Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку
Назва теми

Рівень вищої освіти Перший(бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121«Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ. 200119.01.02.ПЗ

Виконав студент ІІІ курсу група ІПЗс-20-1 _____ А.І. Балагур
Підпис Ініціали, прізвище

Керівник канд. пед. наук, доцент _____ О.Г. Онишко
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд.техн.наук, доцент _____ І.В. Гурман
Науковий ступінь, звання Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

2 червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
05 02 2023 р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

Балагуру Андрію Ігоровичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку

Керівник проекту (роботи) Онишко Оксана Григорівна, канд. пед. наук, доцент
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2023 р. № 6
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Три креслення: UML –діаграма варіантів використання, UML – діаграма класів, UML – діаграма розгортання

6. Консультанти розділів кваліфікаційної роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Гурман І. В., к.т.н, доцент кафедри ІПЗ	02.06.2023	02.06.2023
Антиплагіат	Гурман І. В., к.т.н, доцент кафедри ІПЗ	02.06.2023	02.06.2023

7. Дата видачі завдання « 02 » січня 2023р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) кваліфікаційної роботи	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою кваліфікаційної роботи (КР), визначення та узгодження індивідуальних тем КР	01.12– 31.12.2022	
2 Збір матеріалу за темою КР; дослідження предметної області, в якій планується використання програмного забезпечення (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02 2023	
4 Програмна реалізація з використанням відповідних засобів розробки	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення переліку джерел посилання та додатків. Оформлення пояснювальної записки КР згідно вимог	01.05 – 25.05.2023	
7 Попередній захист КР	Травень 2023 (згідно графіка)	
8 Перевірка КР на плагіат, нормоконтроль, отримання відгуків, рецензій та інших супровідних документів. Брошурування (зшиття) пояснювальної записки.	26.05 – 30.05.2023	
9 Здача КР на кафедру; підготовка КР для розміщення у репозитарії ХНУ; підготовка до захисту та захист КР	з 01.06.2023	

Студент _____ А.І. Балагур
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ О.Г. Онишко
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку

Автор роботи: Балагур Андрій Ігорович.

Керівник роботи: Онишко Оксана Григорівна.

Пояснювальна записка: 67 с.,32 рис.,2 табл., 4дод., 41 джерел.

Графічна частина: 21 слайдів.

МОБІЛЬНИЙ ДОДАТОК, MYSQL, NODEJS, REACT NATIVE, БАЗА ДАНИХ

Метою кваліфікаційної роботи є розробка повнофункціонального програмного забезпечення для оптимізації роботи кав'ярень, скорочення часу виконання щоденних операцій та забезпечення точного обліку даних, що сприятиме ефективному управлінню бізнесом.

В кваліфікаційній роботі було виконано аналіз предметної області, проведено аналіз існуючих альтернативних мобільних додатків, проведено аналіз існуючих інструментів та технологій розробки мобільних додатків з клієнт-серверною архітектурою, визначені модулі системи та здійснена їх програмна реалізація.

Для розробки застосунку була використана мова програмування JavaScript, Node.js, фреймворки React Native та ExpressJS, база даних MySQL.

В результаті було розроблено мобільний додаток для працівників, що оптимізує роботу зі складом кав'ярні та покращує роботу з замовленнями. Розроблено веб додаток для власника з метою контролю обігу продуктів, замовлень та роботи працівників.

01.06.2023
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ. 200119.01.02.ПЗ	Пояснювальна записка	67		
2	А4		Завдання на кваліфіка-	1		
			ційну роботу			
3	А4		Анотація	1		
			Графічні документи			
5	А3	КвРІПЗ. 200119.01.02.Е8	UML-діаграма варіантів	1		
			використання			
6	А3	КвРІПЗ. 200119.01.02.Е8	UML-діаграма	1		
			розгортання			
7	А3	КвРІПЗ. 200119.01.02.Е8	UML-діаграма класів	1		

Змн.	Арк.	№ докум.	Підпис	Дата	ЗППШПЗ.200119.01.02.ВД		
Виконав.		Балагур А.І.			Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку Відомість документів	Літ.	Арк. 1 / Акрушів 1
Керівник.		Онишко О.Г					
Рецензент						ХНУ, ІПЗс-20-1	
Н. Контр.		Гурман І.В.		2.06			
Затверд.		Бедратюк Л.П.		2-06			

ЗМІСТ

ВСТУП **6**

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ **8**

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей 8

1.2 Аналіз наявного програмно-технічного забезпечення предметної області .. 9

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання 17

1.4 Висновки. Постановка задачі 21

2 ПРОЕКТУВАННЯ ЗАСТОСУНКУ **22**

2.1 Аналіз та вибір архітектури застосунку 22

2.2 Опис структури даних та моделі бази даних 25

2.3 Проектування серверної частини застосунку 31

2.4 Проектування інтерфейсу мобільного застосунку 33

2.5 Аналіз та вибір технологій і методів реалізації застосунку 35

3 ПРОГРАМНА РЕАЛІЗАЦІЯ **42**

3.1 Розробка бази даних 42

3.2 Розробка програмних модулів 44

3.3 Керівництво користувача 51

3.4 Технічні характеристики застосунку 55

3.5 Розгортання та встановлення системи 57

3.6 Тестування застосунку 59

ВИСНОВКИ **62**

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ **63**

ДОДАТОК А 68

ДОДАТОК Б 73

ДОДАТОКВ 75

ДОДАТОК Г 97

ВСТУП

У сучасному світі робота в кафе стає все більш популярною, і водночас зростає потреба в ефективному управлінні бізнесом. Одним із способів покращити роботу кав’ярні є використання програмного забезпечення для автоматизації повсякденних операцій і забезпечення точного запису даних.

Метою проекту є розробка програми для автоматизації роботи кав’ярні з використанням обліку даних. Метою дослідження є розробка повнофункціонального програмного забезпечення для оптимізації роботи кав’ярень, скорочення часу виконання щоденних операцій та забезпечення точного обліку даних, що сприятиме ефективному управлінню бізнесом.

Актуальність теми проекту полягає у тому, що для власників кав'ярень, управління бізнесом може бути складним завданням, особливо коли йдеться про управління фінансами та бухгалтерським обліком.

Додатки для автоматизації роботи кав'ярні можуть бути корисними для різних аспектів управління. Ось список деяких з їхніх потенційних застосувань:

– допомогти автоматизувати процес приймання та обробки замовлень, включаючи можливість замовлення через мобільні пристрої, автоматичне повідомлення про готовність замовлення та сповіщення для споживачів.

– надавати функціональність для резервації столиків, управління наявністю та розташуванням столиків, а також моніторингу зайнятості столиків.

– допомагати в управлінні меню кав'ярні, включаючи легке оновлення та налаштування списку продуктів, їх описів, цін та наявності.

– взаємодіяти з платіжними системами для прийому платежів в кав'ярні, надаючи зручні та безпечні способи оплати.

– вести облік запасів кав'ярні, автоматично поповнюючи запаси на підставі продажів та повідомляючи про потребу в поповненні запасів.

– надавати звіти та аналітику про продажі, витрати, популярність продуктів та інші ключові показники, що допомагають управлінцям приймати обґрунтовані рішення.

– містити функціональність для планування розкладу роботи, контролю робочого часу, заробітної плати та інших аспектів керування персоналом.

– надавати зручний спосіб спілкування з клієнтами, включаючи можливість замовлення через додаток, отримання сповіщень та збереження історії замовлень.

Очікується, що розроблений застосунок забезпечить кав'ярню зручним та ефективним інструментом для управління бухгалтерськими процесами, знизить час, затрачений на ведення обліку, а також дозволить зберігати дані в безпечному та надійному середовищі.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Сучасний розвиток кав'ярень та кавової культури вимагає ефективного управління та автоматизації різних аспектів бізнесу. Одним з важливих елементів є бухгалтерський облік, який забезпечує контроль над фінансовою діяльністю, включаючи прибуток, витрати, операції з постачальниками та клієнтами, аналіз фінансових показників та формування звітності.

Реалізація додатку для автоматизації роботи кав'ярень з використанням бухгалтерського обліку даних може мати значний вплив на всі аспекти бізнесу. Детальніше розглянемо кожну з функцій і можливостей, які можуть бути включені в такий додаток:

– облік продажів: Додаток дозволить реєструвати всі продажі кав'ярні, включаючи інформацію про товари, кількість, ціни, знижки та сплату. Він також може автоматично відслідковувати запаси товарів і сповіщати про необхідність поповнення;

– контроль витрат: Додаток дозволить фіксувати всі витрати, пов'язані з діяльністю кав'ярні, такі як закупівля інгредієнтів, покупка обладнання, зарплата працівників тощо. Це допоможе здійснювати аналіз витрат та планувати бюджет;

– управління постачальниками: Додаток може включати базу даних постачальників, їх контактну інформацію, умови постачання та ціни. Це спростить процес замовлення товарів та взаємодії з постачальниками;

– фінансовий аналіз: Додаток може генерувати звіти та аналітичну інформацію про фінансовий стан кав'ярні, включаючи звіти про продажі, прибуток, витрати, рентабельність тощо. Це допоможе власникам та керівникам кав'ярні приймати обґрунтовані рішення та вдосконалювати стратегію розвитку;

– управління персоналом: Додаток може включати функції для реєстрації робочого часу, розрахунку зарплати та інших аспектів, пов'язаних з управлінням персоналом;

– замовлення інгредієнтів та постачальників: Додаток може мати систему для замовлення інгредієнтів, яка автоматично відслідковує запаси і попереджає про необхідність поповнення. Він також може враховувати умови постачання та ціни різних постачальників, щоб допомогти здійснювати оптимальний вибір та забезпечувати ефективне управління запасами;

– управління меню: Додаток може включати функціональність для управління меню кав'ярні. Власники та менеджери зможуть додавати нові товари, оновлювати ціни, визначати акції та знижки. Це дозволить швидко адаптуватися до змінних смаків та попиту клієнтів;

– реєстрація замовлень та платежів: Додаток може мати можливість реєструвати замовлення клієнтів та автоматично обробляти платежі. Це спростить процес обслуговування та забезпечить точність при розрахунках з клієнтами;

– аналіз продажів та звітність: Додаток може генерувати різноманітну звітність, що дозволить власникам та керівникам кав'ярень аналізувати продажі, визначати найпопулярніші товари, оцінювати ефективність рекламних акцій та вивчати тенденції споживання. Це допоможе приймати обґрунтовані рішення щодо асортименту та маркетингових стратегій;

– мобільний доступ: Додаток може бути розроблений як мобільний додаток, що дозволить власникам та персоналу кав'ярень отримувати доступ до даних та керувати бізнесом навіть поза офісом або кав'ярнею. Це забезпечить більшу гнучкість та ефективність в управлінні.

Реалізація такого додатку допоможе покращити ефективність та точність бухгалтерського обліку в кав'ярнях, зменшити ризики помилок та оптимізувати робочі процеси. В результаті це може призвести до зростання прибутків, зниження витрат та покращення задоволеності клієнтів.

Загалом, розробка додатку для автоматизації бухгалтерського обліку в кав'ярнях має потенціал значно полегшити роботу, збільшити ефективність та знизити витрати. Він допоможе власникам та керівникам кав'ярень зосередитися на стратегічних аспектах бізнесу, покращити якість обслуговування клієнтів і приносити більше прибутку.

Таким чином, було проведено детальний аналіз обраної предметної області, визначено і частково описано об’єкт розробки.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Після того як було проведено аналіз існуючого програмного забезпечення, можна скласти список вже існуючих рішень. Серед них можна виділити такі застосунки як:

– «Lavu»;

– «JoinPoster»;

– «Toast POS»;

Властивості всіх згаданих веб-додатків описані нижче з урахуванням їх характеристик, зовнішнього вигляду, переваг і недоліків, а також особистого досвіду та відгуків користувачів.

"Lavu" надає можливість автоматизувати багато рутинних процесів управління кав'ярнею, таких як замовлення інгредієнтів, облік запасів, обробка замовлень та платежів. Це дозволяє зменшити ручну працю, знизити ризик помилок і збільшити продуктивність. На рисунку 1.1 зображено головну сторінку сайту “Lavu”, а на рисунку 1.2 зовнішній вигляд застосунку.

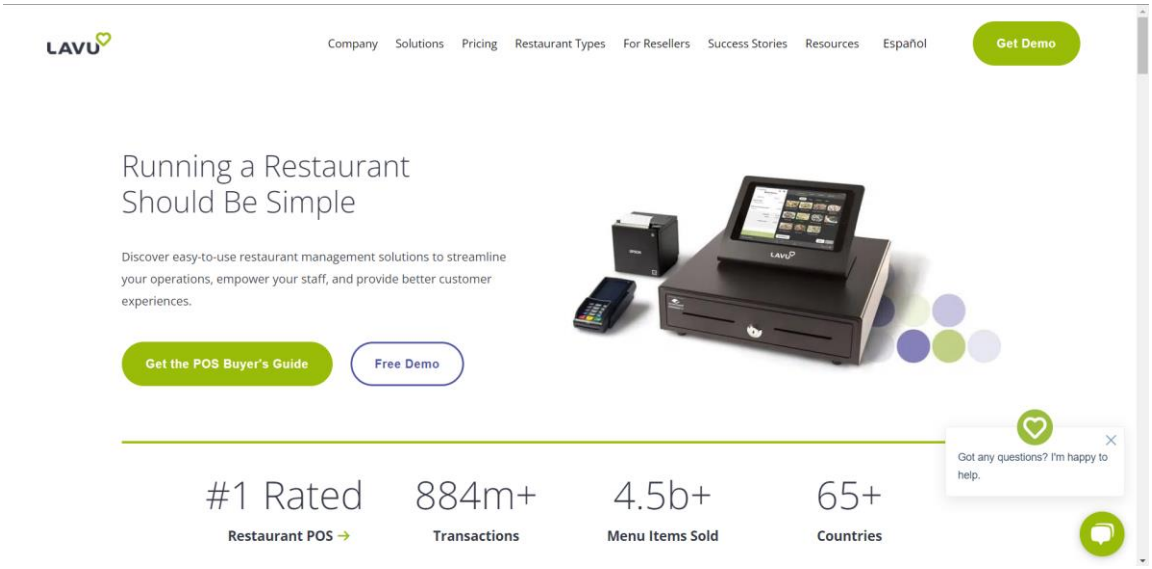


Рисунок 1.1 – Головна сторінка сайту “Lavu” – https://lavu.com

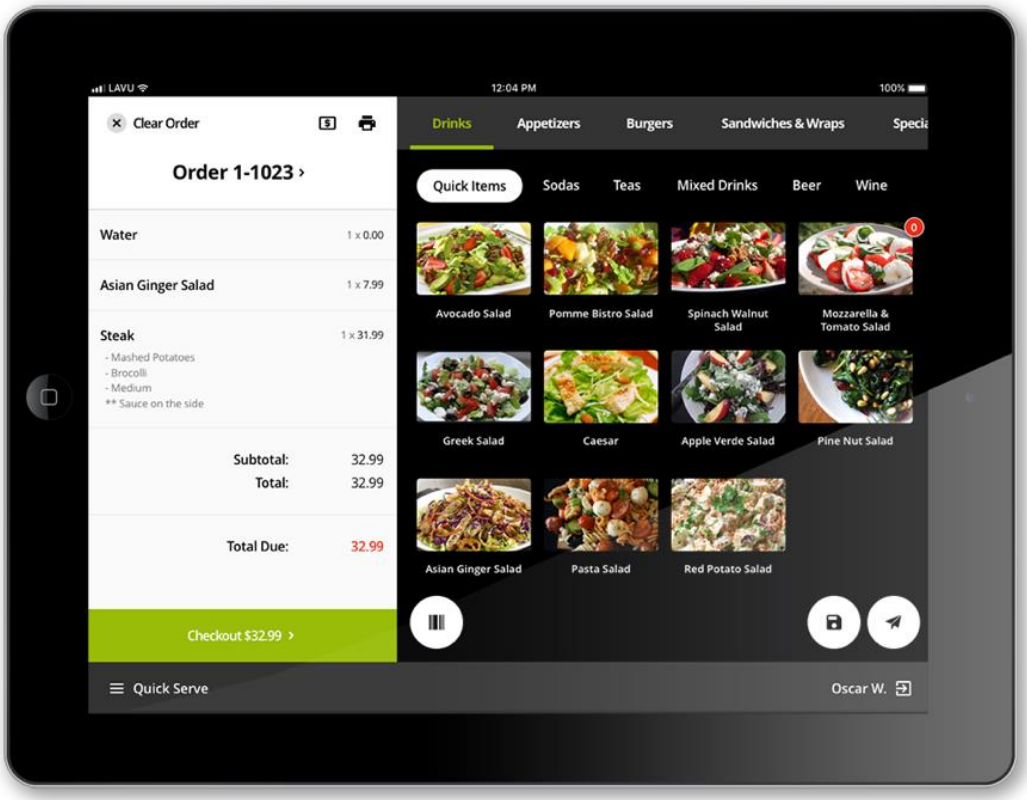


Рисунок 1.2 – Зовнішній вигляд додатку “ Lavu ” – https://lavu.com

Переваги Lavu:

– застосунок надає широкий спектр функцій, необхідних для ефективного управління кав'ярнею, включаючи замовлення, касові операції, управління клієнтами, столиками та персоналом;

– компанія пропонує мобільні додатки для замовлення та обслуговування клієнтів, що дозволяє персоналу зруч;о приймати замовлення та оплачувати рахунки безпосередньо за столиком.

– додаток має можливість інтеграції з іншими системами, такими як системи управління запасами, обліку праці, платіжні шлюзи та інші. Це дозволяє підприємствам забезпечити цілісну систему управління;

– Lavu надає розширені звіти та аналітику, які допомагають аналізувати продажі, витрати, популярність певних страв та інші ключові показники, що сприяє прийняттю дієвих управлінських рішень.

Недоліки Lavu:

– застосунок відноситься до преміум-класу рішень POS, і вартість використання може бути значною для деяких підприємств, особливо для менших кав'ярень з обмеженим бюджетом;

– у деяких випадках, налаштування може бути складним завданням, особливо для користувачів без попереднього досвіду з POS-системами;

– додаток потребує стабільного Інтернет-з'єднання для безперебійної роботи. Відсутність Інтернету може призвести до перебоїв у роботі та обслуговуванні клієнтів;

– користувачі часто повідомляють про проблеми з технічною підтримкою додатку, зокрема, про час відповіді та вирішення проблем.

Наступним розглянемо застосунок “JoinPoster”. На рисунку 1.3 зображено головну сторінку сайту, а на рисунку 1.4 зовнішній вигляд застосунку для Android.

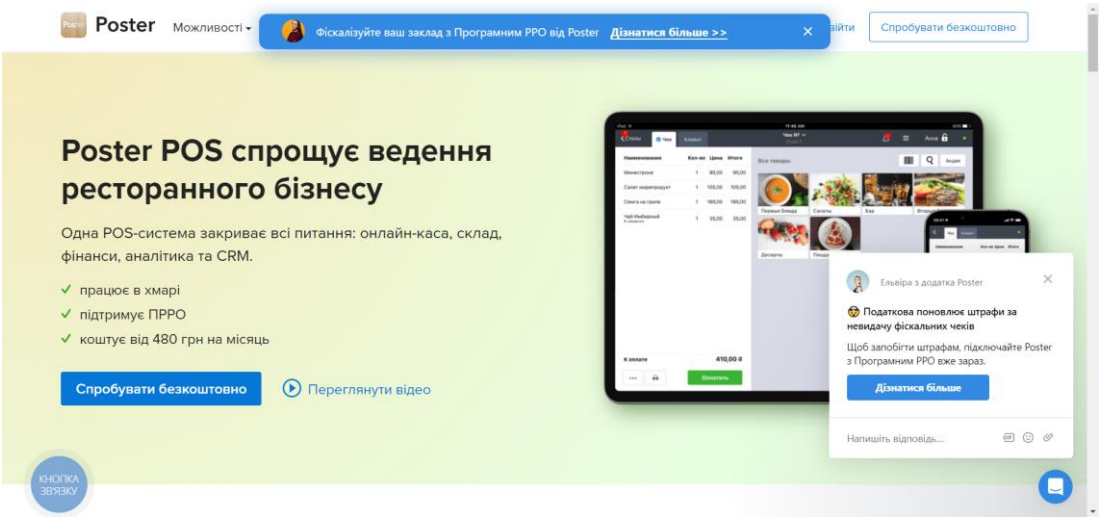


Рисунок 1.3 – Головна сторінка сайту “Joinposter” – https://joinposter.com/

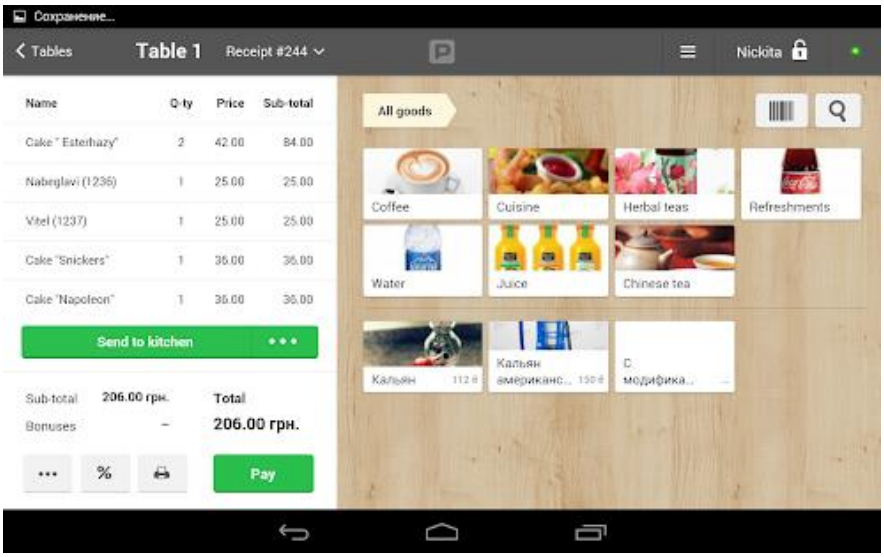


Рисунок 1.4 – Зовнішній вигляд додатку “Joinposter” - https://joinposter.com/

Переваги додатку "JoinPoster":

застосунок надає інтегровану платформу для автоматизації різних аспектів роботи кав'ярень, таких як замовлення, облік продажів, управління запасами, бухгалтерський облік, звіти тощо. Це дозволяє ефективно керувати всіма процесами з однієї централізованої платформи;

– застосунок має зрозумілий та інтуїтивний інтерфейс користувача, що спрощує навчання персоналу та полегшує роботу з програмою. Користувачі швидко зможуть оволодіти всіма функціями та здійснювати операції без зайвих зусиль;

– компанія пропонує мобільний додаток, який дозволяє керувати кав'ярнею з будь-якого місця та в будь-який час. Власники та керівники можуть отримувати доступ до важливої інформації та здійснювати керування підприємством навіть під час відсутності на місці;

– застосунок дозволяє автоматизувати багато рутинних задач, що дозволяє працювати більш ефективно і зосередитися на важливих аспектах бізнесу. Відстеження продажів, управління запасами та замовленнями, підготовка звітів - все це виконується автоматично, що допомагає зекономити час та зусилля;

Недоліки додатку "JoinPoster":

– плата за використання програми або підписка на певні функціональні можливості можуть бути значними для недосвідчених підприємців;

– для використання застосунку необхідний доступ до Інтернету. Це означає, що в разі відсутності Інтернет-з'єднання може виникнути проблема доступу до важливих даних та функціоналу програми. Крім того, повільне або нестабільне з'єднання може уповільнити роботу з програмою та створити нестабільність в процесах керування;

– хоча застосунок надає багато функцій і можливостей для автоматизації, він може бути обмежений у налаштуванні до конкретних потреб кожної кав'ярні. Якщо ви маєте специфічні вимоги або потребуєте індивідуального підходу до управління, "JoinPoster" може не задовольнити всі ваші вимоги.

Далі розглянемо застосунок “Toast POS”. На рисунку 1.5 зображено головну сторінку сайту, а на рисунку 1.6 зовнішній вигляд застосунку для Windows.

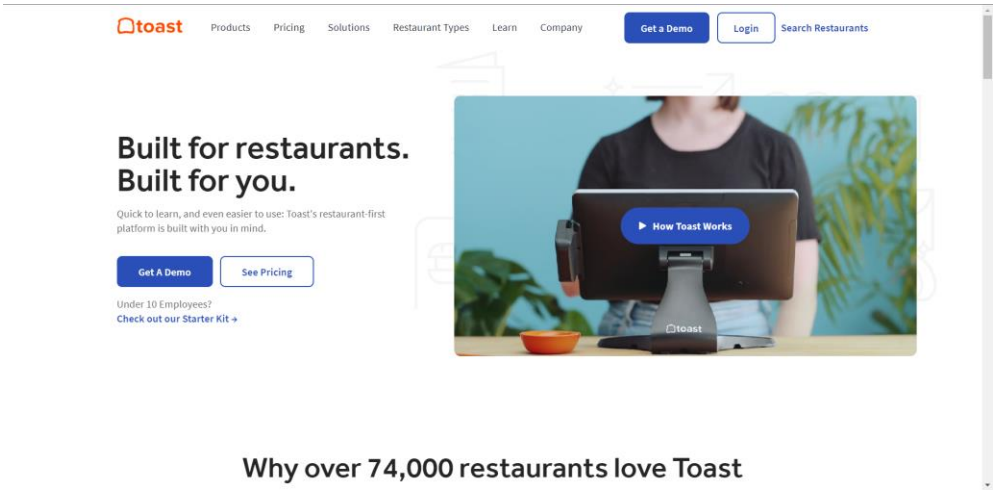


Рисунок 1.5 – Головна сторінка сайту “Toast POS” - https://pos.toasttab.com

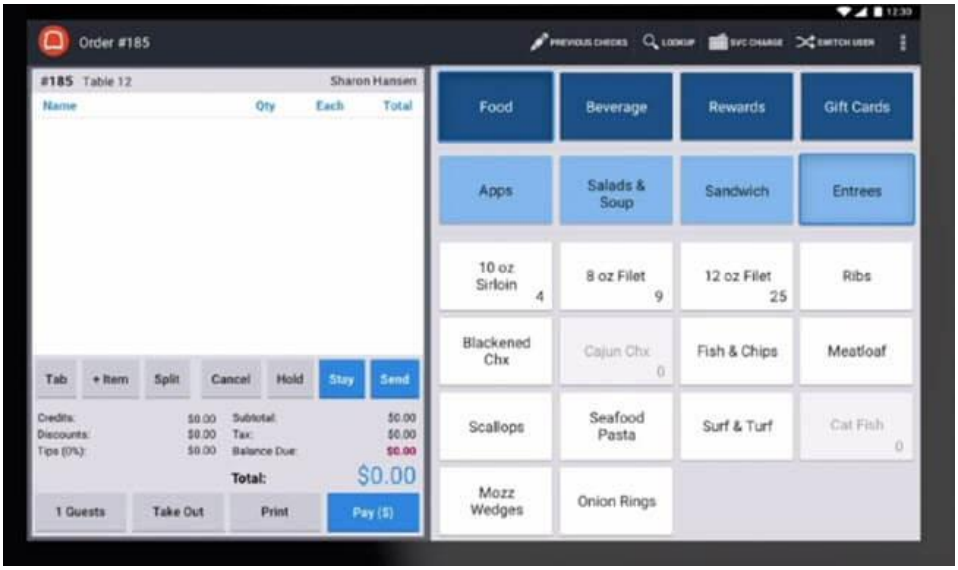


Рисунок 1.6 – Зовнішній вигляд додатку “Toast POS” - https://pos.toasttab.com

Переваги "Toast POS":

– має зручний і легкий у використанні інтерфейс, що спрощує навчання персоналу і швидку адаптацію до системи;

– надає широкий спектр функцій, що дозволяють автоматизувати замовлення, оплату, управління столиками, інвентарем та звітністю. Це сприяє покращенню ефективності роботи кав'ярні;

– має можливість інтеграції з іншими системами, такими як бухгалтерські програми, системи доставки, програми лояльності тощо. Це дозволяє злагоджено працювати з різними аспектами бізнесу;

– надає детальну звітність і аналітику про продажі, інвентар, покупців та інші аспекти бізнесу. Це дозволяє керівникам отримувати цінну інформацію для прийняття стратегічних рішень.

Недоліки "Toast POS":

– може бути дорогим варіантом, особливо для невеликих кав'ярень з обмеженим бюджетом. Вартість включає не лише встановлення програмного забезпечення, але і підписку на обслуговування та підтримку.

– працює на основі хмарної технології і вимагає стабільного Інтернет-з'єднання. Проблеми з Інтернетом можуть вплинути на доступність системи та її функціонал.

– деякі функції можуть бути більш корисними для певних типів кав'ярень або ресторанів. Наприклад, якщо у вас немає потреби в доставці або управлінні столиками, деякі функції можуть бути незастосовними.

У таблиці 1.1 зроблено порівняння застосунків описаних вище.

Таблиця 1.1 – Порівняльна таблиця застосунків

Характеристика	Lavu	JoinPoster	Toast POS
Платформи	iOS, Android	iOS, Android	iOS, Android
Ціна	Вартість пакетів починається від $69/місяць	Ціни варіюються в залежності від потреб бізнесу	Інформація про ціну надається виробником
Керування запасами	Так	Так	Так
Замовлення доставки	Так	Так	Так

Кінець таблиці 1.1

Характеристика	Lavu	JoinPoster	Toast POS
Спеціалізовані функції	- Керування баром та кухнею	- Резервування столиків та часовий графік персоналу	- Управління клієнтськими замовленнями та доставкою
Користувацький інтерфейс	Зручний та інтуїтивний	Сучасний та простий у використанні	Інтуїтивний та простий у використанні
Можливості розширення	Можливість інтеграції з багатьма сторонніми програмами та сервісами	Вибір з різних модулів та додаткових функцій	Гнучкі настройки знижок та акцій, а також інтеграція з іншими системами

В порівнянні трьох додатків - Toast POS, Lavu і Joinposter, які використовуються для автоматизації роботи в кав'ярнях, можна зробити наступні висновки:

Toast POS є інтуїтивним і простим у використанні додатком з повною автоматизацією процесів. Він пропонує розширену аналітику та можливість інтеграції з іншими системами. Однак, вартість Toast POS може бути високою, що може бути фактором для невеликих кав'ярень з обмеженим бюджетом.

Lavu є зручним і легким у використанні додатком з фокусом на автоматизацію замовлень та оплати. Він також пропонує звітність та аналітику продажів. Lavu має можливість інтеграції з іншими системами, але вартість його використання є середньою.

Joinposter також є інтуїтивним і простим у використанні додатком з автоматизацією замовлень та оплати. Він пропонує звітність та аналітику продажів, а також можливість інтеграції з іншими системами. Варто відзначити,

що Joinposter має різні плани ціноутворення, що може бути зручним для різних типів бізнесів.

При виборі додатку для автоматизації роботи в кав'ярні важливо враховувати специфіку свого бізнесу, бюджет та потреби. Кожен з цих додатків має свої переваги і недоліки, тому ретельне обстеження та порівняння функціоналу, вартості та підтримки можуть допомогти зробити правильний вибір для оптимальної автоматизації робочих процесів в кав'ярні.

Тому детальний аналіз існуючих аналогічних рішень та їх атрибутів допомагає виявити основні характеристики, функціональні та нефункціональні вимоги проекту, що розробляється.

1.3 Аналіз вимог до програмного забезпечення та розробка технічного завдання

В цьому розділі ми розглянемо процес аналізу вимог до програмного забезпечення та розробку технічного завдання. Аналіз вимог є важливим кроком у розробці програмного продукту, оскільки він допомагає зрозуміти бізнес-потреби, функціональні та нефункціональні вимоги, а також очікування користувачів.

Під час аналізу вимог проводиться вивчення бізнес-процесів, визначення потреб користувачів, аналіз конкурентного середовища та інші дії, що допомагають зрозуміти контекст та обсяг проекту. На основі зібраних вимог формулюється технічне завдання - документ, який визначає функціональні та нефункціональні вимоги до програмного забезпечення, архітектуру системи, технічні обмеження, графічний інтерфейс та інші аспекти проекту.

Застосунок для автоматизації роботи кав'ярень з використанням бухгалтерського обліку даних повинен включати такі функції:

– облік замовлень - можливість створювати та зберігати замовлення клієнтів, включаючи інформацію про страви, інгредієнти, ціни та дати замовлення;

– формування звітів - можливість формування звітів про замовлення, товари, фінансові операції тощо;

– управління співробітниками - можливість створювати та підтримувати профілі співробітників, контролювати їх доступ до системи, а також зберігати інформацію про їхні зарплати та графіки роботи;

– контроль залишків продуктів - можливість відстежувати продукти та контролювати їх кількість на складах, щоб уникнути кулінарного дефіциту;

– фінансовий облік - можливість створювати та зберігати інформацію про всі фінансові операції, що проводяться в кав'ярні, включаючи оплату замовлень та заробітну плату співробітників;

– аналіз продажів - програма повинна збирати та обробляти дані про продажі, замовлення та витрати. Таким чином, власники кафе матимуть можливість отримувати аналітичні дані про продажі та прибутки, що дозволить їм приймати кращі рішення щодо розвитку бізнесу.

– розширена панель адміністратора - менеджери кафе повинні мати доступ до розширеної панелі адміністратора, де вони можуть керувати замовленнями, меню та іншими функціями програми.

Для визначення вимог до програмного забезпечення було використано уніфіковану мову моделювання (UML), стандартизовану мову моделювання, призначену для представлення, визначення, візуалізації та документування об'єктно-орієнтованих моделей програмного забезпечення.

Діаграма прецедентів (або діаграма варіантів використання) — в UML, діаграма, на якій зображено відношення між акторами та прецедентами в системі.

Діаграма прецедентів є графом, що складається з множини акторів, прецедентів (варіантів використання) обмежених межею системи (прямокутник), асоціацій між акторами та прецедентами, відношень серед прецедентів, та

відношень узагальнення між акторами. Діаграми прецедентів відображають елементи моделі варіантів використання.

Суть діаграми прецедентів полягає в тому, що проєктована система подається у вигляді множини сутностей чи акторів, що взаємодіють із системою за допомогою так званих варіантів використання. Варіант використання (англ. use case) використовують для описання послуг, які система надає актору. Іншими словами, кожен варіант використання визначає деякий набір дій, який виконує система під час діалогу з актором. При цьому нічого не говориться про те, яким чином буде реалізовано взаємодію акторів із системою. У таблиці 1.2

Таблиця 1.2 – Опис варіантів використання програмної системи

Актор	Найменування ВВ	Опис ВВ
Працівник	Вхід/Вихід	Працівник може авторизуватись в системі використовуючи логін і пароль, які їм створив власник
	Формування замовлення	Працівник може формувати замовлення
	Перегляд інформації про товари, пошук, сортування по категоріях	Працівник може переглядати наявні товари у списку, шукати їх по назві, сортувати по категоріях. Переглядати детальну інформацію про них (назву, картинку, опис, ціну)
Власник	Вхід/Вихід	Власник може авторизуватись в системі використовуючи логін і пароль
	Обробка інформації про працівників працівників	Власник може керувати працівниками(редагувати, додавати, видаляти)

Кінець таблиці 1.2

Актор	Найменування ВВ	Опис ВВ
Власник	Обробка інформації про товари	Власник може керувати даними товарів (редагувати, додавати, видаляти)
	Перегляд інформації про всі замовлення	Власник може переглядати інформацію про всі замовлення
	Обробка інформації про категорії	Власник може керувати даними категорій (редагувати, додавати, видаляти)

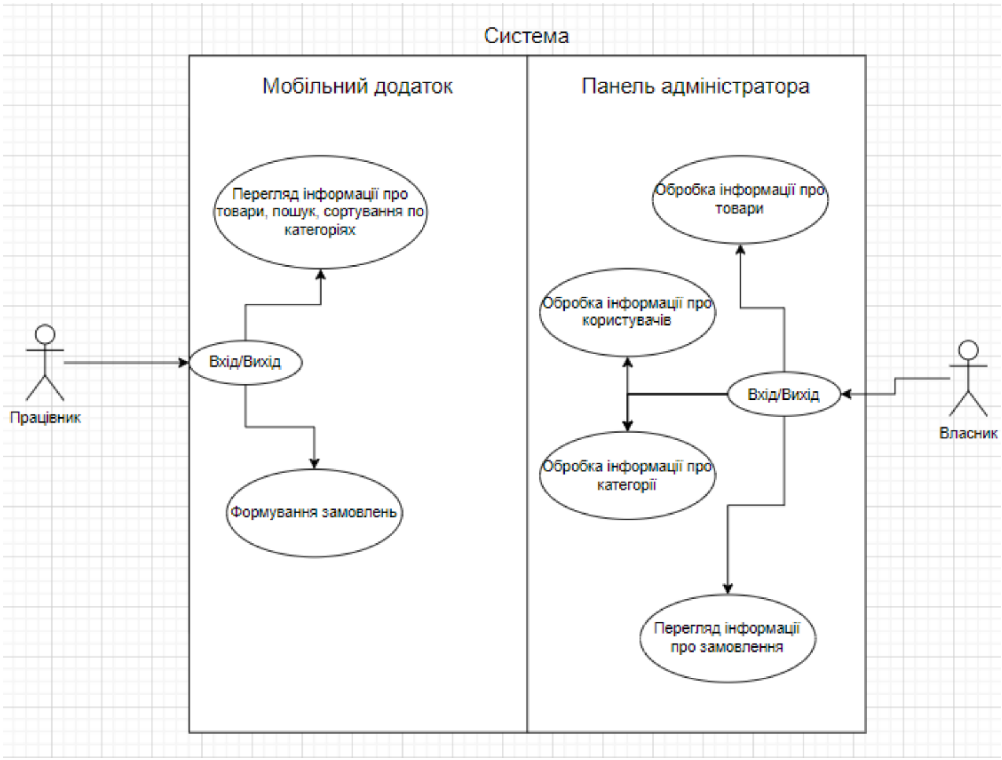


Рисунок 1.7 – Головні варіанти використання

1.4 Висновки. Постановка задачі.

У даному розділі розглядаються особливості предметної області, виявлені проблеми та шляхи їх рішення. Були виявлені переваги та недоліки наявних

програм. Було складено перелік функціональних і нефункціональних вимог до проекту, який зараз розробляється.

Виконавши аналіз усіх вимог та предметної області кваліфікаційної роботи ми можемо створити список завдань які будуть стояти перед нами під час виконання кваліфікаційної роботи.

Застосунок для автоматизації роботи кав’ярень повинен виконувати такі задачі:

– надати змогу працівникам отримувати інформацію про наявні товари з розподілом їх на категорії;

– за допомогою застосунку працівник може формувати замовлення з товарів, що є на складі;

– дати можливість власнику створювати працівників і змінювати інформацію про них;

– власник має можливість обробляти дані про товари та керувати складом;

– за допомогою адмін панелі власник має мати змогу переглядати дані про замовлення, які були створені працівниками.

Висновки:

Провівши детальний аналіз предметної області зі сторення застосунку для автоматизації роботи кав’ярень можна прийти до висновку що ця сфера має велику перспективу розвитку адже є чимало кав’ярень, які потребують схожих застосунків. Ринок має великі перспективи для подальшого розвитку. Проведення аналізу існуючих рішень дозволило виявити переваги та недоліки сайтів аналогів. З проведеного аналізу можна прийти до висновків що аналогів є багато, але вони не гнучкі, доволі часто важкі для навчання працівників і мають доволі високі ціни на підписки.. Була проведена розробка технічного завдання, що дозволило визначити основний функціонал для застосунку та розробити функціонал який в майбутньому може бути використаний для його розширення. В загальному можна прийти до висновку, що застосунок має потенціал бути успішним проектом.

2 ПРОЕКТУВАННЯ ЗАСТОСУНКУ

2.1 Аналіз та вибір архітектури застосунку

Клієнт-серверна архітектура є однією з найпоширеніших архітектурних моделей у розробці програмного забезпечення. Вона базується на розподіленому обміні даними між двома видами компонентів - клієнтами та серверами. У цій моделі клієнти і сервери виконують різні функції та взаємодіють між собою за допомогою мережі.

Основна ідея клієнт-серверної архітектури полягає в тому, що клієнти використовують ресурси і послуги, які надаються серверами. Клієнти - це програми або пристрої, що взаємодіють з користувачем і відправляють запити до серверів для отримання інформації або виконання певних операцій. Сервери - це компоненти, які надають ці ресурси і обробляють запити клієнтів.

Основні переваги клієнт-серверної архітектури включають:

– розподіл ролей: Клієнти і сервери виконують різні функції, що дозволяє краще організувати систему і розподілити завдання між компонентами;

– масштабованість: Архітектура дозволяє масштабувати систему шляхом додавання нових серверів або розподілу навантаження між серверами, що дозволяє підтримувати високу продуктивність навіть при збільшенні обсягу даних або кількості користувачів;

– централізований контроль: Сервери можуть надавати централізований контроль над доступом до ресурсів, аутентифікацією та авторизацією користувачів, що забезпечує безпеку і керованість системи;

– простота супроводу: Розподілені функції між клієнтами і серверами дозволяють здійснювати зміни в окремих компонентах системи без впливу на інші частини.

Незважаючи на переваги, клієнт-серверна архітектура також має свої недоліки:

– робота клієнтів залежить від доступності та продуктивності серверів. Якщо сервери перестануть працювати або будуть перевантажені, це може вплинути на доступність та продуктивність клієнтських додатків;

– використання мережі для взаємодії між клієнтами і серверами означає, що недолік у мережі може призвести до зниження продуктивності або недоступності системи;

– потреба у великому обсязі даних: Обмін даними між клієнтами і серверами може вимагати значних ресурсів мережі та обробки, особливо при обробці великого обсягу даних;

– архітектура може створювати вразливості з точки зору безпеки, оскільки сервери мають доступ до цінної інформації, що вимагає відповідних заходів безпеки для захисту даних.

Клієнт-серверна архітектура є важливим інструментом для розробки розподілених програмних систем. Вона дозволяє ефективно організувати взаємодію між компонентами системи та забезпечує гнучкість та масштабованість.

Також для адмін панелі можна використати архітектуру клієнтського інтерфейсу - це організаційна структура, яка визначає, як компоненти користувацького інтерфейсу взаємодіють між собою і з користувачем. Її основна мета полягає в розділенні логіки відображення і взаємодії з користувачем від логіки обробки даних на сервері. Це дозволяє забезпечити більшу гнучкість, підтримку повторного використання і покращену керованість в процесі розробки клієнтського програмного забезпечення.

Основні компоненти архітектури клієнтського інтерфейсу включають:

– модель даних - це представлення даних, з якими взаємодіє користувач. Модель даних може бути створена як об'єктна модель або структура даних, яка відображає основні сутності і відношення між ними. Вона містить логіку для отримання і збереження даних на сервері;

– представлення - відповідають за візуальне відображення даних і інтерфейс користувача. Вони можуть бути створені з використанням різних технологій, таких як HTML, CSS і JavaScript. Представлення приймають дані з моделі даних і генерують відповідний вигляд для користувача;

– контролери - відповідають за обробку подій користувача і взаємодію з моделлю даних. Вони приймають вхідні дані від користувача через представлення і здійснюють відповідні дії, такі як оновлення моделі даних або виклик певних функцій на сервері. Контролери забезпечують зв'язок між представленням і моделлю даних;

– маршрутизатор - визначає, які дії повинні бути виконані для конкретного URL або користувацького взаємодії. Він визначає маршрути і відповідні контролери, які мають бути викликані при зміні URL або подіях;

– сервіси - представляють загальні функціональність, яка може бути використана в різних компонентах інтерфейсу. Вони забезпечують реалізацію логіки, яка не пов'язана безпосередньо з відображенням або моделлю даних, валідацію даних, взаємодію з зовнішніми сервісами або обробку подій.

Ось деякі переваги такої архітектури:

– компоненти можуть бути розроблені незалежно від серверної частини, що спрощує процес розробки. Кожен компонент може бути відповідальний за свою функціональність і може бути розроблений окремо.

– архітектура дозволяє виконувати багато обчислень та маніпуляцій без необхідності звертатися до сервера. Це дозволяє створювати більш швидкодіючі та реагуючі додатки, що покращує враження користувача.

– клієнтська архітектура може дозволити працювати додатку в офлайн-режимі, коли немає з'єднання з сервером. Це може бути корисним для застосунків, які потребують доступу до даних безпосередньо на пристрої користувача.

– вона може виконувати багато обробки та обчислень на самому пристрої користувача, зменшуючи навантаження на сервер та покращуючи продуктивність системи.

Недоліки архітектури клієнтського інтерфейсу:

– оскільки архітектура може вимагати зберігання даних на пристрої користувача, це може призвести до збільшення обсягу даних, особливо при наявності багатофункціональних додатків зі складним інтерфейсом.

– через те, що дані можуть бути збережені на пристрої користувача, це створює додаткові проблеми з безпекою і захистом даних. Важливо забезпечити відповідні механізми шифрування та захисту даних.

– архітектура може бути менш безпечною, оскільки код додатку виконується на пристрої користувача і може бути підвержений варантності. Недостатньо захищений код може стати точкою входу для атак і зловживань.

– різні пристрої та операційні системи можуть мати різні обмеження та підтримку для клієнтської архітектури. Необхідно забезпечити, щоб додаток працював на різних платформах та пристроях.

Використання архітектури клієнтського інтерфейсу дозволяє розбити складність програмного забезпечення на окремі компоненти, які можуть бути розроблені, тестовані і зберігати окремо. Це сприяє збереженню коду більш читабельним і керованим, спрощує розширення та підтримку додатків, а також забезпечує більшу гнучкість у відображенні даних та взаємодії з користувачем.

Таким чином, для реалізації завдання квалафікаційної роботи, було обрано клієнт-серверну архітектуру для мобільного додатку та серверу та архітектуру клієнтського інтерфейсу для адмін панелі.

2.2 Опис структури даних та моделі бази даних

Більшість застосунків потребують збереження даних для цього в клієнт-серверній архітектурі зазвичай використовують бази даних.

База даних (БД) - це організована колекція даних, яка зберігається та керується за допомогою спеціального програмного забезпечення. Бази даних використовуються для зберігання великого обсягу структурованих даних, які можуть бути доступні та оброблювані в ефективний спосіб. Вони є ключовим елементом для багатьох додатків та систем, що потребують постійного доступу до інформації.

База даних складається з:

– модель даних - це спосіб організації даних у базі даних. Модель даних визначає, як дані будуть представлені, зберігатися та взаємодіяти один з одним. Існують різні моделі даних, такі як ієрархічна, мережева, реляційна, об'єктно-орієнтована та інші. Кожна модель має свої переваги та використовується відповідно до вимог конкретного проекту;

– таблиці - є основними структурами для збереження даних у реляційних базах даних. Вони представляють собою двовимірні матриці, де кожен рядок відповідає запису, а кожний стовпчик - полю даних. Кожен запис має унікальний ідентифікатор (ключ), що дозволяє однозначно ідентифікувати його;

– запити - використовуються для отримання, оновлення або видалення даних з бази даних. Запити можуть бути простими, які повертають всі записи з таблиці, або складними, які об'єднують дані з різних таблиць за певними умовами. Запити дозволяють ефективно взаємодіяти з даними та отримувати потрібну інформацію;

– індекси - використовуються для прискорення пошуку та доступу до даних у базі даних. Вони створюються для певних полів або комбінацій полів у таблицях, що дозволяє швидше здійснювати пошук за цими полями. Індекси покращують продуктивність запитів, зменшуючи кількість даних, які потрібно переглядати;

– транзакції - використовуються для забезпечення атомарності та консистентності даних. Вони представляють собою послідовність дій, які виконуються як єдине ціле. Якщо одна дія транзакції не вдалася, то всі попередні

дії скасовуються (rollback), а якщо всі дії успішні, то вони фіксуються (commit) та стають постійними;

– нормалізація - процес організації даних у базі даних для досягнення ефективності, цілісності та уникнення дублювання даних. Це включає розбиття даних на окремі таблиці та встановлення взаємозв'язків між ними. Нормалізація допомагає забезпечити консистентність та зменшити ризик пошкодження даних.

Бази даних використовуються в різних сферах, включаючи бізнес, науку, освіту та інформаційні системи. Вони дозволяють ефективно зберігати та керувати великими обсягами даних, що є важливим для прийняття обґрунтованих рішень та підтримки різних бізнес-процесів. Розуміння принципів баз даних допомагає розробникам та адміністраторам впоратися з комплексними завданнями, пов'язаними з організацією та управлінням даними.

Складовою частиною багатьох баз даних є система керування базами даних (СКБД), яка виконує функції зберігання, керування та доступу до даних. Нижче наведено опис деяких популярних СКБД:

MySQL: MySQL є однією з найпопулярніших відкритих реляційних СКБД. Вона володіє широким спектром можливостей та підтримує мову SQL для роботи з даними. MySQL відома своєю швидкодією та надійністю, що робить її популярним вибором для веб-додатків та багатьох інших сценаріїв.

Oracle: Oracle є однією з найбільш відомих комерційних реляційних СКБД. Вона пропонує високу продуктивність, масштабованість та надійність, і широко використовується в корпоративному середовищі. Oracle підтримує багато продвинутих функцій, таких як реплікація, кластеризація та безпека даних.

Microsoft SQL Server: Microsoft SQL Server є комерційною реляційною СКБД, розробленою компанією Microsoft. Вона надає широкий набір функцій, включаючи багатопотоковість, високу масштабованість та інтеграцію з іншими продуктами Microsoft. SQL Server підтримує розширення SQL Server Integration Services (SSIS) для екстракції, трансформації та завантаження даних.

PostgreSQL: PostgreSQL є відкритою реляційною СКБД з акцентом на розширені функції та дотримання стандартів SQL. Вона володіє потужними

можливостями, такими як розширені типи даних, відображення та тригери. PostgreSQL також підтримує реплікацію, високу доступність та географічну реплікацію для забезпечення надійності та шкальованості.

MongoDB: MongoDB є документ-орієнтованою СКБД, яка зберігає дані у вигляді документів у форматі JSON. Вона відома своєю гнучкістю та швидкодією. MongoDB дозволяє легко масштабувати бази даних та здійснювати горизонтальне розширення шляхом розподілення даних на кластери.

Redis: Redis є ключ-значенням СКБД, що зберігає дані у вигляді пар ключ-значення. Вона володіє дуже високою швидкодією та масштабованістю. Redis широко використовується для кешування даних, сесійного сховища та черг повідомлень.

SQLite: SQLite є вбудованою СКБД, яка надає локальне зберігання даних у файловій системі. Вона легка, компактна та не вимагає окремого сервера. SQLite використовується в багатьох мобільних додатках та вбудованих системах, де потрібна локальна база даних.

Це лише деякі з популярних СКБД, які використовуються в індустрії. Кожна з них має свої переваги, недоліки та використовується для різних сценаріїв. Вибір певної СКБД залежить від ваших потреб, масштабу проекту та інших факторів.

Для розробки даного застосунку буде використана СКБД MySQL, адже вона повністю підходить під вимоги даного завдання, є простою і не має зайвих елементів, які могли б зменшувати швидкість роботи серверу.

Беручи за основу створену раніше діаграму варіантів використання (див. рисунок 1.7), було визначено дані, які будемо зберігати в базу даних, а саме дані працівників, продукти, категорії і замовлення. Таким чином було створено таблиці даних «Category», «Order», «Employee» та «Product». Також для зв’язку продуктів і замовлень було створено проміжну таблицю «OrderProduct». Схема спроектованої бази даних зображено на рисунку 2.1

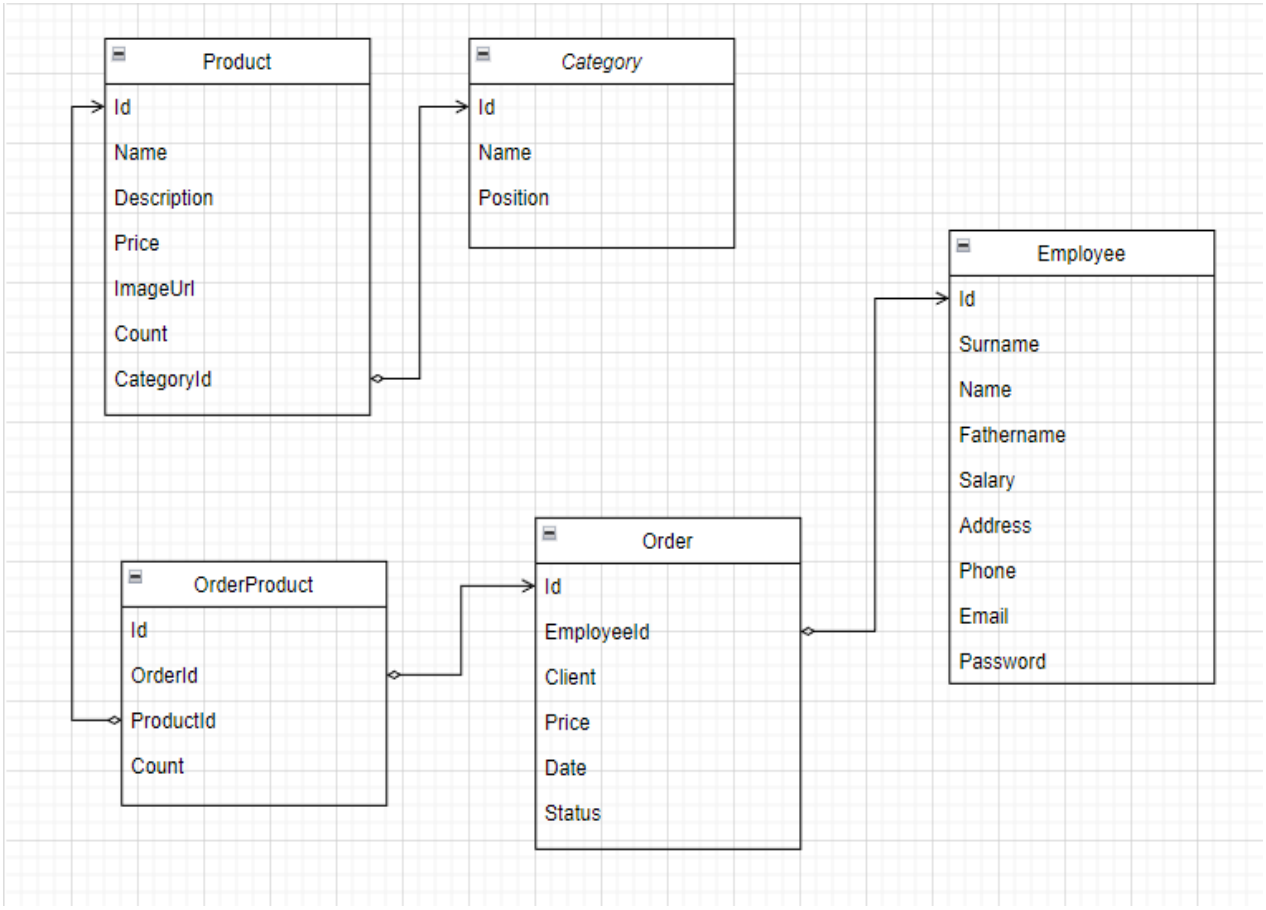


Рисунок 2.1 – Спроектована база даних

Перелік таблиць, що будуть використані:

Category – список категорій продуктів, що існують в даній системі. Має наступні поля:

– Id – унікальний ідентифікатор категорії;

– Name – назва;

– Position – номер в списку для сортування.

Product – список продуктів. Має наступні поля:

– Id – унікальний ідентифікатор продукту;

– Name – назва;

– Description – опис.

– Price – ціна за штуку;

– ImageUrl – посилання на картинку;

– Count – кількість на складі;

– CategoryId – ідентифікатор категорії.

Order – список замовлень створених працівниками. Має наступні поля:

– Id – унікальний ідентифікатор замовлення;

– EmployeeId – ідентифікатор працівника;

– Client – дані про клієнта.

– Price – загальна ціна замовлення;

– Date – дата останньої зміни статусу;

– Status – статус замовлення (створене, в обробці, завершене, повернення коштів).

Employee – список працівників кав’ярні. Має наступні поля:

– Id – унікальний ідентифікатор користувача;

– Surname – прізвище;

– Name – ім’я.

– Fathername – по батькові;

– Salary – ставка (заробітня плата);

– Address – адреса працівника

– Phone – номер телефону;

– Email – пошта працівника (для авторизації)

– Password - хеш (закодована послідовність символів) паролю працівника, який використовується для входу в обліковий запис;

OrderProduct – таблиця для зв’язку продуктів і замовлень. Має наступні поля:

– Id – унікальний ідентифікатор таблиці;

– OrderId – ідентифікатор замовлення;

– ProductId – ідентифікатор товару;

– Count – кількість товарів.

Таким чином, створюється логічна модель бази даних, на основі якої можна проектувати фізичну модель.

2.3 Проектування серверної частини застосунку

Серверна частина в клієнт-серверній архітектурі відіграє важливу роль у забезпеченні функціонування системи та наданні послуг клієнтам. Вона відповідає за обробку запитів, збереження та управління даними, забезпечення безпеки та здійснення комунікації з клієнтами.

Вона складається з серверних програм, які запущені на сервері та виконують різноманітні завдання. Ці програми можуть бути написані на різних мовах програмування та виконувати функції, такі як обробка запитів, маршрутизація, бізнес-логіка та доступ до бази даних.

Також вона забезпечує збереження та управління даними, які використовуються у системі. Вона може використовувати бази даних для збереження інформації та забезпечення доступу до неї. Управління даними включає операції, такі як створення, читання, оновлення та видалення даних.

Серверна частина відповідає за забезпечення безпеки системи. Це включає автентифікацію та авторизацію користувачів, захист доступу до конфіденційної інформації, контроль прав доступу та захист від зловживань.

Вона забезпечує комунікацію з клієнтами. Це може включати обробку запитів клієнтів, передачу даних між клієнтами та сервером за допомогою різних протоколів, таких як HTTP, WebSocket тощо. Крім того, сервер може надавати різні API (Application Programming Interface) для взаємодії з клієнтами та іншими сервісами.

Також сервр повинен бути здатний масштабуватися для обробки великої кількості запитів від клієнтів. Це може включати використання розподіленої архітектури, навантажувального балансування, кешування та інших методів для забезпечення високої продуктивності та доступності системи.

Узагальнюючи, серверна частина в клієнт-серверній архітектурі відповідає за обробку запитів, збереження даних, безпеку, комунікацію та

масштабованість. Вона грає ключову роль у функціонуванні системи та забезпеченні задоволення потреб клієнтів.

Щоб створювати запити та керувати ними, необхідно використовувати інтерфейс прикладного програмування RESTAPI.

REST API (Representational State Transfer Application Programming Interface) - це архітектурний стиль та набір правил та принципів, які використовуються для створення веб-сервісів, що дозволяють взаємодіяти з іншими програмами за допомогою HTTP-протоколу. REST API є одним з найпоширеніших способів взаємодії між клієнтськими додатками та серверами.

REST API використовує розподілену архітектуру, де клієнтські додатки та сервери взаємодіють незалежно один від одного. Це забезпечує розділення обов'язків та підвищує масштабованість системи.

Він передає становий протокол, що означає, що кожен запит клієнта до сервера містить всю необхідну інформацію для обробки запиту, і сервер не зберігає жодної інформації про стан клієнта. Це забезпечує простоту та незалежність між запитами.

REST API використовує різні HTTP-методи, такі як GET, POST, PUT та DELETE, для виконання різних операцій над ресурсами на сервері. Наприклад, GET використовується для отримання ресурсу, POST - для створення нового ресурсу, PUT - для оновлення ресурсу, DELETE - для видалення ресурсу.

Він підтримує різні формати даних для передачі інформації між клієнтом та сервером, такі як JSON (найпоширеніший), XML або HTML. Клієнти можуть вказати бажаний формат даних у заголовках запиту.

REST API може використовувати механізми кешування, щоб зберегти копію відповіді сервера на стороні клієнта. Це дозволяє зменшити час відповіді та обтяження сервера при повторних запитах.

REST API є широко використовуваним стандартом для розробки веб-сервісів і додатків, оскільки він дозволяє легко взаємодіяти з сервером, передавати дані та виконувати операції над ресурсами. Він забезпечує гнучкість, розширюваність та простоту в розробці.

Отже для розробки серверної частини додатку було описано архітектуру і обрано REST API, яка буде зв’язуватись з мобільним додатком.

2.4 Проектування інтерфейсу мобільного застосунку

Для того, щоб створити дизайн проекту було вирішено використати метод прототипізації. Прототипізація – це етап розробки ПЗ, на якому створюється макет або прототип програмного засобу, що дає змогу візуалізувати інтерфейс користувача на ранніх етапах розробки, для того щоб продемонструвати майбутнє ПЗ замовнику. Прототипування дає змогу уникнути більшості ризиків на етапі проектування.

Спочатку необхідно описати сторінку, на якій буде знаходитись користувач після першого входу в додаток. Це сторінка «авторизації». Вона повинна містити в собі поля пошти, паролю і кнопку логіну. На рисунку 2.2 зображений прототип сторінки авторизації

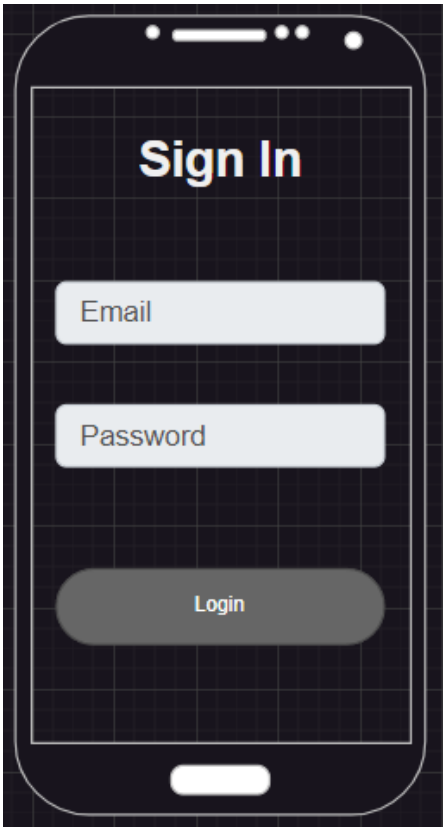


Рисунок 2.2 – Прототип сторінки авторизації

Після авторизації працівник одразу потрапляє на сторінку вибору товару. На ній повинні знаходитись поле для пошуку товару, категорії і список товарів. Прототип даної сторінки зображено на рисунку 2.3

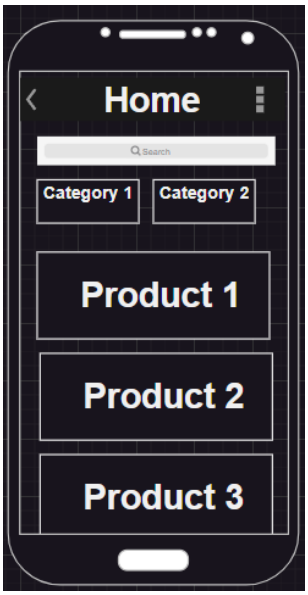


Рисунок 2.3 – Прототип сторінки товарів

Після вибору товару працівник потрапляє на його детальну сторінку, де може переглянути назву, опис, картинку, ціну, вибрати кількість товару і додати його в кошик. Прототип детальної сторінки товару зображено на рисунку 2.4.

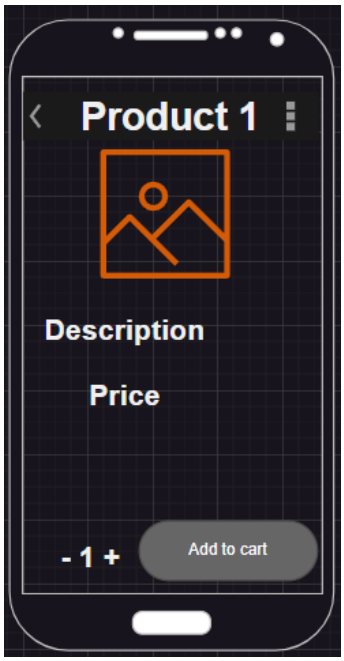


Рисунок 2.4 – Прототип детальної сторінки товару

Коли працівник додасть товар у кошик він зможе перейти на сторінку кошику, де буде список всіх товарів, там він зможе змінити їх кількість, видалити товар або оформити замовлення. Прототип кошику зображено на рисунку 2.5.

Рисунок 2.5 – Прототип сторінки кошику.

Отже, серед основних сторінок мобільного додатку працівника повинні бути сторінка авторизації, головна (список товарів), детальна сторінка товару і сторінка кошику.

2.5 Аналіз та вибір технологій і методів реалізації веб-додатка

Сучасні застосунки вимагають сучасних методів розробки програмного забезпечення. Основною мовою програмування для розробки обрано об’єктно-

орієнтовану мову програмування JavaScript. Ця мова розробки програмного забезпечення є найпоширенішою у світі. Код JavaScript є майже на кожній платформі з відкритим кодом. Усі веб-сайти створюються з використанням кількох технологій. Це включає фреймворки, бібліотеки, плагіни, шаблони тощо.

Для вибору технологій потрібно зобразити, як буде працювати система, для цього можна використати діаграму розгортання.

Діаграма розгортання є одним з видів діаграм UML (Unified Modeling Language) і використовується для моделювання архітектури системи та фізичного розміщення її компонентів. Вона надає графічне відображення фізичної конфігурації системи, включаючи апаратне та програмне забезпечення, мережеві ресурси, сервери, пристрої та їх зв'язки.

Основна мета діаграми розгортання - це показати, як компоненти системи фізично розташовані та взаємодіють один з одним. Вона допомагає зрозуміти, як система буде розгортатися на реальних пристроях та інфраструктурі, і візуалізує взаємозв'язки між компонентами.

На діаграмі розгортання використовуються наступні основні елементи:

– вузол:Представляє фізичний або віртуальний об'єкт, як-от сервер, комп'ютер або пристрій. Вузли вказують, де розміщені компоненти системи;

– компонент: Представляє програмний модуль, який виконує певну функцію. Це може бути програмне забезпечення, додаток, модуль або сервіс;

– взаємозв'язок: Відображає зв'язок між компонентами або між компонентами та вузлами. Наприклад, це може бути мережеве з'єднання, інтерфейс або протокол комунікації;

– артефакт: Представляє фізичний або логічний об'єкт, який може бути розгорнутий на вузлі. Це може бути програмне забезпечення, конфігураційні файли, бази даних тощо.

Діаграма розгортання дозволяє визначити структуру та конфігурацію системи, відобразити фізичну організацію компонентів та їх зв'язки. Вона допомагає розробникам, архітекторам та інженерам отримати уявлення про розгортання системи та забезпечити його ефективність та масштабованість.

На рисунку 2.6 зображена діаграма розгортання для даного застосунку.

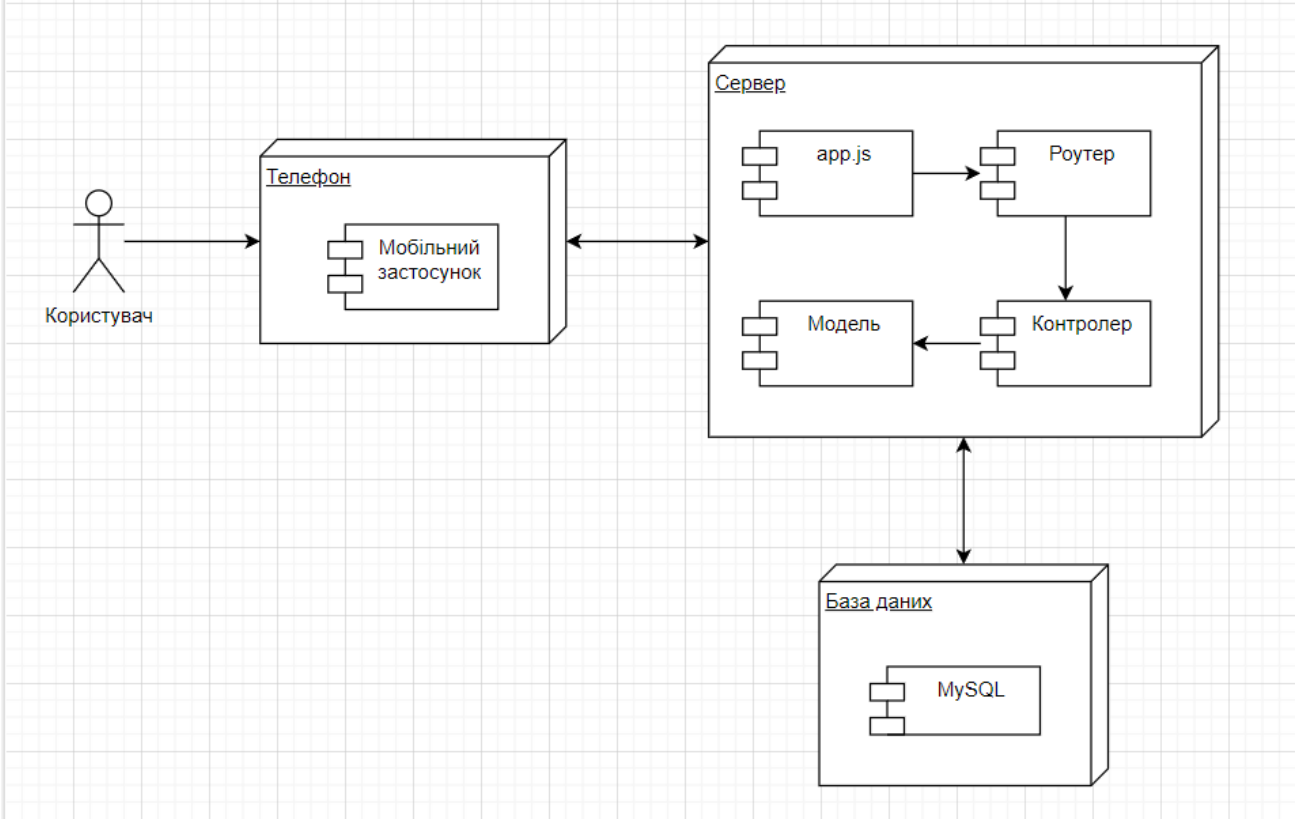


Рисунок 2.6 – діаграма розгортання

Для розробки даного застосунку було обрано стек, що складається з таких компонентів MySQL (СКБД), NodeJS (двигун для серверу), Express (фреймворк для маршрутизації серверної частини), React Native (фреймворк для розробки мобільного застосунку), AdminJS (бібліотека для створення адмін панелі)

Розглянемо всі компоненти обраного стеку технологій.

1) MySQL - це одна з найпопулярніших систем керування базами даних (СКБД), яка використовується для зберігання та управління структурованою інформацією. Вона є відкритою системою з вільно розповсюджуваним вихідним кодом та широко використовується як у комерційних, так і відкритих проектах.

MySQL базується на реляційній моделі даних, де дані організовані у вигляді таблиць з рядками та стовпцями. Це дозволяє встановлювати зв'язки між таблицями та здійснювати складні запити для отримання необхідної інформації.

Він використовує структуровану мову запитів SQL (Structured Query Language) для взаємодії з базою даних. SQL дозволяє створювати, модифікувати

та отримувати дані з бази даних, виконувати операції об'єднання, фільтрації, сортування та багато інших.

MySQL надає різні механізми для забезпечення безпеки даних, такі як автентифікація користувачів, ролева модель доступу, шифрування даних та інші. Це дозволяє контролювати доступ до бази даних та забезпечувати конфіденційність та цілісність інформації.

Він має багатий набір функціональності, такий як вбудовані функції обробки рядків та чисел, агрегатні функції, підтримку геоданих, текстовий пошук, резервне копіювання та відновлення даних і багато іншого.

MySQL є популярним вибором для розробників і досить простий у використанні. Він підтримується великою спільнотою розробників, що забезпечує наявність багато документації, підручників та підтримки.

2) Node.js є відкритою, подією-орієнтованою платформою для розробки серверних додатків на мові JavaScript. Вона побудована на двигуні V8, який також використовується в браузері Google Chrome. Однак, в випадку Node.js, V8 виконується на сервері, що дозволяє використовувати JavaScript для розробки бекенд-коду.

Основна особливість Node.js полягає у тому, що він працює за асинхронним та подієвим принципом. Це означає, що Node.js може обробляти багато запитів одночасно без блокування виконання інших операцій. Це забезпечує високу продуктивність та швидке реагування на запити.

Завдяки асинхронному підходу та використанню ефективного двигуна V8, Node.js демонструє високу продуктивність та швидкодію. Він добре підходить для обробки великого обсягу одночасних запитів та високонавантажених додатків.

Node.js добре підходить для розробки масштабованих додатків. Він має побудовуватись на основі принципу горизонтального масштабування, що дозволяє додавати нові вузли для обробки запитів. Крім того, існують різні інструменти та бібліотеки, такі як класи пулу з'єднань, які полегшують масштабування додатків на Node.js.

Node.js підтримується на різних операційних системах, таких як Windows, macOS та Linux. Це дозволяє розгортати додатки на різних серверних середовищах залежно від потреб проекту.

Node.js може використовуватися для розробки різноманітних додатків, включаючи веб-сервери, API, чат-додатки, реального часу додатки, мікросервіси та багато іншого. Він став популярним вибором для розробників завдяки своїй продуктивності, гнучкості та широким можливостям.

3) Express - це легка та гнучка веб-фреймворк для розробки серверних додатків на мові JavaScript з використанням платформи Node.js. Він дозволяє швидко створювати веб-додатки та API, спрощуючи процес розробки серверної частини.

Він надає потужні механізми для визначення маршрутів, що дозволяє обробляти HTTP-запити на певні URL-шаблони та виконувати необхідні дії. Це дозволяє логічно структурувати додаток та визначати різні обробники запитів для конкретних URL.

Express базується на концепції middleware, що дозволяє обробляти запити перед їх досягненням обробника маршруту. Middleware може виконувати різні операції, такі як логування, перевірка автентифікації, обробка статичних файлів та багато іншого. Це дозволяє створювати повторно використовувані компоненти та забезпечувати гнучкість у керуванні потоком запитів.

Він надає простий спосіб обробки статичних файлів, таких як зображення, CSS-файли, JavaScript-файли тощо. Це дозволяє встановлювати папки зі статичними ресурсами та використовувати їх безпосередньо з сервера.

Express є одним з найпопулярніших фреймворків для розробки веб-додатків на Node.js завдяки своїй простоті використання, гнучкості та великій спільноті розробників. Він дозволяє створювати швидкі та масштабовані веб-додатки за допомогою JavaScript.

4) React Native є відкритою платформою для розробки мобільних додатків, що базується на JavaScript та бібліотеці React. Він дозволяє розробляти

кросплатформні додатки, які працюють як на iOS, так і на Android, використовуючи один і той же код.

Завдяки використанню React Native, розробники можуть створювати мобільні додатки для різних платформ, таких як iOS та Android, використовуючи спільний код. Це дозволяє економити час та зусилля, оскільки не потрібно писати окремий код для кожної платформи.

Він використовує компіляцію до нативного коду, що забезпечує високу продуктивність та швидкодію додатків. Він використовує нативні компоненти, що дозволяє додаткам працювати на більш низькому рівні, ніж традиційні гібридні додатки.

React Native базується на компонентній архітектурі, що дозволяє розробникам повторно використовувати компоненти та швидко будувати складні інтерфейси. Компоненти можуть бути написані один раз і використовуватись на різних платформах, що спрощує розробку та підтримку додатків.

React Native надає доступ до багатьох готових компонентів та бібліотек, що дозволяють швидко розробляти функціонально багаті додатки. Існують також інструменти, як Expo, які полегшують розробку та тестування додатків.

Розробка мобільних додатків з використанням React Native дозволяє зменшити зусилля, необхідні для розробки та підтримки кросплатформених додатків, і забезпечує високу продуктивність та широкі можливості для створення інтерактивних та потужних додатків для мобільних пристроїв.

5) AdminJS - це відкритий фреймворк для швидкої розробки адміністративних панелей для веб-додатків. Він дозволяє розробникам легко створювати властиві адміністраторські інтерфейси для управління даними, конфігурацією та іншими аспектами веб-додатків.

Він побудований на принципах модульності, що дозволяє розширювати його функціональність за допомогою плагінів. Розробники можуть використовувати наявні плагіни або створювати власні для налаштування адміністративного інтерфейсу під свої потреби.

AdminJS працює з різними типами даних, такими як реляційні бази даних (наприклад, PostgreSQL, MySQL), NoSQL-сховища (наприклад, MongoDB) та інші. Це дозволяє розробникам працювати з різноманітними джерелами даних у єдиному адміністративному інтерфейсі.

Фреймворк надає можливість легко налаштовувати інтерфейс адміністративної панелі. Розробники можуть визначати моделі даних, поля, форми редагування, фільтри, дії та інші елементи відповідно до своїх потреб.

Він має вбудовану систему аутентифікації та авторизації, яка дозволяє обмежувати доступ до адміністративного інтерфейсу і визначати права доступу для користувачів.

AdminJS надає API для створення власних компонентів, плагінів та middleware, що дозволяє розробникам розширювати функціональність фреймворку і додавати власні функції та можливості.

Він має сучасний інтерфейс з використанням компонентів із Material UI або Ant Design, що забезпечує зручний і приємний досвід користувача.

Висновки

У цьому розділі було розроблено систему та обрано оптимальний варіант цієї архітектури залежно від цільової області. Інтерфейс користувача розроблено на основі специфікації вимог. Розроблено структуру бази даних. Розроблено архітектуру системи та її компоненти.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Node.js полегшує взаємодію з базами даних MySQL. Для цього потрібна бібліотека sequelize для підключення та запиту бази даних. Ви можете встановити його за допомогою такої команди:

```
npm install sequelize
```

Далі, необхідно встановити локально MySQL і створити в ній базу даних. Я використовую операційну систему Manjaro, тому нижче буде описано встановлення для цієї операційної системи.

Для початку потрібно встановити MySQL. Для цього потрібно запустити команду:

```
sudo pacman -S mysql
```

І натиснути у в діалоговому вікні. Приклад зображено на рисунку 3.1

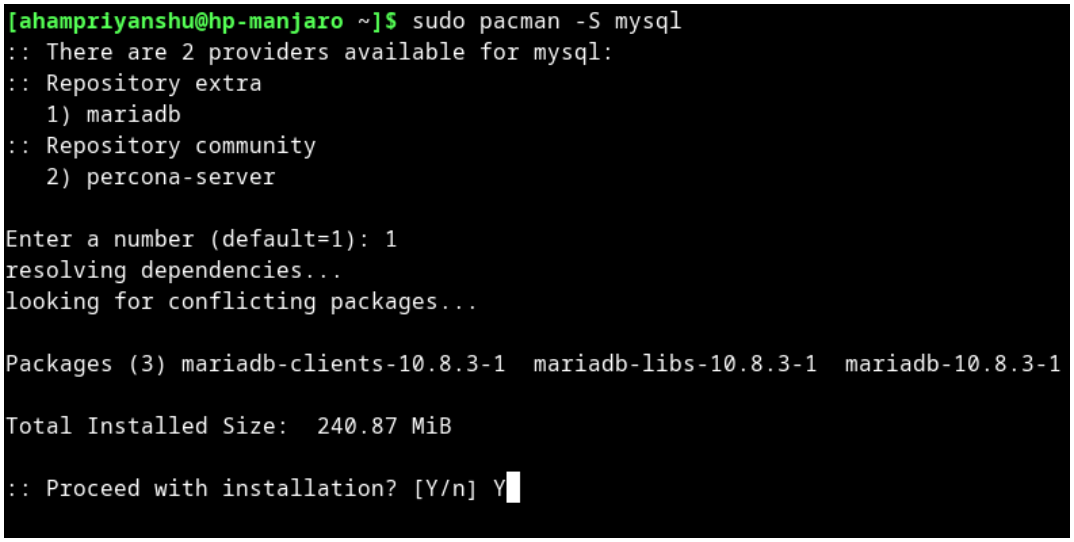


Рисунок 3.1 – встановлення MySQL

Далі потрібно запустити MySQL сервер:

```
sudo systemctl start mysqld
```

Після цього потрібно надати MySQL права для запуску в системі, для цього потрібно запустити команду:

```
sudo systemctl enable mysqld
```

Після встановлення і отримання дозволів потрібно конфігурувати, для цього введіть в консоль:

```
sudo mysql_secure_installation
```

Після того як усе виконається можна запустити MySQL:

```
sudo mysql
```

У новому вікні створіть нового користувача, за допомогою якого будете входити у базу даних і надайте йому превілеї для цього:

```
CREATE USER '<username>'@'localhost' IDENTIFIED BY '<password>';
GRANT ALL PRIVILEGES ON *.* TO '<username>'@'localhost' WITH GRANT OPTION;
FLUSH PRIVILEGES;
```

Далі можете вийти з mysql виконавши команду:

```
exit;
```

І ввійти вже за паролем і логіном юзера, який створювали раніше:

```
mysql -u <username> -p
```

У новому вікні введіть пароль, як показано на рисунку 3.2

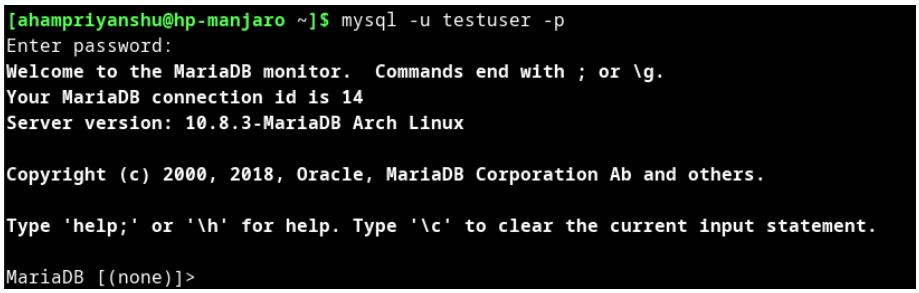


Рисунок 3.2 – Вхід у MySQL за допомогою створеного користувача

Далі можна створити нову базу даних для цього введіть в консолі команду:

```
CREATE DATABASE <dbname>;
```

Після чого можна побачити її у списку виконавши команду (приклад результату зображено на рисунку 3.3):

```
SHOW DATABASES
```

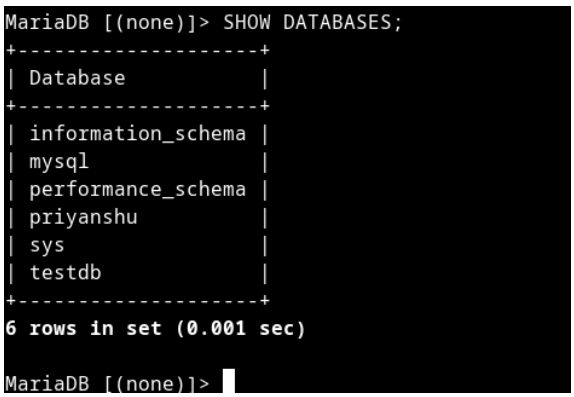


Рисунок 3.3 – Список баз даних після створення

Тепер можна використовувати localhost:3306 для підключення до бази даних. Також потрібно в файлах серверу створити файл конфігурації sequelize.js, за допомогою якого сервер і буде підключатись до бази даних. Приклад вмісту файлу зображено на рисунку 3.4

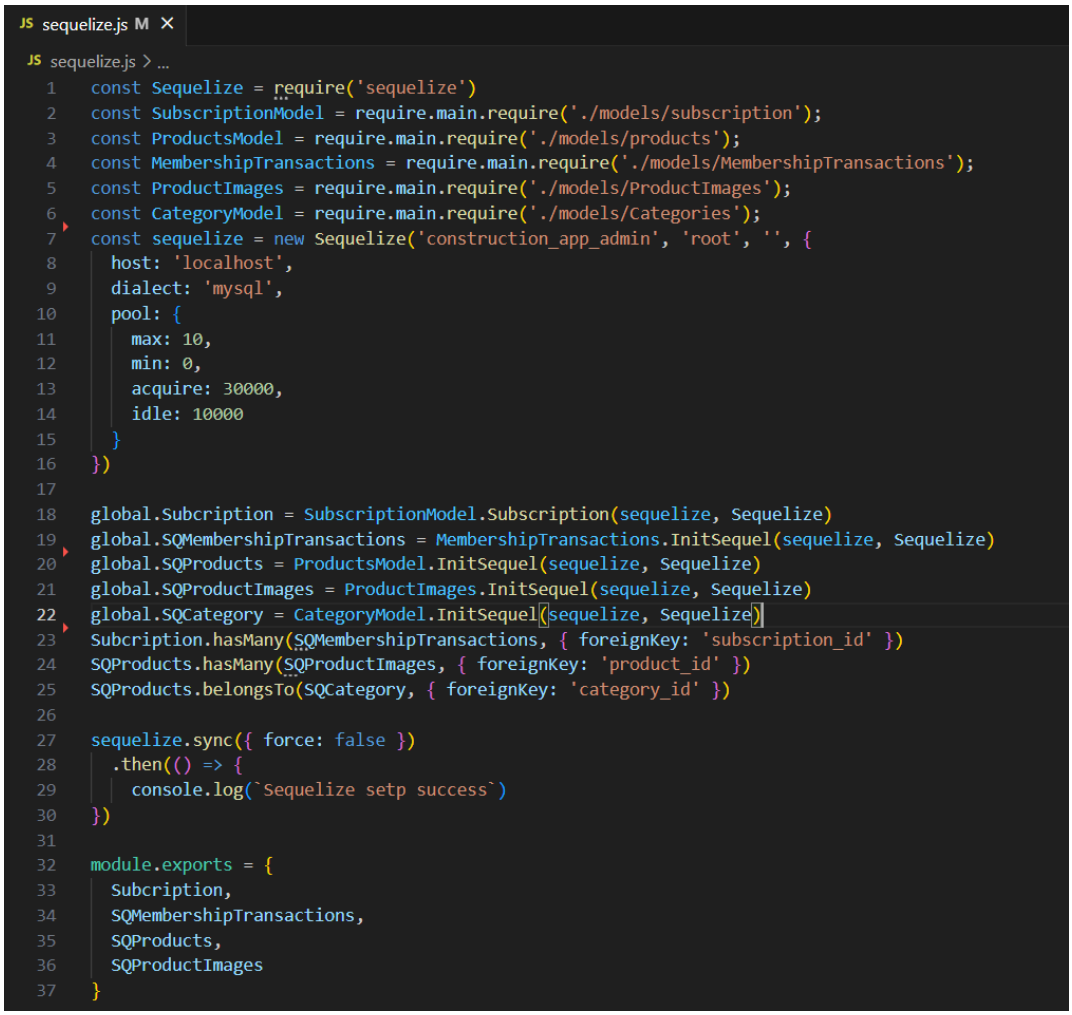


Рисунок 3.4 – приклад файлу конфігурації sequelize.js

Після чого MySQL успішно підключено до нашого проекту і можна починати створювати моделі, які і будуть використовуватись як класи створених таблиць для збору з них даних.

3.2 Розробка програмних модулів

Для початку опишемо серверну частину проекту. Її структура зображена на рисунку 3.5:

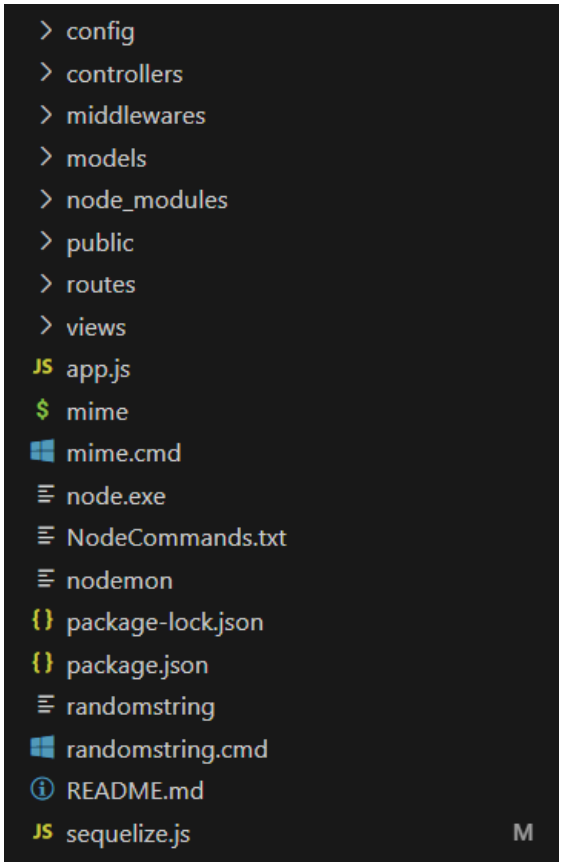


Рисунок 3.5 - Структура серверної частини проекту

Серед структури необхідно виділити такі файли та папки:

– config – файли конфігурації (всередині знаходиться db.js – файл конфігурації бази даних)

– controllers – контролери, основна його функція полягає в обробці вхідних даних від користувача, взаємодії з моделлю (бізнес-логікою) і оновленні поверненні результату для клієнту. Контролер вирішує, яка дія потрібна відповідно до запиту користувача та координує виконання цієї дії. Структура папки controllers зображена на рисунку 3.6

Всередині є файли AuthController (для авторизації через API), OrdersController (Для створення замовлень через API), ProductsController (Для повернення товарів кав'ярні)

Також є папка admin, всередині якої контролери для адмін панелі: AuthController (для авторизації в адмін панель), CategoriesController (CRUD

категорій), OrdersController (інформація про замовлення), ProductsController (CRUD товарів), UsersController (інформація про працівників)

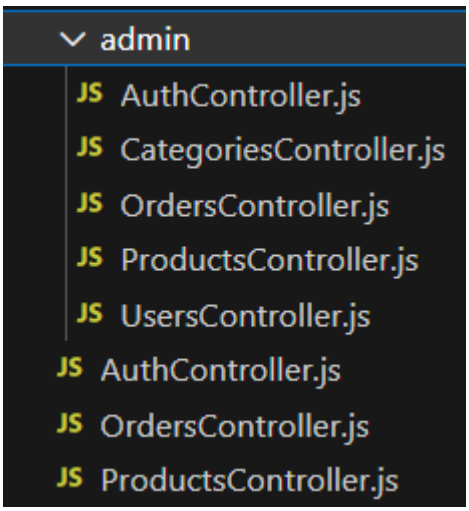


Рисунок 3.6 – Структура папки controllers

– middlewares – посередники між контролерами і роутами, в даному випадку, потрібні для валідації даних і авторизації. Структура папки зображена на рисунку 3.7

Всередині є файл AuthMiddleware (необхідний для авторизації через API), папка validators, в якій містяться валідатори API

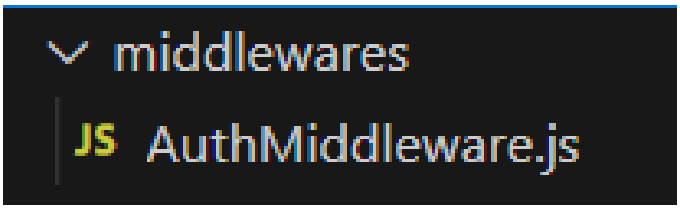


Рисунок 3.6 – Структура папки middlewares

– models – моделі, які слугують для зв’язку з таблицями в базі даних. Структуру папки зображено на рисунку 3.7. Всередині є файли Category, Employee, Order, OrderProduct, Product

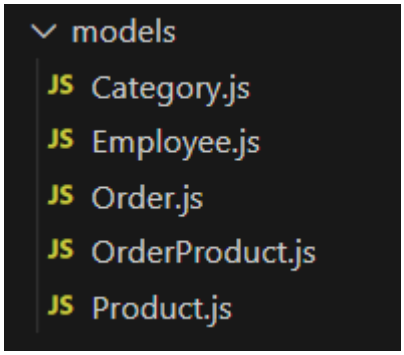


Рисунок 3.6 – Структура папки models

Приклад моделі:

```
import {Sequelize} from "sequelize";
import db from "../config/database.js";

const {DataTypes} = Sequelize;

const User = db.define('user', {
    name: {
        type: DataTypes.STRING
    },
    surname: {
        type: DataTypes.STRING
    },
    fathername: {
        type: DataTypes.STRING
    },
    email: {
        type: DataTypes.STRING
    },
    password: {
        type: DataTypes.STRING
    },
    salary:{
        type: DataTypes.INTEGER
    },
    address:{
        type: DataTypes.TEXT
    }
    phone:{
        type: DataTypes.TEXT
    }

}, {
    freezeTableName: true,
    timestamps:false
});
(async () => {
    await db.sync()
})();

export default User;
```

– public – всередині зберігаються картинки, які завантажуються адміністраторами для продуктів.

– routes - роути - це файли, що визначають шляхи доступу до різних функцій та ресурсів веб-додатку. Роути визначають, які URL-шляхи будуть відповідати певним операціям та які дії повинні бути виконані при обробці таких запитів. Структура папки зображена на рисунку 3.7.

Всередині є файли index (всередині підключаються всі роути), AuthRouter (авторизація), OrderRouter (створення замовлень), ProductRouter (повернення продуктів)

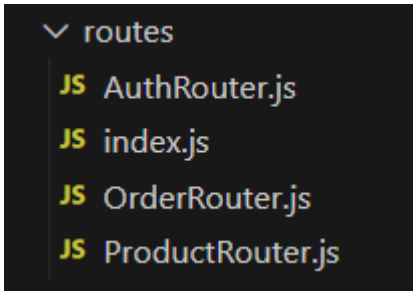


Рисунок 3.7 – структура папки routes

Приклад роута:

```
import { Router } from 'express'
import { authMiddleware } from '../middlewares/AuthMiddleware.js'
import ProductController from
'../controllers/ProductController.js'

const productRouter = Router({ mergeParams: true })

productRouter.get(
  '/all',
  authMiddleware,
  ProductController.getAllProducts
)
productRouter.get(
  '/:productId',
  authMiddleware,
  ProductController.getProductById
)
productRouter.get(
    '/category/:categoryId',
    authMiddleware,
    ProductController.getProductByCategoryId
  )

productRouter.get('search/:text', authMiddleware,
ProductController.findProduct)
```

```
export default productRouter
```

– views – представлення, які слугують для створення сторінок адмін панелі і повернення даних на них. Структура папки зображена на рисунку 3.8.

Всередині є файл dashboard(головна сторінка), login (авторизація), Categories – add (додавання категорії), edit(редагування категорії), list(список категорій), Products – add (додавання товару), edit(редагування товару), list(список товарів), Users – add (додавання користувача), edit(редагування користувача), list(список користувачів)

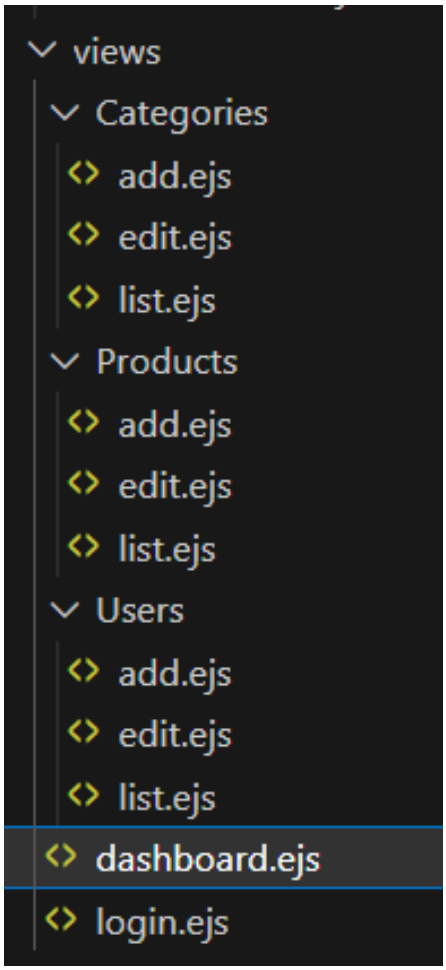


Рисунок 3.8 – структура папки views

Далі опишемо клієнтську частину проекту (мобільний додаток). Її структура зображена на рисунку 3.9:

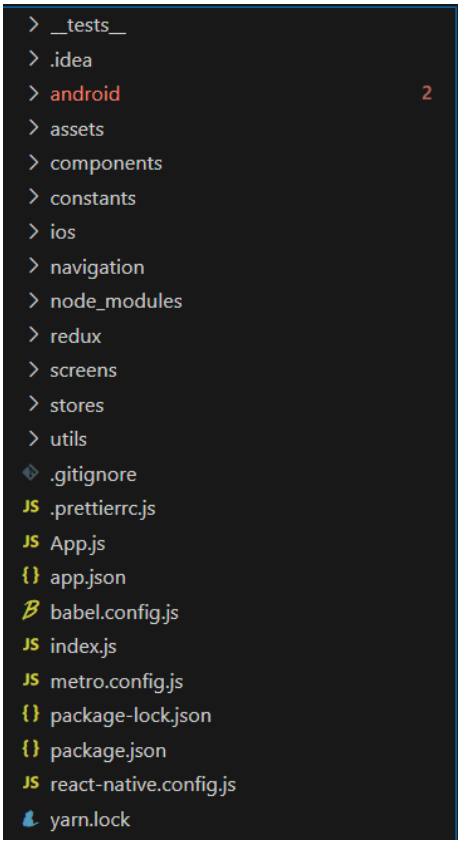


Рисунок 3.9 –Структура клієнтської частини проекту

Серед структури необхідно виділити такі папки:

– android – файли, що відповідають за нативні модулі для систем на базі операційної системи android

– ios – файли, що відповідають за нативні модулі для систем на базі операційної системи ios

– assets – стилі, іконки і картинки що використовуються в застосунку

– components – реакт компоненти, з яких будується JavaScript код.

– constants – файли з сталими значеннями, які використовуються в проекті (можуть бути як просто числа, так і цілі компоненти)

– navigation – файл для навігації по сторінкам

– redux – файли для сесії всередині системи, для цього використовується Redux - це популярна бібліотека для керування станом додатків у середовищі JavaScript, яка використовується переважно в контексті розробки веб-додатків з використанням фреймворків, таких як React, Angular або Vue.js.

– screens – сторінки застосунку

– utils – функції, які використовуються кілька разів в різних компонентах

3.3 Керівництво користувача

Керівництво користувача потрібно описувати з двох сторін: адмін панель і мобільний застосунок.

Для початку опишемо мобільний застосунок. Коли користувач заходить у застосунок, першою сторінкою, яка йому відкриється буде сторінка авторизації. На цій сторінці буде форма, в яку він має ввійти з логіном і паролем, які були задані при створенні користувача в адмін панелі. Дана сторінка зображена на рисунку 3.10.

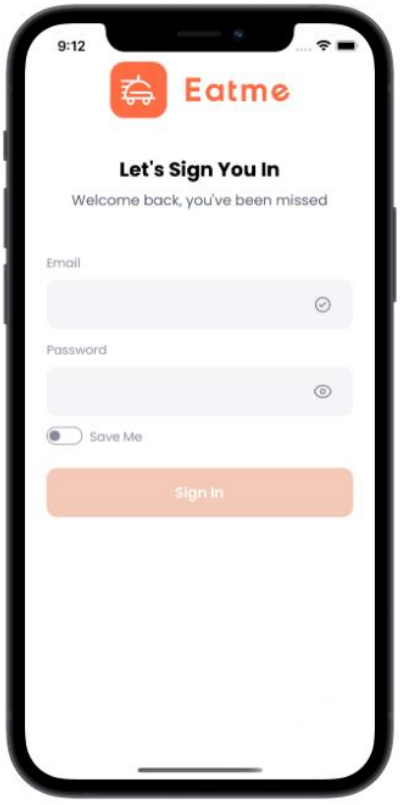


Рисунок 3.10 – Сторінка авторизації працівника

Після авторизації працівник опиниться на сторінці, де він зможе переглянути всі товари, що є, переглянути товари певної категорії і знайти товар по назві. Зовнішній вигляд даної сторінки зображено на рисунку 3.11.

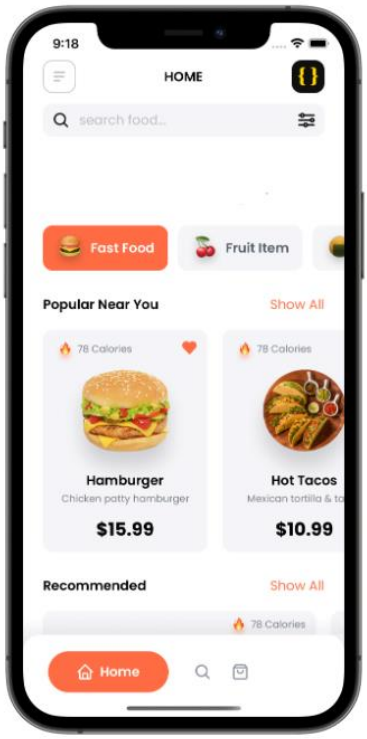


Рисунок 3.10 – Головна сторінка застосунку

Далі працівник повинен обрати товар і клікнути на нього, після чого він опиниться на детальній сторінці товару. Тут він зможе переглянути інформацію про товар (картинку, назву, опис), вибрати кількість товару і додати його в замовлення. Її зовнішній вигляд зображено на рисунку 3.11

Рисунок 3.11 – Детальна сторінка товару

Після того як працівник додасть товар до замовлення і в правому верхньому куті клікна на іконку кошика він опиниться на сторінці кошику, де будуть всі його товари, тут він зможе змінити кількість товару і видалити його. Зовнішній вигляд сторінки зображено на рисунках 3.12 і 3.13.

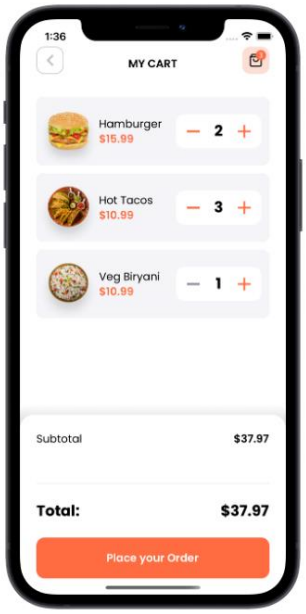


Рисунок 3.12 – Сторінка кошику

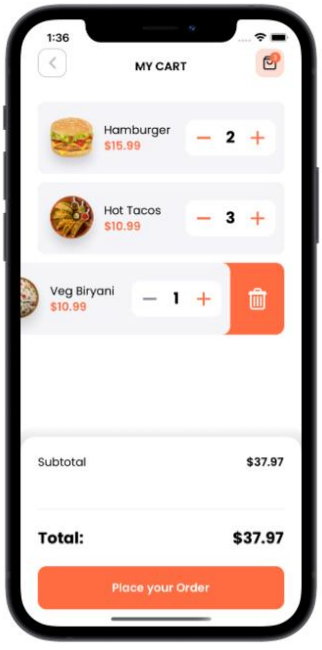


Рисунок 3.13 – Сторінка кошику з активною кнопкою видалення товару

Після розміщення замовлення його статус зміниться на завершене і працівник знову опиниться на головній сторінці (дивитись рисунок 3.10)

Після цього можна описати адмін панель. Коли користувач зайде у веб-застосунок, він опиниться на сторінці авторизації. Де адміністратору потрібно буде ввійти в систему під логіном і паролем, що захардкоджені. Зовнішній вигляд цієї сторінки зображено на рисунку 3.14.

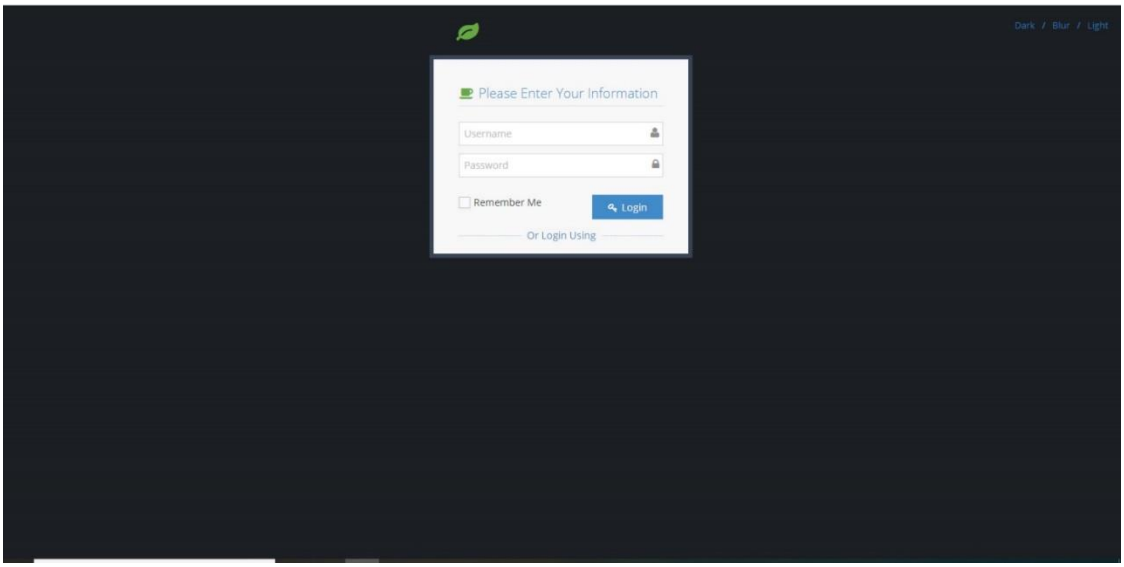


Рисунок 3.14 – Зовнішній вигляд авторизації в адмін панелі

Після авторизації користувач опиниться на сторінці списку продуктів, тут він зможе переглянути всі товар, відредагувати їх, видалити або додати новий. Зовнішній вигляд сторінки зображено на рисунку 3.15

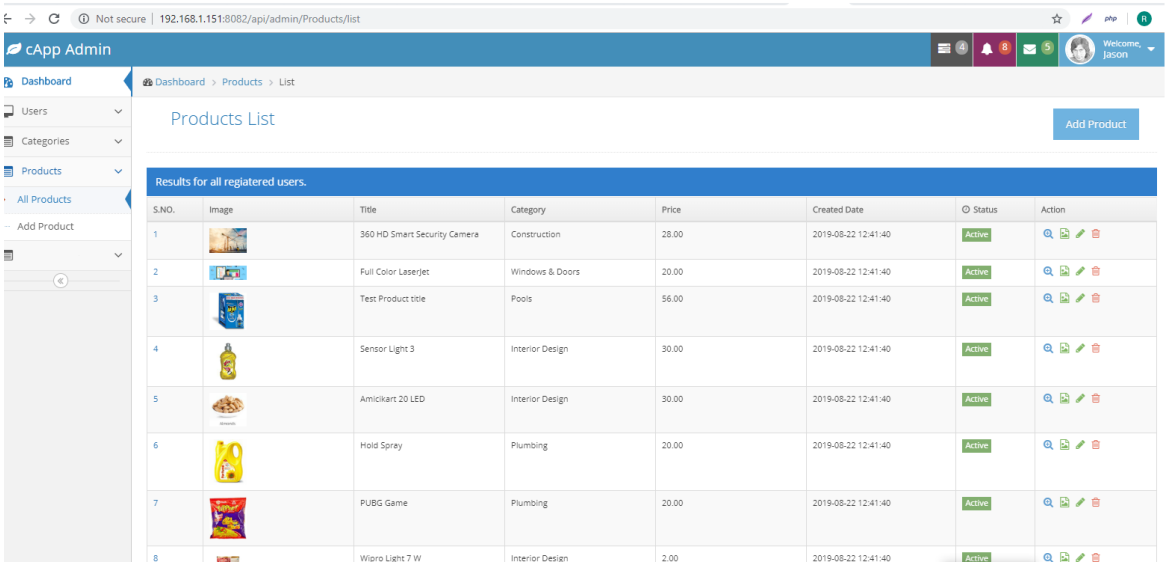


Рисунок 3.15 – Зовнішній вигляд сторінки товарів

При кліку на Add Product користувач опиниться на сторінці додавання товару, де він зможе заповнити дані про товар і додати його в систему. Зовнішній вигляд сторінки зображено на рисунку 3.16

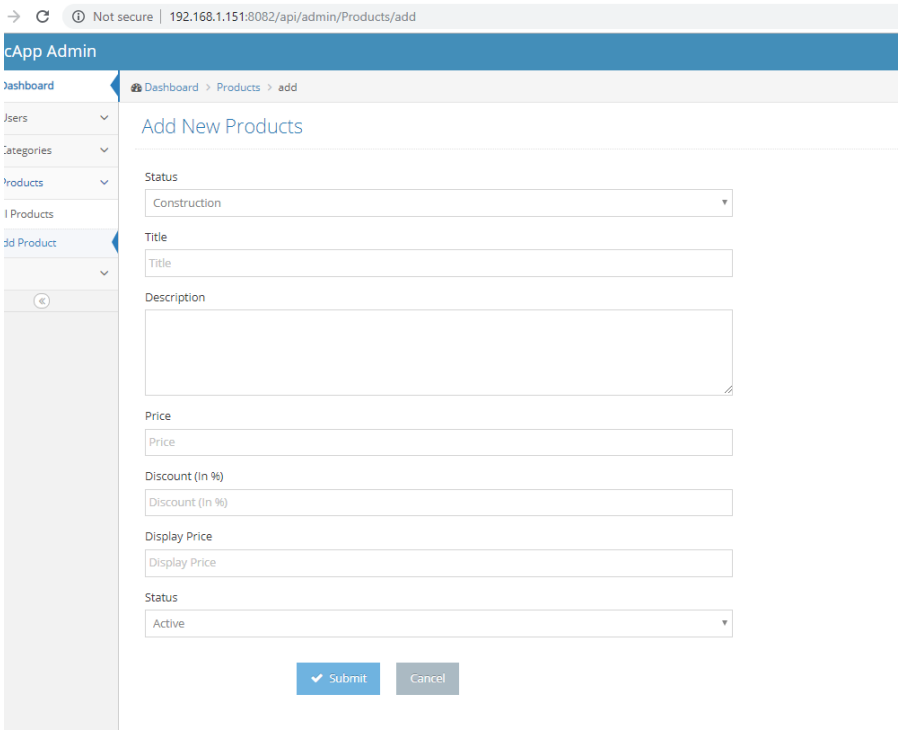


Рисунок 3.16 – Зовнішній вигляд сторінки додавання товару

Таку ж структуру мають вкладки Users і Categories. Вкладка Orders має лише сторінку зі списком.

Отже, після опису керівництва користувача для адмін панелі і мобільного застосунку даний підрозділ завершено і можна приступити до опису технічних характеристик платформи.

3.4 Технічні характеристики інтернет-платформи

Мінімальні технічні характеристики для пристрою для запуску серверу і перегляду адмін панелі можуть включати наступне:

– Операційна система: Підтримка сучасних операційних систем, таких як Windows, macOS, Linux, Android або iOS.

– Веб-браузер: Використання найновішої версії сумісного веб-браузера, такого як Google Chrome, Mozilla Firefox, Safari, Microsoft Edge.

– Процесор: Мінімум двоядерний процесор з частотою не менше 1.8 ГГц.

– Оперативна пам'ять (RAM): Рекомендовано мати не менше 4 ГБ оперативної пам'яті для плавної роботи веб-застосунку і запобігання зависанням або повільному завантаженню сторінок.

– Дисплей: Мінімальне розширення екрану 1280x800 пікселів для зручного відображення веб-інтерфейсу.

– Інтернет-підключення: Стабільне та достатньо швидке Інтернет-підключення для завантаження сторінок, зображень та інших ресурсів веб-застосунку.

– Вбудовані функції: Підтримка JavaScript та включені вбудовані функції, такі як відтворення звуку та відео, для повного функціоналу веб-застосунку.

Мінімальні технічні характеристики для пристрою для запуску мобільного застосунку:

– Операційна система: Підтримка сучасних мобільних операційних систем, таких як Android або iOS.

– Версія операційної системи: Застосунок може вимагати певну мінімальну версію операційної системи, яка повинна бути встановлена на пристрої.

– Процесор: Мінімум двоядерний процесор зі швидкістю не менше 1.2 ГГц.

– Оперативна пам'ять (RAM): Рекомендовано мати не менше 1 ГБ оперативної пам'яті для забезпечення плавної роботи застосунку і запобігання збоїв або повільному відгуку.

– Дисплей: Мінімальне розширення екрану залежить від типу пристрою, але рекомендоване розширення 480x800 пікселів або вище для забезпечення зручного відображення веб-інтерфейсу.

– Інтернет-підключення: Мобільний застосунок може вимагати стабільне та достатньо швидке підключення до Інтернету для завантаження даних, оновлень або взаємодії з сервером.

3.5 Розгортання та встановлення системи

Для розгортання цього проекту необхідно мати такі компоненти на вашому комп'ютері, як Node.JS, MySQL, Android Studio. Почніть з відкриття папки з серверною частиною проекту. Для запуску виконайте наступні кроки

- Виконайте команду "npm install" для встановлення залежностей проекту.
- Запустіть команду "npm start" для запуску серверної частини.

Для запуску клієнтської частини відкрийте папку з клієнтською частиною в терміналі і виконайте такі кроки:

- Запустіть Android Studio, у відкритому вікні виберіть More Actions і у списку виберіть Virtual Device Manager, як показано на рисунку 3.17

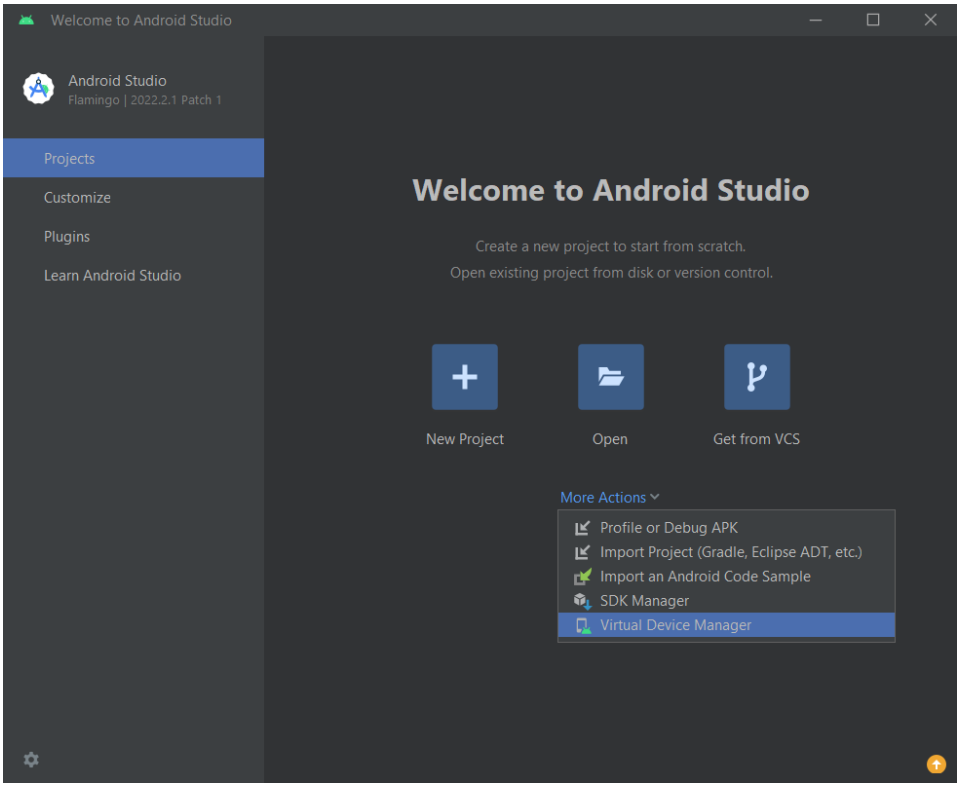


Рисунок 3.17 – Головна сторінка Android Studio

– У вікні, що з’явиться, яке зображено на рисунку 3.18 створіть новий пристрій або запустіть вже наявний. Після чого у вас відкриється нове вікно, що зображено на рисунку 3.19 де потрібно виконати комбінацію клавіш (Ctrl + P) для запуску емулятора смартфону.

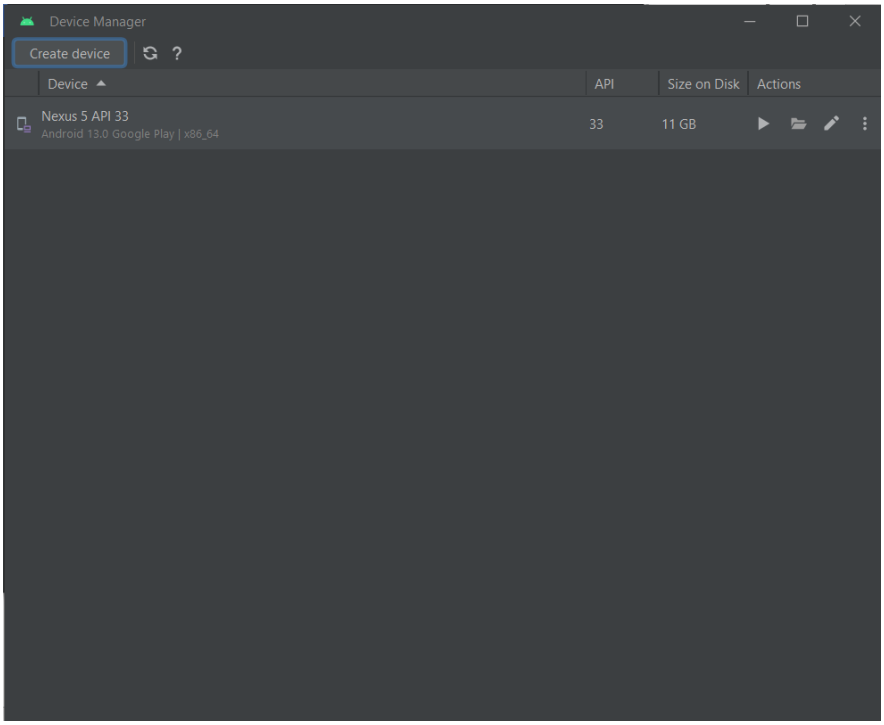


Рисунок 3.18 – Сторінка вибору пристрою в Android Studio

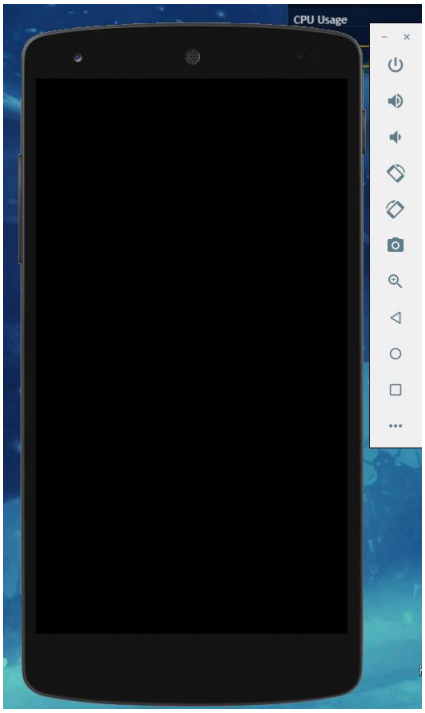


Рисунок 3.19 – Зовнішній вигляд емулятора смартфону

– Далі у терміналі, що відкрили раніше, виконайте команду "npm install" для встановлення залежностей проекту.

– Запустіть команду "npm run start" для запуску клієнтської частини.

– Запустіть команду "npm run android" для запуску застосунку в емуляторі.

Після виконання цих дій сервер, адмін панель і мобільний застосунок будуть запущені і готові до тестування.

3.6 Тестування веб-застосунку

Створення тестів для React Native додатку включає кілька етапів і вимагає використання певних інструментів та підходів. Ось обширний опис процесу створення тестів для React Native додатку:

– Визначення тестових вимог: Перш за все, потрібно визначити, які частини додатку будуть тестуватися. Це можуть бути основні функціональність, екрани, компоненти, взаємодії зі зовнішніми сервісами та інше. Важливо чітко визначити мету тестування та очікувані результати.

– Вибір тестового фреймворку: Для створення тестів вибирають підходящий тестовий фреймворк. У випадку React Native часто використовують такі фреймворки, як Jest або Detox. Jest є популярним фреймворком для модульного тестування React Native компонентів та функцій, а Detox - для написання інтеграційних тестів, які симулюють взаємодію користувача з додатком.

– Написання тестових сценаріїв: На цьому етапі ви створюєте тестові сценарії, які перевірятимуть очікувану поведінку вашого додатку. Тестові сценарії можуть включати перевірку рендерингу компонентів, взаємодії з користувачем, валідацію даних, навігацію по екранах та інші важливі аспекти функціональності.

– Запуск тестів: За допомогою обраного тестового фреймворку ви запускаєте свої тестові сценарії. Фреймворк автоматично виконує тестові

сценарії і повідомляє про результати. Ви можете перевірити, чи пройшли тести успішно, чи виникли помилки та інші релевантні дані про виконання тестів.

- Виправлення помилок: Якщо тести не пройшли успішно, ви виправляєте помилки, які були виявлені. Це може включати виправлення дефектів у коді, налаштування компонентів або виправлення проблем зі зв'язком зі зовнішніми сервісами.

- Автоматизація тестів: Якщо ви бажаєте мати повноцінний набір тестів, ви можете автоматизувати їх запуск. Це означає, що ви створюєте скрипти або налаштовуєте засоби Continuous Integration (CI), щоб автоматично запускати тести при кожному зміні в коді або при регулярному розкладі.

- Моніторинг результатів: Після кожного запуску тестів ви моніторите результати і аналізуєте дані про виконання тестів. Це допомагає виявляти проблеми та помилки, що можуть виникати під час розробки та внесення змін в код.

Важливо зазначити, що процес створення тестів для React Native додатку може варіюватися залежно від вимог проекту, обсягу роботи та вибраного підходу до тестування. Завжди рекомендується дотримуватися найкращих практик тестування та створювати розширені набори тестів для забезпечення якості та надійності вашого React Native додатку.

Приклад тесту, використовуючи бібліотеку Jest:

```
import React from 'react';
import { render, fireEvent } from '@testing-library/react-native';
import App from '../App'; // Підставте шлях до вашого основного компонента додатку

describe('App', () => {
  test('перевірка відображення привітання', () => {
    const { getByTestId } = render(<App />);
    const greetingText = getByTestId('greeting-text');
    expect(greetingText.props.children).toBe('Привіт, світ!');
  });

  test('перевірка взаємодії кнопки', () => {
    const { getByTestId } = render(<App />);
    const button = getByTestId('greeting-button');
    const greetingText = getByTestId('greeting-text');

    fireEvent.press(button);
```

```
          expect(greetingText.props.children).toBe('Привіт,
користувачу!');
        });
      });
```

У цьому прикладі ми тестуємо компонент App, який відображає привітання і має кнопку, яка змінює привітання при кліку. Ми використовуємо функцію render з @testing-library/react-native для рендерингу компонента в тестовому середовищі.

У першому тесті ми перевіряємо, чи правильно відображається початкове привітання. Ми використовуємо getByTestId для отримання елементу з атрибутом testID, який ми встановили у відповідному елементі в компоненті App. Потім ми перевіряємо, чи значення тексту відповідає очікуваному.

У другому тесті ми перевіряємо, чи працює взаємодія з кнопкою. Ми знову використовуємо getByTestId для отримання кнопки і тексту привітання. Потім ми симулюємо клік на кнопку за допомогою fireEvent.press і перевіряємо, чи змінилося значення тексту на очікуване.

Висновки

Після програмної реалізації були зроблені деякі висновки. Були певні труднощі з програмною реалізацією, але загалом вона виконана в повному обсязі та має перспективу для розширення програми та додавання нових функцій. База даних була створена під час програмної реалізації, щоб надати можливість автентифікації користувача і збереження даних. Розроблено код для серверної та клієнтської частин програми. У розділі тестування додатків, створюючи набори тестів і пишучи модульні тести, ми покращуємо безпеку та працездатність додатків і створюємо основу для подальшого розширення функціоналу застосунку.

ВИСНОВКИ

У цій кваліфікаційній роботі був реалізований застосунок для автоматизації роботи кав'ярень з впровадженням сервісу бухгалтерського обліку. Розробка такого застосунку має великий потенціал для покращення ефективності та точності бухгалтерського обліку в кав'ярнях, а також спрощення їх повсякденної роботи.

Застосунок забезпечує різноманітні функціональні можливості, необхідні для ефективного управління кав'ярнею. Він включає модулі для замовлення продуктів, обліку запасів, управління персоналом, керування категоріями товарів, перегляд замовлень.

За допомогою цього сервісу власники кав'ярень можуть отримати швидкий та точний звіт про фінансовий стан свого бізнесу, що сприяє прийняттю обґрунтованих управлінських рішень.

Застосунок використовує сучасні технології розробки, зокрема мову програмування JavaScript, для серверної частини NodeJS, фреймворк ExpressJS, систему керування базами даних MySQL, для адмін панелі фреймворк AdminJS та для розробки мобільної частини застосунку фреймворк React Native. Це дозволяє забезпечити швидку розробку, масштабованість та кросплатформенну підтримку.

Результати роботи підтверджують, що розроблений застосунок ефективно сприяє автоматизації робочих процесів кав'ярень і полегшує їхнє управління. Застосунок дозволяє економити час, зменшувати помилки в бухгалтерському обліку та підвищувати задоволеність клієнтів. Його можна використовувати в різних кав'ярнях та адаптувати до їхніх специфічних потреб.

В подальшому розробку застосунку можна розширити шляхом введення додаткових функцій, таких як онлайн-замовлення, лояльність клієнтів, зв'язок зі зовнішніми платіжними системами тощо. Це дозволить ще більше поліпшити роботу кав'ярень та забезпечити їх успішне функціонування в сучасному цифровому середовищі.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Л. П. Бедратюк. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Л. П. Бедратюк, Г. І. Радельчук, Ю. В. Форкун. О. М. Яшина – Хмельницький: ХНУ, 2020. – 77с . (дата звернення – 07.05.2022).
2. Fowler, M. (2002). "Patterns of Enterprise Application Architecture". Addison-Wesley Professional.
3. Застосунок аналог Toast POS https://pos.toasttab.com/ (дата звернення - 30.05.23).
4. Застосунок аналог JoinPoster https://joinposter.com/ (дата звернення - 21.05.23).
5. Застосунок аналог Lavu https://lavu.com/ (дата звернення - 26.05.23).
6. Choi, C.H., Kim, J.K., & Kang, B.H. (2013). "An Android-based Smart Coffee Machine Control System". 2013 6th International Conference on Human System Interactions (HSI), pp. 452-457.
7. Özel, S., & Vural, S. (2019). "CoffeeShop: An Android Application for Managing Coffee Shops". International Journal of Computer Science and Software Engineering, 8(2), 39-45.
8. Sreevalli, S., Sai Krishna, M., Eswara Sowjanya, R., & Harini, M.S. (2018). "Smart Coffee Maker". 2018 3rd International Conference on Communication and Electronics Systems (ICCES), pp. 925-929.
9. Christidis, K., & Devetsikiotis, M. (2016). "Blockchains and Smart Contracts for the Internet of Things". IEEE Access, 4, 2292-2303.
10. Wilson, B., & Kusuma, A. (2014). "Development of a Mobile Application for Coffee Shop Loyalty Points System". Proceedings of the International Multiconference of Engineers and Computer Scientists, 1, 326-329.
11. Attar, R., Alarifi, A., & Alfarraj, O. (2017). "Cloud-Based Accounting Information Systems: Opportunities and Challenges". Journal of Accounting, Auditing & Finance, 32(4), 651-666.

12. Lim, Y. (2016). "The Adoption of Cloud Computing in the Malaysian Hotel Industry". Journal of Hospitality and Tourism Technology, 7(2), 167-181.

13. Srinivasa, K.G., Shyamala, M.P., & Rashmi, S.R. (2014). "Cloud Computing for Accounting: Analysis of Security Issues and Challenges". 2014 International Conference on Circuits, Power and Computing Technologies [ICCPCT-2014], pp. 1738-1743.

14. Shahzad, G., & Khan, R.A. (2017). "Cloud Computing: An Analysis of Different Security Issues and Their Solutions". 2017 14th International Bhurban Conference on Applied Sciences and Technology (IBCAST), pp. 298-303.

15. Yassin, R.M., & Abdullah, R. (2019). "Cloud-Based Accounting Information Systems (CLAIS) for Small and Medium Enterprises (SMEs): Opportunities, Challenges, and Lessons". 2019 International Conference on Computer and Information Sciences (ICCIS), pp. 1-6.

16. Lynch, R.L., & Cross, K.F. (1991). "Measure Up! Yardsticks for Continuous Improvement". Blackwell Business

17. Maharjan, B., & Shakya, A. (2017). "Cloud based accounting: A solution for small businesses in developing countries". 2017 Fourth International Conference on Advances in Electrical, Electronics, Information, Communication and Bio-Informatics (AEEICB), pp. 1-5.

18. Subramanian, K.R. (2015). "Cloud Based Accounting Services to Small and Medium Sized Enterprises". 2015 Fifth International Conference on Advanced Computing (ICoAC), pp. 340-343.

19. Chen, C., & Zhao, J. (2017). "Design and implementation of accounting management system in cloud computing environment". 2017 29th Chinese Control And Decision Conference (CCDC), pp. 2017-2022.

20. Zhang, F., Miao, C.Y., Tian, J.P., & Luo, L.G. (2011). "Design and implementation of accounting management system based on cloud computing". 2011 Second International Conference on Networking and Distributed Computing (ICNDC), pp. 1-4.

21. Yiu, C.W. (2017). "Development of an electronic accounting practice set". 2017 IEEE 7th Annual Computing and Communication Workshop and Conference (CCWC), pp. 1-6.

22. Chen, J., & Wang, C. (2018). "Discussion on the Teaching Reform of Accounting Information System Course Based on Cloud Computing". 2018 4th International Conference on Education, Law, Management and Computing (ELMC), pp. 74-78.

23. Shen, Z., Li, X., Zhang, H., Gong, G., & Wang, X. (2017). "Research and Design on Accounting System in Cloud Computing Environment". 2017 IEEE International Conference on Computer and Communications (ICCC), pp. 1912-1917.

24. Arvind, A., & Aruna, M. (2017). "Smart Coffee Ordering and Billing System Using RFID". 2017 International Conference on Electrical, Electronics, Communication, Computer, and Optimization Techniques (ICEECCOT), pp. 394-398.

25. Pabla, M.S., & Chhabra, J.K. (2017). "Wi-Fi enabled coffee machine for secured payment and billing system". 2017 IEEE Uttar Pradesh Section International Conference on Electrical, Computer and Electronics Engineering (UPCON), pp. 342-345.

26. Jeong, J.H., Yu, J.J., Park, Y.S., & Kim, J.W. (2016). "Smart Cafe: A novel mobile application development for cafe business". 2016 IEEE/ACIS 15th International Conference on Computer and Information Science (ICIS), pp. 1-5.

27. Ngo, N.B., & Le, T.D. (2017). "Designing a smart payment system using QR codes for a coffee shop". 2017 4th NAFOSTED Conference on Information and Computer Science (NICS), pp. 165-170.

28. Wang, J., & Xie, Y. (2011). "The Design of Smart Coffee Vending Machine System Based on ARM Platform". 2011 International Conference on Electronics, Communications and Control (ICECC), pp. 4314-4317.

29. Kuhn, A.J. (2010). "Accounting in the Cloud". The CPA Journal, 80(3), 6-7.

30. Mathew, J.G. (2017). "Developing the Next Generation of Accounting Software". Strategic Finance, 98(12), 23-24.

31. Maheswari, D., & Srinivasan, S.K. (2017). "An empirical study of accounting software selection and implementation criteria for small and medium-sized enterprises". 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI), pp. 2065-2071.

32. Fadlallah, H.R., & Omar, S. (2014). "ERP Systems Usage in a Developing Country: An Empirical Study". 2014 International Conference on Computer, Control, Informatics and Its Applications, pp. 100-105.

33. Banker, P., & Kauffman, P. (2004). "The Economics of ERP Systems: How Should We Account for ERP Benefits?". Management Accounting Quarterly, 5(4), 37-44.

34. Nilsson, M.F., & Rapp, K. (2009). "Impact of Enterprise Resource Planning Systems on Management Accounting". International Journal of Accounting Information Systems, 10(3), 131-147.

35. Subramanian, K.R. (2017). "Investigating the Influence of Cloud Computing on Organizational Performance". 2017 International Conference on Innovations in Information, Embedded and Communication Systems (ICIIECS), pp. 1-4.

36. Nabuco, F. (2017). "Accounting Information Systems in a Cloud Computing Environment". 2017 IEEE 7th Latin American Conference on Cloud Computing and Communications (LatinCloud), pp. 1-6.

37. Leukel, J., Kuhn, A., Schaber, M., & Suhl, R. (2011). "The Influence of Cloud Computing on Accounting". 2011 44th Hawaii International Conference on System Sciences, pp. 1-10.

38. Jain, R., Zafarani, H.F., & Kumar, R. (2011). "Cloud Computing and Accounting Information System Research". 2011 44th Hawaii International Conference on System Sciences, pp. 1-10.

39. Goswami, S., Shah, S., & Shah, M. (2012). "Accounting Software in the Cloud: The Challenges of E-Accounting". 2012 IEEE 5th International Conference on Cloud Computing, pp. 505-511.

40. Li, L., & Yu, H. (2010). "Development of Accounting Software Under the Cloud Environment". 2010 International Conference on E-Business and E-Government (ICEE), pp. 1147-1150.

41. Ge, M.F., Ma, L.Y., & Zhang, W. (2010). "Cloud computing technology on accounting software". 2010 IEEE 2nd Symposium on Web Society (SWS), pp. 74-78.

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки застосунку для автоматизації роботи кав’ярень з реалізацією сервісу бухгалтерського обліку. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на дипломний проект», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Застосунок для автоматизації роботи кав’ярень з реалізацією сервісу бухгалтерського обліку

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням додатку автоматизація роботи кав’ярні, допомога власникам в керуванні нею і в обліку даних про її стан.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на мобільних пристроях. Кінцевим користувачем додатку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для працівника:

- авторизація;
- перегляд списку товарів;
- пошук товару за назвою;
- вибір товарів за категоріями;
- перегляд детальної інформації про товари;
- додавання товарів в корзину;
- видалення товарів з корзини;
- формування замовлень;

Для адміністратора:

- авторизація;

– перегляд всіх замовлень всіх працівників;

– перегляд інформації про працівників;

– обробка даних про працівників;

– перегляд інформації про товари;

– обробка даних товари;

– перегляд інформації про категорії товарів;

– обробка даних про категорії товарів;

3.2 Вимоги до надійності

Застосунок повинен забезпечувати такі вимоги до надійності:

– обробляти невірні дії користувача і попереджати його про можливі наслідки;

– можливість самостійно відновлюватись у разі збою;

– можливість резервного копіювання бази даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Мобільний застосунок повинен працювати на мобільних пристроях, які мають стабільний доступ до мережі і мають операційні системи iOS або Android.

Веб-застосунок повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп’ютерах. Браузер може бути будь-яким.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

– мінімумдвоядерний процесор зі швидкістю не менше 1.2 ГГц;

– рекомендовано мати не менше 1 ГБ оперативної пам'яті для забезпечення плавної роботи застосунку і запобігання збоїв або повільному відгуку;

– рекомендоване розширення 480x800 пікселів або вище

3.4 Вимоги до інформаційної та програмної сумісності

При розробцівеб-додатку буде використовуватись високорівнева об’єктно-орієнтована мова JavaScript. Для розробки серверної частини використовуватиметься платформа Node.js та фреймворк Express. Для розробки

клієнтської частини була використана бібліотека інтерфейсу користувача React Native та бібліотека управління станом даних Redux. В якості СКБД використана реляційна база даних MySQL. Для розробки адмін-панелі був використаний фреймворк AdminJS.

3.5 Спеціальні вимоги

Програма повинна мати зручний, гарний та зрозумілий зовнішній інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

– текст програми;

– опис програми;

– технічне завдання;

– керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки застосунку для автоматизації роботи кав’ярень з реалізацією сервісу бухгалтерського обліку

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури

Кінець таблиці А.1

Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування додатку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

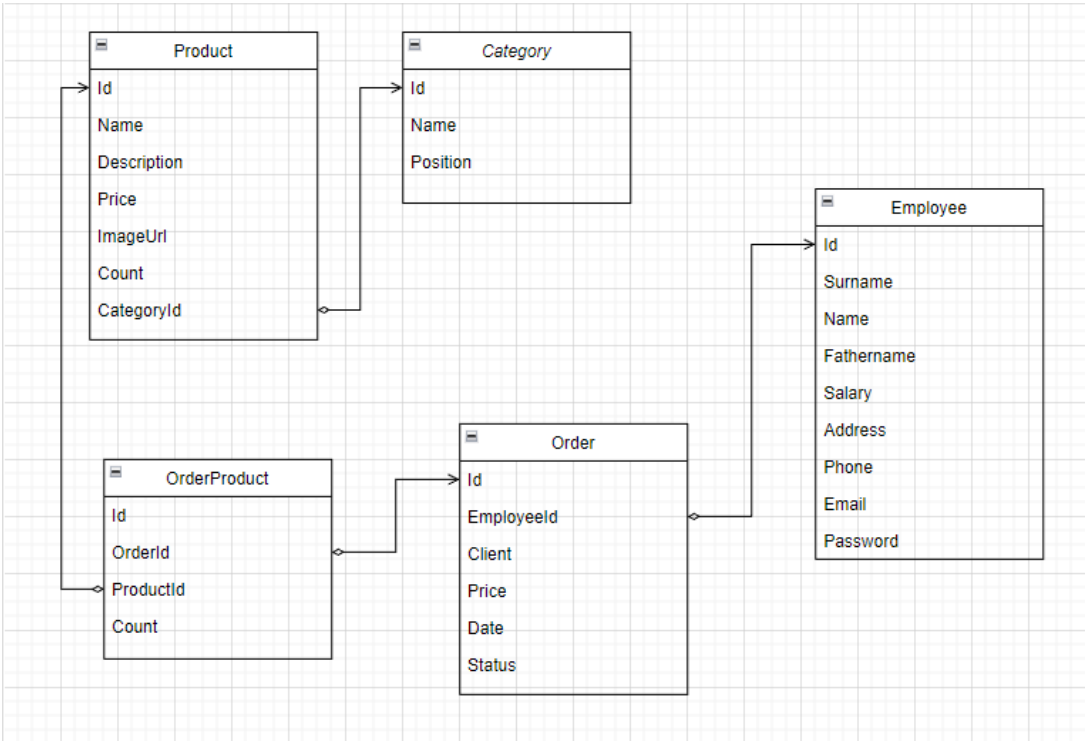


Рисунок Б.1 - Схема бази даних

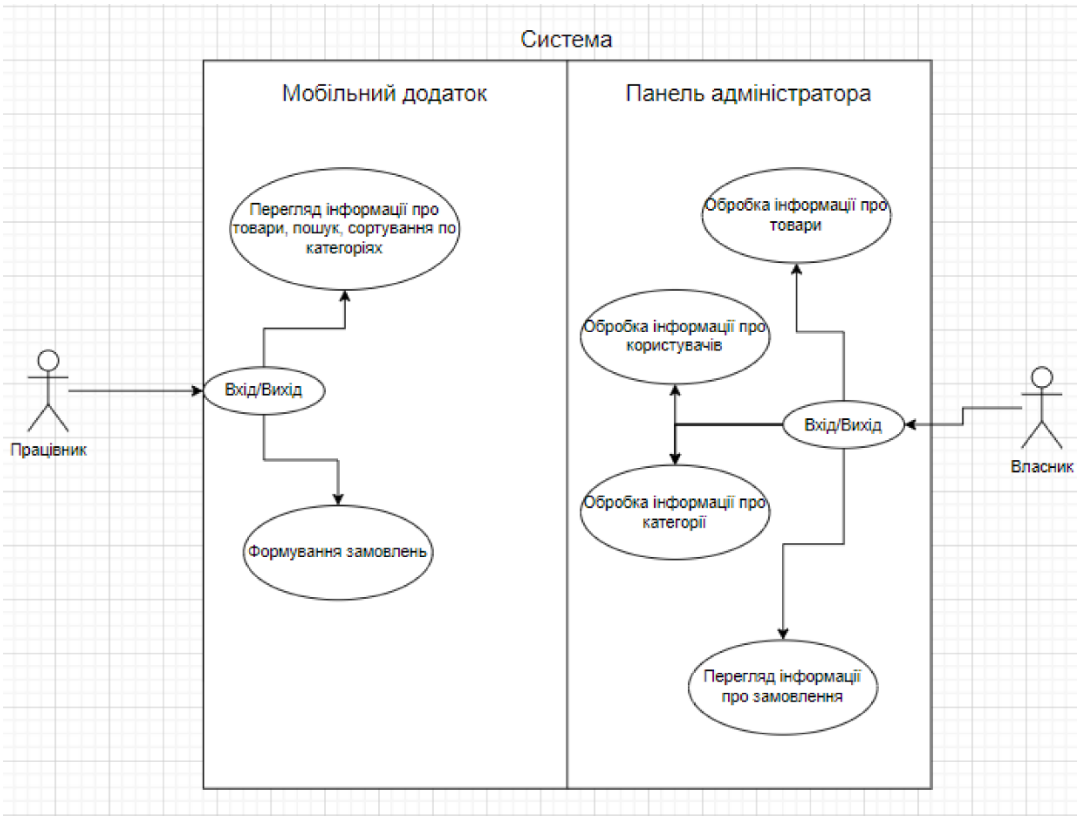


Рисунок Б.2 - Діаграма варіантів використання

ДОДАТОК В
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

App.js

```
import React from 'react'
import { createStackNavigator } from '@react-navigation/stack'
import { NavigationContainer } from '@react-navigation/native'
import SplashScreen from 'react-native-splash-screen'

import { Provider } from 'react-redux'

import createStore from './redux/store'
import CustomDrawer from './navigation/CustomDrawer'
import { OnBoarding, SignIn, SignUp, ForgotPassword, Otp } from './screens'

const Stack = createStackNavigator()

const store = createStore()

const App = () => {
 React.useEffect(() => {
  SplashScreen.hide()
 }, [])

 return (
  <Provider store={store}>
   <NavigationContainer>
    <Stack.Navigator
     screenOptions={{
      headerShown: false,
     }}
     initialRouteName={'OnBoarding'}
    >
     <Stack.Screen name="Home" component={CustomDrawer} />
     <Stack.Screen name="OnBoarding" component={OnBoarding} />

     <Stack.Screen name="SignIn" component={SignIn} />

     <Stack.Screen name="SignUp" component={SignUp} />

     <Stack.Screen name="ForgotPassword" component={ForgotPassword} />

     <Stack.Screen name="Otp" component={Otp} />
    </Stack.Navigator>
   </NavigationContainer>
  </Provider>
 )
}

export default App
```

index.js

```
/**
```

```
 * @format
 */

import {AppRegistry} from 'react-native';
import App from './App';
import {name as appName} from './app.json';

AppRegistry.registerComponent(appName, () => App);
```

AuthLayout.js

```
import React from 'react'
import { View, Text, Image } from 'react-native'
import { KeyboardAwareScrollView } from 'react-native-keyboard-aware-scroll-view'

import { SIZES, images, FONTS, COLORS } from '../../constants'

const AuthLayout = ({ title, subTitle, titleContainerStyle, children }) => {
  return (
    <View style={{ flex: 1, paddingVertical: SIZES.padding, backgroundColor: COLORS.white }}>
      <KeyboardAwareScrollView
        keyboardDismissMode="on-drag"
        contentContainerStyle={{ flex: 1, paddingHorizontal: SIZES.padding }}
      >
        {/*App Icon*/}

        <View style={{ alignItems: 'center' }}>
          <Image source={images.logo_02} resizeMode="contain" style={{ width: 200, height: 100 }} />
        </View>

        {/*Title & Subtitle*/}
        <View
          style={{
            marginTop: SIZES.padding,
            ...titleContainerStyle,
          }}
        >
          <Text style={{ textAlign: 'center', ...FONTS.h2 }}>{title}</Text>
          <Text
            style={{
              textAlign: 'center',
              color: COLORS.gray,
              marginTop: SIZES.base,
              ...FONTS.body3,
            }}
          >
            {subTitle}
          </Text>
        </View>

        {/*Content / Children*/}
        {children}
```

```
    </KeyboardAwareScrollView>
   </View>
  )
}

export default AuthLayout
```

SignIn.js

```
import React from 'react'
import { View, Text, TouchableOpacity, Image, KeyboardAvoidingView } from 'react-native'

import { FONTS, SIZES, COLORS, icons } from '../../constants'
import { CustomSwitch, FormInput, TextButton, TextIconButton } from '../../components'
import { utils } from '../../utils'
import { AuthLayout } from '../'

const SignIn = ({ navigation }) => {
 const [email, setEmail] = React.useState('')
 const [password, setPassword] = React.useState('')
 const [emailError, setEmailError] = React.useState('')

 const [showPass, setShowPass] = React.useState(false)
 const [saveMe, setSaveMe] = React.useState(false)

 const isEnableSignIn = () => {
  return email !== '' && password !== '' && emailError === ''
 }

 return (
  <AuthLayout title="Let's Sign You In" subTitle="Welcome back, you've been missed">
   <View style={{ flex: 1, marginTop: SIZES.padding * 2 }}>
    {/*Form Inputs */}
    <FormInput
     label="Email"
     keyBoardType="email-address"
     autoCompleteType="email"
     onChange={(value) => {
      utils.validateEmail(value, setEmailError)
      setEmail(value)
     }}
     inputStyle={{ color: COLORS.black }}
     errorMsg={emailError}
     appendComponent={
      <View style={{ justifyContent: 'center' }}>
       <Image
        source={
         email === '' || (email !== '' && emailError === '') ? icons.correct : icons.cancel
        }
        style={{
         height: 20,
         width: 20,
```

```
        tintColor:
          email === ''
            ? COLORS.gray
            : email !== '' && emailError !== ''
            ? COLORS.green
            : COLORS.red,
        }}
      />
    </View>
  }
/>

<FormInput
  label="Password"
  secureTextEntry={!showPass}
  keyBoardType="email-address"
  autoCompleteType="password"
  containerStyle={{ marginTop: SIZES.radius }}
  inputStyle={{ color: COLORS.black }}
  onChange={(value) => {
    setPassword(value)
  }}
  appendComponent={
    <TouchableOpacity
      style={{ width: 40, alignItems: 'flex-end', justifyContent: 'center' }}
      onPress={() => setShowPass((oldState) => !oldState)}
    >
      <Image
        source={showPass ? icons.eye_close : icons.eye}
        style={{ height: 20, width: 20, tintColor: COLORS.gray }}
      />
    </TouchableOpacity>
  }
/>

{/*Save me & Forgot Password*/}
<View
  style={{
    flexDirection: 'row',
    marginTop: SIZES.radius,
    justifyContent: 'space-between',
  }}
>
  <CustomSwitch value={saveMe} onChange={(value) => setSaveMe(value)} />
  <TextButton
    label="Forgot Password?"
    buttonContainerStyle={{ backgroundColor: null }}
    labelStyle={{ color: COLORS.gray, ...FONTS.h4 }}
    onPress={() => navigation.navigate('ForgotPassword')}
  />
</View>
```

```
 {/*Sign In*/}

 <TextButton
  label="Sign In"
  disabled={!isEnableSignIn()}
  buttonContainerStyle={{
   height: 55,
   alignItems: 'center',
   marginTop: SIZES.padding,
   borderRadius: SIZES.radius,
   backgroundColor: isEnableSignIn() ? COLORS.primary : COLORS.transparentPrimray,
  }}
 />

 {/*Sign Up*/}
 <View
  style={{
   flexDirection: 'row',
   marginTop: SIZES.radius,
   justifyContent: 'center',
  }}
 >
  <Text style={{ color: COLORS.darkGray, ...FONTS.body3 }}>Don't have an account?</Text>
  <TextButton
   label="Sing Up"
   buttonContainerStyle={{
    backgroundColor: null,
    marginLeft: 3,
   }}
   labelStyle={{
    color: COLORS.primary,
    ...FONTS.h3,
   }}
   onPress={() => navigation.navigate('SignUp')}
  />
 </View>
</View>

{/*Footer*/}
<View>
 {/*Facebook*/}

 <TextIconButton
  containerStyle={{
   height: 50,
   alignItems: 'center',
   borderRadius: SIZES.radius,
   backgroundColor: COLORS.blue,
  }}
  icon={icons.fb}
  iconPosition="LEFT"
  iconStyle={{ tintColor: COLORS.white }}
```

```
        label="Continue With Facebook"
        labelStyle={{ marginLeft: SIZES.radius, color: COLORS.white }}
        onPress={() => console.log('Facebook')}
       />

       {/*Google*/}
       <TextIconButton
        containerStyle={{
         height: 50,
         marginTop: SIZES.radius,
         alignItems: 'center',
         borderRadius: SIZES.radius,
         backgroundColor: COLORS.lightGray2,
        }}
        icon={icons.google}
        iconPosition="LEFT"
        iconStyle={{ tintColor: null }}
        label="Continue With Google"
        labelStyle={{ marginLeft: SIZES.radius }}
        onPress={() => console.log('Google')}
       />
     </View>
   </AuthLayout>
  )
}

export default SignIn
```

Home.js

```
import React from 'react'
import { View, Text, TouchableOpacity, Image, TextInput, FlatList } from 'react-native'

import { SIZES, FONTS, COLORS, dummyData, icons } from '../../constants'
import { HorizontalFoodCard, VerticalFoodCard } from '../../components'

import FilterModal from './FilterModal'

const Section = ({ title, onPress, children }) => {
  return (
   <View>
    {/*Header*/}
    <View
     style={{
      flexDirection: 'row',
      marginHorizontal: SIZES.padding,
      marginTop: 30,
      marginBottom: 20,
     }}
    >
     <Text style={{ flex: 1, ...FONTS.h3 }}>{title}</Text>
     <TouchableOpacity onPress={onPress}>
```

```
        <Text style={{ color: COLORS.primary, ...FONTS.body3 }}>Show all</Text>
       </TouchableOpacity>
      </View>

      {/*Content*/}
      {children}
    </View>
  )
}

const Home = () => {
  const [selectedCategoryId, setSelectedCategoryId] = React.useState(1)
  const [selectedMenuType, setSelectedMenuType] = React.useState(1)
  const [popular, setPopular] = React.useState([])
  const [recommends, setRecommends] = React.useState([])
  const [menuList, setMenuList] = React.useState([])

  const [showFilterModal, setShowFilterModal] = React.useState(false)

  React.useEffect(() => {
    handleChangeCategory(selectedCategoryId, selectedMenuType)
  }, [selectedCategoryId, selectedMenuType])

  // Handler
  const handleChangeCategory = (categoryId, menuTypeId) => {
    // Retrieve the popular menu
    const selectedPopular = dummyData.menu.find((a) => a.name === 'Popular')

    // Retrieve the recommended menu
    const selectedRecommend = dummyData.menu.find((a) => a.name === 'Recommended')

    // Find the menu based on the menuTypeId
    const selectedMenu = dummyData?.menu.find((a) => a.id === menuTypeId)

    // Set the popular menu based on the categoryId
    setPopular(selectedPopular?.list.filter((a) => a.categories?.includes(categoryId)))

    // Set the recommended menu based on the categoryId
    setRecommends(selectedRecommend.list.filter((a) => a.categories?.includes(categoryId)))

    // Set the menu based on the categoryId
    setMenuList(selectedMenu?.list?.filter((a) => a.categories?.includes(categoryId)))
  }
  const renderSearch = () => {
    return (
      <View
        style={{
          flexDirection: 'row',
          height: 50,
          alignItems: 'center',
          marginHorizontal: SIZES.padding,
          marginVertical: SIZES.base,
```

```
        paddingHorizontal: SIZES.radius,
        borderRadius: SIZES.radius,
        backgroundColor: COLORS.lightGray2,
       }}
      >
       {/*Icon*/}

       <Image source={icons.search} style={{ height: 20, width: 20, tintColor: COLORS.black }} />

       {/*Text Input*/}

       <TextInput
        style={{
         flex: 1,
         marginLeft: SIZES.radius,
         color: COLORS.black,
         marginBottom: -5,
         ...FONTS.body3,
        }}
        placeholderTextColor={COLORS.black}
        placeholder="search food..."
       />

       {/*Filter button*/}
       <TouchableOpacity onPress={() => setShowFilterModal(true)}>
        <Image style={{ height: 20, width: 20, tintColor: COLORS.black }} source={icons.filter} />
       </TouchableOpacity>
      </View>
     )
    }

    const renderMenuTypes = () => {
     return (
      <FlatList
       horizontal
       data={dummyData.menu}
       keyExtractor={(item) => `${item.id}`}
       showsHorizontalScrollIndicator={false}
       contentContainerStyle={{ marginTop: 30, marginBottom: 20 }}
       renderItem={({ item, index }) => (
        <TouchableOpacity
         style={{
          marginLeft: SIZES.padding,
          marginRight: index === dummyData.menu.length - 1 ? SIZES.padding : 0,
         }}
         onPress={() => setSelectedMenuType(item.id)}
        >
         <Text
          style={{
           color: selectedMenuType === item.id ? COLORS.primary : COLORS.black,
           ...FONTS.h3,
          }}
```

```
        >
          {item.name}
        </Text>
      </TouchableOpacity>
    )}
   />
  )
}

const renderRecommendSection = () => {
  return (
    <Section title="Recommended" onPress={() => console.log('Show all recommended')}>
      <FlatList
        data={recommends}
        keyExtractor={(item) => `${item.id}`}
        horizontal
        showsHorizontalScrollIndicator={false}
        renderItem={({ item, index }) => {
          return (
            <HorizontalFoodCard
              containerStyle={{
                height: 180,
                width: SIZES.width * 0.85,
                marginLeft: index === 0 ? SIZES.padding : 18,
                marginRight: index === recommends.length - 1 ? SIZES.padding : 0,
                paddingRight: SIZES.radius,
                alignItems: 'center',
              }}
              imageStyle={{ marginTop: 35, height: 150, width: 150 }}
              item={item}
              onPress={() => console.log('Horizontal Food Card')}
            />
          )
        }}
      />
    </Section>
  )
}

const renderPopularSection = () => {
  return (
    <Section title="Popular Near You" onPress={() => console.log('Show all popular items')}>
      <FlatList
        data={popular}
        keyExtractor={(item) => `${item.id}`}
        horizontal
        showsHorizontalScrollIndicator={false}
        renderItem={({ item, index }) => {
          return (
            <VerticalFoodCard
              containerStyle={{
                marginLeft: index === 0 ? SIZES.padding : 18,
```

```
            marginRight: index === recommends.length - 1 ? SIZES.padding : 0,
            paddingRight: SIZES.radius,
           }}
           item={item}
           onPress={() => console.log('Vertical Food Card')}
          />
         )
        }}
       />
      </Section>
    )
}

const renderFoodCategories = () => {
  return (
    <FlatList
      data={dummyData.categories}
      keyExtractor={(item) => `${item.id}`}
      horizontal
      showsHorizontalScrollIndicator={false}
      renderItem={({ item, index }) => {
        return (
          <TouchableOpacity
            style={{
              flexDirection: 'row',
              height: 55,
              marginTop: SIZES.padding,
              marginLeft: index === 0 ? SIZES.padding : SIZES.radius,
              marginRight: index === dummyData.categories.length - 1 ? SIZES.padding : 0,
              paddingHorizontal: 8,
              borderRadius: SIZES.radius,
              backgroundColor:
                selectedCategoryId === item.id ? COLORS.primary : COLORS.lightGray2,
            }}
            onPress={() => setSelectedCategoryId(item.id)}
          >
            <Image source={item.icon} style={{ marginTop: 5, height: 50, width: 50 }} />
            <Text
              style={{
                alignSelf: 'center',
                marginRight: SIZES.base,
                color: selectedCategoryId === item.id ? COLORS.white : COLORS.darkGray,
                ...FONTS.h3,
              }}
            >
              {item.name}
            </Text>
          </TouchableOpacity>
        )
      }}
    />
  )
```

```
}

const renderDeliveryTo = () => {
 return (
  <View style={{ marginTop: SIZES.padding, marginHorizontal: SIZES.padding }}>
   <Text style={{ color: COLORS.primary, textTransform: 'uppercase', ...FONTS.body3 }}>
    Delivery to
   </Text>
   <TouchableOpacity
    style={{ flexDirection: 'row', marginTop: SIZES.base, alignItems: 'center' }}
   >
    <Text style={{ ...FONTS.h3 }}>{dummyData?.myProfile?.address}</Text>
    <Image
     source={icons.down_arrow}
     style={{ marginLeft: SIZES.base, height: 20, width: 20 }}
    />
   </TouchableOpacity>
  </View>
 )
}

return (
 <View style={{ flex: 1 }}>
  {/*Search*/}
  {renderSearch()}

  {/*Filter*/}
  {showFilterModal && (
   <FilterModal isVisible={showFilterModal} onClose={() => setShowFilterModal(false)} />
  )}

  {/*List*/}
  <FlatList
   data={menuList}
   keyExtractor={(item) => `${item.id}`}
   showsVerticalScrollIndicator={false}
   ListHeaderComponent={
    <View>
     {/*Delivery To*/}
     {renderDeliveryTo()}

     {/*Food Category*/}
     {renderFoodCategories()}

     {/*Popular*/}
     {renderPopularSection()}

     {/*Recommended*/}
     {renderRecommendSection()}

     {/*Menu Type*/}
     {renderMenuTypes()}
```

```
        </View>
      }
      renderItem={({ item }) => {
        return (
          <HorizontalFoodCard
            containerStyle={{
              height: 130,
              alignItems: 'center',
              marginHorizontal: SIZES.padding,
              marginBottom: SIZES.radius,
            }}
            imageStyle={{
              marginTop: 20,
              height: 110,
              width: 110,
            }}
            item={item}
            onPress={() => console.log('Horizontal Food Cart')}
          />
        )
      }}
      ListFooterComponent={<View style={{ height: 220 }} />}
    />
  </View>
 )
}

export default Home
```

ProductRouter.js

```
import { Router } from 'express'
import { authMiddleware } from '../middlewares/AuthMiddleware.js'
import ProductController from '../controllers/ProductController.js'

const productRouter = Router({ mergeParams: true })

productRouter.get(
 '/all',
 authMiddleware,
 ProductController.getAllProducts
)
productRouter.get(
 '/:productId',
 authMiddleware,
 ProductController.getProductById
)
productRouter.get(
  '/category/:categoryId',
  authMiddleware,
  ProductController.getProductByCategoryId
 )
```

```
productRouter.get('search/:text', authMiddleware, ProductController.findProduct)

export default productRouter
```

app.js

```
var express = require('express');
global.app = express();
global.moment = require('moment');
const expressValidator = require('express-validator');
const fileUpload = require('express-fileupload');
const cors = require('cors');
const bodyParser = require('body-parser');


// Required module
app.use(expressValidator());
app.use(cors());
app.use(fileUpload());

global.connectPool = require('./config/db.js');

global.Mails = require('./controllers/MailsController');   // Mail function
global.Common = require('./controllers/CommonsController');   // Common function
global.Auth = require('./controllers/AuthController');   // Mail function
// Constants
//global.nodeSiteUrl = 'http://192.168.1.151/constructionApp/nodeApi/'; // node
global.nodeSiteUrl = 'http://192.168.1.151:8082'; // node
global.nodeAdminUrl = 'http://192.168.1.151:8082/api/admin'; // node

global.siteUrl = 'http://192.168.1.151/constructionApp/code/api/'; // laravel
global.WebsiteURL = 'http://192.168.1.151/constructionApp/code/';  // laravel
global.fromEmail = 'info@construction-app.com'; // laravel
global.SITE_NAME = 'ConstructionApp'; // laravel
global.noImageUsers = 'http://192.168.1.151/constructionApp/code/public/upload/avtar.png';
global.noImageConstructor = 'http://192.168.1.151/constructionApp/code/public/upload/No_Image_Available.png';
global.noImageConstructor = 'http://192.168.1.151/constructionApp/code/public/upload/No_Image_Available.png';
global.noImageProduct = 'http://192.168.1.151:8082/images/No_Image_Available.png';
global.pageLimit = 10;
global.successStatus = 200;
global.failStatus = 401;
global.SessionExpireStatus = 500;
global.CustomerRole = 1;
global.ConstructorRole = 2;
global.SITE_URL = 'http://192.168.1.151/constructionApp/code/';
global.CURRENCY = '$';
global.FIREBASE_LEGACY_KEY = 'AIzaSyCODHZEPkjPGNy-X0jdLu1i9NVRFAaeumQ';
global.LOCATION_RANGE = 50;

// Notification type
global.PROJECT_NOTIFICATION_TYPE = 1;
global.MILESTONE_UPDATE_NOTIFICATION_TYPE = 2;
```

```
global.MILESTONE_PAYMENT_NOTIFICATION_TYPE = 3;
global.REVIEW_NOTIFICATION_TYPE = 4;
global.siteTitle = 'cApp Admin';


/* Admin section code */
app.set('view engine', 'ejs');
//app.set('view engine', 'pug')
var path = require('path');
app.set('views', path.join(__dirname, 'views'));
app.use(express.static(__dirname +'/public'));
var flash = require('express-flash-messages')
app.use(flash())

var cookieParser = require('cookie-parser');
var expressSession = require('express-session');
app.use(cookieParser());
app.use(expressSession({secret: 'D%$*&^lk32', resave: false,saveUninitialized: true}));
// app.use(expressSession({ cookie: { maxAge: 60000 },
//     secret: 'woot',
//     resave: false,
//     saveUninitialized: false}));

app.use(function (req, res, next) {
   res.header('Content-Type', 'application/json');
   next();
});
app.use(bodyParser.json());
app.use(express.urlencoded({limit: '100mb',extended: true }));



const nodemailer    = require("nodemailer");
global.smtpTransport = nodemailer.createTransport({
   host: 'smtp.gmail.com',
   port: 587,
   secure: false,
   auth: {
      user: 'test@octalsoftware.com',
      pass: 'octal@123'
   }
});

var apiRouter = require('./routes/api');
app.use('/api', apiRouter);
var server = app.listen(8082, function () {
   console.log("Example app listening at http://192.168.1.151:%s", server.address().port);
});
// var apiRouter = require('./routes/admin');
// app.use('/', adminRouter);
// var server = app.listen(8081, function () {
//     console.log("Example app listening at http://localhost:%s", server.address().port);
```

```
// });
process.on('uncaughtException', function (err) {
   console.log('Caught exception: ' + err);
});

// Check session of logged user
global.CheckPermission = function(req, res){
   if(typeof req.session.user !== "undefined"){
      LoginUser = req.session.LoginUser;
      if(LoginUser){
         return true;
      }else{
         res.redirect(nodeAdminUrl+'/login');
      }
   }else{
      return true;
   }
   return true;
};
```

login.ejs

```
<%- include elements/loginheader.ejs %>

<div class="main-container">
      <div class="main-content">
         <div class="row">
            <div class="col-sm-10 col-sm-offset-1">
               <div class="login-container">
                  <div class="center">
                     <h1>
                        <i class="ace-icon fa fa-leaf green"></i>
                        <span class="red">Construction </span>
                        <span class="white" id="id-text2">APP</span>
                     </h1>
                  </div>

                  <div class="space-6"></div>

                  <div class="position-relative">
                     <div id="login-box" class="login-box visible widget-box no-border">
                        <div class="widget-body">
                           <div class="widget-main">
                              <h4 class="header blue lighter bigger">
                                 <i class="ace-icon fa fa-coffee green"></i>
                                 Please Enter Your Information
                              </h4>

                              <div class="space-6"></div>

                              <form method="post" action="/api/admin/login">
                                 <input type="hidden" name="role_id" value="0">
```

```
                    <input type="hidden" name="device_token" value="1254512">
                    <input type="hidden" name="device_type" value="212121">
                    <fieldset>
                      <label class="block clearfix">
                        <span class="block input-icon input-icon-right">
                          <input type="text" name="email" class="form-control" placeholder="Username"
value="<%=data.email%>"/>
                          <i class="ace-icon fa fa-user"></i>
                        </span>
                        <span class="error"><%=errorData.email%></span>
                      </label>

                      <label class="block clearfix">
                        <span class="block input-icon input-icon-right">
                          <input type="password" name="password" class="form-control" placeholder="Password"
value="<%=data.password%>"/>
                          <i class="ace-icon fa fa-lock"></i>
                        </span>
                        <span class="error"><%=errorData.password%> </span>
                      </label>

                      <div class="space"></div>

                      <div class="clearfix">
                        <label class="inline">
                          <input type="checkbox" class="ace" />
                          <span class="lbl"> Remember Me</span>
                        </label>

                        <button type="submit" class="width-35 pull-right btn btn-sm btn-primary">
                          <i class="ace-icon fa fa-key"></i>
                          <span class="bigger-110">Login</span>
                        </button>
                      </div>

                      <div class="space-4"></div>
                    </fieldset>
                  </form>

                  <div class="social-or-login center">
                    <span class="bigger-110">Or Login Using</span>
                  </div>
                </div><!-- /.widget-main -->

                <!-- <div class="toolbar clearfix">
                  <div>
                    <a href="#" data-target="#forgot-box" class="forgot-password-link">
                      <i class="ace-icon fa fa-arrow-left"></i>
                      I forgot my password
                    </a>
                  </div>
                </div> -->
```

```
    </div><!-- /.widget-body -->
</div><!-- /.login-box -->

<div id="forgot-box" class="forgot-box widget-box no-border">
    <div class="widget-body">
        <div class="widget-main">
            <h4 class="header red lighter bigger">
                <i class="ace-icon fa fa-key"></i>
                Retrieve Password
            </h4>

            <div class="space-6"></div>
            <p>
                Enter your email and to receive instructions
            </p>

            <form>
                <fieldset>
                    <label class="block clearfix">
                        <span class="block input-icon input-icon-right">
                            <input type="email" class="form-control" placeholder="Email" />
                            <i class="ace-icon fa fa-envelope"></i>
                        </span>
                    </label>

                    <div class="clearfix">
                        <button type="button" class="width-35 pull-right btn btn-sm btn-danger">
                            <i class="ace-icon fa fa-lightbulb-o"></i>
                            <span class="bigger-110">Send Me!</span>
                        </button>
                    </div>
                </fieldset>
            </form>
        </div><!-- /.widget-main -->

        <div class="toolbar center">
            <a href="#" data-target="#login-box" class="back-to-login-link">
                Back to login
                <i class="ace-icon fa fa-arrow-right"></i>
            </a>
        </div>
    </div><!-- /.widget-body -->
</div><!-- /.forgot-box -->

<div id="signup-box" class="signup-box widget-box no-border">
    <div class="widget-body">
        <div class="widget-main">
            <h4 class="header green lighter bigger">
                <i class="ace-icon fa fa-users blue"></i>
                New User Registration
            </h4>
```

```
<div class="space-6"></div>
<p> Enter your details to begin: </p>

<form>
  <fieldset>
    <label class="block clearfix">
      <span class="block input-icon input-icon-right">
        <input type="email" class="form-control" placeholder="Email" />
        <i class="ace-icon fa fa-envelope"></i>
      </span>
    </label>

    <label class="block clearfix">
      <span class="block input-icon input-icon-right">
        <input type="text" class="form-control" placeholder="Username" />
        <i class="ace-icon fa fa-user"></i>
      </span>
    </label>

    <label class="block clearfix">
      <span class="block input-icon input-icon-right">
        <input type="password" class="form-control" placeholder="Password" />
        <i class="ace-icon fa fa-lock"></i>
      </span>
    </label>

    <label class="block clearfix">
      <span class="block input-icon input-icon-right">
        <input type="password" class="form-control" placeholder="Repeat password" />
        <i class="ace-icon fa fa-retweet"></i>
      </span>
    </label>

    <label class="block">
      <input type="checkbox" class="ace" />
      <span class="lbl">
        I accept the
        <a href="#">User Agreement</a>
      </span>
    </label>

    <div class="space-24"></div>

    <div class="clearfix">
      <button type="reset" class="width-30 pull-left btn btn-sm">
        <i class="ace-icon fa fa-refresh"></i>
        <span class="bigger-110">Reset</span>
      </button>

      <button type="button" class="width-65 pull-right btn btn-sm btn-success">
        <span class="bigger-110">Register</span>
```

```
                                <i class="ace-icon fa fa-arrow-right icon-on-right"></i>
                              </button>
                            </div>
                          </fieldset>
                        </form>
                      </div>

                      <div class="toolbar center">
                        <a href="#" data-target="#login-box" class="back-to-login-link">
                          <i class="ace-icon fa fa-arrow-left"></i>
                          Back to login
                        </a>
                      </div>
                    </div><!-- /.widget-body -->
                  </div><!-- /.signup-box -->
                </div><!-- /.position-relative -->

                <div class="navbar-fixed-top align-right">
                  <br />
                   
                  <a id="btn-login-dark" href="#">Dark</a>
                   
                  <span class="blue">/</span>
                   
                  <a id="btn-login-blur" href="#">Blur</a>
                   
                  <span class="blue">/</span>
                   
                  <a id="btn-login-light" href="#">Light</a>
                       
                </div>
              </div>
            </div><!-- /.col -->
          </div><!-- /.row -->
        </div><!-- /.main-content -->
      </div><!-- /.main-container -->

<%- include elements/loginfooter.ejs %>
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

Кваліфікаційна робота на тему:
Застосунок для автоматизації роботи кав’ярень з реалізацією сервісу бухгалтерського обліку

Виконав:
студент 3 курсу, група ІПЗс-20-1
Балагур Андрій Ігорович

Керівник:
Доцент, кандидат педагогічних наук
Онишко О.Г.

Актуальність теми

У сучасному світі робота в кафе стає все більш популярною, і водночас зростає потреба в ефективному управлінні бізнесом. Одним із способів покращити роботу кав’ярні є використання програмного забезпечення для автоматизації повсякденних операцій і забезпечення точного запису даних.

Основні переваги автоматизації процесу роботи кав'ярень включають ефективнішу організацію робочих процесів, оптимізацію ресурсів, зменшення ризиків помилок у бухгалтерському обліку та збільшення швидкості обробки інформації.

Крім того, розробка додатку може стати конкурентною перевагою для підприємства на ринку та забезпечити зручний та швидкий доступ до необхідної інформації про фінансовий стан закладу.

У зв'язку з цим, тема дипломного проекту може бути актуальною та має потенціал для практичного використання. Однак, для того, щоб розроблений додаток був успішним, важливо врахувати потреби та очікування кав'ярень та їх клієнтів.

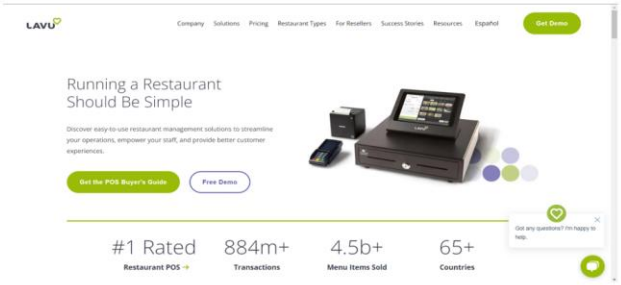
Мета і завдання проекту

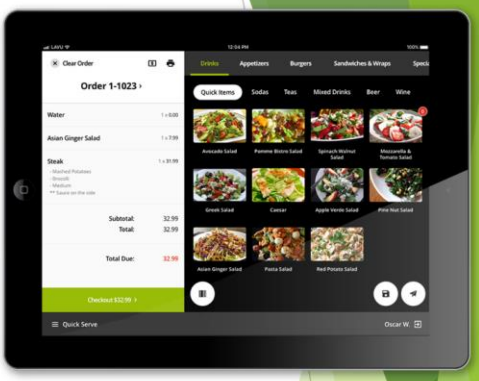
Метою проектування є розробка повнофункціонального програмного забезпечення для оптимізації роботи кав'ярень, скорочення часу виконання щоденних операцій та забезпечення точного обліку даних, що сприятиме ефективному управлінню бізнесом.

Потрібно виконати наступні завдання:

- провести аналіз предметної області та виявити її особливості;
- провести аналіз ринку на наявність готового вирішення поставленої проблеми дипломного проекту;
- обрати стиль серверної архітектури розроблювальної платформи;
- обрати інструменти та технології для розробки, які зможуть задовільнити поставлені вимоги;
- реалізувати серверну та клієнтську частини платформи;
- написати інструкцію користування платформою;
- розписати процес запуску;
- протестувати готовий застосунок.

Lavu

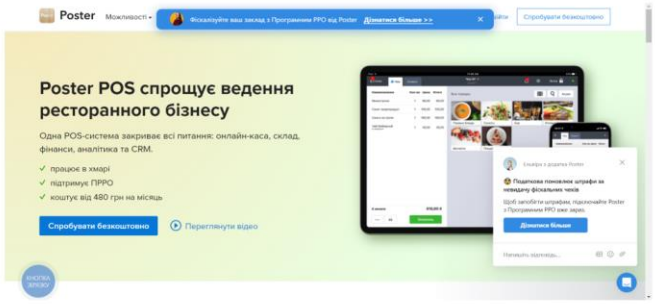


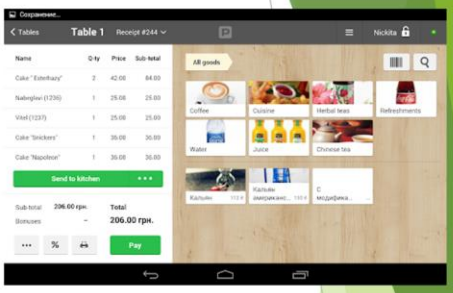


Серед недоліків можна виділити:

- скарги користувачів на затримки у випуску оновлень та недостатню якість підтримки з боку Lavu. Це може вплинути на вирішення проблем та доступ до нових функцій.
- хоча Lavu працює на різних пристроях, включаючи планшети та смартфони, вона може бути обмеженою в підтримці певних моделей та версій пристроїв.застосунок має розширений набір функцій і може вимагати деякого часу та зусиль, щоб повністю оволодіти всіма можливостями та налаштуваннями додатку.
- Вартість пакетів починається від $69 на місяць, що може перевищувати бюджети деяких підприємств.

JoinPoster

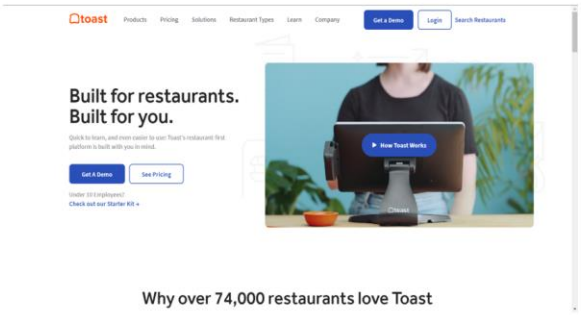


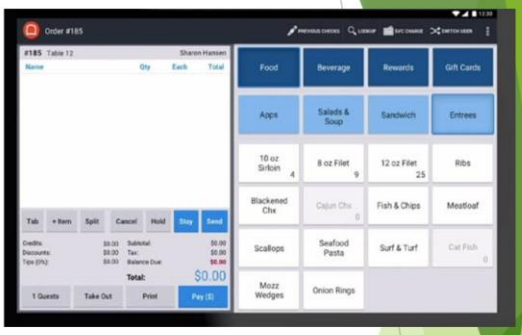


Має ряд недоліків:

- витрати на використання додатку можуть бути значними для певних ресторанів з обмеженим бюджетом;
- його користувачі часто повідомляють про виникнення помилок або непередбачувану поведінку Joinposter.
- застосунок має розширений набір функцій і може вимагати деякого часу та зусиль, щоб повністю оволодіти всіма можливостями та налаштуваннями додатку.

Toast POS





Серед недоліків можна виділити те, що Toast POS відноситься до преміум-класу ресторанних ПЗ і може бути досить дорогим для менших підприємств з обмеженим бюджетом.

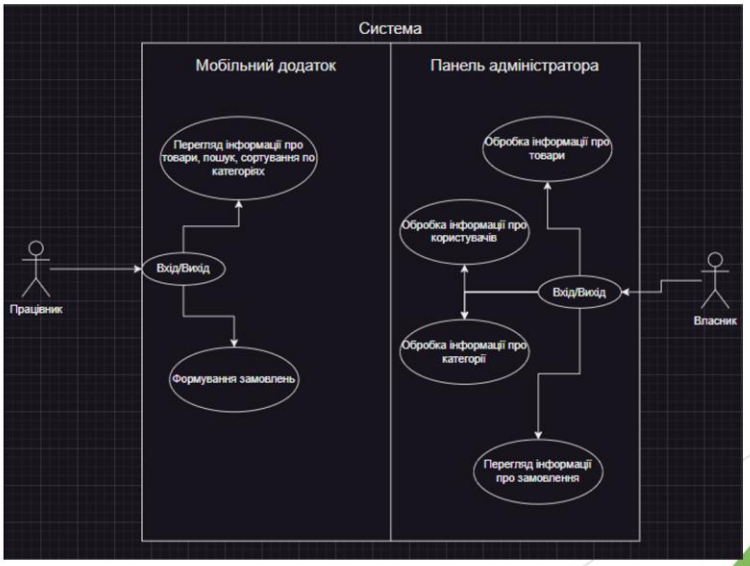
Також інтеграція та налаштування застосунку можуть бути складними, особливо для користувачів з обмеженим досвідом в області технологій.

Toast POS може мати обмежену сумісність та інтеграцію з іншими системами, що використовуються в ресторанному бізнесі, такими як системи управління запасами або системи лояльності.

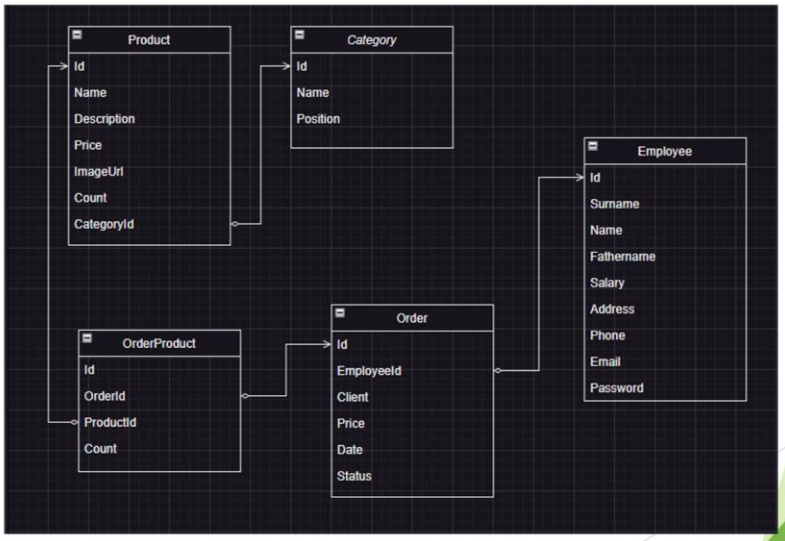
Порівняння наявного ПЗ

Характеристика	Lavu	JoinPoster	Toast POS
Платформи	iOS, Android	iOS, Android	iOS, Android
Ціна	Вартість пакетів починається від $69/місяць	Ціни варіюються в залежності від потреб бізнесу	Інформація про ціну надається виробником
Керування запасами	Так	Так	Так
Замовлення доставки	Так	Так	Так
Інтеграція зі сторонніми системами	Так	Так	Так
Спеціалізовані функції	Керування баром та кухнею	Резервування столиків та часовий графік персоналу	Управління клієнтськими замовленнями та доставкою
Підтримка мов та валют	Так	Так	Так
Користувацький інтерфейс	Зручний та інтуїтивний	Сучасний та простий у використанні	Інтуїтивний та простий у використанні
Можливості розширення	Можливість інтеграції з багатьма сторонніми програмами та сервісами	Вибір з різних модулів та додаткових функцій	Гнучкі настройки знижок та акцій, а також інтеграція з іншими системами

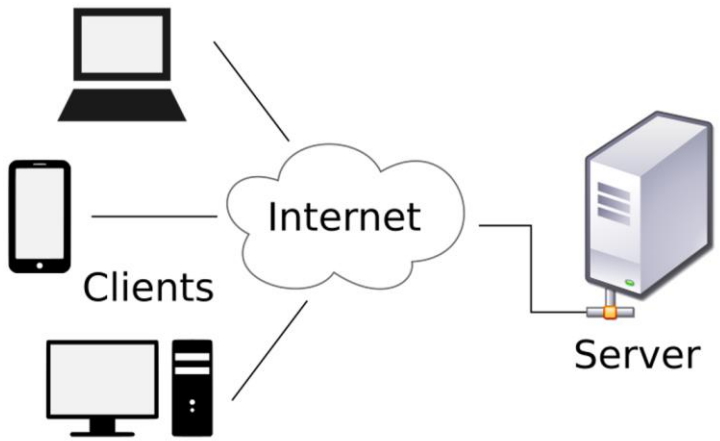
Діаграма варіантів використання



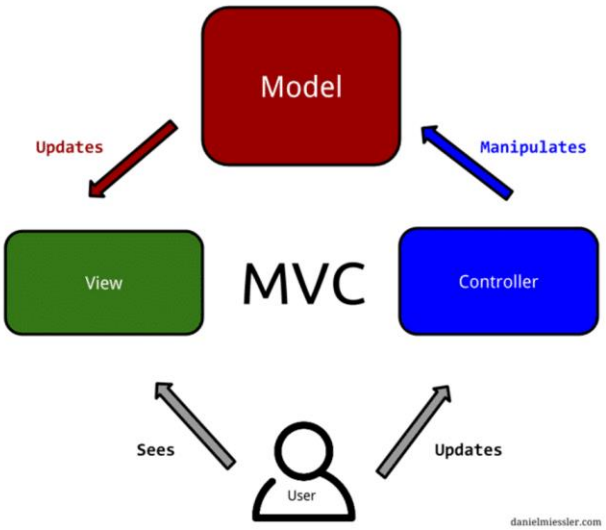
Модель бази даних



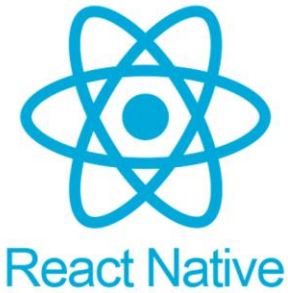
Клієнт-серверна архітектура



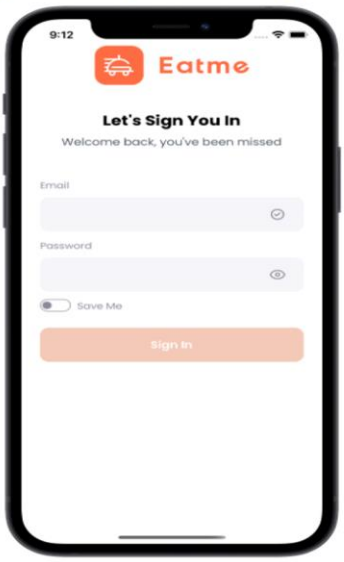
Model View Controller



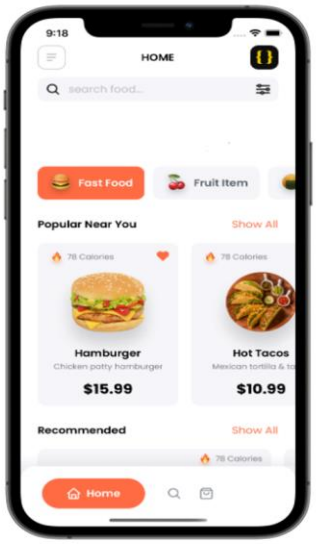
Інструменти та технології для реалізації



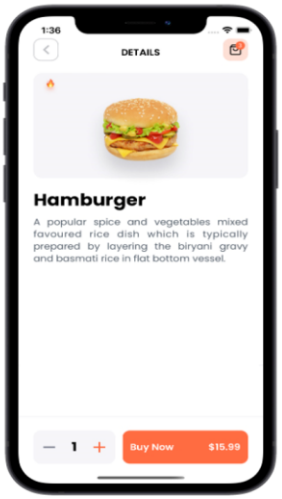
Реалізація проекту. Авторизація



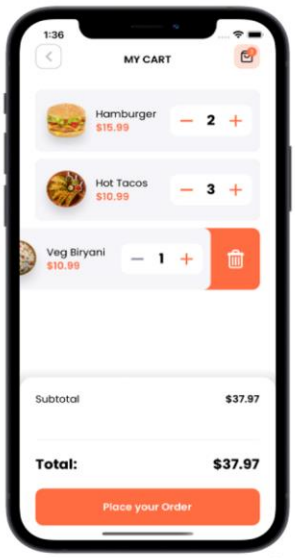
Головна сторінка застосунку



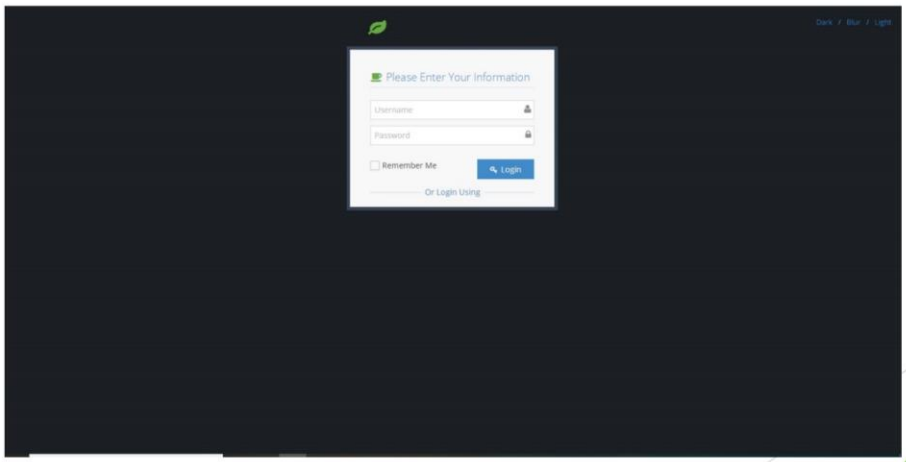
Реалізація проекту. Детальна сторінка товару



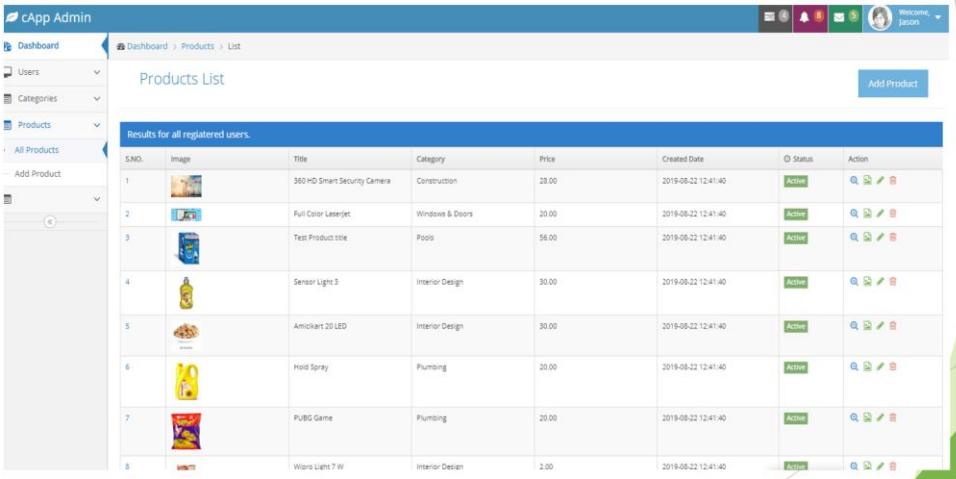
Реалізація проекту. Сторінка кошику



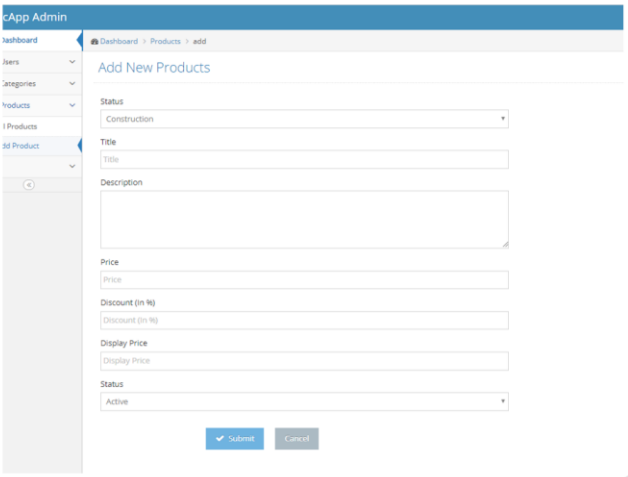
Реалізація проекту. Сторінка авторизації в адмін - панелі



Реалізація проекту. Сторінка списку товарів



Реалізація проекту. Сторінка створення товару



Висновки

Протягом даної кваліфікаційної роботи було спроектовано і розроблено застосунок для автоматизації кав'ярень. Для розробки додатку були використані такі технології як PostgreSQL, Node.js, Express.js, Admin.js та React Native.

Було проведено аналіз вимог та функціональності, які повинен містити розроблений додаток, а також була спроектована база даних, створено адмін-панель, серверну та клієнтську частину додатку

Застосування клієнт-серверної архітектури дозволило забезпечити ефективне управління базою даних та надійну роботу додатку. Використання PostgreSQL дозволило зберігати та обробляти дані про замовлення, клієнтів та складський облік, а Node.js та Express.js допомогли реалізувати серверну частину додатку та забезпечити взаємодію з базою даних.

Також патерн MVC дозволив реалізувати адмін-панель.

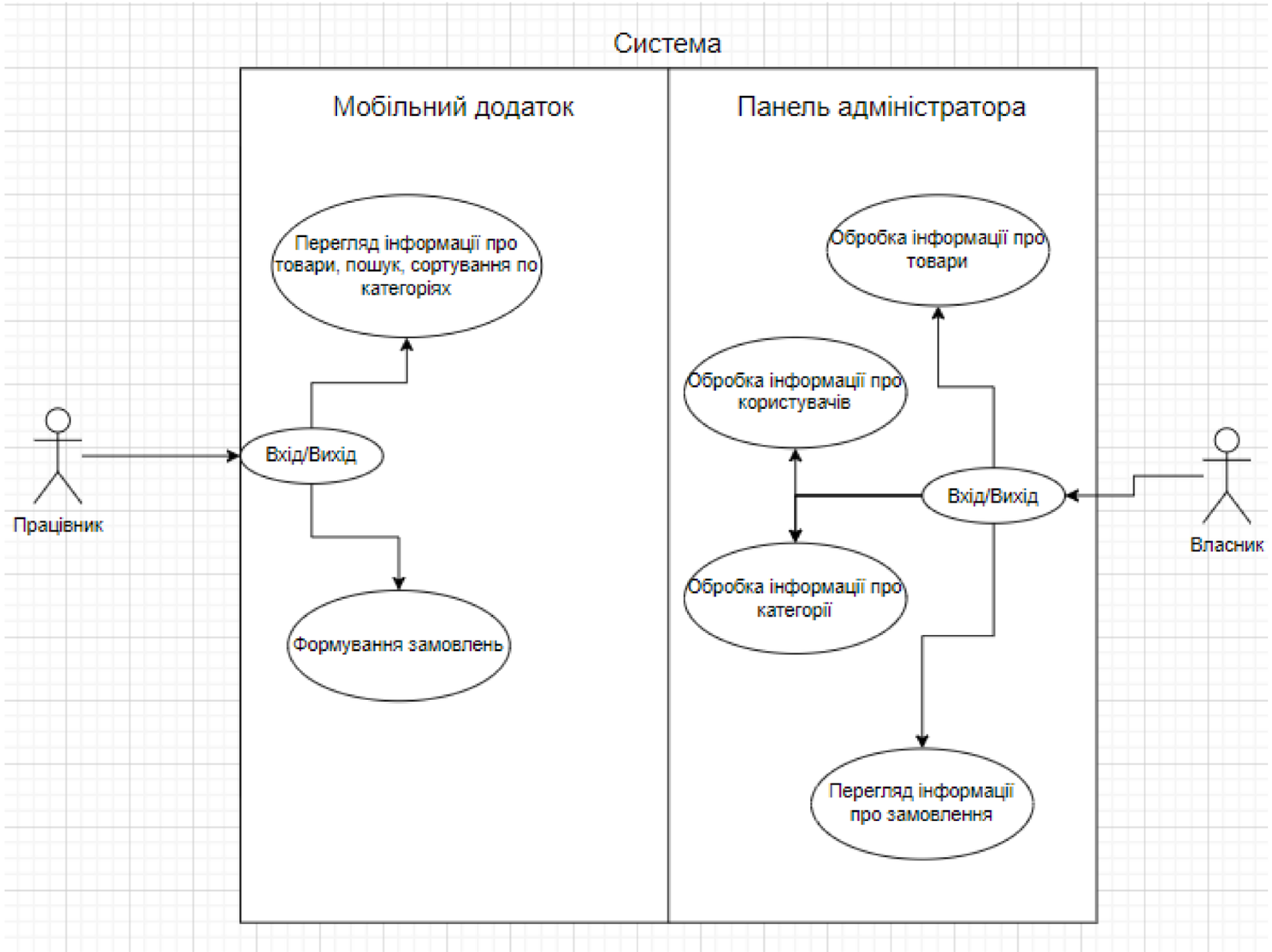
React Native було використано для розробки клієнтської частини додатку. Використання цієї технології дозволило створити мобільний додаток з нативним виглядом та взаємодією з платформою, що забезпечує високу продуктивність та надійність додатка.

Даний продукт розроблений і вже може використовуватись клієнтами.


Дякую за увагу!

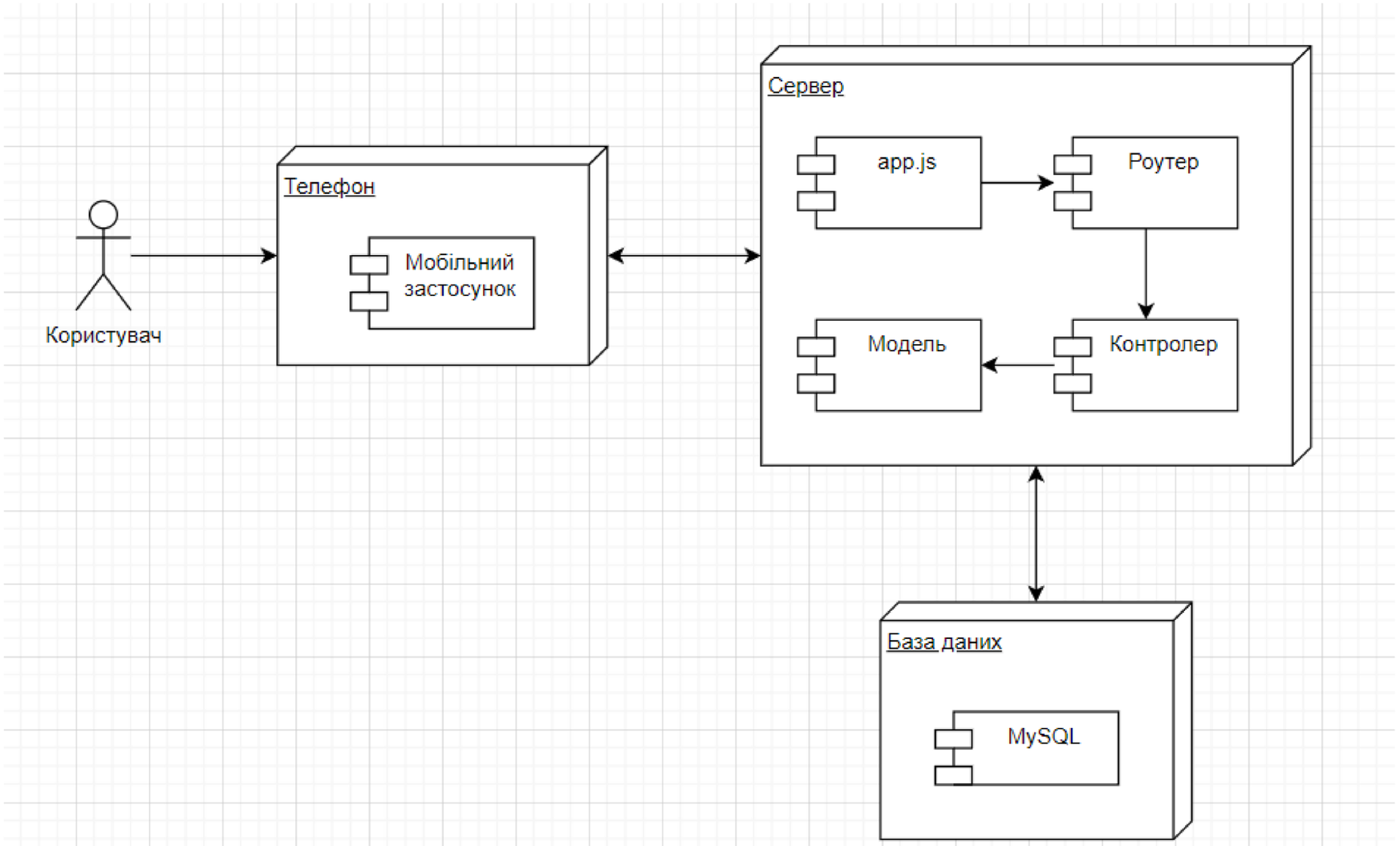
ГРАФІЧНА ЧАСТИНА

Рисунок 1 - Діаграма варіантів використання



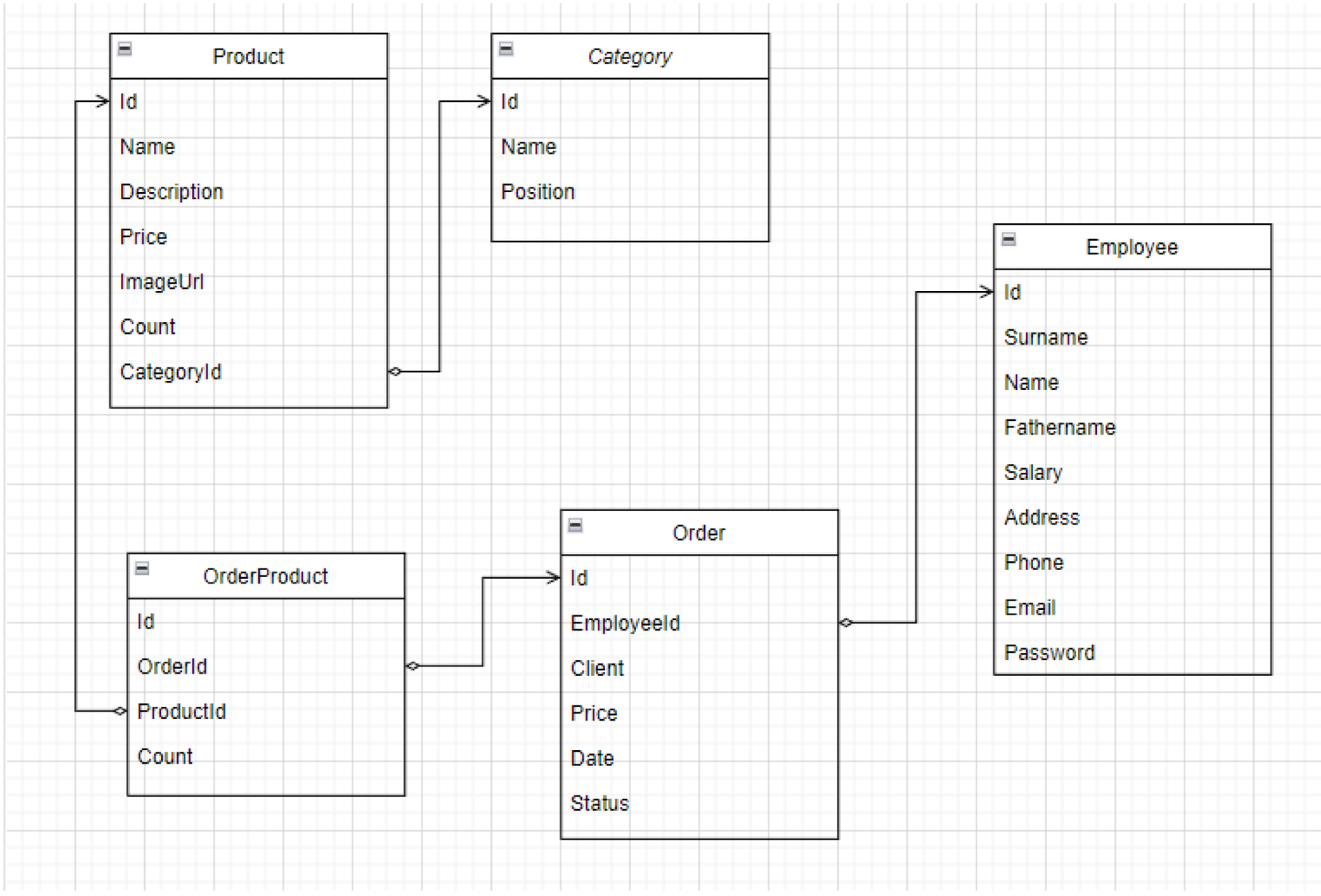
					КвРІП3.200119.01.02.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	USE-CASE Діаграма Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку	Літера / Маса / Масштаб
Розробив		Балагур А.І.		22.05.23		
Керівник		Онишко О.Г.		22.05.23		Аркуш 1 / Аркушів 3
Консульт.						
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІП3с-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

Рисунок 2 - Діаграма розгортання



					КвРІП3.200119.01.02.Е8	
					Deployment- діаграма Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку	Літера / Маса / Масштаб
Зм.	Арк.	№докум.	Підпис	Дата		
Розробив		Балагур А.І.		22.05.23		Аркуш 2 / Аркушів 3
Керівник		Онишко О.Г.		22.05.23		
Консульт.						ХНУ, ІПЗс-20-1
Н.Контр.		Гурман І.В		22.05.23		
Зав.каф.		Бедратюк Л.П.		22.05.23		

Рисунок 3 - Структура бази даних



					КвРІП3.200119.01.02.Е8	
Зм.	Арк.	№докум.	Підпис	Дата	Схема БД Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку	Літера / Маса / Масштаб
Розробив		Балагур А.І.		22.05.23		
Керівник		Онишко О.Г.		22.05.23		Аркуш 3 / Аркушів 3
Консульт.						
Н.Контр.		Гурман І.В		22.05.23		ХНУ, ІПЗс-20-1
Зав.каф.		Бедратюк Л.П.		22.05.23		

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи ІПЗс-20-1
Балагура А.І.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку

(керівник роботи – Онищко Оксана Григорівна)
Прізвище, ім'я, по батькові

05.02.2023
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти

Балагура А.І.

Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-20-1

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності в Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту і застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку й збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та/або Anti-Plagiarism) і використання роботи для виявлення академічного плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

30.05.2023

дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Балагур Андрій Ігорович
Прізвище, ім'я, по батькові

студент III курсу факультету інформаційних технологій, кафедри інженерії
здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/ науково-педагогічний працівник (назва кафедри)

програмного забезпечення
назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням про систему забезпечення академічної доброчесності у Хмельницькому національному університеті, законодавства України.

«_05_» __лютого_ 2023 р.

_____
Підпис

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 6.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 10%**

	Документ		Сумарний збіг по Базі Даних	
ID: 114575 Назва: БКР Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку Додано в БД: 2023-06-02 Автора: Балагур А.І. Керівники: Онишко О.Г. к.п.н. доц. Консультанти: Опоненти:	Символи	Лексеми	Символи	Лексеми
	68377	641	6875 (10%)	64 (10%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

ID перевірки:
1015392921

Дата перевірки:
02.06.2023 13:30:41 EEST

Тип перевірки:
Doc vs Internet + Library

Дата звіту:
02.06.2023 13:32:51 EEST

ID користувача:
100005589

Назва документа: **КвР_Балагур_ІПЗс20-1_Антиплагіат**

Кількість сторінок: **62** Кількість слів: **10629** Кількість символів: **84610** Розмір файлу: **3.47 MB** ID файлу: **1015057356**

11.4%
Схожість

Найбільша схожість: **3.9%** з джерелом з Бібліотеки (ID файлу: **1015017362**)

7.64% Джерела з Інтернету	671	Сторінка 64
8.28% Джерела з Бібліотеки	118	Сторінка 68

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

РІШЕННЯ ЕКСПЕРНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: «Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку»
Автор: Балагур Андрій Ігорович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Онишко Оксана Григорівна, кандидат педагогічних наук, доцент
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої й електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того, як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті кваліфікаційної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, відомість документів), у структурі змісту, назвах розділів/підрозділів тощо, у назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посиланання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені модифікації тексту не впливають на відсоток схожості.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 11,4% і адресується до 789 джерел, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь кваліфікаційної роботи.

Завідувач кафедри _____ Леонід БЕДРАТЮК

Гарант освітньої програми _____ Леонід БЕДРАТЮК

Керівник кваліфікаційної роботи _____ Оксана ОНИШКО

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Балагур Андрій Ігорович

Тема Застосунок для автоматизації роботи кав'ярень з реалізацією сервісу бухгалтерського обліку

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень _____; кількість сторінок записки _____

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було досліджено і проаналізовано предметну область, визначено усі функціональні та нефункціональні вимоги. Був проведений аналіз існуючих застосунків на ринку, розглянуто їх переваги і недоліки, та виведено актуальність розробки нового програмного забезпечення. Розглянуто інструменти для реалізації спроектованих рішень, в результаті чого створено програмне забезпечення. Також було проведено тестування програми, за результатами якого було виявлено, що розроблене програмне забезпечення працює коректно та готове до експуатації.

2. Висновок про відповідність роботи поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням всіх вимог.

3. Характеристика виконання кожного розділу роботи, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання дипломного проектування. У першому розділі проведено аналіз предметної області, розглянуто існуючі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного програмного забезпечення. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. Крім того було виконано тестування системи та проведено його у відповідності до функціональних вимог, в результаті було підтверджено коректну роботу застосунку.

4. Позитивні сторони роботи Тематика кваліфікаційної роботи є актуальною, оскільки на сьогодні на ринку немає достатньо зручних та функціональних застосунків аналогів. Також було застосовано новітні технології для побудови програмного продукту та актуальні архітектурні рішеня.

5. Негативні сторони роботи У роботі було використано достатньо складні в супроводжені технології. Безпеці передачі файлів медіа матеріалів, в межах системи, не було приділено належної уваги.

6. Оцінка графічного оформлення та пояснювальної записки Графічне оформлення виконано відповідно до теми кваліфікаційної роботи та подано у вигляді діаграм і рисунків. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Матеріал пояснювальної записки структурований, послідовний, чіткий та простий, що дозволяє чітко зрозуміти викладений матеріал у рамках тематики проектування. Графічний матеріал дає можливість наочно побачити деталі проектування системи.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ професор кафедри комп'ютерної інженерії та інформаційних систем, д.т.н., професор Лисенко Сергій Миколайович

"02" червня 2023 р. (підпис)