

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Комп'ютерних наук і технологій
(повне найменування факультету)

Комп'ютерні системи та мережі
(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалаврський
(ступінь вищої освіти)

на тему: РОЗРОБКА КОМП'ЮТЕРНОЇ СИСТЕМИ ВЕБРЕСУРСУ
СТУДІЇ «CREATIVEPEOPLE»

Виконав(ла): студент(ка) 4 курсу,
групи КНТз-513сп

Спеціальності

123 Комп'ютерна інженерія
(код і найменування спеціальності)

Освітня програма (спеціалізація)

Комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ШАПОВАЛ М.О.

(ПРИЗВИЩЕ та ініціали)

Керівник СКРУПСЬКИЙ С.Ю.

(ПРИЗВИЩЕ та ініціали)

Рецензент ЩЕРБАКОВ В.І.

(ПРИЗВИЩЕ та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет _____ комп'ютерних наук і технологій
Кафедра _____ комп'ютерних систем та мереж
Ступінь вищої освіти _____ бакалаврський
Спеціальність _____ 123 комп'ютерна інженерія
(код і найменування)
Освітня програма (спеціалізація) _____ комп'ютерна інженерія
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри КУДЕРМЕТОВ Р.К.

« » _____ 2026 року

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

ШАПОВАЛ Марини Олександрівни

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Розробка комп'ютерної системи вебресурсу студії «CreativePeople»

керівник проєкту (роботи) К.Т.Н., доцент, СКРУПСЬКИЙ С.Ю.

(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від « 14 » квітня 2026 року № 160

2. Строк подання студентом проєкту (роботи) 01.06.2026 р.

3. Вихідні дані до проєкту (роботи) опис предметної області, технології розробки клієнтської та серверної частини

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1) Аналіз технічного завдання;

2) Моделювання програмного забезпечення;

3) Проєктування системи;

4) Реалізація вебресурсу.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, кількість слайдів, плакатів)

Слайди презентації

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4	СКРУПСЬКИЙ С.Ю.		
Нормоконтроль	ДЬЯЧУК Т.С.		

7. Дата видачі завдання «14» квітня 2026 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз предметної області	до 18.04.2026	
2	Визначення та аналіз вимог до системи	до 21.04.2026	
3	Аналіз технологій реалізації вебсервісу	до 25.04.2026	
4	Моделювання програмного забезпечення	до 28.04.2026	
5	Проектування архітектури вебсервісу	до 10.05.2026	
6	Проектування бази даних системи	до 14.05.2026	
7	Розробка frontend-частини	до 18.05.2026	
8	Реалізація backend-частини	до 22.05.2026	
9	Тестування функціональності сервісу	до 24.05.2026	
10	Оформлення пояснювальної записки	до 26.05.2026	
11	Проходження нормоконтролю	до 30.05.2026	
12	Перевірка на наявність академічного плагіату	до 01.06.2026	
13	Проходження рецензування	до 02.06.2026	

Студент(ка) _____
(підпис)

Марина ШАПОВАЛ
(Ім'я ПРИЗВИЩЕ)

Керівник проєкту (роботи) _____
(підпис)

Степан СКРУПСЬКИЙ
(Ім'я ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра:
66 с., 50 рис., 9 табл., 1 дод., 21 джерел.

CSS, HTML, JAVASCRIPT, SQLITE, АРХІТЕКТУРА КЛІЄНТ-СЕРВЕР,
БАЗА ДАНИХ, ВЕБРЕСУРС, СИСТЕМА КЕРУВАННЯ ВМІСТОМ

Предмет розробки – клієнт-серверна комп'ютерна система для вебресурсу студії «CreativePeople».

Мета роботи – підвищення ефективності роботи компанії за рахунок використання комп'ютерної системи.

Практична цінність роботи полягає в отриманні результатів, які можуть стати основою для створення веб-ресурсів, спрямованих на ефективне представлення компаній, продуктів або послуг в Інтернеті.

В межах роботи виконано аналіз предметної області та існуючих технологій для реалізації вебресурсу, визначено функціональні та нефункціональні вимоги до системи. Описано моделювання програмного забезпечення, наведено діаграму використання (допомагає визначити взаємодію користувачів з вебдодатком на рівні бізнес-логіки, не вдаючись у деталі реалізації), діаграму послідовності (ілюструє процес подачі заявки користувачем на вебресурсі), діаграму розгортання (показує фізичні компоненти вебресурсу), діаграму компонентів (показує, як різні компоненти системи взаємодіють один з одним). Описано архітектуру вебсервісу та розглянуто проєктування бази даних (визначені таблиці та зв'язки між ними). Проведено реалізацію вебсервісу (розробка frontend-частини та backend-частини, інтеграція frontend та backend) та тестування функціональності вебсервісу студії «CreativePeople».

ЗМІСТ

Вступ.....	6
1 Аналіз технічного завдання.....	8
2 Моделювання програмного забезпечення	16
3 Проєктування системи.....	22
4 Робробка вебресурсу	34
4.1 Розробка frontend-частини.....	34
4.2 Реалізація backend-частини	39
4.3 Інтеграція frontend та backend.....	43
4.4 Тестування функціональності сервісу	44
Висновки	54
Перелік джерел посилання	56
Додаток А.....	58

ВСТУП

У сучасному світі цифрової трансформації вебресурси набувають особливого значення як для комерційних, так і для некомерційних організацій. Вони виконують функцію не тільки віртуального представництва, але й слугують потужним засобом взаємодії з користувачами, забезпечуючи цілодобовий доступ до інформації, товарів і послуг.

Швидкий розвиток інтернет-технологій, зростання функціональних можливостей браузерів та збільшення кількості мобільних користувачів підштовхують до необхідності створення зручних, адаптивних, швидких і багатофункціональних вебресурсів. Більшість досліджень та практичних рішень у галузі веброзробки зосереджені на вдосконаленні користувацького інтерфейсу, забезпеченні позитивного досвіду користувача, а також впровадженні сучасних фреймворків і бібліотек, які полегшують процес розробки та підвищують якість кінцевого продукту.

Вебресурс відіграє ключову роль у формуванні цифрової присутності бізнесу, впливаючи на його імідж і конкурентні позиції на ринку. Це зумовлює важливість створення сучасного вебсайту, який відповідатиме вимогам адаптивності, зручності використання, інтерактивності та естетичного дизайну, адже такі характеристики є невід'ємною частиною ефективної цифрової стратегії компанії.

Предметом розробки є створення вебресурсу, включаючи проектування інтерфейсу та реалізацію функціональних компонентів. Метою роботи є підвищення ефективності роботи компанії за рахунок використання комп'ютерної системи.

У першому розділі виконано аналіз предметної області та існуючих технологій для реалізації вебресурсу, визначено функціональні та нефункціональні вимоги до системи.

У другому розділі описано моделювання програмного забезпечення.

Наведено діаграму використання (допомагає визначити взаємодію користувачів з вебдодатком на рівні бізнес-логіки, не вдаючись у деталі реалізації), діаграму послідовності (ілюструє процес подачі заявки користувачем на вебресурсі), діаграму розгортання (показує фізичні компоненти вебресурсу), діаграму компонентів (показує, як різні компоненти системи взаємодіють один з одним). Побудовано концентральну модель бази даних: визначені таблиці та зв'язки між ними.

У третьому розділі виконано проєктування вебсервісу, описано архітектуру системи.

У четвертому розділі описано реалізацію вебсервісу. Описано розробка frontend-частини та backend-частини, інтеграція frontend та backend, тестування функціональності.

Практична цінність роботи полягає в отриманні результатів, які можуть стати основою для створення вебресурсів, спрямованих на ефективне представлення компаній, продуктів або послуг в Інтернеті. Розроблений веб-сайт слугує прикладом адаптивного ресурсу з інтерактивними елементами, що дозволяє його масштабувати та застосовувати для подальшого розширення онлайн-присутності бізнесу.

1 АНАЛІЗ ТЕХНІЧНОГО ЗАВДАННЯ

Вебресурс студії «CreativePeople» розробляється як сучасне цифрове представництво для агентства, що надає комплексні послуги у сфері консалтингу, маркетингу, веброзробки та дизайну.

Цільову аудиторію (користувачів ресурсу) можна розділити на кілька ключових сегментів:

- представники малого та середнього бізнесу (SMEs) (власники компаній та приватні підприємці, які шукають надійного партнера для масштабування своєї діяльності, оптимізації внутрішніх процесів або запуску нових продуктів на ринку);
- маркетологи та бренд-менеджери (фахівці, які потребують професійних послуг з просування, розробки digital-стратегій, створення контенту для соціальних мереж або налаштування таргетованої реклами);
- стартапи та технологічні проекти (команди, яким необхідна розробка візуального стилю, інтерфейсів для мобільних додатків (UX/UI дизайн) або створення сучасних корпоративних вебсайтів «під ключ» для залучення інвестицій та клієнтів.

Головна мета проєкту - трансформувати відвідувачів сайту на потенційних клієнтів та сформувати високий рівень довіри до бренду. Вебресурс успішно вирішує такі завдання:

- презентація експертизи та позиціонування компанії;
- демонстрація практичного досвіду;
- забезпечення прозорості та соціального доказу;
- автоматизація збору потенційних клієнтів та зворотного зв'язку;
- утримання аудиторії та робота з лояльністю;
- адаптивність та доступність.

Вебресурс чітко комунікує головну ціннісну пропозицію компанії - надання ефективних рішень для бізнесу. Через розділ переваг користувач одразу розуміє

філософію та сильні сторони команди: орієнтація на стратегію, суворе дотримання дедлайнів (пунктуальність), старанність та використання передових технологій.

Однією з найважливіших функцій сайту є розділ «Портфоліо». Завдяки зручному розподілу за категоріями, потенційний клієнт може швидко ознайомитися з релевантними кейсами у сферах:

- розробки вебсайтів (інтернет-магазини, посадкові сторінки тощо);
- проектування мобільних додатків (інтерфейси для банкінгу, медитацій тощо);
- графічного дизайну та ілюстрацій;
- маркетингових кампаній.

Наявність детальних описів до кожного кейсу допомагає користувачеві оцінити глибину занурення команди в бізнес-проблематику.

Знайомству з командою, візуалізація ключових фахівців (дизайнерів, розробників, маркетологів) робить бренд «людяним» та відкритим.

Демонстрація логотипів компаній, які вже співпрацюють з агентством, виступає гарантом якості.

Ресурс виступає активним інструментом продажів. Завдяки наявності інтерактивних кнопок виклику дії та вбудованої модальної форми зворотного зв'язку, клієнт може в один клік залишити свої контактні дані (ім'я, телефон, email) та короткий коментар до задачі. Це оптимізує роботу менеджерів із продажу, які отримують уже прогріту та готову до діалогу аудиторію.

Наявність форми підписки у футері сайту дозволяє компанії збирати власну базу зацікавлених користувачів. У перспективі це дає можливість запускати регулярні розсилки з корисними кейсами, новинами ринку чи спеціальними пропозиціями, утримуючи увагу клієнтів та стимулюючи повторні продажі.

Вебресурс вирішує задачу зручного доступу до інформації з будь-якого пристрою. Завдяки оптимізації під мобільні, планшетні та десктопні екрани, а також наявності мобільного навігаційного меню, користувач отримує однаковий рівень комфорту при взаємодії з ресурсом як у дорозі зі смартфона, так і в офісі за

комп'ютером.

Для розробки вебресурсу студії «CreativePeople», яка спеціалізується на веброзробці, необхідно врахувати як функціональні, так і нефункціональні вимоги. Це дасть змогу створити платформу, що відповідатиме потребам клієнтів і забезпечуватиме ефективну роботу сервісу. Функціональні вимоги визначають чітко окреслені функції та можливості, які система має виконати для задоволення запитів користувачів і досягнення бізнес-цілей [1]. Вони охоплюють те, що саме система повинна реалізовувати. До цього належать базові операції, процеси та дії, які система повинна забезпечити для користувачів чи інтеграції з іншими системами. У таблиці 1.1 наведено перелік функціональних вимог до розроблюваної системи.

Таблиця 1.1 – Функціональні вимоги до системи

Вимога	Опис
Оформлення запитів на розробку сайту	Користувачі повинні мати можливість оформити заявку на послугу за допомогою зручної форми
Інтерактивна галерея робіт	Студія має можливість демонструвати свої попередні проекти за допомогою інтерактивної галереї
Контактна форма	Має бути можливість зворотнього зв'язку з командою студії через контактну форму на сайті
Оцінка вартості послуг	Користувач повинен мати можливість отримати попередню вартості проєкту, заповнивши форму
Публікація новин та оновлень	Адміністратор повинен мати можливість додавати новини та оновлювати інформацію на сайті
Інтеграція з соціальними мережами	Вебресурс повинні мати можливість ділитися контентом та проєктами студії між користувачами через соціальні мережі
Пошукова функціональність	Користувачі повинні мати можливість зробити пошук потрібних їм проєктів та послуг за допомогою пошукової системи

Нефункціональні вимоги визначають, яким чином система повинна функціонувати. Вони не належать до окремих функцій, а зосереджуються на якості роботи системи, її продуктивності, безпеці, масштабованості та інших ключових аспектах. Такі вимоги встановлюють обмеження або стандарти, яких

система має дотримуватись для забезпечення ефективної роботи в умовах реального використання. У таблиці 1.2 наведено нефункціональні вимоги для системи.

Таблиця 1.2 – Нефункціональні вимоги до системи

Вимога	Опис
Продуктивність	Вебресурс повинен швидко завантажуватись (максимум 3 секунди на сторінці)
Безпека	Всі персональні дані користувачів повинні бути захищені за допомогою шифрування та безпечного з'єднання (HTTPS)
Масштабованість	Система повинна мати можливість масштабування для підтримки великої кількості користувачів та наданих їм послуг (проектів)
Доступність	Вебресурс має бути доступний для всіх користувачів, включаючи людей з обмеженими можливостями (ADA compliance)
Адаптивний дизайн	Вебресурс повинен коректно відображатися на різних пристроях: мобільних телефонах, планшетах та десктопах
Надійність	При високому навантаженні система повинна бути стабільною і працювати без збоїв
Локалізація	Вебресурс має підтримувати кілька мов, щоб задовольнити потреби різних регіонів (наприклад, англійська та українська)
Оновлення та підтримка	Система повинна підтримувати автоматичні оновлення та зручний механізм для обслуговування

Функціональні вимоги визначають, які завдання система повинна виконувати, щоб задовольнити потреби користувачів і досягти бізнес-цілей. Вони охоплюють ключові аспекти взаємодії з вебресурсом, такі як реєстрація та аутентифікація користувачів, подання заявок на послуги, перегляд портфоліо через інтерактивну галерею, комунікація з командою за допомогою контактної форми та отримання попередньої оцінки вартості послуг. Важливими також є інтеграція з соціальними мережами та пошукова функція, які сприяють зручності для користувачів і підвищують видимість студії.

Нефункціональні вимоги стосуються характеристик системи, що

забезпечують її якість і ефективність. До таких вимог належать продуктивність, безпека, масштабованість, доступність та адаптивність. Виключно важливим є зменшення часу завантаження сайту до 3 секунд, щоб підтримувати високий рівень задоволеності користувачів. Безпека даних повинна гарантуватися за рахунок використання шифрування та безпечного з'єднання через HTTPS. Масштабованість має забезпечувати зростання системи разом із бізнесом, підтримуючи збільшення потоків користувачів і кількість проєктів. Доступність і адаптивний дизайн гарантують комфорт користувачів, у тому числі людей з обмеженими можливостями, а також коректне відображення на різних пристроях. Стабільна робота системи забезпечується завдяки надійності платформи та автоматичним оновленням, що мінімізують можливість збоїв навіть за високих навантажень. Локалізація сайту відкриває шлях до різних мовних аудиторій, сприяючи виходу на нові ринки й розширенню клієнтської бази.

Для створення вебресурсу для студії «CreativePeople», яка спеціалізується на веброзробці, вибір відповідного технологічного стеку є надзвичайно важливим етапом. Обране рішення має забезпечувати ефективність, високу швидкість, надійність і зручність розробки, а також надавати можливість подальшого масштабування системи у разі необхідності. Основу веброзробки складають такі базові технології, як HTML, CSS і JavaScript [2-4]. Вони визначають структуру, стиль і функціональність користувацького інтерфейсу, що є фундаментом будь-якого вебресурсу. HTML є стандартною мовою розмітки, призначеною для створення вебсторінок. Вона дозволяє створювати структуру контенту, впорядковуючи його у вигляді заголовків, абзаців, таблиць, списків, посилань, зображень та інших елементів. Як основний будівельний блок вебресурсів [2]. HTML визначає всі елементи сторінки за допомогою спеціальних тегів. Ці теги мають характерну структуру: вони відкриваються і закриваються, щоб позначити початок і кінець кожного елемента. Таким чином, HTML забезпечує базис для створення інтуїтивно зрозумілих і функціональних вебдодатків.

HTML володіє широким набором тегів, які забезпечують семантичне оформлення контенту. Він підтримує інтеграцію мультимедійних елементів на

веб-сторінки, зокрема зображень за допомогою тега `<img>`, відео через тег `<video>` та аудіо за допомогою тега `<audio>`. Крім цього, HTML дає змогу створювати форми для збору та обробки даних користувачів.

HTML об'єднує всі ці елементи в єдину структуру, яка візуалізується в браузері, формуючи базовий каркас будь-якого вебресурсу.

CSS виступає мовою стилів, що задає зовнішній вигляд HTML-елементів на вебсторінці [3]. Завдяки CSS можна налаштовувати кольори, шрифти, розміри, відступи, рамки та позиціонування елементів. Вона дозволяє створювати сучасний і привабливий дизайн вебсторінок, а також забезпечує їх адаптивність для різних пристроїв. Для застосування стилів CSS використовує селектори, які можуть бути пов'язані з тегами, класами, ідентифікаторами чи атрибутами.

Кожен елемент на вебсторінці представляє собою прямокутний контейнер, що складається з вмісту, внутрішніх відступів (`padding`), рамок (`border`) та зовнішніх відступів (`margin`). Завдяки CSS можна детально налаштовувати розміри елементів, а також відстані між ними. Цей інструмент дозволяє змінювати стиль тексту, задавати розміри та товщину шрифтів, налаштовувати міжрядковий інтервал і багато іншого.

CSS також відкриває можливості для позиціонування елементів на сторінці за допомогою властивостей `position`, `top`, `left`, `right` і `bottom`. Для створення складних макетів можна використовувати сучасні методи верстання, такі як `Flexbox` і `Grid`. За допомогою медіазапитів (`media queries`) стає можливим розробка адаптивного дизайну, який автоматично підлаштовується під різні розміри екранів - від мобільних пристроїв до планшетів і настільних комп'ютерів. Завдяки CSS можна реалізовувати широкі можливості для створення естетичних і зручних інтерфейсів, що забезпечують привабливий вигляд та функціональність вебресурсів.

JavaScript - це популярна мова програмування, яка використовується для додавання інтерактивності на вебсторінки [4]. Вона дозволяє реалізовувати функції, такі як обробка подій, робота з HTML-елементами, виконання асинхронних запитів до сервера і внесення змін на сторінку без необхідності її

перезавантаження. Завдяки цьому мова забезпечує ефективну взаємодію користувачів із вебресурсом, наприклад, обробку кліків, введення тексту чи роботу з елементами при наведенні курсора. JavaScript тісно інтегрується з DOM (Document Object Model), що дозволяє розглядати HTML-елементи як об'єкти, а також динамічно змінювати контент, структуру та стилі сторінки. Однією з ключових переваг є підтримка асинхронного виконання коду, що дає можливість здійснювати серверні запити через AJAX або Fetch API без блокування основного робочого потоку. Це забезпечує плавність роботи додатків і прискорює їх взаємодію з сервером. Мова широко використовується для створення динамічних ефектів, як-от анімації, зміни стилів, модальні вікна чи слайдери, що значно покращують користувацький досвід. На базі JavaScript створено безліч потужних фреймворків і бібліотек, включаючи React, Vue.js і Angular, які дозволяють розробляти складні та інтерактивні веб-додатки з меншими затратами часу та ресурсів [5-8]. Загалом JavaScript є важливою складовою сучасної веброзробки, дозволяючи створювати додатки з високим рівнем інтерактивності та комфорту для користувачів.

Для реалізації серверної частини вебресурсу було обрано Python у поєднанні з фреймворком Flask. Такий вибір зумовлений простотою використання, гнучкістю та високою швидкістю розробки. Python є однією з найпопулярніших мов програмування, яка вирізняється зрозумілим синтаксисом, високою читабельністю та широким набором бібліотек, що дозволяють ефективно створювати вебдодатки [9]. Flask, у свою чергу, являє собою мінімалістичний фреймворк для Python, який забезпечує швидке налаштування серверної частини та обробку HTTP-запитів [10]. Такий підхід ідеально підходить для проєктів малого та середнього масштабу, де важливі простота рішення та можливість подальшого додавання функціональних здібностей. Flask надає базові інструменти для створення функціональної серверної частини: маршрутизацію запитів, обробку форм, інтеграцію з базами даних через SQLAlchemy тощо. Враховуючи потребу студії «CreativePeople» у простому механізмі для зберігання даних, було вирішено використовувати SQLite як базу даних [11]. Це легка

реляційна база даних, яка не потребує окремого сервера і зручно інтегрується з Python за допомогою SQLAlchemy. SQLite ідеально підходить для невеликих проєктів, забезпечуючи швидке й просте налаштування без необхідності складної конфігурації серверних систем. У таблиці 1.3 наведено порівняльну характеристику технологій, що увійшли до вибраного стеку.

Таблиця 1.3 – Порівняння технологій для стеку

Компонент	Технологія	Переваги	Недоліки
Frontend	HTML, CSS, JavaScript	Стандарти для веброзробки. Підтримка всіх сучасних браузерів. Велика кількість фреймворків та бібліотек для швидкої розробки.	Потребує знань у галузі адаптивного дизайну та кросбраузерної сумісності. Без сторонніх бібліотек складно створити складні інтерфейси.
Backend	Python (Flask)	Простота в освоєнні та читабельність коду. Швидка розробка завдяки мінімалістичній структурі Flask. Велика кількість бібліотек для різних завдань.	Може бути недостатньо швидким для дуже високонавантажених додатків. Потребує додаткових інструментів для великих проєктів.
База даних	SQLite	Легка в налаштуванні, не потребує окремого сервера. Добре підходить для невеликих проєктів. Підтримка SQL-операцій через SQLAlchemy.	Обмежена масштабованість для великих проєктів. Можуть виникати проблеми з обробкою великих обсягів даних.

Вибір технологічного стеку для створення вебресурсу студії «CreativePeople» ґрунтується на прагненні досягти балансу між простотою, ефективністю та гнучкістю в умовах обмежених ресурсів. Технології HTML, CSS і JavaScript забезпечують розробку стандартного й інтерактивного інтерфейсу, що є невід’ємною складовою будь-якого вебпроєкту. Використання Python разом із фреймворком Flask дозволяє швидко та зручно реалізувати серверну частину з

мінімальними зусиллями на початкове налаштування, зберігаючи потенціал для майбутнього масштабування. Водночас база даних SQLite виступає оптимальним рішенням для невеликих проєктів, де немає необхідності впроваджувати складну інфраструктуру. Така комбінація технологій ідеально підходить для створення вебресурсу студії, забезпечуючи швидкість розробки, легкість у підтримці та зниження витрат на інфраструктуру.

2 МОДЕЛЮВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Для розробки чіткої архітектури веб-додатку слід створити кілька видів діаграм, які демонструють ключові аспекти функціонування системи. Діаграма використання (Use Case Diagram) призначена для відображення функцій системи, доступних різним типам користувачів (акторів) [12]. Вона дозволяє ефективно визначити способи взаємодії користувачів із веб-додатком на рівні бізнес-логіки без заглиблення у технічні деталі реалізації. На рисунку 2.1 представлено приклад діаграми використання.



Рисунок 2.1 – Діаграма використання

Діаграма використання демонструє взаємодію користувачів і адміністратора з вебресурсом студії. Користувачі мають можливість зареєструватися, подати заявку на отримання послуги, залишити відгук та ознайомитися з доступними послугами. Адміністратор, у свою чергу, може управляти послугами, переглядати заявки, опрацьовувати їх і аналізувати відгуки. Визначена діаграма надає загальне уявлення про ключові функції системи як для користувачів, так і для

адміністратора.

Діаграма послідовності показує взаємодію між об'єктами системи в рамках визначеної послідовності. Вона ілюструє процес обміну повідомленнями між об'єктами та виконання операцій в межах конкретного сценарію роботи. На рисунку 2.2 представлено графічний зображення цієї діаграми, яке деталізує перебіг відповідних процесів.

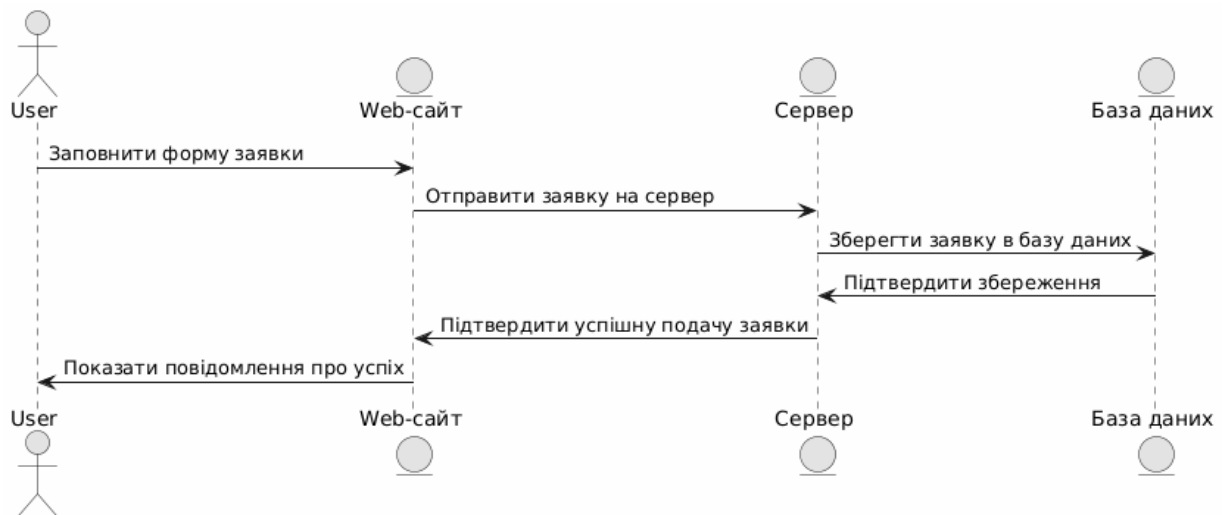


Рисунок 2.2 – Діаграма послідовності

Діаграма послідовності демонструє процес подання заявки користувачем на вебплатформі. Спочатку користувач заповнює форму, яка надсилається на сервер. Сервер зберігає дані заявки у базі, отримує підтвердження про успішне збереження та відправляє користувачу повідомлення про успішне завершення процедури. Завдяки такій діаграмі можна чітко побачити взаємодію елементів системи під час виконання цього процесу.

Діаграма розгортання ілюструє взаємодію фізичних елементів, таких як системи, сервери, бази даних та пристрої користувачів, у межах функціонування вебдодатку. Вона дозволяє зрозуміти, де саме розгорнуті компоненти системи і як вони співпрацюють для досягнення заданих цілей. На рисунку 2.3 представлена діаграма розгортання, яка наочно демонструє ці аспекти.

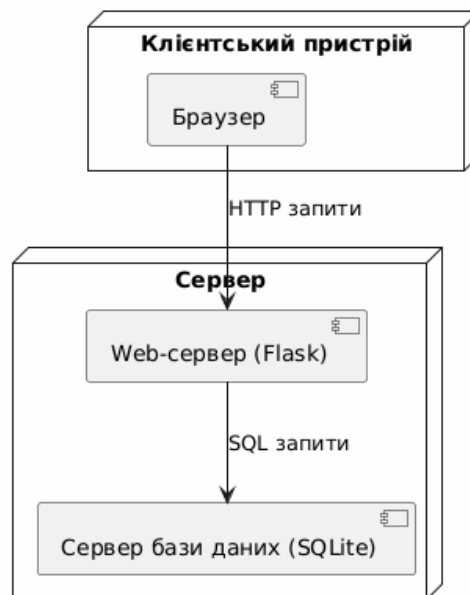


Рисунок 3.3 – Діаграма розгортання

Діаграма розгортання ілюструє фізичну структуру вебресурсу та взаємодію його частин. Клієнтський пристрій, наприклад браузер, передає запити до web-сервера, який працює на виділеному сервері. Web-сервер, побудований на Flask, обробляє ці запити та звертається до сервера бази даних SQLite для збереження або отримання необхідних даних. Ця діаграма наочно демонструє взаємодію між компонентами на фізичному рівні.

Натомість діаграма компонентів показує логічну структуру системи та відображає зв'язки між її елементами. Вона допомагає зрозуміти, як різні компоненти веб-додатку реалізують його функціонал. На рисунку 2.4 представлено приклад такої діаграми, що відображає логічну організацію компонентів і їх взаємодію.

Діаграма компонентів ілюструє ключові складові вебдодатку. Фронтенд реалізований за допомогою HTML, CSS та JavaScript, які взаємодіють із бекендом, розробленим на Flask. Бекенд, у свою чергу, взаємодіє з базою даних, що представлена SQLite. Діаграма на логічному рівні демонструє структуру компонентів додатку, відображаючи їх взаємозв'язки між фронтендом, бекендом та базою даних.

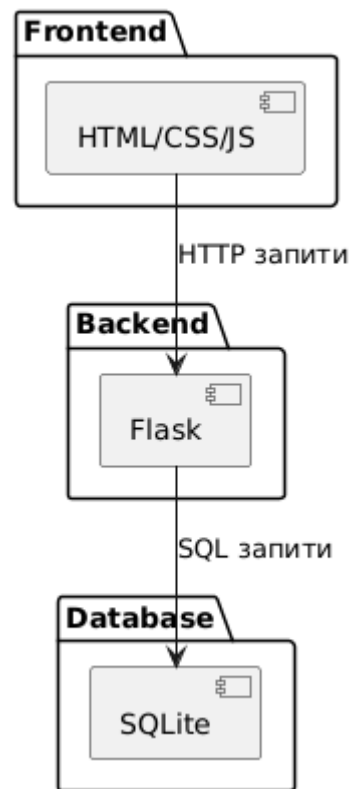


Рисунок 2.4 – Діаграма компонентів

Створення концептуальної моделі бази даних є одним із ключових етапів створення вебресурсів, адже саме на цьому етапі визначається, як система буде зберігати, обробляти та отримувати дані. Для вебресурсу студії «CreativePeople» необхідно розробити структуру бази даних, яка включатиме всі необхідні сутності. У таблиці 2.1 наведено інформацію про структуру таблиці «Users».

Таблиця 2.1 – «Users» (Користувачі)

Поле	Тип даних	Опис
user_id	INT (PK)	Унікальний ідентифікатор користувача
first_name	VARCHAR(255)	Ім'я користувача
last_name	VARCHAR(255)	Прізвище користувача
email	VARCHAR(255)	Електронна пошта користувача (унікальна)
role	VARCHAR(50)	Роль користувача (наприклад, «client», «admin»)
created_at	TIMESTAMP	Дата реєстрації користувача
updated_at	TIMESTAMP	Дата останнього оновлення даних користувача

У таблиці 2.2 відображено дані таблиці «Services».

Таблиця 2.2 – «Services» (Послуги)

Поле	Тип даних	Опис
service_id	INT (PK)	Унікальний ідентифікатор послуги
name	VARCHAR(255)	Назва послуги
description	TEXT	Опис послуги
price	DECIMAL(10,2)	Ціна послуги
created_at	TIMESTAMP	Дата додавання послуги

У таблиці 2.3 відображено дані таблиці «Requests».

Таблиця 2.3 – «Requests» (Заявки)

Поле	Тип даних	Опис
request_id	INT (PK)	Унікальний ідентифікатор заявки
user_id	INT (FK)	Посилання на таблицю Users (Foreign Key)
service_id	INT (FK)	Посилання на таблицю Services (Foreign Key)
contact_email	VARCHAR(255)	Електронна пошта для зворотного зв'язку
message	TEXT	Повідомлення або опис вимог
status	VARCHAR(50)	Статус заявки (наприклад, «обробляється», «завершено», «відхилено»)
created_at	TIMESTAMP	Дата подачі заявки
updated_at	TIMESTAMP	Дата останнього оновлення статусу заявки

У таблиці 2.4 відображено дані таблиці «Feedback».

Таблиця 2.4 – «Feedback» (Відгуки)

Поле	Тип даних	Опис
feedback_id	INT (PK)	Унікальний ідентифікатор відгуку
service_id	INT (FK)	Посилання на таблицю Services (Foreign Key)

Продовження таблиці 2.4

Поле	Тип даних	Опис
user_id	INT (FK)	Посилання на таблицю Users (Foreign Key)
rating	INT	Оцінка послуги (наприклад, від 1 до 5)
comment	TEXT	Текстовий відгук
created_at	TIMESTAMP	Дата створення відгуку

Пояснення:

- таблиця «Users» («Користувачі») таблиця містить основну інформацію про користувачів, таких як ім'я, прізвище, електронна пошта та роль;
- таблиця «Services» («Послуги») зберігає інформацію про послуги, які пропонує студія, включаючи назву, опис і ціну;
- таблиця «Requests» («Заявки») зберігає заявки клієнтів на послуги, де вказується, яку послугу замовлено, контактні дані клієнта та статус заявки;
- таблиця «Feedback» («Відгуки») дозволяє клієнтам залишати відгуки про послугу або проєкт, оцінюючи їх та надаючи коментарі.

Зв'язки між таблицями:

- «Users» → «Requests»: користувачі можуть подавати декілька заявок;
- «Services» → «Requests»: всі послуги можуть бути замовлена різними користувачами;
- «Users» → «Feedback»: користувач може залишити декілька відгуків;
- «Services» → «Feedback»: всі послуги можуть мати декілька відгуків.

На рисунку 2.5 відображено ER-модель у вигляді діаграми класів.

Пояснення ER-моделі:

- таблиця «User» має зв'язок один до багатьох з таблицею «Requests» (кожен користувач може подати кілька заявок);
- таблиця «Service» має зв'язок один до багатьох з таблицею «Requests» (кожна послуга може бути замовлена кількома користувачами);

– таблиця «Service» має зв'язок один до багатьох з таблицею «Feedback» (кожна послуга може мати кілька відгуків);

таблиця «User» має зв'язок один до багатьох з таблицею «Feedback» (кожен користувач може залишити кілька відгуків).

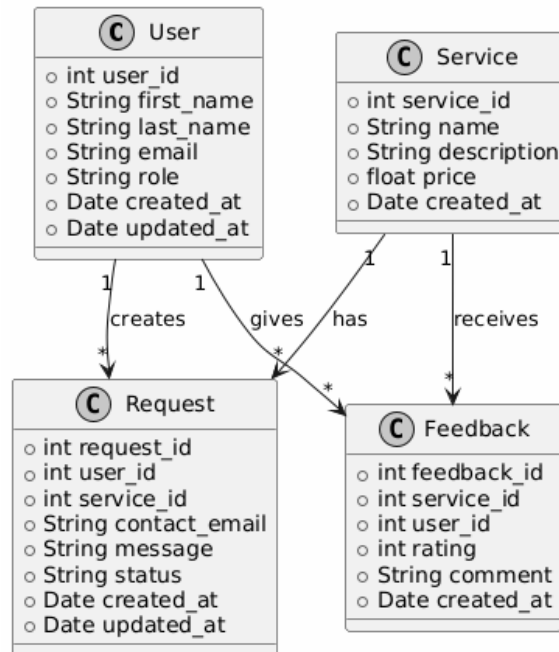


Рисунок 2.5 – ER-модель у вигляді діаграми класів

3 ПРОЄКТУВАННЯ СИСТЕМИ

Сучасний веброзробницький процес охоплює безліч аспектів, починаючи зі створення інтерактивних користувацьких інтерфейсів (frontend) та закінчуючи забезпеченням стабільної та надійної роботи серверної частини (backend). Зі зростанням вимог до швидкодії, зручності й функціональності вебдодатків виникає потреба у впровадженні новітніх інструментів і підходів для обох напрямків. У цьому матеріалі буде розглянуто актуальні технології, що знаходять застосування у веброзробці, із особливим акцентом на сучасні фреймворки, мови програмування та методології. Розробка frontend охоплює технічні рішення та

інструменти, орієнтовані на створення видимої частини вебдодатка, з якою безпосередньо взаємодіють користувачі. Сьогодні в основі frontend-розробки залишаються три головні технології: HTML, CSS і JavaScript. Вони є базою для побудови будь-якого вебресурсу або додатка. Однак із розвитком галузі з'явилася значна кількість нових фреймворків та інструментів, які полегшують процес створення більш складних, інтерактивних і високошвидкісних вебсервісів. З розвитком функціональних вимог розробникам усе частіше доводиться звертатися до використання фреймворків і бібліотек. Вони дозволяють прискорити робочий процес і ефективніше створювати складні інтерактивні додатки. Серед найпопулярніших рішень у цій сфері особливу увагу привертає React - бібліотека JavaScript для розробки користувацьких інтерфейсів [5]. Одним із ключових її принципів є побудова додатків за допомогою компонентів, які можна легко використовувати повторно для створення складних інтерфейсів. React відомий своїми можливостями для розробки односторінкових додатків (SPA), які забезпечують високу швидкість роботи та зручність для користувача. Інший популярний інструмент - Vue.js, який також ґрунтується на концепції компонентів [8]. Легкість у використанні та простота інтеграції роблять його чудовим рішенням для вдосконалення існуючих додатків. Ще одним вагомим представником сучасних фреймворків є Angular, який вважається одним із найбільш потужних і всеохопних інструментів для створення складних вебдодатків [7]. Його особливістю є двостороння прив'язка даних, що дозволяє автоматично синхронізувати зміну даних між сервером і інтерфейсом користувача. У сучасному світі мобільні додатки стали невід'ємною частиною цифрових продуктів. Для їх створення дедалі активніше застосовуються фреймворки для кросплатформенної розробки, такі як React Native і Flutter. Ці інструменти дозволяють розробляти мобільні додатки для операційних систем iOS і Android з використанням універсальної кодової бази. Це не лише спрощує сам процес розробки, але й значно скорочує витрати на обслуговування додатків. Окремо варто згадати TypeScript - надбудову над JavaScript, яка додає можливості статичної типізації [13]. Це помітно знижує ризик виникнення помилок у коді на

етапі його виконання завдяки виявленню проблем ще під час компіляції. Сьогодні TypeScript активно використовується у провідних фреймворках, зокрема в React та Angular, та є важливою тенденцією у світі frontend-розробки. На рисунку 3.1 показано основні принципи взаємодії між frontend і backend частинами сучасного вебдодатка.

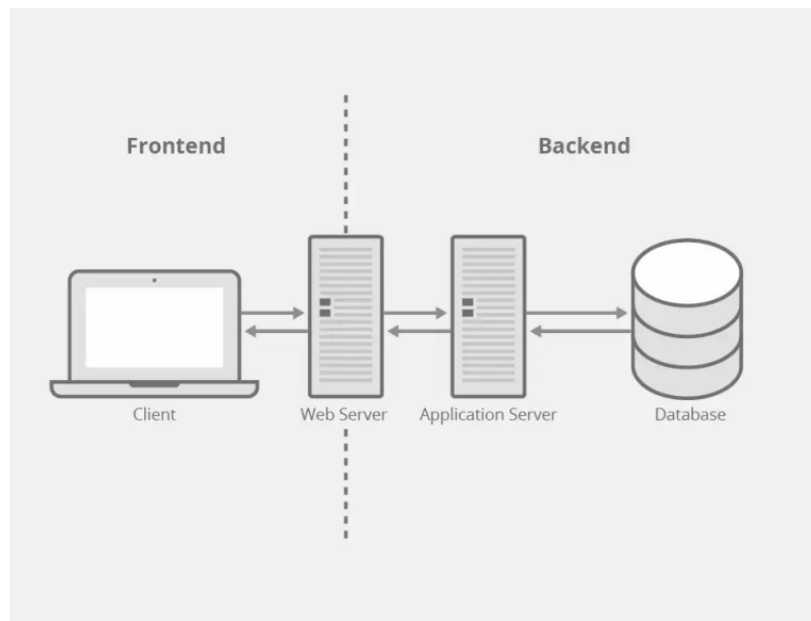


Рисунок 3.1 – Backend та frontend розробка

Розробка бекенду охоплює серверну складову вебдодатків і включає управління базами даних, обробку клієнтських запитів, реалізацію бізнес-логіки та інтеграцію з іншими сервісами. Завдяки сучасним технологіям ця сфера стає дедалі гнучкішою та ефективнішою, що сприяє створенню й підтримці складних і масштабованих вебсистем. Одним із ключових напрямів розвитку в бекенд-розробці є впровадження мікросервісної архітектури. Використання мікросервісів дозволяє розділити додаток на автономні модулі, кожен з яких відповідає за виконання конкретної функції [14]. Для розгортання таких рішень часто застосовуються контейнери, як-от Docker, а для управління та масштабування - системи оркестрації на зразок Kubernetes, які спрощують автоматизацію процесів. У таблиці 3.1 наведено порівняння новітніх технологій Frontend і Backend.

Таблиця 3.1 – Порівняння нових технологій Frontend та Backend

Технологія	Тип	Опис	Основні переваги
React.js	Frontend	Бібліотека для створення інтерактивних інтерфейсів	Висока продуктивність, гнучкість, велика екосистема
Vue.js 3.0	Frontend	Фреймворк для створення SPA та інтерфейсів з реактивними компонентами	Простота, масштабованість, підтримка TypeScript
Svelte	Frontend	Новий підхід до розробки UI, який компілюється в ефективний код	Продуктивність, легкість у використанні, менше навантаження на клієнт
Tailwind CSS	Frontend	Утилітарний CSS-фреймворк для швидкої розробки стильних інтерфейсів	Зручність у використанні, мінімальний CSS-код
TypeScript	Frontend/Backend	Статично типізований суперклас JavaScript для покращення масштабованості та безпеки	Статична типізація, зменшення помилок у коді
Node.js	Backend	Платформа для виконання JavaScript на сервері	Висока продуктивність, асинхронність, велика екосистема
Deno	Backend	Новий серверний фреймворк з підтримкою TypeScript та вбудованою безпекою	Безпека, вбудована підтримка TypeScript
Go (Golang)	Backend	Мова програмування для створення високопродуктивних серверів і мікросервісів	Простота, висока швидкість, низьке споживання ресурсів
GraphQL	Backend	Мова запитів для API, яка дозволяє отримувати тільки необхідні дані	Ефективність, зменшення кількості запитів, гнучкість
Microservices	Backend	Архітектура, при якій додаток складається з незалежних сервісів	Масштабованість, гнучкість, незалежність сервісів

У сучасній веброботі впровадження новітніх технологій як на стороні frontend, так і backend дає змогу створювати ефективніші, швидші та більш масштабовані системи. Інструменти на кшталт React.js, Vue.js, Node.js, Go, разом із сучасними підходами, такими як мікросервісна архітектура та GraphQL, сприяють досягненню високого рівня продуктивності й спрощують

обслуговування додатків [14, 15]. Вибір оптимальних технологій визначається вимогами проєкту, особливостями завдань і масштабами розроблюваної системи.

Створення вебсервісів є складним і багатогранним процесом, який потребує ретельного вибору архітектури для забезпечення стабільної роботи, масштабованості, безпеки та високої продуктивності. Вебсервіси є невід'ємною частиною сучасних додатків, які дозволяють різним компонентам ефективно взаємодіяти через мережу, зокрема Інтернет. Вибір архітектурного підходу при проєктуванні веб-сервісів визначає, як відбуватиметься комунікація між клієнтом і сервером, які методи будуть використовуватися для обміну даними, а також яким чином буде забезпечуватись масштабованість і підтримка додатка на різних етапах його життєвого циклу. Існує чимало архітектурних стилів, які застосовуються у створенні вебсервісів.

Клієнт-серверний підхід є класичною моделлю, у якій взаємодія здійснюється шляхом стандартного обміну запитами та відповідями між клієнтом і сервером [16]. У цьому процесі клієнт направляє запит, сервер обробляє отриману інформацію і повертає відповідь. Такий підхід вважається одним із найпростіших і найпоширеніших у розробці веб-сервісів, адже забезпечує основні можливості взаємодії між складовими компонентами додатка. Схему клієнт-серверної моделі можна побачити на рисунку 3.2.

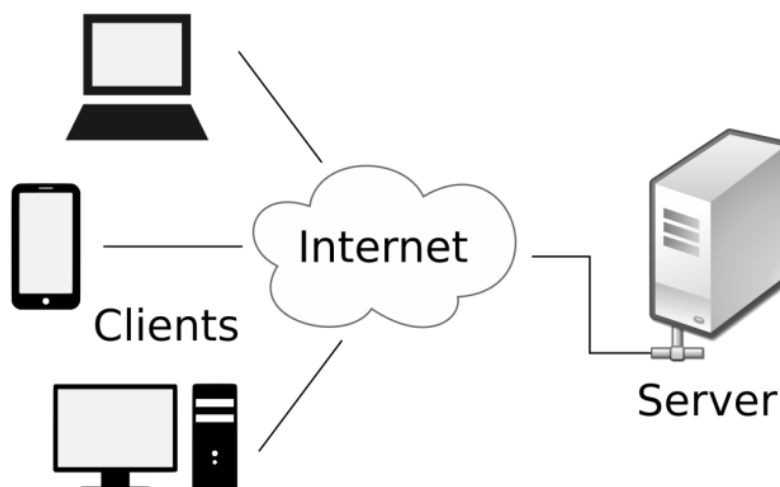


Рисунок 3.2 – Клієнт-сервер

REST (Representational State Transfer) - це архітектурний стиль, який базується на стандартних HTTP-протоколах для взаємодії клієнтської та серверної частини. Він є одним із найпоширеніших підходів до розробки веб-сервісів завдяки своїй простоті, легкості у використанні та зручності для масштабування систем [7]. RESTful веб-сервіси застосовують HTTP-методи, такі як GET, POST, PUT, DELETE, що дозволяє ефективно працювати з ресурсами через уніфіковані URL-адреси. Архітектурна схема REST представлена на рисунку 3.3.

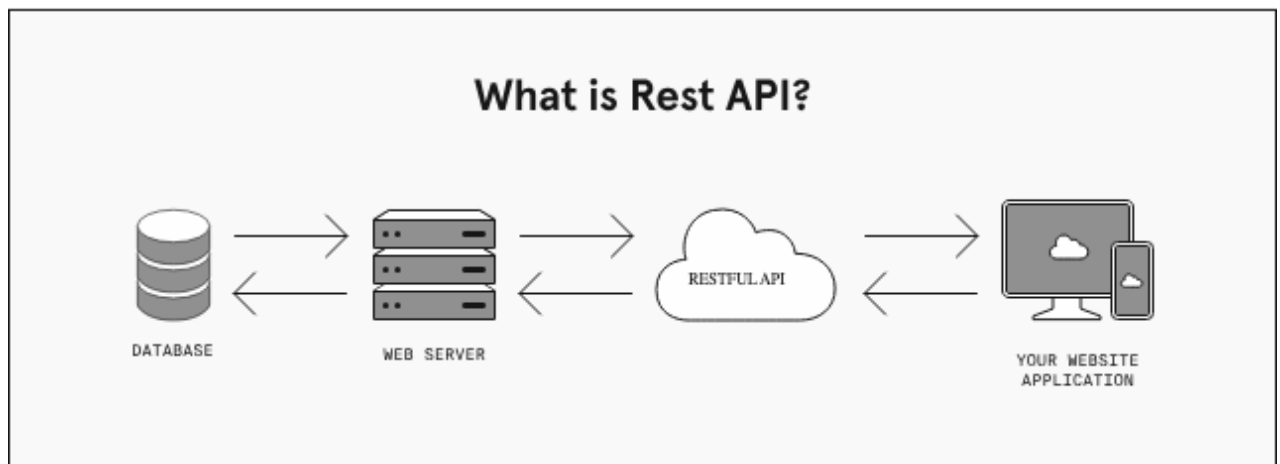


Рисунок 3.3 – REST

Мікросервіси - це архітектурний підхід, за якого система ділиться на велику кількість невеликих, незалежних, функціонально ізольованих сервісів, які взаємодіють між собою за допомогою стандартних протоколів, таких як HTTP, або через системи обміну повідомленнями, наприклад, Kafka чи RabbitMQ [14]. Кожен із сервісів зосереджений на виконанні конкретного завдання, що полегшує підтримку та масштабування додатків, одночасно забезпечуючи гнучкість у виборі технологій для реалізації кожного з них. Архітектура мікросервісів ілюструється на рис. 3.4.

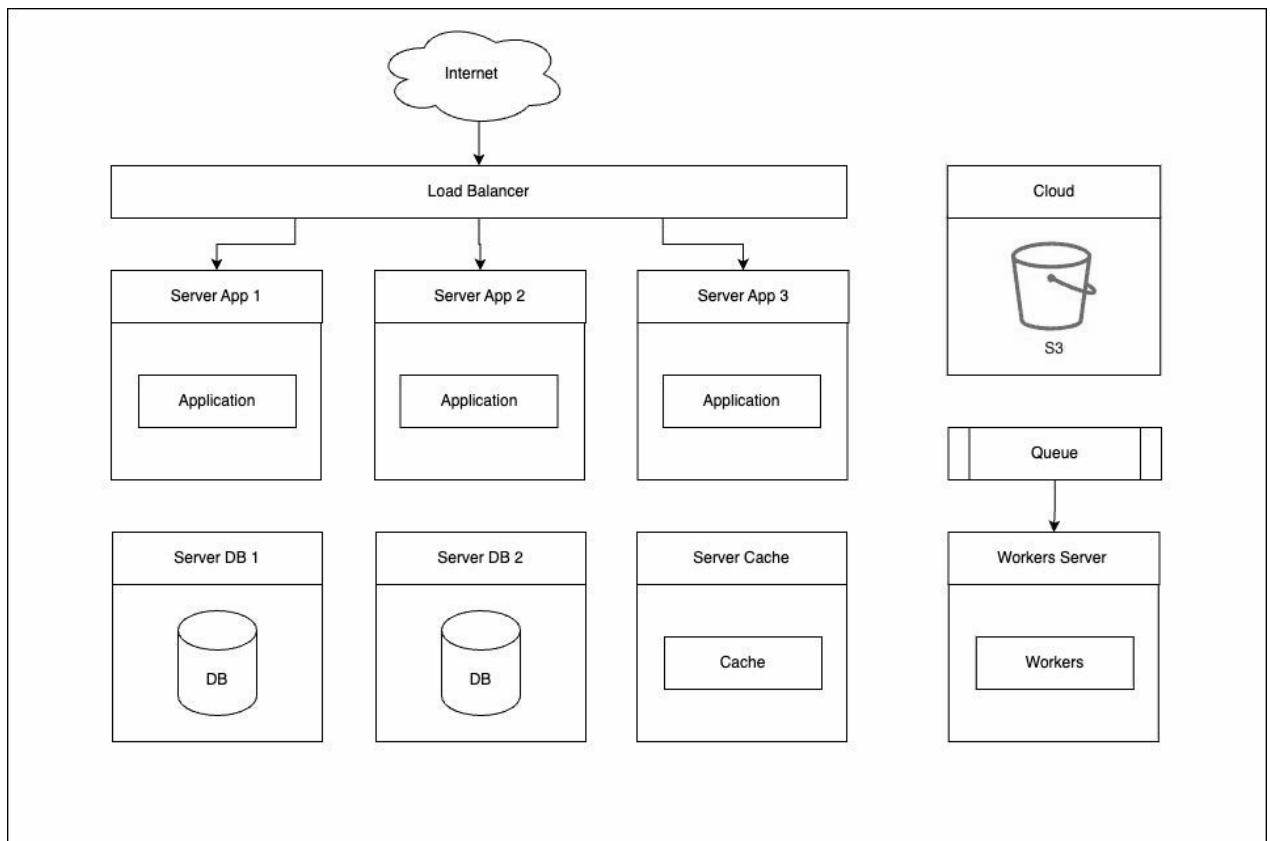


Рисунок 3.4 – Мікросервіси

SOA (Service-Oriented Architecture) – це сервісно-орієнтована архітектура, у якій додаток будується з декількох самостійних, але взаємопов'язаних сервісів. Ці сервіси взаємодіють між собою шляхом обміну даними через стандартизовані інтерфейси, наприклад, такі як SOAP (Simple Object Access Protocol) або REST [17]. Основною метою SOA є інтеграція, що робить її особливо корисною для розробки комплексних корпоративних систем, де важливо забезпечити зручність та сумісність між різними програмними компонентами. На рисунку 3.5 зображено архітектуру SOA.

Масштабованість означає здатність системи адаптуватися до збільшення навантаження. Вебсервіси повинні створюватися таким чином, щоб їх легко можна було масштабувати як вертикально, шляхом збільшення ресурсів одного сервера, так і горизонтально, додаючи нові сервери для обробки додаткових запитів. Одним із ефективних підходів є мікросервісна архітектура, яка дозволяє масштабувати кожен сервіс окремо, без впливу на інші компоненти системи. Для забезпечення постійної доступності вебсервісів необхідно організувати

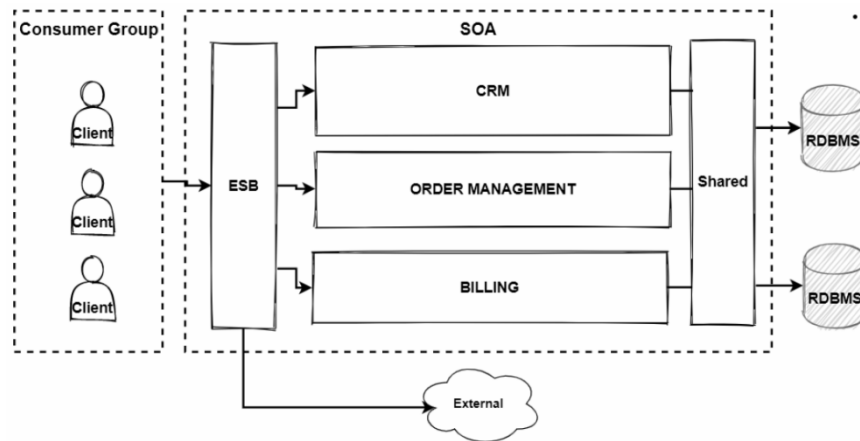


Рисунок 3.5 – SOA

балансування навантаження між кількома серверами, дублювання ключових компонентів та застосовувати системи автоматичного відновлення у випадку збоїв. Архітектура повинна бути достатньо стійкою, щоб навіть при виході з ладу однієї частини системи інші її компоненти продовжували функціонувати без помітних перебоїв. Значну увагу слід приділяти забезпеченню працездатності вебсервісів у ситуаціях відмови окремих компонентів. Для цього застосовуються такі методи, як дублювання даних, використання черг повідомлень для асинхронної обробки запитів, а також стратегія "circuit breaker", яка допомагає уникнути загального збою системи через падіння окремого модуля. Надійний захист від несанкціонованого доступу також має бути пріоритетним завданням для вебсервісів. Це досягається завдяки використанню процедур аутентифікації та авторизації на основі токенів (наприклад, JWT), шифруванню даних за допомогою протоколів SSL/TLS, обмеженню доступу відповідно до IP-адрес або ролей користувачів, а також дотриманню принципу найменших прав доступу для кожного окремого користувача або процесу системи.

Сучасні вебсервіси здебільшого обмінюються даними через API (інтерфейс прикладного програмування). API визначає набір методів і протоколів для комунікації між різними системами чи їх компонентами. Виділяють кілька основних типів API:

- RESTful API;

- SOAP;
- WebSockets.

RESTful API - інтерфейси, що базуються на використанні методів HTTP для обробки запитів і відправлення відповідей.

SOAP (Simple Object Access Protocol) - протокол для передачі структурованих даних. Хоча раніше SOAP був популярним, зараз його все частіше замінюють REST через більшу гнучкість останнього.

WebSockets - технологія, яка забезпечує двосторонню взаємодію між клієнтом і сервером, що є особливо корисним для реалізації функцій у режимі реального часу, таких як чати чи онлайн-ігри.

Одним із ключових аспектів архітектури вебсервісів є асинхронна обробка запитів. Для цього активно застосовуються черги повідомлень, такі як RabbitMQ або Kafka, які забезпечують можливість тимчасового зберігання запитів чи задач до моменту, поки сервер буде готовий до їхньої обробки. Такий підхід сприяє ефективному управлінню навантаженням і підвищує стійкість системи під час високої інтенсивності роботи серверів. Вебсервіси часто розгортаються у контейнерах, що дозволяє ізолювати їхнє середовище виконання. Технології на кшталт Docker дозволяють легко упакувати додатки разом із усіма необхідними залежностями. Для керування та масштабування контейнерів використовується Kubernetes - система оркестрації, яка автоматизує процеси розгортання, масштабування та управління контейнеризованими додатками, забезпечуючи їхню стабільну та зручну експлуатацію.

Сучасні вебсервіси мають відповідати високим стандартам масштабованості, доступності, надійності та безпеки. Вибір архітектури залежить від особливостей проекту, вимог до продуктивності та очікуваної кількості користувачів. Мікросервісний підхід, впровадження RESTful API, асинхронна обробка запитів і використання технологій контейнеризації є одними з ключових напрямів, які активно впливають на створення ефективних веб-рішень у сучасній розробці.

Вебдодатки, як і будь-яке програмне забезпечення, стикаються з

численними викликами, пов'язаними з безпекою та оптимізацією. Захист даних є надзвичайно важливим, адже вебдодатки зазвичай оперують чутливою інформацією користувачів і нерідко стають мішенню для зловмисників, які прагнуть отримати несанкціонований доступ або використати дані у своїх цілях. Оптимізація, зі свого боку, спрямована на забезпечення максимально ефективної роботи додатка, зниження навантаження на сервери та покращення користувацького досвіду. Ці два аспекти мають вирішальний вплив на успішність, масштабованість і рівень довіри до вебдодатків [18, 19]. У цьому контексті забезпечення захисту й оптимізація є невід'ємною складовою створення надійних і функціональних вебрішень. Безпека вебдодатків охоплює широкий спектр технологій та методів, спрямованих на запобігання атакам, а також на захист ресурсів і конфіденційних даних від несанкціонованого доступу. З розвитком інформаційних технологій виникають нові види атак та загроз, які можуть поставити під сумнів ефективність традиційних захисних підходів. Тому особливого значення набуває постійна адаптація стратегій захисту відповідно до сучасних викликів.

Однією з ключових загроз для вебдодатків є SQL-ін'єкції [20]. Ці атаки виникають, коли зловмисники впроваджують шкідливий SQL-код у запити до бази даних, що може призвести до викрадення, модифікації або видалення даних. Для захисту від таких загроз необхідно застосовувати параметризовані запити або використовувати ORM-бібліотеки (Object-Relational Mapping), які автоматично екранізують небезпечні символи у запитах. Ще однією серйозною проблемою є XSS-атаки (Cross-Site Scripting), які дозволяють зловмисникам вставляти небезпечний JavaScript-код у вебсторінки. Цей код потім виконується у браузері інших користувачів, що може спричинити викрадення сесійних даних, доступ до конфіденційної інформації або навіть маніпуляції з діями користувачів. Для захисту від XSS-атак важливо здійснювати санітарну обробку даних (escaping), яка полягає у видаленні або заміні небезпечних елементів у введених даних. Окрему увагу варто приділити загрозі міжсайтових запитів (CSRF - Cross-Site Request Forgery). Такий тип атак ґрунтується на тому, щоб змусити користувача

виконати небажану дію у веб-додатку (наприклад, змінити пароль чи здійснити фінансовий переказ), навіть не підозрюючи, що це спровоковано шкідливим запитом. Для захисту від CSRF використовуються спеціальні токени, які унеможливають підробку запитів. Також ефективними є перевірка реферера та налаштування політики SameSite для файлів cookie.

Одним із найефективніших способів захисту вебдодатків є шифрування даних. Зазвичай для цього використовується протокол HTTPS, який забезпечує безпечне з'єднання між клієнтом і сервером завдяки застосуванню SSL/TLS-сертифікатів. Це гарантує захист переданої інформації від перехоплення та модифікації з боку зловмисників. Водночас важливо правильно налаштувати SSL/TLS, використовуючи сучасні алгоритми шифрування та надійні ключі, а також забезпечити безпечне зберігання сертифікатів на сервері. Окрему увагу слід приділити управлінню сесіями та автентифікації користувачів. Для підвищення рівня безпеки можуть бути застосовані такі методи, як двофакторна автентифікація (2FA), створення складних і надійних паролів, а також хешування паролів за допомогою сучасних алгоритмів, наприклад bcrypt чи Argon2. При роботі з сесіями необхідно використовувати захищені cookie, які неможливо підробити, і проводити регулярний моніторинг сесій для виявлення підозрілих дій. Крім того, важливим етапом є постійне тестування безпеки веб-додатків. Для цього застосовуються методи, такі як тестування на проникнення (penetration testing), а також використання автоматизованих інструментів для виявлення слабких місць, наприклад сканерів для пошуку відомих вразливостей.

Оптимізація вебдодатків є ключовим етапом, адже від швидкості роботи додатка залежать досвід користувачів і його позиція в пошукових системах. Вона охоплює різні аспекти, починаючи зі скорочення часу завантаження сторінок і закінчуючи раціональним використанням серверних ресурсів. Одним із найефективніших підходів є мініфікація та стиснення ресурсів. Це передбачає зменшення розміру файлів JavaScript, CSS та HTML шляхом видалення зайвих пробілів, коментарів та інших непотрібних елементів. Автоматизувати цей процес можна за допомогою таких інструментів, як Webpack, Gulp або Grunt, що

позитивно впливає на продуктивність додатка. Іншим важливим методом є кешування. Воно дозволяє суттєво знизити навантаження на сервер, зменшити кількість запитів і забезпечити швидший доступ до ресурсів, які використовуються найчастіше. Кешування застосовується на різних рівнях: на боці клієнта (браузерне кешування), на сервері (за допомогою інструментів, таких як Redis або Memcached), а також у мережах доставки контенту (CDN). Використання CDN суттєво прискорює завантаження сторінок, оскільки статичні ресурси на кшталт зображень, відео чи JavaScript-файлів доставляються з найближчого сервера до кінцевого користувача [21].

Одним із ключових підходів у сучасній веброзробці є використання техніки відкладеного завантаження (lazy loading), яка передбачає завантаження ресурсів лише тоді, коли вони стають необхідними, замість одночасного завантаження всього контенту. Важливою складовою також є оптимізація запитів до бази даних. Часто проблеми з продуктивністю виникають через надто великі або недостатньо оптимізовані SQL-запити, що спричиняє затримки у відповіді сервера. Для вирішення таких проблем важливо використовувати індекси, грамотно проектувати схеми бази даних і мінімізувати зайві об'єднання таблиць. Ще одним ефективним рішенням є використання NoSQL-баз даних, таких як MongoDB, які краще підходять для збереження неструктурованих або динамічно змінних даних. Для максимально ефективного використання серверних ресурсів можна впроваджувати багатопоточність або асинхронну обробку запитів. Це дозволяє скоротити час очікування користувачів і збільшити кількість запитів, які сервер може обробляти одночасно. Безпека та оптимізація роботи вебдодатків є фундаментальними аспектами успішної розробки. Надійний захист від кібератак, правильне управління сесіями, шифрування конфіденційних даних і регулярне тестування на вразливості є основою безпеки додатків. Разом із цим, оптимізація сприяє зменшенню часу завантаження сторінок, зниженню навантаження на сервер і покращенню взаємодії з користувачами. Поєднання цих двох підходів дозволяє створювати сучасні, ефективні та безпечні вебдодатки, які відповідають вимогам швидкості, функціональності та зручності користування.

4 РОБРОБКА ВЕБРЕСУРСУ

4.1 Розробка frontend-частини

Шапка є першим елементом, який користувач бачить під час відвідування сайту, тому її роль у загальній структурі сторінки дуже важлива. У ній містяться ключові елементи, які сприяють швидкому доступу до необхідної інформації або виконанню користувачем певних дій. Зокрема, у шапці розташований логотип компанії, виконаний у простому та сучасному стилі (рис. 4.1).

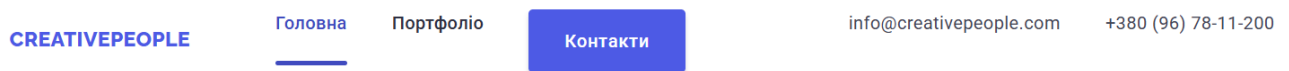


Рисунок 4.1 – Шапка сайту (Header)

Логотип виконано у двох варіантах кольору, що підкреслює стиль і характер бренду. Частина логотипу виділена фіолетовим кольором, який створює привабливий контраст і підсилює візуальну естетику. Таке рішення допомагає чітко закарбувати уявлення про компанію. Поруч із логотипом розташоване навігаційне меню, що включає три основні розділи: «Головна», «Портфоліо» та «Контакти». Ці пункти меню забезпечують зручну навігацію і дозволяють користувачам швидко переходити до потрібного розділу сайту. Кожне посилання структуровано так, щоб процес пошуку необхідної інформації був максимально інтуїтивним і зрозумілим. Важливим елементом верхньої частини сайту є також контактні дані компанії: електронна пошта та телефон. Вони розташовані таким чином, щоб бути легко помітними, що забезпечує швидкий доступ до зв'язку з компанією. Це сприяє більшому довірі до бренду та полегшує комунікацію з відвідувачами сайту.

Головна частина сайту має вирішальне значення, оскільки саме в ній відвідувач знайомиться з ключовими пропозиціями компанії, її послугами та проектами. Цей розділ поділяється на кілька секцій, кожна з яких виконує свою

функцію і сприяє досягненню комунікаційних цілей ресурсу (рис. 4.2).

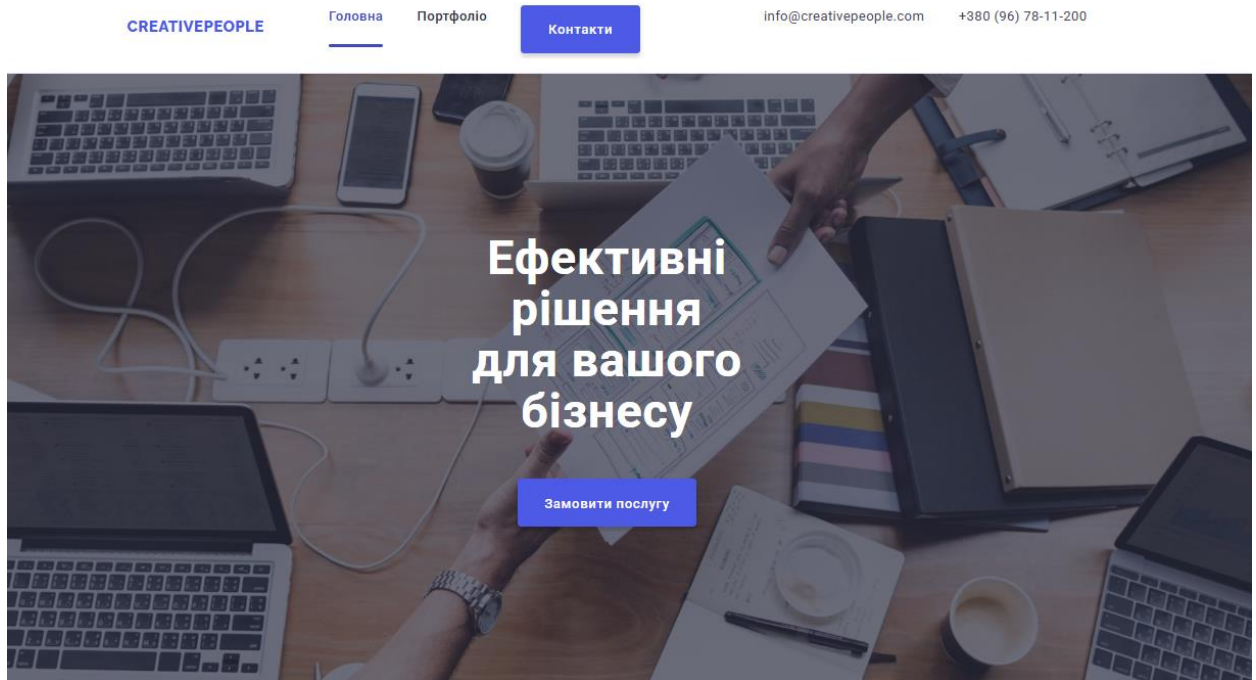


Рисунок 4.2 – Головна частина (Hero Section)

Перша секція виконує роль героїчного блоку та є ключовим елементом будь-якого сучасного вебсайту. Це обширний банер або фонове зображення, що займає значну частину екранного простору. У наведеному прикладі цей блок містить великий заголовок із заявою «Ефективні рішення для вашого бізнесу». Такий сильний заклик до дії миттєво привертає увагу відвідувачів і чітко демонструє, що компанія надає інноваційні бізнесові рішення. Кнопка «Замовити послугу» додає елемент інтерактивності та спонукає користувача зробити перший крок до взаємодії з компанією.

У наступній частині акцентовано ключові переваги студії (рис. 4.3). Для наочності застосовано піктограми та лаконічні описи, що пояснюють, чому ця фірма є оптимальним вибором для потенційних замовників. Піктограми з різними символами - наприклад, антена (стратегії), годинник (пунктуальність) та діаграма (технології) - сприяють більшому розумінню інформації та її візуальній привабливості. Кожна позиція детально характеризує одну із сильних сторін компанії, формуючи довіру до її професійності.



Стратегії

Наша мета — визначити бізнес-проблему, щоб знайти ідеальне та креативне рішення.



Пунктуальність

Досягайте цільової аудиторії за найкращою ціною та в найкоротші терміни.



Старанність

Підтвердуйте бренди з найбільшими можливостями для цифрового зростання.



Технології

Дизайнерська практика, зосереджена на цифровому досвіді.

Рисунок 4.3 – Переваги (Benefits Section)

Розділ із проєктами демонструє портфоліо компанії (рис. 4.4). У ньому представлені зображення з поясненнями щодо реалізованих успішних робіт. Це дає змогу потенційним замовникам оцінити рівень виконання та стиль компанії. Завдяки застосуванню адаптивних зображень вебсайт коректно відображається на різних пристроях, що є суттєвим для сучасних інтернет-ресурсів, орієнтованих на мобільний трафік.

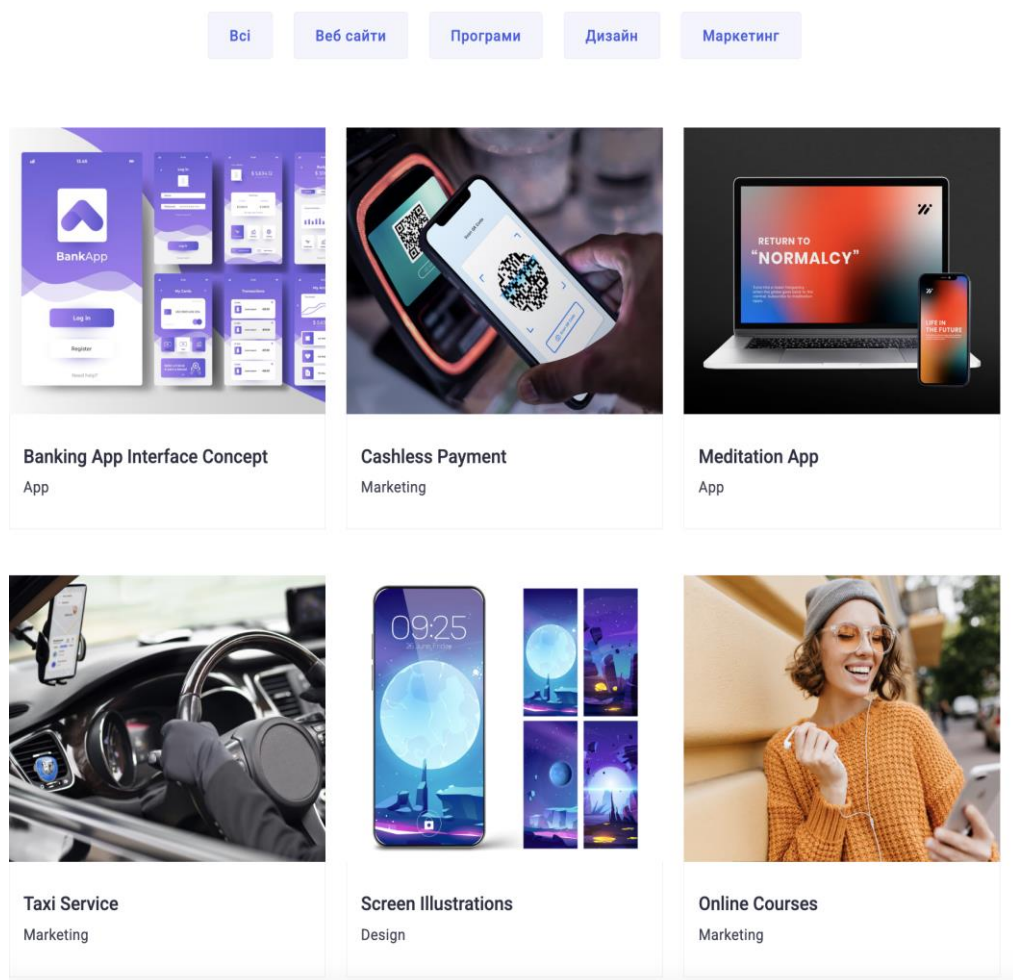


Рисунок 4.4 - Проєкти (Projects Section)

Розділ, присвячений команді, підкреслює значення людського капіталу в компанії (рис. 4.5). Тут розміщено фотографії членів команди, їхні імена та ролі. Кожному співробітнику надано посилання на профілі в соціальних мережах, що створює враження відкритості та доступності організації. Цей елемент додає бренду «людський» вимір і допомагає користувачам усвідомити, що за компанією стоять реальні люди, з якими можна взаємодіяти.

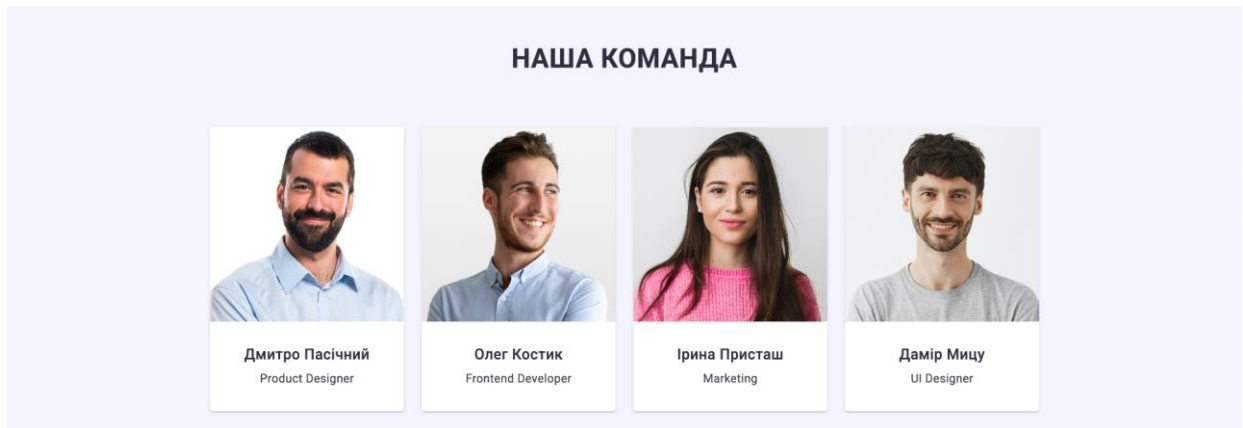


Рисунок 4.5 – Команда (Team Section)

У розділі клієнтів розміщені логотипи або іконки компаній, з якими вже працювали. Це показує, який є досвід співпраці з різними замовниками і підвищує репутацію студії (рис. 4.6). Цей розділ також допомагає потенційним клієнтам побачити, кому вже довіряє компанія.



Рисунок 4.6 – Клієнти (Customers Section)

Футер - це важлива частина сайту. Тут можна знайти додаткову інформацію про компанію, таку як ще один логотип, опис послуг і посилання на соціальні мережі. Це дозволяє користувачам знайти всю потрібну інформацію навіть внизу

сторінки. Іконки соціальних мереж дають можливість підписатися на новини компанії, що важливо для підтримки зв'язку з аудиторією (рис. 4.7).

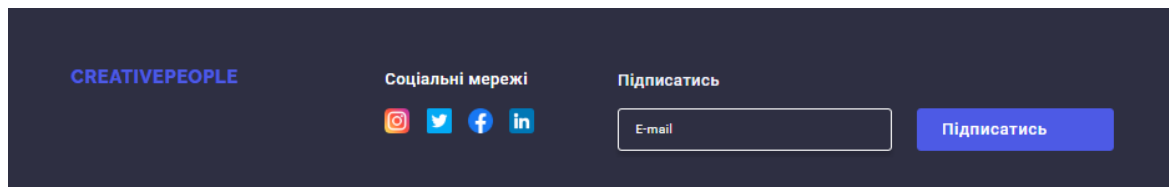


Рисунок 4.7 – Футер (Footer)

Сайт має вікно для зворотного зв'язку (рис. 4.8). Це форма, де користувач може вказати свої контактні дані (ім'я, телефон, електронну пошту), щоб з ним могли зв'язатися пізніше. Цей елемент дуже важливий для покращення спілкування з клієнтами, адже він дозволяє зв'язатися з ними навіть без необхідності телефонувати чи надсилати електронний лист.

Залиште свої контакти, і ми вам передзвонимо

Ім'я

Телефон

Email

Коментар

Text input

☐ I accept the terms and conditions of the [Privacy Policy](#)

Відправити

Рисунок 4.8 – Модальне вікно (Modal Window)

4.2 Реалізація backend-частини

Модальні вікна (або pop-up вікна) використовуються для відображення додаткової інформації або для взаємодії з користувачем без необхідності оновлення сторінки. На рисунку 4.9 подано файл modal.js, який демонструє обробку модальних вінок.

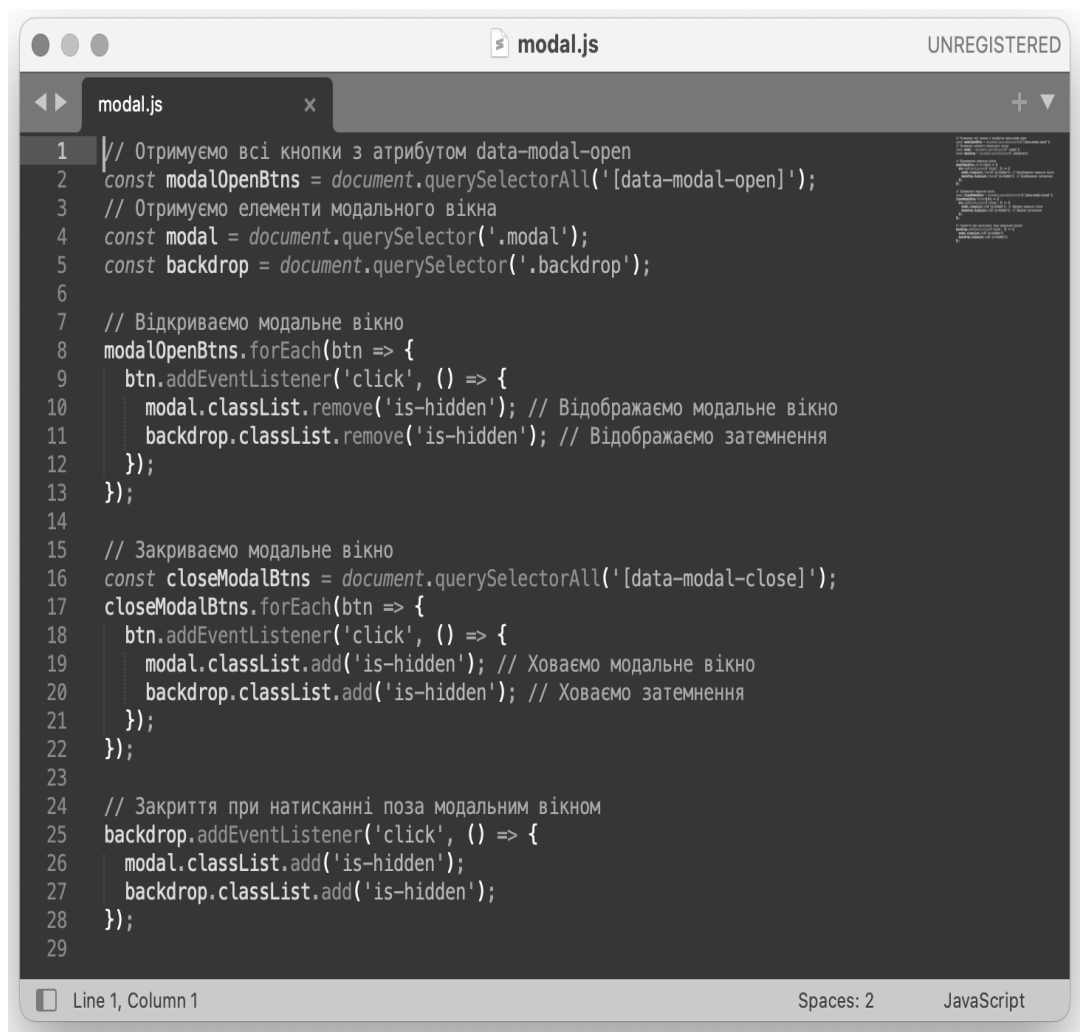


Рисунок 4.9 – Обробка модальних вінок

Даний програмний код відповідає за функціональність відкриття та закриття модальних вікон вебресурсу. У першому рядку створюється об'єкт refs, який містить посилання на важливі елементи сторінки:

- openModalBtn (кнопка для відкривання модального вікна);

- closeModalBtn (кнопка для закривання модального вікна);
- modal (модальне вікно, яке буде відкриватися і закриватися).

Для кнопки, яка відкриває модальне вікно (openModalBtn), і кнопки, що закриває модальне вікно (closeModalBtn), додаються обробники подій на натискання. Ці обробники викликають функцію toggleModal(). Функція toggleModal() додає або прибирає клас is-hidden з модального вікна. Це дає можливість показувати або ховати модальне вікно. Клас is-hidden (який зазвичай має стиль display: none) використовується для управління видимістю елемента. Цей скрипт дозволяє створити інтерфейс з модальними вікнами, які можна відкривати та закривати. Це може бути корисно для форм зворотного зв'язку, повідомлень або інструкцій для користувачів, не змінюючи вміст сторінки.

На рисунку 4.10 подано файл mobile-menu.js, який демонструє програмний код мобільного меню.

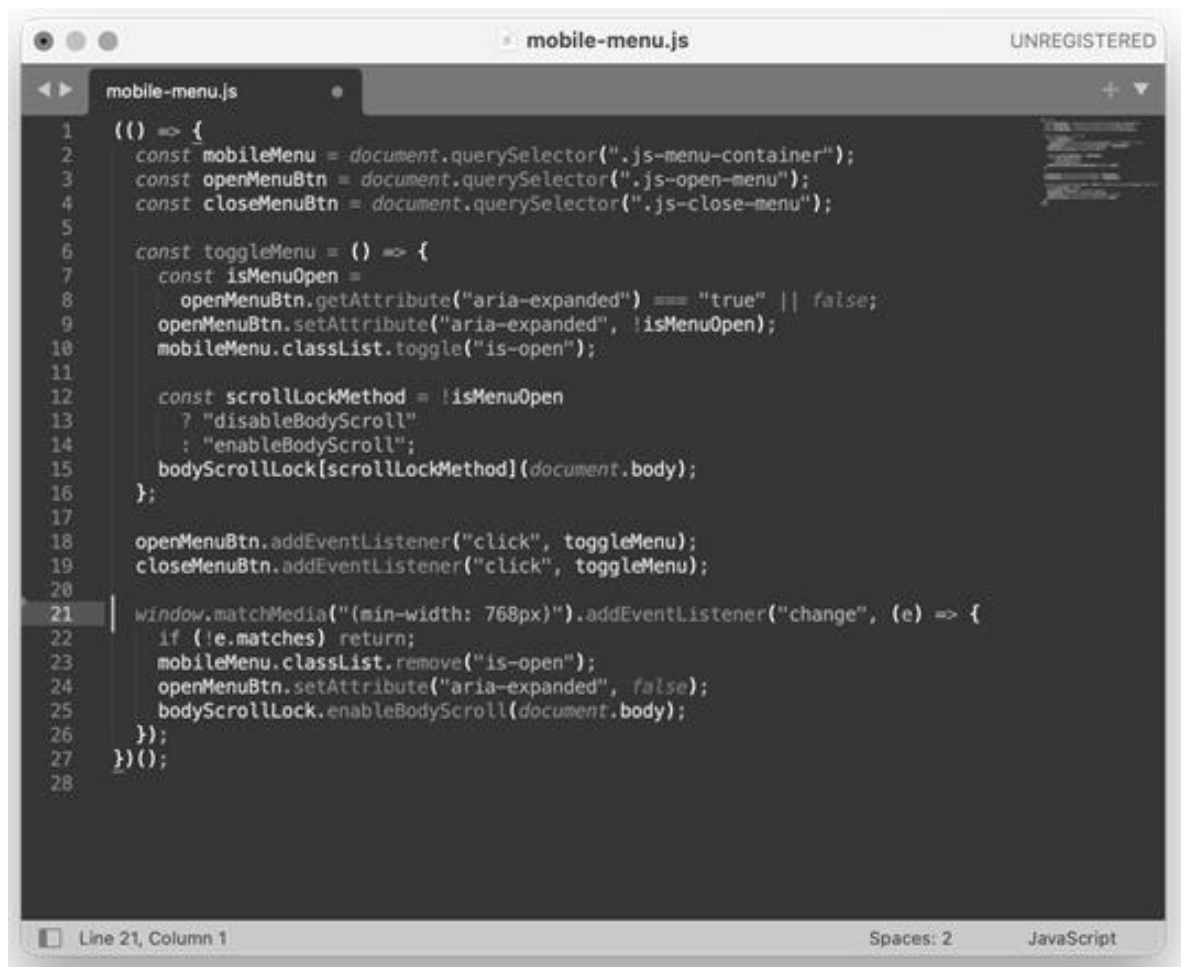


Рисунок 4.10 – Мобільне меню

Даний файл відповідає за функціональність мобільного меню, яке відкривається і закривається при натисканні на кнопку. Меню адаптується під розміри екрану мобільних пристроїв, що дозволяє зручно переглядати ведресурс на мобільних телефонах, смартфонах і планшетах.

Змінні і посилання:

- `mobileMenu` (елемент, який містить саме мобільне меню);
- `openMenuBtn` (кнопка для відкриття мобільного меню);
- `closeMenuBtn` (кнопка для закриття мобільного меню).

Функція `toggleMenu()` змінює стан меню (відкрите або закрите). Спочатку вона перевіряє значення атрибута `aria-expanded`, що показує, чи відкрите меню. Якщо меню відкрите (`aria-expanded="true"`), функція закриває його, змінюючи значення атрибута на `false` і видаляючи клас `is-open` з контейнера меню. Якщо меню закрито, то навпаки — меню відкривається. Також є вбудована бібліотека `body-scroll-lock`, яка блокує прокручування сторінки, коли меню відкрите. Це важливо для мобільних пристроїв, щоб уникнути прокручування сторінки під час відкриття меню. Для користувачів з великими екранами (ширше 768px), якщо змінюється орієнтація екрану (наприклад, з вертикальної на горизонтальну), меню автоматично закривається. Це допомагає уникнути проблем з мобільним меню на великих екранах. Цей скрипт дозволяє користувачам відкривати та закривати мобільне меню, забезпечуючи зручний доступ до навігації. Він також підтримує правильну роботу на мобільних пристроях, блокуючи прокручування сторінки при відкритому меню.

На рисунку 4.11 подано файл `card-filtering.js`, який демонструє програмний код фільтрації карток проєктів.

Цей файл відповідає за фільтрацію карток проєктів або робіт на сайті. Він допомагає користувачам відфільтровувати контент за категоріями, такими як "всі", "веб-дизайн", "маркетинг" тощо. Це дозволяє швидко знаходити цікаві проєкти. Обробка події натискання кнопки фільтра: після завантаження сторінки (це потрібно для правильної роботи DOM) до кожної кнопки фільтра додається слухач події. Кнопки фільтрів мають класи, які показують категорії (наприклад,

"all", "web-design", "marketing"). Коли натискають одну з цих кнопок, запускається функція, яка перевіряє, який клас кнопки натиснули. Для кожної картки (яка має клас `project-card-item`) скрипт перевіряє, чи відповідає вона обраному фільтру. Якщо картка відповідає фільтру (має відповідний клас), вона відображається на сторінці (`style.display = 'block'`). Якщо картка не підходить під вибраний фільтр, вона приховується (`style.display = 'none'`). Якщо користувач натисне кнопку з фільтром "all", всі картки на сторінці стануть видимими, і жодна картка не буде прихована. Цей скрипт робить сайт інтерактивним, дозволяючи користувачам фільтрувати проекти за категоріями, що значно спрощує пошук потрібного контенту. Це забезпечує зручний і швидкий доступ до інформації, особливо для великих портфоліо чи списків проектів.

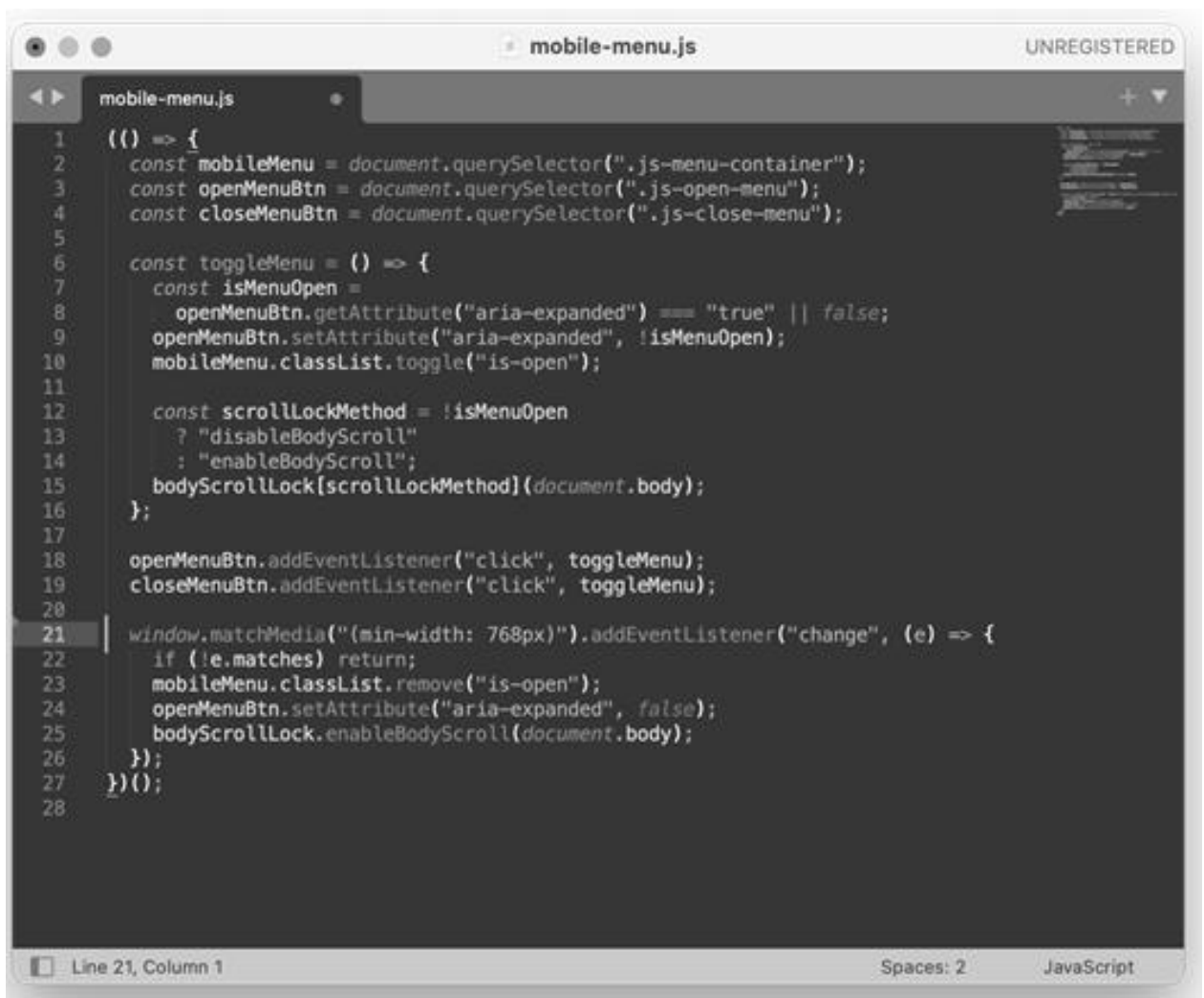


Рисунок 4.11 – Фільтрація карток проектів

Усі три скрипти надають важливі функції на сайті, що значно покращують користувацький досвід:

- `modal.js` дає можливість створювати модальні вікна для взаємодії з користувачем;
- `mobile-menu.js` адаптує меню для мобільних пристроїв, забезпечуючи зручність використання вебресурсу;
- `card-filtering.js` додає можливість фільтрації контенту, що значно покращує навігацію по вебсторінкам.

4.3 Інтеграція frontend та backend

На фронтенді JavaScript відповідає за отримання значень з форми для контактів клієнтів (наприклад, ім'я, телефон, email, коментар) за допомогою стандартних методів доступу до елементів DOM за допомогою `document.getElementById`, `document.querySelector` тощо (рис. 4.12).

```
const name = document.getElementById("name").value;  
const phone = document.getElementById("phone").value;  
const email = document.getElementById("email").value;  
const comment = document.getElementById("comment").value;
```

Рисунок 4.12 – Методи доступу до елементів DOM

JavaScript реагує на події, що виконуються на сторінці, наприклад, натискання кнопки «Відправити». Це досягається за допомогою додавання обробників подій `addEventListener` (рис. 4.13).

```
document.getElementById("contact-form").addEventListener("submit", function(event)  
    event.preventDefault(); // Запобігаємо стандартному відправленню форми  
    // Збір і відправка даних  
});
```

Рисунок 4.13 – Обробників подій `addEventListener`

Після обробки запиту сервером повертає відповідь (як правило, у форматі JSON). JavaScript на клієнтській стороні обробляє відповідь і виконує відповідні дії (наприклад, показує повідомлення про успіх або помилку (рис. 4.14).

```
.then(response => response.json())
.then(data => {
  if (data.success) {
    // Вивести повідомлення про успіх
    alert("Ваші дані успішно надіслані!");
  } else {
    // Вивести повідомлення про помилку
    alert("Помилка при відправці даних.");
  }
})
```

Рисунок 4.14 – Програмний код повідомлення про успіх або помилку

JavaScript взаємодіє з клієнтською частиною наступним чином:

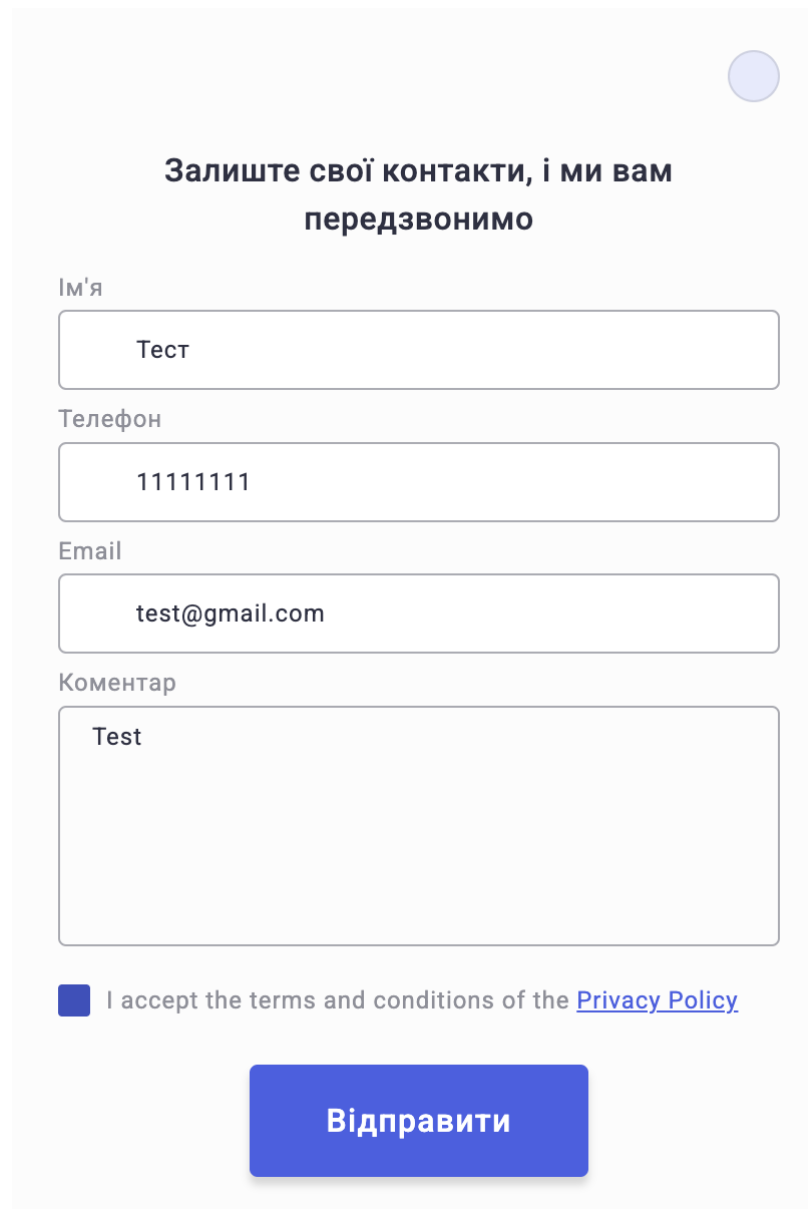
- збирає дані з форми;
- відправляє дані на сервер за допомогою запитів;
- обробляє відповідь від сервера (дані успішно надіслані або повідомляє про помилку).

4.4 Тестування функціональності сервісу

При випробуванні функціональності вебсервісу проведемо декілька тестів.

Перевіримо, чи правильно працює модальне вікно для відправки запиту, чи персональні дані правильно передаються на сервер та чи обробляються правильно відповіді від сервера. Для цього виконаємо наступні кроки:

- відкриття модального вікна форми;
- заповнення форми відповідними даними;
- відправка персональних даних;
- обробка відповіді сервера;
- закриття вікна форми після відправки.



Залиште свої контакти, і ми вам передзвонимо

Ім'я

Тест

Телефон

11111111

Email

test@gmail.com

Коментар

Test

☐ I accept the terms and conditions of the [Privacy Policy](#)

Відправити

Рисунок 4.15 – Введення даних у форму

На рисунку 4.15 та 4.16 подано введення даних у форму внесення персональних даних для зворотнього зв'язку. Форма перевіряє правильність введення даних (рис. 4.16). У коді валідація даних здійснюється через HTML5 атрибути (приклад програмного коду у лістингі 4.1):

- `required` (використовується для обов'язкових полів, які потрібно заповнити перед відправкою форми (наприклад, `name`, `phone`, `email`));
- `type="email"` (перевіряє, чи введений `email` має правильний формат (наприклад, `example@domain.com`));
- `pattern` (використовується для перевірки формату введеного значення

через регулярні вирази).

The image shows a contact form titled "Залиште свої контакти, і ми вам передзвонимо". It contains three input fields: "Ім'я" (Name) with the value "Тест", "Телефон" (Phone) with the value "322424", and "Email" with the value "werwe@". The "Email" field has a red dotted underline. Below the form, a white error message box with an orange exclamation mark icon states: "Введіть частину адреси після символу '@'. Адрес 'werwe@' неполный." (Enter the part of the address after the '@' symbol. The address 'werwe@' is incomplete.). Below the error message is a checkbox labeled "I accept the terms and conditions of the [Privacy Policy](#)". At the bottom is a blue button labeled "Відправити" (Send).

Рисунок 4.16 – Введення невірних даних у форму

Лістинг 4.1 – Програмний код валідації внесених даних

```
<input type="text" id="name" name="name" required />
<input type="email" id="email" name="email" required />
<input type="tel" id="phone" name="phone"
pattern="^\+?\d{1,4}?\[\d\s\-\]{5,15}$" required />
```

Перед відправкою форми дані атрибути забезпечують базову валідацію без необхідності написання додаткового JavaScript коду. При не заповненні обов'язкового поля користувачем або при введенні невірної форми браузер автоматично виведе повідомлення про помилку і не дозволить відправити форму.

В результаті даного тесту можемо сказати, що:

- модальне вікно відкривається і закривається коректно через певні кнопки;
- дані коректно відправляються на сервер;
- відповідь сервера коректно обробляється, користувач отримує повідомлення про успіх чи помилку.

Наступне випробування: тест перегляду портфолію, перевірка вкладок і роботи сортування. Метою випробування є перевірка, чи коректно працюють вкладки для перегляду портфолію, чи відбувається сортування проєктів студії за категоріями, чи відхиляються правильно проєкти в результаті пошуку. Для цього виконаємо наступні кроки:

- перевірка вкладок портфолію;
- перевірка сортування проєктів;
- перевірка роботи кнопки «Показати більше»;
- перевірка адаптивності.

На рисунку 4.17 зображено сторінку, на якій користувач може переглядати всі проєкти без застосування фільтрів. Усі доступні проєкти відображаються на одному екрані та впорядковуються за всіма категоріями. Це надає можливість ознайомитися з повним переліком робіт без жодних обмежень і сформулювати загальне уявлення про можливості компанії або платформи. Проєкти можуть бути подані у вигляді карток або списку.

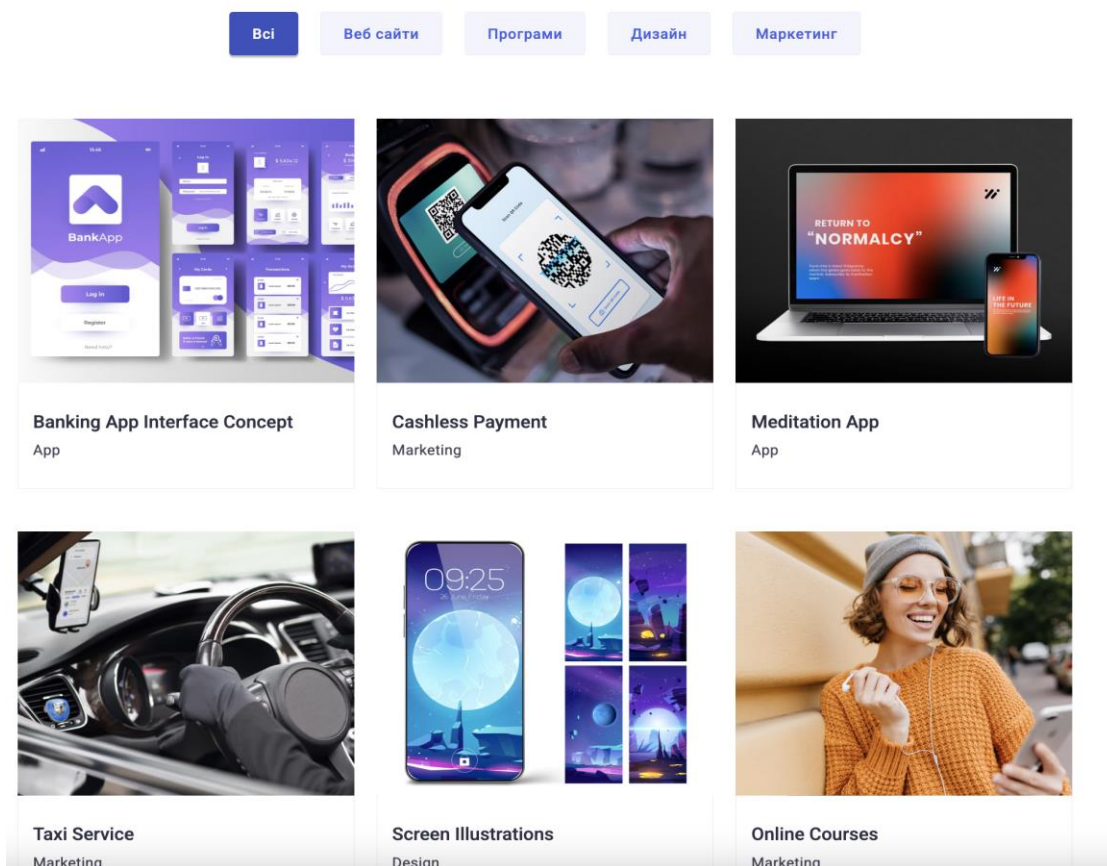


Рисунок 4.17 – Сторінка з усіма проєктами студії

На рисунку 4.18 представлено результат фільтрації проєктів за категорією «вебресурси». Після вибору відповідної вкладки або категорії користувач бачить лише ті проєкти, що пов’язані з розробкою вебресурсів. Усі роботи цього напрямку відображаються окремо, що дає змогу зосередити увагу виключно на проєктах у сфері вебдизайну та створення сайтів.

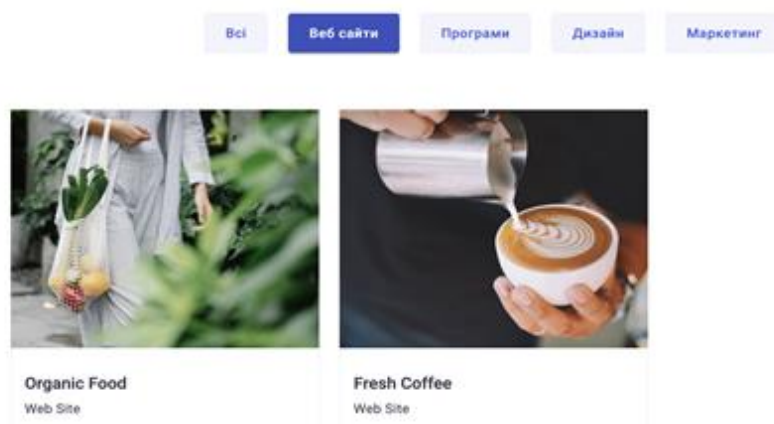


Рисунок 4.17 – Сторінка з проєктами студії категорії «вебресурси»

На рисунку 4.18 показано процес фільтрації проєктів за категорією «програми». У цьому режимі користувачеві відображаються лише ті проєкти, що пов'язані з розробкою програмного забезпечення. До них можуть належати мобільні застосунки, десктопні програми, системи автоматизації та інші програмні рішення. Такий спосіб сортування дає змогу швидко знаходити приклади робіт у сфері програмної розробки.

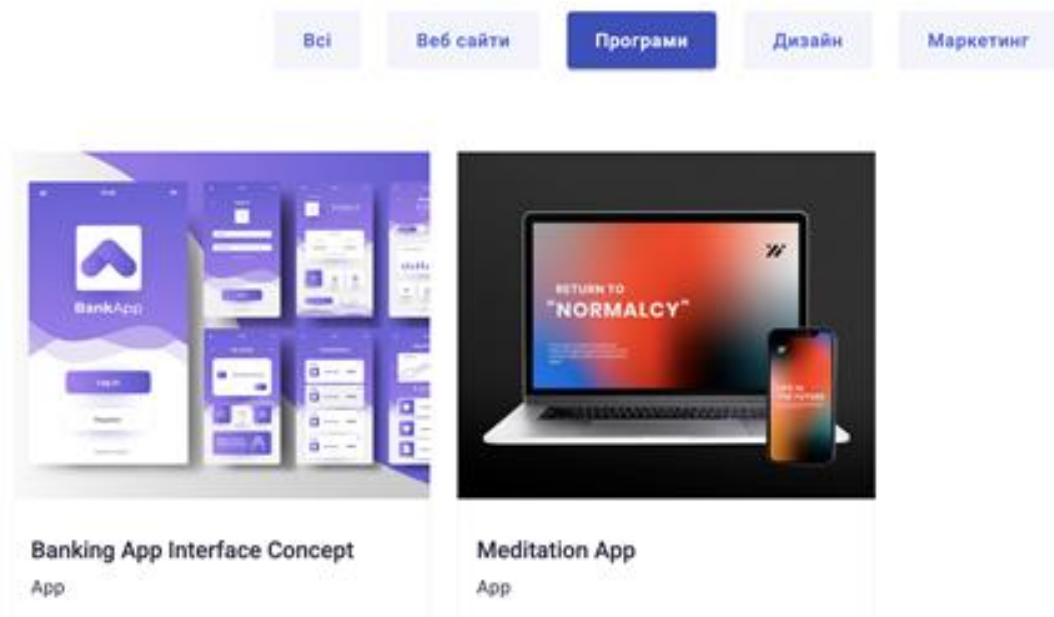


Рисунок 4.18 – Сторінка з проєктами студії категорії «програми»

На рисунку 4.19 зображено результат сортування проєктів за категорією «дизайн». Після вибору цієї категорії користувач отримує доступ лише до проєктів, пов'язаних із графічним дизайном, UI/UX-дизайном, брендингом та іншими напрямками дизайнерської діяльності. Такий підхід дає змогу швидко переглядати роботи, що належать до сфери дизайну.

На рисунку 4.20 представлено результат фільтрації проєктів за категорією «маркетинг». Після вибору цієї категорії користувачеві відображаються лише проєкти, пов'язані з маркетинговими стратегіями, рекламними кампаніями, SEO-оптимізацією та іншими маркетинговими послугами. Такий спосіб сортування дає змогу швидко знаходити маркетингові проєкти серед великої кількості інших робіт.

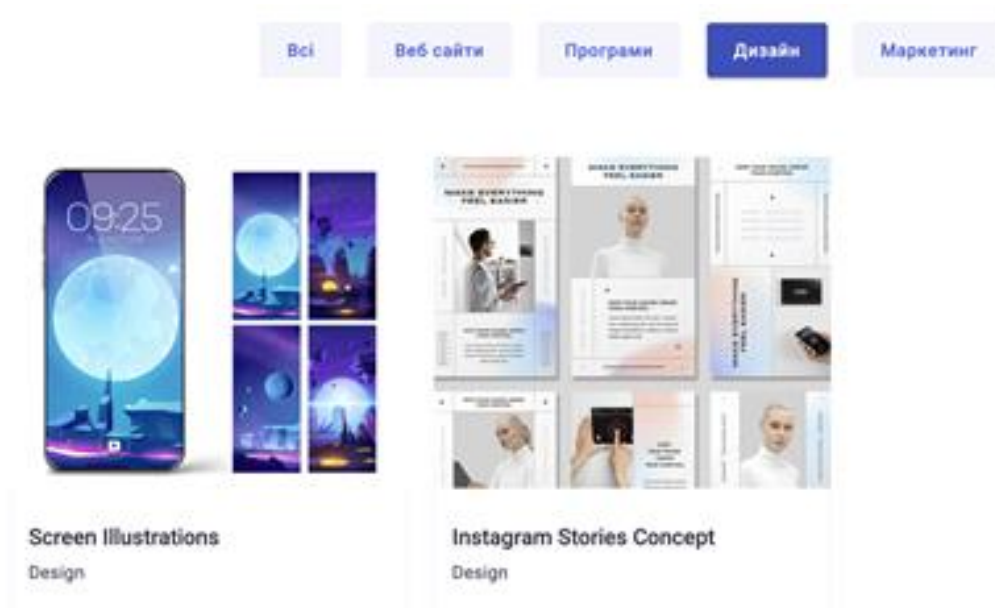


Рисунок 4.19 – Сторінка з проєктами студії категорії «дизайн»

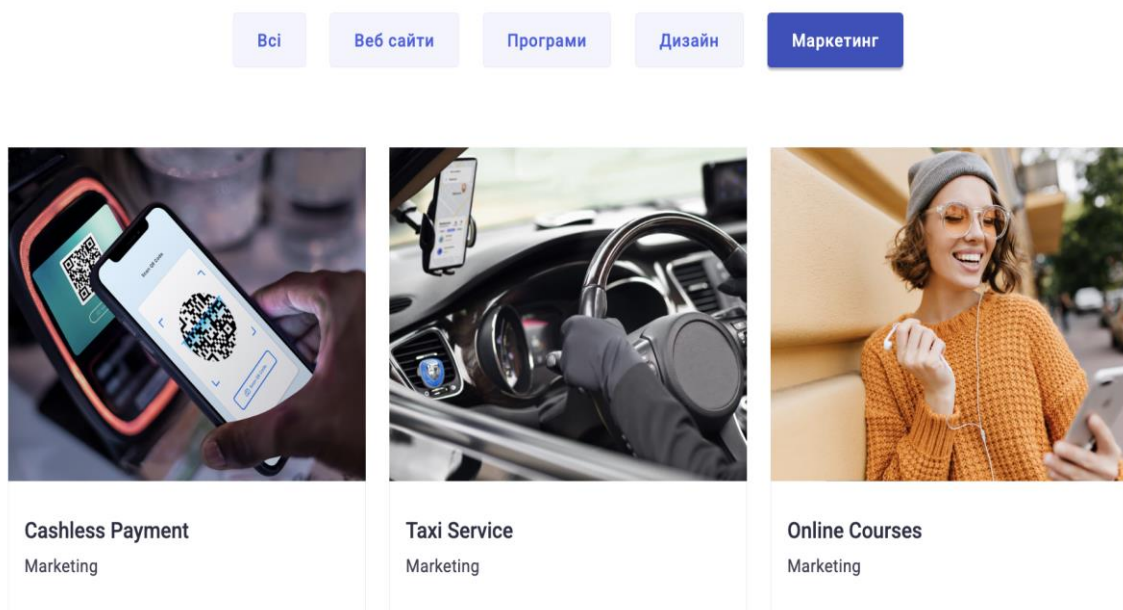


Рисунок 4.20 – Сторінка з проєктами студії категорії «маркетинг»

Кожен із рисунків (4.17 - 4.20) підтверджує, що після натискання на відповідну вкладку користувачеві відображаються лише ті проєкти, які належать до обраної категорії. Завдяки ефективній системі фільтрації та сортування користувач бачить тільки релевантні роботи. В результаті проведення тесту:

- вкладки працюють коректно, а для кожної категорії відображаються відповідні проєкти;

- кнопка «Показати більше» завантажує додаткові проекти без перезавантаження сторінки;
- фільтрація та сортування функціонують належним чином для всіх категорій;
- інтерфейс правильно відображається на мобільних пристроях.

Наступне випробування: попередній перегляд тексту проектів. Мета тесту є перевірка можливості попереднього перегляду текстового опису проектів перед переходом до детальної інформації про них. Для цього виконаємо наступні кроки:

- проведемо перевірку попереднього перегляду;
- проведемо перевірку правильності відображення тексту;
- проведемо перевірку адаптивності попереднього перегляду;
- проведемо перевірку закриття попереднього перегляду.

Після наведення на картку проекту, відкривається попередній перегляд цього проекту з детальним описом (рис. 4.21 -4.23).

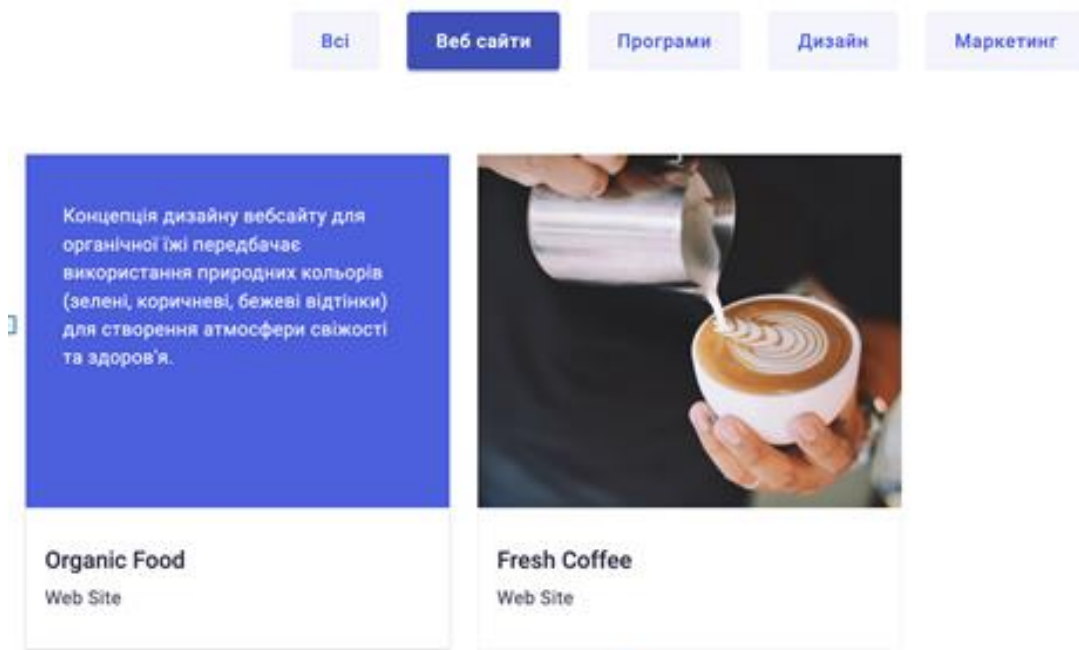


Рисунок 4.21 – Сторінка з описом проекту категорії «вебресурси»

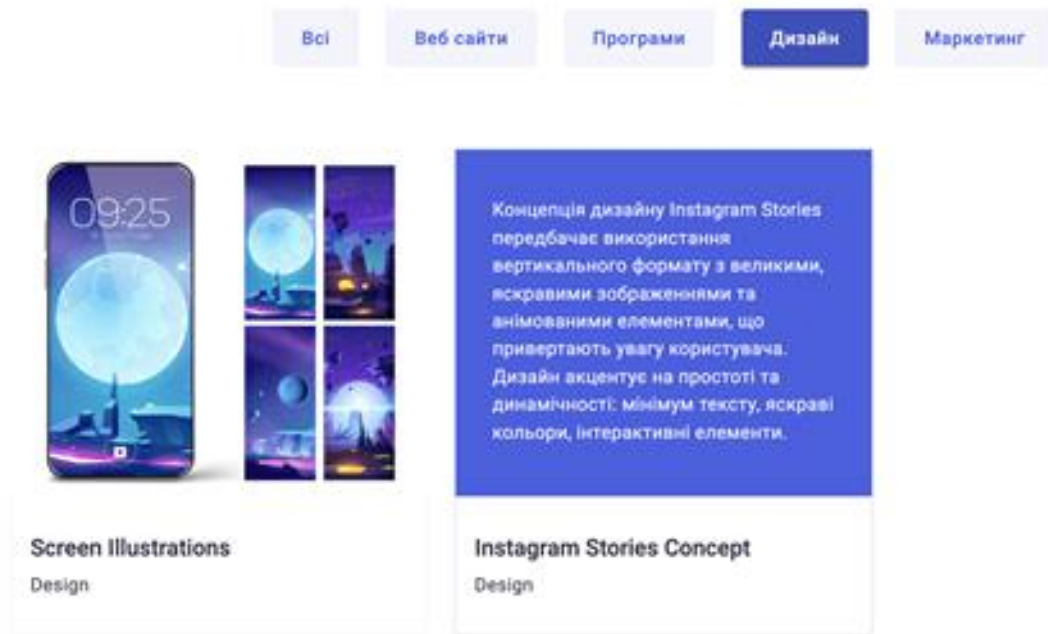


Рисунок 4.22 – Сторінка з описом проєкту категорії «дизайн»

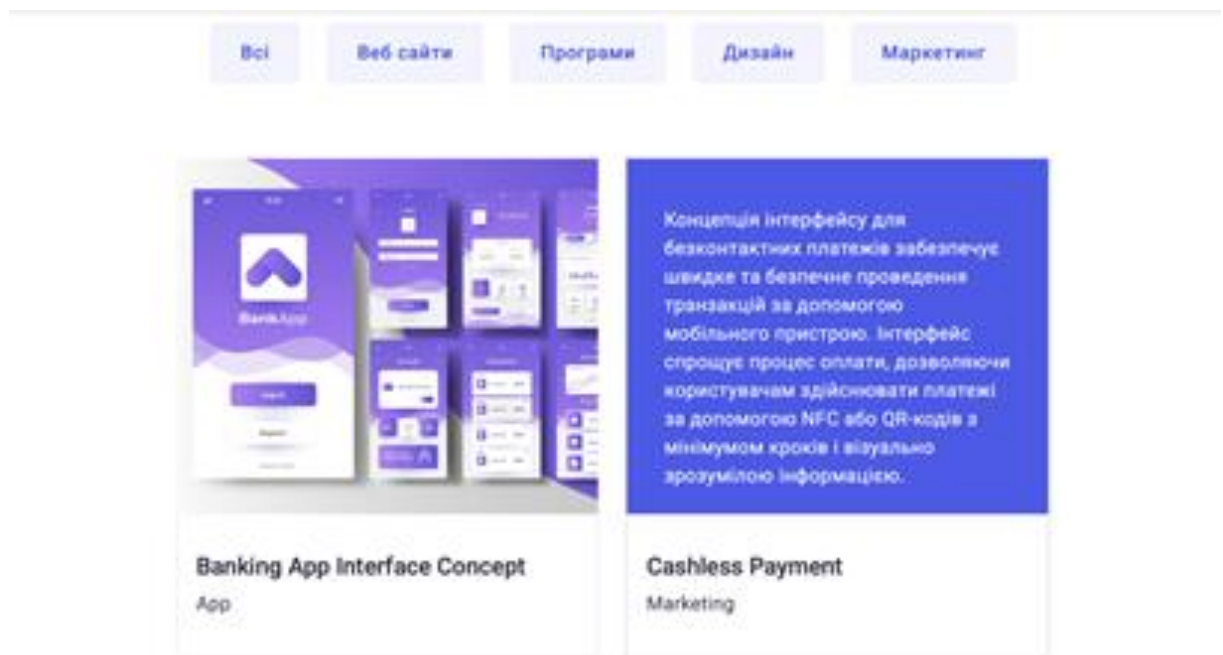


Рисунок 4.23 – Сторінка з описом проєкту категорії «маркетинг»

Таким чином у поточному розділі було детально описано процес розробки та інтеграції ключових компонентів вебресурсу студії «CreativePeople». У результаті виконаної роботи створено функціональний, зручний і адаптивний інтерфейс, який забезпечує високий рівень взаємодії з користувачами. Шапка

сайту, як один із основних елементів інтерфейсу, забезпечує зрозумілу та зручну навігацію, що допомагає користувачам швидко знаходити потрібну інформацію. Головна секція формує перше враження та привертає увагу відвідувачів, а секції переваг, проєктів, команди й клієнтів надають необхідні відомості про компанію. Функціональні кнопки та інтерактивні елементи сприяють комфортному перегляду контенту й покращують користувацький досвід.

Важливу роль у забезпеченні інтерактивності та адаптивності сайту відіграють модальне вікно, мобільне меню та система фільтрації карток проєктів. Використання модальних вікон і мобільного меню підвищує зручність взаємодії із сайтом, а фільтрація проєктів дає змогу швидко знаходити необхідну інформацію, роблячи навігацію більш ефективною.

Взаємодія між frontend та backend забезпечує ефективний обмін даними через форму зворотного зв'язку, завдяки чому користувачі можуть залишати свої контактні дані для подальшої комунікації з компанією. Усі елементи інтерфейсу взаємодіють із серверною частиною за допомогою JavaScript, що дозволяє обробляти запити та відображати результати у вигляді повідомлень про успішне виконання або виникнення помилки.

Тестування всіх компонентів підтвердило коректну роботу модального вікна, правильне надсилання даних форми на сервер, стабільне функціонування вкладок сортування проєктів, а також адаптивність попереднього перегляду тексту проєктів для різних типів пристроїв і розмірів екранів.

ВИСНОВКИ

У межах дипломної роботи було реалізовано клієнт-серверну комп'ютерну систему вебсервісу студії «CreativePeople» для демонстрації послуг компанії, поєднуючи зручний інтерфейс, адаптивний дизайн та інтерактивний функціонал. У процесі виконання роботи досягнуто поставленої мети підвищення ефективності доступу до інформації про послуги студії «CreativePeople».

Використання сучасних технологій веброзробки дало змогу створити зручний та інтуїтивно зрозумілий інтерфейс, який забезпечує ефективну взаємодію з користувачем. Завдяки застосуванню стандартів HTML5, CSS3 та JavaScript було досягнуто стабільного й швидкого відображення контенту вебсайту. Реалізація адаптивного дизайну забезпечила коректну роботу та відображення ресурсу на різних типах пристроїв - від смартфонів до широкоформатних моніторів.

У процесі виконання дипломної роботи виконано аналіз предметної області та існуючих технологій для реалізації вебресурсу, визначено функціональні та нефункціональні вимоги до системи. Описано моделювання програмного забезпечення. Наведено діаграму використання, яка допомагає визначити взаємодію користувачів з вебдодатком на рівні бізнес-логіки, не вдаючись у деталі реалізації, діаграму послідовності, яка ілюструє процес подачі заявки користувачем на вебресурсі, діаграму розгортання, яка показує фізичні компоненти вебресурсу, діаграму компонентів, яка показує, як різні компоненти системи взаємодіють один з одним. Виконано проєктування вебсервісу. Описано архітектуру вебсервісу. Детально розглянуто проєктування бази даних: визначені таблиці та зв'язки між ними.

Вебресурс містить інтерактивні елементи, зокрема модальні вікна, фільтри для сортування проєктів та динамічне оновлення контенту за допомогою JavaScript. Реалізовано функцію попереднього перегляду проєктів, яка дає змогу користувачам ознайомлюватися з деталями без переходу на окремі сторінки.

Також впроваджено систему сортування та фільтрації проєктів за категоріями, що забезпечує швидкий і зручний доступ до потрібних робіт.

Для перевірки коректності роботи основних компонентів вебресурсу проведено функціональне тестування, під час якого було перевірено форми, механізми сортування, модальні вікна та адаптивність інтерфейсу.

Вебресурс успішно пройшов тестування на різних пристроях і в різних браузерах, що підтвердило його високий рівень сумісності та адаптивності. Результати проєкту продемонстрували, що використання сучасних вебтехнологій дозволяє створювати зручні, функціональні та адаптивні вебресурси, які відповідають потребам сучасних користувачів. Забезпечити комфортну навігацію, швидкий доступ до інформації та ефективні засоби взаємодії вдалося завдяки якісній реалізації frontend- та backend-частин системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Функціональні та нефункціональні вимоги. URL: <https://visuresolutions.com/uk/посібник-з-відстеження-управління-вимогами/функціональні-та-нефункціональні-вимоги/> (дата звернення: 10.03.2026).
2. W3C. HTML5 Specifications. URL: <https://www.w3.org/TR/html5/> (дата звернення: 01.03.2026).
3. W3C. CSS: Cascading Style Sheets. URL: <https://www.w3.org/TR/css/> (дата звернення: 10.03.2026).
4. Flanagan, D. JavaScript: The Definitive Guide. O'Reilly Media, 2020, 600 p.
5. What is REST? URL: <https://www.codecademy.com/article/what-is-rest> (дата звернення: 10.03.2026).
6. Bendix, D., & Sharma, S. Mastering Web Application Development with React. Packt Publishing, 2021, 378 p.
7. Misko Hevery. Angular: The Complete Guide. URL: <https://angular.io/> (дата звернення: 01.03.2026).
8. Vue.js Documentation. Vue.js - The Progressive JavaScript Framework. URL: <https://vuejs.org/> (дата звернення: 01.03.2026).
9. Python. URL: <https://www.python.org/> дата звернення: 01.03.2026).
10. Flask. URL: <https://flask.palletsprojects.com/en/stable/> (дата звернення: 01.03.2026).
11. SQLite URL: <https://sqlite.org/> (дата звернення: 10.03.2026).
12. What is a UML Diagram? | Different Types and Benefits - Miro. URL: <https://miro.com/diagramming/what-is-a-uml-diagram/> (дата звернення: 10.03.2026).
13. TypeScript. URL: <https://www.typescriptlang.org/> (дата звернення: 06.04.2026).
14. What are microservices? URL: <https://microservices.io/> (дата звернення: 10.03.2026).

15. GraphQL Foundation. GraphQL Specification. URL: <https://spec.graphql.org/> (дата звернення: 10.03.2026).
16. Client-Server Model. URL: <https://www.geeksforgeeks.org/client-server-model/> (дата звернення: 10.03.2026).
17. What is SOA (Service-Oriented Architecture). URL: <https://aws.amazon.com/what-is/service-oriented-architecture/> (дата звернення: 10.03.2026).
18. Fielding, R. T. Architectural Styles and the Design of Network-based Software Architectures (PhD Thesis). University of California, Irvine, 2000, 162 p.
19. Shostack, A. Threat Modeling: Designing for Security. Wiley, 2014, 624 p.
20. Hoffman, A.. Web Application Security: Exploitation and Countermeasures for Modern Web Applications. O'Reilly Media, 2019, 217 p.
21. What is a content delivery network (CDN)? URL: <https://www.cloudflare.com/learning/cdn/what-is-a-cdn/> (дата звернення: 10.03.2026).

ДОДАТОК А

СЛАЙДИ ПРЕЗЕНТАЦІЇ

Національний університет «Запорізька політехніка»
Кафедра комп'ютерних систем та мереж



Дипломна робота на тему:
«Розробка комп'ютерної системи вебресурсу
студії «CreativePeople»

Виконавець: ст. гр. КНТз-513сп Шаповал М.О.

Керівник дипломної роботи: Скрупський С.Ю.

Рисунок А.1 – Слайд 1

Предмет розробки – клієнт-серверна комп'ютерна система для вебресурсу студії «CreativePeople».

Мета роботи – підвищення ефективності роботи компанії за рахунок використання комп'ютерної системи.

Основні етапи розробки комп'ютерної системи:

- аналіз технічного завдання;
- моделювання вебресурсу;
- проектування системи;
- реалізація та тестування системи.

Рисунок А.2 – Слайд 2

Основні задачі комп'ютерної системи:

- презентація експертизи та позиціонування компанії;
- демонстрація практичного досвіду;
- забезпечення прозорості та соціального доказу;
- автоматизація збору потенційних клієнтів та зворотного зв'язку;
- утримання аудиторії та робота з лояльністю;
- адаптивність та доступність.

3

Рисунок А.3 – Слайд 3

Функціональні вимоги до системи

Вимога	Опис
Оформлення запитів на розробку сайту	Користувачі повинні мати можливість оформити заявку на послугу за допомогою зручної форми
Інтерактивна галерея робіт	Студія має можливість демонструвати свої попередні проекти за допомогою інтерактивної галереї
Контактна форма	Має бути можливість зворотнього зв'язку з командою студії через контактну форму на сайті
Оцінка вартості послуг	Користувач повинен мати можливість отримати попередню вартість проекту, заповнивши форму
Публікація новин та оновлень	Адміністратор повинен мати можливість додавати новини та оновлювати інформацію на сайті
Інтеграція з соціальними мережами	Вебресурс повинні мати можливість ділитися контентом та проектами студії між користувачами через соціальні мережі
Пошукова функціональність	Користувачі повинні мати можливість зробити пошук потрібних їм проектів та послуг за допомогою пошукової системи

4

Рисунок А.4 – Слайд 4

Нефункціональні вимоги до системи

Вимога	Опис
Продуктивність	Вебресурс повинен швидко завантажуватись (максимум 3 секунди на сторінці)
Безпека	Всі персональні дані користувачів повинні бути захищені за допомогою шифрування та безпечного з'єднання (HTTPS)
Масштабованість	Система повинна мати можливість масштабування для підтримки великої кількості користувачів та наданих їм послуг (проектів)
Доступність	Вебресурс має бути доступний для всіх користувачів, включаючи людей з обмеженими можливостями (ADA compliance)
Адаптивний дизайн	Вебресурс повинен коректно відображатися на різних пристроях: мобільних телефонах, планшетах та десктопах
Надійність	При високому навантаженні система повинна бути стабільною і працювати без збоїв
Локалізація	Вебресурс має підтримувати кілька мов, щоб задовольнити потреби різних регіонів (наприклад, англійська та українська)
Оновлення та підтримка	Система повинна підтримувати автоматичні оновлення та зручний механізм для обслуговування

5

Рисунок А.5 – Слайд 5

Діаграма використання



Діаграма використання демонструє взаємодію користувачів і адміністратора з вебресурсом студії.

6

Рисунок А.6 – Слайд 6

Діаграма послідовності «Подання заявки»



Діаграма послідовності демонструє процес подання заявки користувачем на вебплатформі.

7

Рисунок А.7 – Слайд 7

Діаграма розгортання

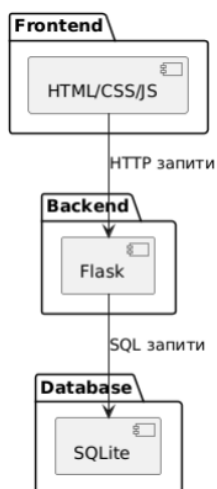


Діаграма розгортання ілюструє фізичну структуру вебресурсу та взаємодію його частин

8

Рисунок А.8 – Слайд 8

Діаграма компонентів

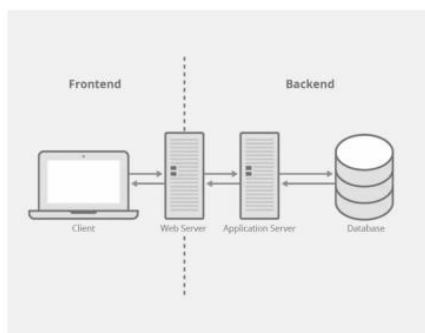
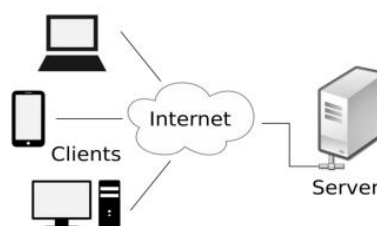


Діаграма компонентів ілюструє ключові складові вебресурсу

9

Рисунок А.9 – Слайд 9

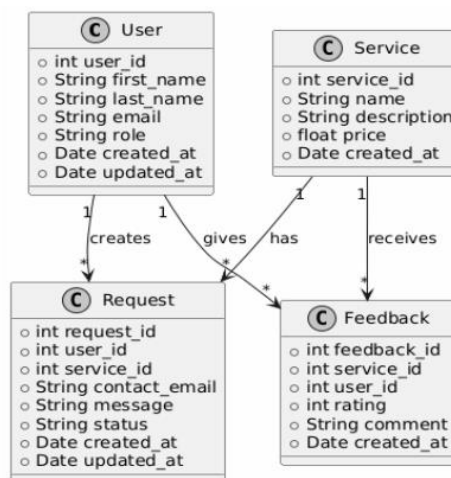
Архітектура вебресурсу



10

Рисунок А.10 – Слайд 10

ER-модель у вигляді діаграми класів



11

Рисунок А.11 – Слайд 11

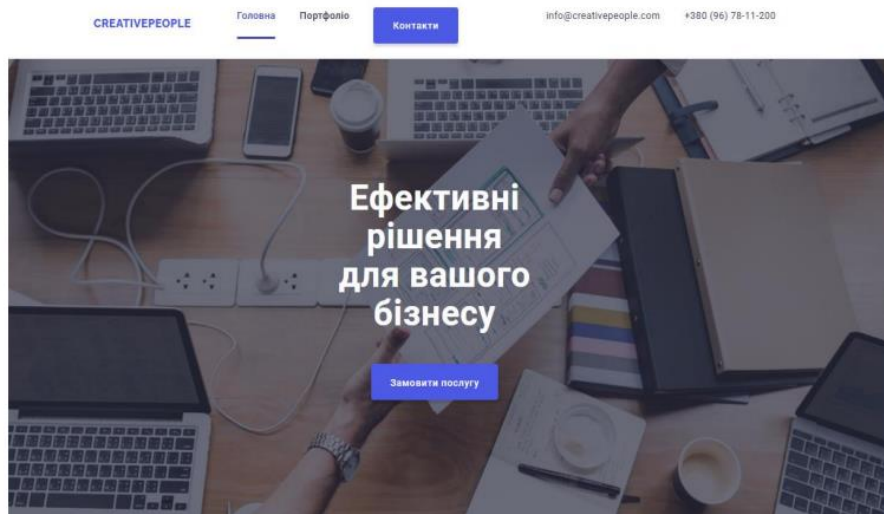
Технології для розробки системи

Компонент	Технологія	Переваги	Недоліки
Frontend	HTML, CSS, JavaScript	Стандарти для веброзробки. Підтримка всіх сучасних браузерів. Велика кількість фреймворків та бібліотек для швидкої розробки.	Потребує знань у галузі адаптивного дизайну та кросбраузерної сумісності. Без сторонніх бібліотек складно створити складні інтерфейси.
Backend	Python (Flask)	Простота в освоєнні та читабельність коду. Швидка розробка завдяки мінімалістичній структурі Flask. Велика кількість бібліотек для різних завдань.	Може бути недостатньо швидким для дуже високонавантажених додатків. Потребує додаткових інструментів для великих проєктів.
База даних	SQLite	Легка в налаштуванні, не потребує окремого сервера. Добре підходить для невеликих проєктів. Підтримка SQL-операцій через SQLAlchemy.	Обмежена масштабованість для великих проєктів. Можуть виникати проблеми з обробкою великих обсягів даних.

12

Рисунок А.12 – Слайд 12

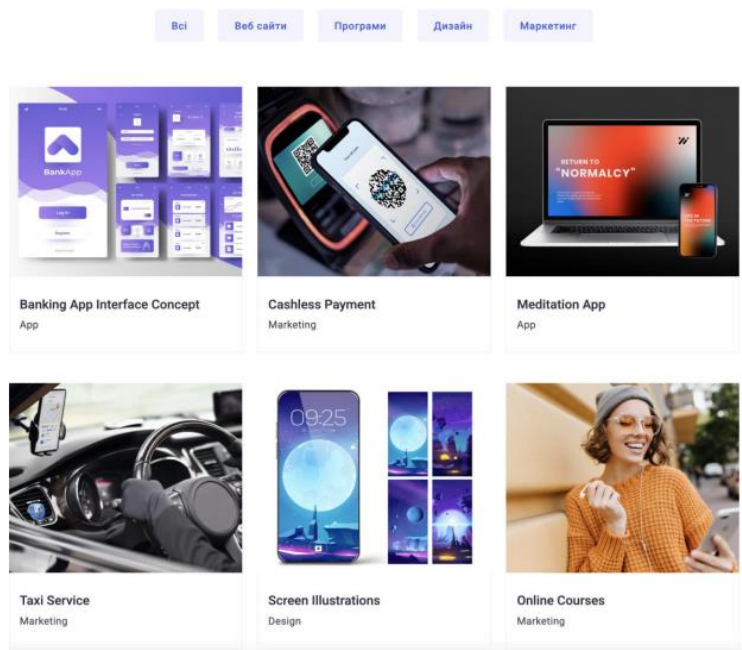
Головна секція вебресурсу



13

Рисунок А.13 – Слайд 13

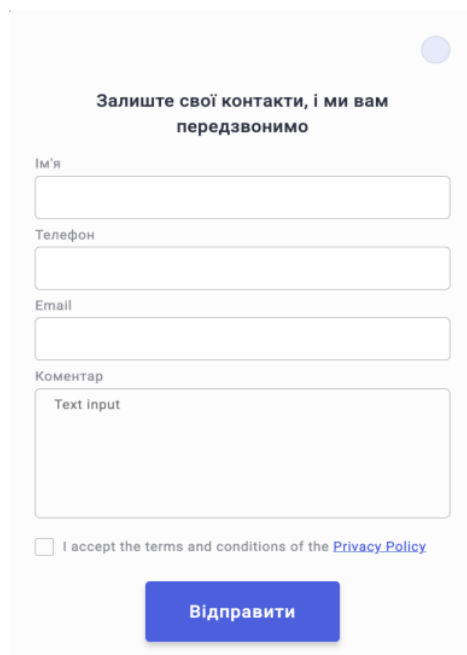
Секція з переліком послуг



14

Рисунок А.14 – Слайд 14

Форма заповнення даних клієнтів



Залиште свої контакти, і ми вам передзвонимо

Ім'я

Телефон

Email

Коментар

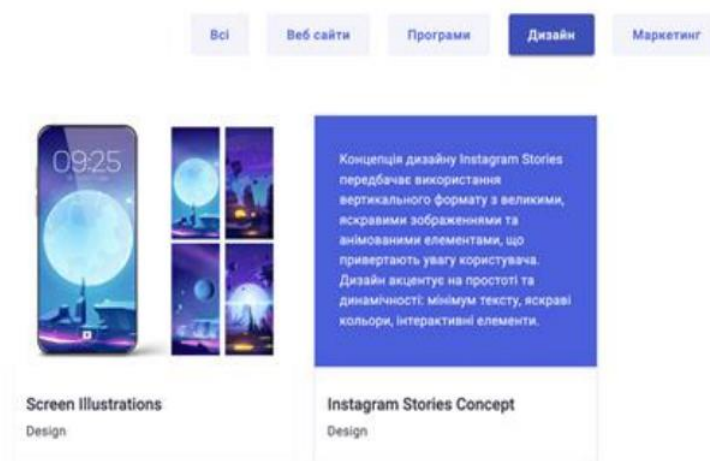
☐ I accept the terms and conditions of the [Privacy Policy](#)

Відправити

15

Рисунок А.15 – Слайд 15

Сторінка з описом проєкту категорії «дизайн»



16

Рисунок А.16 – Слайд 16

Висновки

В дипломній роботі було спроектовано та розроблено клієнт-серверну комп'ютерну систему вебсервісу студії «CreativePeople» .

Дана система реалізує поставлені задачі, а саме:

- презентація експертизи та позиціонування компанії;
- демонстрація практичного досвіду;
- забезпечення прозорості та соціального доказу;
- автоматизація збору потенційних клієнтів та зворотного зв'язку;
- утримання аудиторії та робота з лояльністю;
- адаптивність та доступність.

Розроблена система має зручний інтерфейс, адаптивний дизайн та інтерактивний функціонал.

Дякую за увагу!

17

Рисунок А.17 – Слайд 17