише російська мова. У мобільному додатку реалізовані каталоги в яких ми можемо замовити продукти за різними категоріями такі як: Кава та десерти, Їжа, Супермаркети, здоров'я та краса, Магазини (рис 1.1). Також перевагою є доступність всіх розділів з головного екрану застосунка.

Рисунок 1.1 – Застосунок Glovo

Zakaz.ua – додаток в якому для користування потрібна реєстрація. В додатку реалізовано підтримку трьох мов: українська, російська, англійська. Також додаток має підтримку відразу 7 мереж супермаркетів. В кожному підрозділу каталогу товарів реалізований пошук з різноманітними фільтрами. У кожного товару реалізована сторінка з детальною інформацією про товар. З недоліків можна виділити доволі навантажений інтерфейс який може вводити користувача в оману. Додаток zakaz.ua зображено на рис 1.2.

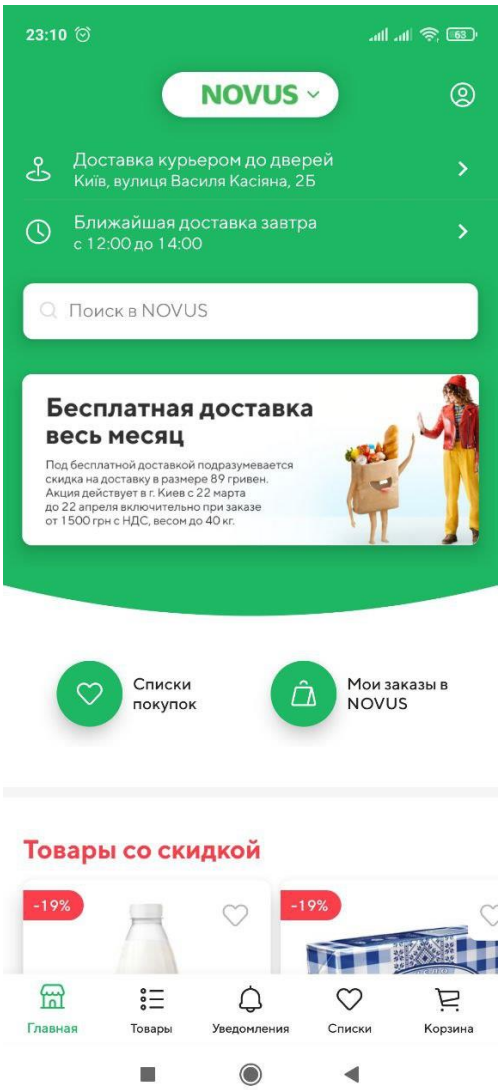


Рисунок 1.2 – Застосунок Zakaz.ua

Rocket – додаток в якому для роботи потрібна реєстрація. В додатку реалізована підтримка лише російської мови в той час назви ресторанів і продуктів пишуться на українській мові. Сервіс має не велику базу ресторанів, супермаркетів тощо. На головній сторінці можна відмітити розділ з акціями та є можливість перейти у всі можливі розділи цього додатку. Можна відмітити не дуже простий інтерфейс та малу кількість товарів які доступні для замовлення. Додаток зображено на рис 1.3.

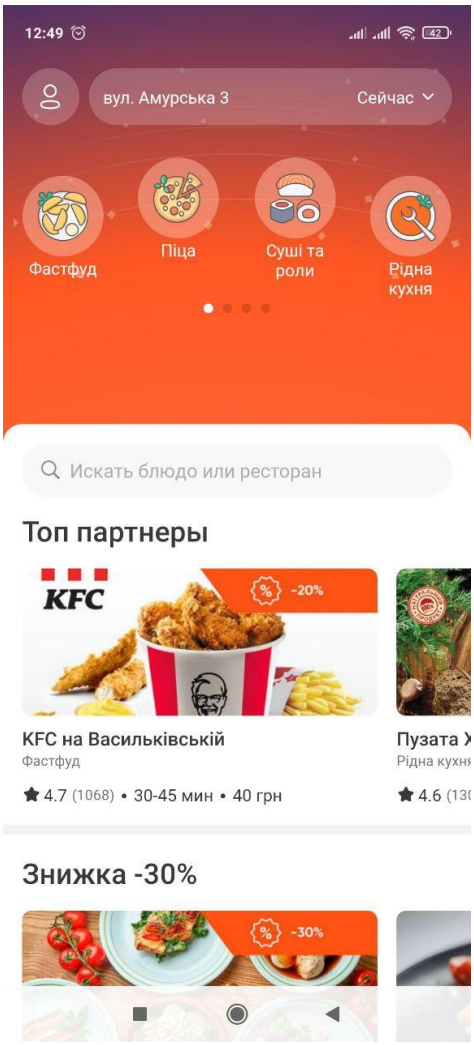


Рисунок 1.3 – Застосунок Rocket

Bolt Food – додаток потребує реєстрації для користування. Так як додаток інтернаціональний і працює в багатьох країнах в ньому реалізовано підтримку 17 іноземних мов. В порівнянні з попередніми додатками можемо побачити в нижній частині застосунку є 4 розділи які виконують певні функції. Можна відмити, що через поділення інтерфейсу на такі підрозділи з головного екрану нема можливості отримати доступ до кожної функції додатку. Застосунок можна побачити на рис 1.4.

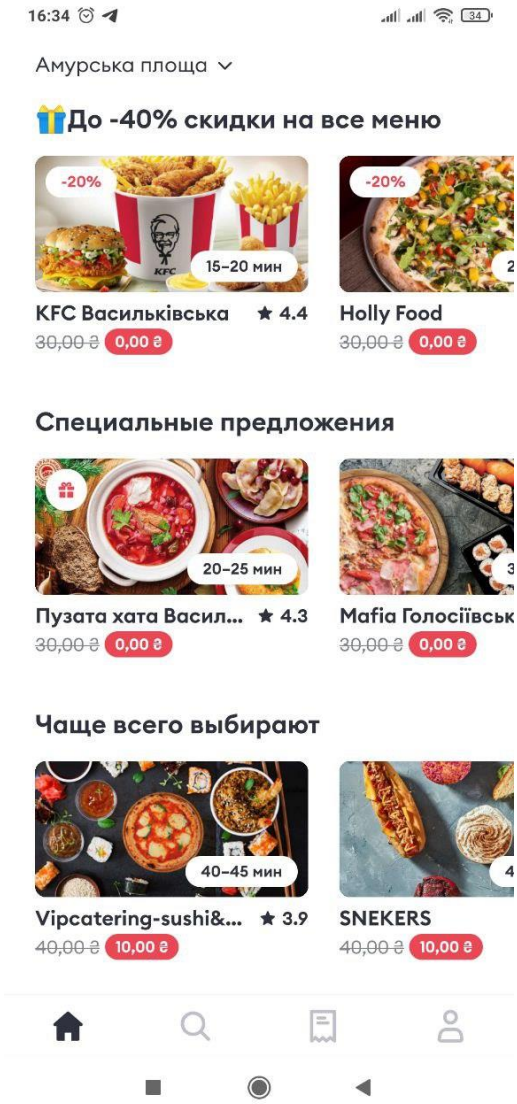


Рисунок 1.4 – Застосунок Bolt Food

Noww – додаток для роботи якого не потрібна обов’язкова реєстрація. Розроблений та був введений в експлуатацію нещодавно тому багато розділів знаходятся в розробці та недоступні. Має підтримку трьох іноземних мов. Має 4 функціональні розділи такі як:

- головне меню з каталогами доставок;
- історія замовлень;
- NCoin;
- налаштування.

З унікальних можливостей можна побачити систему кешбеку та розділ з спеціальними умовами продуктів для продажу. Додаток зображено на рис 1.5.

Рисунок 1.5 – Застосунок Noww

Вище перераховані додатки можна завантажити в Google Play Market. Узагальнене порівняння переваг та недоліків цих додатків представлено у таблиці 1.1

Таблиця 1.1 – Переваги та недоліки існуючих додатків

Назва застосунку	Переваги	Недоліки
Glovo	- Проста у використані - Великий вибір різноманітних кафе, ресторанів тощо - Доступність в багатьох регіонах України	- відсутній вибір мови - відсутня можливість скасувати замовлення - низька якість застосунку
Zakaz.ua	- Проста у використані - Великий вибір супермаркетів - Стабільна робота застосунку	- Навантажений інтерфейс
Rocket	- можливість вибору мови з трьох представлених - Великий вибір різноманітних кафе, ресторанів тощо	- відсутній пошук по категоріям магазинів
Bolt Food	- можливість вибору мови з сімнадцяти різноманітних - Розділений інтерфейс - Блок з різноманітними акціями	- доступність лише в Києві
Noww	- розділ з унікальними пропозиціями - система кешбеку - можливість вибору мови з трьох представлених	- відсутній пошук по категоріям - доступність у декількох районах Києва

Також існують і інші додатки в Google Play Market [8], усі вони в деякій мірі схожі між собою.

Розглянуті були найбільш популярні застосунки. Проаналізувавши можна прийти до висновку, що в кожному додатку існують недоліки для користувача.

Частину переваг та недоліків було враховано при розробці нового ПЗ. Створений додаток отримав назву «Fasty». Він призначений для швидкого

замовлення та інформування користувача о усіх етапах оформлення та отримання замовлення.

Мета застосунку – створити зручні умови для вибору товару та його оформленню.

Вибір операційної системи проводилось в залежності від її популярності. За даними ресурсу https://gs.statcounter.com/ [16] можна дізнатися наскільки популярна певна операційна система. Дані були взяті з березня 2020-го року по березень 2021-го року.

Загалом статистика мобільних операційних систем виглядає наступним чином (рис.1.6.):

- Android – 71.81%;
- iOS – 27.43%;
- Samsung – 0.38%;
- KaiOS – 0.14%;
- Інші – 0.14%;
- Linux – 0.02%.

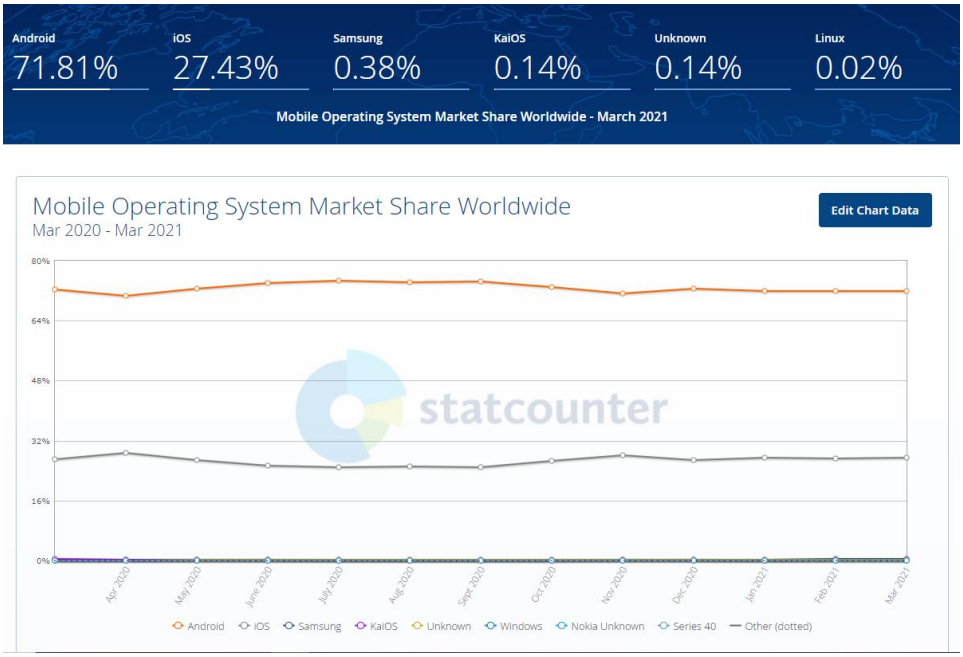


Рисунок 1.6 – Статистика мобільних ОС

На основі статистики мобільних операційних систем було вирішено розроблювати застосунок для ОС Android.

1.5 Постановка завдання бакалаврської роботи

Завданням бакалаврської роботи є аналіз ринку доставок продуктів харчування та подальша розробка мобільного додатку для замовлення онлайн продуктів.

Для цього потрібно виконати наступні задачі:

- Проаналізувати тенденції розвитку ринку сервісів з доставки продуктів;
- Провести апробацію отриманих результатів та оформити результати бакалаврської роботи;
- Виділити оптимальні інструменти для розробки серверної так клієнтської частини застосунку;
- Вибрати мову програмування в залежності від середовища розробки;
- Проаналізувати вибір програмних засобів для реалізації проєкту;
- Спроектувати інтерфейс проекту;
- Розробити серверну частину додатку;
- Розробити програмне забезпечення на основі функціональних вимог.

РОЗДІЛ 2. ВИБІР ПРОГРАМНИХ ЗАСОБІВ ДЛЯ РОЗРОБКИ МОБІЛЬНОГО ДОДАТКУ

2.1 Вибір середовища розробки

Для розробки мобільного додатку розглянемо пару найбільш популярних середовища для розробки ПЗ: Eclipse, Android Studio, Unity.

Eclipse – безкоштовне середовище для розробки модульних кроссплатформених додатків від розробника Eclipse Foundation.

Серед переваг Eclipse можна виділити:

- простота в інсталяції та використані;
- має відкритий вихідний код, що дозволяє використовувати безкоштовно;
- має гнучку архітектуру і легко налаштовується за допомогою великої кількості плагінів;
- написана на Java, що дозволяє працювати на будь-якій платформі;
- має велику кількість підтримуваних мов програмування;
- має можливості для роботи з різними типами баз даних;
- має підтримку безперервної інтеграції.

Також вона має важливі недоліки такі як:

- працює повільно та використовує багато системних ресурсів;
- має великий об’єм;
- функції відкладки не так розширені як у інших.

Android Studio – безкоштовна інтегрована середа для розробки мобільних додатків від компанії Google. Дана програма є дуже гнучкою у використані. Для розробки додатків в ній реалізована підтримка емулятора мобільних пристроїв. З недавнього часу Android Studio отримав підтримку мови програмування Kotlin який надав ще більше можливостей для розробки ПЗ та спростив написання коду. Інтерфейс програми реалізований так що б за

допомогою спеціальних блоків можна було переміщати на вікно розробки тим самим прискорюючи розробку програми. Так само є можливість написати за допомогою коду розмітку інтерфейсу в програмі. Однак є і проблеми з цим середовищем для розробки ПЗ. Такі як: застаріла Офіційна документація. Повільна робота IDE, проблеми при компіляції програми.

Unity – кроссплатформенне середовище для розробки додатків створена компанією Unity Technologies. За допомогою платформ з'являється можливість розробляти ПЗ для 25 різних платформ таких як: мобільні пристрої, приставки, персональні комп’ютери і інші. Unity досить схожа на Android Studio тим що теж має інтерфейс за допомогою якого можна просто перетягуючи елементи створювати інтерфейс для програми. Засоби так-же має широкі можливості для відкладання такі як консоль, профілювальник, які дозволяють більш якісно виконувати відкладання ПЗ. Також має велике ком'юніті, що дає можливість знайти будь-яку відповідь на будь-яке питання.

Отже, проаналізувавши три найпупулярніших середовища було вибрано Unity. Воно надає усі можливості для розробки мобільного додатку на платформі Android.

2.2 Інтерфейс та основні компоненти середовища розробки Unity

Для наглядного прикладу на рисунку 2.1 представлено інтерфейс середовища розробки Unity на якому зображено один із розділів розроблюваного додатку.

Основні функції Unity:

- Зручний редактор сцен на якому можна в режимі реального часу Побачити як він буде виглядати після компіляції.
- Можливість емулювати запуск проекту і подивитися як він буде працювати.
- Інспектор, за допомогою якого можна вибрати потрібний елемент. Та редагувати його властивості.

- Консоль. Використовується для виведення динамічної інформації налагодження, помилок і попереджень.

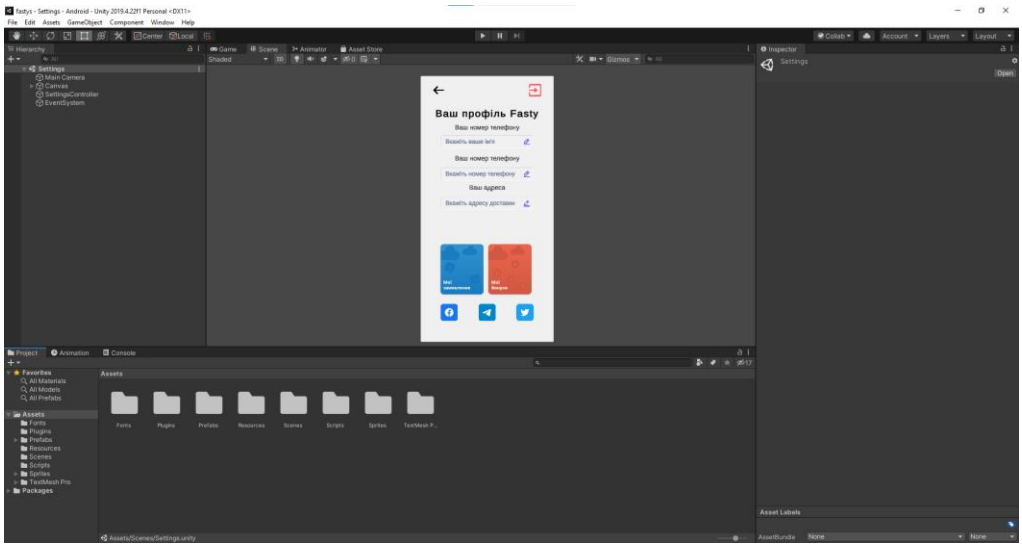


Рисунок 2.1 – Інтерфейс Unity

Сама архітектура додатка базується на сценах. За допомогою сцен Unity [9] розуміє які скрипти потрібно запускати і який інтерфейс використовувати. В додатку може бути багато сцен.

У проекті в основному є такі папки за допомогою яких організовується зберігання тих чи інших ресурсів проекту.

В папці Fonts організовано зберігання всіх шрифтів які використовується проекті.

В папці Prefabs зберігаються готові блокові елементи які використовуються в додатку.

У папках Scenes зберігаються сцени проекти до яких з папки Scripts отримують доступ.

Панель ресурсів зображена на рисунку 2.2

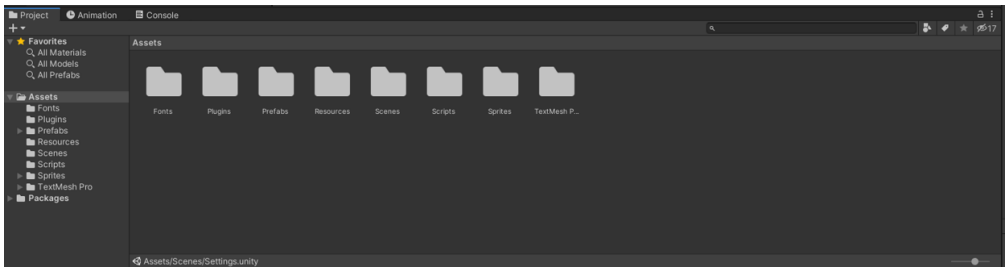


Рисунок 2.2 Панель ресурсів Unity

Панель інструментів забезпечує доступ до найбільш важливим робочим функціям. Зліва він містить основні інструменти для управління видом сцени і об'єктами всередині неї. У центрі розташовані елементи управління відтворенням, паузою і кроком. Кнопки праворуч надають вам доступ до ваших хмарним службам Unity і вашого профілю Unity, за яким слід меню видимості шарів і, нарешті, меню макета редактора (яке надає кілька альтернативних макетів для вікон редактора і дозволяє вам зберігати свої власні настройки) макети. Панель інструментів зображена на рисунку 2.3

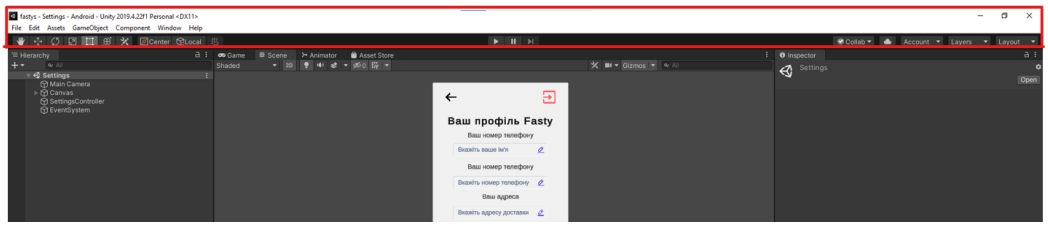


Рисунок 2.3 Панель інструментів Unity

В Unity реалізована можливість авто-генерування файлу маніфесту. Файл маніфест зберігає основну інформацію таку як: назву, версію, іконки, які йому потрібні дозволи, усі сцени та скрипти. Так само є можливість перевизначити файл маніфесту за допомогою створення його в певній директорії проекту. Що дає можливість більш тонко налаштувати під потреби проекту.

2.3 Вибір IDE для роботи з Unity

Серед найпопулярніших IDE для роботи над проектом .Net є: Visual Studio, JetBrains Rider.

Visual Studio – інтегроване середовище розробки ПЗ створене компанією Microsoft. Дозволяє створювати як консольні додатки, так і ігри і додатки з графічним інтерфейсом а також веб-додатки для усіх платформ сімейства Windows. В Visual Studio входить вбудований редактор вихідного коду з підтримкою технології IntelliSense та можливістю рефакторінга коду [14]. Інші вбудовані інструменти включають в себе редактор форм для спрощеного конструювання додатків з графічним інтерфейсом. Visual Studio має підтримку магазину плагінів за допомогою яких є можливість розширити функціонал IDE. Серед переваг Visual Studio 2019 можна виділити:

- підтримку платформ .NET;
- хмарне зберігання проекту;
- безкоштовна версія Community;
- магазин плагінів.

Також існують і мінуси які можуть надати складнощі при розробці. Серед таких мінусів є:

- складність інтерфейсу та велика кількість функцій які можуть ускладнити вивчення програми новачку;
- баги при оновленні версії та інші помилки оптимізації.

Інтерфейс Visual Studio зображено на рисунку 2.4

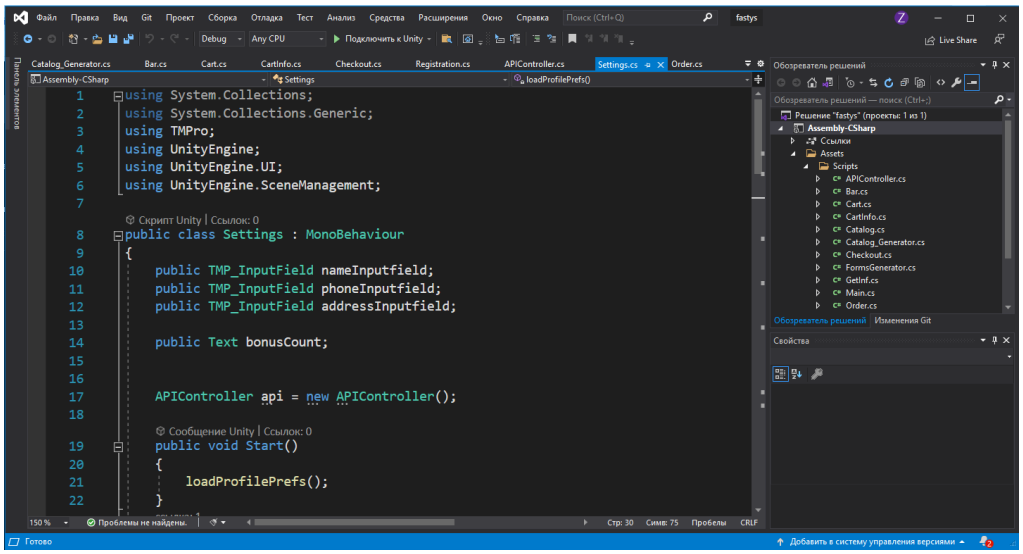


Рисунок 2.4 – Інтерфейс IDE Visual Studio

JetBrains Rider – кроссплатформена інтегрована середа розробки для платформи .NET. Підримує магазин вбудованих плагінів для розширення функціоналу. З переваг JetBrains Rider можна виділити:

- Підтримка декількох запущених проектів. Multiple runtime;
- Кроссплатформеність. IDE може працювати як на Windows так і на ОС сімейства Linux та macOS;
- Потужний плагін ReSharper який дозволяє підвищити продуктивність IDE;
- Підтримка контролю версій таких як Git, Mercurial та TFS;

Серед плюсів можна виділити ще й такі мінуси:

- Молодість IDE. Багато функціональностей відсутні або в розробці. Багато помилок при використані деяких функцій;
- Ціна. IDE є платною і стартує з ціною 14 доларів США.

Інтерфейс Visual Studio зображено на рисунку 2.5.

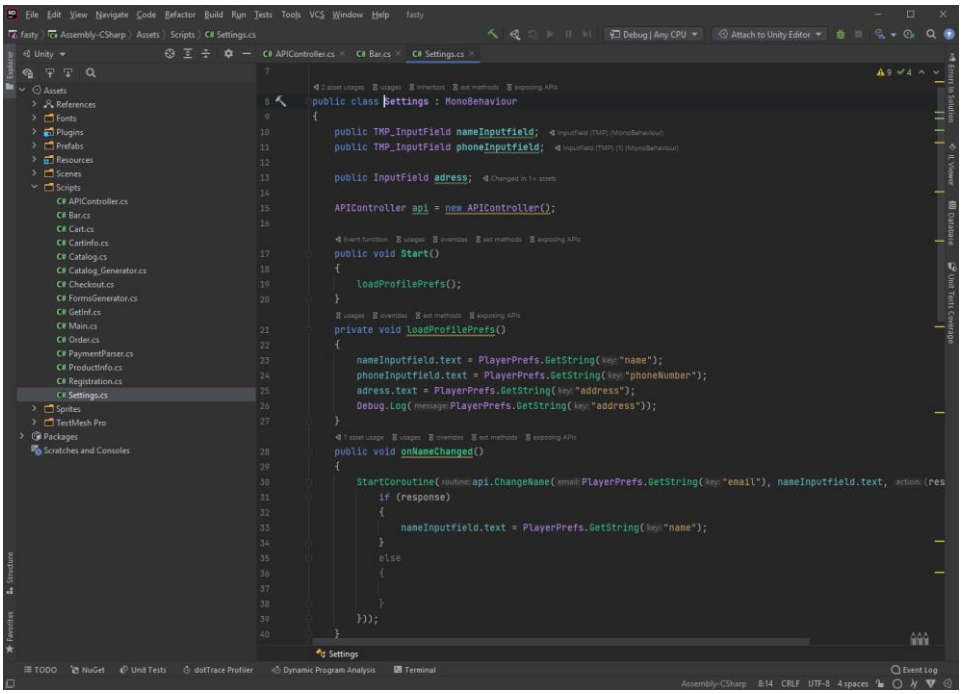


Рисунок 2.5 – Інтерфейс IDE JetBrains Rider

Отже переглянувши усі переваги та недоліки доцільним буде використання Visual Studio 2019 в якості основної середовища для написання коду.

2.4 Вибір системи управління базами даних

Система управління базами даних – сукупність комплексу програм для створення бази даних та можливістю вставляти, оновлювати, видаляти та вибирати дані. Також СУБД забезпечує безпеку зберігання та цілісність даних [5, с.45].

Для розробки стояло питання як потрібно зберігати дані. Розглянемо найбільш популярні системи управління базами даних такі як: MySQL, PostgreSQL, MongoDB, Redis.

MySQL - вільна реляційна система управління базами даних розроблена компанією Oracle. MySQL підходить як для маленьких так і для середніх проектів. Гнучкість цієї СУБД полягає в великій підтримці різних

типів таблиць [3, ст.52]. Також має велике ком'юніті і велика кількість документації яка допомагає вирішити будь-які проблеми.

PostgreSQL - об'єктно-реляційна СУБД. За основу був узятий мову SQL. Має більш поліпшену архітектуру і дозволяє створювати більш гнучкі бази даних в плані налаштувань. До плюсів PostgreSQL можна віднести:

- Високу надійність транзакцій і реплікації;
- Можливість розширення;
- Спадкування.

MongoDB - це документоорієнтована база даних, що означає, що вона зберігає дані в документах типу JSON. Є однією з сімейства NoSQL СУБД. Має перевагу в масштабуванні. За допомогою сегментування об'єкти бази даних розподіляються по різних вузлів кластера. Може використовуватися для застосувань в таких галузях:

- Мобільні додатки;
- Системи зберігання контенту;
- Ігри;
- Збереження даних.

Redis – NoSQL база даних, яка використовує пам'ять як спосіб зберігання. Доступ до записів можна отримати лише за ключем доступу.

Redis має не дуже гарні обмеження серед яких можно відмитити такі:

- Розмір БД залежить від доступної памяті;
- Так як БД NoSQL, то в ній відсутня буд-яка можливість сегментувати користувачів за групами доступу.

Проаналізувавши основні популярні системи управлінням базами даних було вибрано використовувати MySQL так як вона найбільш поширена, реляційна, що дає можливість встановлювати контроль доступу до таблиці та організовувати табличну архітектуру.

2.4.1 Використання інструменту phpMyAdmin для роботи з базою даних

Для найбільш ефективної роботи з базою даних будемо використовувати інструмент phpMyAdmin.

phpMyAdmin – це безкоштовне програмне забезпечення, написане на PHP, для управління MySQL через Інтернет. phpMyAdmin підтримує широкий спектр операцій MySQL та MariaDB. Часто використовувані операції (управління базами даних, таблицями, стовпцями, взаємозв'язками, індексами, користувачами, дозволами тощо) можна виконувати через користувальницький інтерфейс, одночасно дозволяючи безпосередньо виконувати оператор SQL. Загалом phpMyAdmin підтримує такі можливості:

- Імпорт данних з таких форматів як CSV, SQL;
- Експорт даних у формати: PDF, CSV, XML;
- Робота з декількаами серверами;
- Перетворення даних, що зберігаються в будь-якому форматі, за допомогою набору визначених функцій, таких як перегляд даних BLOB в вигляді зображення або посилання для завантаження;
- Створення запитів з використанням Query-by-example (QBE);
- Керування групами користувачів та їх превілегіями;
- Графічне відображення уявлення вашої бази даних з посиланнями.

Інтерфейс інструменту можна побачити на рисунку 2.6

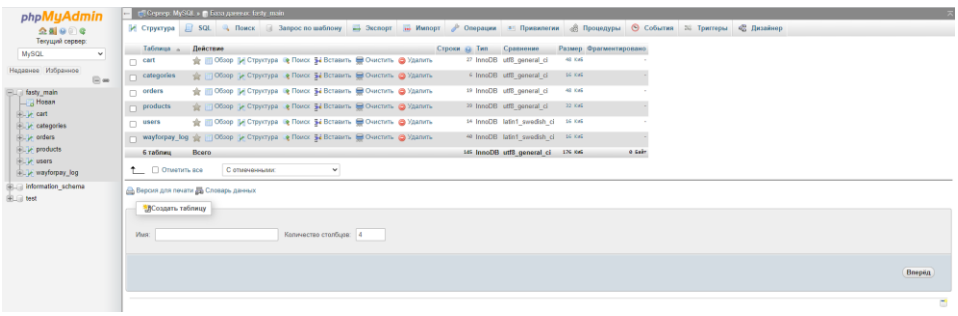


Рисунок 2.6 – Інтерфейс phpMyAdmin

Отже, phpMyAdmin є надійним та перевіреним інструментом для роботи з базами даних MySQL.

2.5 Використання системи контролю версій Git та Github

Система контроль версій (СКВ) дозволяє працювати над однією програмою декільком розробникам. У разі помилок система контролю версій дозволяє повернуться до попередніх версій коду.

СКВ бувають як локальними, централізованими, розподіленими. Основна різниця між ними тим що локальна СКВ зберігається тільки на локальному комп'ютері і отримати доступ до неї може тільки локальний користувач.

Централізована система контролю версій має виділений сервер на якому зберігається репозиторій проекту що дає можливість користувачам отримувати оновлення і дізнаватися хто що змінює в проекті. Однак дана СКВ має істотний мінус це єдиний сервер. У разі його відмови розробники втрачають можливість використовувати контроль версій. Найбільш безпечним спосіб є використання розподіленої системи керування версіями. Її логіка побудована на використанні як локальних так і централізованого віддаленого сховища. Кожен розробник може мати локальну версію проекту і в будь-який момент опублікувати її на віддаленому репозиторії тим самим оберігає розробників від будь-яких збоїв. Приклад роботи розпозділеної системи контролю версій можна побачити на рисунку 2.7

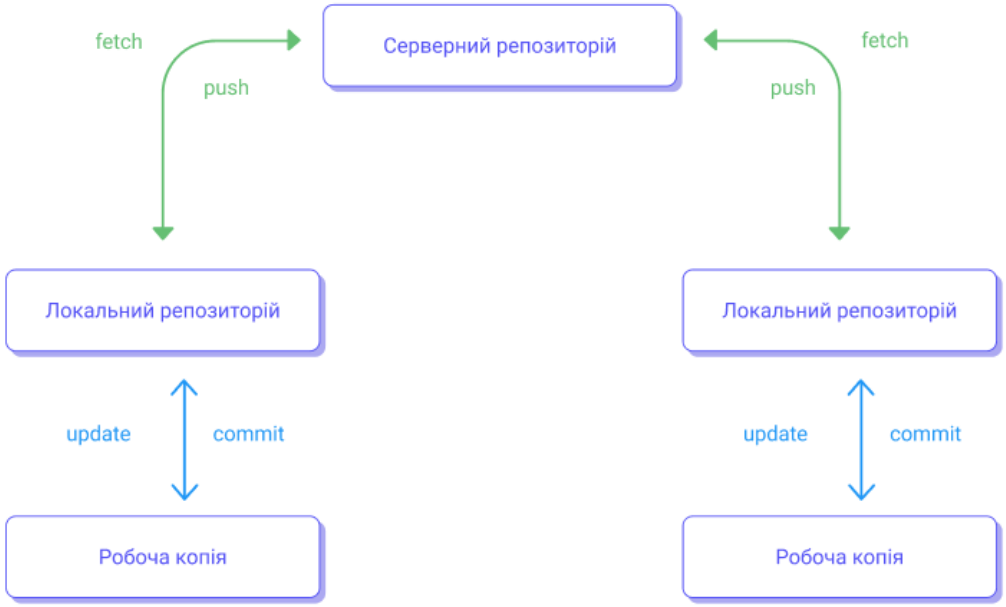


Рисунок 2.7 – Розподілена система контролю версій

Для роботи з розподіленою системою контролю версій будемо використовувати Git. Git - це розподілена система контролю версій дозволяє розробникам працювати над одним і тим же проектом, а так само відстежувати будь-які зміни на проекті.

GitHub – це сервіс хостингу для репозиторіїв Git. Має усі можливості представлені у Git. За допомогою GitHub розробники мають можливість зберігати свій проект в онлайні та застосовувати коміти для внесення змін (рис. 2.8).

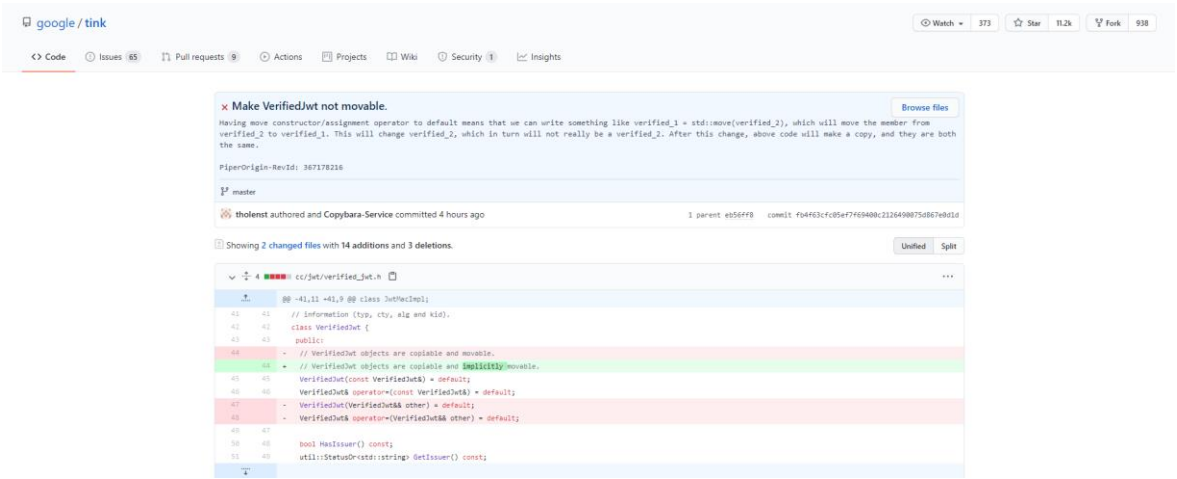


Рисунок 2.8 – Коміт на GitHub

Також GitHub підтримує такі функції:

- Історія комітів;
- Порівняння різних комітів;
- Коментування окремих ділянок коду;
- Створення запитів на зміну коду іншими розробниками;
- Легка інтеграція проекту з різними інструментами із розділу Marketplace.

2.6 Figma як інструмент для проектування інтерфейсу

Для розробки інтерфейсу існують багато програм. Розглянемо найбільш популярні. До них можна віднести такі як: Figma, Adobe XD, Sketch.

Sketch – графічний редактор для операційних систем сімейства macOS. Використовується для проектування як веб-інтерфейсу, так і для мобільних телефонів. Існує підтримка інтерактивних прототипів та можливість поділитися ними. Sketch можна побачити на рис. 2.10

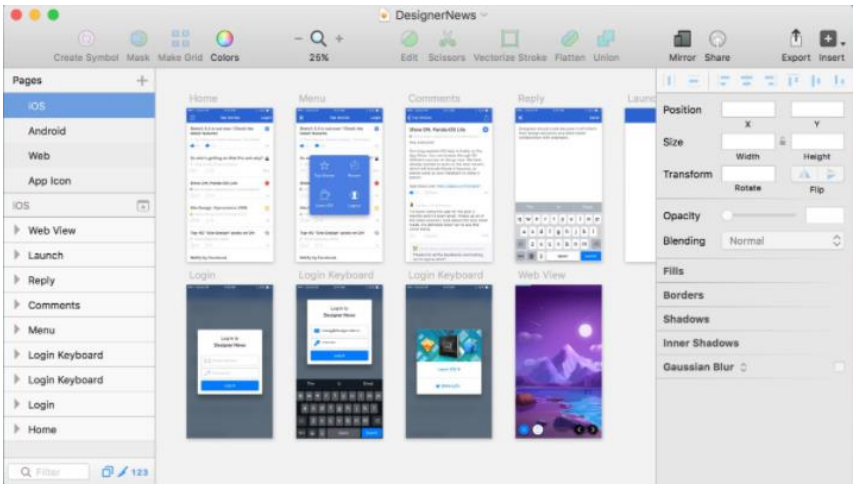


Рисунок 2.10 – Інтерфейс Sketch

Так як Adobe XD має багато помилок і погано оптимізована для операційної системи Windows ми не будемо її використовувати так як і

Sketch з тих причин, що вона доступна лише на операційних системах macOS. Тому використовувати будемо програму Figma.

Figma – безкоштовний кроссплатформенний онлайн-сервіс для розробки дизайну. Має можливості розроблювати дизайн інтерфейсів як в онлайн режимі так і в офлайн за допомогою додатку Figma Desktop [10]. Інтерфейс Figma з проектом зображено на рисунку 2.11

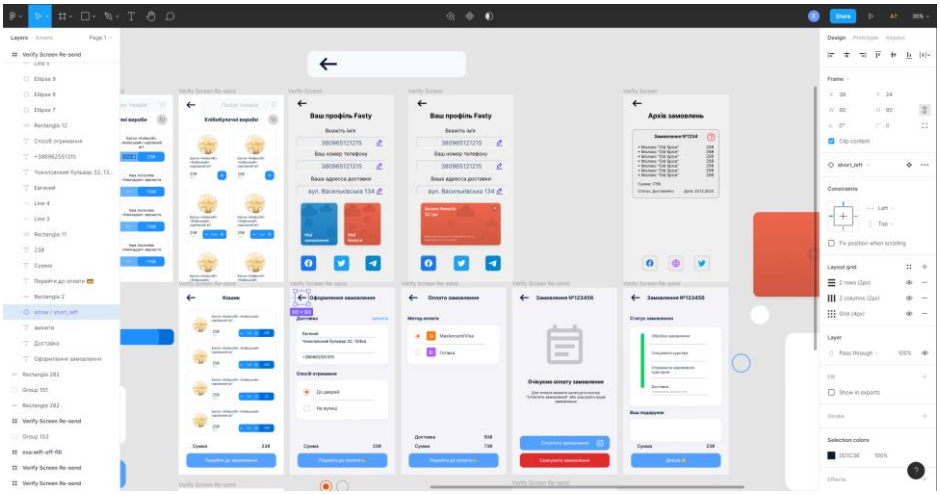


Рисунок 2.11 – Інтерфейс Figma

Завдяки безкоштовному використані додатку у розробників або дизайнерів є можливість використовувати усі функції які притамані таким популярним додаткам як Sketch. Figma підтримує можливості протитупування, експорту коду в формати CSS, обмін для перегляду та коментування іншим користувачам.

РОЗДІЛ 3. ПРОЕКТУВАННЯ ТА РЕАЛІЗАЦІЯ

3.1 Завдання та можливості програмного забезпечення

Завдання яке повинно виконувати додаток це надавати користувачеві можливість замовити продукти не виходячи з дому чи офісу.

У додатку важливо реалізувати необхідний функціонал для забезпечення користувачів можливості зручно їм користуватись. Для цього потрібно розробити інтуїтивно зрозумілий і легкий інтерфейс. Розроблюється, має підтримку телефонів з Android 5 і вище.

Таким чином в додатку повинні бути реалізовані такі функції:

- Реєстрація та авторизація користувача;
- Валідація користувача при реєстрації;
- Головний екран з вибором доступного функціоналу;
- Каталог продуктів за категоріями;
- Налаштування користувача;
- Форму оплати замовлення і статус його доставки.

Для початку роботи з додатком користувачеві потрібно буде пройти реєстрацію або авторизацію, якщо він вже зареєстрований. Наступним кроком користувач потрапляє на головний екран застосунку.

На головному екрані реалізована навігація по таким розділам додатку: каталог, кошик, настройки.

В розділі каталогу реалізована можливість пошуку товарів за назвою, відображення товарів за вибраною категорією. В інформації про товар містяться такі пункти як назва товару, грамовки, ціна. Особливо можна виділити ділення товару на дві категорії: поштучна та за вагою. Поштучні товари продаються лише за кількістю їхньою. За вагою тільки за кількістю ваги. Також в розділі реалізований зручний перехід у кошик та повернення до головного екрану.

В розділі налаштувань користувач має можливість змінити такі дані як: Ім’я, мобільний телефон, адресу доставки. Додатково до цього розділу включено такі можливості як перегляд історії замовлень та бонусна програма. В розділі кошику можна ознайомитись з вибраними товарами та перейти до сплати попередньо вибравши яким способом отримати замовлення та час доставки.

3.2 Моделювання об’єкту проектування

3.2.1 Створення моделі використання додатку

Для більш чіткого розуміння процесу розробки важливо опрацювати всі можливі варіанти використання, а так само знати які можливості матиме користувач.

Для відображення процесу використання програми є модель UML. UML - мова моделювання з системою позначень які застосовуються при об'єктно-орієнтованому аналізі і проектуванні [13, с. 36]. Його можна використовувати для візуалізації, конструювання, а також документування програмних систем. На рисунку 3.1 зображена діаграма використання програми «Fasty»

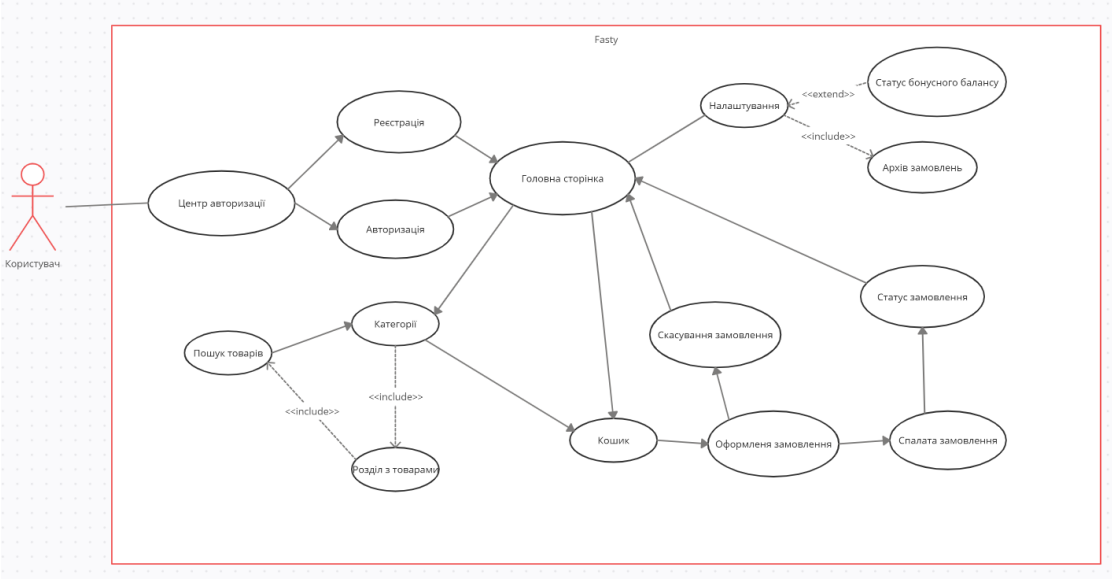


Рисунок 3.1 – UML діаграма викостання додатку «Fasty»

На побудованій діаграмі ми можемо побачити усі послідовні сценарії які приводять користувача до певних результатів.

3.2.2 Діаграма діяльності застосунку

Діаграма діяльності – діаграма за допомогою якої можна описувати логіку программи за допомою спеціальних блок-схем. Такі діаграми зазвичай використовують для опису логіку процедур, бізнес процесів.

Діаграма діяльності застосунку «Fasty» зображена на рисунку 3.2

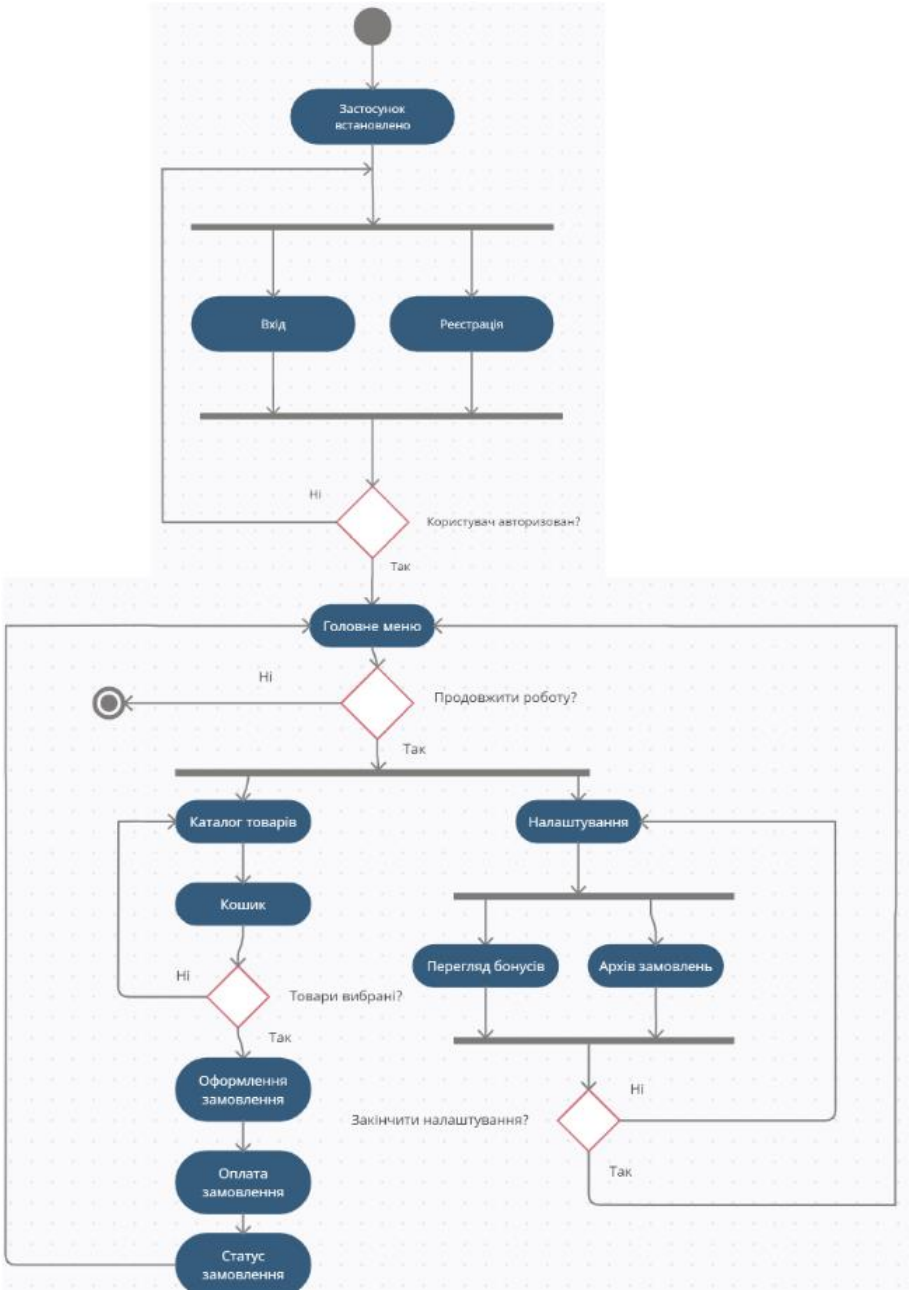


Рисунок 3.2 – Діаграма діяльності застосунку «Fasty»

Кожна з цих схем орієнтована на порядок виконання певних елементарних дій або операцій, які в сукупності призводять до бажаного результату.

3.3 Серверна частина проекту

3.3.1 API

Для роботи додатку «Fasty» йому необхідно створити серверну частину. В ролі серверної частини в нас буде виступати API.

API – програмний інтерфейс застосунку за допомогою якого можна задати функції та правила для взаємодії між програмним забезпеченням, яке надає API і іншими програмними компонентами. API було побудовано на основі REST API.

REST API – архітектурний стиль взаємозв’язку компонентів у мережі. Також REST надає певний набір обмежень при проектуванні розподілених систем.

Для підвищення якості API будемо використовувати тестування. За допомогою тестів є можливість виявити помилки та інші недоліки під час розробки.

Інструмент який використовувався під час тестування API називається Postman. Postman – програмне забезпечення яке призначене для роботи з API [15]. Має можливості такі як:

- Стоворення скриптів для роботи;
- Змінні для зберігання даних;
- Mock сервери;
- Колекції з запитами.

Таблиця 3.1 – Класи об'єктів API

Назва класу	Опис класу
Auth	Відповідає за авторизації, реєстрацію і підтвердження користувача
Order	Відповідає за створення, оновлення, обробки замовлень, обробки змін кошика
Product	Відповідає за вивід каталогу за певною категорією та пошук в категоріях
Profile	Відповідає за реєстрацію, авторизацію користувачів та їх підтвердження. Також виконує функції оновлення даних користувача.
Wayforpay	Використовується для створення інвойсів для оплати і обробки даних з Wayforpay

На рисунку 3.3 приведено приклад тестування створеного API за допомогою тестових запитів в Postman.

Рисунок 3.3 – Тестування API в Postman

За допомогою тестування API можна виправити усі проблеми на стадії розробки та заощадити час на перевірці ПЗ.

3.3.2 Шифрування даних та захист передачі

Передача даних вимагає захисту від шахраїв і інших вразливостей. Для цього для передачі даних будемо використовувати SSL.

SSL – криптографічний протокол який забезпечує безпечну передачу даних в Інтернеті. У проекті він буде реалізований для протоколу HTTP.

Так як проект використовує не тільки API але і CDN для доставки контенту. Ми будемо використовувати сервіс CloudFlare. За допомогою цього сервера є можливість застосовувати наскрізне шифрування через розподілену мережу CloudFlare. За допомогою даного сервісу вдається створити CDN мережу яка дозволяє в середньому збільшити швидкість отримання ресурсів з сайту на 20-30%. На рисунку 3.4 зображено схему проходження трафіку від мобільного додатку до серверу API.

Рисунок 3.4 – Схема проходження трафіку через сервіс CloudFlare

3.3.3 Використання бази даних MySQL

Для роботи с з даними та їх зберіганням було обрано БД MySQL. MySQL – реляційна система управління базами даних. Реалізовує модель «клієнт-сервер».

Розглянемо структуру бази даних застосунку «Fasty». На рисунку x.x ми можемо побачити такі таблиці: cart, categories, orders, products, users, wayforpay_log.

Таблиця cart зберігає такі дані як:

- Id – унікальний індитифікатор запису;
- Order_id – Номер замовлення;
- Product_id – Код товару з таблиці products;
- Amount – Кількість товару;
- Multiplier – Коефіцієнт товару (якщо цей товар продається за вагою).

Таблиця categories зберігає такі дані:

- Id – унікальний індетифікатор запису в таблиці;
- Name – назва категорії;
- Description – опис категорії;
- Created – дата та час створення запису;
- Modified – дата та час оновлення запису.

Таблиця orders зберігає такі дані як:

- Order_id – унікальний індентифікатор номеру замовлення в таблиці;
- User_id – індетифікатор запису номеру користувача;
- Billing_log – номер запису чеку в таблиці wayforpay_log;
- Status - поточний статус замовлення;
- Progress – поточний статус прогрессу доставки замовлення;
- Created – дата та час створення запису.

Таблиця products зберігає такі дані як:

- Id – унікальний індентифікатор номеру продукту в таблиці;
- Name – назва продукту;
- Description – додатковий опис товару;
- Weight – вага товару;
- Price – ціна товару;
- Category_id – індетифікатор номеру категорії в таблиці categories;
- Image_name – назва рисунку для загрузки з CDN;

- Type – тип розрахунку ціни продукту (multi – за вагою, unit – за одиницю товару) ;
- Created – дата та час створення запису;
- Modified – дата та час оновлення запису.

Таблиця users зберігає такі дані як:

- Id – унікальний індентифікатор номера користувача в таблиці;
- Name – ім’я користувача;
- Email – email користувача;
- phoneNumber – мобільний телефон користувача;
- address – адреса для доставки замовлень;
- bonus_balance – бонусний баланс користувача;
- password – хешований пароль користувача;
- Created_date – дата та час створення запису.

Таблиця wayforpay_log зберігає такі дані як:

- Id – унікальний індентифікатор номера інвойсу в таблиці;
- orderReference – індетифікатор номера замовлення з таблиці orders;
- merchantSignature – підпис даних за допомогою хеш функції;
- amount – сума замовлення в інвойсі;
- currency – валюта замовлення в інвойсі;
- authCode – код авторизації присвоєний банком;
- email – email клієнта;
- phone – мобільний телефон клієнта;
- cardPan – маскований номер карти;
- cardType – тип карти: Visa / MasterCard;
- issuerBankCountry – країна реєстрації картки;
- issuerBankName – банк реєстрації картки;
- transactionStatus – статус транзакції;
- reason – причина відмови;

- paymentSystem - платіжна система, через яку був здійснений платіж;
- created - дата та час створення запису у форматі timestamp.

Загалом структуру таблиць та їх зв'язків можна побачити на рис. 3.5.

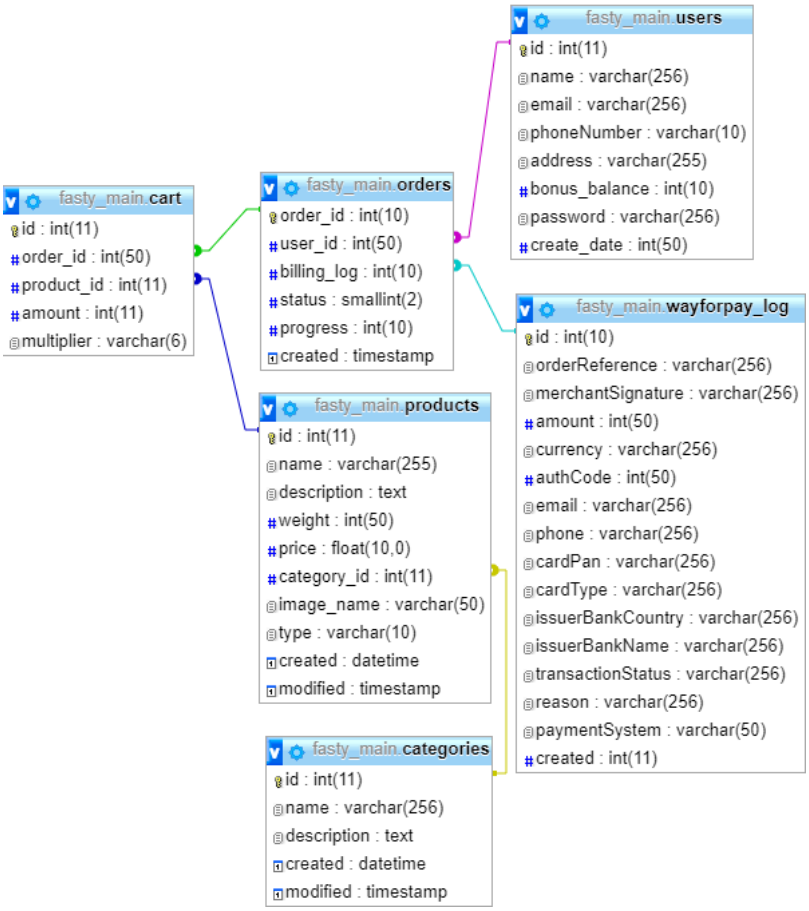


Рисунок 3.5 – Огляд структури БД в phpMyAdmin

Для роботи з застосунком база була інтегрована в створене API за допомогою якого виконується передача даних в форматі JSON.

3.4 Клієнтська частина застосунку

3.4.1 Архітектура проекту

У застосунку було використано клієнт-серверну архітектуру. Вона поділяється на дві ролі клієнта та сервера.

Клієнт - локальний пристрій за допомогою якого виконується відправка даних на сервер для можливості отримання або відправки даних.

Сервер - системне обладнання, яке призначається для вирішення певного кола завдань з обробки вхідних і вихідних даних. Так само він надає користувачеві доступ до системних ресурсів і зберігає дані в базі даних.

Особливості такої моделі полягають в тому, що користувач відправляє певний запит на сервер, де той системно обробляється і кінцевий результат відсилається клієнту. У можливості сервера входить одночасне обслуговування відразу декількох клієнтів.

Процеси, якими займається сервер:

- Обробка вхідних запитів;
- Відправка відповіді клієнту;
- Зберігання та захист даних.

Процеси, якими займається клієнт:

- Формування запитів та їх відправка;
- Отримання відповідей та їх подальша обробка;
- Майданчик для отримання зовнішніх ресурсів графічного інтерфейсу.

На рисунку 3.6 зображено уявлення взаємодії описаної архітектури.

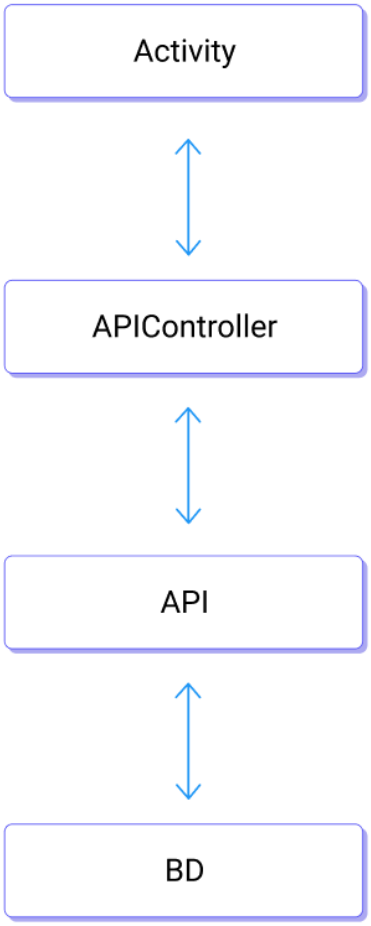


Рисунок 3.6 – Графічне уявлення взаємодії кліент-серверної архітектури

Тим самим така реалізація архітектури надає змогу застосунку отримувати актуальну інформацію і повноцінно працювати.

3.4.2 Створення макетів та компонентів дизайну

Для початку розробки розробки дизайну додатку попередньо потрібно при створені проекту в Unity вибрати тип застосунку який будемо розроблювати. Серед доступних для розробки є такі готові шаблони:

- 2D;
- 3D;
- High Definition RP;
- Universal Render Pipeline.

Так як в нас застосунок звичайне без використання 3D та іншої графіки оберемо 2D та створимо проект. Див рисунок 3.7

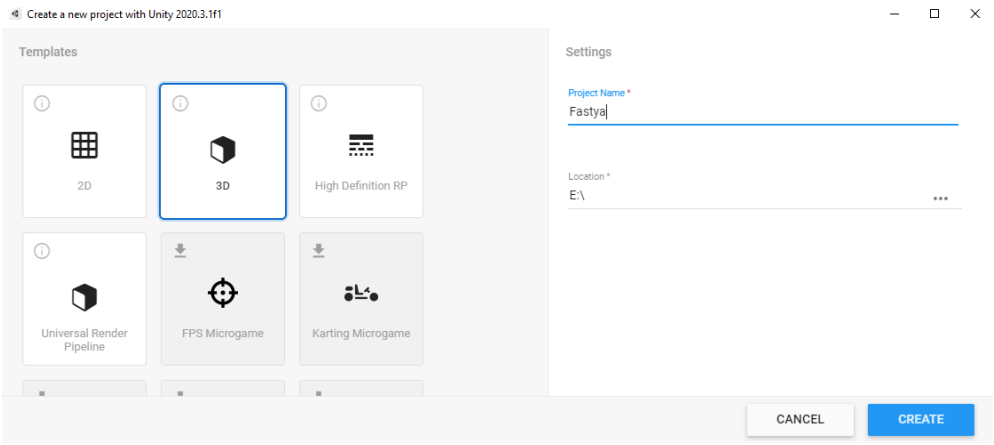


Рисунок 3.7 – Вибір шаблону створюваного проекту

Після створення проекту у нас завантажується екран розробки на якому зображені усі інструменти для подальшого проектування. Для створення перших макетів за допомогою інструмента “Ієрархія” ми починаємо створювати такі важливі частину інтерфейсу як: Main Camera та Canvas. На рисунку 3.8 зображено єрархію елементів які знаходяться на сцені.

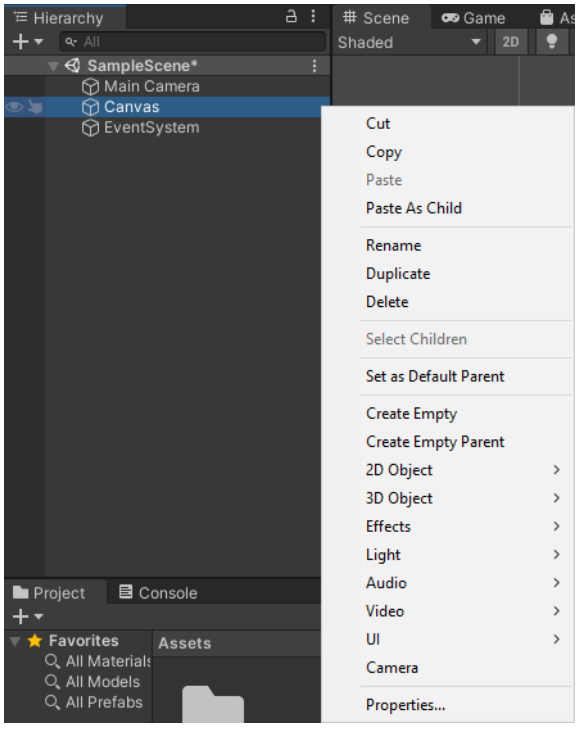


Рисунок 3.8 – Ієрархія елементів сцени

Main Camera відповідає за рендерінг робочої сцени та відображає його користувачу.

Canvas компонент який представляє з себе абстрактний простір в якому відбувається налаштування та відображення UI. Усі елементи UI повинні бути нащадками об’єктів к яким приєднаний компонент Canvas.

Також важливим інструментом для роботи з сценою та об’єктами є інспектор. На рисунку 3.9 зображено інспектор об’єктів Unity.

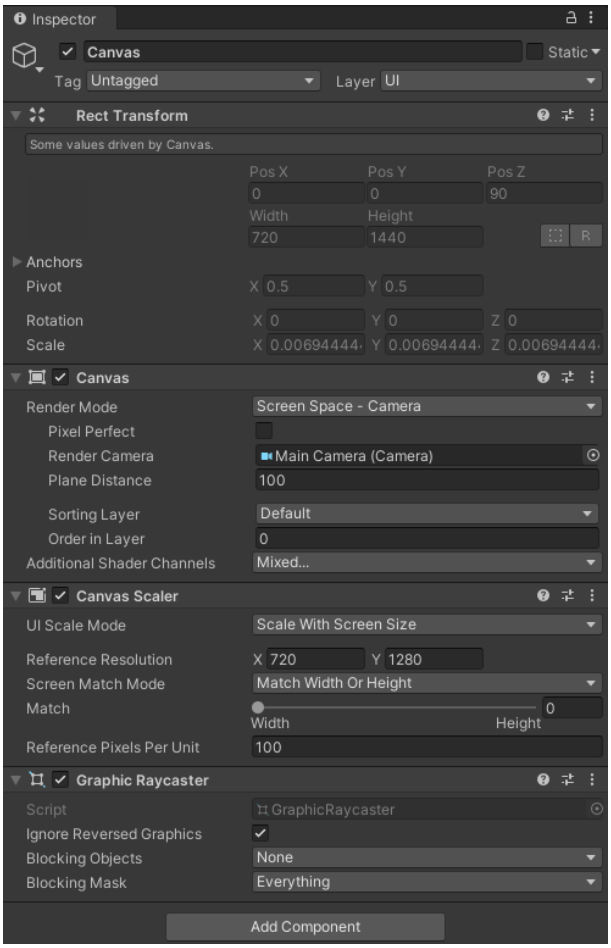


Рисунок 3.9 – Інспектор об’єктів

За допомогою інспектора можна передивитись основу інформацію про елементи які виділені та їх властивості. Люба властивість може бути змінена за допомогою цього інструмента без модифікації самого скрипту. Якщо в скрипту визначили публічну зміну, то в інспекторі можна задати їй будь-яке значення з доступних.

3.4.3 Класи взаємодії з серверною частиною API

Для розуміння роботи додатку потрібно розглянути основні класи які відповідають за взаємодію елементів застосунку. Один з найважливіших класів у додатку є контроллер API.

Для роботи з даними у застосунку та для реалізації клієнт-серверної архітектури було створено клас APIController. Для роботи з даними які приходять з серверу та передачі на сервер використовується JSON.

У мові програмування відсутні можливості роботи з форматом JSON. Тому реалізована серіалізація та десереалізація. Серіалізація відповідає за процес перетворення байтів у текстову інформацію. Десеріалізація є оберненим процесом до серіалазіції. Реалізація серіалізації та десереалізації була використана в контроллері API.

Розглянемо деякі методи які реалізовані у класі APIController:

- CheckEmail – метод за допомогою якого виконується перевірка наявності електронної пошти в базі даних. Використовується клас під час реєстрації користувача та при присутності пошти повертає відповідний параметр. На рис. 3.10 зображено оформлення методу.
- GetProfile – метод який використовується при реєстрації або авторизації. За допомогою цього методу застосунок отримує важливу інформацію про користувача таку як: Ім’я, електрону пошту, адресу, баланс бонусів.
- GetCatalog – метод який визивається після вибору потрібного розділу каталогу їжі. За допомогою цього методу застосунок отримує масив інформації о товарах, цінах, наявності їх у кошику користувача.
- CreateInvoice – метод для роботи з еквайрингом. До нього передається електронна пошта користувача та після відправки запиту на сервер API. На сервері формується остаточна ціна товару та інформація користувача про тип доставки, час, тип та

повертає посилання для проведення оплати. Також існують схожі на цей метод методи. До таких відносятся CheckInvoice та DeleteInvoice. Виконують схожі функції але CheckInvoice робить перевірку статусу інвойсу, а DeleteInvoice скасування його.

```
public IEnumerator checkEmail(string email, Action<bool> action)
{
    if (CheckInternetConnection())
    {
        RequestToEmail regEmail = new RequestToEmail();
        regEmail.email = email;
        string json = regEmail.SaveToString();
        var request = new UnityWebRequest(ApiTarget + "/auth/verify.php", "POST");
        byte[] jsonToSend = new System.Text.UTF8Encoding().GetBytes(json);
        request.uploadHandler = (UploadHandler)new UploadHandlerRaw(jsonToSend);
        request.downloadHandler = (DownloadHandler)new DownloadHandlerBuffer();
        request.SetRequestHeader("Content-Type", "application/json");
        yield return request.SendWebRequest();
        if (request.isNetworkError || request.isHttpError)
        {
            Debug.Log(request.error);
            Debug.Log(request.downloadHandler.text);
        }
        else
        {
            Debug.Log(request.downloadHandler.text);
            Request ApiReguest = Request.CreateFromJSON(request.downloadHandler.text);
            Debug.Log(ApiReguest.message);
            if (ApiReguest.message.Equals("1"))
            {
                action(true);
            }
            else
            {
                action(false);
            }
        }
    }
```

Рисунок 3.10 – Метод CheckEmail

3.4.4 Реєстрація та авторизація користувача

На головному екрані додатку нас зустрічає сторінка авторизації. На ній ми можемо обрати авторизуватися чи зареєструватися. Розглянемо основну процедуру реєстрації користувача.

Для початку реєстрації нам потрібно вибрати поле “Не зареєстровані?”. Далі потрібно ввести такі дані як: email та пароль. Далі система робить запит до API з перевіркою інформації про реєстрацію вказаного email. Після перевірки пошти користувач потрапляє на сторінку підтвердження електронної скриньки. На вказану електронну пошту відправляється 4-х значний код підтвердження який потрібно ввести в поле для вводу. Після

успішної перевірки актуальності додаток відправляє запит на API для реєстрації користувача в системі. Далі користувач потрапляє в головне меню додатку та може користуватись.

3.4.5 Завантаження меню каталогу

Розглянемо таку важливу частину застосунку як каталог продуктів. Всього в застосунку 9 категорій продуктів:

- Хліб та хлібобулочні вироби;
- М’ясо, риба, птиця;
- Фрукти та овочі;
- Молочні вироби та яйця;
- Кулінарія;
- Кава та чай;
- Напої;
- Солодощі;
- Снеки.

На кожній категорії є спеціальний унікальний індентифікатор категорії за допомогою якого додаток робить запит через клас APIController. Та відбувається перехід на сторінку каталогу з згенерованими панелями товарів. На рисунку 3.11 зображено приклад відкритої категорії продуктів. Також важливо підмітити, що товари можуть продаватись як поштучно так і за вагою. Тому у нас є класи такі як Catalog_Generator та ProductInfo.

Catalog_Generator – клас за допомогою якого виконується запит до контроллеру API та подальша генерація і заповнення каталогу товарів. Так само картинки які використовуються для товарів завантажуються з сервера CDN тим самим роблячи можливість змінювати інформацію про товар і її фотографії в режимі реального часу.

ProductInfo – клас який містить в собі таку інформацію про товар: Назву, опис, вагу, ціну, тип товару. Також відповідає за розрахунок вартості товару залежно від його типу.

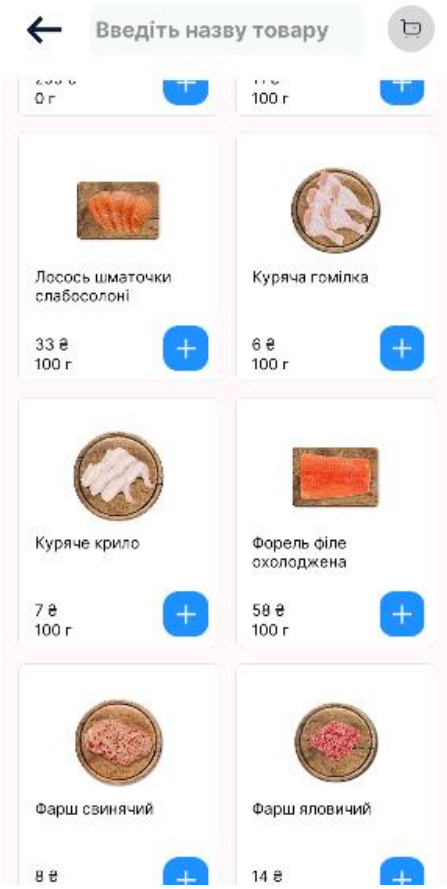


Рисунок 3.12 – Відкритий розділ продуктів

3.4.6 Використання еквайрингу від WAYFORPAY для проведення платежів

Для обробки замовлень в автоматичному режимі та її оплаті було обрано еквайрінг від WAYFORPAY. Алгоритм звя’зку з еквайрінгом був реалізований за даним алгоритмом. При створенні рахунку додаток робить POST запит на API з командою сформувати рахунок. Після створення рахунку еквайрінг повертає нам відповідь в якості посилання на QR код для оплати та посилання на сторінку оплати. Окремо можна зазначити, що є можливість скасування замовлення через спеціальну кнопку в додатку. Отже, після успішної оплати або скасування WAYFORPAY робить запит на

технічне посилання API зі статусом рахунку. Увесь цей час додаток виконує циклічні запити до API з перевіркою стану рахунку.

Розглянемо основні методи які відповідають за роботу модуля оплати рахунків у додатку.

- CheckInvoiceStatus метод який відповідає за обробку замовленя. За допомогою нього клієнт відсилає запити до API з перевіркою інформації по оплаті інвойсу.
- CheckOrderStatus метод який містить інформацію про актуальний стан замовлення. Постійно робить запити до API отримуючи інформацію по замовленю.
- InvoiceManualPay – метод за допомогою якого користувач переходить на сторінку оплати та містить інформацію про посилання на інвойс в WAYFORPAY.

На рисунку 3.13 можна побачити інтерфейс для переходу для оплати замовлення. Також на рисунку 3.14 відображено моніторинг статусу замовлення.

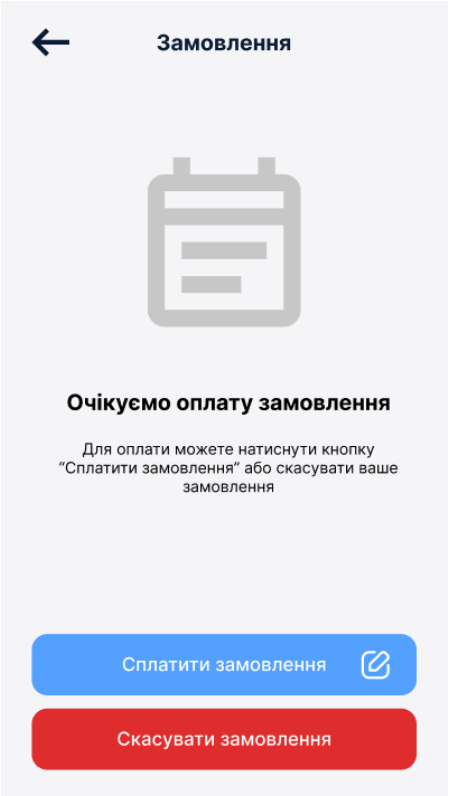


Рисунок 3.13 – Інтерфейс оплати замовлення

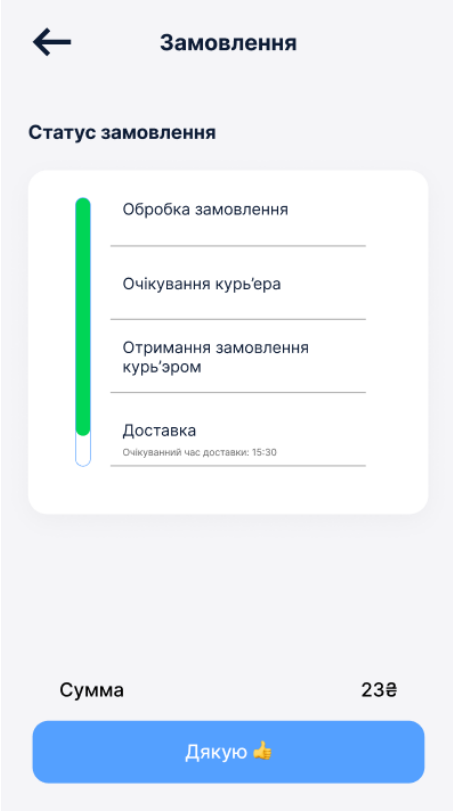


Рисунок 3.14 – Інтерфейс моніторингу статусу замовлення

Отже, ми можемо побачити, що усі основні функціональні можливості притамані розділу з оплатою замовлення реалізовано.

ВИСНОВОК

На основі проведених аналітичних досліджень було розроблено програмний застосунок, що здійснює обробку замовлень клієнтів та оптимізує бізнес процеси у сфері доставки продуктів.

Було проведено дослідження попиту сервісів доставок продуктів на ринку яке підтверджує актуальність роботи та наукову новизну. А саме, тенденції розвитку доставок в Україні та огляд аналогів додатків по замовленню продуктів.

Розглядаючи переваги та недоліки існуючих програмних засобів, було визначено основні функціональні вимоги до ПЗ та шляхи для вдосконалення процесу взаємодії користувача з додатком.

Також було розглянуто основні засоби для розробки інтерфейсу застосунку. Встановлено, що досить зручною мовою програмування для додатку є C# та середовище розробки Unity які вміщує в собі переваги в швидкості обробки та якості ПЗ. Додатково к застосунку було розроблено серверну частину з використанням мови програмування PHP, в якості API для реалізації клієнт-серверної архітектури.

Для тестування якості обробки запитів API було використано інструмент Postman.

Розроблений програмний застосунок складається з таких модулів:

- Авторизація;
- Каталог товарів;
- Оплата замовлення;
- Особистий кабінет користувача.

Додаток буде ефективним для роздрібних торгових мереж які потребують наявність зручного засобу для взаємодії з клієнтом. Особливої актуальності набуває у часи епідемії COVID-19, оскільки у людей які перебувають на самоізоляції відсутня можливість відвідувати торгові мережі.

СПИСОК ЛІТЕРАТУРИ

1. Чистяков. А.Л. Как заработать на доставке еды. Из пункта А в пункт $: Монографія / А.Л.Чистяков – Москва.: Ресторанні відомості, 2020. – 180 с.
2. Страшинська Л.В. Стратегія розвитку продовольчого ринку в Україні: Монографія / Л.В.Страшинська / За ред. Б.М. Данилишина. –К.: Профі, 2008. – 627 с.
3. Чинтан М. MySQL 8 для больших данных - Чинтан Мехта / Мехта Чинтан., 2018. – 226 с.
4. Эндрю Т. Язык программирования C# 7 и платформы .NET и .NET Core / Т. Эндрю, Д. Филипп., 2019. – 1328 с.
5. Джеффри, Д.У. Основы реляционных баз данных. – Лори, 2016. – 382 с.
6. Stellman A. Head First C#: A Learner's Guide to Real-World Programming with C# and .NET Core / A. Stellman, J. Greene., 2020. – 1697 с.
7. Гультяев А.К. Проектирование и дизайн пользовательского интерфейса – С- Пб: Корона-принт, 2010 г. — 349 с.
8. Google Play Market [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: www.play.google.com (дата звернення 15.02.2021).
9. Unity [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: unity.com.
10. Figma [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: figma.com.
11. Хлебников, А.А. Информационные технологии: Учебник / А.А. Хлебников. — М.: КноРус, 2014. — 472 с
12. Ian G. Clifton. «Android User Interface Design: Implementing Material Design for Developers (2nd Edition)», 2017
13. Hay D. UML and Data Modeling: A Reconciliation / David Hay., 2011. – 242 с.

14. Visual Studio [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: visualstudio.microsoft.com.
15. Postman [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: www.postman.com.
16. Statcounter [Електронний ресурс] : [Веб-сайт]. – Електронні дані. – Режим доступу: www.gs.statcounter.com.

ДОДАТОК А

Код класу реєстрації Registration

```
using System.Collections;
using System.Collections.Generic;
using TMPro;
using UnityEngine;
using UnityEngine.SceneManagement;
using UnityEngine.UI;

public class Registration : MonoBehaviour
{
    public GameObject ReSendCodeBlock;
    public TMP_InputField emailInputfield;
    public TMP_InputField passwordInputfield;
    public TMP_InputField AuthEmailInputfield;
    public TMP_InputField AuthPasswordInputfield;
    public TextMeshProUGUI alertMessage;

    public GameObject FirstStage;
    public GameObject SecondStage;
    public GameObject ThirdStage;

    public TMP_InputField firstInputfield;
    public TMP_InputField secondInputfield;
    public TMP_InputField thirdInputfield;
    public TMP_InputField fourthInputfield;

    public Text errorCount;

    string email;
    string password;

    APIController api = new APIController();
    public void Init()
    {
        email = emailInputfield.text;
        password = passwordInputfield.text;
        StartCoroutine(api.checkEmail(email,(isConnected) => {
            if (isConnected)
            {
                alertMessage.gameObject.SetActive(true);
```

```
            }
            else
            {
                alertMessage.gameObject.SetActive(false);
                Debug.Log(password.Length);
                if (password.Length >= 6)
                {
                    string passCode = Random.Range(1111, 9999).ToString();
                    StartCoroutine(api.sendMail(email, passCode, (isDeparted) => {
                        if (isDeparted)
                        {
                            PlayerPrefs.SetString("password", password);
                            SecondStageInit();
                        }
                        else
                        {

                        }
                    }));
                }
                else
                {
                    errorCount.gameObject.SetActive(true);
                }
            }
        }));
    }
    public void checkVerifyCode()
    {
        email = emailInputfield.text;
        password = passwordInputfield.text;
        string code = firstInputfield.text + secondInputfield.text + thirdInputfield.text
+ fourthInputfield.text;

        if (code.Equals(PlayerPrefs.GetString("passCode")))
        {
            StartCoroutine(api.sendRegistration(email,
PlayerPrefs.GetString("password"), (isRegistered) => {
                if (isRegistered)
                {
                    StartCoroutine(api.getProfile(email, (isOk) =>
                    {
                        if (isOk)
                        {
```

```
                    SceneManager.LoadScene("Main");
                }
                else
                {

                }
            }));
        }
    }));
  }

}
public void ReSendCode()
{
  string passCode = Random.Range(1111, 9999).ToString();
  StartCoroutine(api.sendMail(email, passCode, (isDeparted) => {
    if (isDeparted)
    {
      PlayerPrefs.SetString("password", password);
    }
  }));

}
public void InitAuth()
{
  email = AuthEmailInputfield.text;
  password = AuthPasswordInputfield.text;
  StartCoroutine(api.AuthUser(email, password, (isAuth) => {
    if (isAuth)
    {
      StartCoroutine(api.getProfile(email, (isOk) =>
      {
        if (isOk)
        {
          SceneManager.LoadScene("Main");
        }
      }));
    }
  }));

}
public void SecondStageInit()
{
  FirstStage.gameObject.SetActive(false);
```

```
        SecondStage.gameObject.SetActive(true);
    }
    public void AuthStageInit()
    {
        FirstStage.gameObject.SetActive(false);
        SecondStage.gameObject.SetActive(false);
        ThirdStage.gameObject.SetActive(true);
    }
    public void RegStageInit()
    {
        FirstStage.gameObject.SetActive(true);
        SecondStage.gameObject.SetActive(false);
        ThirdStage.gameObject.SetActive(false);
    }

}
```

ДОДАТОК Б

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка мобільного сервісу для замовлення продуктів на мові програмування C#»

Виконав студент 4 курсу
групи ПД-42
Шульчевський Є.О.
Керівник роботи
Жебка В.В.

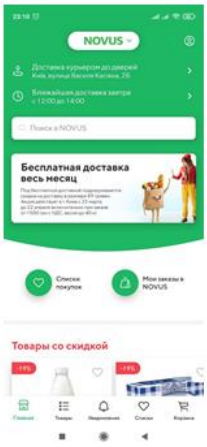
Київ – 2021

МЕТА, ОБ'ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

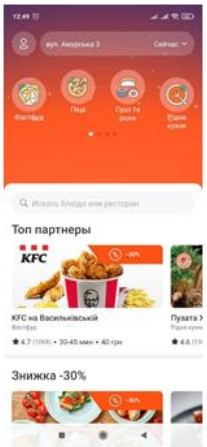
- **Мета роботи** – розробити ефективне програмне забезпечення яке дозволить замовляти продукти харчування через мобільний додаток
- **Об'єкт дослідження –** підвищення якості обслуговування клієнтів сфери доставок продуктів.
- **Предмет дослідження –** додаток для системи замовлення та доставки продуктів клієнтам.

АНАЛОГИ

Glovo



Zakaz.ua



Rocket

Bolt Food

Noww

АНАЛОГИ

Частину переваг та недоліків було враховано при розробці нового ПЗ. Створений додаток отримав назву «Fasty». Він призначений для швидкого замовлення та інформування користувача о усіх етапах оформлення та отримання замовлення.

Назва застосунку	Переваги	Недоліки
Glovo	- Проста у використані - Великий вибір різноманітних кафе, ресторанів тощо - Доступність в багатьох регіонах України	- відсутній вибір мови - відсутня можливість скасувати замовлення - низька якість застосунку
Zakaz.ua	- Проста у використані - Великий вибір супермаркетів - Стабільна робота застосунку	- Навантажений інтерфейс
Rocket	- можливість вибору мови з трьох представлених - Великий вибір різноманітних кафе, ресторанів тощо	- відсутній пошук по категоріям магазинів
Bolt Food	- можливість вибору мови з сімнадцяти різноманітних - Розділений інтерфейс - Блок з різноманітними акціями	- доступність лише в Києві
Noww	- розділ з унікальними пропозиціями - система кешбеку - можливість вибору мови з трьох представлених	- відсутній пошук по категоріям - доступність у декількох районах Києва

ТЕХНІЧНІ ЗАВДАННЯ

- *проаналізувати аналогічні додатки на ринку доставок продуктів;*
- *визначити основні функціональні вимоги до ПЗ;*
- *спроектувати архітектуру програми;*
- *Розробити клієнтську частичну додатку;*
- *Розробити серверну частину;*

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

Програмні засоби які використовувались для реалізації ПЗ:

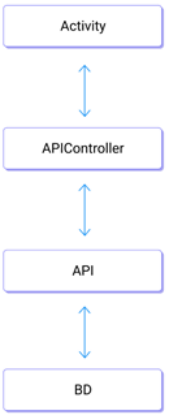
- Unity;
- Visual Studio
- Figma;
- Postman;
- phpMyAdmin;

МЕТОДИ ТА КЛАСИ ПРОГРАМИ

В програмному забезпеченні реалізована модель клієнт-серверна архітектури. Тобто йде тісна взаємодія клієнта та сервера, а саме API.

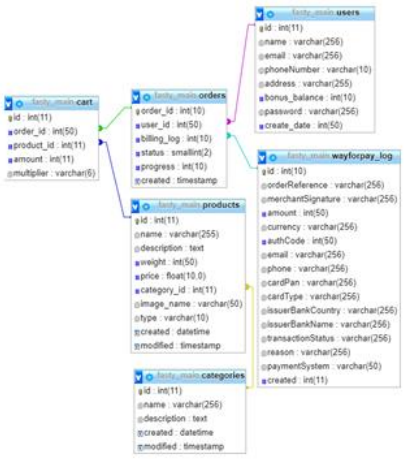
Програмнее забезпечення загалом складається з 9 основних класів:

- APIController – відповідає за роботу з API;
- Cart – клас для роботаю з кошиком користувача;
- Catalog – клас для роботою з каталогом;
- Catalog_Generator – клас для завантаження меню каталогу;
- Checkout – клас для роботи з замовленнями;
- FormsGenerator – клас для генерації форми;
- PaymentParser – клас для моніторингу статусу замовлення;
- Registration – клас для роботи з авторизацією та реєстрацією;
- Settings – клас для налаштувань користувача.



Модель клієнт-серверної архітектури

Структура бази даних та серверної частини



Структура таблиць бази даних

Назва класу	Опис класу
Auth	Відповідає за авторизації, реєстрацію і підтвердження користувача
Order	Відповідає за створення, оновлення, обробки замовлень, обробки змін кошика
Product	Відповідає за вивід каталогу за певною категорією та пошук в категоріях
Profile	Відповідає за реєстрацію, авторизацію користувачів та їх підтвердження. Також виконує функції оновлення даних користувача.
Wayforpay	Використовується для створення інвойсів для оплати і обробки даних з Wayforpay

Класи об'єктів API

Опис розробленого додатку

Процесс реєстрації та авторизації

Для початку роботи з додатком потрібно пройти процес авторизації. На рис. 1 представлена форма авторизації, а так-же можливість пройти реєстрацію.

Додатково для реєстрації використовується додаткове підтвердження за допомогою електронної пошти клієнта. Див рис. 2

Також реєстрація і авторизація виконуються за допомогою запитів до створеного API.

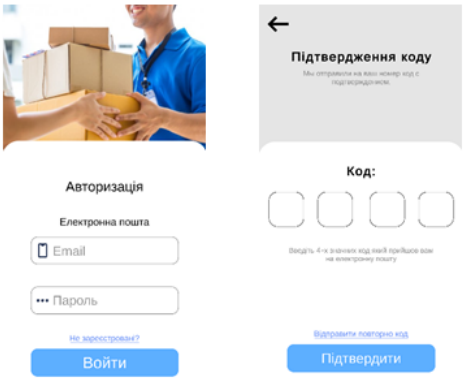


Рисунок 1 – Форма авторизації

Рисунок 2 – Форма авторизації

Опис розробленого додатку

Меню каталогу та кошик замовлення

Для замовлення продуктів реалізовані розділи каталогу. На рис. 3 можна побачити відкритий розділ молочної продукції.

Для оптимизації завантаження продуктів і актуальної інформації про товари використовується API. За допомогою класу APIController виконується завантаження інформації про товари та їх картинки.

Також був реалізован кошик (рис. 4) для перегляду товарів на замовлення.
Для нього також використовується класс APIController для підвантаження даних з бази даних.

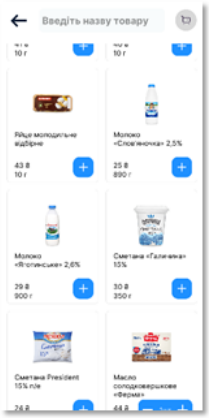


Рисунок 3 – Відкритий розділ молочної продукції

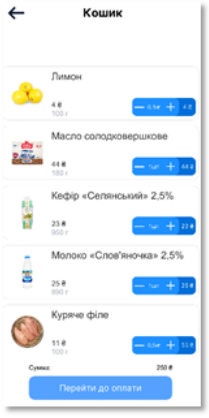


Рисунок 4 – Кошик користувача

Опис розробленого додатку

Оформлення та оплата замовлення

Оформлення замовлення оформлюється за допомогою таких параметрів: товару у кошику, контактної інформації користувача та часу і способу отримання (рис. 5).

Для оплати замовлення був підключений еквайрінг від WayForPay. Після переходу до оплати замовлення відкривається сайт платіжної системи для оплати замовлення (рис. 6).

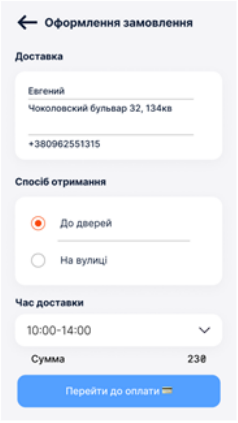


Рисунок 5 – Відкритий розділ молочної продукції

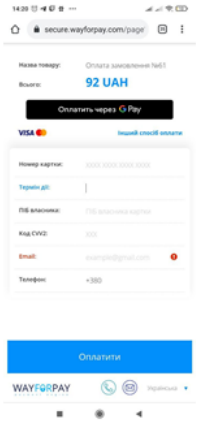


Рисунок 6 – Кошик користувача

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

- Шульчевський Є.О. Аналіз тенденцій розвитку ринку доставки продуктів харчування // Сучасні інфокомунікаційні технології: Матеріали XI НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ СТУДЕНТІВ ТА МОЛОДИХ ВЧЕНИХ. Збірник тез. 11.12.2020, К.ДУТ, 2020 – 91с.
- Шульчевський Є.О. Огляд аналогів сервісів з доставки продуктів // Застосування програмного забезпечення в інфокомунікаціних технологіях: Матеріали ВСЕУКРАЇНСЬКОЇ НАУКОВО-ТЕХНІЧНОЇ КОНФЕРЕНЦІЇ ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОКОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ. Збірник тез. 12.02.2021, К.: ДУТ, 2021 – 129с.

ВИСНОВКИ

1. Сервіси з доставок продуктів будут становитися дедалі популярнішими на ринку.
2. Розглянуто основні аналоги додатків з доставки продуктів.
3. Було визначено оптимальні програмні засоби та мову програмування для реалізації програмного забезпечення.
4. Розглянувши переваги та недоліки існуючих програмних засобів, було визначено основні функціональні вимоги до ПЗ та шляхи для вдосконалення процесу взаємодії користувача з додатком.

Особливої актуальності набуває у часи епідемії COVID-19, оскільки у людей які перебувають на самоізоляції відсутня можливість відвідувати торгові мережі. Тому на ринку доставок з'являється попит на такі сервіси.