Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»

Кафедра інформатики факультету інформатики

АВТОМАТИЗОВАНА ІНФОРМАЦІЙНА СИСТЕМА У СФЕРІ ОБСЛУГОВУВАННЯ

Текстова частина до курсової роботи

за спеціальністю „Інженерія програмного забезпечення”

Керівник курсової роботи
Яремко С.А.

_____

“6” червня 2022 р.

Виконала студентка
Гуза М.-В.М.
“6”червня 2022 р.

Київ – 2022

Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Зав.кафедри інформатики,
доцент, кандидат наук
_____С. С. Гороховський

„____”_____2022 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці Гуза М.-В.М. факультету інформатики 3 курсу
ТЕМА Автоматизована інформаційна система у сфері обслуговування

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1. Аналіз предметної області.
2. Проектування системи.
3. Реалізація системи.
Висновки
Список літератури
Додатки

Дата видачі „___” _____ 2021 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Календарний план виконання курсової роботи

Тема: Автоматизована інформаційна система в сфері обслуговування

Календарний план виконання роботи:

№ п/п	Назва етапу дипломного проекту (роботи)	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу	02.11.2021	
2.	Виконати аналіз предметної області	1.04.2022	
3.	Проектування системи	1.05.2022	
4.	Програмування розробленої системи	22.05.2022	
5.	Написання текстової частини роботи	6.06.2022	
6.	Попередня здача курсової роботи	6.06.2022	
7.	Захист курсової роботи	13.06.2022	

Студентка Гуза М.-В. Михайлівна

Керівник Яремко Соломія Андріївна

“__” _____

Зміст

Перелік прийнятих скорочень

АІС – автоматизована інформаційна система;

СКБД – система управління базами даних;

HTML – HyperText Markup Language;

CSS – Cascading Style Sheets;

ER-модель – Entity-Relationship-модель

Анотація

У даній курсовій роботі передбачається розробка веб-застосування для квіткового магазину та створення автоматизованої інформаційної системи для обробки, аналізу та управляння даними. Веб-застосунок написаний на JavaScript, а саме Node.js з використанням додаткових бібліотек та фреймворків, таких як React, Redux, Bootstrap 5, Express.js з підключенням до бази даних PostgreSQL.

Вступ

У сучасному світі дуже актуальне питання переміщення будь-якого бізнесу, в тому числі в сфері обслуговування на онлайн платформи. В нинішні часи люди віддають перевагу створенням покупок онлайн для економії свого часу. Також пошук товарів в інтернеті значно розширює перелік доступних варіантів продукту, що забезпечує користувачам завжди знаходити бажане.

Одним з рішень цих проблем є створення веб-застосунку з інтерфейсом для користувача, який передбачає можливість перегляду асортименту магазину та з панеллю управління для адміністратора та співробітника, яка допомагає контролювати перелік товарів, здійснювати облік та аналіз даних. Цей веб-сайт значно полегшить ведення невеликого бізнесу та допоможе привернути увагу більшої кількості клієнтів. Також даний застосунок може підійти для будь-якої сфери, так як має в своїй основі доволі універсальну модель бази даних.

Метою курсової роботи є створення автоматизованої інформаційної системи у сфері обслуговування, а саме у сфері флористики. Ця АІС надає можливість переглядати наявний товар, робити замовлення онлайн, переглядати старі замовлення тощо. Для власників бізнесу ця система допоможе контролювати дані за допомогою нескладної панелі управління, переглядати та аналізувати всі необхідні дані.

Основним завданням роботи є дослідження та аналіз існуючих веб-застосунків для того щоб виявити, який дизайн найбільше всього приваблює клієнтів та дослідити основні помилки існуючих рішень. Ще одним важливим завданням є проектування універсальної системи, яка забезпечить можливість зберігати всі необхідні дані.

Об’єктами досліджень роботи є існуючі веб-застосунки, які передбачають продаж товарів, бібліотеки Node.js, а саме React, Express.js, Redux тощо, використання СКБД PostgreSQL.

Робота складається з трьох основних розділів – аналіз предметної області та її актуальності, опис, проєктування та реалізація системи. Перший розділ містить аналіз існуючих веб-сайтів у сфері флористики та аналіз опитування потенційних користувачів. Другий розділ описує створену систему та основні її компоненти. Третій розділ це програмна реалізація спроектованої системи, опис інтерфейсу та мапа сайту.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1.Призначення системи

Дана автоматизована інформаційна система призначена для упорядкування, обробки та аналізу даних у сфері обслуговування, а саме для квіткового магазину.

Основна ціль цієї АІС зробити ведення бізнесу більш простим, надати покупцям зручний сайт, де вони можуть переглянути актуальний перелік товарів та робити замовлення онлайн.

1.2.Обгрунтування актуальності системи

1.2.1. Опитування кінцевих користувачів

Перед початком розробки даної АІС для квіткового магазину, було вирішено провести опитування серед потенційних клієнтів для того, щоб зрозуміти актуальність майбутнього продукту. В цьому опитуванні взяли участь люди різного віку від 18 до 62. Було задано ряд питань, а саме – як часто ви купуєте квіти, де ви їх купуєте, чи користувались би ви сайтом для квіткового магазину тощо. Проаналізувавши відповіді людей, було зроблено висновок, що в основному люди купують квіти на свята, тобто найбільший потік клієнтів буде саме в ці дні, що люди частіше купують квіти в магазинах, рідше в соціальних мережах та майже ніколи не користуються веб-сайтами з квітами.

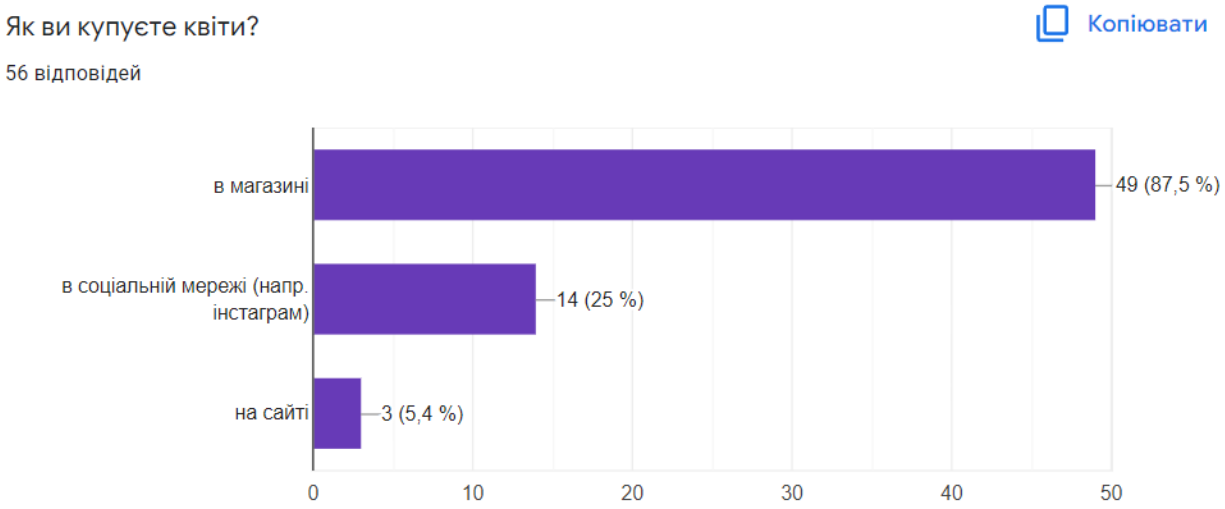

Рисунок 1.1 Де люди купують квіти

Цікаво було побачити, що більша частина респондентів користувались би веб-сайтом квіткового магазину і робили там замовлення, якщо такі сайти існували б. Більше ніж 80% опитуваних вважають, що квітковому магазину потрібен сайт та вони б ним користувались.

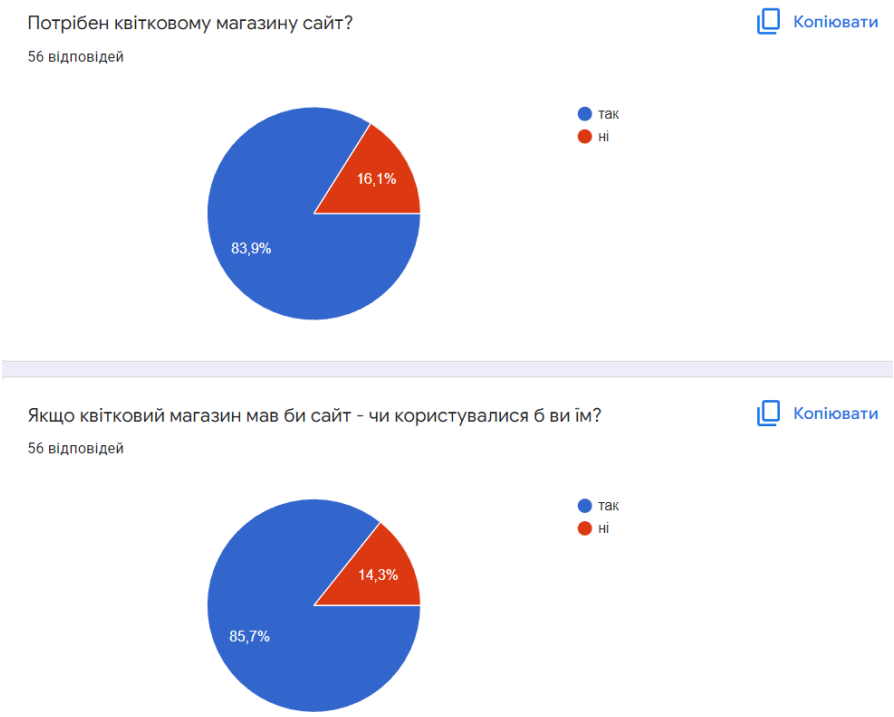

Рисунок 2.2 Актуальність квіткового сайту

Але було декілька відповідей про те, що веб-сайти часто бувають незручними і що вони купують квіти тільки наживо.

Також було проведено невелике опитування людини, яка починала розробляти свій квітковий бізнес, тому дуже важливо проаналізувати її відповіді і оцінити актуальність даної системи не тільки для користувачів, а й для власників бізнесу. З її слів зрозуміло, що цей веб-сайт був би дуже корисний у роботі, так як це дуже сильно спростило їй ведення обліку замовлень, оцінки роботи флористів. Також цей сайт надав би зручний та простий інтерфейс не тільки власнику і співробітникам, але й дуже сильно допомогло клієнтам орієнтуватись в актуальних товарах та цінах на нього.

Проаналізувавши відповіді потенційних клієнтів, можна зробити висновок, що сайт для квіткового магазину – тема дуже актуальна, тому що зараз на ринку майже нема зручних подібних сайтів. Власники бізнесу вимушені користуватись excel для ведення свого обліку, або ж записувати все вручну, а користувачі незадоволені якістю існуючих сайтів, через що майже ніколи ними не користуються, але з їх відповідей очевидно, що якщо б був гарний аналог квіткового веб сайту, то вони б залюбки ним користувались.

1.2.2. Аналіз існуючих веб-сайтів

Для того, щоб остаточно впевнитись в актуальності майбутнього веб-сайту, було проаналізовано вже існуючі.

Головним недоліком всіх сайтів є застарілий та поганий дизайн, неякісні фотографії, що одразу відштовхує майбутніх клієнтів. Одна з перших речей, на які дивиться клієнт, коли заходить на сайт – це те, як він виглядає, а вже потім на асортимент. Тому дуже важливо одразу привернути увагу клієнта дизайном.

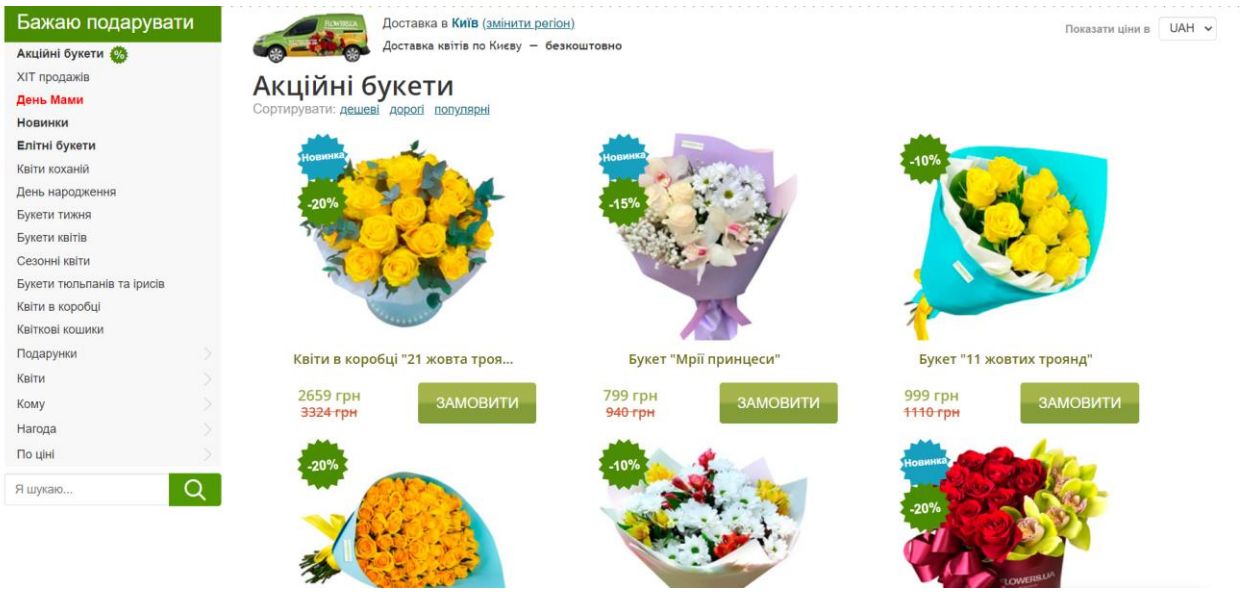

Рисунок 1.3 Приклад сайту з поганим дизайном

Однак також зустрілось декілька сайтів з гарним дизайном та зручним інтерфейсом. В основному на таких сайтах можна було реєструватись, сортувати та фільтрувати товари, переглядати детальну інформацію про нього, виконувати пошук по сайту та переглядати схожі пропозиції. Тобто зручні сайти для квіткових магазинів є, але вони не дуже популярні і не відомо, чи мають зручний інтерфейс для адміністратора та продавця.

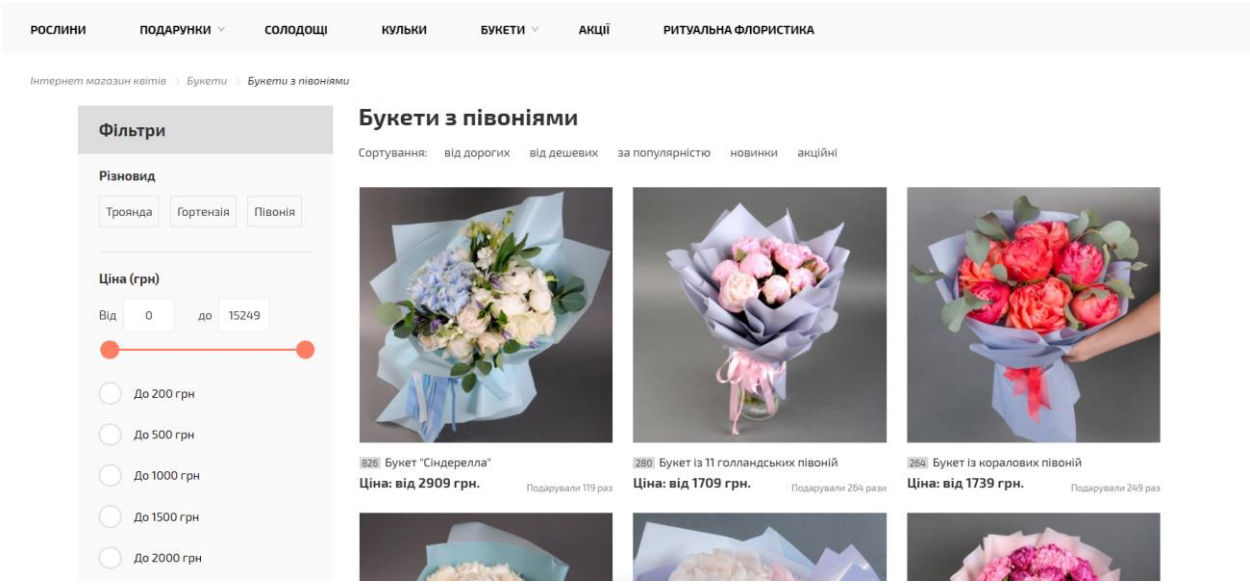

Рисунок 1.4 Приклад сайту з гарним дизайном

Також досліджуючи ринок флористики, було помічено, що в основному квіткові магазини віддають перевагу веденню соціальних мереж, наприклад Instagram, де вони продають і рекламують свої товари. Зрозуміло, що цей спосіб дозволяє ділитись актуальними та свіжими фотографіями букетів, показувати фотографії клієнтів, тобто вести цілий блог свого магазину. Але саме такий формат основної сторінки може бути проблемою для магазину, тому що клієнт не має можливості побачити всі квіти в наявності та їх актуальну ціну, не може здійснювати пошук бажаного виду квітів та фільтрувати товари. Для того, щоб зробити замовлення в такому магазині, клієнту потрібно звертатися до менеджера, а дочекавшись на відповідь запитувати про товари в наявності, уточнювати ціну та варіанти пакування. В той час як менеджеру потрібно швидко відповідати клієнту, вручну заповнювати замовлення, самому відслідковувати кількість товару, що залишився та багато іншого.

РОЗДІЛ 2. ОПИС СПРОЕКТОВАНОЇ СИСТЕМИ

2.1. Типи користувачів та їх опис

Користувачами даного веб-сайту є адміністратор/власник бізнесу, флорист/продавець та клієнт.

- Адміністратор

Адміністратор повністю контролює роботу бізнесу. Має можливість переглядати всі дані про товар, додавати/редагувати/видаляти товари, наймати та звільняти продавців. Переглядати різні статистичні дані, такі як

прибуток за місяць, найбільш популярні квіти, всі замовлення за певний період тощо. Отримує сповіщення про закінчення товару.

- Продавець/флорист

Продавець має можливість додавати, редагувати товар. Може власноруч додавати замовлення, якщо воно було зроблено офлайн. Отримує сповіщення про нове замовлення, яке він має прийняти, обробити, виконати. Може виконувати списання товару, якщо з ним щось сталося.

- Клієнт

Клієнт може переглядати весь товар, який є в наявності. Може шукати товар за фільтрами/пошуком/категоріями. Має можливість зареєструватися на сайті для перегляду статусу замовлень. Може бачити кількість товару в наявності, додавати товар у кошик та робити замовлення. Отримує сповіщення про статус його замовлення.

2.2. Вимоги до даних

- Співробітник

В одному філіалі може працювати декілька співробітників – флористів. Про них зберігаються такі дані – табельний номер, прізвище, ім’я, по-батькові (може бути відсутнім), телефони, електронна адреса, адреса проживання (місто, вулиця, номер будинку, опціонально – номер квартири), зарплатня та дата прийому на роботу

- Клієнт

За бажанням клієнт може зареєструватися в системі (самостійно або за допомогою співробітника), вказавши своє прізвище, ім’я, по-батькові, номер телефону та електронну адресу.

- Замовлення

Замовлення може бути зареєстровано продавцем, якщо покупка була виконана в магазині, або ж замовлення може бути створене безпосередньо на сайті клієнтом. Під час замовлення офлайн, клієнт може вказати свої дані, якщо він вже був зареєстрований в системі.

Кожне замовлення має свій номер, дату створення, адрес доставки, якщо замовлення потрібно кудись відправити, вид пакування (буде наданий перелік), кінцева вартість замовлення та його стан (в процесі, виконується, доставляється, готове, виконано).

- Одиниця замовлення

Замовлення може складатись з декількох товарів. Вказується кількість певного товару.

- Товар

В квітковому магазині продаються різні товари, але в основному квіти. Кожен товар має свій унікальний номер, назву, детальний опис (якщо це готовий букет – описується склад букету, розмір тощо), фотографії, ціну, за яку товар був куплений та кінцеву ціну і опціонально можна вказати колір. Кожен товар може бути в наявності в різних філіалах в різній кількості.

- Категорія

Кожен товар належить певній категорії (можливо й не одній) для покращення пошуку. Категорія має свій унікальний номер та назву.

2.3. Вимоги до роботи з даними

- Можливості адміністратора:
    1. Додати, оновити, переглянути або видалити інформацію про співробітника.
    2. Додати, оновити, переглянути або видалити інформацію про покупця.
    3. Додати, оновити, переглянути або видалити інформацію про товари.

4. Додати, оновити, переглянути або видалити інформацію про філіали.
5. Додати, оновити, переглянути інформацію про категорії.
6. Різні статистичні дані.

- Можливості продавця/флориста
  1. Створити нове замовлення, змінити інформацію в замовленні (наприклад, стан, адресу замовлення), видалити замовлення.
  2. Додати, оновити, переглянути або видалити інформацію про покупця.
  3. Додати, оновити, переглянути інформацію про товар.
  4. Додати або переглянути інформацію про категорію.
  5. Різні статистичні дані.

- Можливості клієнта
  1. Зареєструватись в системі.
  2. Отримати інформацію про будь-який товар, який є в наявності.
  3. Додати, оновити, переглянути або видалити замовлення.
  4. Шукати товар за категорією, кольором, назвою тощо.
  5. Відфільтрувати товар за ціною, назвою, популярністю.
  6. Видалити свій профіль.

2.4. ER-модель

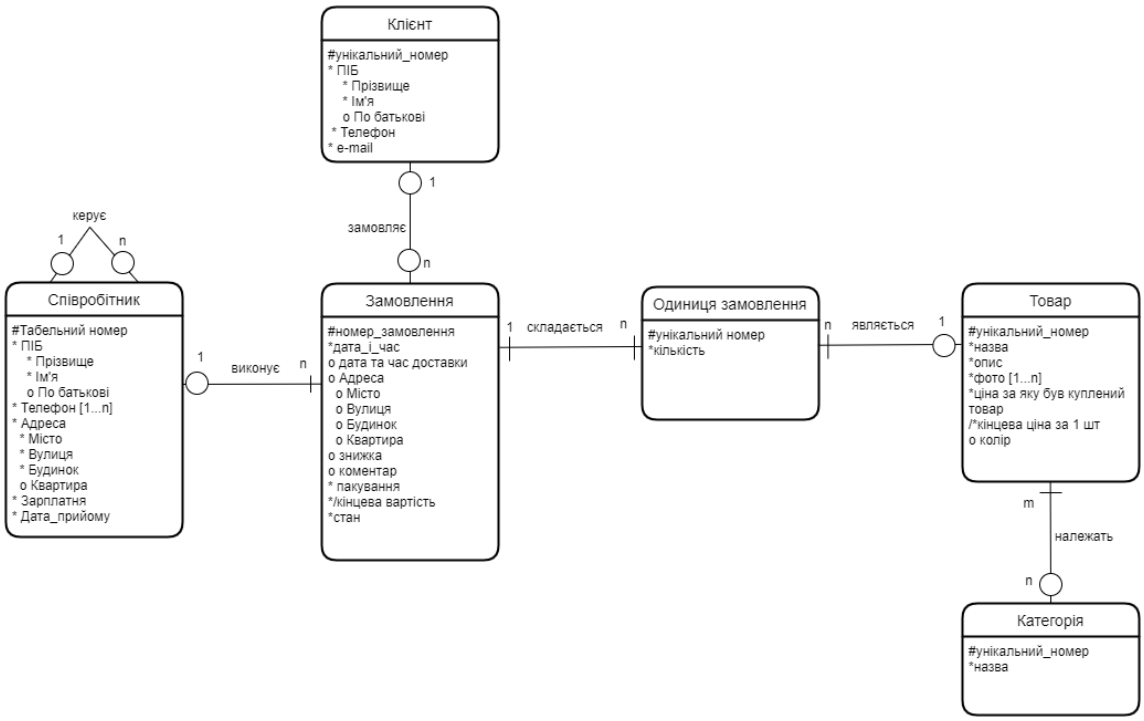

Рисунок 2.1 ER-модель

2.5. Реляційна модель

Див. Додаток В.

2.6. Нефункціональні вимоги

- Швидкість

Основною вимогою до розробленого веб-сайт є – швидке відображення даних. Так як основним контентом нашого сайту є товари, інформація про які зберігається в базі даних, необхідно забезпечити їх швидке виведення на екран, так як навряд чи майбутні клієнти будуть чекати поки сторінка завантажиться. Якщо ж брати панель адміністратора та флориста, вона також базується на інформації отриманої з бази даних. Тому якщо цієї

інформації буде дуже багато, потрібно забезпечити можливість переходу по сторінках, на яких буде відображатися фіксована кількість даних.

- Зручність

Головною ідеєю сайту було забезпечення зручним та простим інтерфейсом як клієнта так і адміністратора/флориста, щоб будь-хто зміг інтуїтивно розібратися, як користуватися сайтом. Рішенням цієї проблеми було створення максимально простого дизайну, не перевантаженого різними кнопками та можливостями.

- Надійність

Доступ до різних частин сайту розподіляється відповідно до ролей авторизованого юзера – адміністратор, флорист, клієнт. Доступ до головної частини сайту з переліком товарів та можливістю створювати замовлення є навіть у неавторизованого користувача. А ось щоб мати можливість переглядати створені замовлення – необхідно зареєструватись на сайті та створювати покупки як авторизований клієнт. Відповідно доступ до панелі управління адміністратора та флориста є тільки у користувачів з відповідною роллю.

Ще головним пунктом розроблення веб-сайту було забезпечити його коректну роботи при неправильних діях – некоректно введені дані під час реєстрації або неправильне заповнення даних для створення замовлення тощо.

2.7. Архітектура системи

Для реалізації веб-застосунку було вирішено використовувати стандартний підхід до розробки - це розділення застосунку на дві частини, взаємодія між якими базується на принципах HTTP-протокола:

- Клієнтську

Відображає кінцевий інтерфейс користувача, забезпечує коректну обробку дій з інтерфейсом, відправляє відповідні запити на сервер та оброблює його відповіді.

- Серверну

Серверна частина забезпечує обробку запитів, прийнятих від клієнта, та надає можливість доступу до бази даних.

Дану архітектуру можна представити наступним чином :

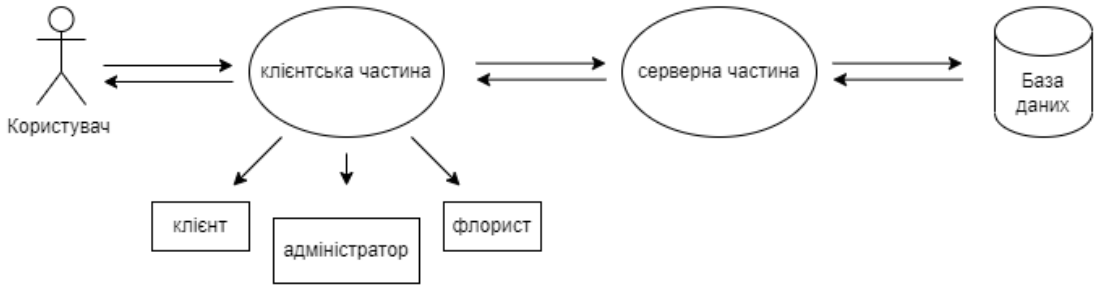

Рисунок 2.2 Клієнт-серверна архітектура

Спочатку користувач заходить на головну сторінку веб-сайту і має можливість взаємодіяти з інтерфейсом сайту. А саме він може зареєструватись або авторизуватись в системі. Відповідно до своєї ролі після авторизації він має доступ до різних можливостей сайту. Якщо користувач зайшов під роллю «Адміністратор» - він починає взаємодіяти з клієнтською частиною, а саме частиною для обробки взаємодій адміністратора. Так само відбувається і з іншими типами користувачів.

Після того як користувач відправив запит з клієнтської частини він потрапляє на сервер та починає опрацьовуватись відповідно до посилання в методі. Серверна частина розділена відповідно до розробленої бази даних. Тобто окремо оброблюються запити, які стосуються авторизації, клієнта,

співробітника, товару, категорії та замовлення. Далі серверна частина відправляє запит до бази даних і її відповідь відправляє на клієнтську частину.

2.8. Прототипи сторінок інтерфейсу користувача

Перед початком реалізації проєкту було створено найпростіші прототипи деякіх сторінок для того, щоб мати візуалізацію того, яким саме чином буде виглядати інтерфейс користувача.

Нижче представлений прототип описує структуру сторінки неавторизованого користувача. В цілому всі сторінки користувача сайту складаються з трьох частин – навігаційна панель, основна частина сайту та футер.

Рисунок 2.3 Прототип сторінки неавторизованого користувача

Після того як користувач буде авторизований як клієнт на навігаційній панелі зміниться лише один пункт – «Увійти». Замість нього буде відображатись ім’я користувача, при натисканні на яке з’являється випадаючий список з кнопками – «Профіль» та «Вийти».

Також було продумано вигляд сторінки з конкретним товаром, де повинна бути відображена вся необхідна інформація про товар для користувача.

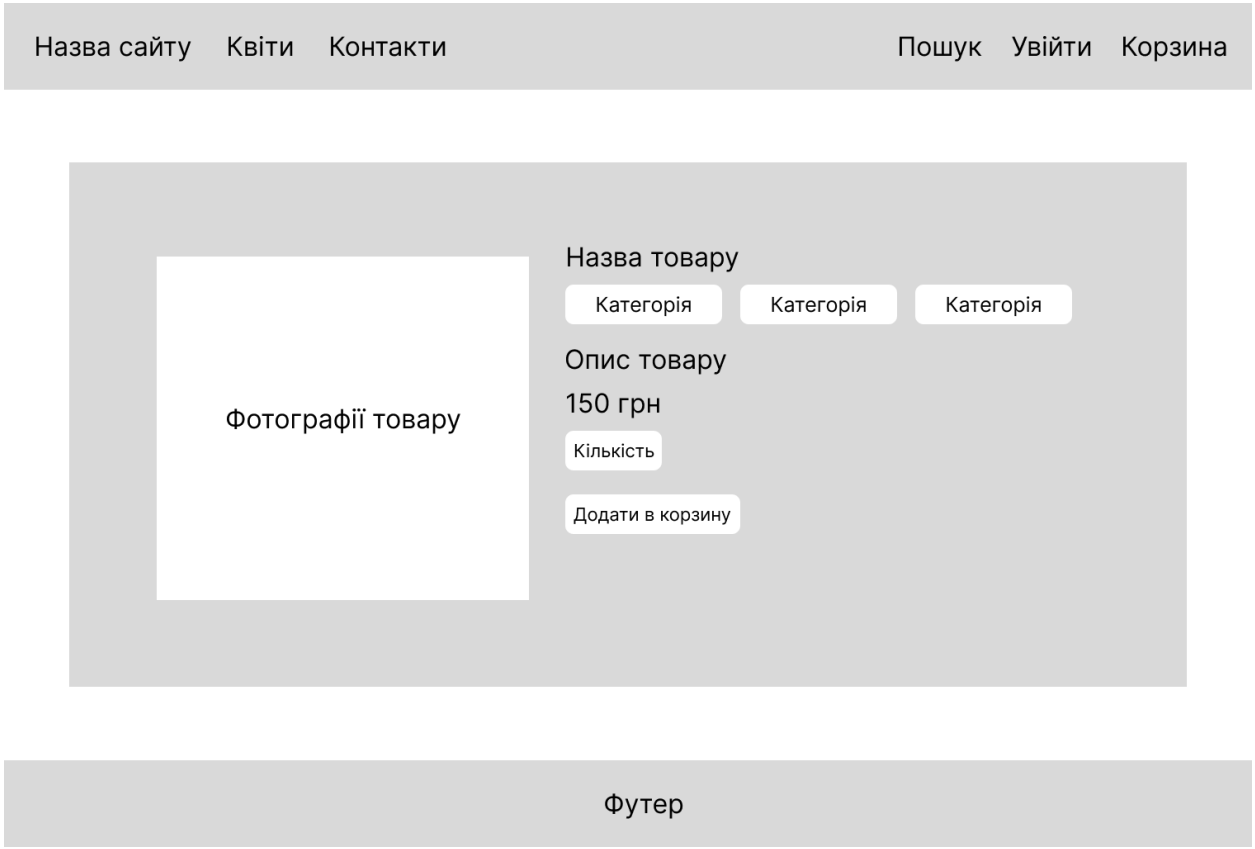

Рисунок 2.4 Прототип сторінки товару

Останнє, що потрібно було для початку реалізації проєкту - це прототипи панелі для адміністратора та флориста. Вони мають схожі функціональні можливості, тому прототипи їх сторінок нічим не відрізняються.

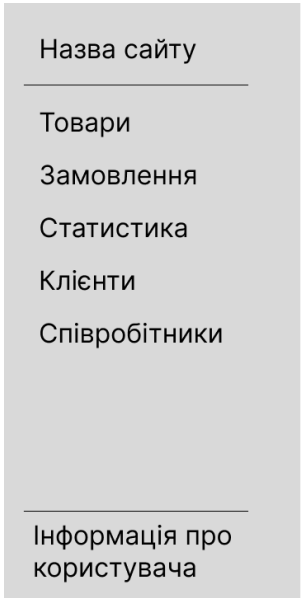

Рисунок 2.5 Прототип панелі адміністратора

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ПРОЄКТУ

3.1. Обрані технології та СКБД.

Для роботи з базою даних була обрана PostgreSQL, як СКБД. Причини цього вибору:

- повністю безкоштовна
- дуже популярна, багато великих компаній обирають саме PostgreSQL як СКБД
- знаходиться у відкритому доступі
- зручна у використанні та має інтерфейс для доступу до роботи з базами даних
- підтримує велику кількість типів даних, також можна додавати свої власні типи

Тобто ця СКБД повністю задовольняє потреби невеликого веб-сайту, тому для початку роботи над ним вона є ідеальним варіантом.

Для створення клієнтської та серверної частини проєкту, було вирішено використовувати JavaScript, так як це досить легка та зручна мова і повністю створена для розробки веб-застосувань. Також існує безліч різних фреймворків та бібліотек для різних завдань з метою полегшення розробки.

Для реалізації клієнтської частини проєкту використовувалася бібліотека React – фреймворк для створення інтерактивного інтерфейсу. Ця бібліотека спрощує та пришвидшує розробку інтерфейсу, так як за допомогою неї можна створювати різні окремі компоненти, які можна потім використовувати по декілька разів на сторінках. Саме таким чином створюється навігаційна панель сайту або ж товар, який на екрані може зустрічатись багато разів. Загалом почати використовувати і вчити React дуже просто, тому що ця бібліотека має гарну документацію та безліч

навчальних матеріалів, а також через те, що досить часто використовує синтаксис HTML. З єдиних мінусів, які були помічені, це те, що нова версія React дуже сильно відрізняється від старої, а саме деяким синтаксисом та назвою елементів, тому потрібно читати найновіші навчальні матеріали та документацію, але зараз і їх вже досить багато. Для дизайну деяких елементів використовувався HTML та CSS, дуже рідко використовувався Bootstrap 5.

Також для доступу до такої інформації як корзина користувача, було вирішено додати Redux, який значно спрощує передачу цієї інформації між сторінками інтерфейсу.

Для серверної частини було вирішено використовувати один з найпопулярніших фреймворків - Express JS. Створювати серверну частину за допомогою цього фреймворку досить легко та швидко. Також є можливість під'єднати різні бази даних такі як PostgreSQL, MongoDB, MySQL тощо. Дозволяє створювати RESTful API, який базується на HTTP протоколі (обмін інформації відбувається за допомогою 4 основних методів – GET, POST, PUT, DELETE). Саме так і відбувається взаємодія між клієнтською та серверною частиною розробленого веб-сайту. Тому Express JS був ідеальним варіантом для розробки веб-сайту.

3.2. Реалізація клієнтської частини застосунку

Клієнтська частина базується на автоматичному створенні структури сайту за допомогою команди npm create-react-app назва_папки. Після цієї команди створюється головний клієнтський файл index.js, який має кореневий елемент root, який потім відповідно до певного посилання передає компонент на відображення (приклад наведено нижче) та інші необхідні файли для роботи з бібліотекою React.

```
const root = ReactDOM.createRoot(document.getElementById('root'));
```

```
root.render(

  <React.StrictMode>
    <Provider store={store} >
      <App />
    </Provider>
  </React.StrictMode>
);
```

Обробка відповідного посилання і компонента здійснюється за допомогою бібліотеки react-router-dom, а саме BrowserRouter, Routes та Route. Використання цих класів - це стандартний підхід для створення багатосторінкового веб-застосунку.

Одним із найскладніших завдань веб-сайту є створення корзини з вибраними користувачем товарів та оформлення замовлення. Для того, щоб користувач мав доступ до списку товарів, які він обрав, необхідно десь зберігати цей список, щоб будь-який компонент сайту мав до нього доступ. Було вирішено реалізувати це за допомогою Redux, який зберігає стан застосунку.

Створення змінної зі станом застосунку:

```
const store = configureStore({
  reducer: {
    cart : cartReducer,
  }
});
```

При додаванні або видаленні товарів з корзини, а також зміні їх кількості за допомогою функції dispatch() ми викликаємо певний reducer, який відповідає нашій дії і який змінює стан застосунку. Але такий підхід є не дуже ефективним, так як дані при оновленні сторінки не зберігаються, тому

поки що стан застосунку, а саме корзина користувача, кожен раз зберігається в localStorage, що звісно не дуже відповідає надійності веб-застосунку. Для цього існують спеціальні бібліотеки, які зберігають стан Redux, тому це є одним з варіантом покращення цього веб-застосунку.

Ще одним важливим моментом розробки клієнтської частини – це відправка запитів на сервер. Це реалізовується за допомогою fetch запитів, приклад якого буде наведений нижче.

```
try {
    const response = await fetch("http://localhost:8000/auth/login", {
      method:"POST",
      headers: {"Content-Type" : "application/json"},
      body: JSON.stringify(body)
    });

    const parseResponse = await response.json();
  } catch (error) {
    console.log(error.message);
  }
```

Першим параметром цього методу є посилання, на яке відправляється наш запит, другий параметр – це тіло нашого запиту, в яке можна додати будь що. Наведений код використовується для відправки даних отриманих з форми авторизації.

Більшу частину коду клієнтської частини складає створення дизайну компонентів сайту та валідація даних, які передаються на сервер.

3.3. Реалізація серверної частини застосунку

Серверна частина складається з декількох елементів, які поєднуються в основному файлі index.js, який запускає роботу всього серверу.

Один з елементів серверної частини – це підключення бази даних і попередньо створення самої бази даних та таблиць. В додатку Г наведений код для створення бази даних та всіх таблиць в нії. Саме ж підключення відбувається за допомогою спеціальної бібліотеки для роботи з PostgeSQL–pg і виглядає наступним чином.

```
const Pool = require('pg').Pool;

const pool = new Pool({
   user: 'postgres',
   host:'localhost',
   database:'flower_shop',
   password:'пароль',
   port:'5432'
});
module.exports = pool;
```

Далі всюди, де потрібно відправляти запити до бази даних, потрібно імпортувати константу pool і використовувати саме її. Відправка запитів та запуск сервісу було розділено на дві відповідні окремі частини. Для кожної таблиці існує файл, в якому знаходяться основні методи взаємодії з нею, і вже потім цей файл підключається до основного.

Нижче будуть наведені одні з запитів, які використовуються для отримання необхідних даних в реалізованому веб-сайті.

Цей запит забезпечує отримання всієї інформації про товар, а саме – назви, опису, ціни, кількості і навіть посилання на одну фотографію

пов’язану з товаром. Це необхідно для коректного відображення списку наявних товарів у клієнта та таблиці всієї інформації про товар на панелі управління.

```
router.get("/all", async (req, res) => {
  try {
    const items = await db.query(
      "SELECT item.item_id, item.*, photo_url FROM item INNER JOIN (SELECT MAX(photo_url)  AS photo_url, item_id FROM item_photo WHERE item_photo.item_id=item_id GROUP BY item_id) AS p ON item.item_id=p.item_id WHERE item.amount>0" );
    return res.status(200).json({
      status:"success",
      data: { items: items.rows}});
  } catch (err) {
    res.status(500).send("Server error");
  }
});
```

Інколи в веб-застосунку потрібно отримати повну інформацію про конкретний товар, а саме всі категорії, до яких він відноситься, та всі його фотографії. Це можна отримати за допомогою іншого методу :

```
router.get("/categoriesAndPhoto/:id", async (req, res) => {
    try {
      const photos = await db.query(
        "SELECT photo_url FROM item_photo WHERE item_id = $1 ",[req.params.id] );
      const categories = await db.query(
        "SELECT category_name FROM category_item INNER JOIN category ON category_item.category_id=category.category_id WHERE item_id = $1 ",[req.params.id] );
     const item = await db.query(
```

```
      "SELECT item_id, item_name, item_description, sell_price, amount FROM item WHERE item_id = $1 ",[req.params.id] );
    return res.status(200).json({
     status:"success",
     data: { categories: categories.rows,
             photos:photos.rows,
             item:item.rows[0]}});
  } catch (err) {
   res.status(500).send("Server error");
  }
 });
```

Спочатку відправляємо перший запит для того, щоб отримати всі фотографії пов’язані з товаром, далі таким же чином отримуємо повну інформацію про категорії товару та сам товар. Всі необхідні дані відправляємо на клієнтську частину.

Ще один цікавий метод - це створення нового товару. Так як кожен товар повинен пов’язуватись з категоріями та фотографіями, то його додавання до бази даних ускладнюється.

```
router.post("/", authorize, async (req, res) => {
 try {
  const item = await db.query(
    "INSERT INTO item (item_name, item_description, order_price, sell_price, color, amount) VALUES ($1, $2, $3, $4, $5, $6) RETURNING* ",
    [req.body.item_name, req.body.item_description, req.body.order_price, req.body.sell_price, req.body.color, req.body.amount]);

  for (let i = 0; i < req.body.itemCategories.length; i++) {
    const cat = await db.query(
     "INSERT INTO category_item (item_id, category_id) VALUES ($1, $2) RETURNING*",
```

```
      [item.rows[0].item_id, req.body.itemCategories[i].category_id]);
    }

    for (let i = 0; i < req.body.itemPhotos.length; i++) {
      const ph = await db.query(
        "INSERT INTO item_photo (item_id, photo_url) VALUES ($1, $2)
RETURNING*",
        [item.rows[0].item_id, req.body.itemPhotos[i]]);
    }
    return res.status(200).json({
      status: "success",
      data: {
        item: item.rows[0]
      }
    });
  } catch (err) {
    res.status(500).send("Server error");
  }
});
```

Сам сервер запускається досить просто. Для цього потрібно імпортувати бібліотеку Express.Js, запустити її, підключити всі файли з запитами до бази даних та почати оброблювати запити на конкретному порті.

```
const express = require('express');
const cors = require('cors');
const app = express();
const {PORT} = require('./constants');

app.use(cors());
app.use(express.json());
```

```
app.use('/auth',require('./routes/jwtAuth'));
app.use('/user',require('./routes/user'));
app.use('/admin',require('./routes/admin'));
app.use('/item',require('./routes/item'));
app.use('/worker',require('./routes/worker'));
app.use('/order',require('./routes/order'));

//app start
const appStart = () => {
   try {
      app.listen(PORT, ()=>{
         console.log(`The server is running at http://localhost:${PORT}`)
      })
   } catch (error) {
      console.log(`Error: ${error.message}`)
   }
}

appStart();
```

3.4. Інтерфейс сайту

Інтерфейс сайту був розроблений відповідно до аналізу вже існуючих сайтів з подібною тематикою. Головна сторінка містить невелику інформацію про існуючий магазин. Ця сторінка має на меті познайомити користувача з тематикою сайту, розповісти про бізнес та зацікавити користувача в продуктах сайту. Розроблена сторінка – це найпростіший варіант для демонстрації, який необхідно допрацювати дизайнерам для того, щоб привернути увагу користувача.

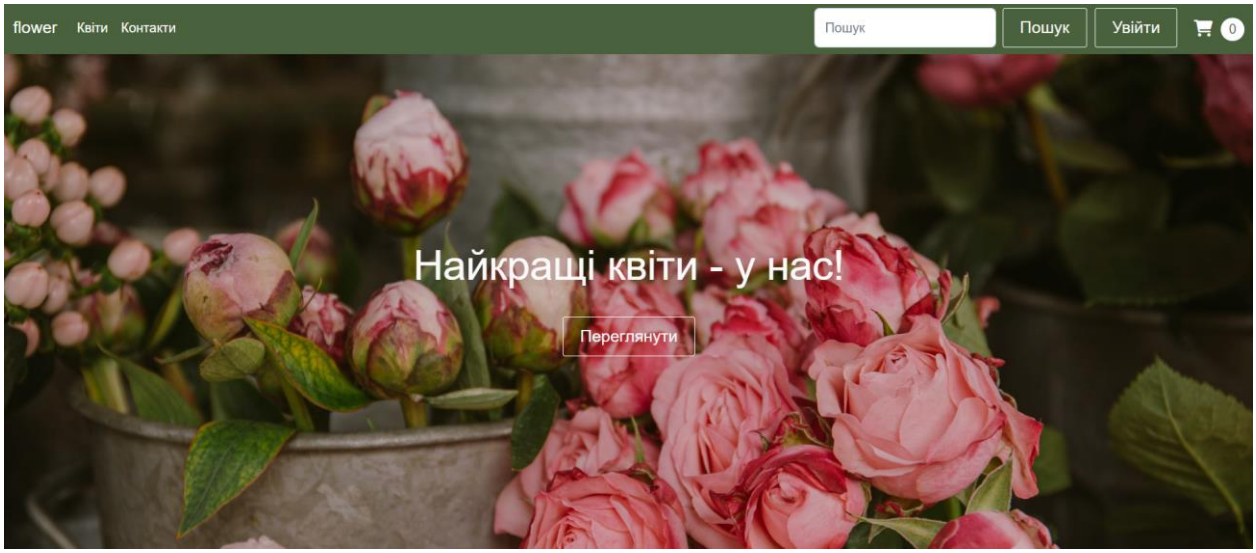

Рисунок 3.1 Головна сторінка неавторизованого користувача

Наступна сторінка – це перелік всіх доступних товарів на сайті. Відображаються тільки ті товари, які є в наявності. При натисканні на фотографію товару відкривається сторінка з детальним описом товару. Також в правому верхньому куті можна помітити корзину з відображенням кількості товарів в ній. Якщо користувач натисне на кнопку «Додати», то він буде перенаправлений на сторінку корзини, де він буде мати можливість вибрати кількість товару, яку він хоче замовити.

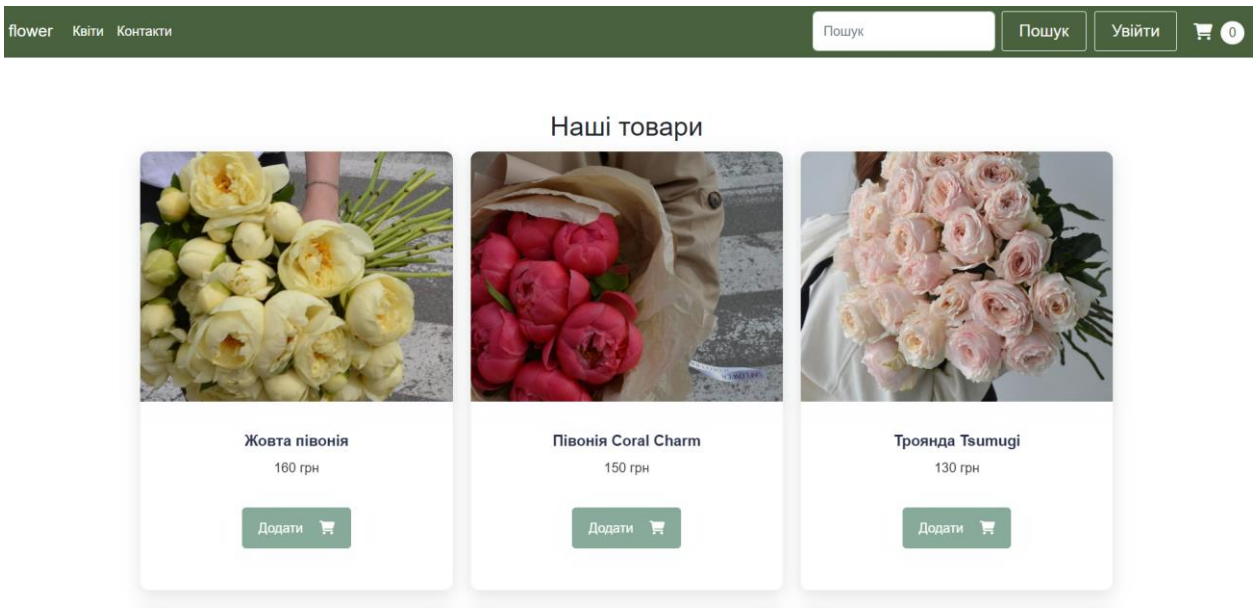

Рисунок 3.2 Сторінка з товарами

На сторінці з детальною інформацією про товар можна переглянути категорії, до яких відноситься даний товар, прочитати його опис, побачити кількість товару, який залишився в наявності, і звісно можна переглядати фотографії товару.

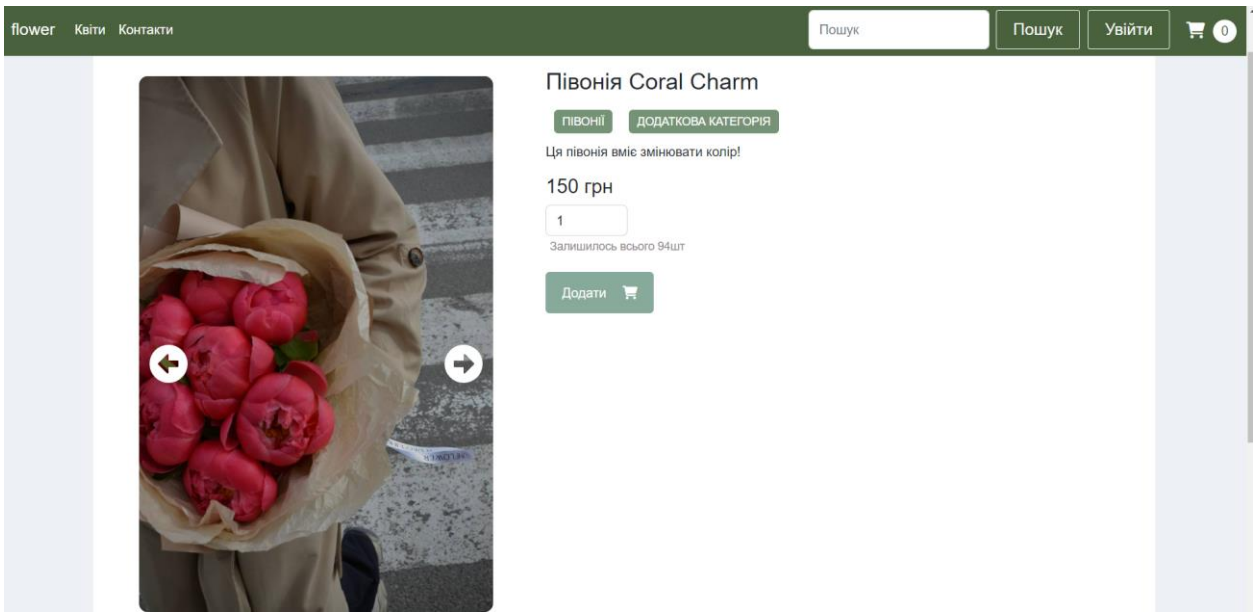

Рисунок 3.3 Детальна інформація про конкретний товар

На сторінці корзини користувач має можливість переглядати обраний товар, видаляти його, змінювати кількість товару або ж повністю очищати корзину. Далі можна перейти до оформлення замовлення, де потрібно ввести всі необхідні дані для доставки або ж вибрати самовивіз з філіалу магазину. Також якщо користувач незареєстрований, то йому обов'язково потрібно ввести дані про себе, а саме електронну пошту, телефон та ім'я для того, щоб була можливість зв'язатись з ним.

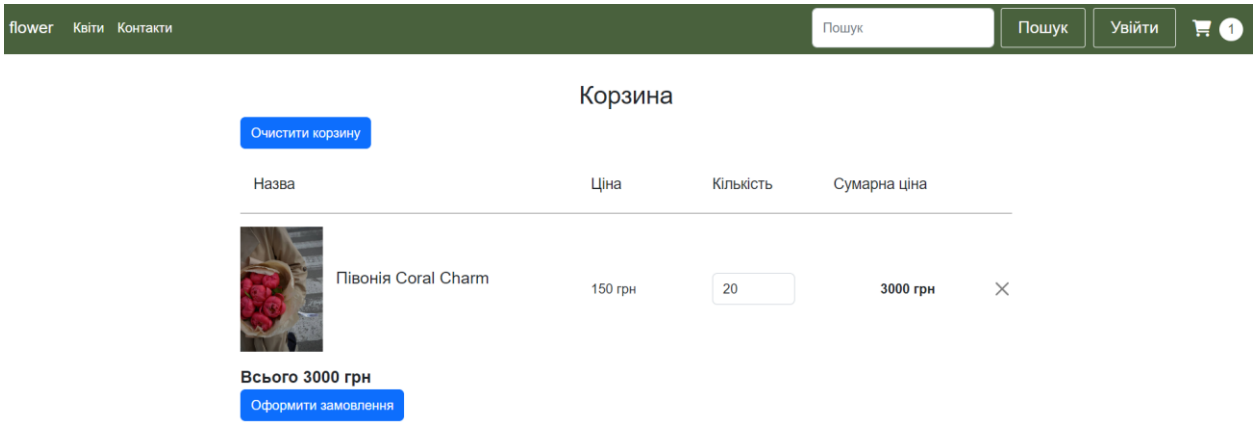

Рисунок 3.4 Корзина з товарами

Якщо користувач авторизований та робив замовлення, то він має можливість переглядати їх та слідкувати за статусом. Можливі варіанти статусу:

– «в процесі» - означає, що замовлення тільки надійшло і ще обробляється.

– «виконується» - означає, що замовлення почалося виконуватись флористом.

– «доставляється» - означає, що замовлення відправилось за адресом доставки.

– «готове» - означає, що замовлення виконалось і його можна прийти забрати, якщо клієнт вказав самовивіз.

– «виконане» - означає, що клієнт отримав своє замовлення.

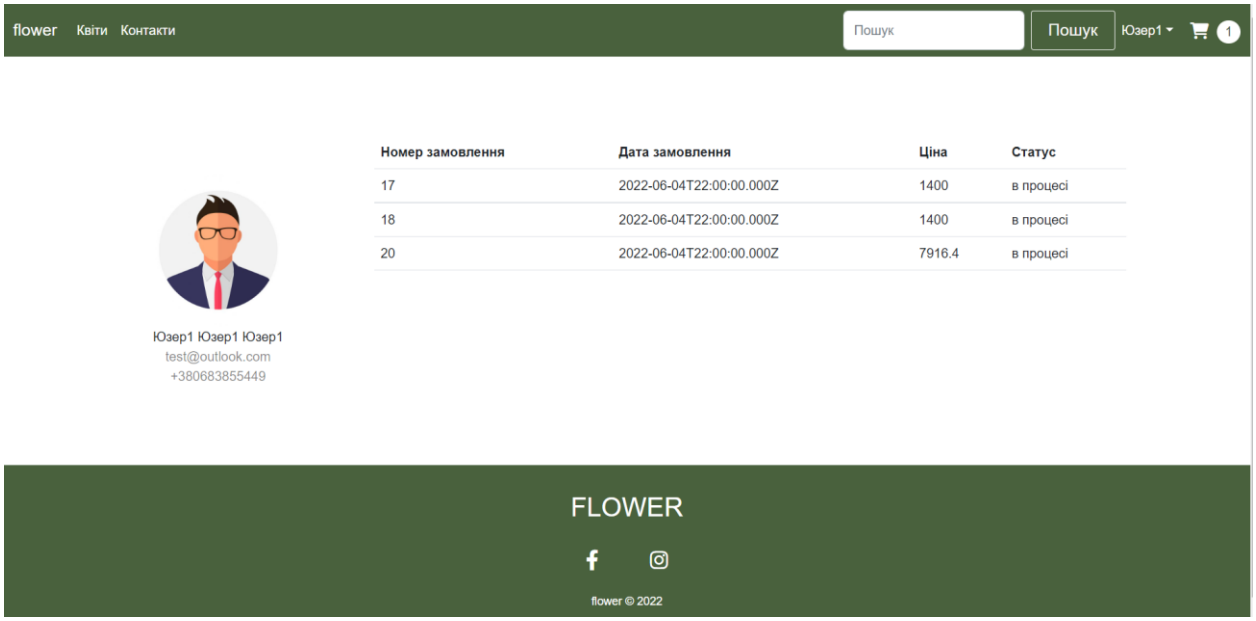

Рисунок 3.5 Профіль користувача

При натисканні на замовлення можна переглянути його деталі, а саме, які товари в нього входили, адрес доставки тощо.

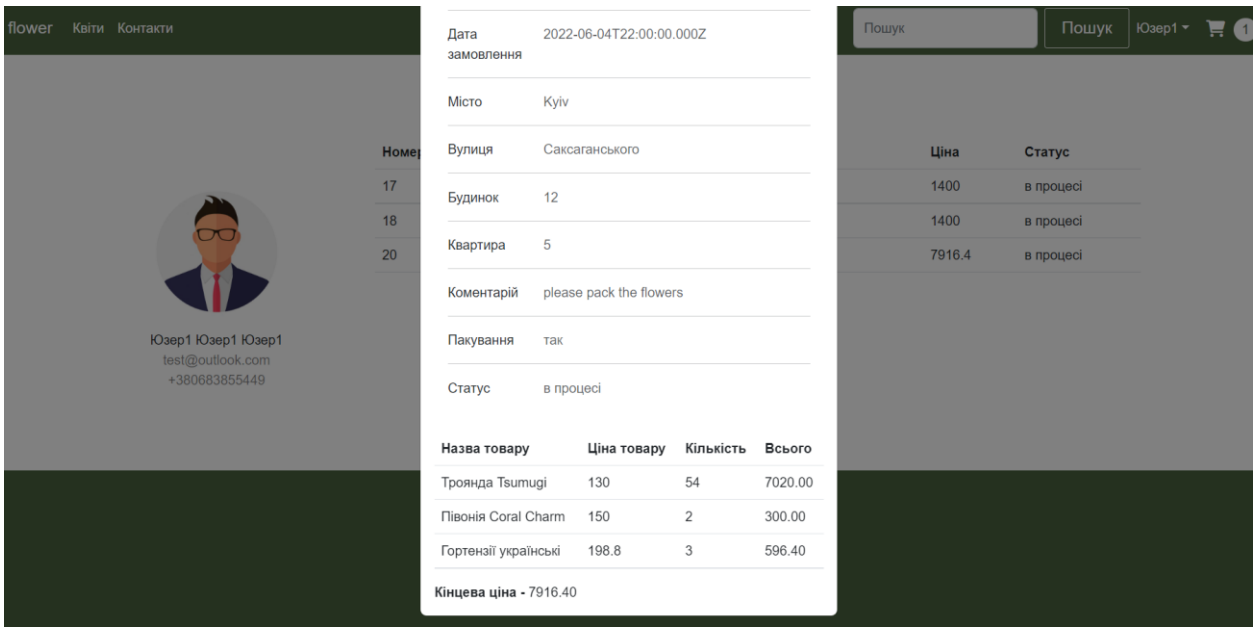

Рисунок 3.6 Перегляд деталей замовлення

Якщо під час авторизації у користувача буде вказана роль адміністратора, він буде перенаправлений на панель управління. Головна сторінка відображає перелік всіх товарів, які зберігаються в базі даних. Якщо кількість товарів 0, то цей товар буде підсвічуватись жовтим, щоб привернути уваги, що його потрібно замовити.

Рисунок 3.7 Адміністраторська панель управління

На адміністраторській панелі є багато можливостей для управління даними. Наприклад, можна додати товар до нашого сайту. Під час додавання товару необхідно заповнити всі обов’язкові поля, додати хоча б одну категорію до товару та фото.

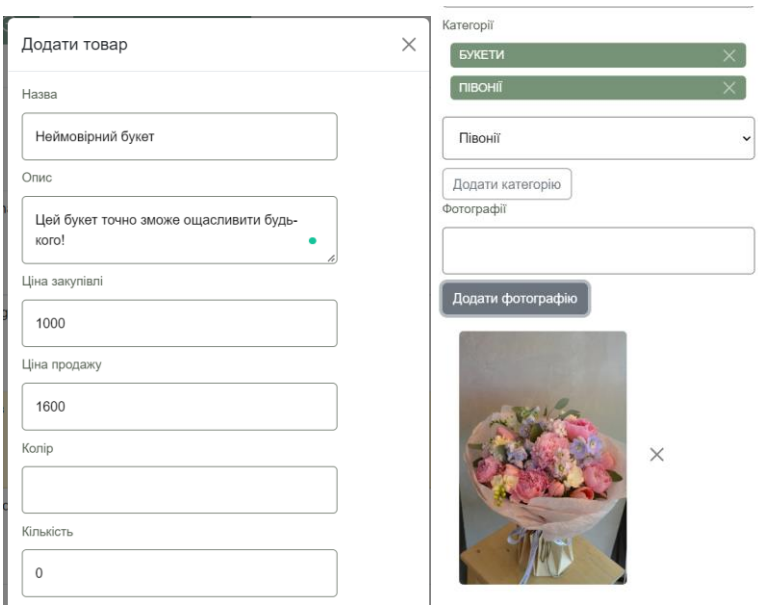

Рисунок 3.8 Додавання товару

Є можливість переглядати створені категорії та додавати нову, а також дивитись детальну інформацію про товар.

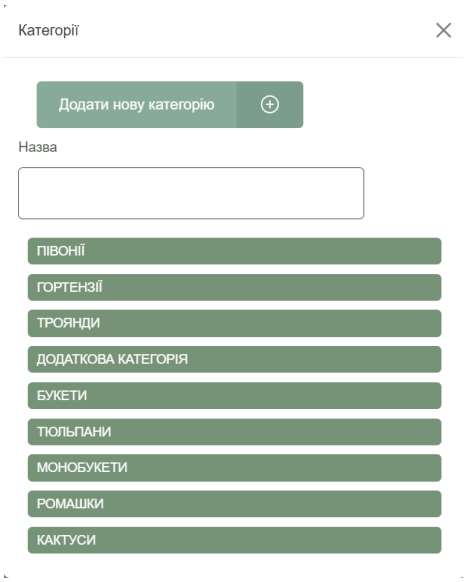

Рисунок 3.9 Перегляд та додавання категорій

Рисунок 3. 10 Перегляд товару

Також у адміністратора є можливість переглядати інформацію про всіх клієнтів, співробітників та замовлень на відповідних сторінках. Флорист має майже ті самі можливості, що й адміністратор, тому його панель управління не дуже відрізняється.

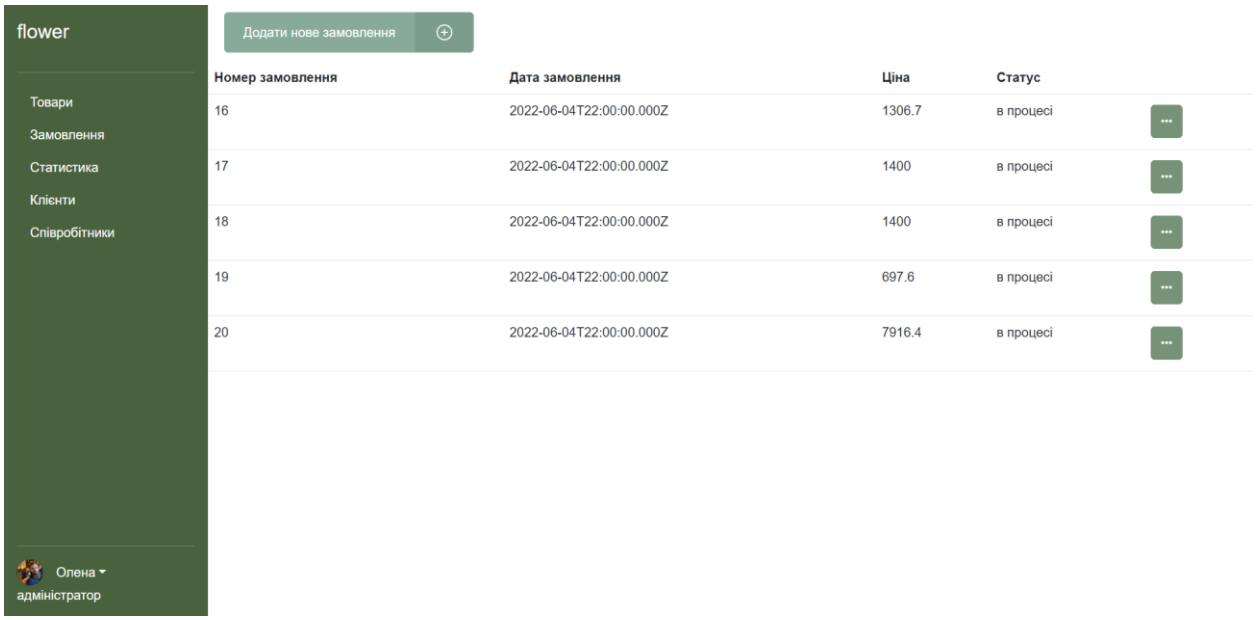

Рисунок 3. 11 Перегляд замовлень

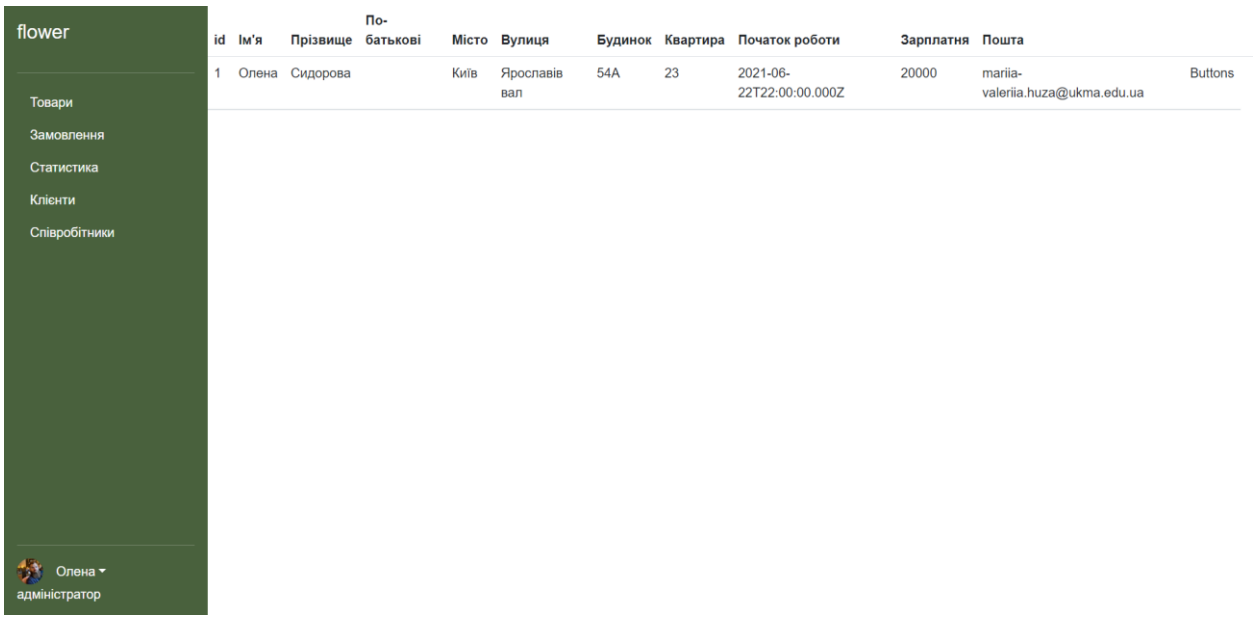

Рисунок 3.12 Перегляд співробітників

3.5. Мапа сайту

Для розуміння структури сайту була розроблена мапа, яка демонструє послідовність взаємодії з реалізованим веб-сайтом.

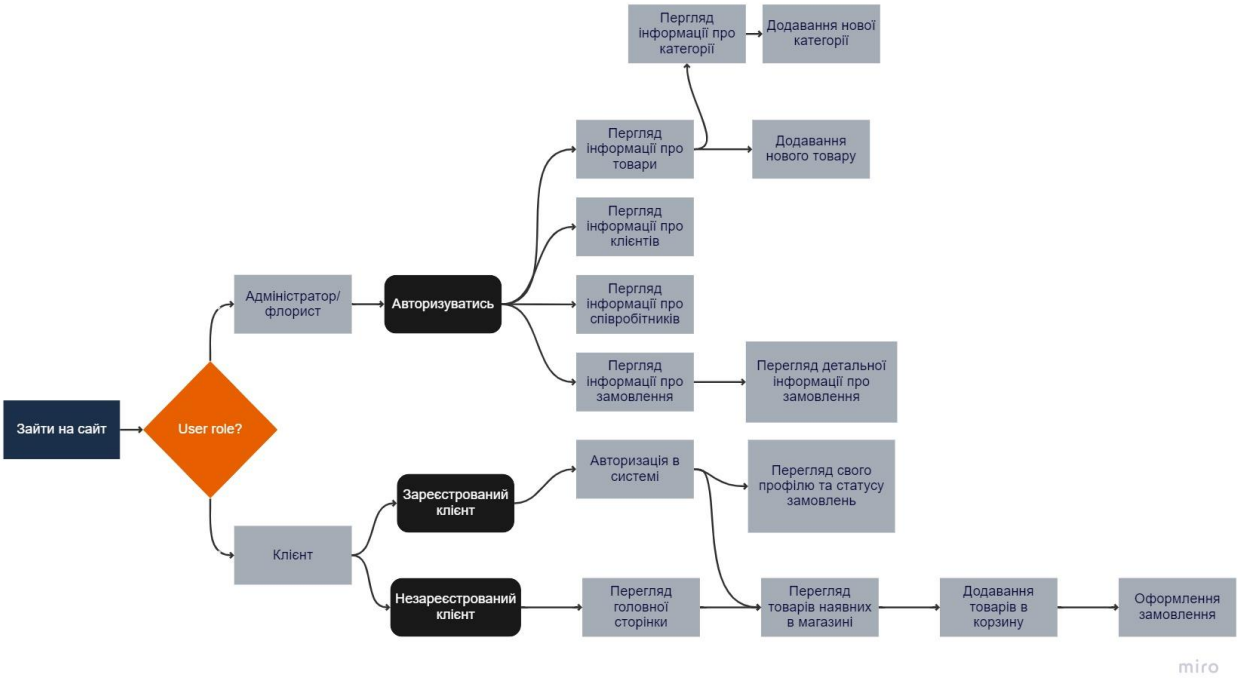

Рисунок 3.13 Мапа сайту

Висновки

Результатом роботи стало готове веб-застосування для продажу квітів з можливостю авторизуватись як клієнт, адміністратор та флорист. У ході роботи був проведений аналіз кінцевих користувачів для визначення актуальності даного веб-сайту, дослідження існуючих рішень для розуміння того, яким повинен бути кінцевий результат, та виявлення недоліків. Була розроблена універсальна ER-модель та на базі неї побудований кінцевий веб-сайт.

Для його реалізації була обрана клієнт-серверна архітектура застосунку. Він реалізований на мові JavaScript, а саме бібліотеках React, Redux, Express.js тощо з підключенням до бази даних PostgeSQL.

Розроблений проєкт має вигляд багатосторінкового веб-сайту, дані якого динамічно виводяться з бази даних. Є можливість авторизуватись як клієнт, адміністратор, флорист або ж переглядати сайт як неавторизований користувач. Також клієнт може додавати обрані товари в корзину, переглядати свою корзину, створювати замовлення тощо. На панелі управляння для адміністратора та флориста є наступні функції: перегляд або додавання товару, перегляд або додавання категорії, перегляд інформації про співробітників, клієнтів та замовлення.

Отже, розроблений веб-застосунок для квіткового магазину є повноцінною автоматизованою інформаційною системою, якою можуть користуватись і клієнти, і співробітники магазину. В ході розробки даного застосунку було вивчено PostgreSQL та використано багато бібліотек JavaScript.

Перелік використаних джерел

1. Веб-сайт квіткового магазину [Електронний ресурс]. – Режим доступу: https://flowers.ua/
2. Магазин Камелія [Електронний ресурс]. – Режим доступу: https://shop.camellia.ua/ua
3. Hello World. React documentation [Електронний ресурс]. – Режим доступу: https://reactjs.org/docs/hello-world.html
4. Introduction in Bootstrap 5 [Електронний ресурс]. – Режим доступу: https://getbootstrap.com/docs/5.0/getting-started/introduction/
5. Routing Express.js [Електронний ресурс]. – Режим доступу: https://expressjs.com/en/guide/routing.html
6. Writing middleware for use in Express apps [Електронний ресурс]. – Режим доступу: https://expressjs.com/en/guide/writing-middleware.html
7. cors [Електронний ресурс]. – Режим доступу: https://www.npmjs.com/package/cors
8. Getting Started with PostgreSQL [Електронний ресурс]. – Режим доступу: https://www.postgresqltutorial.com/postgresql-getting-started/
9. Redux Fundamentals [Електронний ресурс]. – Режим доступу: https://redux.js.org/tutorials/fundamentals/part-1-overview
10. React Router Tutorial [Електронний ресурс]. – Режим доступу: https://reactrouter.com/docs/en/v6/getting-started/tutorial

Додаток А

(обов'язковий)

Опитування кінцевих користувачів

1. Хто ви?

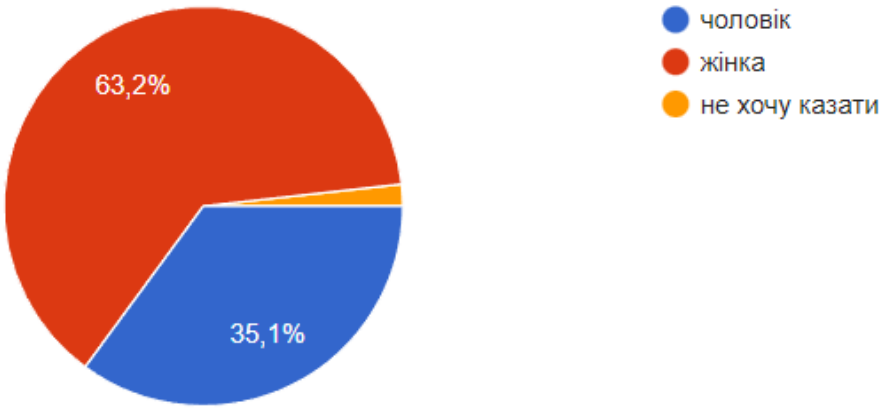

Рисунок А.1 Відповіді на питання

2. Скільки вам років?

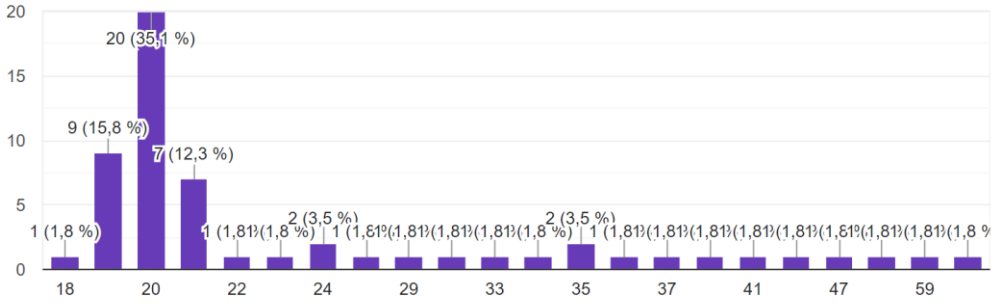

Рисунок А.2 Відповіді на питання

3. Як часто ви купуєте квіти?

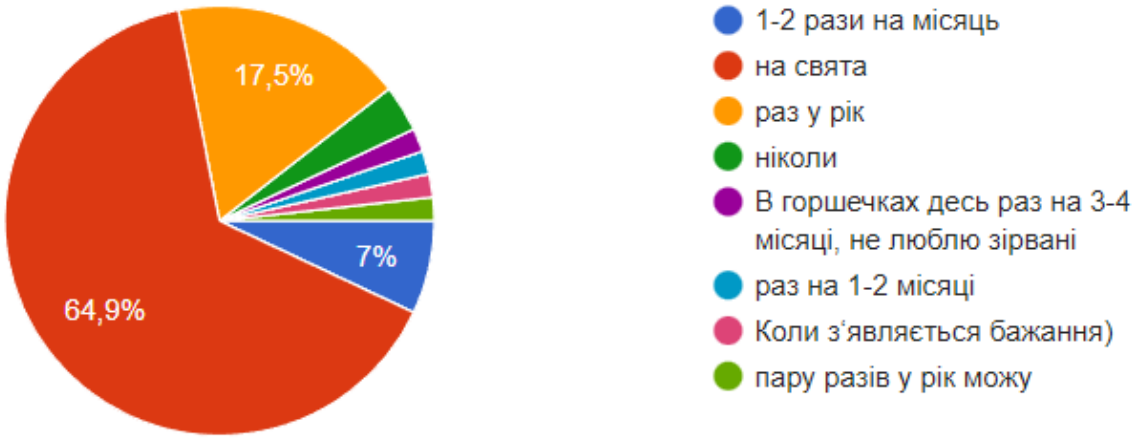

Рисунок А.3 Відповіді на питання

4. Як ви купуєте квіти?

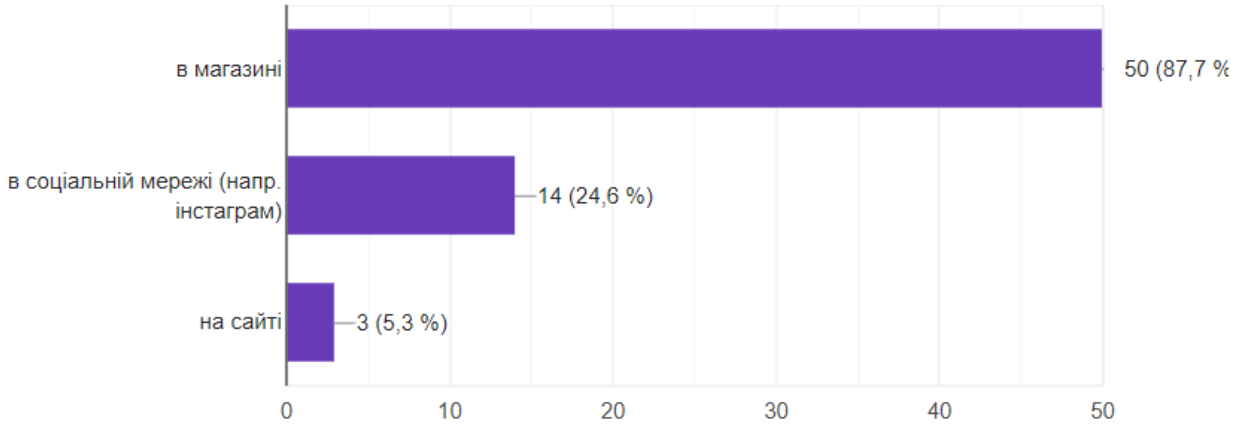

Рисунок А.4 Відповіді на питання

5. Чи потрібен квітковому магазину сайт?

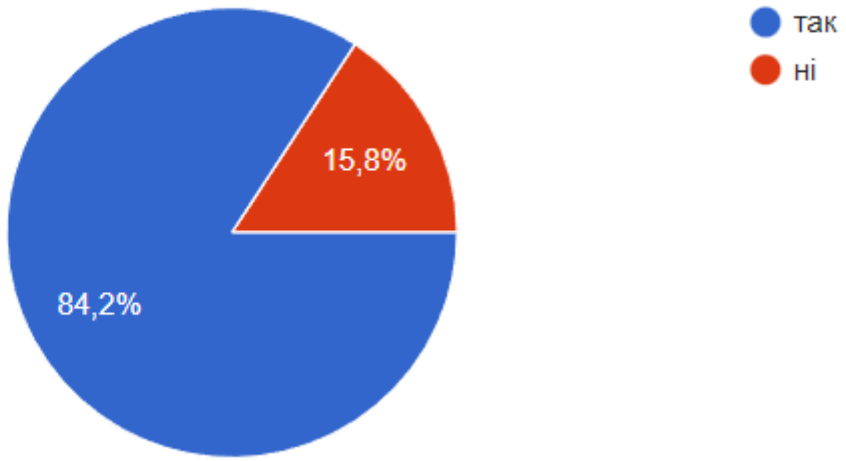

Рисунок А.5 Відповіді на питання

6. Якщо квітковий магазин мав би сайт – чи користувались би ви ним?

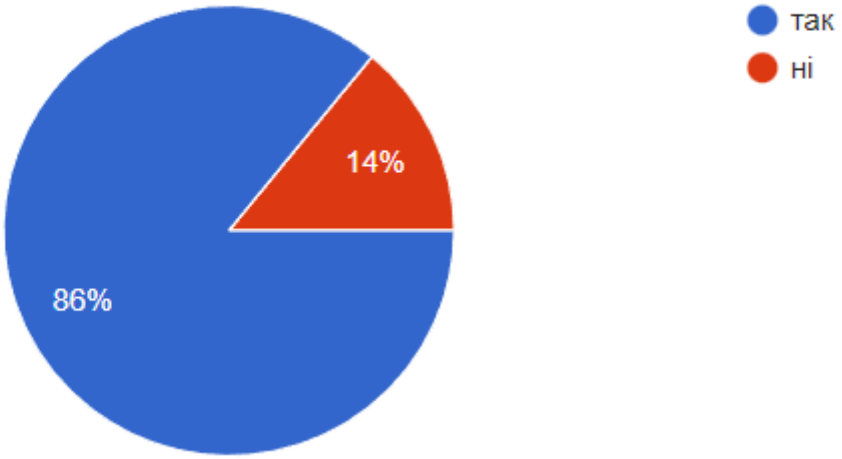

Рисунок А.6 Відповіді на питання

7. Що б ви хотіли бачити на сайті, якби ви користувались ним?

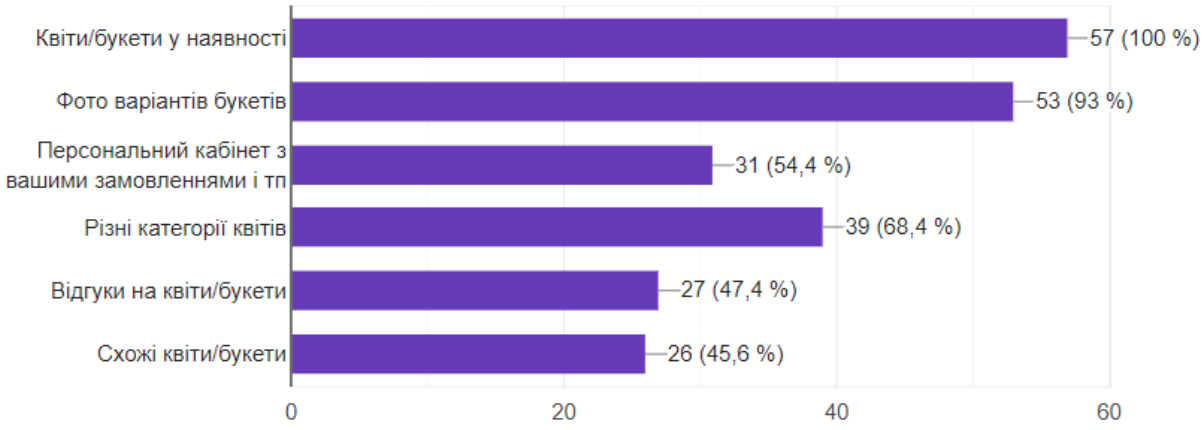

Рисунок А.7 Відповіді на питання

Додаток Б

(обов'язковий)

Частина інтерв'ю з людиною, яка починала створювати квітковий бізнес

Моє питання :

«Як я правильно зрозуміла, головними труднощами ведення будь-якого бізнесу - це є ведення документації, опрацювання та контролювання замовлень, перевірка кількості товарів, навіть якщо ви маєте невелику квіткову крамницю.

На вашу думку, чи корисно було б навіть одному невеликому магазину мати спеціальну автоматизовану систему, де буде зберігатись інформація про кожне замовлення, товар, ким було опрацьоване замовлення, хто був клієнтом? Тобто повноцінний сайт, де не тільки флорист може отримувати всі необхідні статистичні дані (наприклад, який товар був найпопулярніший в певний період, скільки залишилось товару на складі, всі замовлення на сьогоднішню дату і тп), а й клієнт може перевіряти який товар є в наявності, зробити замовлення онлайн і тп. Можливо, вас би цікавили ще якісь додаткові можливості для того, щоб зробити ведення вашого бізнесу простішим там ефективнішим?»

Відповідь:

«Загалом, усі підприємства (не лише у сфері флористики) усіх масштабів займаються товарним обліком, аби відстежувати кількість продажів, планувати закупівлі, уникати недостач, надлишків та помилок. Також це необхідно, щоб аналізувати попит.

У своїй діяльності я використовувала excel, інколи за браком часу щось записувалось за папері. Звісно, це не дуже надійно і хотілось би користуватись якимось зручнішим сайтом або додатком, націленим саме на

підприємницький облік і ведення клієнтської бази. Ще краще, якби ним могли користуватись й споживачі, мати доступ до актуальної інформації про наявність квітів, робити онлайн-замовлення, отримувати сповіщення про акційні пропозиції тощо.»

Додаток В

(обов’язковий)

Реляційна модель даних

client Відповідає – “Клієнт“					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	client_id	integer (serial)	NN		Простий первинний ключ, автоінкремент
	username	varchar(50)	NN		Частина складеного атрибуту ПІБ, максимальна кількість символів - 50
	usersurname	varchar(50)	NN		Частина складеного атрибуту ПІБ, максимальна кількість символів - 50
	userfathername	varchar(50)	N		Частина складеного атрибуту ПІБ, необов'язковий атрибут, максимальна кількість символів - 50
	userphone	varchar(50)	NN		Простий обовязковий атрибут, максимальна кількість символів – 50. Було вирішено представляти телефон у вигляді стрічки, так як при реєстрації проситься вводити повний формат телефону з +380 – на

					початку.
	email	varchar(255)	NN		Простий обовязковий атрибут, максимальна кількість символів - 255

worker Відповідає – “Співробітник”					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	worker_id	Integer (serial)	NN		Простий первинний ключ, автоінкремент
	worker_surname	varchar(50)	NN		Частина складеного атрибуту ПІБ, максимальна кількість символів - 50
	worker_name	varchar(50)	NN		Частина складеного атрибуту ПІБ, максимальна кількість символів - 50
	worker_fathername	varchar(50)	N		Частина складеного атрибуту ПІБ, необов'язковий атрибут, максимальна кількість символів - 50
	worker_city	varchar(50)	NN		Частина складеного атрибуту адреса, максимальна кількість символів - 50
	worker_street	varchar(50)	NN		Частина складеного атрибуту адреса, максимальна кількість символів - 50
	worker_house	varchar(50)	NN		Частина складеного атрибуту адреса, максимальна кількість

					символів - 50
	worker_flat	integer	N		Частина складеного атрибуту адреса, необов'язковий атрибут
	worker_salary	numeric	NN		Простий обов'язковий атрибут, для представлення грошових одиниць було вирішено використовувати тип numeric для спрощення роботи з цим атрибутом
	start_work_at	date	NN		Простий обов'язковий атрибут
FK	worker_admin_id	integer	N	ON DELETE – set null ON UPDATE - CASCADE	Посилання на співробітника, який керує даним співробітником. При видаленні інформації про співробітника – зробити поле Null. Оновлювати посилання при зміні даних.

phone Відповідає – "Телефон"					
Ключ	*Ім'я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	phone	varchar(50)	NN		Простий первинний ключ
FK	fk_worker_id	integer	NN	ON DELETE – CASCADE ON UPDATE - CASCADE	Посилання на співробітника. При видаленні співробітника – видаляти номер телефону. Оновлювати посилання при зміні даних.

order Відповідає – “Замовлення”					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	order_id	Integer(serial)	NN		Простий первинний ключ, автоінкремент
	order_date	date	NN		Простий обов’язковий атрибут
	order_delivery	date	N		Простий необов’язковий атрибут, доставка може бути в інший день
	order_street	varchar(50)	N		Частина необов'язкового складеного атрибуту адреса
	order_city	varchar(50)	N		Частина необов'язкового складеного атрибуту адреса
	order_house	varchar(50)	N		Частина необов'язкового складеного атрибуту адреса
	order_flat	integer	N		Частина необов'язкового складеного атрибуту адреса
	order_comment	varchar	N		Простий необов’язковий атрибут
	package	boolean	NN		Простий обов’язковий атрибут, при створенні замовлення обов’язково вказується чи потрібно пакування, до кінцевої ціни додається фіксована

					сума за пакування
	discount	numeric	N		Простий необов’язковий атрибут
	order_price	numeric	NN		Простий обов’язковий атрибут
	order_state	varchar(50)	NN		Стан замовлення може бути такий – оброблюється, у процесі, доставляється, виконаний
FK	fk_client_id	integer	N	ON DELETE – Set null ON UPDATE – cascade	Посилання на клієнта, який оформив замовлення. При створенні замовлення офлайн, посилання на клієнта немає. Так як замовлення повинні зберігатись в базу будь-якому разі, при видаленні клієнта – встановити полю значення null. Оновлювати посилання при зміні даних.
FK	Fk_florist_id	integer	NN	ON DELETE – set null ON UPDATE - CASCADE	Посилання на флориста, який виконує замовлення. Це потрібно для зберігання статистико по флористу. Так як замовлення повинні зберігатись в базу будь-якому разі, при видаленні робітника – встановити полю значення null. Оновлювати посилання при зміні даних.

order_unit

Відповідає – “Одиниця замовлення”

Ключ	*Ім'я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	unit_id	Integer(serial)	NN		Простий первинний ключ, автоінкремент
	unit_num	integer	NN		Простий обов'язковий атрибут
FK	fk_order_id	integer	NN	ON DELETE – NO ACTION ON UPDATE - CASCADE	Посилання на замовлення, до якого ця одиниця відноситься. Видаляти замовлення не можна, тому важливо й зберігати інформацію про одиниці замовлення. Оновлювати посилання при зміні даних.
FK	fk_item_id	integer	NN	ON DELETE – NO ACTION ON UPDATE - CASCADE	Посилання на товар. Видаляти інформацію про товар не можна, так як повинна зберігатись інформація про замовлення. Оновлювати посилання при зміні даних.

item Відповідає – "Товар"					
Ключ	*Ім'я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	item_id	Integer(serial)	NN		Простий первинний ключ, автоінкремент

	item_name	varchar(255)	NN		Простий обов’язковий атрибут
	item_description	varchar	NN		Простий обов’язковий атрибут
	order_price	numeric	NN		Простий обов’язковий атрибут
	sell_price	numeric	NN		Простий обов’язковий атрибут, похідний атрибут, до ціни закупівлі додається певний процент.
	color	varchar(50)	N		Простий необов’язковий атрибут, потрібен для фільтрування товарів

item_photo Відповідає – “Фото”					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PK	photo_url	varchar(255)	NN		Простий первинний ключ
FK	item_id	integer	NN	ON DELETE – CASCADE ON UPDATE - CASCADE	Посилання на товар, до якого відноситься це фото. Видаляти фото при видаленні товару, оновлювати посилання при змінні даних.

category Відповідає – “Категорія”					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT*	*FK:* *ON DELETE,*	*Пояснення (перехід від ER-моделі)*

			NULL	*ON UPDATE*	
PK	category_id	Integer(serial)	NN		Простий первинний ключ, автоінкремент
	category_name	varchar(50)	NN		Простий обов’язковий атрибут

category_item Відповідає – “Зв’язок багато-до-багатьох між категорією і товаром”					
Ключ	*Ім’я атрибуту*	*Тип атрибуту*	*NULL / NOT NULL*	*FK:* *ON DELETE, ON UPDATE*	*Пояснення (перехід від ER-моделі)*
PPK, FK	category_id	Integer(serial)	NN	ON DELETE – CASCADE ON UPDATE - CASCADE	Частина первинного ключа, посилання на категорію. При видаленні категорії – видаляти дані. При зміні даних – оновлювати.
PPK, FK	item_id	Integer(serial)	NN	ON DELETE – CASCADE ON UPDATE - CASCADE	Частина первинного ключа, посилання на товар. При видаленні категорії – видаляти дані. При зміні даних – оновлювати.

Додаток Г

(обов’язковий)

Створення таблиць в базі даних

```
create database flower_shop;

create table users(
    user_id serial primary key,
    email varchar(255) unique not null,
    userRole varchar(255) not null,
    password varchar(255) not null
);

create table clients(
    client_id serial primary key,
    userName varchar(255) not null,
    userSurname varchar(255) not null,
    userFatherName varchar(255),
    userPhone varchar(255) not null,
    email varchar(255) unique not null
);

CREATE TABLE worker(
    worker_id SERIAL PRIMARY KEY,
    worker_name VARCHAR(50) NOT NULL,
    worker_surname VARCHAR(50) NOT NULL,
    worker_fatherName VARCHAR(50),
    worker_city VARCHAR(50) NOT NULL,
    worker_street VARCHAR(50) NOT NULL,
```

```
    worker_house VARCHAR(50) NOT NULL,
    worker_flat INTEGER,
    start_work_at DATE NOT NULL,
    worker_salary NUMERIC NOT NULL,
    worker_admin_id INTEGER,
    worker_email VARCHAR(255) ,
    CONSTRAINT worker_admin_id FOREIGN KEY(worker_admin_id) REFERENCES worker(worker_id) ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE phone(
    phone_id VARCHAR(50) PRIMARY KEY,
    worker_id INTEGER NOT NULL,
    CONSTRAINT fk_worker_id FOREIGN KEY(worker_id) REFERENCES worker(worker_id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE order_c(
    order_id SERIAL PRIMARY KEY,
    order_date DATE NOT NULL,
    order_delivery DATE,
    order_city VARCHAR(50) NOT NULL,
    order_street VARCHAR(50) NOT NULL,
    order_house VARCHAR(50) NOT NULL,
    order_flat INTEGER,
    order_comment VARCHAR,
    package BOOLEAN NOT NULL,
    discount NUMERIC,
    order_price NUMERIC NOT NULL,
    order_state VARCHAR(50) NOT NULL,
```

```
    client_id INTEGER,
    worker_id INTEGER,
    CONSTRAINT fk_client_id FOREIGN KEY(client_id) REFERENCES
clients(client_id) ON DELETE SET NULL ON UPDATE CASCADE,
    CONSTRAINT fk_florist_id FOREIGN KEY(worker_id) REFERENCES
worker(worker_id) ON DELETE SET NULL ON UPDATE CASCADE
);

CREATE TABLE order_unit(
    unit_id SERIAL PRIMARY KEY,
    unit_num INTEGER NOT NULL,
    order_id INTEGER NOT NULL,
    item_id INTEGER NOT NULL,
    CONSTRAINT fk_order_id FOREIGN KEY(order_id) REFERENCES
order_c(order_id) ON DELETE NO ACTION ON UPDATE CASCADE,
    CONSTRAINT fk_item_id FOREIGN KEY(item_id) REFERENCES
item(item_id) ON DELETE NO ACTION ON UPDATE CASCADE
);

CREATE TABLE item(
     item_id SERIAL PRIMARY KEY,
     item_name VARCHAR(255) NOT NULL,
     item_description VARCHAR NOT NULL,
     order_price NUMERIC NOT NULL,
     sell_price NUMERIC NOT NULL,
     amount INTEGER NOT NULL,
     color VARCHAR(255)
);

CREATE TABLE item_photo(
    photo_url VARCHAR(255) PRIMARY KEY,
```

```
    item_id INTEGER NOT NULL,
    CONSTRAINT fk_item_id FOREIGN KEY(item_id) REFERENCES item(item_id) ON DELETE CASCADE ON UPDATE CASCADE
);

CREATE TABLE category(
    category_id SERIAL PRIMARY KEY,
    category_name VARCHAR(50) NOT NULL
);

CREATE TABLE category_item(
    item_id INTEGER NOT NULL,
    category_id INTEGER NOT NULL,
    CONSTRAINT fk_item_id FOREIGN KEY(item_id) REFERENCES item(item_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT fk_category_id FOREIGN KEY(category_id) REFERENCES category(category_id) ON DELETE CASCADE ON UPDATE CASCADE,
    CONSTRAINT id PRIMARY KEY (item_id,category_id)
);
```