

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»**

Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2025 р.

Дипломний проєкт

на здобуття ступеня бакалавра

**за освітньо-професійною програмою «Інженерія програмного
забезпечення мультимедійних та інформаційно-пошукових систем»**

спеціальності 121 Інженерія програмного забезпечення

**на тему: «Вебзастосунок для онлайн створення та купівлі букетів з
квітів»**

Виконав:

студент IV курсу, групи КП-12

Онасенко Олексій Володимирович _____

Керівник:

асистент кафедри ПЗКС, д-р філософії,

Юсин Яків Олексійович _____

Консультант з нормоконтролю:

доцент кафедри ПЗКС, к.т.н., доцент,

Онай Микола Володимирович _____

Рецензент:

доцент кафедри ПСТ ФІТ «КНУ

ім. Тараса Шевченка», д.т.н., с.н.с.,

Порєв Геннадій Володимирович _____

Засвідчую, що у цьому дипломному
проєкті немає запозичень з праць інших
авторів без відповідних посилань.

Студент _____

Київ – 2025 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет прикладної математики

Кафедра програмного забезпечення комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Інженерія програмного забезпечення
мультимедійних та інформаційно-пошукових систем»

ЗАТВЕРДЖУЮ

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2024 р.

ЗАВДАННЯ

на дипломний проєкт студенту

Онасенко Олексію Володимировичу

1. Тема проєкту «Вебзастосунок для онлайн створення та купівлі букетів з квітів», керівник проєкту Юсин Яків Олексійович, асистент кафедри ПЗКС, доктор філософії, затверджені наказом по університету № 1808-С від «29» травня 2025 р.
2. Термін подання студентом проєкту «13» червня 2025 р.
3. Вихідні дані до проєкту: див. Технічне завдання.
4. Зміст пояснювальної записки:
 - огляд та аналіз існуючих рішень;
 - обґрунтування вибору засобів розроблення;
 - розроблення вебзастосунку;
 - аналіз розробленого вебзастосунку.
5. Перелік обов'язкового графічного матеріалу:
 - схема бази даних системи (креслення);
 - діаграма послідовностей дій користувача (креслення);
 - структурна схема вебзастосунку (плакат);

– структура вебсайту (плакат).

6. Консультанти розділів проєкту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Нормоконтроль	Онай М.В., доцент		

7. Дата видачі завдання «31» жовтня 2024 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1.	Вивчення літератури за тематикою проєкту	13.11.2024	
2.	Розроблення та узгодження технічного завдання	25.11.2024	
3.	Розроблення структури вебзастосунку	15.12.2024	
4.	Підготовка матеріалів першого розділу дипломного проєкту	30.12.2024	
5.	Розроблення дизайну сторінок та графічних елементів	03.02.2025	
6.	Підготовка матеріалів другого розділу дипломного проєкту	19.02.2025	
7.	Програмна реалізація вебзастосунку	10.03.2025	
8.	Тестування вебзастосунку	17.03.2025	
9.	Підготовка матеріалів третього розділу дипломного проєкту	30.03.2025	
10.	Підготовка матеріалів четвертого розділу дипломного проєкту	12.04.2025	
11.	Підготовка графічної частини дипломного проєкту	21.04.2025	
12.	Оформлення документації дипломного проєкту	26.05.2025	

Студент

Олексій ОНАСЕНКО

Керівник проєкту

Яків ЮСИН

АНОТАЦІЯ

Даний дипломний проєкт присвячено розробці вебзастосунку для онлайн створення та купівлі букетів. Основною метою проєкту є надання користувачам можливості легко і швидко формувати індивідуальні квіткові композиції, а також замовляти їх доставку без необхідності відвідування фізичних магазинів. Застосунок орієнтований на клієнтів, що шукають персоналізовані рішення для подарунків або особливих подій.

Аналіз наявних рішень на ринку дозволив виявити низку проблем: обмежений вибір готових букетів, відсутність можливості збирати композицію онлайн та складний процес замовлення. У даному проєкті ці недоліки враховано та реалізовано у вигляді інтуїтивного інтерфейсу з редактором, що дозволяє створювати букети на свій смак, переглядати їх у реальному часі та відразу оформляти покупку.

Застосунок складається з клієнтської та серверної частини. Інтерфейс адаптований до різних типів пристроїв, що забезпечує зручність користування як з комп'ютера, так і зі смартфона. Зареєстровані користувачі можуть створювати власні композиції, обирати готові букети, зберігати улюблені варіанти, оформлювати замовлення, а також переглядати історію покупок.

Розроблений вебзастосунок покликаний зробити процес дарування квітів більш доступним, персоналізованим та приємним для користувача. У поєднанні з простотою взаємодії та сучасним підходом до візуалізації – він здатен значно покращити досвід покупця та задовольнити потреби як індивідуальних клієнтів, так і квіткових салонів.

ABSTRACT

This diploma project is dedicated to the development of a web application for online creation and purchase of bouquets. The main goal of the project is to provide users with the ability to easily and quickly create individual floral compositions, as well as order their delivery without the need to visit physical stores. The application is targeted at clients who are looking for personalized solutions for gifts or special occasions.

The analysis of existing market solutions made it possible to identify a number of problems: a limited selection of ready-made bouquets, the absence of the possibility to assemble a composition online, and a complicated ordering process. These shortcomings were taken into account in this project and addressed through the implementation of an intuitive interface with an editor that allows users to create bouquets to their taste, preview them in real time, and immediately place an order.

The application consists of a client-side and a server-side part. The interface is adapted for different types of devices, which ensures convenient use both from a computer and from a smartphone. Registered users can create their own compositions, choose ready-made bouquets, save favorite options, place orders, and view their purchase history.

The developed web application is intended to make the process of giving flowers more accessible, personalized, and pleasant for the user. Combined with ease of interaction and a modern approach to visualization, it is capable of significantly improving the customer experience and meeting the needs of both individual clients and flower shops.

ДП.045440-01-90 Вебзастосунок для онлайн створення та купівлі букетів з квітів.
Відомість проєкту

Позначення	Найменування	Кіл-ть	Примітка
	Документація проєкту		
ДП.045440-02-91	Вебзастосунок для онлайн	4	
	створення та купівлі букетів		
	з квітів. Технічне завдання		
ДП.045440-03-81	Вебзастосунок для онлайн	62	
	створення та купівлі букетів		
	з квітів. Пояснювальна		
	записка		
ДП.045440-04-51	Вебзастосунок для онлайн	4	
	створення та купівлі букетів		
	з квітів. Програма та		
	методика тестування		
ДП.045440-05-34	Вебзастосунок для онлайн	10	
	створення та купівлі букетів		
	з квітів. Керівництво		
	користувача		
ДП.045440-06-99	Вебзастосунок для онлайн	1	
	створення та купівлі букетів		
	з квітів. Схема бази даних.		
	ER-діаграма		
ДП.045440-07-99	Вебзастосунок для онлайн	1	
	створення та купівлі букетів		
	з квітів. Діаграма		
	послідовності для		
	замовлення букета. Діаграма		
	послідовності		

[illegible]

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОНЛАЙН СТВОРЕННЯ ТА КУПІВЛІ
БУКЕТІВ З КВІТІВ

Технічне завдання

ДП.045440-02-91

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ОНАСЕНКО

ЗМІСТ

1. Найменування та галузь застосування.....	3
2. Підстава для розроблення	3
3. Призначення розробки.....	3
4. Вимоги до програмного продукту	3
5. Вимоги до проєктної документації	4
6. Етапи проєктування	4
7. Порядок тестування розробки.....	4

1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ ЗАСТОСУВАННЯ

Назва розробки: Вебзастосунок для онлайн створення та купівлі букетів з квітів.

Галузь застосування: інформаційні технології.

2. ПІДСТАВА ДЛЯ РОЗРОБЛЕННЯ

Підставою для розроблення є завдання на дипломне проєктування, затверджене кафедрою програмного забезпечення комп'ютерних систем Національного технічного університету України «Київський політехнічний інститут імені Ігоря Сікорського» (КПІ ім. Ігоря Сікорського).

3. ПРИЗНАЧЕННЯ РОЗРОБКИ

Розробка вебзастосунку призначена для створення зручного інструменту для онлайн створення унікальних букетів з квітів та їх подальшого замовлення.

4. ВИМОГИ ДО ПРОГРАМНОГО ПРОДУКТУ

Вебзастосунок повинен забезпечувати такі основні функції:

- 1) можливість динамічного оновлення вмісту вебсторінок;
- 2) можливість реєстрації у сервісі;
- 3) можливість авторизації;
- 4) можливість переглядати каталог квітів, згрупованих за категоріями;
- 5) можливість створювати власні букети за допомогою інтерактивного конструктора;
- 6) можливість додавати букети до кошика та оформлювати замовлення;

7) можливість переглядати історію замовлень.

5. ВИМОГИ ДО ПРОЄКТНОЇ ДОКУМЕНТАЦІЇ

У процесі виконання проєкту повинна бути розроблена наступна документація:

- 1) пояснювальна записка;
- 2) програма та методика тестування;
- 3) керівництво користувача;
- 4) креслення:
 - «Діаграма послідовності для замовлення букета»;
 - «Схема бази даних системи. ER-діаграма».

6. ЕТАПИ ПРОЄКТУВАННЯ

Вивчення літератури за тематикою роботи.....	13.11.2024
Розроблення та узгодження технічного завдання.....	25.11.2024
Розроблення структури вебзастосунку.....	15.12.2024
Розроблення дизайну сторінок та графічних елементів.....	03.02.2025
Програмна реалізація вебзастосунку.....	10.03.2025
Тестування вебзастосунку	17.03.2025
Підготовка матеріалів текстової частини проєкту.....	12.04.2025
Підготовка матеріалів графічної частини проєкту.....	21.04.2025
Оформлення технічної документації проєкту.....	26.05.2025

7. ПОРЯДОК ТЕСТУВАННЯ РОЗРОБКИ

Тестування розробленого програмного продукту виконується відповідно до «Програми та методики тестування».

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

«ЗАТВЕРДЖЕНО»

Завідувач кафедри

_____ Євгенія СУЛЕМА

«___» _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОНЛАЙН СТВОРЕННЯ ТА КУПІВЛІ
БУКЕТІВ З КВІТІВ

Пояснювальна записка

ДП.045440-03-81

«ПОГОДЖЕНО»

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ОНАСЕНКО

ЗМІСТ

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ	4
ВСТУП	7
1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ	9
1.1. Аналіз аналогічних програмних рішень	9
1.2. Порівняльний аналіз існуючих рішень	13
1.3. Загальні вимоги до програмного застосунку	15
1.4. Формування вимог до вебзастосунку.....	16
1.5. Висновки до розділу	16
2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ	18
2.1. Вибір технологій розробки серверної частини	18
2.2. Вибір технологій розробки користувацької частини	21
2.3. Вибір СКБД.....	26
2.4. Висновки до розділу	30
3. РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ	32
3.1. Загальний опис програми	32
3.2. Вибір архітектури вебзастосунку	33
3.3. Аналіз функціональних вимог до вебзастосунку	38
3.4. Структурна організація застосунку	39
3.5. Особливості реалізації серверної частини застосунку	41
3.6. Особливості реалізації клієнтської частини застосунку	44
3.7. Структура бази даних	45
3.8. Висновки до розділу	49
4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ	51
4.1. Аналіз реалізованого вебзастосунку	51
4.2. Тестування реалізованого застосунку	52
4.3. Рекомендації щодо подальшого вдосконалення	54
4.4. Висновки до розділу	55
ВИСНОВКИ.....	57

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРИНИХ ДЖЕРЕЛ.....	59
ДОДАТКИ.....	62

СПИСОК ТЕРМІНІВ, СКОРОЧЕНЬ ТА ПОЗНАЧЕНЬ

Вебзастосунок – програмне забезпечення, яке виконується у браузері та забезпечує взаємодію користувача з інтерфейсом через інтернет.

Флористика – це мистецтво створення композицій із живих, сухих або штучних квітів і рослин.

REST API – архітектурний стиль для розробки вебсервісів, який забезпечує обмін даними між клієнтом і сервером через HTTP.

HTTP (HyperText Transfer Protocol) – протокол передачі даних між клієнтом і сервером в Інтернеті.

SPA (Single Page Application) – тип вебзастосунку, що завантажується один раз і динамічно оновлює вміст без повного перезавантаження сторінки.

MPA (Multi Page Application) – архітектурний підхід до розробки вебзастосунків, при якому кожна сторінка є окремим HTML-документом і завантажується з сервера при переходах між сторінками.

AJAX (Asynchronous JavaScript and XML) – технологія, що дозволяє надсилати та отримувати дані з сервера у фоновому режимі без повного перезавантаження вебсторінки.

SSR (Server-Side Rendering) – спосіб рендерингу вебсторінок, при якому HTML генерується на сервері та надсилається клієнту вже в готовому вигляді, що покращує SEO та зменшує час першого завантаження.

SSG (Static Site Generation) – підхід до створення вебсайтів, при якому сторінки генеруються на етапі компіляції й розміщуються як статичні файли, що забезпечує високу швидкість завантаження та надійність.

SEO (Search Engine Optimization) – сукупність методів і технік, спрямованих на підвищення видимості вебсайту в результатах пошукових систем шляхом оптимізації його структури, контенту та технічних характеристик

SRF (Cross-Site Request Forgery) – тип атаки, при якій зловмисник змушує користувача виконати небажану дію на вебзастосунку, в якому той вже автентифікований, без його відома.

XSS (Cross-Site Scripting) – вразливість безпеки, що дозволяє вставляти шкідливий скрипт на вебсторінку, який виконується в браузері інших користувачів.

Frontend – частина вебзастосунку, з якою взаємодіє користувач, включає інтерфейс та логіку відображення.

Backend – серверна частина застосунку, що відповідає за обробку даних, логіку роботи, зберігання інформації та взаємодію з базами даних.

JSON (JavaScript Object Notation) – текстовий формат обміну даними, що використовується для зберігання та передачі структурованої інформації між клієнтом і сервером у вебзастосунках.

RDBMS (Relational Database Management System) – система керування базами даних, яка зберігає інформацію у вигляді взаємозв'язаних таблиць та забезпечує роботу з даними за допомогою мови SQL.

ACID – набір властивостей (атомарність, узгодженість, ізолюваність, довговічність), які гарантують надійне виконання транзакцій у системах керування базами даних.

JIT (Just-In-Time compilation) – технологія динамічної компіляції, яка перетворює байт-код у машинний код під час виконання програми, що дозволяє підвищити продуктивність виконання.

MVP (Minimum Viable Product) – мінімально життєздатний продукт, який містить лише основні функції, достатні для задоволення перших користувачів і збору зворотного зв'язку для подальшого розвитку.

MVC (Model-View-Controller) – архітектурний шаблон проектування програмного забезпечення, який розділяє застосунок на три компоненти: модель (обробка даних), подання (інтерфейс користувача) та контролер (керування логікою взаємодії між моделлю та поданням).

ORM (Object-Relational Mapping) – технологія, що забезпечує зв'язок між об'єктно-орієнтованим кодом і реляційною базою даних, дозволяючи працювати з даними у вигляді об'єктів без написання SQL-запитів.

JWT (JSON Web Token) – стандарт передачі даних між клієнтом і сервером у вигляді компактного та безпечного токена, що використовується для автентифікації та авторизації в вебзастосунках.

SEO (Search Engine Optimization) – комплекс заходів, спрямованих на підвищення видимості вебзастосунку в пошукових системах шляхом оптимізації його структури, контенту та технічних характеристик.

ВСТУП

У світі, де цифрові технології стрімко перетворюють повсякденність, немає нічого дивного в тому, що все більше звичних нам речей переходять в онлайн. Суспільство поступово звикає до високого рівня зручності, оперативності та цілодобової доступності послуг з будь-якої точки світу. Особливо помітним цей процес є у сфері торгівлі, де використання мобільних пристроїв, інтернету та вебзастосунків суттєво змінює формат взаємодії між бізнесом і споживачем.

Флористика, як одна з галузей, що поєднує естетичну складову з підприємницькою діяльністю, не є винятком. Традиційний формат придбання квіткових композицій – особистий візит до фізичної крамниці – поступово поступається місцем онлайн-замовленням, що дозволяють економити час і забезпечувати гнучкість у виборі. Водночас важливо розуміти, що квіти – це не лише товар, а й носій емоційного посилу, інструмент комунікації в особливо значущі життєві моменти: від урочистостей до періодів скорботи.

У цьому контексті надзвичайно актуальним постає завдання створення вебзастосунку, який би поєднував зручність цифрового сервісу з емоційною складовою флористики. Йдеться не лише про надання користувачам доступу до каталогу готових букетів, а про можливість самостійного, інтуїтивно зрозумілого формування індивідуальних композицій у режимі реального часу. Такий підхід дозволяє не лише підвищити рівень персоналізації, а й стимулює креативність користувачів, розширюючи межі взаємодії зі сервісом.

З боку бізнесу впровадження подібного рішення відкриває низку переваг. Зокрема, мова йде про автоматизацію обробки замовлень, оптимізацію логістичних процесів, ефективніше управління асортиментом та зниження навантаження на персонал. Як наслідок – підвищення якості

обслуговування, формування лояльної клієнтської бази та посилення конкурентних позицій на ринку.

Запропонований дипломний проєкт спрямований на розробку сучасного вебзастосунку, який забезпечить користувачам інструмент для створення унікальних квіткових композицій, водночас надаючи бізнесу ефективні засоби керування замовленнями, обробкою та доставкою продукції. Результатом реалізації стане функціональна, масштабована та візуально приваблива система, що відповідатиме як очікуванням клієнтів, так і стратегічним потребам підприємств флористичного сектору.

Таким чином, реалізація проєкту «Вебзастосунок для онлайн створення та купівлі букетів» є актуальною відповіддю на цифрові виклики часу, поєднуючи у собі естетику флористики та інноваційні рішення, що відповідають очікуванням сучасного користувача.

1. ОГЛЯД ІСНУЮЧИХ ПРОГРАМНИХ РІШЕНЬ

1.1. Аналіз аналогічних програмних рішень

1.1.1. *Interflora*

Одним із найбільш відомих сервісів у сфері доставки квітів є міжнародна платформа Interflora [1]. Цей сервіс об'єднує квіткові магазини з різних країн, дозволяючи здійснювати замовлення букетів у більш ніж 140 країнах світу. Основними перевагами платформи є широке географічне охоплення, наявність професійної кур'єрської служби, а також співпраця з локальними флористами, що дозволяє забезпечити швидку та якісну доставку [2]. Крім того, користувачам пропонується широкий асортимент готових букетів і супутніх товарів, що створює відчуття зручності й впевненості у виборі.

Однак, попри ряд переваг, сервіс має й певні обмеження. Наприклад, можливості для персоналізації букетів через онлайн-інтерфейс досить обмежені – клієнти здебільшого обирають з-поміж фіксованих шаблонів, що знижує індивідуальність подарунка. Також важливим недоліком є висока вартість доставки, особливо у випадках міждержавних замовлень. Додатково слід зазначити, що інтерфейс платформи не завжди адаптований до мови конкретного користувача, що може ускладнювати процес замовлення для деяких клієнтів. Нижче на рисунку 1.1 зображено головну сторінку вебзастосунку Interflora.

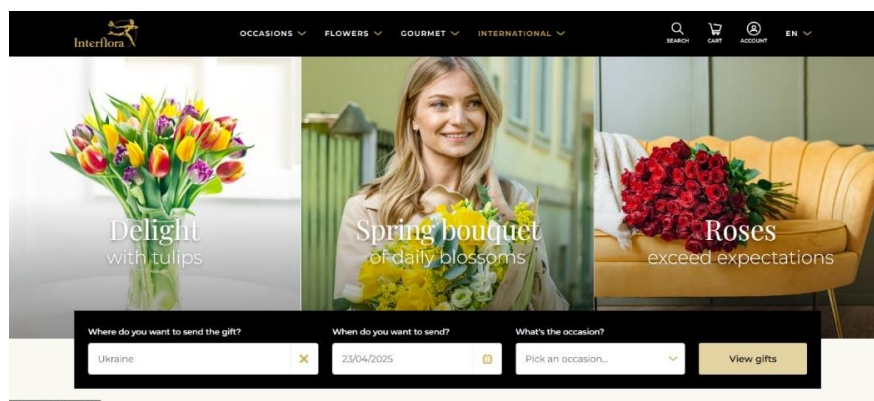


Рис. 1.1. Головна сторінка вебзастосунку Interflora

Interflora – зручне рішення для міжнародних замовлень, однак воно не надає широких можливостей персоналізації букетів, що може бути мінусом для користувачів, які прагнуть створити унікальну композицію.

1.1.2. Flowers.ua

Flowers.ua є одним із провідних українських онлайн-сервісів доставки квітів, який спеціалізується на швидкому оформленні та оперативній доставці букетів по всій території України. Сайт пропонує зручну навігацію завдяки розподілу товарів за категоріями, зокрема за типом квітів, подією чи додатковими подарунками [3].

Особливістю сервісу є інтерфейс, доступний трьома мовами – українською, російською та англійською, що дозволяє комфортно користуватися платформою широкому колу клієнтів. Серед сильних сторін Flowers.ua варто відзначити можливість здійснення доставки в той самий день у багатьох містах, що є важливою перевагою для користувачів, які прагнуть зробити приємне близьким у короткі терміни. Додаткову цінність додає опція включення до замовлення супутніх подарунків, таких як солодощі чи м'які іграшки. Відгуки клієнтів, розміщені безпосередньо на сторінках товарів, сприяють прозорості та підвищенню довіри до сервісу.

Окрему увагу Flowers.ua приділяє присутності в соціальних мережах, зокрема в Instagram [4]. Офіційна сторінка компанії активно оновлюється та виступає як вітрина найкращих флористичних композицій, прикладів доставок і сезонних акцій. Завдяки естетично привабливому контенту, інтерактивним сторіз та відгукам клієнтів, Instagram-акаунт сприяє підвищенню впізнаваності бренду та залученню нових клієнтів через візуальне представлення послуг.

Водночас платформа має і певні недоліки. Зокрема, персоналізація букетів обмежується вже готовими варіантами, що знижує рівень персоналізації замовлення. Також періодично надходять скарги від користувачів щодо розбіжностей між зображенням букета на сайті та його

фактичним виглядом після доставки. Окрім того, у періоди високої завантаженості, зокрема під час свят, клієнтська підтримка може відповідати з помітними затримками, що негативно впливає на загальний користувацький досвід.

Нижче на рисунку 1.2 зображено головну сторінку вебзастосунку Flowers.ua.

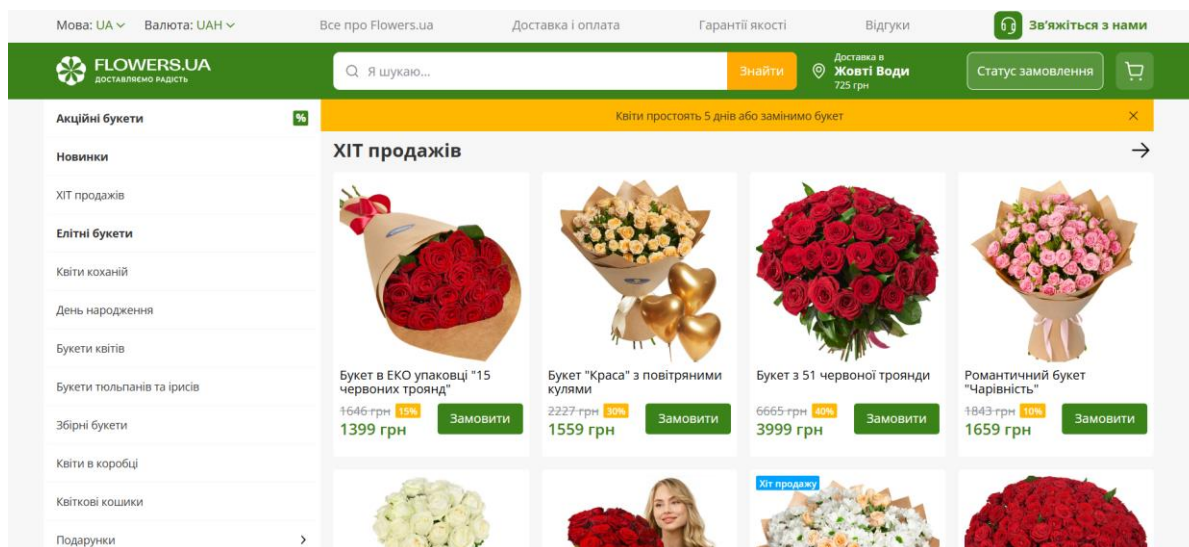


Рис. 1.2. Головна сторінка вебзастосунку Flowers.ua

Flowers.ua добре підходить для швидкого замовлення букета з доставкою, проте сервіс не дає широких можливостей для індивідуального складання композицій, немає можливості побачити букет після замовлення та мала різновидність квітів.

1.1.3. Sunrose

Sunrose – це український сервіс доставки квітів, який динамічно розвивається та робить акцент на високій якості флористичних композицій і рівні клієнтського обслуговування. Однією з ключових переваг Sunrose є орієнтація на сучасний дизайн композицій, що відповідає останнім тенденціям у флористиці. Онлайн-чат із флористом забезпечує швидку комунікацію та дозволяє уточнити деталі замовлення, що підвищує рівень

персоналізації послуги. Крім того, сервіс надсилає фото готового букета перед його доставкою, що створює додаткову прозорість та впевненість клієнта у відповідності замовлення [5].

Втім, певні обмеження сервісу все ще помітні. На момент аналізу географія доставки залишається обмеженою, охоплюючи переважно великі міста. Асортимент товарів також є дещо скромнішим у порівнянні з більш масштабними гравцями ринку. Крім цього, наразі відсутній мобільний застосунок, що може знижувати зручність використання сервісу для мобільних користувачів. Хоча існує можливість індивідуального підбору квітів, функція гнучкої персоналізації композицій безпосередньо через сайт поки що не реалізована.

Нижче на рисунку 1.3 зображено головну сторінку вебзастосунку Sunrose.

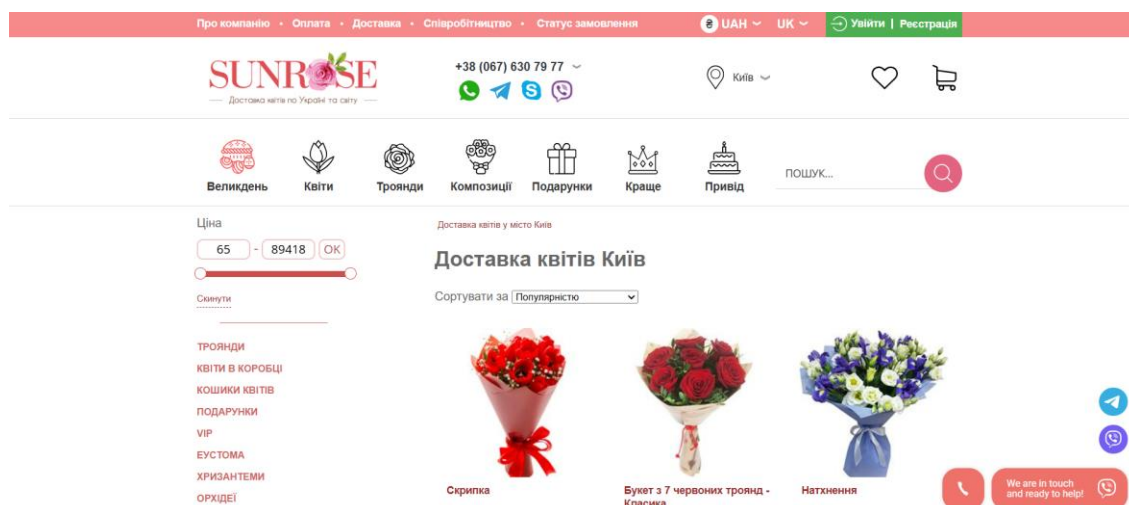


Рис. 1.3. Головна сторінка вебзастосунку Sunrose

Sunrose — зручний варіант для користувачів, які цінують індивідуальність і сучасний підхід у флористиці, проте поки що сервіс поступається більшим конкурентам за охопленням та функціональністю.

1.2. Порівняльний аналіз існуючих рішень

У результаті проведеного аналізу вдалося визначити ключові функціональні можливості, переваги та недоліки існуючих на ринку сервісів доставки квітів. На основі зібраної інформації сформовано порівняльну таблицю, яка дозволяє візуально оцінити особливості кожного з розглянутих застосунків (табл. 1.1).

Таблиця 1.1

Порівняння можливостей розглянутих застосунків

Застосунок	Interflora	Flowers.ua	Sunrose
Функціональні можливості			
Онлайн конструктор букетів	—	—	—
Можливість додати листівку або подарунок до замовлення	+	+	—
Міжнародна доставка	+	—	—
Кур'єрська доставка по Україні	+	+	—
Розширений вибір подарункових наборів	+	+	—
Система бонусів або програма лояльності	+	—	—
Підтримка мобільних пристроїв	+	+	+
Відгуки та рейтинг товарів	—	+	+
Можливість оплати онлайн	+	+	+
Інтеграція з соцмережами	—	+	+
Персоналізовані пропозиції / рекомендації	+	—	—
Підтримка 24/7	+	+	+

Порівняльна таблиця демонструє, що кожен із розглянутих вебзастосунків доставки квітів має свої сильні та слабкі сторони. Безумовно, усі сервіси виконують свою основну функцію – забезпечують зручне замовлення квіткових композицій онлайн, однак спостерігаються і деякі спільні проблеми.

Перш за все, жоден сервіс не пропонує можливість конструктору букетів онлайн, що значно обмежує персоналізацію замовлень для користувачів інших платформ. В умовах сучасного ринку важливо надати клієнтам можливість самостійно формувати букет з урахуванням бюджету, смаків та події.

Ще однією проблемою є відсутність багатомовної підтримки в частині застосунків, що ускладнює доступ для іноземних клієнтів або тих, хто не володіє українською мовою. А це, у свою чергу, зменшує потенційну аудиторію.

Програма лояльності або бонусна система представлена лише в одному сервісі, хоча саме така функціональність могла би сприяти утриманню постійних клієнтів та збільшенню частоти замовлень.

Також більшість сервісів обмежується базовою функціональністю, без реалізації персоналізованих рекомендацій, що дозволяють користувачу швидше знаходити релевантні пропозиції. Наявність такої функції значно підвищила б якість взаємодії з сайтом і допомогла б збільшити конверсію замовлень.

Ще одним важливим недоліком є відсутність інтерактивної взаємодії з іншими користувачами, наприклад, можливості залишити коментарі, оцінки чи фото отриманих букетів. Це зменшує рівень довіри нових клієнтів до сервісу.

Загальною проблемою для всіх проаналізованих вебзастосунків є відсутність розширеної персоналізації, обмежена взаємодія з користувачем та часткова залежність від зовнішніх каналів комунікації, що ускладнює комплексну підтримку клієнта в режимі 24/7.

1.3. Загальні вимоги до програмного застосунку

Жоден із розглянутих вебзастосунків для онлайн створення та купівлі букетів не задовольняє всі функціональні вимоги, які могли б повністю вирішити основні потреби сучасних користувачів. Кожен із сервісів має певні сильні сторони, однак в жодному з них не реалізований повноцінний набір можливостей, що забезпечували б гнучкість, персоналізацію, зручність та ефективну взаємодію з клієнтом.

У зв'язку з цим, на даному етапі доцільно визначити та сформулювати чіткий перелік функціональних можливостей, які мають бути реалізовані у розроблюваному програмному забезпеченні для досягнення конкурентної переваги на ринку та максимального задоволення потреб користувачів.

Для представлення проблем складено дерево проблем (рис. 1.4).

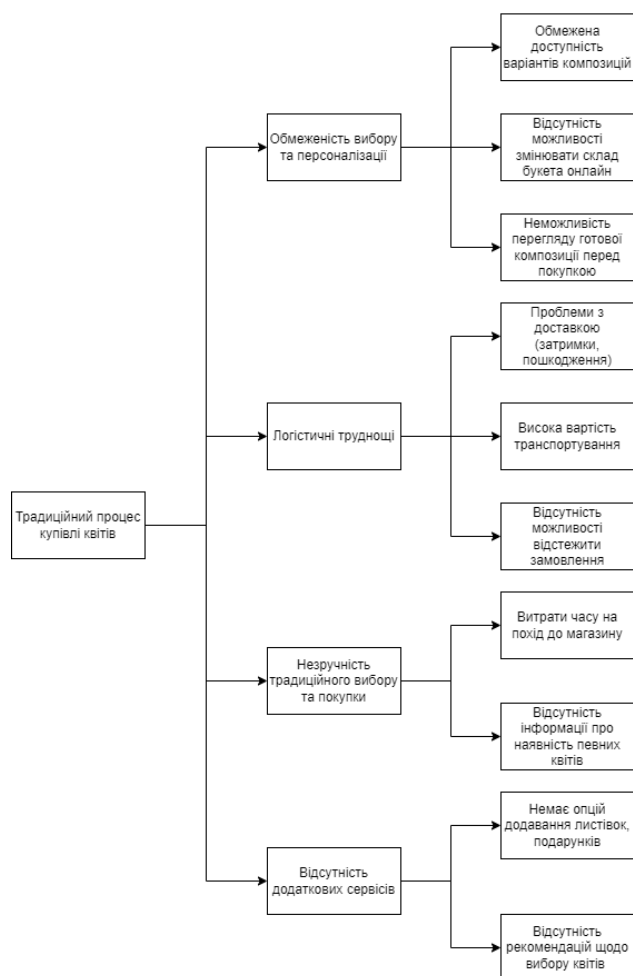


Рис. 1.4. Дерево проблем, які вирішує розроблене ПЗ

1.4. Формування вимог до вебзастосунку

Визначимо вимоги до функціональності, продуктивності та зручності користування майбутнім сервісом.

- Доступність – застосунок повинен працювати в будь-якому сучасному веббраузері без потреби встановлення додаткового програмного забезпечення.
- Зрозумілість – інтерфейс і функціональність мають бути інтуїтивно зрозумілими для покупців, флористів та адміністраторів.
- Конструктор букетів – користувачі повинні мати змогу створювати власні квіткові композиції, обираючи різні типи квітів, кольори та аксесуари.
- Збереження та редагування – користувачі повинні мати можливість зберігати створені букети, редагувати їх та експортувати у форматах .png та .svg.
- Онлайн-замовлення та оплата – необхідно забезпечити можливість оформлення замовлення, вибору способу доставки та оплати онлайн.
- Доступність у інтернет-версії – застосунок повинен бути доступним як вебсервіс без потреби встановлення на пристрій користувача.

1.5. Висновки до розділу

Під час написання першого розділу дипломного проекту було проведено аналіз існуючих рішень на ринку онлайн-сервісів з доставки квітів. У ході огляду їхніх функціональних можливостей та технічних реалізацій вдалося виявити ключові аспекти, які безпосередньо впливають на зручність користування та задоволення потреб клієнтів. Зібрані дані були систематизовані у вигляді порівняльної таблиці функціональних характеристик.

Аналіз отриманої інформації дозволив дійти висновку, що жоден із розглянутих сервісів не забезпечує повного охоплення потреб цільової

аудиторії. Більшість рішень мають обмежену функціональність, застарілий інтерфейс або відсутність гнучкого конструктора букетів. Це свідчить про наявність простору для розробки нового вебзастосунку, який, спираючись на виявлені переваги та враховуючи наявні недоліки, зможе запропонувати більш якісний користувацький досвід.

2. ОБГРУНТУВАННЯ ВИБОРУ ЗАСОБІВ РОЗРОБЛЕННЯ

2.1. Вибір технологій розробки серверної частини

Для вибору технологій для розробки вебзастосунку доцільно звернутися до статистики. За даними порталу W3Techs (рис. 2.1), можемо виділити три популярні рішення – PHP, Ruby (Ruby on Rails) та ASP.NET Core.

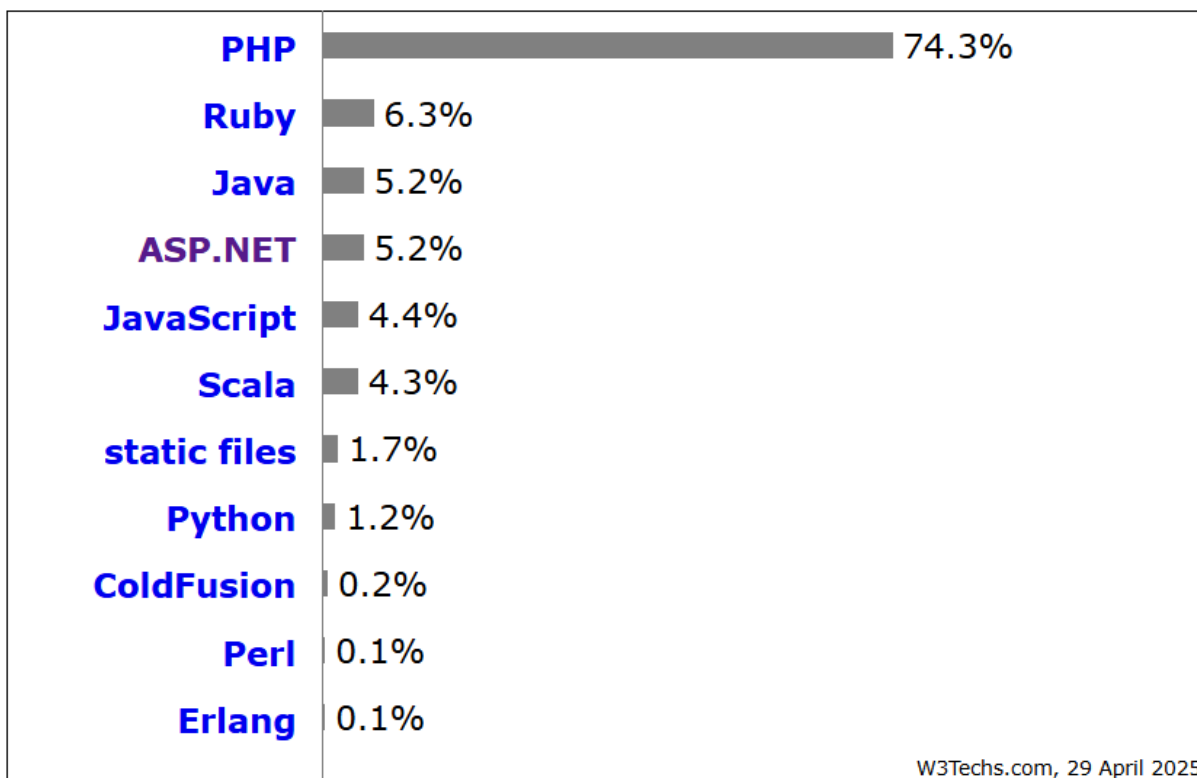


Рис. 2.1. Статистика вебсайтів backend яких використовує різні мови програмування [6]

2.1.1. PHP

PHP є однією з найстаріших та найпопулярніших мов програмування для серверної частини. Станом на 2025 рік більше 74% вебсайтів, серверна частина яких відома, використовують саме PHP. Це обумовлено його простотою, низьким порогом входу, доступністю хостингів і величезною спільнотою, яка накопичила велику кількість бібліотек та готових CMS-систем таких як WordPress чи Joomla.

Проте разом із перевагами PHP має і ряд недоліків, які роблять його не найкращим вибором для сучасних застосунків:

- Застаріла структура багатьох проєктів. Значна частина популярних сайтів використовує старі версії PHP (5.x або 7.x), що не відповідають сучасним вимогам до безпеки та продуктивності.
- Відсутність строгих стандартів. У PHP-кодi легко зробити помилку, а підтримка великих проєктів часто ускладнена через неструктурованість.
- Проблеми з масштабуванням. Попри існування сучасних фреймворків, таких як Laravel або Symfony, PHP не забезпечує такої продуктивності та керованості, як більш сучасні платформи.
- Невисока продуктивність у порівнянні з компільованими або JIT-технологіями.

З огляду на це, PHP доцільно використовувати для простих сайтів або невеликих корпоративних застосунків, але для дипломного проєкту, що передбачає складну логіку, гнучку архітектуру та подальший розвиток, ця технологія не є оптимальною.

2.1.2. Ruby on Rails

Фреймворк Ruby on Rails (або просто Rails), побудований на мові Ruby, став відомим завдяки гаслу «розробка в задоволення» та принципам «Convention over configuration» [7] і DRY (Don't Repeat Yourself). Він дозволяє швидко створювати прототипи та мінімізує обсяг шаблонного коду. Rails активно використовували у стартапах 2010-х років (GitHub, Airbnb тощо), і він зарекомендував себе як ефективний інструмент для швидкої розробки MVP.

Переваги Ruby on Rails:

- Швидкість розробки завдяки генераторам, міграціям, scaffolding тощо.
- Зрозуміла архітектура MVC, яка дисциплінує структуру проєкту.

- Багата екосистема гемів (бібліотек), що покривають більшість типових потреб.

Однак із розвитком проєктів починають проявлятися недоліки:

- Складність масштабування. Rails погано масштабується в умовах високих навантажень через специфіку інтерпретованої мови Ruby.
- Менша продуктивність у порівнянні з ASP.NET Core [8] чи Java.
- Скорочення спільноти. Популярність фреймворку поступово зменшується, оновлення виходять рідше, а підтримка може бути ускладненою.
- Складніший хостинг і специфічні вимоги до серверного середовища.

Таким чином, хоча Ruby on Rails підходить для швидкої розробки невеликих проєктів, для дипломного застосунку з розрахунком на розвиток, навантаження та безпечну інтеграцію з зовнішніми сервісами він не є найкращим рішенням.

2.1.3. ASP.NET Core

ASP.NET Core – це технологія веброзробки від компанії Microsoft, яка вже багато років використовується для створення високонадійних, продуктивних і захищених вебсистем. Завдяки постійній підтримці з боку Microsoft, ASP.NET Core залишається актуальною платформою, особливо в корпоративному середовищі.

Серед головних переваг ASP.NET Core:

- Висока продуктивність завдяки компіляції коду.
- Підтримка різних архітектур – як монолітної, так і мікросервісної [9].
- Вбудовані механізми безпеки, включаючи автентифікацію, авторизацію, захист від XSS [10] та CSRF [11].
- Можливість роботи з різними СКБД, зокрема PostgreSQL, SQL Server, MySQL.

- Сумісність з іншими сервісами Microsoft (Azure, Active Directory тощо).
- Зручна інтеграція з REST API [12], сторонніми платіжними та логістичними сервісами.

2.1.4. Підсумок вибору технології розробки серверної частини

Порівняльний аналіз показує, що хоча PHP та Ruby on Rails мають свої переваги, ASP.NET Core є найкращим вибором для реалізації дипломного проєкту. Ця технологія поєднує в собі сучасний підхід до веброзробки, високу продуктивність і підтримку сучасних вимог до безпеки. Завдяки широким можливостям розширення та інтеграції ASP.NET Core забезпечить стабільну роботу вебзастосунку та дозволить реалізувати всі необхідні функції, включаючи реєстрацію користувачів, обробку замовлень, управління контентом і взаємодію з базою даних.

2.2. Вибір технологій розробки користувацької частини

Сучасна розробка вебзастосунків неможлива без використання JavaScript або одного з фреймворків, побудованих на його основі. Згідно зі статистикою порталу W3Techs (рис. 2.2) станом на 29 квітня 2025 року, JavaScript використовується на 98.8% усіх вебсайтів, де відома технологія клієнтської частини. Цей показник є красномовним свідченням того, що JavaScript став фактичним стандартом у сфері розробки клієнтської частини.

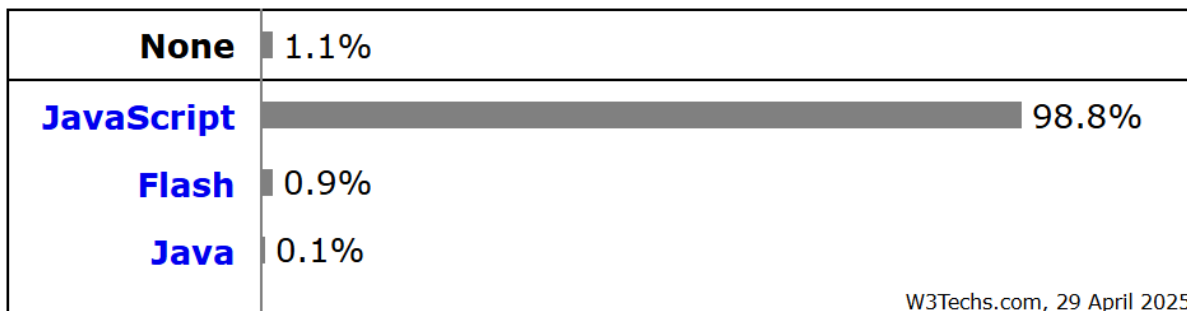


Рис. 2.2. Статистика вебсайтів frontend яких використовує різні мови програмування [13]

Попри домінування JavaScript як мови, самотійно вона рідко використовується в складних проєктах. Для зручності та ефективності створення інтерфейсів розробники все частіше вдаються до використання фреймворків, які значно пришвидшують роботу та забезпечують кращу структуру коду.

2.2.1. React

React – це бібліотека JavaScript з відкритим кодом, яка використовується для розробки інтерфейсів користувача на основі компонентів. Вона була розроблена компанією Facebook у 2013 році та нині підтримується спільнотою з відкритим кодом і самою компанією Facebook.

Це найпоширеніша бібліотека на основі JavaScript завдяки зручності створення вебзастосунків. Крім побудови інтерфейсу користувача, вона також надає низку додаткових можливостей, таких як Flux та React Native.

Однією з найбільших переваг використання React є можливість створювати як веб, так і мобільні застосунки на спільній кодовій базі за допомогою фреймворку React Native. Першим автором бібліотеки був Джордан Волке у 2011 році [14].

Кожен компонент у застосунку на React виконує роль будівельного блоку, що робить застосунок високорозширюваним. Повторне використання компонентів у JavaScript зменшує час розробки та складність.

Переваги використання React:

- Повторно використовувані компоненти. Під час розробки застосунків на React зменшується складність і обсяг коду завдяки можливості повторного використання компонентів. Це забезпечує більшу функціональність, чистоту коду та легкість у його підтримці.
- Продуктивність. React не залежить від традиційного DOM і використовує структуру на JavaScript – віртуальний DOM. Це покращує продуктивність і збільшує швидкість роботи програм.

- Низький поріг входу. React випереджає конкурентів завдяки простоті у вивченні. Web-розробникам, які щойно освоїли HTML та JavaScript, легко почати працювати з React.

Обмеження використання React:

- Темп розвитку. Оскільки React є найпопулярнішою бібліотекою JavaScript, вона часто оновлюється, що змушує розробників постійно вивчати нові концепції та пристосовуватись до темпу змін у React.
- Вивчення JSX. Для початку роботи з React не потрібно попередніх знань, однак JSX може здатися складним для багатьох розробників. Крім JSX, деякі скаржаться на використання вбудованого скриптингу (inline scripting) у React, що також вважається незручним і трудомістким.

2.2.2. *Vue.js*

Vue.js – це прогресивний фреймворк з відкритим кодом, який здобув широку популярність завдяки створенню односторінкових застосунків (SPA) [15]. Vue.js був випущений у 2014 році та привернув увагу розробників завдяки зрозумілому інтерфейсу та простоті у вивченні.

Навчання Vue.js здавалося простішим порівняно з більшістю інших технологій на той час. Vue орієнтується на початківців-розробників, допомагаючи їм створювати динамічні вебзастосунки без попереднього складного навчання.

Переваги Vue.js:

- Коротка крива навчання. У React і Angular потрібно володіти, відповідно, JavaScript та TypeScript. Vue же є дружнім до новачків і не вимагає попередніх знань.
- Розмір проєкту. Компактність проєкту на Vue має низку переваг, зокрема позитивно впливає на SEO, адже Google Search Console не

показує важкі вебсайти у верхніх рядках результатів, якщо ті повільно завантажуються.

- Форуми та спільнота. Наявність активної спільноти та форумів є важливим фактором при виборі технології, адже це сприяє легшому навчанню та подальшій підтримці у роботі з фреймворком.

Недоліки Vue.js:

- Екосистема. Екосистема відіграє важливу роль у здатності застосунків адаптуватися до різних браузерів і операційних систем. Vue має відносно вузьку екосистему, тому може не підтримувати старі версії ОС та веббраузерів.
- Недовіра серед розробників. Інші фреймворки, як-от Angular та React, мають підтримку великих корпорацій, що автоматично викликає довіру. Vue ж, не маючи такої, часто сприймається менш надійно серед розробників.

2.2.3. Angular

Angular – це повноцінний фреймворк для вебзастосунків на основі TypeScript, який є безкоштовним і має відкритий код. Angular широко використовується для створення односторінкових застосунків (SPA). Він був офіційно випущений компанією Google у 2016 році як продовження AngularJS, що з'явився ще у 2010 році [16].

На початкових етапах свого існування Angular привернув увагу багатьох розробників завдяки потужним функціям, а також через підтримку технологічного гіганта Google. Багато хто плутає Angular з AngularJS, але AngularJS – це перша версія, випущена Google, тоді як Angular – повністю оновлена версія з кращою продуктивністю та сучасною архітектурою.

Переваги використання Angular:

- Надійність у майбутньому. Оскільки Angular підтримується Google, розробники мають велику довіру до нього для створення

масштабованих застосунків, знаючи, що фреймворк буде підтримуватись у довгостроковій перспективі.

- Документація. Документація Angular детальна та добре пояснює функціональність і принципи роботи фреймворку. Крім того, кожна концепція подається з прикладами та простою мовою, що є корисним навіть для новачків.
- Масштабованість. Робота над проектом Angular у команді є дуже масштабованою, адже невеликі зміни, зроблені одним із розробників, не потребують зміни всієї структури застосунку. Крім того, кодова база залишається послідовною та легко читабельною, що підвищує ефективність розробки.

Недоліки використання Angular:

- Крива навчання. Angular вимагає знань TypeScript, який, згідно з опитуванням StackOverflow, є популярним лише серед 30% розробників. Тому вивчення Angular потребує більше зусиль порівняно з іншими фреймворками.
- Розмір проекту. Розмір проекту впливає на багато параметрів застосунку. Це особливо помітно в невеликих проектах, тоді як у великих – різниця в розмірі не суттєва, бо всі вони мають приблизно однакову вагу.

2.2.4. Підсумок вибору технології розробки користувацької частини

Отже, проаналізувавши описані вище фреймворки Angular та Vue.js, а також бібліотеку React, можемо зробити висновок, що Angular є найбільш доцільним вибором для реалізації дипломного проекту. Він надає чітку структуру, потужну модульну архітектуру, підтримує повторне використання компонентів, має вбудовані засоби для тестування та добре масштабується для командної розробки. Незважаючи на дещо складніший

поріг входу, Angular компенсує це широкими можливостями та інтеграцією інструментів на професійному рівні.

Крім того, його тісна інтеграція з ASP.NET Core на серверній частині дозволяє створити цілісну систему, у якій вони можуть працюють узгоджено. Саме це робить Angular логічним і стратегічно правильним вибором для реалізації складного та сучасного вебзастосунку.

2.3. Вибір СКБД

Згідно з рейтингом DB-Engines (рис. 2.3) за травень 2025 року, серед найбільш популярних СКБД лідерами залишаються Oracle, MySQL, Microsoft SQL Server, PostgreSQL та MongoDB. У веброзробці особливо широке застосування мають MySQL, MongoDB та PostgreSQL [17].

424 systems in ranking, May 2025

Rank			DBMS	Database Model	Score		
May 2025	Apr 2025	May 2024			May 2025	Apr 2025	May 2024
1.	1.	1.	Oracle	Relational, Multi-model ⓘ	1226.57	-4.49	-9.72
2.	2.	2.	MySQL	Relational, Multi-model ⓘ	964.98	-22.12	-118.76
3.	3.	3.	Microsoft SQL Server	Relational, Multi-model ⓘ	774.89	-10.12	-49.41
4.	4.	4.	PostgreSQL 📈	Relational, Multi-model ⓘ	674.32	+7.07	+28.77
5.	5.	5.	MongoDB 📈	Document, Multi-model ⓘ	402.51	+2.46	-19.14
6.	6.	📈 9.	Snowflake	Relational	172.01	+3.92	+50.67
7.	7.	📉 6.	Redis	Key-value, Multi-model ⓘ	152.19	-1.92	-5.61
8.	📈 9.	8.	IBM Db2	Relational, Multi-model ⓘ	126.40	+0.36	-2.06
9.	📉 8.	📉 7.	Elasticsearch	Multi-model ⓘ	123.81	-4.27	-11.54
10.	10.	10.	SQLite	Relational	117.77	+3.68	+3.45

Рис. 2.3. Статистика популярності різних СКБД [18]

2.3.1. MySQL

MySQL – це система керування реляційними базами даних з відкритим вихідним кодом, яка використовує мову структурованих запитів (SQL) для керування та обробки даних. Вона широко використовується у веброзробці завдяки своїй стабільності, високій продуктивності та підтримці багатьох платформ.

Переваги MySQL:

- Простота використання. MySQL легко встановлюється та налаштовується, навіть для користувачів із мінімальним технічним досвідом.
- Висока продуктивність. Особливо добре підходить для застосунків, де переважають операції читання даних над записом, що типово для більшості вебпроектів.
- Активна спільнота. Завдяки відкритому коду MySQL має широку базу користувачів, яка надає велику кількість ресурсів, документації та підтримки.
- Безкоштовна ліцензія. Базова версія MySQL є повністю безкоштовною, хоча також доступні комерційні варіанти з розширеними можливостями.

Недоліки MySQL:

- Обмеження при складних запитах. MySQL може демонструвати нижчу продуктивність при виконанні складних запитів, наприклад із вкладеними підзапитами, віконними функціями чи великою кількістю об'єднань.
- Менш гнучке управління транзакціями. У порівнянні з PostgreSQL, підтримка транзакцій у MySQL менш потужна, що обмежує її використання у застосунках із високими вимогами до обробки даних.
- Масштабування. Горизонтальне масштабування реалізоване лише частково, що ускладнює використання MySQL у великих або швидкозростаючих проєктах.

2.3.2. *MongoDB*

MongoDB – це система керування базами даних типу NoSQL, яка призначена для роботи з неструктурованими та напівструктурованими даними. Вона зберігає інформацію у вигляді гнучких документів, подібних

до JSON, що дозволяє реалізовувати динамічні схеми та забезпечує високу масштабованість. MongoDB відома своєю продуктивністю, простотою використання та здатністю ефективно працювати з великими обсягами даних.

Переваги MongoDB:

- Гнучкість схем. MongoDB не вимагає жорстко заданої структури даних, що дає змогу адаптувати модель даних до змінних вимог проєкту.
- Горизонтальне масштабування. Завдяки вбудованій підтримці шардингу MongoDB легко масштабується горизонтально, що особливо корисно для високонавантажених систем.
- Швидкодія. Добре підходить для сценаріїв із великою кількістю операцій запису та читання, особливо коли дані мають неоднорідну або частково структуровану природу.
- Обробка великих обсягів даних. MongoDB добре справляється з великими масивами інформації та потоками даних у реальному часі, що робить її актуальною для Big Data-рішень.

Недоліки MongoDB:

- Обмежена відповідність ACID. Система забезпечує повну підтримку транзакцій лише на рівні окремих документів, що може бути недостатньо для застосунків, де критично важлива цілісність даних у межах кількох операцій.
- Складність у реалізації складних запитів. MongoDB поступається реляційним базам даних (наприклад, PostgreSQL або MySQL) у можливостях запитів – зокрема щодо об'єднань таблиць, вкладених запитів та віконних функцій.
- Навантаження на індексацію. Часте та надмірне використання індексів може спричиняти зниження продуктивності, особливо у застосунках з великим обсягом операцій запису.

2.3.3. PostgreSQL

PostgreSQL – це система керування реляційними базами даних з відкритим вихідним кодом, яка відома своєю надійністю, розширюваністю та високим рівнем відповідності стандартам SQL.

Система підтримує розширені функції, зокрема складні запити, зовнішні ключі, тригери, уявлення (views) та збережені процедури, що робить її придатною як для невеликих проєктів, так і для великих систем обробки даних, зокрема сховищ даних.

Переваги PostgreSQL:

- Відповідність принципам ACID. Повна підтримка транзакційних властивостей ACID гарантує цілісність даних під час їх обробки.
- Підтримка складних типів даних. PostgreSQL дозволяє працювати з масивами, JSON, а також користувацькими типами, що робить його гнучким у використанні.
- Розвинені можливості запитів. Система підтримує складні SQL-запити з об'єднаннями таблиць (JOIN), віконними функціями та рекурсивними запитами.
- Розширюваність. PostgreSQL дозволяє створювати власні функції, оператори та типи даних, що надає розробникам додаткову гнучкість.
- Надійність. Завдяки ефективному управлінню транзакціями PostgreSQL вважається одним із найстабільніших рішень для критично важливих застосунків.

Недоліки PostgreSQL:

- Вища складність для початківців. Через велику кількість можливостей новачкам може бути складно швидко освоїти PostgreSQL.
- Складність горизонтального масштабування. Хоча розширення можливе, воно потребує більше налаштувань і технічних знань.

2.3.4. Підсумок вибору СКБД

Після аналізу трьох основних систем керування базами даних – MySQL, PostgreSQL та MongoDB, було вирішено використати PostgreSQL. Це рішення є найбільш вдалим з огляду на особливості майбутнього застосунку та вимоги до обробки даних.

PostgreSQL забезпечує високу надійність і повну підтримку транзакцій, що особливо важливо для збереження цілісності даних у процесах замовлення та оплати. Крім того, вона дозволяє виконувати складні SQL-запити, що дає змогу з часом реалізувати розширену аналітику або інші функціональні можливості. Система також легко масштабується, що дозволяє використовувати її не лише для невеликих, але й для зростаючих проєктів.

2.4. Висновки до розділу

У цьому розділі було проведено огляд сучасних технологій, які найчастіше використовуються для розроблення вебзастосунків. Зокрема, проаналізовано популярні фреймворки та мови для клієнтської частини і серверної логіки, а також системи керування базами даних. Мета – обрати ті інструменти, які найкраще відповідатимуть завданням та особливостям дипломного проєкту – вебзастосунку для онлайн створення та купівлі букетів.

Для клієнтської частини вибір зупинився на Angular – це повноцінний фреймворк, який «з коробки» пропонує інструменти для створення масштабованого, логічно структурованого застосунку. Оскільки інтерфейс має бути інтерактивним, швидким та зручним для користувача (наприклад, перегляд товарів, додавання до кошика, оформлення замовлення тощо), Angular забезпечує потрібну функціональність і стабільність для реалізації всіх цих сценаріїв.

Серверна частина реалізується за допомогою ASP.NET Core. Це сучасне, продуктивне середовище для створення вебсервісів, яке дозволяє

чітко організувати бізнес-логіку, швидко працювати з API та забезпечувати безпеку застосунку. На відміну від PHP, ASP.NET Core добре інтегрується зі структурованою логікою і підходить для побудови надійної серверної частини, особливо у випадках, коли застосунок повинен обробляти транзакції та взаємодіяти з реляційною базою даних. Крім того, використання ASP.NET Core у поєднанні з Angular дозволяє паралельно запускати клієнтську та серверну частини під час розробки, що суттєво пришвидшує процес.

У ролі СКБД обрано PostgreSQL. Це надійна реляційна база даних, яка добре підходить для структури інтернет-магазину – товари, клієнти, замовлення, кошик тощо. PostgreSQL забезпечує повну підтримку транзакцій і складних SQL-запитів, що важливо для безпечної обробки покупок і можливості аналітики в майбутньому.

Таким чином, технологічний стек проекту складається з Angular, ASP.NET Core та PostgreSQL. Це сучасне, надійне та практичне поєднання інструментів, яке дозволяє ефективно реалізувати всі функціональні вимоги застосунку й забезпечити хорошу основу для подальшого масштабування.

3. РОЗРОБЛЕННЯ ВЕБЗАСТОСУНКУ

3.1. Загальний опис програми

При першому вході на сторінку вебзастосунок користувачу автоматично відкривається головна сторінка. Якщо користувач раніше вже входив із цього пристрою, система це розпізнає і відкриває застосунок під його обліковим записом. У разі необхідності користувач може зареєструватися, перейшовши до сторінки «Профіль».

Головна сторінка застосунок містить популярні товари – готові композиції букетів, що користуються найбільшим попитом. З головної сторінки користувач може перейти на окрему сторінку «Готові композиції», де представлений повний каталог букетів, створених адміністраторами. Каталог можна відфільтрувати за різними критеріями, зокрема за ціною, популярністю та типами використаних квітів. Для користувачів, які бажають створити індивідуальний букет, доступна сторінка «Конструктор букета», де можна самостійно обрати квіти, обгортки, елементи декору та подарунки.

Зареєстровані користувачі мають доступ до особистого профілю, де відображається інформація, введена під час реєстрації, а також історія активності: оформлені замовлення, улюблені букети та залишені відгуки. У кошику зберігаються товари, які користувач планує придбати. Після його заповнення доступне оформлення замовлення з вибором способу доставки та оплати.

Для реалізації вебзастосунок було обрано клієнт-серверну архітектуру. Серверна частина відповідає за бізнес-логіку – обробку запитів, перевірку та збереження даних у базі, керування ролями користувачів (покупець, адміністратор) та забезпечення відповідей на дії клієнта. Клієнтська частина забезпечує взаємодію з користувачем: відображає інтерфейс, керує логікою створення букетів, надсилає запити до сервера й виводить результати у зручному вигляді.

3.2. Вибір архітектури вебзастосунку

Правильно обраний архітектурний підхід відіграє вирішальну роль у побудові ефективного вебзастосунку. Він не лише визначає структуру взаємодії між компонентами системи, але й значно впливає на простоту розробки, підтримки та майбутнього масштабування. Завдяки чіткому розподілу обов'язків між клієнтом і сервером, система стає більш керованою, модульною та гнучкою до змін. Такий підхід дозволяє легко впроваджувати нову функціональність, розширювати проєкт відповідно до зростаючих потреб користувачів, а також спрощує інтеграцію з зовнішніми сервісами.

3.2.1. Multi-Page Application

Ще не так давно, до стрімкого розвитку фреймворків на кшталт React чи Angular, більшість вебсайтів будувались за класичною моделлю багатосторінкових застосунків (Multi Page Application, або ж МРА) [19]. Кожен перехід за посиланням означав повне оновлення сторінки: браузер надсилав запит на сервер, де, скажімо, PHP-скрипт динамічно формував HTML-сторінку разом із CSS та JavaScript, а потім повертав її клієнту. При кожному переході весь документ оновлювався, попередній вміст видалявся, а новий завантажувався заново. Це спричиняло характерні «мигтіння» інтерфейсу, що були типовими для таких сайтів.

З точки зору пошукових систем це був і досі залишається один із найбільш сприятливих підходів – усі сторінки вже згенеровані на сервері, тому боти без проблем індексують їх повністю. Це забезпечує високі позиції у видачі та задоволення клієнтів, які хочуть бачити свій сайт у топі.

Однак для користувачів досвід не завжди був комфортним. Кожен новий перехід вимагав часу на повторне завантаження контенту, і навіть із кешуванням деяких елементів усе одно виникала затримка та візуальний дискомфорт через перерендеринг усієї сторінки.

Загалом архітектура МРА є простою та зрозумілою, що робить її придатною для багатьох типів вебзастосунків. Втім, вона може бути менш ефективною для складних та інтерактивних застосунків, які потребують плавнішої й динамічнішої взаємодії з користувачем.

На рисунку 3.1 наведено діаграму життєвого циклу МРА застосунку.

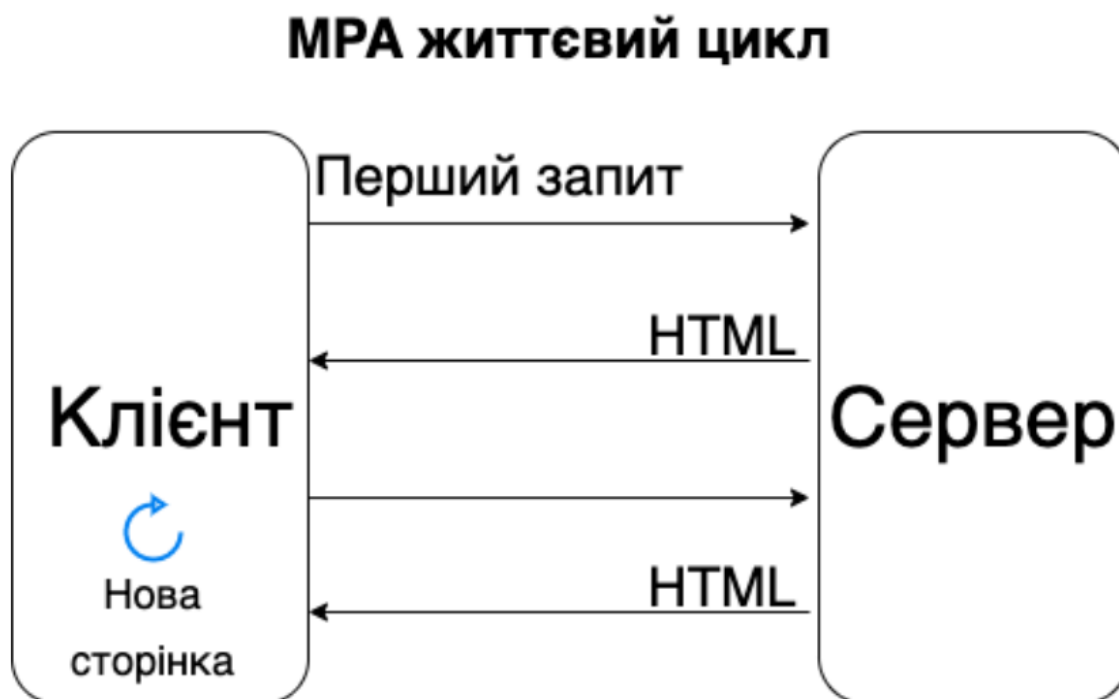


Рис. 3.1. Життєвий цикл МРА застосунку [20]

3.2.2. *Single-Page Application*

Революцією в старому підході стала поява фреймворків, які дозволили реалізовувати так звані односторінкові сайти – Single Page Application (SPA). Суть у тому, що користувач один раз завантажує CSS, JS та практично порожню HTML-сторінку, а далі вже сам JavaScript будує все у браузері користувача. При переході між сторінками JS перебудовує тільки ту частину, яка змінилась, і, якщо потрібно, робить AJAX-запит до сервера, щоб підтягнути дані.

Завдяки цьому все працює набагато швидше і виглядає сучасно. Користувач отримує плавний досвід взаємодії. Саме тому SPA активно

використовують у динамічних застосунках: поштові клієнти, панелі адміністратора, чати, маркетплейси тощо.

Звичайно, SPA також має свої недоліки. Оскільки вся логіка виконується на стороні клієнта, стартове завантаження застосунку може займати більше часу – особливо якщо проєкт великий або містить багато сторонніх бібліотек та залежностей. Користувач може чекати довше, поки все «підтягнеться», перш ніж побачить перший контент.

Ще один важливий момент – це SEO. Оскільки сторінки в SPA генеруються динамічно за допомогою JavaScript, пошукові системи не завжди можуть коректно індексувати такий контент. Хоча сучасні фреймворки частково вирішують цю проблему за допомогою SSR або SSG, це все одно додає складності в налаштування.

На рисунку 3.2 наведено діаграму життєвого циклу SPA застосунку.

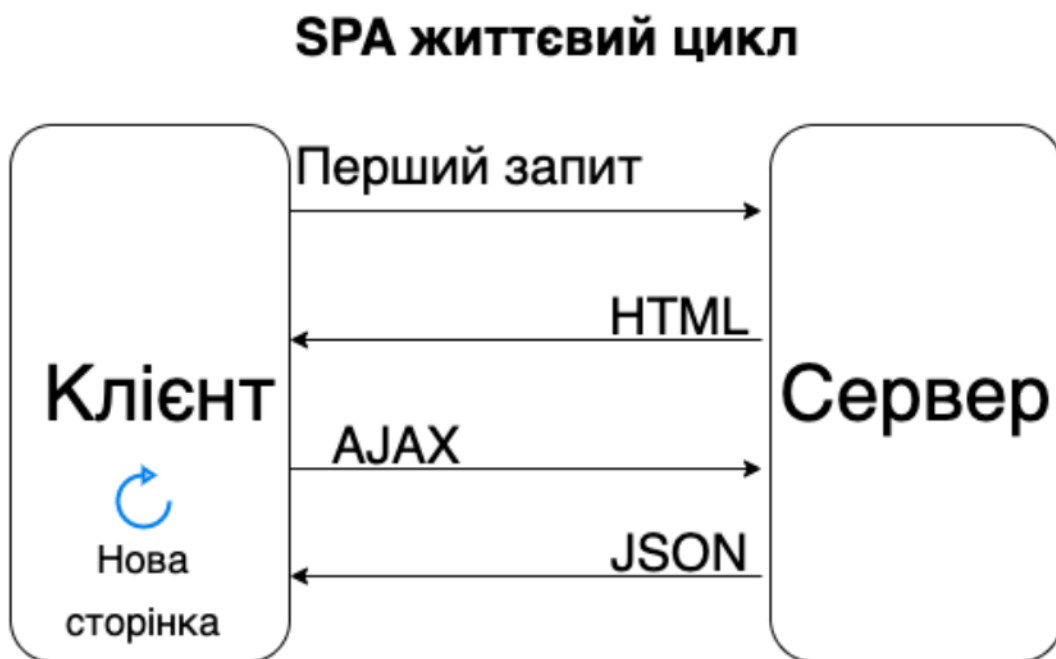


Рис. 3.2. Життєвий цикл SPA застосунку

3.2.3. SSR (Server-Side Rendering)

Головна особливість цього підходу в тому, що вся сторінка рендериться на сервері при першому запиті, а вже потім у браузері

ініціалізується Angular, і застосунок працює як звичайний SPA, виконуючи, за потреби, HTTP-запити до API. Такий підхід в Angular реалізується за допомогою Angular Universal [21].

Користувач отримує готову HTML-сторінку з актуальними даними, що завантажуються безпосередньо із сервера при кожному запиті. Це позитивно впливає на SEO – пошукові системи бачать повноцінний вміст сторінки. Користувачі теж задоволені: сторінка завантажується швидко, а далі все працює як у звичайному SPA – переходи, взаємодія, динаміка.

Так, тут є свої нюанси. Потрібен сервер, який обробляє кожен запит, виконує рендеринг і повертає результат. Але це компенсується тим, що для популярних сторінок можна налаштувати кешування, що значно пришвидшує відповіді сервера та зменшує навантаження.

На рисунку 3.3 наведено діаграму життєвого циклу SSR застосунку.

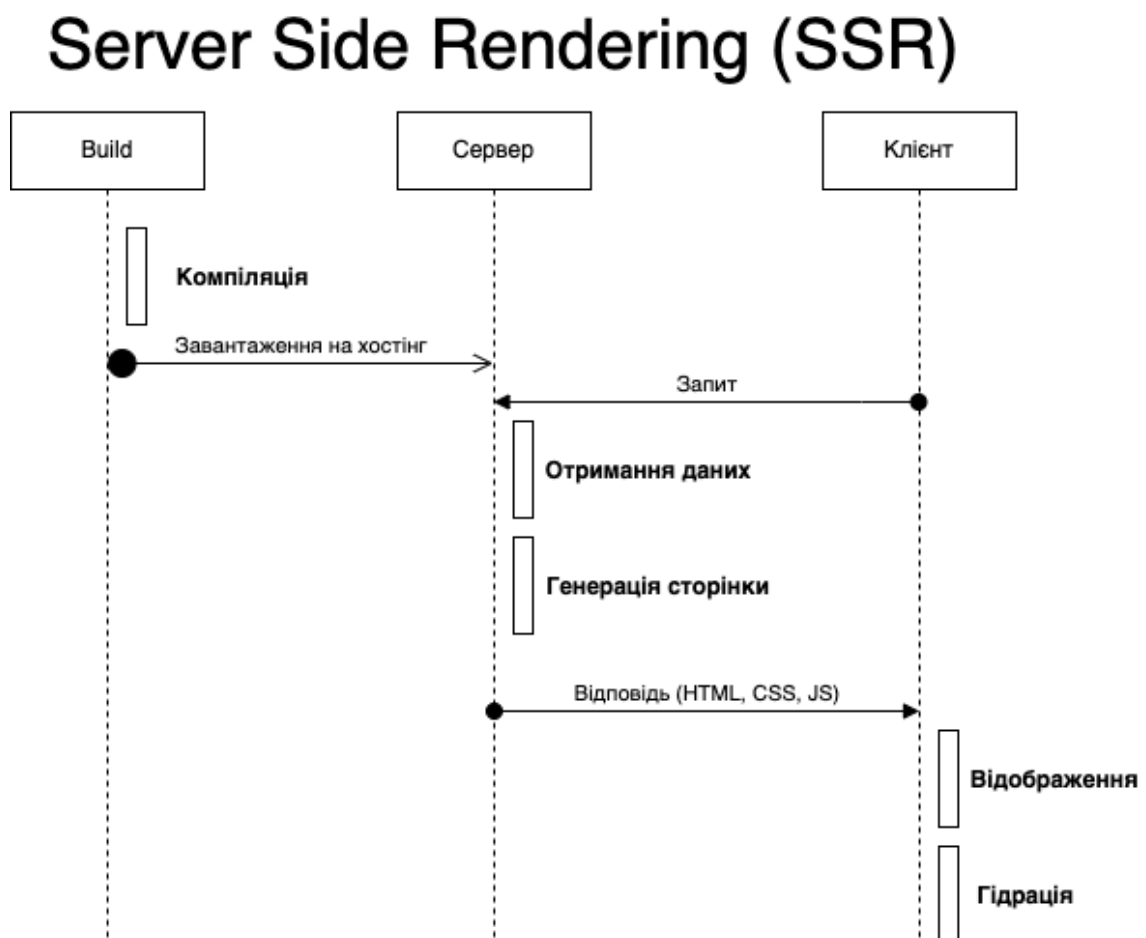


Рис. 3.3. Життєвий цикл SSR застосунку

3.2.4. Підсумок вибору архітектури

Проаналізувавши найбільш популярні рішення архітектури для вебзастосунку, було створено порівняльну таблицю (табл. 3.1).

Таблиця 3.1

Порівняння архітектур вебзастосунку

Критерій	МРА	SPA	SSR
Перше завантаження	Помірно, залежить від сторінки	Довго	Помірно
Навігація	Необхідне перезавантаження	Без перезавантаження	Можливе перезавантаження
Критерій	МРА	SPA	SSR
SEO	Так	Частково	Так
Навантаження на сервер	Середнє, залежить від сторінки	Низьке	Середнє, залежить від сторінки
Навантаження на клієнта	Низьке	Високе	Середнє
Складність розробки	Низька	Середня	Висока
Користувачський досвід	Низький	Високий	Високий

Після порівняння трьох основних архітектур: МРА, SPA і SSR – було прийнято рішення використовувати SPA для розробки дипломного проекту.

Цей варіант підходить найкраще саме для вебзастосунку з кількох причин:

- Користувачу зручно: сторінка не перезавантажується кожного разу при переходах, все відбувається швидко і «без ривків». Наприклад,

клієнт може спокійно переглядати товари, додавати їх у кошик або переходити до оформлення замовлення – все працює плавно.

- Простіше у розробці: SPA легше реалізувати у дипломному проєкті, бо не потрібно писати складну серверну логіку.
- Магазин не потребує складного SEO: хоча SPA менш зручна для пошукових систем, для невеликого магазину це не критично. Головне, щоб усе добре працювало для людей.
- Проєкт виглядає сучасно: зараз більшість популярних інтернет-магазинів працюють саме за такою схемою.

3.3. Аналіз функціональних вимог до вебзастосунку

Провівши аналіз існуючих рішень було сформовано перелік функціональних вимог, котрі необхідні для повноцінного функціонування застосунку. Перелік функціональних вимог наведено у таблиці 3.2.

Таблиця 3.2

Функціональні вимоги до вебзастосунку

Код вимоги	Назва вимоги	Опис вимоги
F-1	Реєстрація в системі	Щоб зареєструвати нового користувача в системі, необхідно ввести такі дані: • адресу електронної пошти; • логін; • повне ім'я; • пароль. У разі введення некоректних даних у відповідні поля, система має відображати відповідні повідомлення про помилки.
F-2	Авторизація в системі	Для авторизації в системі необхідно ввести логін та пароль.

Продовження табл. 3.2

Код вимоги	Назва вимоги	Опис вимоги
F-3	Створення індивідуального букета	Користувач може сформувати букет, вибираючи квіти, зелень, упаковку та додаткові елементи (листівка, стрічка тощо). Всі зміни відображаються в реальному часі у редакторі.
F-4	Перегляд готових композицій	Користувач має можливість переглядати каталог готових букетів із описами, цінами та фото.
F-5	Додавання букетів до кошика	Користувач може додавати як власноруч створені, так і готові букети до «Кошика» для подальшого перегляду.
F-6	Оформлення замовлення	Після вибору композиції користувач може оформити замовлення, вказавши адресу доставки, бажану дату й час, а також контактні дані отримувача.
F-7	Перегляд історії замовлень	У профілі користувача доступний список усіх попередніх замовлень.

3.4. Структурна організація застосунку

Застосунок «FLWRS» складається з клієнтського (браузерного) та серверного модулів, які реалізовані відповідно до клієнт-серверної архітектури. Клієнтська частина є адаптивною, що дозволяє забезпечити коректне відображення інтерфейсу на різних розмірах екранів – як на комп'ютерах, так і на мобільних пристроях.

В процесі роботи застосунку постійно здійснюється взаємодія між клієнтською та серверною частинами через REST API. Клієнтський модуль, реалізований за допомогою Angular 19, надсилає HTTP-запити до серверної частини, використовуючи вбудований Angular HttpClient. Запити містять

URL-адресу ресурсу, до якого здійснюється звернення, а також (у разі POST, PUT або DELETE операцій) – об’єкт з даними, які необхідно зберегти або змінити в базі даних.

Серверна частина застосунку реалізована на платформі ASP.NET Core. Вона містить логіку обробки запитів та взаємодіє з базою даних. Для доступу до бази даних використовується ORM-фреймворк Entity Framework Core [22], який дозволяє створювати моделі для кожної сутності (наприклад, букет, замовлення, користувач) і забезпечує типізований доступ до даних.

Обробка HTTP-запитів виконується у відповідних контролерах API, які реалізують необхідні CRUD-операції (створення, читання, оновлення, видалення), модифіковані під потреби функціональності застосунку. Після обробки запиту сервер надсилає відповідь у форматі JSON, що містить:

- запитувані або змінені дані;
- статус виконання запиту;
- додаткову службову інформацію (наприклад, повідомлення про помилки).

Після отримання відповіді, Angular-застосунок оновлює інтерфейс користувача відповідно до нових даних або інформує про результат виконання операції. Завдяки реактивному підходу, зміни миттєво відображаються в інтерфейсі без необхідності перезавантаження сторінки.

Нижче на рисунку 3.4 представлена структурна схема розроблюваного вебзастосунку.

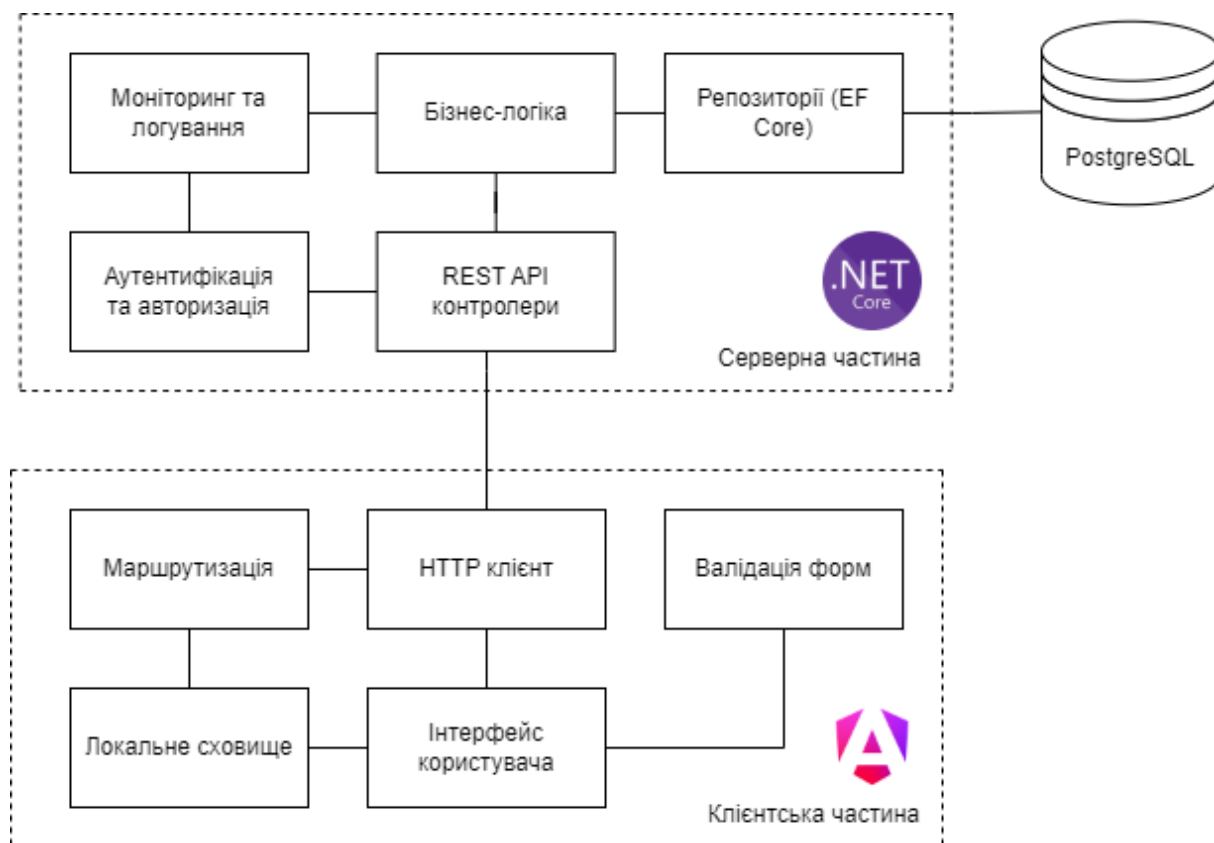


Рис. 3.4. Структурна схема вебзастосунку

3.5. Особливості реалізації серверної частини застосунку

На сьогоднішній день існує кілька основних підходів до побудови архітектури застосунків, серед яких найпоширенішими є монолітна та мікросервісна архітектури. Монолітна архітектура передбачає побудову застосунку як єдиного цільного блоку, в той час як мікросервісна – як набір незалежних сервісів, кожен з яких відповідає за певну частину функціональності. Компромісом між цими двома підходами є модульний моноліт, де застосунок залишається єдиним цілим з технічної точки зору, однак логічно розділяється на незалежні частини (модулі або шари), кожна з яких має чітко визначену відповідальність.

Для серверної частини вебзастосунку «FLWRS» було обрано архітектурний підхід модульного моноліту з чітким розділенням логіки. Це дозволило досягти таких переваг:

- зручність супроводу та розширення функціональності;
- покращена продуктивність;

- швидше розгортання;
- простіше тестування;
- легша адаптація до майбутнього переходу на мікросервісну модель, якщо така потреба з'явиться.

3.5.1. Особливості реалізації модулів серверної частини застосунку

Серверна частина поділена на три основні модулі:

- `flwts-api`: Він містить контролери API, які обробляють вхідні HTTP-запити від клієнтської частини (написаної на Angular) та спрямовують їх до відповідних сервісів бізнес-логіки. Саме тут реалізована маршрутизація запитів, перевірка авторизації, а також формування HTTP-відповідей.
- `flwts-domain`: це ядро застосунку, яке містить основну бізнес-логіку, доменні моделі (наприклад, `Bouquet`, `Order`, `User`), інтерфейси сервісів та репозиторіїв. Цей шар є повністю ізольованим і не залежить від зовнішніх бібліотек чи інфраструктури, що робить його універсальним і легко тестованим.
- `flwts-infrastructure`: цей модуль реалізує взаємодію з зовнішніми системами, такими як база даних, сервіси відправки електронної пошти або повідомлень, файлове сховище тощо. Саме тут розташовані реалізації інтерфейсів репозиторіїв, доступ до бази даних через Entity Framework Core, а також файли міграцій.

Такий підхід до побудови архітектури дозволяє гнучко розширювати функціональність проєкту без порушення вже наявної логіки. За потреби легко можна додати нові шари або функціональні блоки, не впливаючи на загальну стабільність системи.

3.5.2. Особливості реалізації автентифікації та авторизації

У вебзастосунку для онлайн-створення та купівлі букетів реалізовано сучасну систему автентифікації та авторизації, що ґрунтується на механізмах JWT у поєднанні з ASP.NET Core Identity [23].

Для автентифікації користувача використано готові кінцеві точки API, надані бібліотекою ASP.NET Core Identity, які забезпечують повний цикл роботи з обліковими записами: реєстрацію, вхід у систему, вихід, генерацію токенів та оновлення access-токенів за допомогою refresh-токенів. Після успішної авторизації клієнтська частина (Angular) отримує два токени:

- access-токен – короткоживучий (наприклад, 1 хвилина), який вбудовується у заголовок кожного запиту (Authorization: Bearer <token>);
- refresh-токен – зберігається у cookies і використовується для автоматичного оновлення access-токена без повторного входу користувача.

Це дозволяє забезпечити як безпеку, так і зручність для користувача.

Авторизація реалізована шляхом обробки claims, що містяться у JWT-токені. Таким чином, система може ідентифікувати користувача та визначити рівень його доступу до окремих ресурсів. Наприклад, кожне замовлення містить зовнішній ключ (UserId) на користувача, що його створив, а всі операції із замовленнями виконуються лише від імені автентифікованого користувача.

Завдяки використанню стандартних інструментів платформи ASP.NET Core, реалізована система автентифікації є гнучкою, безпечною та розширюваною, з можливістю підключення ролей, кастомних claims, або інтеграції з зовнішніми провайдерами у майбутньому.

Приклад підключення схеми автентифікації наведено в лістингу 3.1.

Лістинг 3.1. Підключення схеми автентифікації

```
builder.Services.AddDbContext<DemoDbContext>(opt =>
{
```

```

opt.UseNpgsql(builder.Configuration.GetConnectionString(
    "DbConnectionString"));
});

builder.Services.AddIdentityApiEndpoints<IdentityUser>()
    .AddEntityFrameworkStores<DemoDbContext>();

builder.Services.ConfigureAll<BearerTokenOptions>(opt =>
{
    opt.BearerTokenExpiration = TimeSpan.FromMinutes(1);
});

builder.Services.AddAuthorization();

```

3.6. Особливості реалізації клієнтської частини застосунку

Клієнтську частину вебзастосунку реалізовано у вигляді односторінкового застосунку (SPA), побудованого за допомогою фреймворку Angular. Односторінкові вебзастосунки завантажують основну HTML-сторінку один раз, а подальші зміни в інтерфейсі відбуваються динамічно – без повного перезавантаження сторінки. Це забезпечує швидку та зручну взаємодію користувача із застосунком.

Використання Angular Router для реалізації маршрутизації та модульного підходу до створення компонентів інтерфейсу дозволило побудувати зручну структуру застосунку, зображену на рис. 3.5.

Основне меню відображається постійно після авторизації користувача, і забезпечує швидкий перехід між основними розділами.

У застосунку реалізовано такі основні маршрути:

- /home – маршрут для відображення головної сторінки застосунку;
- /login – маршрут для відображення сторінки авторизації;
- /registration – маршрут для відображення сторінки реєстрації;
- /constructor – маршрут для відображення сторінки конструктора букета, яка дозволяє створювати власний букет із наявних квітів;
- /catalog – маршрут для відображення сторінки каталогу готових букетів для перегляду та замовлення;

- /profile – маршрут для відображення сторінки користувача;
- /cart – маршрут для відображення сторінки кошику;
- /checkout – маршрут для відображення сторінки оформлення замовлення;
- /product/{id} – маршрут для відображення сторінки готового букета.

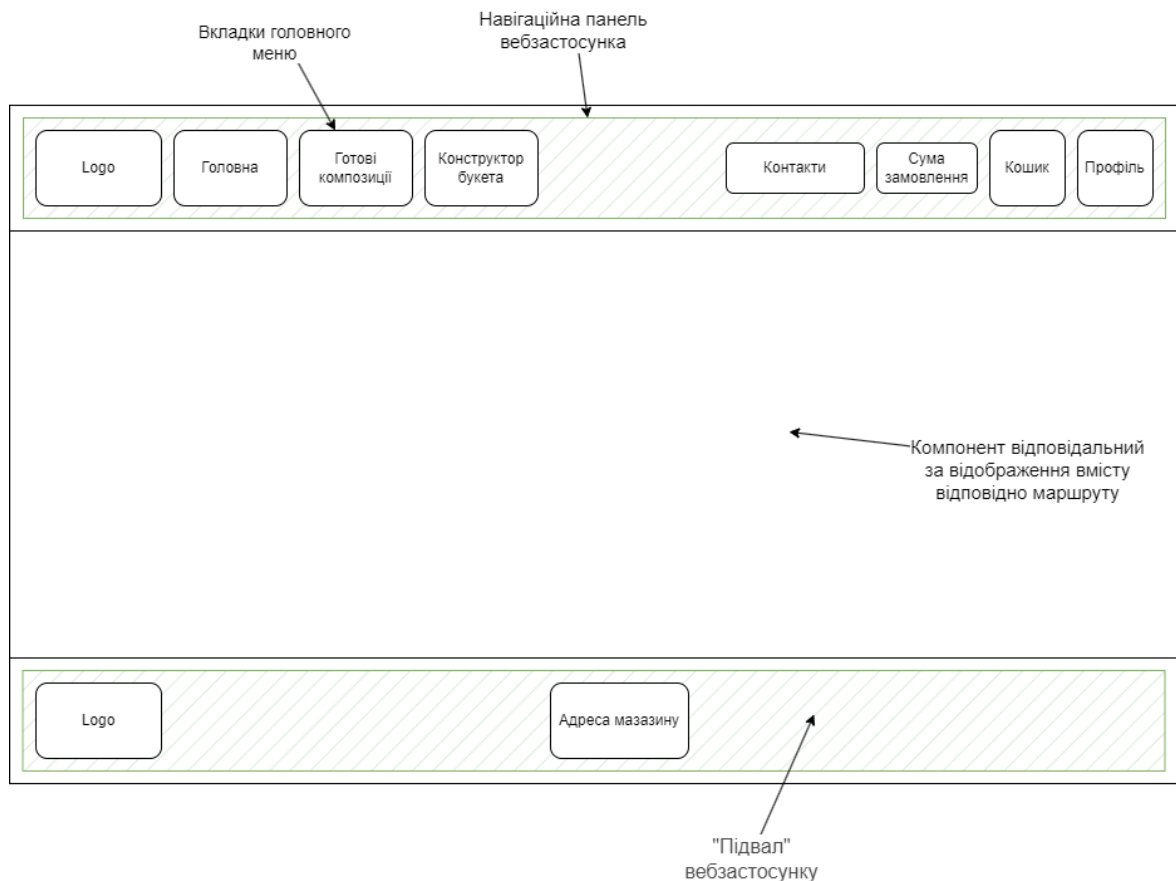


Рис. 3.5. Структура вебсайту

3.7. Структура бази даних

Для реалізації дипломного вебзастосунку була спроектована схема бази даних. Під час проєктування дані були структуровані з урахуванням принципів нормалізації. У результаті було визначено ключові компоненти схеми бази даних – таблиці, їх атрибути та зв'язки між ними.

Нижче наведено схему бази даних у вигляді ER-діаграми (рис. 3.6).

2) Orders (Замовлення) – зберігає інформацію про оформлені користувачами замовлення:

- Id – унікальний ідентифікатор замовлення.
- UserId – зовнішній ключ, що вказує на користувача, який зробив замовлення.
- CreatedAt – дата та час оформлення замовлення.
- Location – адреса доставки.
- Description – опис або коментар користувача до замовлення.
- Price – загальна сума замовлення з урахуванням букетів, обгортки та додаткових елементів.

3) Bouquets (Букети) – таблиця зберігає інформацію про готові або створені користувачами букети:

- Id – унікальний ідентифікатор букета.
- Name – назва букета.
- IsPublic – чи відображається букет у публічному каталозі (наприклад, створені адміністратором).
- CreatedByUserId – користувач, який створив букет.
- WrapperId – зовнішній ключ до обгортки.
- Description – текстовий опис складу букета.
- ImageUrl – посилання на зображення букета.

4) Flowers (Квіти) – зберігає всі доступні типи квітів, які можна використовувати при створенні букетів:

- Id – унікальний ідентифікатор квітки.
- Name – назва квітки (наприклад, «Роза червона 60 см»).
- Price – вартість однієї квітки.
- CategoryId – зовнішній ключ до категорії квітів.

5) FlowerCategories (Категорії квітів) – класифікація квітів для зручної фільтрації та відображення:

- Id – унікальний ідентифікатор категорії.

- Name – назва категорії (наприклад, «Рози», «Тюльпани»).
 - ImageUrl – зображення категорії для відображення в інтерфейсі.
- 6) Wrappers (Обгортки для букетів) – описує доступні варіанти обгортки для букетів:
- Id – унікальний ідентифікатор обгортки.
 - Material – тип матеріалу (наприклад, «крафт-папір», «атласна стрічка»).
 - Price – вартість обгортки.
 - ImageUrl – зображення.
- 7) ExtraItems (Додаткові елементи до букетів) – додаткові декоративні елементи (м'які іграшки, коробки, листівки тощо), які можна додати до букету:
- Id – унікальний ідентифікатор.
 - Name – назва.
 - Price – вартість.
 - ImageUrl – зображення.
- 8) BouquetInOrder (Зв'язок між букетом і замовленням) – проміжна таблиця, яка фіксує, які саме букети і в якій кількості входять до певного замовлення:
- OrderId – зовнішній ключ до таблиці замовлень.
 - BouquetId – зовнішній ключ до таблиці букетів.
 - Quantity – кількість одиниць цього букета в замовленні.
- 9) FlowerInBouquet (Зв'язок між квіткою та букетом) – містить інформацію про те, які квіти і в якій кількості входять до конкретного букета:
- FlowerId – зовнішній ключ до квітки.
 - BouquetId – зовнішній ключ до букета.
 - Quantity – кількість квітів цього типу в букеті.

- 10) **BouquetExtraItem** (Зв'язок між букетом і додатковими елементами) – містить інформацію про додаткові елементи, які входять до певного букета:
- **BouquetId** – зовнішній ключ до букета.
 - **ExtraItemId** – зовнішній ключ до додаткового елемента.
- 11) **AspNetRoles** (Ролі користувачів) – таблиця, яка містить інформацію про ролі, що використовуються в системі для керування доступом:
- **Id** – унікальний ідентифікатор ролі (первинний ключ).
 - **Name** – назва ролі (наприклад, «Admin», «User»).
 - **NormalizedName** – нормалізована назва ролі для ефективного пошуку.
 - **ConcurrencyStamp** – мітка для оптимістичної конкуренції (використовується для контролю змін).
- 12) **AspNetUserRoles** (Зв'язок між користувачами та ролями) – проміжна таблиця, яка фіксує, які ролі має кожен користувач:
- **UserId** – зовнішній ключ до таблиці користувачів (**AspNetUsers**).
 - **RoleId** – зовнішній ключ до таблиці ролей (**AspNetRoles**).

3.8. Висновки до розділу

Вебзастосунок реалізовано на основі клієнт-серверної архітектури. Розробку застосунку розділено на дві основні частини: клієнтську (вебінтерфейс) та серверну (API). Клієнтська частина відповідає за взаємодію з користувачем, відображення інформації та надсилання запитів до сервера. Серверна частина забезпечує обробку цих запитів, реалізацію бізнес-логіки та взаємодію з базою даних.

Серверна частина вебзастосунку реалізована у вигляді модульного моноліту з дотриманням принципів чистої архітектури. Проєкт поділено на три основні шари. Завдяки такій структурі кожен шар чітко відповідає за свою область відповідальності, що підвищує підтримуваність, гнучкість та можливість масштабування застосунку в майбутньому.

Було визначено основні сутності, їх атрибути та зв'язки між ними, що дозволило побудувати повноцінну реляційну базу даних, необхідну для забезпечення повного циклу роботи застосунку.

Клієнтська частина розроблена як односторінковий застосунок (SPA) з використанням Angular. Навігація між сторінками відбувається без перезавантаження браузера за допомогою Angular Router. Основні компоненти інтерфейсу відображаються в залежності від маршруту, який визначається діями користувача: переходами між вкладками, натисканням кнопок тощо.

У результаті реалізовано сучасний, інтерактивний вебзастосунок, який поєднує гнучкість, високу продуктивність та зручність у використанні.

4. АНАЛІЗ РОЗРОБЛЕНОГО ВЕБЗАСТОСУНКУ

4.1. Аналіз реалізованого вебзастосунок

У результаті розробки було створено вебзастосунок для онлайн створення та замовлення букетів. Сервіс надає користувачам зручні інструменти для вибору квітів, персоналізації композицій та швидкого оформлення замовлень. Основною функціональною можливістю застосунку є створення унікальних букетів з доступних квітів і аксесуарів, а також перегляд каталогу готових композицій.

Для доступу до усіх функціональних можливостей користувач повинен пройти процедуру реєстрації або авторизації. Після входу в систему користувач отримує можливість створювати власні букети, додавати їх до кошика, оформлювати замовлення, переглядати історію покупок, редагувати особистий профіль тощо. Для захисту персональних даних і безпеки комунікації між клієнтською та серверною частинами використовується захищений протокол HTTPS.

Ключова взаємодія із застосунком здійснюється через головне меню, яке містить навігаційні вкладки до основних сторінок. Зокрема, користувач може переглядати каталог готових букетів, застосовувати фільтри для пошуку (за типом квітів, ціною, популярністю), або перейти до конструктора, де можна сформувати індивідуальний букет у реальному часі.

Оформлення замовлення виконується інтуїтивно: користувач обирає адресу доставки, зручний час та спосіб оплати. Інтерфейс побудовано з урахуванням принципів адаптивного дизайну, що забезпечує коректне відображення сторінок на різних пристроях (комп'ютери та ноутбуки, планшети, мобільні телефони).

Перевагами створеного вебзастосунку є простота використання, наочний механізм створення букетів, можливість додавання різноманітних аксесуарів, а також швидке оформлення замовлень.

Клієнтська частина застосунку реалізована як односторінковий застосунок (SPA) з використанням Angular. Усі дії користувача, включно з введенням даних, фільтрацією, навігацією та формуванням букету, обробляються в браузері, після чого формуються HTTP-запити до серверної частини. Серверна частина, реалізована з використанням ASP.NET Core, відповідає за обробку запитів, виконання бізнес-логіки, доступ до бази даних та повернення структурованих відповідей у форматі JSON.

4.2. Тестування реалізованого застосунку

Одним з ключових етапів розроблення програмного забезпечення є тестування, оскільки він забезпечує його стабільність, відповідність вимогам та загальну якість. Завдяки тестуванню можна:

- вчасно виявити помилки та дефекти в роботі застосунку;
- покращити рівень надійності системи;
- підвищити довіру та задоволеність користувачів;
- отримати перевагу над конкурентами.

У процесі реалізації цього програмного застосунку тестування проводилось після розроблення кожного окремого логічного етапу. Для перевірки коректності роботи використовувався ручний підхід до тестування. Відповідно до функціональних вимог було створено перелік тестових випадків, описаних в табл. 4.1, з метою виявлення потенційних дефектів.

Таблиця 4.1

Перелік тестових випадків

Номер тестового випадку	Умови тестового випадку	Очікуваний результат
T-1	Користувач вводить правильні логін та пароль.	Користувач переходить на головну сторінку застосунку.

Продовження табл. 4.1

Номер тестового випадку	Умови тестового випадку	Очікуваний результат
T-2	Користувач відкриває сторінку реєстрації. Вводить коректні email, пароль та повторний пароль. Натискає кнопку реєстрації.	Користувач успішно реєструється та потрапляє на головну сторінку або сторінку входу.
T-3	Користувач вводить email у неправильному форматі.	Система відображає повідомлення про помилку в полі електронної пошти.
T-4	Пароль надто короткий (менше 6 символів).	Система показує повідомлення про недопустиму довжину пароля.
T-5	Користувач вводить неправильний пароль.	Система повідомляє про невірний логін або пароль.
T-6	Користувач залишає поля порожніми та натискає «Увійти».	Система показує повідомлення про обов'язковість заповнення полів.
T-7	Користувач вводить недопустимі символи в логін.	Система повідомляє про помилку у форматі логіну.
T-8	Користувач вибирає квіти, упаковку, листівку.	У редакторі відображається актуальний букет з усіма обраними елементами.
T-9	Користувач відкриває розділ «Каталог букетів».	Виводиться список композицій із фото, описом та ціною.

Продовження табл. 4.1

Номер тестового випадку	Умови тестового випадку	Очікуваний результат
T-10	Користувач натискає кнопку «Додати до кошика» на створеному букеті.	Букет з'являється в кошику.
T-11	Користувач додає готову композицію в кошик.	Обрана композиція додається до кошика.
T-12	Користувач заповнює всі поля замовлення правильно.	Замовлення приймається, з'являється повідомлення про успіх.
T-13	Користувач не вказав адресу доставки.	Виводиться повідомлення про обов'язковість цього поля.
T-14	Користувач переходить до оплати, вводять коректні реквізити, підтверджує платіж.	Платіж проходить успішно, з'являється підтвердження.
T-15	Користувач відкриває розділ «Історія замовлень».	Виводиться список усіх попередніх замовлень із деталями.

4.3. Рекомендації щодо подальшого вдосконалення

Попри те, що вебзастосунок виконує основні функції – створення індивідуальних букетів, перегляд готових композицій та оформлення замовлень – існує низка напрямів, у яких його можна вдосконалити для подальшого розвитку й утримання конкурентної позиції на ринку.

Рекомендації щодо покращення функціональних можливостей та розширення можливостей застосунок:

- Впровадження двофакторної автентифікації (2FA). Це забезпечить додатковий рівень захисту облікових записів користувачів.

- Багатомовність інтерфейсу. Це сприятиме підвищенню лояльності з боку клієнтів. Також додавання підтримки кількох мов дозволить залучити користувачів із різних країн і розширити аудиторію.
- Рекомендаційна система. На основі історії покупок та переглянутих композицій можна реалізувати систему персональних рекомендацій, яка пропонуватиме букети або оформлення, що можуть сподобатися користувачу.
- Інтеграція системи аналітики. Збір статистичних даних про дії користувачів дозволить виявляти популярні квіти, упаковки, комбінації та вдосконалювати асортимент на основі реальних уподобань клієнтів.
- Створення онлайн чату з підтримкою. Якщо в застосунку є можливість онлайн спілкування з консультантом, це може сильно покращити користувацький досвід та збільшити гнучкість системи.
- Співпраця з партнерами. Укладання договорів з флористичними магазинами, службами доставки або агентствами з організації та проведення заходів відкриє нові канали для просування та розширить перелік доступних послуг.

4.4. Висновки до розділу

У цьому розділі було представлено розроблений вебзастосунок для створення та замовлення букетів в онлайн-режимі. Детально описана основна функціональність системи, яка дозволяє користувачам формувати індивідуальні композиції, переглядати каталог готових букетів, оформлювати замовлення та здійснювати оплату через зручний інтерфейс. Наведено опис загальної логіки роботи застосунку, а також механізмів взаємодії користувача з системою.

Після впровадження кожної нової функціональної можливості проводилось ручне тестування для перевірки коректності її роботи. Такий підхід дозволив виявити та усунути потенційні помилки ще на етапі

розробки, що позитивно вплинуло на стабільність і якість кінцевого продукту.

Незважаючи на те, що застосунок має ряд переваг у порівнянні з аналогічними рішеннями, описаними в першому розділі, його подальший розвиток вимагає регулярного вдосконалення та впровадження сучасних технологій. У цьому контексті були запропоновані рекомендації щодо покращення системи, зокрема – розширення мовної підтримки, реалізація системи персоналізованих рекомендацій, аналітики, сповіщень і покращеної системи безпеки, включаючи двофакторну автентифікацію.

З урахуванням запропонованих покращень, застосунок має потенціал не лише для збільшення кількості користувачів і покращення взаємодії з ними, але й для трансформації у платформу-посередника між флористичними магазинами різних міст та замовниками, створюючи єдину мережу для зручного вибору, замовлення та доставки квіткової продукції.

ВИСНОВКИ

Метою даного дипломного проєкту було створення сучасного вебзастосунку для онлайн-конструювання та придбання букетів, який поєднує в собі зручність для користувача та гнучкість з боку адміністраторів або флористів. У процесі роботи було проаналізовано існуючі рішення на ринку онлайн-флористики, що дозволило визначити ключові функціональні потреби та конкурентні переваги.

Застосунок реалізовано з використанням сучасного технологічного стеку: ASP.NET Core для створення серверної частини, Entity Framework Core для роботи з реляційною базою даних, Angular 19 для реалізації інтуїтивно зрозумілого та реактивного клієнтського інтерфейсу. Також впроваджено ASP.NET Identity для автентифікації та авторизації користувачів, з урахуванням безпечної роботи з паролями та токенами доступу.

Користувачі мають змогу переглядати та обирати готові букети, створювати власні композиції за допомогою конструктора букетів, додавати обгортки та додаткові елементи (кульки, цукерки тощо), оформлювати замовлення з вказанням контактних даних, переглядати історію своїх замовлень через профіль.

Флористи можуть створювати публічні букети, а також керувати складом продукції. Всі зміни зберігаються у базі даних, а взаємодія між клієнтською та серверною частинами реалізована через REST API.

Усі функціональні можливості були протестовані вручну та інтеграційно. Результати тестування підтвердили правильність логіки та відсутність критичних помилок. Під час розробки особливу увагу було приділено розширюваності архітектури та простоті подальшого вдосконалення.

У майбутньому планується розширити функціональні можливості застосунку, особливу увагу буде приділено покращенню SEO та реалізації чату з флористом у реальному часі.

Таким чином, розроблений вебзастосунок є гнучким, безпечним і придатним до розширення, що робить його ефективним інструментом для сучасного онлайн-магазину з продажу квіткових композицій.

СПИСОК ВИКОРИСТАНИХ ЛІТЕРАТУРИНИХ ДЖЕРЕЛ

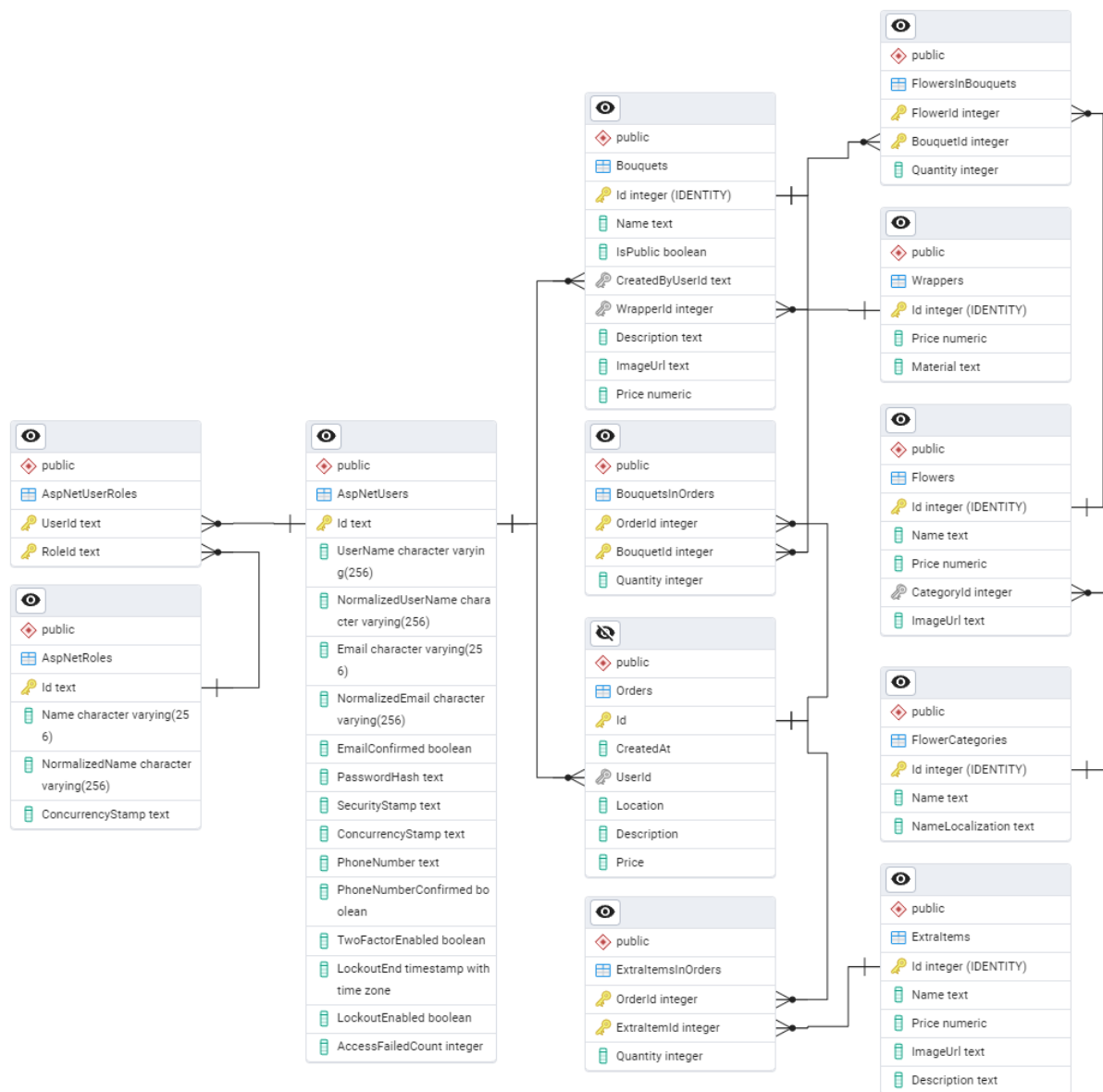
1. Interflora Україна. Доставка квітів: Кур'єр для доставки квітів [Електронний ресурс]. – Режим доступу: <https://www.interflora.ua/uk> – Дата доступу: 20.04.2025 р.
2. Interflora: The flower experts [Електронний ресурс]. – Режим доступу: <https://play.google.com/store/apps/details?id=com.interflora.android&hl=uk> – Дата доступу: 20.04.2025 р.
3. Flowers.ua. Доставка квітів по Україні та світу [Електронний ресурс]. – Режим доступу: <https://flowers.ua/> – Дата доступу: 20.04.2025 р.
4. Доставка Квітів Україна | Київ (@flowers.ua_official) [Електронний ресурс]. – Режим доступу: https://www.instagram.com/flowers.ua_official/ – Дата доступу: 20.04.2025 р.
5. SunRose: Доставка квітів в Києві, замовити квіти с доставкой [Електронний ресурс]. – Режим доступу: <https://sunrose.com.ua/> – Дата доступу: 20.04.2025 р.
6. W3 Techs - Statistics of server-side programming languages [Електронний ресурс]. – Режим доступу: https://w3techs.com/technologies/overview/programming_language – Дата доступу: 20.04.2025 р.
7. Wikipedia - Convention Over Configuration [Електронний ресурс]. – Режим доступу: https://en.wikipedia.org/wiki/Convention_over_configuration – Дата доступу: 20.04.2025 р.
8. Microsoft – ASP.NET [Електронний ресурс]. – Режим доступу: <https://dotnet.microsoft.com/en-us/> – Дата доступу: 20.04.2025 р.
9. Що краще моноліт чи мікросервіси? [Електронний ресурс]. – Режим доступу: <https://iampm.club/ua/blog/shho-krashhe-monolit-chi-mikroservisi-yak-obrati-arhitekturu-projektu/> – Дата доступу: 20.04.2025 р.

10. Owasp - Cross Site Request Forgery (CSRF) [Электронный ресурс]. – Режим доступа: <https://owasp.org/www-community/attacks/csrf> – Дата доступа: 20.04.2025 г.
11. Owasp - Cross Site Scripting (XSS) [Электронный ресурс]. – Режим доступа: <https://owasp.org/www-community/attacks/xss/> – Дата доступа: 20.04.2025 г.
12. Вступ до API [Электронный ресурс]. – Режим доступа: <https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi> – Дата доступа: 20.04.2025 г.
13. W3Techs - Statistics of client-side programming languages [Электронный ресурс]. – Режим доступа: https://w3techs.com/technologies/overview/client_side_language – Дата доступа: 20.04.2025 г.
14. ReactJS: A brief history [Электронный ресурс]. – Режим доступа: <https://medium.com/@sjarancio/reactjs-a-brief-history-3c1e969a477f> – Дата доступа: 20.04.2025 г.
15. Routing in single-page applications [Электронный ресурс]. – Режим доступа: <https://angular.dev/guide/routing/router-tutorial> – Дата доступа: 20.04.2025 г.
16. The History of Angular [Электронный ресурс]. – Режим доступа: <https://medium.com/the-startup-lab-blog/the-history-of-angular-3e36f7e828c7> – Дата доступа: 20.04.2025 г.
17. Comparing MySQL, PostgreSQL, and MongoDB [Электронный ресурс]. – Режим доступа: <https://www.geeksforgeeks.org/comparing-mysql-postgresql-and-mongodb/> – Дата доступа: 20.04.2025 г.
18. Popularity Ranking Of Database Management Systems [Электронный ресурс]. – Режим доступа: <https://db-engines.com/en/ranking> – Дата доступа: 20.04.2025 г.

19. What is a Multi-Page Application (MPA) Exactly? [Електронний ресурс]. – Режим доступу: <https://medium.com/@julianneagu/multi-page-application-mpa-a-good-business-fit-36029c7be9f0> – Дата доступу: 20.04.2025 р.
20. Способи генерації сторінок: CSR, SSR, SSG, ISR [Електронний ресурс]. – Режим доступу: <https://dou.ua/forums/topic/41585/> – Дата доступу: 20.04.2025 р.
21. Angular – Server-Side Rendering [Електронний ресурс]. – Режим доступу: <https://v17.angular.io/guide/ssr> – Дата доступу: 20.04.2025 р.
22. Overview of Entity Framework Core - EF Core [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/ef/core/> – Дата доступу: 20.04.2025 р.
23. Introduction to Identity on ASP.NET Core [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-8.0&tabs=visual-studio> – Дата доступу: 20.04.2025 р.
24. Create an ERD in pgAdmin 4 [Електронний ресурс]. – Режим доступу: <https://www.enterprisedb.com/blog/create-erd-pgadmin-4> – Дата доступу: 20.04.2025 р.

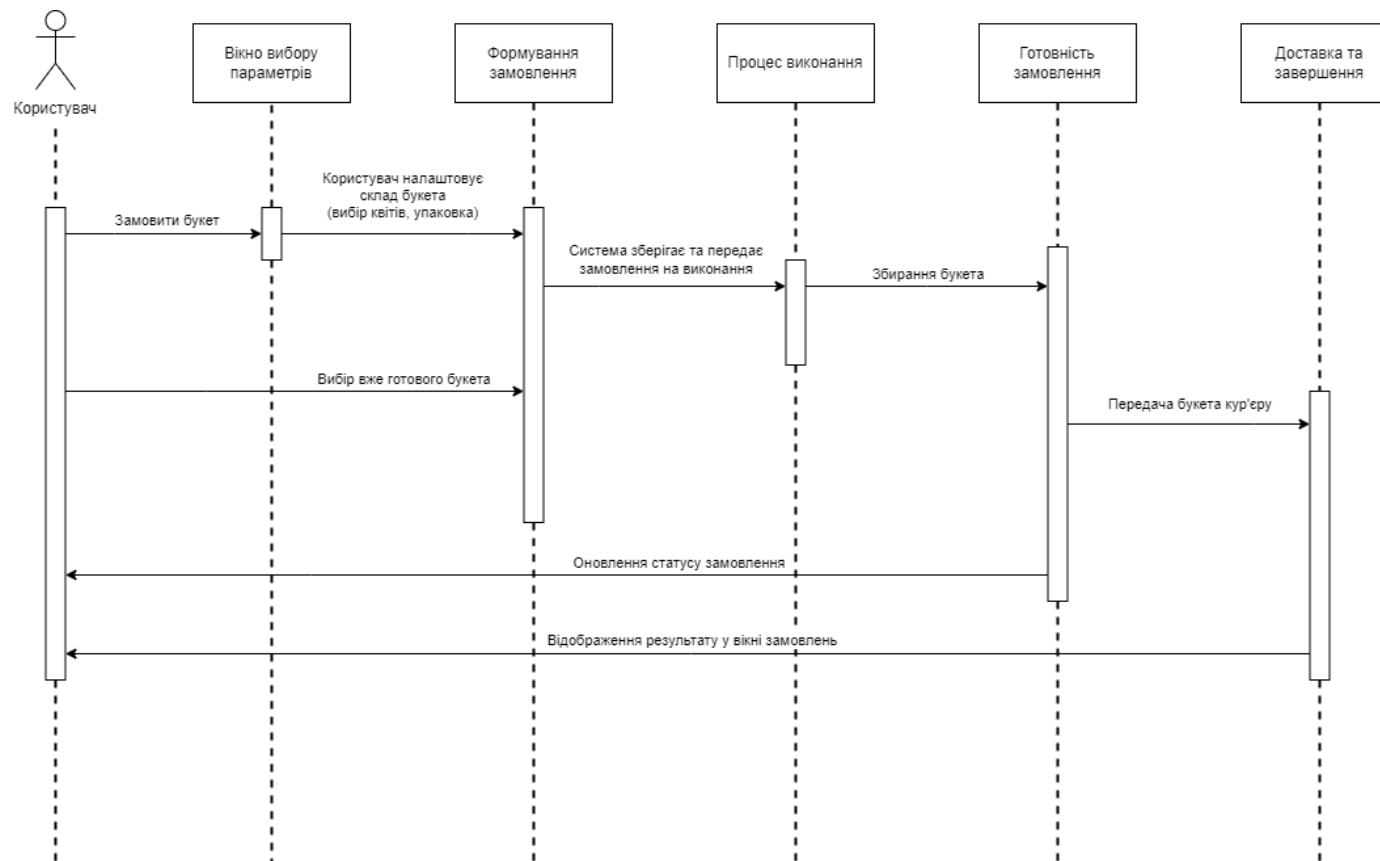
ДОДАТКИ

Додаток 1
Копії графічних матеріалів



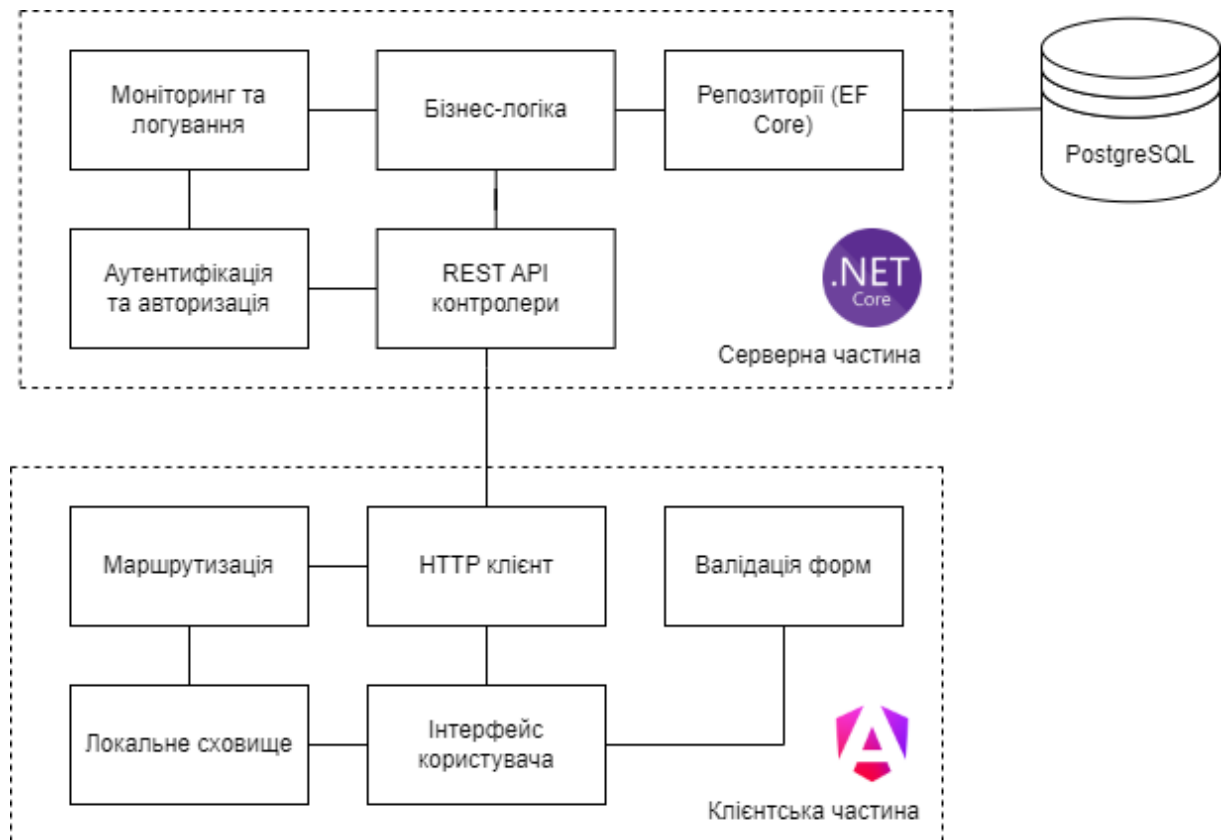
ДП.045440-06-99

Вебзастосунок для онлайн створення та купівлі букетів з квітів. Схема бази даних системи. ER-діаграма



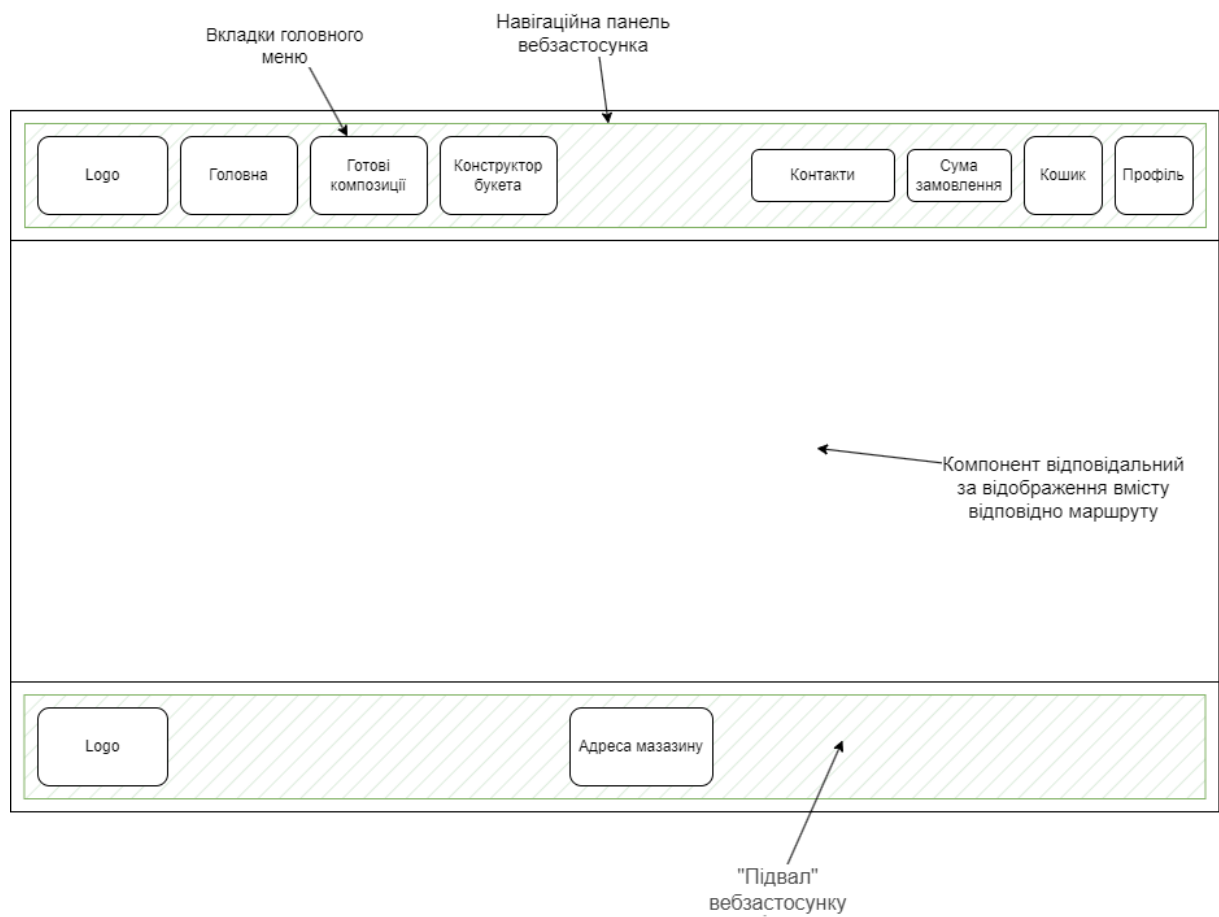
ДП.045440-07-99

Вебзастосунок для онлайн створення та купівлі букетів з квітів. Діаграма послідовності для замовлення букета. Діаграма послідовності



Структурна схема вебзастосунку

Онасенко О. В., група КП-12



Структура вебсайту

Онасенко О. В., група КП-12

Додаток 2
Лістинг програм

Фрагмент коду налаштування серверу застосунка

```
using flwrs_infrastructure;
using Microsoft.AspNetCore.Authentication.BearerToken;
using Microsoft.AspNetCore.Identity;
using Microsoft.EntityFrameworkCore;
using Microsoft.OpenApi.Models;

var builder = WebApplication.CreateBuilder(args);

builder.Services.AddEndpointsApiExplorer();

builder.Services.AddControllers();

builder.Services.AddDbContext<DemoDbContext>(opt =>
{
    opt.UseNpgsql(builder.Configuration.GetConnectionString("DbConnectionString"));
});
builder.Services.AddIdentityApiEndpoints<IdentityUser>()
    .AddRoles<IdentityRole>()
    .AddEntityFrameworkStores<DemoDbContext>();

builder.Services.ConfigureAll<BearerTokenOptions>(opt =>
{
    opt.BearerTokenExpiration = TimeSpan.FromMinutes(60);
});
builder.Services.AddAuthorization();

var app = builder.Build();

if (app.Environment.IsDevelopment())
{
    app.UseSwagger();
    app.UseSwaggerUI();
}

app.UseStaticFiles();
app.UseHttpsRedirection();
app.UseCors("AllowAngularApp");
app.MapIdentityApi<IdentityUser>();
app.UseAuthentication();
app.UseAuthorization();
app.MapControllers();

using (var scope = app.Services.CreateScope())
{
    var roleManager = scope.ServiceProvider.GetRequiredService<RoleManager<IdentityRole>>();
    var roles = new[] { "Admin", "User" };

    foreach (var role in roles)
    {
        if (!await roleManager.RoleExistsAsync(role))
            await roleManager.CreateAsync(new IdentityRole(role));
    }
}
```

```
}
```

```
app.Run();
```

Фрагмент коду створення маршрутизації клієнтської частини

```
import {Routes} from '@angular/router';
import {HomeComponent} from './pages/home/home.component';
import {LoginComponent} from './pages/login/login.component';
import {ProductPageComponent} from './pages/product-page/product-page.component';
import {CartComponent} from './pages/cart/cart.component';
import {PageNotFoundComponent} from './pages/page-not-found/page-not-found.component';
import {RegistrationComponent} from './pages/registration/registration.component';
import {ProfileComponent} from './pages/profile/profile.component';
import {ConstructorComponent} from './pages/constructor/constructor.component';
import {CatalogComponent} from './pages/catalog/catalog.component';
import {CheckoutComponent} from './pages/checkout/checkout.component';

export const routes: Routes = [
  {
    path: '', redirectTo: 'home', pathMatch: 'full'
  },
  {
    path: 'home', component: HomeComponent, title: 'Доставка квітів FLWRS'
  },
  {
    path: 'cart', component: CartComponent, title: 'Кошик FLWRS'
  },
  {
    path: 'login', component: LoginComponent, title: 'Увійдіть до свого облікового запису'
  },
  {
    path: 'registration', component: RegistrationComponent, title: 'Створіть обліковий запис'
  },
  {
    path: 'profile', component: ProfileComponent, title: 'Профіль користувача'
  },
  {
    path: 'product/:id', component: ProductPageComponent, title: 'Доставка квітів FLWRS'
  },
  {
    path: 'constructor', component: ConstructorComponent, title: 'Конструктор букетів FLWRS'
  },
  {
    path: 'catalog', component: CatalogComponent, title: 'Каталог букетів FLWRS'
  },
  {
    path: 'checkout', component: CheckoutComponent, title: 'Оформлення замовлення'
  },
  {
    path: '**', component: PageNotFoundComponent, title: 'Сторінка не знайдена'
  }
];
```

Фрагмент коду налаштування об'єктно-реляційного відображення бази даних

```
using flwrs_domain.Models;
using Microsoft.AspNetCore.Identity.EntityFrameworkCore;
using Microsoft.EntityFrameworkCore;

namespace flwrs_infrastructure;

public class DemoDbContext : IdentityDbContext
{
    public DemoDbContext(DbContextOptions options) : base(options)
    {}

    public DbSet<Flower> Flowers { get; set; }
    public DbSet<FlowerCategory> FlowerCategories { get; set; }
    public DbSet<Wrapper> Wrappers { get; set; }
    public DbSet<Extraltem> Extraltems { get; set; }
    public DbSet<Bouquet> Bouquets { get; set; }
    public DbSet<FlowerInBouquet> FlowersInBouquets { get; set; }
    public DbSet<Order> Orders { get; set; }
    public DbSet<BouquetInOrder> BouquetsInOrders { get; set; }
    public DbSet<ExtraltemInOrder> ExtraltemsInOrders { get; set; }

    protected override void OnModelCreating(ModelBuilder builder)
    {
        base.OnModelCreating(builder);

        builder.Entity<FlowerInBouquet>()
            .HasKey(fb => new { fb.FlowerId, fb.BouquetId });

        builder.Entity<BouquetInOrder>()
            .HasKey(bo => new { bo.OrderId, bo.BouquetId });

        builder.Entity<Flower>()
            .HasOne(f => f.Category)
            .WithMany(c => c.Flowers)
            .HasForeignKey(f => f.CategoryId);

        builder.Entity<Bouquet>()
            .HasOne(b => b.Wrapper)
            .WithMany()
            .HasForeignKey(b => b.WrapperId);

        builder.Entity<Bouquet>()
            .HasOne(b => b.CreatedByUser)
            .WithMany()
            .HasForeignKey(b => b.CreatedByUserId)
            .OnDelete(DeleteBehavior.SetNull);

        builder.Entity<Order>()
            .HasOne(o => o.User)
            .WithMany()
            .HasForeignKey(o => o.UserId);
    }
}
```

```
builder.Entity<ExtraltemInOrder>()  
    .HasKey(eio => new { eio.OrderId, eio.ExtraltemId });  
}  
}
```

Фрагмент коду відображення головної сторінки клієнтської частини

```
<!doctype html>
<html lang="en">
<head>
  <meta charset="utf-8">
  <title>Інтернет Магазин Квітів FLWRS</title>
  <base href="/">
  <meta name="viewport" content="width=device-width, initial-scale=1">
  <link rel="icon" type="image/x-icon" href="favicon.ico">
  <link href="https://fonts.googleapis.com/css2?family=Roboto:wght@300;400;500&display=swap"
rel="stylesheet">
  <link href="https://fonts.googleapis.com/icon?family=Material+Icons" rel="stylesheet">
</head>
<body>
  <app-root></app-root>
</body>
</html>
```

Фрагмент коду логіки сторінки конструктора букета

```
import { Component, OnInit } from '@angular/core';
import { Flower, FlowerCategory } from '../models/flower';
import { FlowerService } from '../services/flower.service';
import { NgForOf, NgIf } from '@angular/common';
import { environment } from '../environments/environment';
import { WrapperService } from '../services/wrapper.service';
import { Wrapper } from '../models/wrapper';
import { ExtraltemsService } from '../services/extra-items.service';
import { Extraltem } from '../models/extraltem';
import { ExtraltemListComponent } from "../components/extra-item-list/extra-item-
list.component";
import { CustomBouquetCartItem } from '../models/cart.model';
import { CartService } from '../services/cart.service';

@Component({
  selector: 'app-constructor',
  standalone: true,
  imports: [
    NgForOf,
    NgIf,
    ExtraltemListComponent
  ],
  templateUrl: './constructor.component.html',
  styleUrls: ['./constructor.component.scss']
})
export class ConstructorComponent implements OnInit {
  categories: FlowerCategory[] = [];
  selectedCategory: FlowerCategory | null = null;
  flowers: Flower[] = [];
  selectedFlowers: (Flower & { quantity: number })[] = [];
  wrappers: Wrapper[] = [];
  selectedWrapper: Wrapper | null = null;
  extraltems: Extraltem[] = [];
  apiUrl = environment.apiUrlHttps;
  isLoading: boolean = true;

  constructor(private flowerService: FlowerService, private wrapperService: WrapperService,
    private extraltemService: ExtraltemsService,
    private cartService: CartService) {}

  ngOnInit(): void {
    this.flowerService.getFlowerCategories().subscribe(data => {
      this.categories = data;
      this.selectCategory(data[0]);
      this.isLoading = false;
    });
    this.wrapperService.getWrappers().subscribe(data => {
      this.wrappers = data;
    });
  }
}
```

```

    this.selectedWrapper = data[0];
  });
  this.extraltemService.getExtraltems().subscribe(data => {
    this.extraltems = data;
  });
}

selectCategory(category: FlowerCategory): void {
  this.selectedCategory = category;
  this.flowers = category.flowers || [];
}

addFlower(flower: Flower) {
  const existing = this.selectedFlowers.find(f => f.id === flower.id);
  if (existing) {
    existing.quantity += 1;
  } else {
    this.selectedFlowers.push({ ...flower, quantity: 1 });
  }
}

increaseQuantity(flower: any) {
  flower.quantity += 1;
}

decreaseQuantity(flower: any) {
  if (flower.quantity > 1) {
    flower.quantity -= 1;
  } else {
    this.selectedFlowers = this.selectedFlowers.filter(f => f.id !== flower.id);
  }
}

deleteFlower(flower: any) {
  this.selectedFlowers = this.selectedFlowers.filter(f => f.id !== flower.id);
}

get totalFlowersPrice(): number {
  return this.selectedFlowers.reduce((sum, flower) => sum + flower.price * flower.quantity, 0);
}

get totalFlowerCount(): number {
  return this.selectedFlowers.reduce((sum, flower) => sum + flower.quantity, 0);
}

get decorationPrice(): number {
  return 50;
}

get wrappingPrice(): number {

```

```

    return this.selectedWrapper?.price ?? 0;
  }

  get deliveryPrice(): number {
    return 100;
  }

  get totalPrice(): number {
    return (
      this.totalFlowersPrice +
      this.decorationPrice +
      this.wrappingPrice +
      this.deliveryPrice
    );
  }

  addToCart(): void {
    if (this.selectedFlowers.length === 0) return;

    const flowerNames = this.selectedFlowers
      .map(f => `${f.name} × ${f.quantity}`)
      .join(', ');

    const wrapperMaterial = this.selectedWrapper?.material

    const customBouquetItem: CustomBouquetCartItem = {
      id: Date.now(),
      name: 'Кастомний букет',
      description: `Букет із ${this.totalFlowerCount} квітів: ${flowerNames}. Обгортка:
    ${wrapperMaterial}.`,
      image: 'icons/empty-constructor.png',
      price: this.totalPrice,
      quantity: 1,
      type: 'custom_bouquet',
      flowers: this.selectedFlowers.map(f => ({ ...f })))
    };

    this.cartService.addToCart(customBouquetItem);

    this.selectedFlowers = [];
    this.selectedWrapper = this.wrappers[0];
  }
}

```

Додаток 3
Копія презентації

НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
“КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ ІМЕНІ ІГОРЯ
СІКОРСЬКОГО”



ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

КАФЕДРА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ КОМП'ЮТЕРНИХ СИСТЕМ

ВЕБЗАСТОСУНОК ДЛЯ ОНЛАЙН СТВОРЕННЯ ТА КУПІВЛІ БУКЕТІВ З КВІТІВ

Виконав: студент групи КП-12 Онасенко Олексій Володимирович

Керівник: асистент кафедри ПЗКС, д-р філософії, Юсин Яків Олексійович

Київ – 2025

ПОСТАНОВКА ЗАДАЧІ



Мета проєкту: розробити вебсервіс для онлайн формування та замовлення букетів із квітів з можливістю персоналізації та зручної покупки.

Завдання:

1. Проаналізувати існуючі рішення.
2. Обрати засоби реалізації.
3. Розробити всі компоненти вебзастосунку.
4. Визначити подальші шляхи розвитку застосунку.

АКТУАЛЬНІСТЬ



1. Популярність онлайн-торгівлі.
2. Зростання попиту на індивідуальні послуги.
3. Швидкий ритм життя – зручність замовлень онлайн.
4. Актуальність доставки квітів на свята та події.
5. Недостатня кількість сервісів у цій сфері.

ІСНУЮЧІ АНАЛОГИ



ФУНКЦІОНАЛЬНІ ВИМОГИ



1. Реєстрація в системі.
2. Авторизація в системі.
3. Створення індивідуального букета.
4. Перегляд готових композицій.
5. Додавання букетів до кошика.
6. Оформлення замовлення.
7. Перегляд історії замовлень.



ЗАСОБИ РЕАЛІЗАЦІЇ

Серверна частина: мова програмування **C#** та фреймворк **ASP.NET Core**.



Клієнтська частина: мова програмування **TypeScript** та фреймворки **Angular** та **Bootstrap**.



База даних: **PostgreSQL**



PostgreSQL



АРХІТЕКТУРА РОЗРОБЛЮВАНОВОГО ПРОГРАМНОГО ЗАБЕЗБЕЧЕННЯ

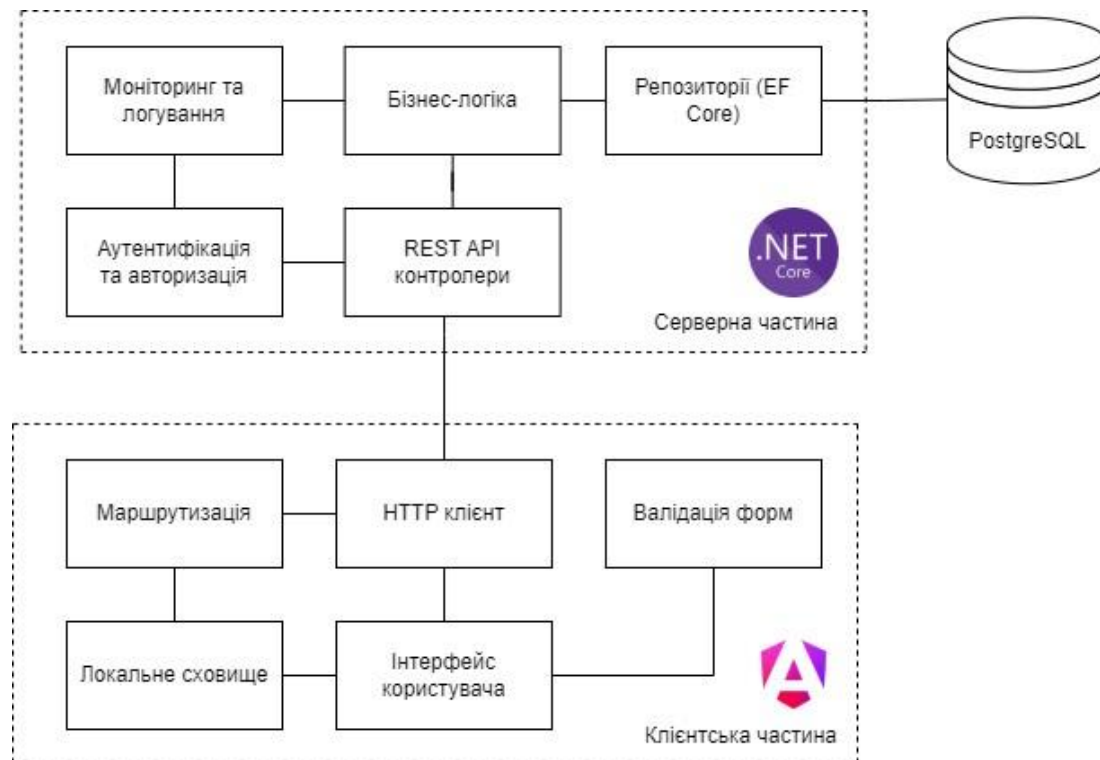
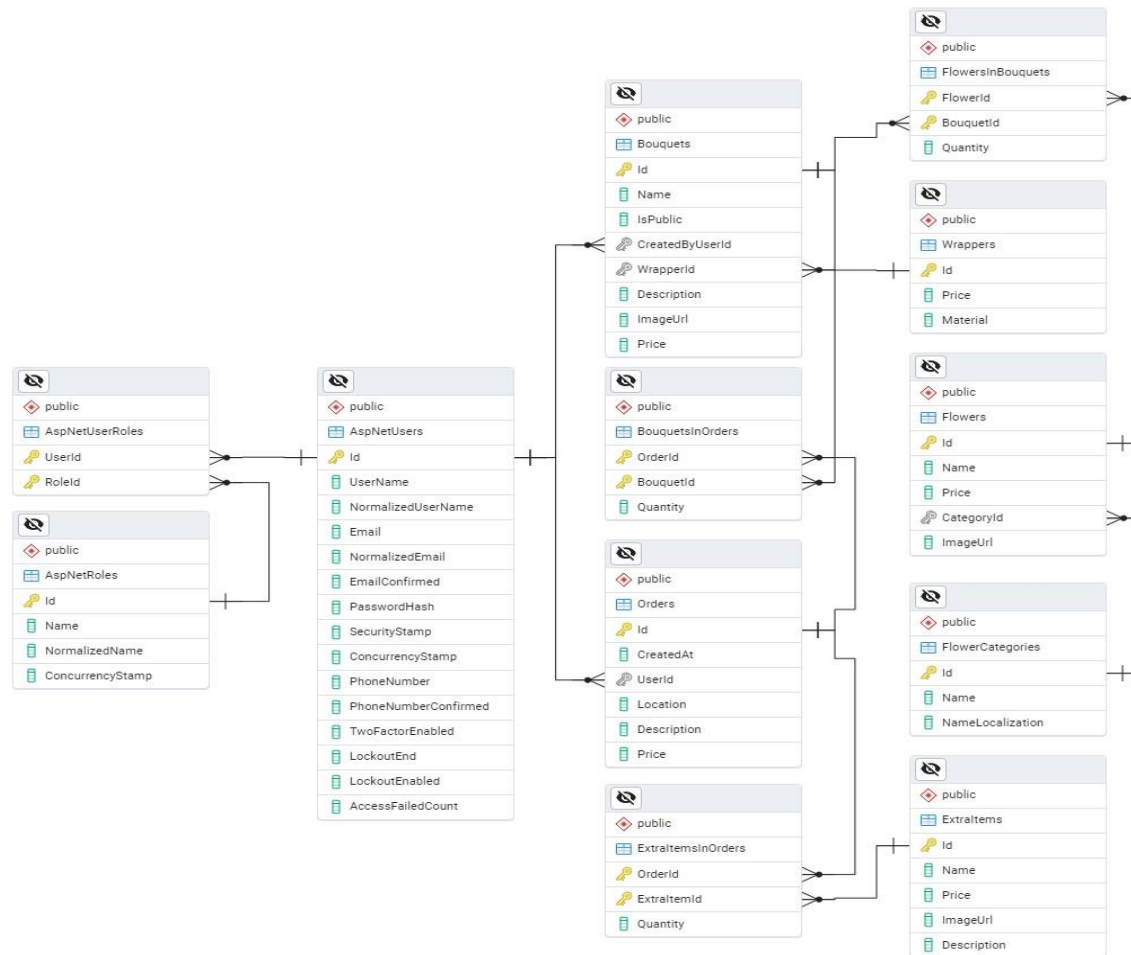
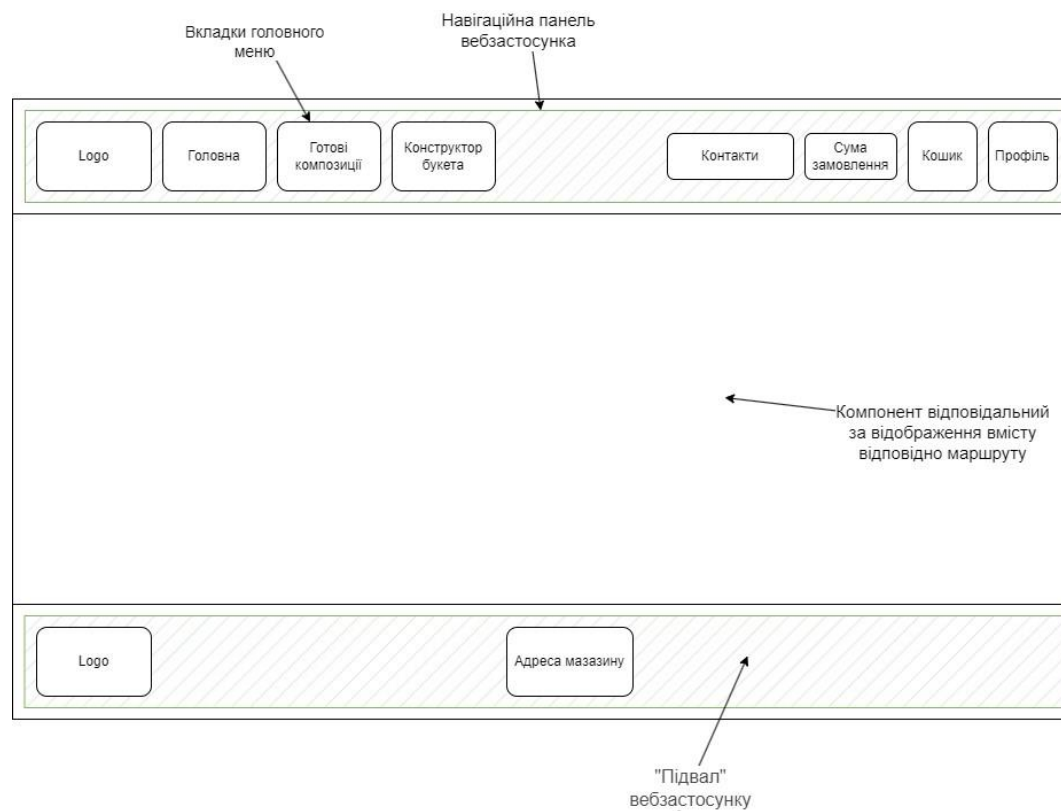


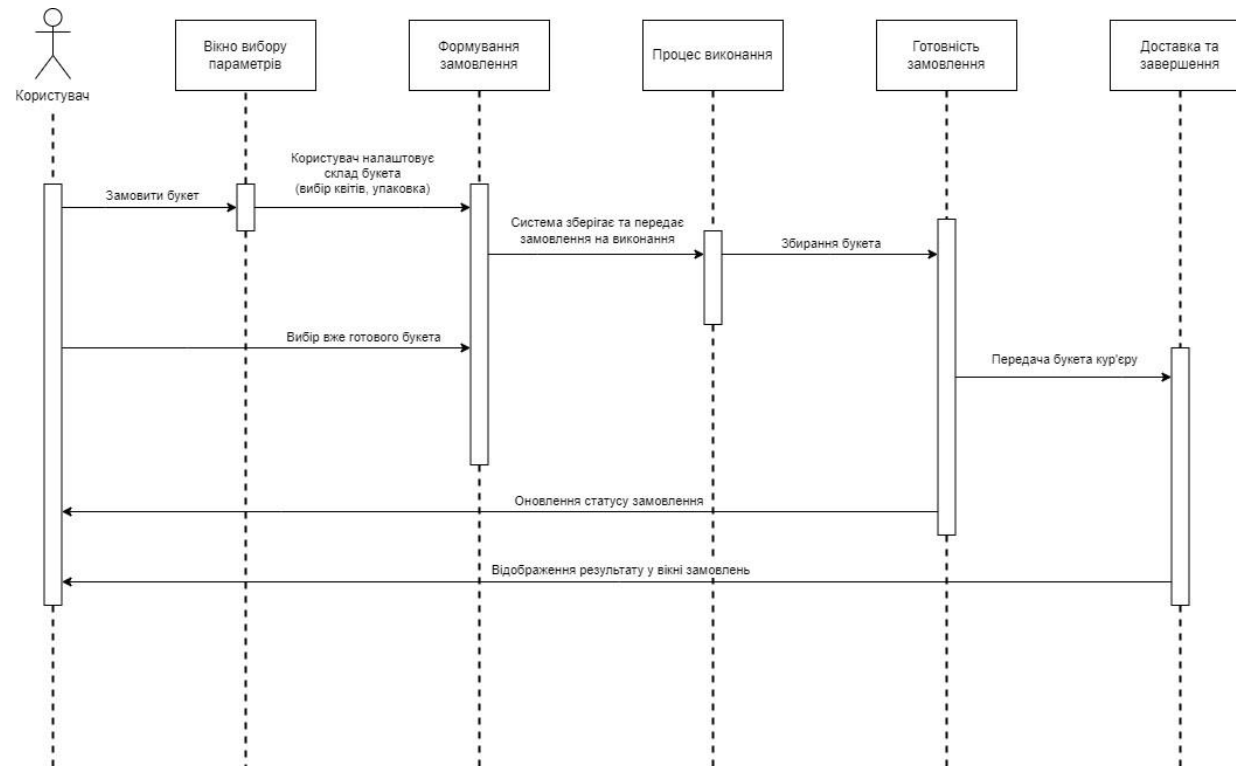
СХЕМА БАЗИ ДАНИХ



СТРУКТУРА КЛІЄНТСЬКОЇ ЧАСТИНИ



ДІАГРАМА ПОСЛІДОВНОСТІ ДЛЯ ЗАМОВЛЕННЯ БУКЕТУ

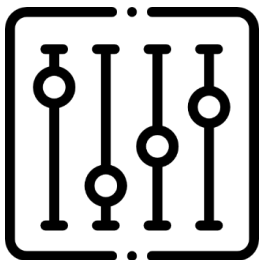


ПОРІВНЯННЯ З АНАЛОГАМИ



Застосунок Функціональні можливості	Interflora	Flowers.ua	Sunrose	FLWRS
Онлайн конструктор букетів	—	—	—	+
Можливість додати листівку або подарунок до замовлення	—	+	—	+
Кур'єрська доставка по Україні	+	+	—	+
Розширений вибір подарункових наборів	—	+	—	+
Система бонусів або програма лояльності	+	—	—	—
Підтримка мобільних пристроїв	+	+	—	+
Інтеграція з соцмережами	—	+	+	+

НАПРЯМИ РОЗВИТКУ



- Впровадження двофакторної автентифікації (2FA).
- Інтеграція системи аналітики.



- Багатомовність інтерфейсу.
- Рекомендаційна система.
- Створення онлайн чату з підтримкою.



- Співпраця з партнерами.

АПРОБАЦІЯ



Отримано акт від ФОП Марченко Валентина Петрівна про те, що результати дипломного проєкту плануються до впровадження



ВИСНОВКИ



1. Виконано аналіз існуючих програмних рішень.
2. Сформульовано функціональні вимоги до програмного забезпечення.
3. Обрано та обґрунтовано інструменти для розробки.
4. Спроектовано архітектуру системи та структуру бази даних.
5. Розроблено серверну та клієнтську частини застосунку.
6. Представленні напрямки подальшого розвитку вебзастосунку.

Розробка виконана у повному обсязі згідно з положенням Технічного завдання, тестування продукту проведено відповідно до затвердженої програми та методики тестування.



Дякую за увагу!

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2024 р.

ВЕБЗАСТОСУНОК ДЛЯ ОНЛАЙН СТВОРЕННЯ ТА КУПІВЛІ
БУКЕТІВ З КВІТІВ

Програма та методика тестування

ДП.045440-04-51

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ОНАСЕНКО

ЗМІСТ

1. Об'єкт випробувань	3
2. Мета тестування	3
3. Методи тестування.....	3
4. Засоби та порядок тестування.....	4

1. ОБ'ЄКТ ВИПРОБУВАНЬ

Вебзастосунок для онлайн створення та купівлі букетів з квітів, який являє собою вебсайт, розроблений з використанням таких технологій як: C#, ASP.NET Core, Angular, PostgreSQL.

2. МЕТА ТЕСТУВАННЯ

У процесі тестування має бути перевірено наступне:

- 1) працездатність компонентів вебсторінки та взаємодія між компонентами;
- 2) маршрутизація користувацької частини застосунку;
- 3) зручність користувацького інтерфейсу;
- 4) доступ серверної до бази даних;
- 5) взаємодія клієнтської та серверної частини застосунку;
- 6) відповідність вимогам Технічного завдання.

3. МЕТОДИ ТЕСТУВАННЯ

На кожному етапі розроблення вебзастосунку виконувалося тестування нових функціональних можливостей. Це дозволило виявити та виправити помилки на ранніх стадіях, що, у свою чергу, допомогло зекономити час і зусилля на їх усунення на більш пізніх етапах розробки.

Увесь процес розроблення супроводжувався постійним тестуванням. Згідно з обраною методикою, тестувальник виступав у ролі кінцевого користувача. Перед початком тестування визначалися межі функціональності, що підлягали перевірці, та формувався список тестових випадків.

Кожен тестовий випадок відтворювався з метою перевірки відповідності фактичних результатів очікуваням. У разі виявлення

невідповідностей виконувалося виправлення помилок у роботі застосунку, після чого тестові сценарії повторно відтворювалися для підтвердження правильності їх реалізації.

4. ЗАСОБИ ТА ПОРЯДОК ТЕСТУВАННЯ

Працездатність вебсервісу перевіряється шляхом:

- 1) динамічного ручного тестування – введенням граничних та недопустимих значень в поля, які можна редагувати;
- 2) динамічного ручного тестування на відповідність функціональним вимогам;
- 3) статичного тестування коду;
- 4) тестування застосунку у різних браузерах;
- 5) тестування стабільності роботи при різних умовах;
- 6) тестування зручності використання;
- 7) тестування інтерфейсу.

Факультет прикладної математики
Кафедра програмного забезпечення комп'ютерних систем

“ЗАТВЕРДЖЕНО”

Завідувач кафедри

_____ Євгенія СУЛЕМА

“ ____ ” _____ 2025 р.

ВЕБЗАСТОСУНОК ДЛЯ ОНЛАЙН СТВОРЕННЯ ТА КУПІВЛІ
БУКЕТІВ З КВІТІВ

Керівництво користувача

ДП.045440-05-34

“ПОГОДЖЕНО”

Керівник проєкту:

_____ Яків ЮСИН

Нормоконтроль:

_____ Микола ОНАЙ

Виконавець:

_____ Олексій ОНАСЕНКО

ЗМІСТ

1. Опис структури вебзастосунку	3
2. Головна сторінка застосунку	3
3. Сторінка «Готові композиції».....	5
4. Сторінка «Конструктор букета»	5
5. Сторінка «Кошик».....	7
6. Процедура реєстрації та авторизації	8
7. Сторінка «Оформлення замовлення».....	9
8. Сторінка «Профіль користувача»	10

1. Опис структури вебзастосунку

Вебзастосунок для онлайн створення та купівлі букетів з квітів складається з вебсторінок, які динамічно формуються у відповідь на взаємодію користувача з інтерфейсом сайту.

Вебзастосунок включає такі сторінки:

- Головна сторінка – вітальна сторінка з короткою інформацією про сервіс та популярними букетами.
- Каталог – сторінка з переглядом доступних букетів, фільтрами за категоріями, ціною тощо.
- Сторінка реєстрації – дозволяє новим користувачам створити обліковий запис.
- Сторінка авторизації – для входу до особистого кабінету.
- Конструктор букета – сторінка, на якій користувач може самостійно створити унікальний букет, обираючи квіти, оформлення тощо.
- Сторінка букета – детальний перегляд окремого букету, з описом, фото та можливістю додати до кошика.
- Профіль користувача – сторінка з особистими даними та історією замовлень.
- Сторінка оформлення замовлення – дозволяє користувачу оформити замовлення.

2. Головна сторінка застосунку

Перейшовши на сторінку «Головна» (рис. 1), користувач бачить вітальний контент, банери, а також навігацію до інших розділів сайту, яка містить наступні елементи:

- вкладка переходу на сторінку «Головна»;
- вкладка переходу на сторінку «Готові композиції»;
- вкладка переходу на сторінку «Конструктор букета»;

- інформація про години роботи магазину та контакти;
- кнопка переходу на сторінку «Профіль користувача»;
- випадючий список з товарами, що були добавлені до кошика.

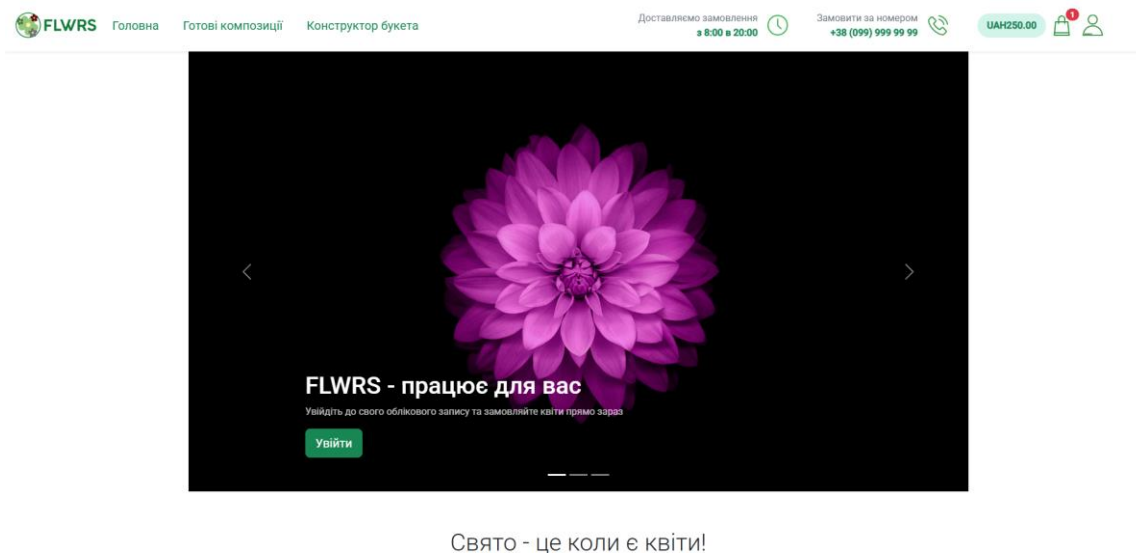


Рис. 1. Головна сторінка

Нижче на цій сторінці знаходиться список популярних букетів на сайті (рис. 2), який містить інформацію про букет: фото, назву та ціну. А також кнопка «У кошик», щоб добавити букет до кошика.

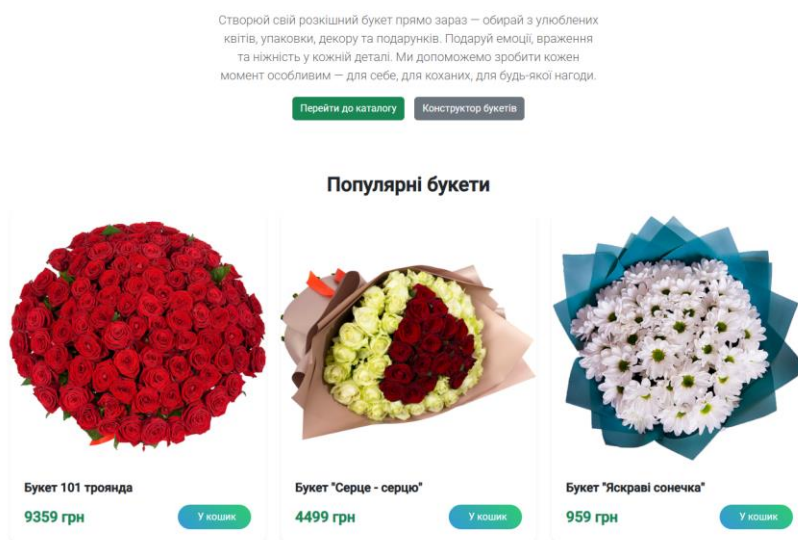


Рис. 2. Популярні букети

3. Сторінка «Готові композиції»

На цій сторінці (рис. 3) представлені готові букети, створені флористами для різних подій і настроїв. Користувач може легко знайти букет за допомогою зручного фільтра, який дозволяє обрати композиції з певними видами квітів – наприклад, троянда, хризантема, еustoma тощо. Для зручності передбачене сортування букетів:

- за популярністю – найпопулярніші букети з високими продажами;
- за ціною – у порядку зростання або спадання вартості.

Сторінка дозволяє швидко ознайомитися з доступними композиціями, переглянути детальну інформацію про кожен букет і додати його до кошика всього в кілька кліків.

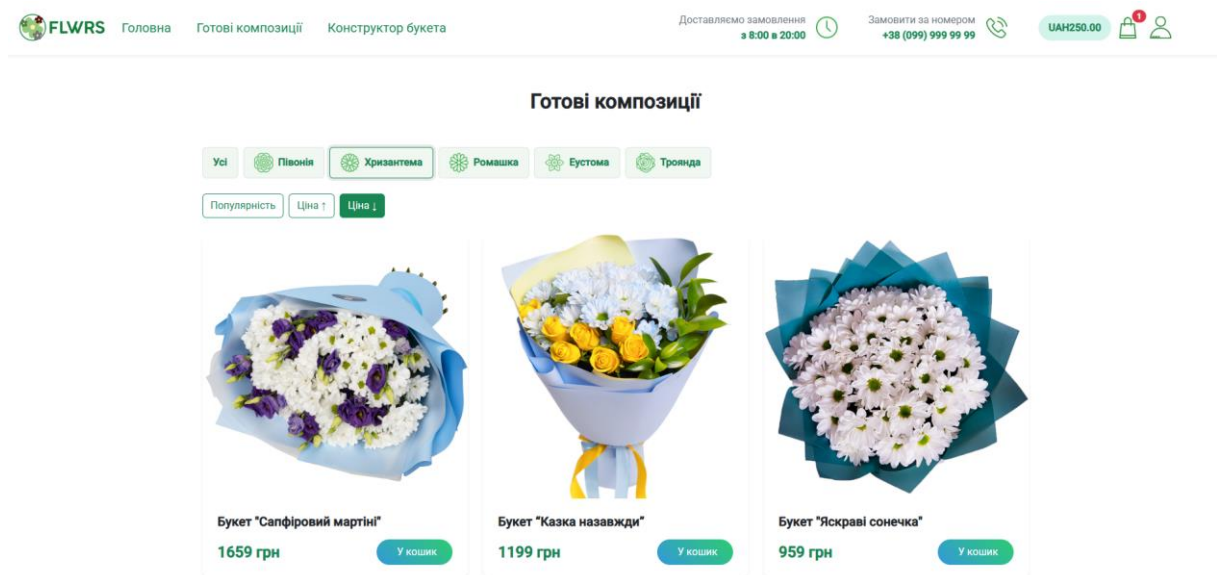


Рис. 3. Готові композиції

4. Сторінка «Конструктор букета»

Конструктор букетів – це інтерактивна сторінка (рис. 4), яка дозволяє користувачу самостійно створити унікальний букет, обравши бажані квіти, пакування та додаткові елементи оформлення.

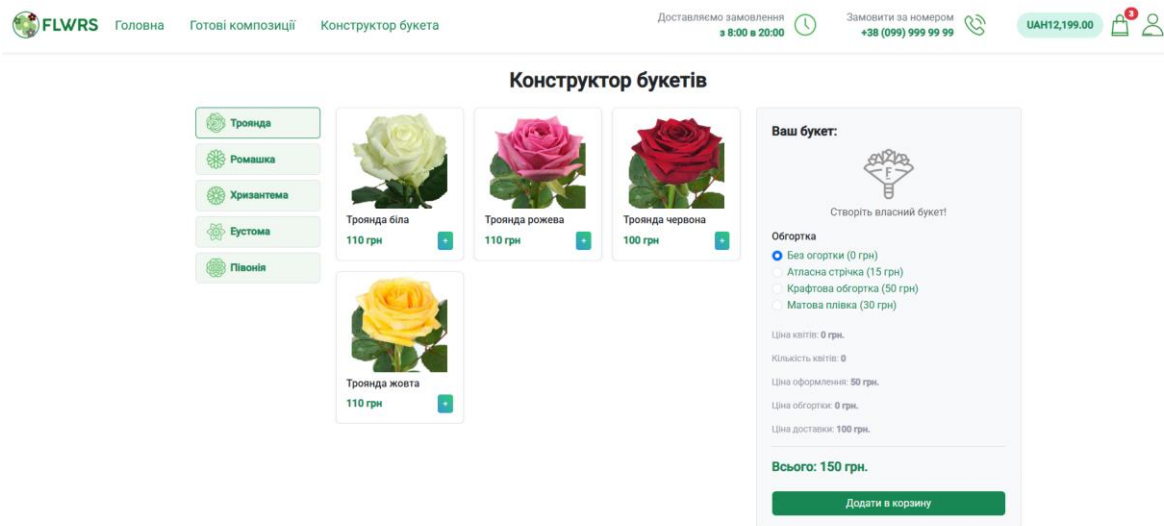


Рис. 4. Конструктор букетів

Основні елементи сторінки:

- Вибір квітів – зліва розташований фільтр за типами квітів. При виборі категорії відображаються відповідні варіанти квітів із зображенням, назвою та ціною. Кожен елемент має кнопку «+» для додавання квітки до букету.
- Попередній перегляд і конфігурація букета – праворуч знаходиться блок «Ваш букет», де відображається обрана композиція. Тут користувач бачить: загальну вартість квітів, кількість обраних квітів, вартість оформлення, обгортку, вартість доставки.
- Підсумок – у нижній частині правого блоку виводиться загальна сума замовлення з урахуванням усіх обраних параметрів. Нижче – кнопка «Додати в корзину», що дозволяє зберегти зібраний букет для подальшого оформлення замовлення.

Нижче на цій сторінці знаходиться список додаткових товарів (рис. 5), які можна придбати разом із букетом.

Рекомендуємо до покупки

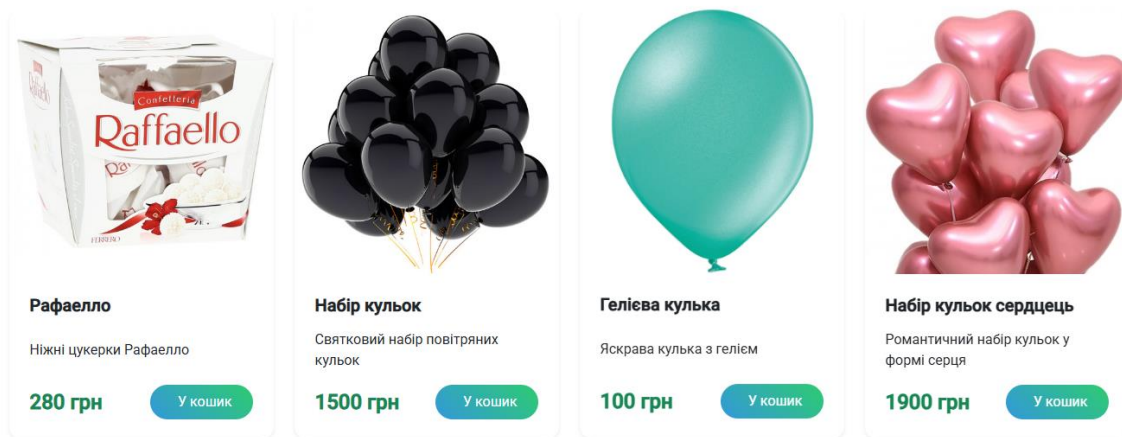


Рис. 5. Рекомендуємо до покупки

5. Сторінка «Кошик»

Сторінка кошика (рис. 6) відображає всі додані товари: букети та додаткові аксесуари. Для кожного товару показано фото, назву, опис, ціну, кількість та підсумкову вартість. Можна змінити кількість або видалити товар. Доступна кнопка очищення всього кошика. Внизу відображається загальна сума замовлення та кнопки: «Перейти до оформлення» і «Продовжити покупки».

[Головна](#)
[Готові композиції](#)
[Конструктор букета](#)

Доставляємо замовлення
з 8:00 в 20:00

Замовити за номером
+38 (099) 999 99 99

UAN12,199.00

Продукт	Назва	Опис	Ціна	Кількість	Разом	Очистити все
	Букет 101 троянда	101 червона троянда, висота 50-60 см	UAN9,359.00	- 1 +	UAN9,359.00	Видалити
	Набір кульок	Святковий набір повітряних кульок	UAN1,500.00	- 1 +	UAN1,500.00	Видалити
	Кастомний букет	Букет із 11 квітів: Троянда біла ×5, Троянда червона ×5, Троянда рожева ×1. Обгортка: Матова плівка.	UAN1,340.00	- 1 +	UAN1,340.00	Видалити
Всього:					UAN12,199.00	

Перейти до оформлення
Продовжити покупки

Рис. 6. Вікно кошику

6. Процедура реєстрації та авторизації

Щоб користувач міг оформити замовлення йому потрібно авторизуватися у системі. Для цього необхідно ввести пошту та пароль на сторінці авторизації (рис. 7) та натиснути кнопку «Увійти».

FLWRS Головна Готові композиції Конструктор букета

Доставляємо замовлення з 8:00 по 20:00

Замовити за номером +38 (099) 999 99 99

UAH250.00

Увійдіть до свого облікового запису

Електронна пошта

Пароль

Увійти

[Ще не маєте акаунту?](#)

Рис. 7. Вікно авторизації

Якщо користувач ще не має облікового запису в системі, він може його створити, натиснувши на кнопку «Ще не маєте акаунту?», після чого відкриється сторінка реєстрації (рис. 8).

FLWRS Головна Готові композиції Конструктор букета

Доставляємо замовлення з 8:00 по 20:00

Замовити за номером +38 (099) 999 99 99

UAH250.00

Створіть обліковий запис

Електронна пошта

Введіть коректну електронну пошту.

Пароль

Пароль повинен містити щонайменше 6 символів, цифру, велику і малу літеру, спецсимвол.

Підтвердіть пароль

Зареєструватися

[Уже маєте акаунт?](#)

Рис. 8. Вікно реєстрації

Щоб створити обліковий запис, необхідно ввести наступні дані: електронну пошту, пароль та підтвердження паролю. Після введення всіх даних потрібно натиснути на кнопку «Зареєструватися». Якщо обліковий запис буде успішно створено, то користувачу відкриється повідомлення про це.

7. Сторінка «Оформлення замовлення»

Сторінка оформлення замовлення (рис. 9) дає змогу користувачеві завершити покупку. У верхній частині форми користувач заповнює обов'язкові поля: місце доставки, номер телефону та коментар до замовлення.

Нижче автоматично відображається перелік товарів, доданих до кошика: букети та додаткові товари. Кожен елемент у списку супроводжується коротким описом, кількістю та ціною. Внизу підраховується загальна сума замовлення.

Після натискання кнопки «Підтвердити замовлення» система обробляє введені дані та виведе користувачу повідомлення про успішне оформлення замовлення, після чого його можна побачити в профілі користувача.

FLWRS Головна Готові композиції Конструктор букета

Доставляємо замовлення з 8:00 до 20:00

Замовити за номером +38 (099) 999 99 99

UAH12,199.00

Оформлення замовлення

Місце доставки

Номер телефону

Опис замовлення

Товари у кошику

Букет 101 троянда (x1) 101 червона троянда, висота 50-60 см	UAH9,999.00
Набір кулькок (x1) Семісезонний набір повітряних кульок	UAH1,000.00
Кастомний букет (x1) Букет із 11 квіток: Троянда біла x5, Троянда червона x5, Троянда рожева x1, Обгорток: Маття півка.	UAH1,199.00

Всього: UAH12,199.00

Після оформлення з Вами зв'яжеться наш менеджер

Підтвердити замовлення

Рис. 9. Вікно оформлення замовлення

8. Сторінка «Профіль користувача»

Сторінка «Профіль користувача» (рис. 10) призначена для перегляду особистої інформації користувача та всієї історії його замовлень у системі. У верхній частині сторінки відображається електронна адреса користувача, яка була вказана під час реєстрації або входу до облікового запису. Тут також розміщено кнопку «Вийти», що дозволяє швидко завершити сеанс.

Нижче знаходиться блок із заголовком «Історія замовлень», де у зручному форматі виводиться список усіх попередніх замовлень користувача. Для кожного замовлення відображаються основні деталі: дата й час оформлення, адреса доставки, короткий опис або коментар до замовлення, загальна вартість покупки, а також перелік придбаних товарів.

Товари в замовленні згруповані за категоріями: букети та додаткові товари. Кожен товар має зображення, назву, ціну за одиницю та зазначену кількість. Це дає змогу користувачу легко згадати зміст кожного замовлення або швидко повторити його у майбутньому.

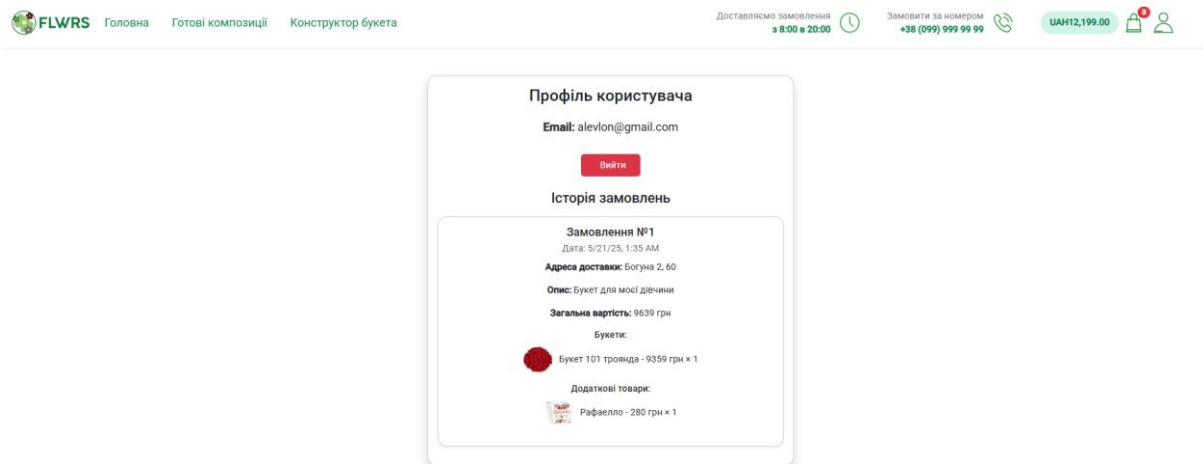


Рис. 10. Вікно профілю користувача