Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук

Бакалаврська кваліфікаційна робота на тему:

«РОЗРОБКА WEB-РЕСУРСУ ДЛЯ РОБОТИ З КЛІЄНТАМИ»

Виконала: студентка 4 курсу, групи 3КН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)
Шевченко О. О.
(прізвище та ініціали)

Керівник: доцент каф. КН
Озеранський В. С.
(прізвище та ініціали)
«4» червня 2025 р.

Рецензент: доцент каф. АІІТ
Барабан М.В.
(прізвище та ініціали)
«5» червня 2025 р.

Допущено до захисту
Зав. кафедри Яровий А.А.
«6» червня 2025 р.

Вінниця ВНТУ – 2025 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних наук
Рівень вищої освіти перший (бакалаврський)
Галузь знань – 12 Інформаційні технології
Спеціальність – 122 Комп'ютерні науки
Освітньо - професійна програма – Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри КН
проф., д.т.н. Яровий А.А.
«5» травня 2025р.

ЗАВДАННЯ
НА БАКАЛАВРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТУ
Шевченко Олені Олександрівні

1. Тема роботи: Розробка WEB-ресурсу для роботи з клієнтами.
керівник роботи: Озеранський Володимир Сергійович, к.т.н., доцент
затверджені наказом вищого навчального закладу "10" березня 2025 року №97

2. Строк подання студентом роботи 30 травня 2025

3. Вхідні дані:
мова програмування – об'єктно-орієнтована, має забезпечувати можливість маніпулювання даними та управління даними;
Кількість користувачів - не менше 100 чол;
Кількість одночасного підключення - не менше 100 чол;
Інтерфейс користувача має бути інтуїтивно зрозумілим.

4. Короткий зміст частин бакалаврської кваліфікаційної роботи: Схема загального алгоритму функціонування системи, структура програмного модуля, модель функціонування системи взаємодії з клієнтом, загальна схема класів.
5. Текстова (пояснювальна записка): вступ, аналіз сучасних технологій WEB-розробки для взаємодії з клієнтами, аналіз існуючих аналогів WEB-ресурсів для роботи з клієнтами, проєктування WEB-ресурсу, формулювання об'єкту, предмету, мети та задач подальшого дослідження, висновки, література, додатки

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-3	Озеранський В.С. к.т.н., доцент		

7. Дата видачі завдання: 5 травня 2025 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз сучасних технологій розробки WEB-ресурсів для взаємодії з клієнтами.	05.05.25	07.05.25	
2	Аналіз переваг та недоліків існуючих аналогічних WEB-ресурсів.	08.05.25	09.05.25	
3	Розробку структури WEB-ресурсу для роботи з клієнтами.	10.05.25	12.05.25	
4	Розробку алгоритму обробки клієнтських запитів через WEB-ресурс	13.05.25	15.05.25	
5	Обґрунтування вибору інструментів технічної реалізації WEB-ресурсу	16.05.25	21.05.25	
6	Програмна реалізація WEB-ресурсу	22.05.25	28.05.25	
7	Тестування розробленого WEB-ресурсу	29.05.25	31.05.25	
8	Оформлення матеріалів до захисту БКР	01.06.25	04.06.25	

Студентка _____ **Шевченко О.О.**
(підпис) (прізвище та ініціали)

Керівник проекту (роботи) _____ **Озеранський В.С**
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Бакалаврська кваліфікаційна робота складається з 62 сторінок формату А4, містить 26 рисунків та список використаних джерел із 20 найменувань.

Робота присвячена розробці вебресурсу для підприємства з виготовлення вікон та супутньої продукції, що має на меті покращити онлайн-присутність компанії, підвищити зручність для користувачів і автоматизувати обробку запитів клієнтів. Основний акцент зроблено на практичному створенні вебресурсу з використанням сучасних технологій веброзробки, таких як Webflow, Zapier, HTML, CSS, JavaScript.

Ключові слова: WEB-ресурс, Webflow, Zapier, Telegram-бот, JavaScript, форма зворотного зв’язку, автоматизація.

ABSTRACT

The bachelor's qualification thesis consists of 62 A4 pages, includes 26 figures, and a list of 20 references.

The work is dedicated to the development of a web resource for a company specializing in the production of windows and related products, aimed at improving the company’s online presence, enhancing user convenience, and automating the processing of client requests. The main focus is on the practical creation of a web resource using modern web development technologies such as Webflow, Zapier, HTML, CSS, and JavaScript.

Keywords: web resource, Webflow, Zapier, Telegram bot, JavaScript, feedback form, automation.

ЗМІСТ

ВСТУП

Актуальність досліджень. В умовах цифровізації бізнесу WEB-ресурси стали невіддільною складовою сучасного підприємництва. Сайт вже давно перестав бути лише візитною карткою компанії — він перетворився на повноцінний інструмент комунікації, продажу, аналітики та зворотного зв'язку [17]. Саме через WEB-ресурс клієнти отримують перше враження про компанію, дізнаються про послуги чи товари, залишають запити, здійснюють замовлення або звертаються за консультацією. Усе це визначає необхідність створення зручних, інформативних і технологічно ефективних сайтів, які відповідають сучасним вимогам користувача.

WEB-ресурси для роботи з клієнтами повинні поєднувати в собі продуману структуру, привабливий інтерфейс, адаптивність до різних пристроїв та функціональні можливості для інтерактивної взаємодії. Крім того, вони мають бути здатними обробляти вхідні запити, інтегруватися з системами сповіщення, базами даних, CRM-платформами тощо. Все це робить процес розробки таких сайтів багаторівневим завданням, яке потребує комплексного підходу.

Метою бакалаврської кваліфікаційної роботи є розширення функціональних можливостей програмного забезпечення для роботи з клієнтами.

Об'єктом дослідження є процес розробки WEB-ресурсу для роботи з клієнтами.

Предметом дослідження є програмні засоби розробки WEB-ресурсу для роботи з клієнтами.

Для досягнення поставленої мети необхідно виконати такі **задачі**:

- проаналізувати сучасні технології розробки WEB-ресурсів для роботи з клієнтами;
- проаналізувати переваги та недоліки програм-аналогів для роботи з клієнтами;
- розробити алгоритм роботи WEB-ресурсу для роботи з клієнтами;
- здійснити програмну реалізацію WEB-ресурсу для роботи з клієнтами;

- виконати тестування програмного забезпечення та провести аналіз отриманих результатів.

Апробація результатів роботи. Результати дослідження впроваджено в роботу ТОВ «ІТІ».

1 АНАЛІЗ СУЧАСНИХ ТЕХНОЛОГІЙ WEB-РОЗРОБКИ ДЛЯ ВЗАЄМОДІЇ З КЛІЄНТАМИ

1.1 Роль WEB-ресурсів у сучасному бізнесі

Сучасні WEB-ресурси, орієнтовані на взаємодію з клієнтами, проєктуються із застосуванням широкого спектру технологій, які забезпечують не лише привабливий зовнішній вигляд, але й функціональність, гнучкість та масштабованість. У зв'язку з постійним зростанням вимог до якості обслуговування, WEB-ресурси повинні відповідати критеріям зручності, швидкодії та надійності, а також адаптуватися до потреб кінцевого користувача. Порівняння технологій веб-розробки описано у таблиці 1.1.

WEB-розробка для взаємодії з клієнтами охоплює кілька ключових напрямів:

- фронтенд: реалізується за допомогою HTML, CSS, JavaScript та сучасних фреймворків (React [12], Vue.js [13], Angular [14]). Саме ця частина відповідає за візуальне сприйняття сайту та зручність взаємодії. Особливої актуальності набуває адаптивний дизайн, що дозволяє користувачам комфортно працювати з сайтом як на комп'ютерах, так і на мобільних пристроях.

- бекенд**:** відповідає за обробку запитів користувачів, зберігання інформації та забезпечення логіки роботи сервісу. Найпоширенішими технологіями для бекенду є PHP, Node.js, Python (Django, Flask), Ruby on Rails тощо. Серверна частина часто інтегрується з базами даних (MySQL, PostgreSQL, MongoDB), CRM-системами та сторонніми API [2].

- системи керування контентом (CMS): дозволяють створювати та керувати контентом сайту без глибоких технічних знань. Популярними є WordPress, Joomla, Drupal, Webflow, Tilda. Для малого та середнього бізнесу вони є зручним рішенням завдяки швидкому розгортанню та простоті використання [1].

- хмарні сервіси та інтеграції: для автоматизації взаємодії з клієнтом широко використовуються хмарні платформи та інтеграції з месенджерами (Telegram, Viber, Facebook Messenger), електронною поштою, Google-формами,

онлайн-опитуваннями. API-сервіси дозволяють автоматизувати процеси зворотного зв’язку, оформлення замовлень, ведення аналітики.

- безпека та конфіденційність: сучасні WEB-ресурси повинні відповідати вимогам GDPR та стандартам захисту даних. Це передбачає впровадження SSL-сертифікатів, систем авторизації та захисту від SQL-ін’єкцій, CSRF, XSS.
- платформи без коду (no-code/low-code): останнім часом значно зросла популярність платформ, які дозволяють створювати сайти без програмування. Такі сервіси, як Webflow, Tilda, Wix, дають змогу швидко зібрати адаптивний сайт із необхідним функціоналом, використовуючи візуальні інструменти, що особливо зручно для малих компаній.

Таблиця 1.1 - Порівняння технологій веб-розробки

Критерії	Фреймворки (React, Vue)	CMS	No-code платформи
Складність розробки	Висока	Середня	Низька
Гнучкість	Максимальна	Обмежена	Дуже обмежена
Час створення	1-3 місяці (середній проєкт)	1-4 тижні	1-7 днів
Вартість	Висока	Низька	Середня
SEO	Залежить від розробника	Готові інструменти	Базові інструменти
Інтеграції	Майже будь-які	Плагіни (обмежений вибір)	Лише підтримувані сервіси
Підтримка	Залежить від розробника	Велика спільнота	Обмежена документацією
Можливості	Складні веб-додатки	Блоги, прості магазини	Лендінги, прості сайти
Приклади	Facebook, Airbnb	BBC Blog, Sony Music	Сайти малого бізнесу, портфоліо

Таким чином, розробка WEB-ресурсів для взаємодії з клієнтами базується на використанні широкого спектру інструментів — від класичних мов програмування до інноваційних візуальних редакторів. Вибір конкретної технології залежить від вимог до функціоналу, бюджету проєкту, термінів реалізації та рівня технічної підготовки розробника.

1.2 Аналіз переваг та недоліків існуючих аналогічних WEB-ресурсів

У сучасних реаліях розвиток інформаційних технологій призвів до значного збільшення кількості WEB-ресурсів, які спрямовані на ефективну взаємодію з клієнтами. Вони стали невід'ємною частиною бізнесу, адже дозволяють не лише надати інформацію про товари чи послуги, а й створити зручний та швидкий канал комунікації. Для того, щоб розробити власний WEB-ресурс, який максимально відповідатиме потребам користувачів і бізнесу, необхідно ретельно проаналізувати існуючі рішення на ринку. Такий аналіз допомагає виокремити найкращі практики, зрозуміти типові помилки, а також визначити напрямки, які потребують вдосконалення.

Сьогодні для створення WEB-ресурсів, що працюють з клієнтами, найчастіше використовують два основні підходи: застосування систем управління контентом (CMS) та використання no-code/low-code конструкторів сайтів. Обидва варіанти мають як свої переваги, так і обмеження, і вибір залежить від конкретних цілей, бюджету та технічних можливостей компанії.

Розглянемо детальніше основні переваги систем управління контентом. CMS, такі як WordPress, Joomla або Drupal, надають розробникам і адміністраторам сайтів великий вибір інструментів для налаштування функціоналу. Завдяки широкій спільноті користувачів і розробників існує безліч готових шаблонів та плагінів, які значно пришвидшують процес створення ресурсу. Крім того, відкритий код таких систем дозволяє гнучко кастомізувати майже всі аспекти роботи сайту, підлаштовуючи його під унікальні потреби бізнесу. Окрім цього, CMS активно підтримують інтеграції з CRM-системами, платіжними сервісами та

різними аналітичними інструментами, що робить їх універсальними рішеннями для малого та середнього бізнесу.

Проте використання CMS пов’язане і з певними викликами. Часто з’являються проблеми з безпекою через вразливості плагінів або застаріле програмне забезпечення, що може призводити до витоку даних або кібератак. Також, щоб підтримувати сайт у належному стані, необхідно регулярно оновлювати систему, плагіни і шаблони, що потребує додаткових ресурсів та часу. Крім того, велика кількість плагінів може негативно впливати на продуктивність сайту [4], уповільнюючи його роботу та погіршуючи користувацький досвід. В деяких випадках масштабування таких сайтів під високі навантаження може бути складним та дорогим процесом.

З іншого боку, no-code і low-code платформи набирають все більшої популярності серед підприємців та маркетологів, які прагнуть швидко і без глибоких технічних знань створити якісний веб-сайт. Такі інструменти, як Webflow, Tilda або Wix, пропонують інтуїтивно зрозумілі інтерфейси для візуального створення сторінок, що значно спрощує роботу з дизайном та контентом [10]. Вони часто включають вбудовані рішення для адаптивної верстки, SEO-оптимізації, аналітики, а також можливості підключення зовнішніх сервісів, таких як месенджери і CRM. Окрім того, всі технічні оновлення і підтримка здійснюються провайдером платформи, що звільняє користувача від технічних турбот. Порівняння CMS та No-code платформ описану у таблиці 1.2.

Разом з тим, no-code рішення мають свої обмеження. Вони не завжди дозволяють реалізувати складну кастомізацію або специфічні функції, що може бути критично важливо для великих компаній з унікальними бізнес-процесами. Вартість підписки на такі сервіси іноді перевищує витрати на власну розробку в довгостроковій перспективі. Крім того, існує певна залежність від платформи: якщо провайдер припинить підтримку чи змінить умови, це може суттєво вплинути на роботу ресурсу. Ще одним фактором є обмеження щодо швидкості і продуктивності при збільшенні кількості користувачів або функціональних модулів.

Отже, підсумовуючи, можна стверджувати, що аналіз існуючих аналогів показує — вибір платформи для створення WEB-ресурсу залежить від конкретних вимог бізнесу, доступних ресурсів і довгострокових планів розвитку. Малий та середній бізнес часто віддає перевагу no-code платформам через їх простоту і швидкість реалізації, тоді як великі підприємства та організації зі складною структурою частіше обирають CMS з власною розробкою бекенду для більшої гнучкості і контролю.

1.3 Вибір платформи для реалізації проєкту

Вибір платформи для розробки WEB-ресурсу є одним із ключових етапів планування проекту. Від цього вибору залежить не тільки швидкість реалізації, але й якість кінцевого продукту, його гнучкість, масштабованість і здатність відповідати вимогам бізнесу та користувачів. Тому дуже важливо ретельно проаналізувати всі доступні варіанти та обрати той, який максимально відповідає поставленим цілям і задачам.

На сучасному ринку існує велика кількість платформ і технологій для створення WEB-ресурсів. Найпоширенішими серед них є системи управління контентом (CMS), а також no-code і low-code конструктори сайтів. Кожен з цих підходів має свої особливості, які варто розглянути більш детально.

Системи управління контентом, такі як WordPress, Joomla, Drupal та інші, є найбільш універсальними та гнучкими інструментами. Вони дають змогу створювати як прості інформаційні сайти, так і складні корпоративні портали з різноманітним функціоналом. Величезна кількість плагінів та модулів дозволяє розширювати можливості сайту без необхідності писати код з нуля, що суттєво економить час і ресурси розробників. Приклад сайту створений на Wordpress наведено на рисунку 1.1. Однак робота з CMS вимагає певного рівня технічної підготовки, а також уваги до підтримки і безпеки ресурсу в процесі його експлуатації. Особливу увагу слід приділити користувацькому досвіду (UX) та дизайну інтерфейсу (UI), оскільки саме ці елементи найчастіше визначають,

наскільки комфортно клієнти будуть взаємодіяти з ресурсом. Вдалий дизайн із простою навігацією, швидким доступом до контактної інформації і зрозумілими формами зворотного зв'язку значно підвищує ймовірність конверсії та формує довіру користувачів. Аналіз найуспішніших аналогів показує, що сайти з прозорою структурою, мінімумом зайвих елементів і швидким завантаженням займають лідируючі позиції на ринку.

Таблиця 1.2 - Порівняння CMS та No-code платформ

Характеристика	CMS (Wordpress, Joomla)	Low-code платформи (Webflow, Tilda)
Функціональність	Висока (плагіни, власний код)	Середня (власний код, обмежені плагіни)
Вартість	$50-300/рік (хостинг+плагіни)	$12-45 (підписка)
Безпека	Середня (потрібні оновлення)	Висока
Простота використання	Середня (потрібне навчання)	Дуже висока
Продуктивність	Залежить від оптимізації	Стабільна (хостинг від платформи)

Рисунок 1.1 - Приклад сайту створений на Wordpress

Low-code платформи, такі як Webflow, Tilda, Wix, набирають популярності завдяки простоті використання та можливості швидко створювати сайти без

глибоких технічних знань. Вони ідеально підходять для малого та середнього бізнесу, де важливі швидкість запуску і зручність редагування контенту. Крім того, такі платформи зазвичай мають вбудовані інструменти для SEO-оптимізації, адаптивного дизайну і інтеграції з різними сервісами, що полегшує подальше просування ресурсу. Втім, варто пам’ятати про певні обмеження, такі як менша гнучкість та залежність від провайдера платформи, які можуть стати суттєвими у разі масштабування проекту. Приклад сайту створений у Webflow наведений на рисунку 1.2.

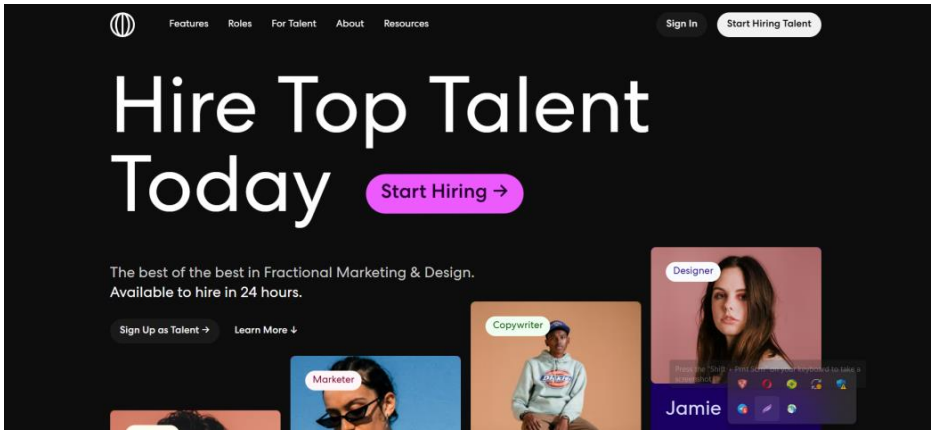


Рисунок 1.2 - приклад сайту створений у Webflow

При виборі платформи для конкретного проекту важливо враховувати кілька ключових факторів: цілі і масштаб проекту, бюджет, технічні ресурси команди, а також вимоги до функціоналу і дизайну. Наприклад, якщо проект потребує швидкого запуску із базовим набором функцій, а команда не має сильних технічних навичок, то no-code платформи будуть оптимальним вибором. Якщо ж необхідна складна інтеграція з внутрішніми системами компанії, висока безпека та можливість розширення функціоналу, то краще обирати CMS або індивідуальну розробку.

Також не варто забувати про питання підтримки і обслуговування ресурсу після його запуску. Обрана платформа повинна забезпечувати можливість регулярних оновлень, виправлення помилок і адаптації до нових вимог бізнесу. Важливо, щоб розробники або команда підтримки мали досвід роботи з обраним

інструментом, що дозволить уникнути зайвих витрат часу та ресурсів у майбутньому.

1.4 Інструменти та сервіси, які застосовувалися у розробці

Процес створення сучасного WEB-ресурсу неможливо уявити без застосування різноманітних інструментів і сервісів, які значно спрощують розробку, підвищують якість кінцевого продукту і забезпечують ефективну взаємодію між учасниками проекту. Вибір таких інструментів залежить від специфіки проекту, складності задач і поставлених цілей.

Одним із ключових етапів розробки є створення дизайну інтерфейсу, який має бути зручним, інтуїтивно зрозумілим та привабливим для користувачів. Для цього широко використовуються сучасні графічні редактори та прототипувальники. Порівняння основних додатків для розробки WEB-ресурсу наведено у таблиці 1.3

Таблиця 1.3 - Порівняння основних додатків для розробки WEB-ресурсу

Інструмент	Роль у проєкті	Переваги	Сфера застосування
Webflow	CMS	Мінімум коду, хостинг, адаптивний дизайн	Лендінги, блоги, корпоративні сайти
WebStorm	IDE для JavaScript/Type Script	Підтримка фреймворків, автодоповнення, дебаггер, інтеграція з Git	Складні веб-додатки
Figma	Дизайн інтерфейсів (UI/UX)	Спільна робота, прототипування, плагіни	Дизайн макетів, UX-дослідження
WordPress	CMS	Велика спільнота, плагіни	Блоги, інтернет-магазини

Таблиця 1.3 - продовження

MySQL	Система управління базами даних	Надійність, SQL-оптимізація,	Зберігання данних, SaaS, вебдодатки
Jest	Тестування коду	Швидкість, інтуїтивний синтаксис, підтримка React/Vue	Юніт та інтеграційні тести
Netlify	Хостинг і CI/CD для статичних сайтів	Автоматичні збірки, HTTPS, інтеграція з GitHub	JAMstack, лендінги, вебдодатки
GitHub	хостинг коду та спільна робота	Git-репозиторії, Actions (CI/CD), Issues	Будь-які проєкти з командною розробкою

Figma — це хмарний інструмент для дизайну інтерфейсів, який дозволяє працювати над проектом одночасно кільком членам команди в режимі реального часу. Інтерфейс програми Figma наведено на рисунку 1.3. Figma підтримує створення інтерактивних прототипів [18], що допомагає одразу оцінити юзабіліті майбутнього ресурсу. Велика кількість плагінів і шаблонів робить цей інструмент дуже гнучким і популярним серед веб-дизайнерів .

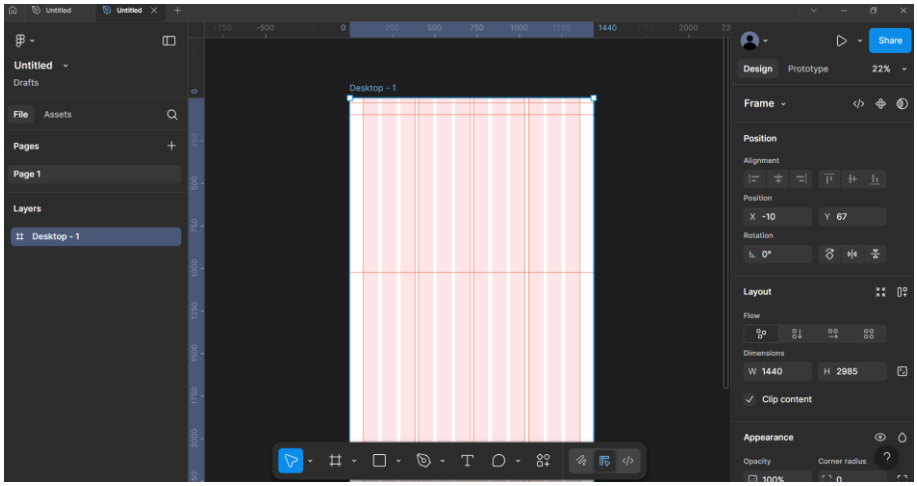


Рисунок 1.3. - Інтерфейс програми Figma

Adobe XD — це професійний інструмент від Adobe, орієнтований на створення прототипів та дизайну інтерфейсів з акцентом на інтерактивність та анімацію. Adobe XD підтримує інтеграцію з іншими продуктами Adobe, що робить його зручним для комплексної роботи над проектом. Приклад інтерфейсу програми Adobe XD наведений на рисунку 1.4.

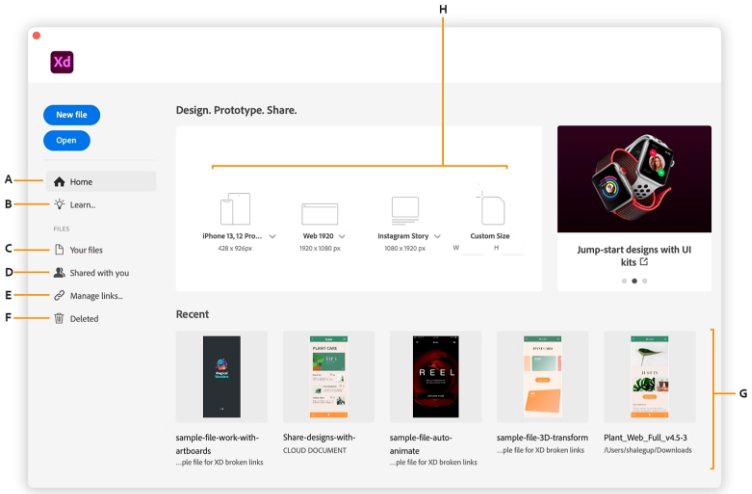


Рисунок 1.4 - Інтерфейс програми Adobe XD

Sketch — популярний дизайн-редактор, особливо серед користувачів macOS. Sketch пропонує зручний інтерфейс для створення макетів і прототипів, а також має широкий вибір плагінів для розширення функціоналу. Приклад інтерфейсу редактору наведений на рисунку 1.5.

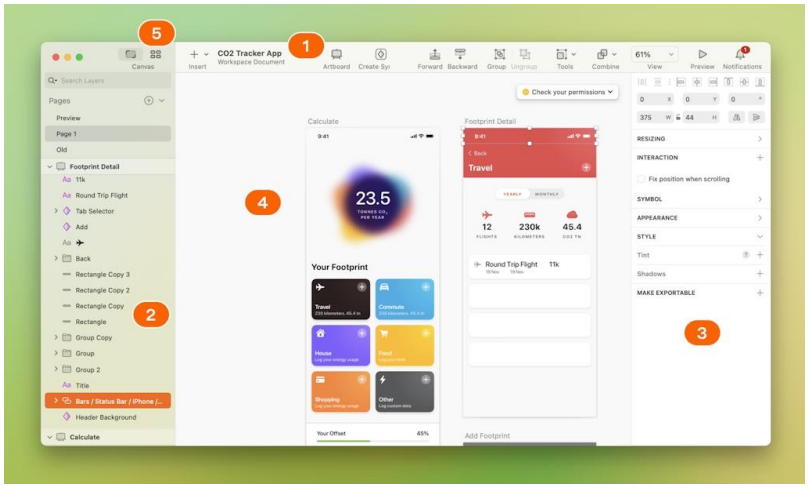


Рисунок 1.5 - Інтерфейс редактору Sketch

Для написання коду і реалізації функціональності сайту використовуються різні середовища розробки.

Visual Studio Code (VS Code) — це один із найпопулярніших безкоштовних редакторів коду, розроблений Microsoft. Він підтримує велику кількість мов програмування, має вбудований термінал, систему розширень і плагінів, що допомагають автоматизувати рутинні завдання. VS Code відомий своєю швидкодією і зручністю, що робить його улюбленим інструментом для фронтенд та бекенд розробників [11]. Приклад робочої зони програми наведено на рисунку 1.6.

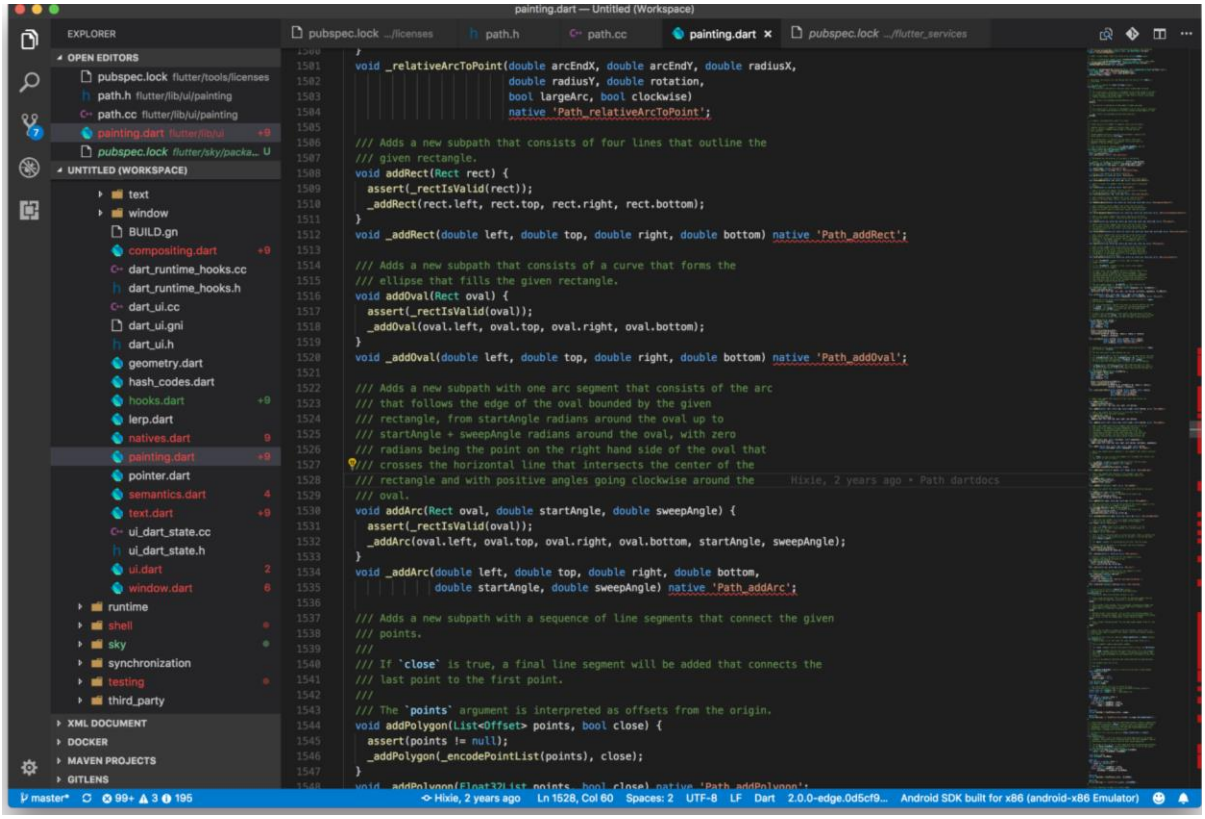


Рисунок 1.6 - Робоча зона програми Visual Studio Code

WebStorm — це комерційне середовище розробки від JetBrains, орієнтоване на JavaScript і пов’язані технології. Воно пропонує інтелектуальне автодоповнення коду, розширені можливості дебагінгу і інтеграцію з популярними фреймворками, що дозволяє підвищити продуктивність розробників. Приклад інтерфейсу WebStorm наведено на рисунку 1.7.

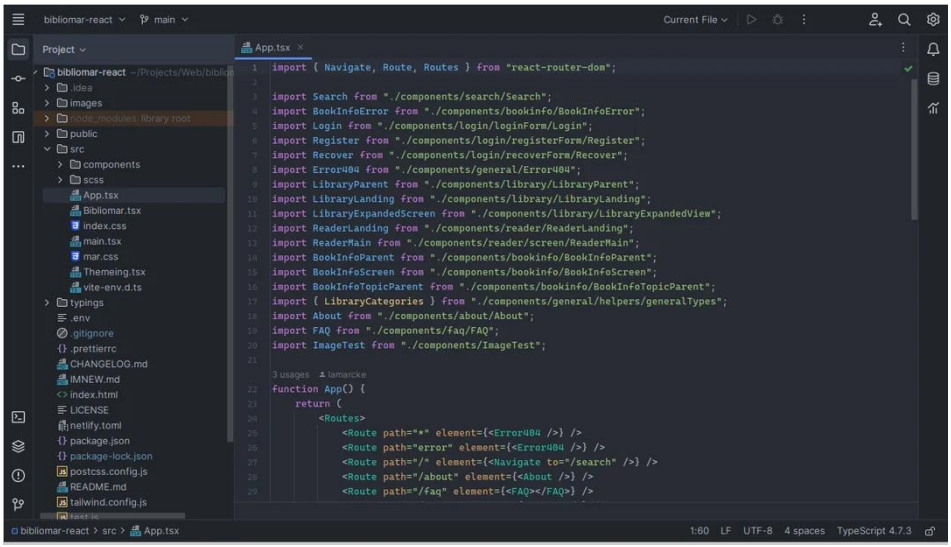


Рисунок 1.7 - Інтерфейс WebStorm

Для проєктів на CMS часто застосовують:

- WordPress — найбільш поширена система управління контентом, що дозволяє швидко створювати сайти різного рівня складності. Завдяки великій кількості тем та плагінів WordPress є дуже гнучким і масштабованим рішенням.

Однак він потребує регулярного оновлення і налаштування безпеки [7]. Приклад інтерфейсу WordPress наведено на рисунку 1.8.

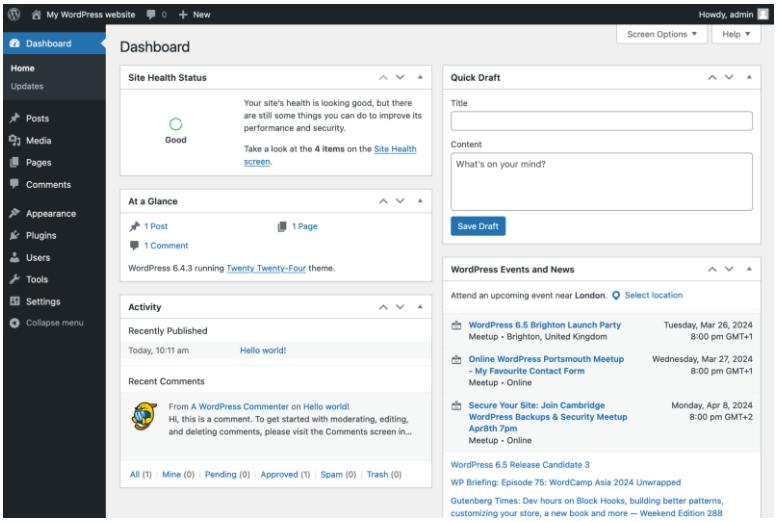


Рисунок 1.8 - Інтерфейс WordPress

Бази даних і серверні технології

Для зберігання і обробки інформації, що надходить від користувачів, використовуються різні системи управління базами даних [15].

- MySQL — це одна з найпопулярніших реляційних баз даних з відкритим кодом. Вона відома своєю стабільністю, масштабованістю і широкою підтримкою серед хостингів і веб-платформ.
- PostgreSQL — це потужна об’єктно-реляційна СУБД, що відрізняється високою надійністю, розширеними функціями і кращою підтримкою складних
- запитів. Часто використовується для проєктів, де потрібна велика гнучкість у роботі з даними.
- MongoDB — це документо-орієнтована база даних, яка зберігає дані у форматі JSON-подібних документів. Вона підходить для проектів з динамічною
- структурою даних та великим обсягом неструктурованої інформації [16].

Інструменти для тестування та контролю якості

Юніт-тестування проводяться з використанням сучасних інструментів, що рекомендуються для веб-розробки [8]. Проте для забезпечення стабільності і безпомилкової роботи ресурсу використовуються автоматизовані інструменти тестування:

- Jest — популярний фреймворк для тестування JavaScript-коду, який широко використовується для юніт-тестування компонентів і функцій.
- Cypress — сучасний інструмент для енд-ту-енд тестування, який дозволяє швидко і ефективно перевіряти функціональність WEB-додатків.

Хостинг і розгортання

Для розміщення розробленого ресурсу в мережі використовуються різні сервіси:

Netlify — сучасна платформа для розгортання фронтенд-додатків, що підтримує автоматичне оновлення сайту при зміні коду у репозиторії, має вбудовані функції для налаштування HTTPS, CDN і серверлес функцій.

- Vercel — хмарний сервіс, що спеціалізується на хостингу веб-додатків із підтримкою популярних фреймворків, таких як Next.js. Відзначається високою швидкістю доставки контенту і простою інтеграцією з Git.
- DigitalOcean — провайдер хмарних серверів (VPS), що дозволяє гнучко налаштовувати інфраструктуру і розгортати як прості, так і складні проекти, включаючи бекенд-сервіси.

Контроль версій

Особливо важливою складовою сучасної розробки є системи контролю версій:

- Git — розподілена система контролю версій, що дозволяє зберігати історію змін у коді, координувати роботу над проектом в команді та забезпечувати безпеку даних.
- GitHub та GitLab — популярні хостинги репозиторіїв Git із додатковими можливостями для управління проєктами, автоматизації CI/CD процесів, огляду коду і колаборації між розробниками.

Таким чином, використання сучасних інструментів і сервісів дає змогу підвищити ефективність розробки, покращити якість продукту і забезпечити комфортну взаємодію як розробників, так і кінцевих користувачів.

1.5 Висновок до розділу 1

У даному розділі було проаналізовано сучасні технології розробки WEB-ресурсів для взаємодії з клієнтами, проведено аналіз існуючих рішень, обрано платформу для реалізації проекту та визначено необхідні інструменти. Ці дані заклали основу для подальшої розробки власного ресурсу.

2 АНАЛІЗ ІСНУЮЧИХ АНАЛОГІВ WEB-РЕСУРСІВ ДЛЯ РОБОТИ З КЛІЄНТАМИ

2.1 Огляд аналогічних WEB-сайтів конкурентів

Перед початком розробки власного WEB-ресурсу **salamandravikna** було проаналізовано декілька сайтів конкурентів, які функціонують у тій самій або суміжній галузі. Такий аналіз є важливим етапом, оскільки дає змогу зрозуміти очікування користувачів, виявити сильні й слабкі сторони інших проєктів та врахувати це при побудові власного рішення, і потрібно врахувати що WEB-ресурс орієнтований на B2B-взаємодію, що вимагає особливої уваги до інтуїтивності інтерфейсу [9].

Зокрема, порівняння проводилося з двома сайтами — **"Вікна Престиж"** та **"Еко-Вікна"**. Обидва ресурси спеціалізуються на виготовленні та встановленні металопластикових вікон і мають функціонал для залучення клієнтів. Проте підхід до реалізації таких ресурсів суттєво відрізняється, як з погляду дизайну, так і за функціональністю.

Сайт **"Вікна Престиж"** справляє враження застарілого. Інтерфейс перевантажений текстовим контентом, навігація ускладнена, а форма зворотного зв’язку відсутня, що потенційно знижує кількість звернень від клієнтів. Мобільна версія сайту має суттєві візуальні помилки, а час завантаження сторінок залишає бажати кращого. Оформлення сторінки конкурента представлено на рисунках 2.1 та 2.2.

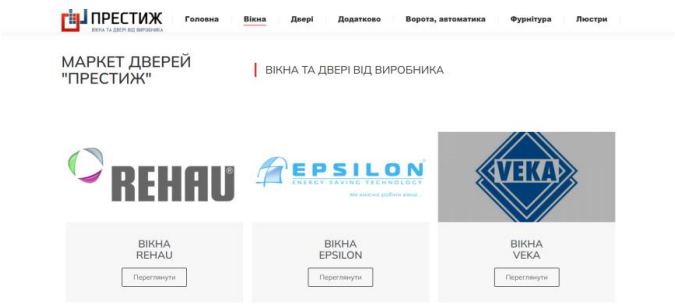


Рисунок 2.1 - Головна сторінка сайту

Рисунок 2.2 - Головна сторінка сайту (мобільна версія)

У випадку з сайтом **"Еко-Вікна"**, ситуація дещо краща, хоча й не без недоліків. Назва орієнтована на екологічність, що потенційно формує позитивне перше враження. Водночас інтерфейс досить примітивний, з відсутністю візуального стилю та графічної привабливості. На сайті присутня повноцінної форми зворотного зв'язку — але пропонуються лише телефон та ім'я, що не завжди зручно для сучасного користувача, бажаючого лишити коментар до звернення. Сайт частково адаптований під мобільні пристрої, однак є проблеми з верстkою на різних роздільних здатностях екранів. Форма, головна сторінка та мобільна версія сайту представлені на рисунках 2.3, 2.4 та 2.5

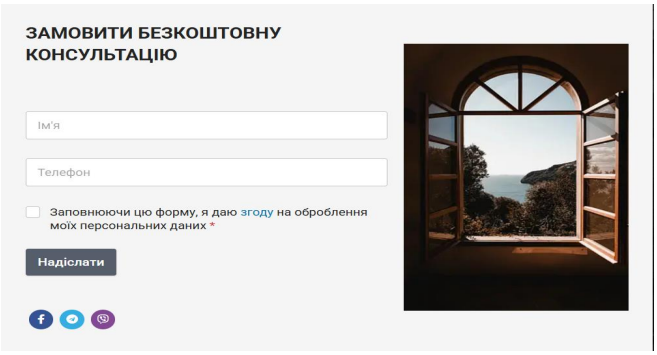


Рисунок 2.3 - Форма сайту "Еко-Вікна"

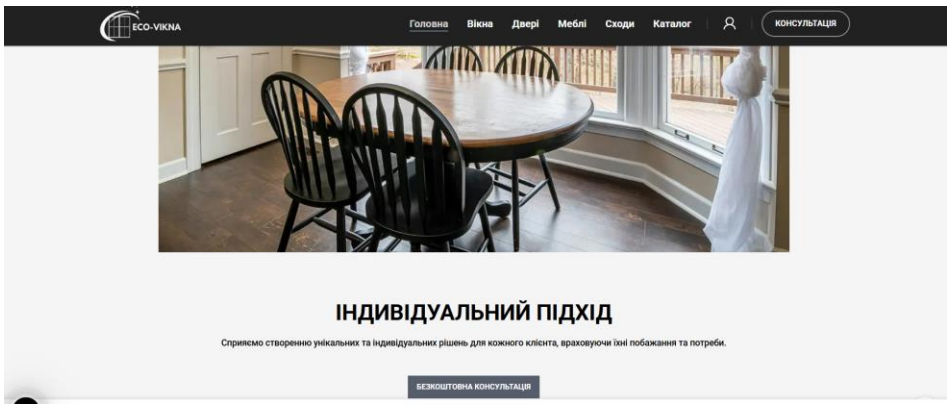


Рисунок 2.4 - Головна сторінка сайту “Еко-Вікна”

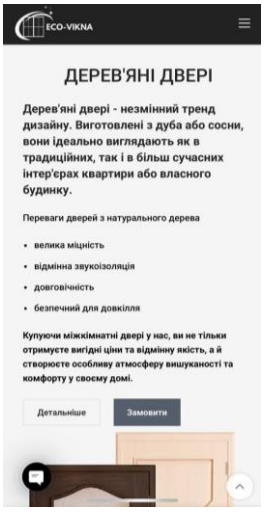


Рисунок 2.5 - Головна сторінка сайту “Еко-Вікна” (мобільна версія)

На цьому тлі сайт **Salamandra Vikna** демонструє значно вищий рівень реалізації. Його інтерфейс виглядає сучасно, структуровано та візуально привабливо(див. рис. 2.6). Велика увага приділялася зручності навігації, інтерактивності, а також адаптивності ресурсу — сайт чудово працює на всіх типах пристроїв, включаючи мобільні телефони та планшети (див. рис. 2.9). Крім того, форма зворотного зв’язку представлена у двох варіантах (див. рис. 2.7 і 2.8), реалізована у спрощеному, зручному для користувача вигляді та доповнена інтеграціями з популярними месенджерами: Telegram. Це дозволяє максимально швидко й ефективно комунікувати з клієнтами. Також важливою перевагою є використання анімацій, якісного візуального контенту та оптимізації швидкості завантаження.

Рисунок 2.6 - Головна сторінка сайту “salamandravikna”

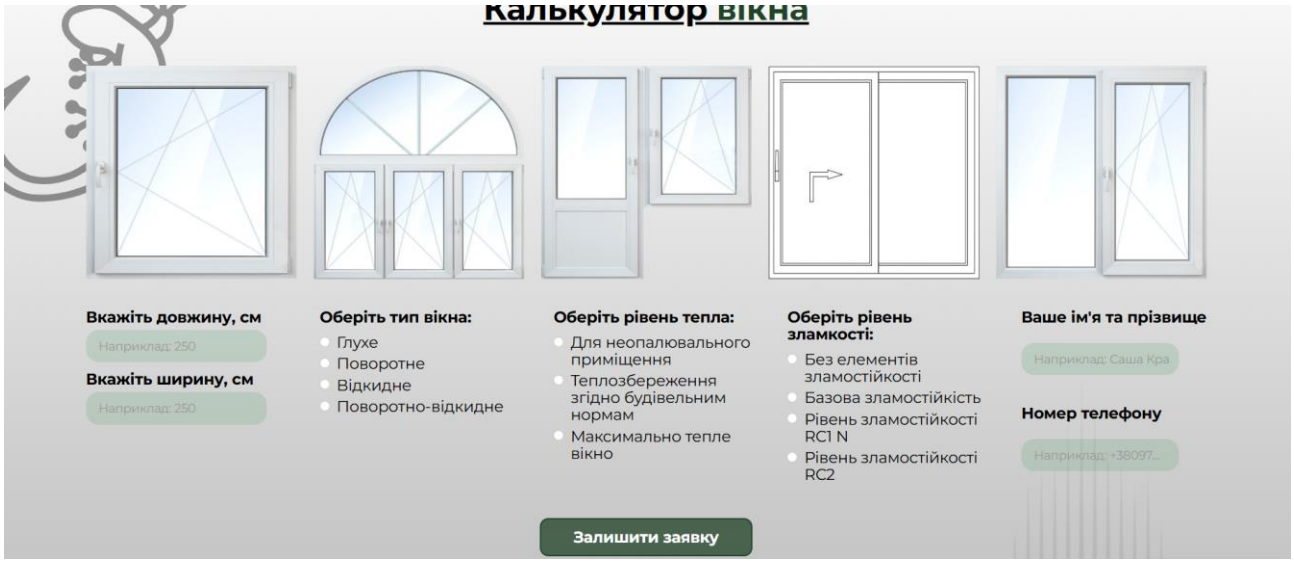


Рисунок 2.7 - Форма зворотного зв’язку 1

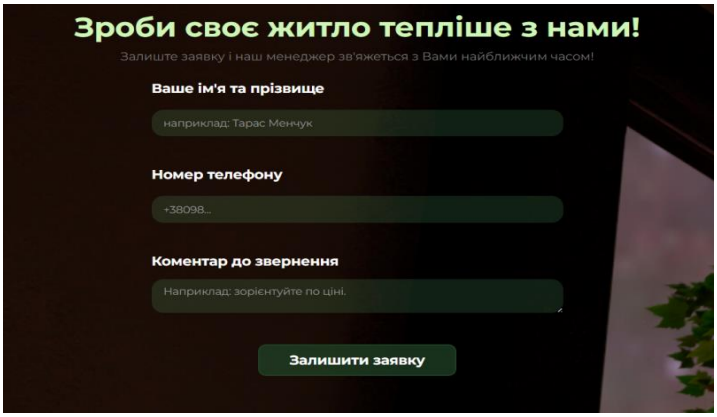


Рисунок 2.8 - Форма зворотного зв’язку 2

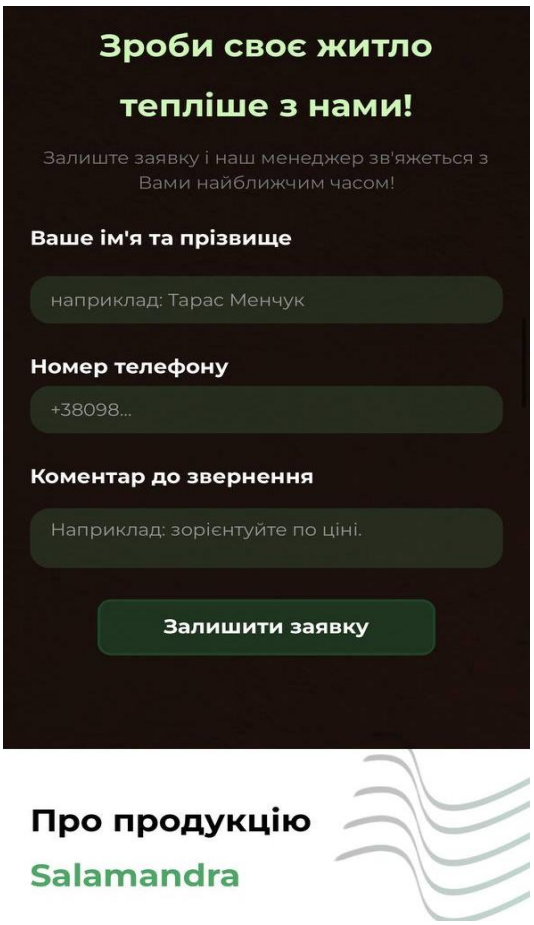


Рисунок 2.9 - Головна сторінка сайту “salamandravikna” (мобільна версія)

Під час аналізу було виявлено, що найбільш ефективні WEB-ресурси мають просту та логічну структуру, зрозумілу навігацію, швидко відкриваються на різних пристроях та мають зручний зворотній зв’язок, наприклад, через чат-ботів або інтеграцію з месенджерами, як-от Telegram чи Viber. Також важливою перевагою є особистий кабінет клієнта, наявність розділу з відгуками та можливість відстеження замовлень онлайн.

Проте часто можна зустріти і недоліки — перевантаженість вмістом, застарілий дизайн, відсутність адаптивної верстки або складна форма зворотного зв’язку. Це може суттєво знижувати конверсію сайту та рівень задоволеності клієнтів. Результати дослідження та порівняння сайтів-конкурентів описано у таблиці 2.1

Таблиця 2.1 - Дослідження та порівняння сайтів-конкурентів

Критерії	salamandravikna	Вікна Престиж	Еко-Вікна
Назва	Асоціативна	Звичайна	З натяком на екологічність, але звучить шаблонно
Зручність інтерфейсу	Інтуїтивний та сучасний дизайн	Застарілий дизайн	Без особливостей
Форма зворотного зв’язку	Є, зручна, швидка	Відсутня	Проста
Адаптивність	Працює коректно на усіх пристроях	Погано адаптована для мобільних пристроїв	Є помилки в адаптації
Швидкість завантаження	Швидко за рахунок адаптації	Середня швидкість	Середня швидкість
Інтеграція з месенджерами	Telegram і WhatsApp	Немає	Лише Viber
Додаткові переваги	Інтерактивний	Мало візуального контенту	Забагато тексту

У результаті аналізу було встановлено, що конкурентні ресурси мають ряд недоліків — від застарілого дизайну до відсутності базової інтерактивності. Це дало змогу краще спроектувати власний сайт з урахуванням виявлених слабких

сторін і забезпечити комфортну взаємодію з клієнтами на всіх рівнях.

2.2 Порівняльна характеристика та аналіз переваг і недоліків існуючих рішень

На основі аналізу WEB-ресурсів конкурентів, проведеного у попередньому підрозділі, можна зробити низку висновків щодо їхніх підходів до організації взаємодії з клієнтами, дизайну та функціональних можливостей. Цей аналіз є надзвичайно цінним у контексті розробки власного ресурсу, оскільки дозволяє не тільки запозичити вдалі рішення, а й уникнути помилок, допущених іншими компаніями.

Загалом, більшість сайтів конкурентів мають базову функціональність: вони надають загальну інформацію про продукцію, контактні дані та форму зворотного зв’язку. Однак лише поодинокі ресурси можуть похвалитися дійсно зручною адаптивністю, сучасним дизайном і якісною інтеграцією з месенджерами [19].

Зокрема, такі сайти, як "Вікна Престиж" чи "Еко-Вікна", не завжди забезпечують достатній рівень користувацького досвіду — часто інтерфейс перевантажений текстовою інформацією, навігація неінтуїтивна, а функції взаємодії з клієнтом обмежені стандартною формою зворотного зв’язку або телефонним дзвінком. Також відсутність чіткої адаптації до мобільних пристроїв робить такі ресурси менш зручними у використанні для сучасного користувача.

Навпаки, у процесі розробки сайту Salamandra Vikna було враховано потреби цільової аудиторії — перевагу швидкого зв’язку, мінімалізм у дизайні, а також швидке завантаження ресурсу. Сайт отримав сучасне оформлення, інтеграцію з Telegram та WhatsApp, а також логічно вибудувану структуру з адаптивною верстkою.

Такий підхід дозволив не тільки виділитися на фоні конкурентів, але й забезпечити високий рівень комфорту для користувача, що прямо впливає на ефективність залучення нових клієнтів.

У підсумку, аналіз конкурентного середовища дав змогу:

- окреслити ключові очікування аудиторії від сучасного WEB-ресурсу;
- сформувати список функцій, які необхідно було реалізувати;
- прийняти зважені рішення щодо дизайну, платформи і технічних рішень;
- уникнути помилок, які допускають інші компанії на ринку.

Таким чином, критичне оцінювання наявних рішень стало основою для створення більш досконалого й ефективного сайту, орієнтованого на користувача

2.3 Візуалізація архітектури системи за допомогою UML-діаграми

У процесі розробки та документування WEB-ресурсу важливою складовою стало проектування логічної структури системи, її компонентів і взаємодії між ними. Для досягнення цієї мети було використано мову моделювання UML (Unified Modeling Language), яка є стандартом у сфері об'єктно-орієнтованого проектування. UML забезпечує уніфікований підхід до візуалізації, специфікації, конструювання та документування елементів програмних систем, що робить її ефективним інструментом як для розробників, так і для аналітиків.

UML-діаграми дозволяють відобразити структуру програмного забезпечення у зручній графічній формі. Вони застосовуються на різних етапах життєвого циклу системи — від початкового аналізу до імплементації та супроводу. Їх використання сприяє кращому розумінню системи як з боку команди розробки, так і зацікавлених осіб, які не обов'язково мають технічну підготовку.

У межах реалізації даного WEB-ресурсу було створено дві UML-діаграми класів (див. рис. 2.10, 2.11), кожна з яких виконує окрему функцію у процесі проектування та демонстрації архітектури. Діаграми є взаємодоповнюючими: базова фокусується на загальній структурі та сутностях, тоді як розширена — на технічних інтеграціях та зовнішніх взаємозв’язках.

Рисунок 2.10 - Базова UML-діаграма класів

Ця діаграма є спрощеним уявленням логічної моделі даних, яка дозволяє зрозуміти основні бізнес-сутності, їхні атрибути та зв’язки між ними. Вона не деталізує реалізаційні аспекти, натомість акцентує увагу на концептуальних елементах системи, що особливо корисно на ранніх етапах проектування.

Зв’язки між класами також мають концептуальне навантаження. Наприклад, між користувачем та замовленням існує зв’язок типу «один до багатьох», що означає можливість кожного користувача створювати декілька замовлень. Водночас, між замовленням і вікнами встановлено зв’язок типу «багато до багатьох», оскільки одне замовлення може містити різні типи вікон, і водночас одне вікно може фігурувати у кількох замовленнях.

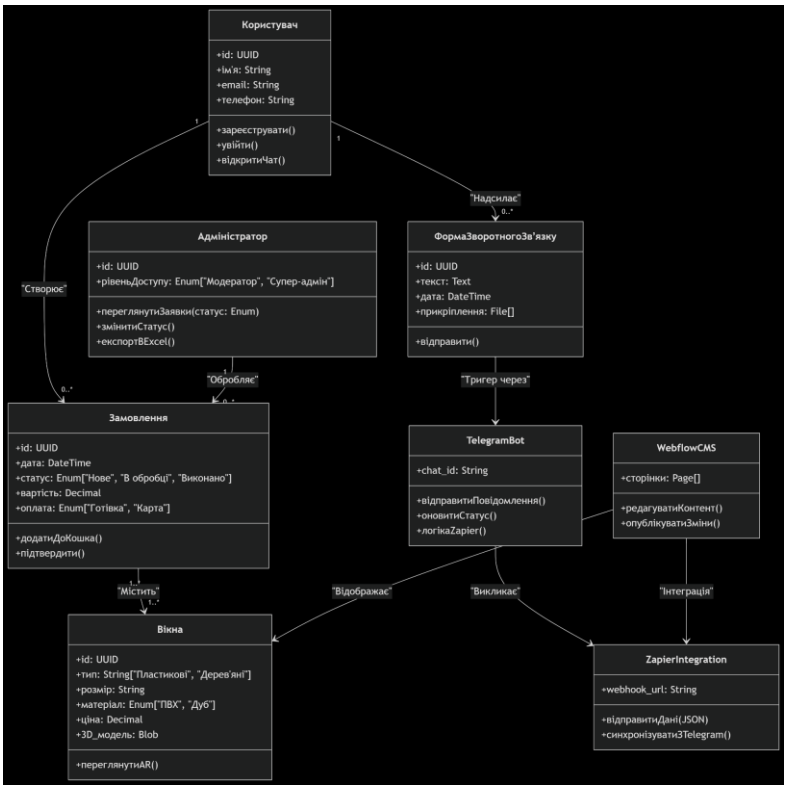


Рисунок 2.11 - Розширена UML-діаграма класів

У ході реалізації системи до неї було підключено низку зовнішніх сервісів, що потребувало побудови більш деталізованої діаграми класів, яка б відображала технічну архітектуру рішення. Така розширена UML-діаграма включає додаткові сутності, пов’язані з інтеграцією API, обробкою форм і динамічним управлінням контентом.

У діаграму було інтегровано такі класи:

- TelegramBot, який реалізує логіку автоматичних сповіщень адміністратора про надходження нових запитів через форму зворотного зв’язку. Має ключові атрибути, зокрема chat_id та метод логікаZapier(), що вказує на активацію бота через події у Zapier.
- ZapierIntegration — клас, що реалізує логіку взаємодії між веб-формами на сайті та зовнішніми сервісами. Його атрибути включають URL для вебхука (WEBhook_url) та логіку синхронізації вхідних даних.
- WebflowCMS — система управління контентом, яка дозволяє змінювати сторінки сайту без втручання у програмний код. Вона слугує інтерфейсом для наповнення сайту динамічним вмістом.

У цій моделі чітко відображено технічні взаємозв’язки. Наприклад, форма зворотного зв’язку надсилає дані через Zapier, що, у свою чергу, активує TelegramBot, забезпечуючи миттєве інформування адміністратора. Крім того, синхронізація між WebflowCMS та ZapierIntegration забезпечує автоматичне оновлення контенту на основі отриманих даних або змін у CMS.

2.4 Висновок до розділу 2

У результаті аналізу конкурентних WEB-ресурсів було виявлено переваги та недоліки існуючих аналогів. Це дозволило чітко сформулювати вимоги до власного сайту та створити основу для його ефективної розробки. Порівняльна характеристика підтвердила доцільність створення сучасного, адаптивного й зручного ресурсу, що відповідає очікуванням цільової аудиторії та вигідно вирізняється на тлі конкурентів.

Додатково, для кращого розуміння структури та логіки системи було використано UML-діаграми. Базова діаграма класів допомогла визначити ключові сутності та їх зв’язки, а розширена — відобразити взаємодію з зовнішніми сервісами, такими як Telegram, Zapier та Webflow. Це дозволило більш чітко спроєктувати архітектуру сайту та полегшило подальшу реалізацію.

3 ПРОЄКТУВАННЯ WEB-РЕСУРСУ

3.1 Обґрунтування вибору мови та середовища програмування

Під час розробки WEB-ресурсу для компанії Salamandra Vikna, основним пріоритетом було досягнення балансу між швидкістю реалізації, масштабованістю, продуктивністю і можливістю подальшого масштабування системи без залучення складних бекенд-інженерів. Враховуючи обмежені ресурси і потребу швидко вивести продукт на ринок, було прийнято рішення використати low-code платформу Webflow як фронтенд-движок проєкту.

Webflow забезпечує візуальне моделювання DOM-структури сторінок через drag-and-drop інтерфейс, генерує валідний, семантично правильний HTML5, CSS3, а також автоматично оптимізує JavaScript для інтерактивних елементів [20]. Використання CSS Flexbox і Grid систем у Webflow дозволило реалізувати адаптивний дизайн із підтримкою медіа-запитів (media queries) без необхідності писати кастомний код вручну (рис. 3.1). Це значно скоротило час на фронтенд-верстку та спростило подальшу підтримку UI.

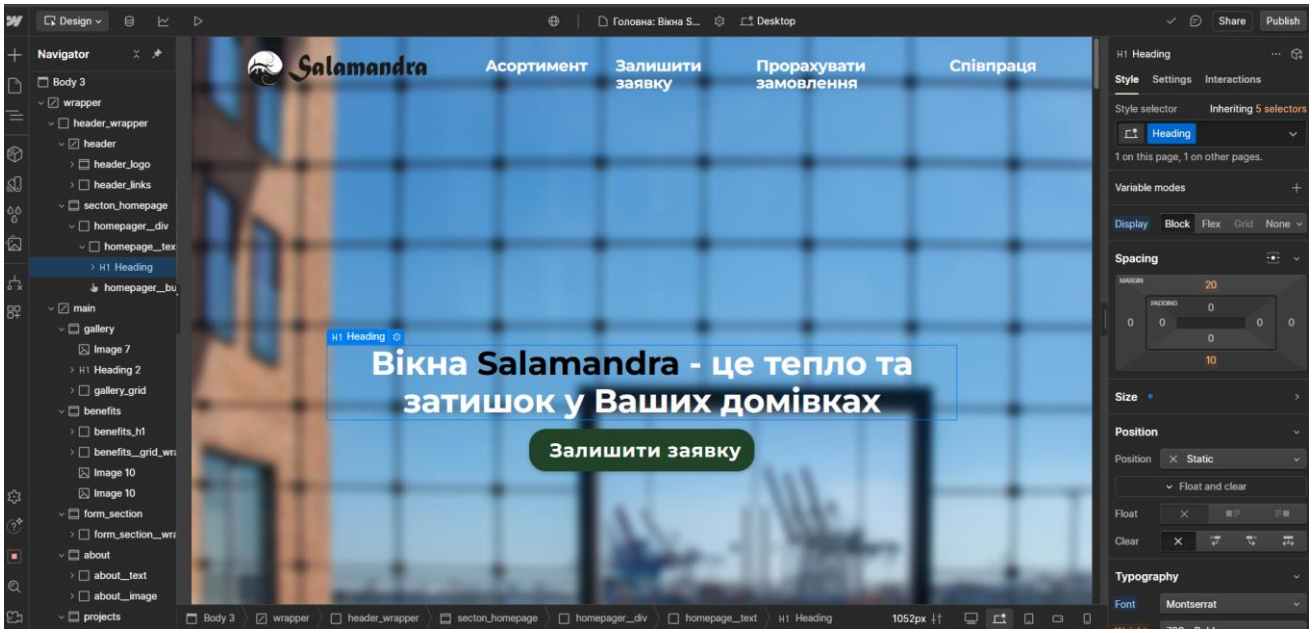


Рисунок 3.1 - Інтерфейс Webflow

Проте Webflow не має вбудованого реляційного або документно-орієнтованого бекенду, а отже не підтримує серверний код чи складну бізнес-логіку на стороні сервера. Враховуючи це, для реалізації функціоналу обробки заявок клієнтів і інтеграції з зовнішніми сервісами була обрана платформа автоматизації **Zapier** (рис. 3.2). Схема інтеграції плагіну продемонстровано на рисунку 3.3.

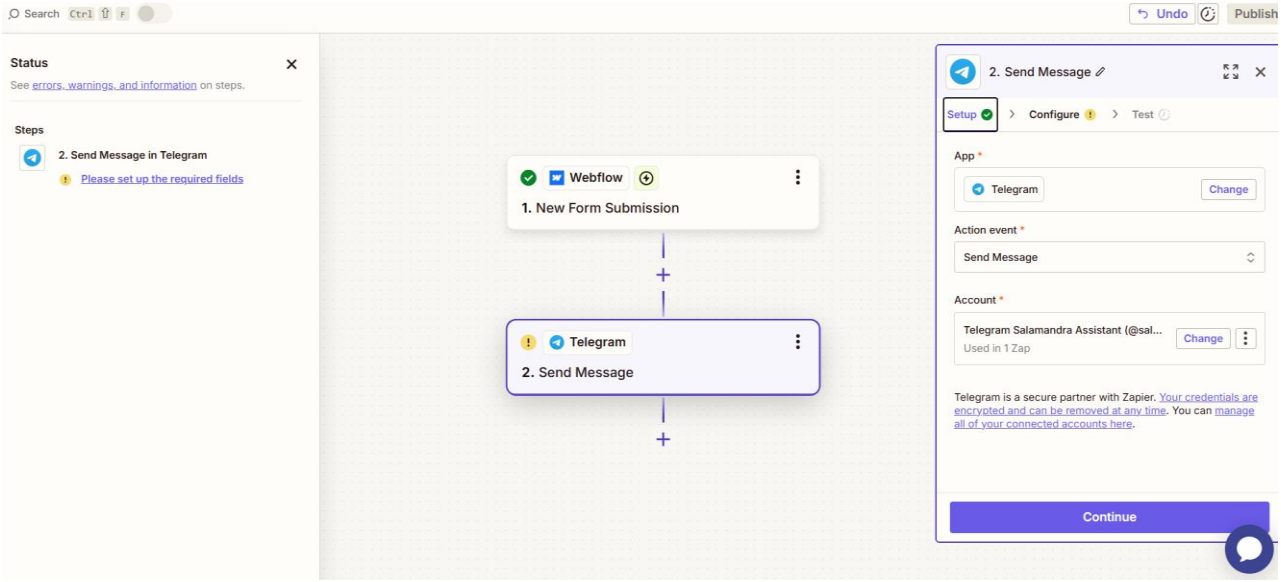


Рисунок 3.2 - Інтерфейс плагіну Zapier

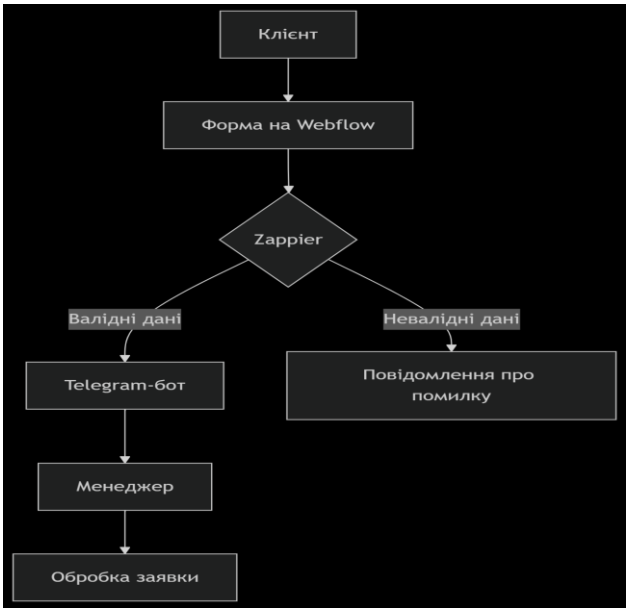


Рисунок 3.3 - Схема інтеграції плагіну Zapier

Zapier виступає як проміжний інтегратор (middleware) [5], що оркеструє workflow — послідовність подій і тригерів між вебформою на Webflow, хмарними сервісами для зберігання даних (наприклад, Google Sheets) і каналами оповіщення (Telegram, WhatsApp) (рис. 3.4).

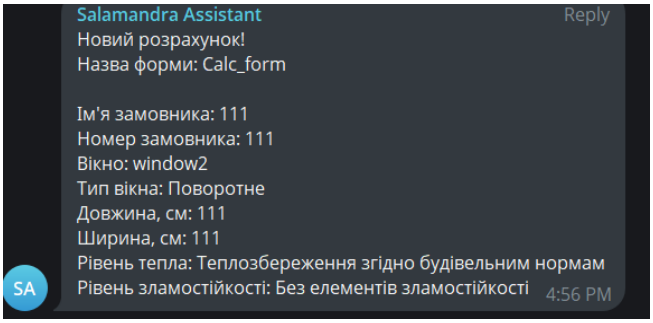


Рисунок 3.4 - Приклад заявки з сайту

Архітектурно, така архітектура відповідає патерну «serverless», коли відсутній класичний серверний бекенд, а функціонал делегується хмарним сервісам і API-зв’язкам. Це позбавляє проєкт від складності управління серверною інфраструктурою, налаштування баз даних або написання API-ендпоїнтів. Водночас, подібна архітектура масштабована, оскільки Zapier має високий рівень надійності та здатен обробляти тисячі подій на добу.

Для контрасту, класичні технології розробки — такі як стек LAMP (Linux, Apache, MySQL, PHP) чи Node.js з Express і MongoDB — передбачають повну розробку бекенд-сервісів, про що йшлось в таблиці 1.3, побудову RESTful або GraphQL API, моделювання бізнес-логіки на сервері, управління життєвим циклом даних у БД, а також написання модульних тестів і організацію CI/CD-процесів. Хоч це і дає максимум контролю, але вимагає значних часових та кадрових ресурсів.

Інші no-code платформи, як-от Wix або Squarespace, не надають гнучкості для тонкого налаштування інтеграцій із зовнішніми системами. WordPress при всій своїй популярності потребує регулярного оновлення плагінів, забезпечення безпеки та іноді ускладнює управління складною бізнес-логікою, що може призвести до проблем з продуктивністю.

Отже, використання Webflow у поєднанні із Zapier забезпечує наступні технічні переваги:

- Автоматизовану генерацію чистого, SEO-оптимізованого коду без "legacy" залежностей.
- Використання реактивного підходу у фронтенді з підтримкою адаптивності через CSS медіа-запити [3].
- Інтеграцію з API сторонніх сервісів за допомогою WEBhook-ів і REST API, оркестрованих Zapier.
- Мінімізацію ризиків, пов’язаних із безпекою та масштабуванням, завдяки serverless-архітектурі.
- Можливість швидкого внесення змін у UI без перекомпіляції коду, що підвищує гнучкість бізнесу.

3.2 Створення основних сторінок та функціональних блоків

Після вибору платформи для реалізації проекту наступним важливим кроком є безпосереднє створення основних розділів WEB-ресурсу та їх функціональних блоків. Цей етап передбачає не лише структурне формування інтерфейсу користувача, а й визначення логіки взаємодії між компонентами сайту для забезпечення комфортного та ефективного користування. Основні розділи, які були створені для даного WEB-ресурсу (рис. 3.5), включають:

- Головна сторінка — вітрина компанії, що містить основну інформацію про послуги, вигідні пропозиції та швидкий доступ до контактів.
- Розділ каталогу товарів або послуг — відображає перелік доступних пропозицій з можливістю фільтрації та сортування.
- Розділ деталей товару — містить розгорнуту інформацію про конкретний товар або послугу, технічні характеристики, фотографії та відгуки.
- Контактний розділ — включає форму зворотного зв’язку, інтеграцію з месенджерами (Telegram, WhatsApp)

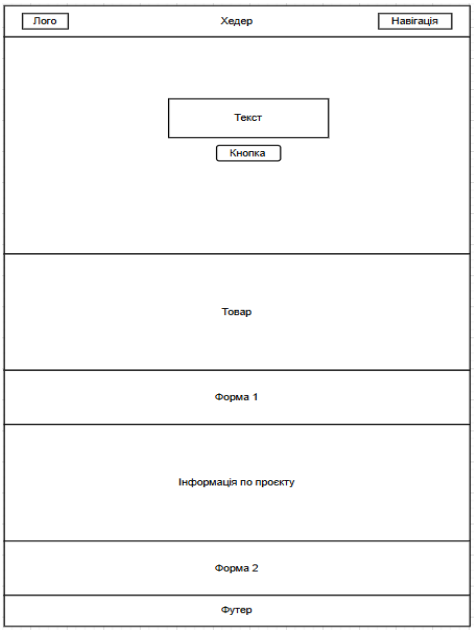


Рисунок 3.5 - Структура головної сторінки

Для створення цих розділів у Webflow було використано компоненти — це частково готові блоки коду або елементи дизайну, які можна повторно застосовувати. Вони забезпечують стандартизацію вигляду та спрощують подальшу підтримку сайту. Як до прикладу, анімації, які використовуються в кількох місцях одночасно, можна дублювати значення без втрати швидкості сайту.

3.3 Програмна реалізація та інтеграція із зовнішніми сервісами (Telegram-бот)

Однією з ключових особливостей сайту стало впровадження інтеграції з Telegram-ботом для забезпечення швидкого й ефективного інформування адміністратора про нові запити від клієнтів. Це рішення було обране як практичне й сучасне — воно дозволяє мінімізувати час очікування відповіді для клієнта та полегшує роботу з обробки звернень. Для безпечного обміну даними між клієнтом і сервером також часто використовуються CORS-політики, [6].

Програмна реалізація веб-ресурсу відбувалась за допомогою конструктора Webflow, що дозволяє створювати адаптивні вебсторінки без ручного написання

коду. Проте для реалізації функціональності зворотного зв’язку з клієнтом була додатково використана інтеграція через сервіс автоматизації Zapier та Telegram-бот, який отримує повідомлення про нові заявки.

Для цього на сайті була створена форма, яка містить поля імені, електронної пошти та повідомлення. Після натискання кнопки «Надіслати» відбувається перевірка правильності заповнення даних за допомогою функції валідації. У разі відсутності помилок, дані надсилаються до Zapier за допомогою HTTP-запиту формату POST. У випадку успішної відправки форму очищується, а користувач отримує повідомлення про успіх. У разі помилки, відображається відповідне

повідомлення про збій. Нижче наведено фрагмент реалізації даної логіки на JavaScript:

```
function validateError(formData) {
  const errors = [];

  if (!formData.name || formData.name.trim() === '') {
    errors.push("Будь ласка, введіть ім'я");
  }

  if (!formData.email || !/^\S+@\S+\.\S+$/.test(formData.email)) {
    errors.push("Будь ласка, введіть коректний email");
  }

  if (!formData.message || formData.message.trim() === '') {
    errors.push("Будь ласка, введіть повідомлення");
  }

  return errors;
}

async function sendToZapier(formData) {
    const zapierWebhookURL = 'https://hooks.zapier.com/hooks/catch/xxxxxx/yyyyyyy/';

  try {
    const response = await fetch(zapierWebhookURL, {
      method: 'POST',
      headers: {
        'Content-Type': 'application/json'
      },
```

```
      body: JSON.stringify(formData)
    });

    if (!response.ok) {
      throw new Error(`Помилка відправки: ${response.statusText}`);
    }

    return true;
  } catch (error) {
    console.error("Помилка при відправці до Zapier:", error);
    return false;
  }
}

async function handleSubmit(event) {
  event.preventDefault();

  const formData = {
    name: document.querySelector("#name").value,
    email: document.querySelector("#email").value,
    message: document.querySelector("#message").value
  };

  const validationErrors = validateError(formData);

  if (validationErrors.length > 0) {
    alert('Помилки:\n' + validationErrors.join('\n'));
    return;
  }
```

```
  const success = await sendToZapier(formData);

  if (success) {
    alert('Дякуємо! Ваше повідомлення відправлено.');
    event.target.reset();
  } else {
    alert('Сталася помилка при відправці. Спробуйте ще раз.');
  }
}

  document.querySelector("#contact-form").addEventListener('submit',
handleSubmit);
```

Завдяки використанню такого підходу, вебресурс забезпечує не лише зручну форму комунікації з клієнтами, а й високий рівень інтерактивності та автоматизації процесів.

Реалізація цієї інтеграції не потребувала створення серверної частини або написання бекенд-коду — завдяки використанню сервісу Zapier. Запуски автоматичних дій (так звані *Zap-и*) (див. рис. 3.6) дають змогу обробляти події з

вебсайту, зокрема — надсилання форми, і далі передавати дані у вигляді повідомлення до месенджера Telegram.

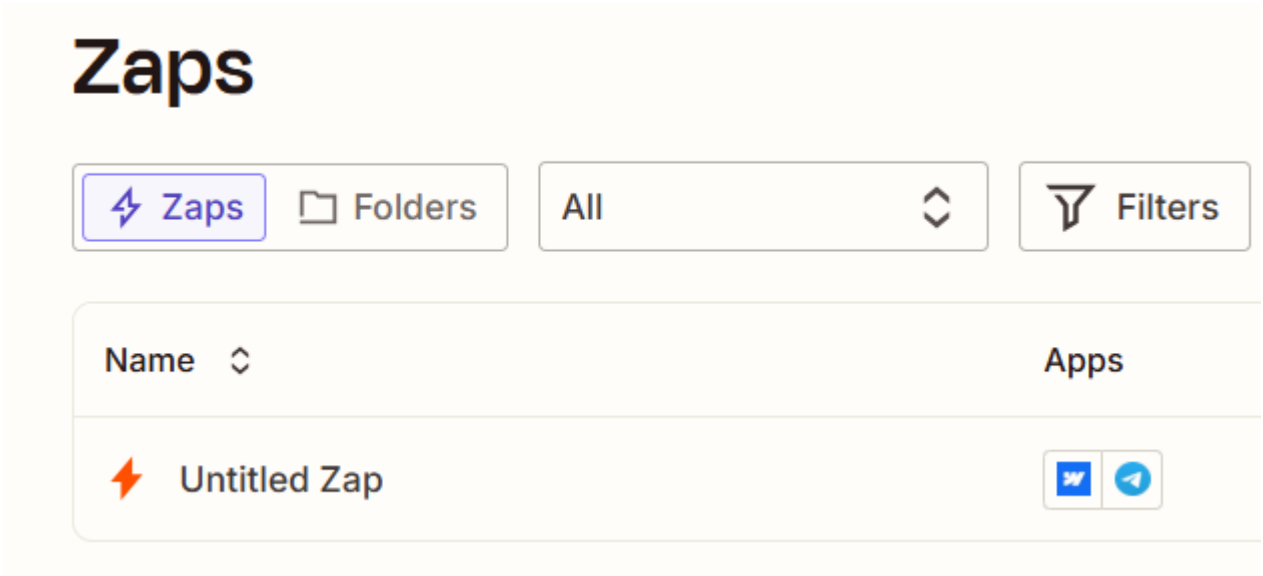


Рисунок 3.6 - Виконання тригеру

Процес налаштування починався зі створення Telegram-бота через сервіс BotFather (рис. 3.7), де було отримано унікальний токен. Далі, в Zapier, було створено ланцюжок дій: при кожному новому сабміті форми, дані з неї відправляються за допомогою WEBhook до платформи Zapier, яка, у свою чергу,

автоматично формує повідомлення і надсилає його до Telegram (рис. 3.4). У повідомленні міститься ім’я клієнта, його email, і зміст звернення.

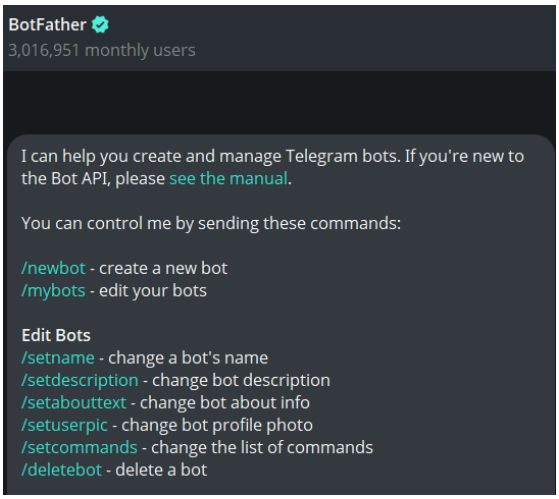


Рисунок 3.7 - Створення Телеграм-бота

З технічної сторони це стало можливим завдяки кастомному скрипту, доданому в налаштування Webflow. Саме цей скрипт після кліку на кнопку «Надіслати» ініціює POST-запит на потрібну адресу WEBhook, де тіло запиту включає дані форми у форматі JSON. Таким чином, сайт взаємодіє із Zapier як

frontend-додаток, не маючи потреби у серверній частині чи базі даних. Нижче наведено фрагмент JavaScript-коду, який відповідає за обробку форми:

```
document.querySelector('#contact-form').addEventListener('submit', async function(event) {
    event.preventDefault();

    const formData = {
        name: document.querySelector('#name').value,
        phone: document.querySelector('#phone').value,
        message: document.querySelector('#message').value
    };

    try {
        await fetch('https://hooks.zapier.com/hooks/catch/123456/abcde', {
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(formData)
        });
```

```
        alert("Дякуємо! Ваше повідомлення відправлено.");
    } catch(error) {
        console.error("Помилка при відправці форми:", error);
        alert("Щось пішло не так. Спробуйте пізніше.");
    }
});
```

У цьому фрагменті скрипт додає обробник події submit до HTML-форми з ідентифікатором contact-form. Після того як користувач натискає кнопку «Надіслати», скрипт збирає дані з полів імені, телефону та повідомлення. Після цього виконується асинхронний запит до зовнішнього сервісу (у даному випадку Zapier), який приймає передані дані через webhook-з'єднання.

Функція fetch() виконує відправку даних методом POST у форматі JSON. У разі успішного виконання з'являється повідомлення-підтвердження для користувача. Якщо ж виникає помилка — вона виводиться у консоль, а користувач бачить відповідне попередження.

Завдяки цьому механізму вебсайт отримує функціональність реального зворотного зв’язку, що є критично важливим для підтримки комунікації з потенційними клієнтами, а також підвищує ефективність обробки запитів у режимі онлайн.

Окрему увагу в процесі реалізації WEB-ресурсу було приділено валідації даних, які користувач вводить у форму зворотного зв’язку. Цей етап надзвичайно важливий, адже він дозволяє уникнути надсилання порожніх або некоректних полів, що, у свою чергу, запобігає помилкам на стороні сервера або втраті важливої інформації.

Задля цього було реалізовано логіку перевірки правильності введених даних на стороні клієнта, яка виконується ще до того, як інформація надсилається у

зовнішню систему (наприклад, у Zapier). Таке попереднє фільтрування вхідних даних забезпечує базовий рівень безпеки, а також підвищує зручність користування, дозволяючи уникнути ситуацій, коли користувачі випадково залишають поля порожніми або вводять неправильний формат електронної пошти.

Основу перевірки складає наступна функція:

```
function validateError(formData) {
  const errors = [];
  if (!formData.name || formData.name.trim() === '') {
    errors.push("Будь ласка, введіть ім'я");
  }
  if (!formData.email || !/^\S+@\S+\.\S+$/.test(formData.email)) {
    errors.push("Будь ласка, введіть коректний email");
  }

  if (!formData.message || formData.message.trim() === '') {
    errors.push("Будь ласка, введіть повідомлення");
  }

  return errors;
}
```

Функція validateError(formData) приймає на вхід об’єкт formData, який містить дані, введені користувачем у форму. В межах цієї функції ініціалізується масив errors, до якого поступово додаються повідомлення про помилки у разі, якщо певне поле не відповідає очікуваним критеріям.

Змінна formData.name перевіряється на наявність і на те, чи не складається вона лише з пробілів. Якщо ім’я не введено — у масив помилок додається

відповідне повідомлення. Аналогічна логіка використовується і для formData.message.

Особливу увагу приділено валідації електронної пошти. Перевірка здійснюється за допомогою регулярного виразу /^\S+@\S+\.\S+$/, який дозволяє переконатися, що рядок містить символ @ та домен, тобто є схожим на реальну e-mail адресу. Якщо рядок не проходить перевірку, у список помилок додається повідомлення "Будь ласка, введіть коректний email".

В результаті, функція повертає масив з усіма виявленими помилками, які потім можуть бути виведені на екран для зручності користувача.

Ось приклад, як ця функція інтегрується у процес відправки даних:

```
const validationErrors = validateError(formData);
if (validationErrors.length > 0) {
  alert('Помилки:\n' + validationErrors.join('\n'));
  return;
}
```

У цьому фрагменті, перед відправкою запиту, здійснюється перевірка: якщо масив помилок не порожній, користувачу виводиться список помилок за допомогою alert, і подальше надсилання переривається. Це дозволяє усунути проблему ще до взаємодії із зовнішніми системами.

З точки зору користувача, ці функції повністю невидимі, але адміністративна частина сайту отримує значну перевагу — можливість обробки нових заявок майже

миттєво, без потреби постійно перевіряти пошту. До того ж, Telegram як канал комунікації є зручним і мобільним, особливо для локального бізнесу.

Цей підхід добре демонструє гнучкість сучасної розробки — завдяки комбінації Webflow і Zapier ми отримали повноцінну інтеграцію з мінімальними витратами часу й без необхідності розгортати сервер або створювати API з нуля.

3.4 Тестування та аналіз отриманих результатів

Для оцінки ефективності розробленого веб-сайту було проведено комплексне тестування, яке включало функціональну перевірку, юзабіліті-аналіз, а також порівняльний аналіз з сайтами конкурентів.

Під час функціонального тестування перевірялася коректність роботи основних елементів сайту, таких як навігація, інтерактивні кнопки, форми зворотного зв’язку, а також адаптивність на різних пристроях (десктоп, планшет, мобільний телефон). Тестування підтвердило стабільність і швидкість завантаження сторінок, відсутність технічних помилок (404, 500 тощо) та коректну роботу всіх інтерактивних елементів.

Юзабіліті-тестування проводилося за участю цільової аудиторії. Користувачі відзначили інтуїтивно зрозумілий інтерфейс, логічну структуру контенту та зручну навігацію. Відповідно до зворотного зв’язку було виявлено незначні побажання щодо покращення контрастності тексту для кращої читабельності на мобільних пристроях, що вже враховано у подальших оновленнях.

Порівняльний аналіз із сайтами основних конкурентів виявив низку недоліків у останніх, з чим можна ознайомитись в таблиці 3.1.

Таблиця 3.1 - Порівняльний аналіз із сайтами конкурентів

Характеристики	salamandra vikna	Вікна Престиж	Еко-Вікна
Адаптивний дизайн	+	+	+
Галерея продукції	+	+	+
Форма зв’язку	+	+	+
Інтеграція з месенджерами	+	–	–
Онлайн-калькулятор	+	–	–

Протягом аналізу були встановлені такі недоліки:

- Ускладнений пошук необхідної інформації через неможливість дізнатись ціну;
- Відсутність інтеграції з месенджерами, що ускладнює роботу менеджерів та несе за собою втрату клієнтів.

Розроблений сайт успішно усунув ці недоліки, забезпечивши сучасний дизайн, високу адаптивність, покращену навігацію та інтеграцію. Це дозволило підвищити загальний рівень задоволеності користувачів, менеджерів та конкурентоспроможність підприємства в інтернет-просторі.

3.5 Висновок до розділу 3

У межах третього розділу було реалізовано технічну частину WEB-ресурсу, що є важливою складовою проєкту. Основна увага приділялась розробці інтерактивного функціоналу, зокрема реалізації форми зворотного зв’язку з валідацією основних полів — ім’я, email, повідомлення або телефон. Це дозволило

забезпечити попередній контроль введених даних, підвищуючи достовірність отриманої інформації.

Однією з ключових переваг реалізації стало підключення до платформи Zapier за допомогою Webhooks, що дало змогу автоматизувати обробку запитів користувачів без потреби у складних серверних рішеннях. Застосування сучасного підходу до роботи з API (метод fetch) забезпечило надійну та швидку передачу даних до зовнішніх сервісів.

Окрему увагу було приділено покращенню взаємодії користувача з інтерфейсом. Реалізація плавної анімації через JavaScript зробила відображення контенту більш динамічним, що сприяє позитивному враженню від користування сайтом. Крім того, механізм обробки помилок дає змогу повідомляти користувача про можливі технічні збої, що підвищує рівень прозорості та довіри до ресурсу.

ВИСНОВКИ

У роботі було проведено комплексний аналіз сучасних технологій розробки WEB-ресурсів для взаємодії з клієнтами, зокрема розглянуто основні підходи, такі як використання конструкторів сайтів (наприклад, Webflow) та традиційні методи програмування. Дослідження показало, що сучасні low-code платформи можуть значно прискорити розробку, забезпечити високу адаптивність та інтеграцію зі зовнішніми сервісами, що є ключовим для ефективної роботи з клієнтськими запитами. На основі порівняльного аналізу різних інструментів було обґрунтовано вибір Webflow як основної платформи для реалізації проекту.

Окрему увагу приділено аналізу конкурентного середовища: проведено детальний огляд існуючих WEB-ресурсів компаній із суміжних галузей. Було досліджено переваги та недоліки аналогічних рішень з точки зору дизайну, функціональності, зручності користування, швидкості завантаження та інтеграції з месенджерами. Це дозволило визначити найкращі практики, а також виявити слабкі місця, які можна покращити у власному рішенні. Отримані висновки сформулювали чіткі рекомендації щодо створення більш ефективного та інтуїтивно зрозумілого WEB-ресурсу, орієнтованого на потреби клієнтів.

У практичній частині роботи виконано розробку структури WEB-ресурсу для взаємодії з клієнтами, включаючи обґрунтування вибору технологічного стеку (Webflow), проектування алгоритму обробки запитів, створення архітектури сайту та логіки його роботи. Особливий акцент зроблено на розробці основних сторінок і функціональних блоків, які забезпечують зручний інтерфейс для користувачів та ефективне управління інформацією. Додатково реалізовано інтеграцію із зовнішніми сервісами, зокрема Telegram-ботом для автоматизації обробки клієнтських запитів, що підвищує швидкість і якість обслуговування. Використання low-code підходу дозволило значно скоротити час розробки, знизити витрати та забезпечити гнучкість проекту для подальшого масштабування.

Отже, всі задачі виконано, мету дослідження досягнуто.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Томашевський О. В., Рисіков В. П. Комп’ютерні технології статистичної обробки даних : навч. посіб. Запоріжжя : ЗНТУ, 2015. 175 с.
2. Цеслів О. В. Коломієць А. С. Технологія проектування та адміністрування баз даних та сховищ даних : навч. посіб. Київ : КПІ ім. І. Сікорського, 2017. 290 с.
3. Кадомський К. К., Ніколюк П. К. Java. Теорія і практика : навч. посіб. Вінниця : Донну, 2019. 197 с.
4. Гриньків Б. В., Богач І. В. Розробка веб-платформи для автоматизації процесів купівлі-продажу. / Науковотехнічна конференція факультету інформаційних технологій та комп'ютерної інженерії, Вінниця: ВНТУ, 2023. 3 с.
5. Гурлі Д. HTTP: повний посібник. / Д. Гурлі – Массачусетс: О’Рейлі Медіа 2002. – 656 с.
6. CORS. [Електронний ресурс]. Режим доступу: https://developer.mozilla.org/en/docs/WEB/HTTP/CORS/
7. Ляхов О.Л. Методи тестування і оцінки якості програмного забезпечення. / О. Л. Ляхов, О.О. Бородіна – Полтава: ПолтНТУ, 2015. – 372 с.
8. WEBsite Testing. [Електронний ресурс]. Режим доступу: https://test.io/coverage/WEBsite-testing/
9. B2B (business to business). 2024 URL: https://www.techtarget.com/searchcio/definition/B2B (дата звернення: 23.05.2025).
10. АНАЛІЗ КОНКУРЕНТІВ ВАШОГО ДОДАТКУ. 2022. URL: https://asomobile.net/uk/blog/analiz-konkurentiv-vashogo-dodatkunovi-instrumenti-vid-asomobile/ (дата звернення 23.05.2025).
11. Advantages and disadvantages of JavaScript URL: https://www.geeksforgeeks.org/advantages-and-disadvantages-of-javascript/ (дата звернення: 23.05.2025).
12. React documentation URL: https://react.dev/reference/react (дата звернення: 23.05.2025).

13. Vue.js documentation URL: https://vuejs.org/guide/introduction.html (дата звернення: 23.05.2025).
14. Angular documentation URL: https://v17.angular.io/guide/architecture (дата звернення: 23.05.2025).
15. Node.js documentation URL: https://nodejs.org/docs/latest/api/ (дата звернення: 23.05.2025).
16. MongoDB documentation URL: https://www.mongodb.com/docs/ (дата звернення: 23.05.2025).
17. Методичні вказівки до виконання бакалаврських кваліфікаційних робіт для студентів спеціальності 122 «Комп'ютерні науки» [Електронний ресурс] / уклад.: А. А. Яровий, О. В. Сілагін, І. В. Богач. – Вінниця : ВНТУ, 2023. – (PDF, 54 с.).
18. Документація HTML і CSS [Електронний ресурс]. – Режим доступу: https://developer.mozilla.org/en-US/docs/WEB/HTML
19. CRM-системи: Salesforce [Електронний ресурс]. – Режим доступу: https://www.salesforce.com/
20. Webflow Documentation [Електронний ресурс]. – Режим доступу: https://Webflow.com/

ДОДАТКИ

ДОДАТОК А (ОБОВ'ЯЗКОВИЙ)
ПРОТОКОЛ ПЕРЕВІРКИ КВАЛІФІКАЦІЙНОЇ РОБОТИ

Назва роботи: Розробка WEB-ресурсу для роботи з клієнтами

Тип роботи: бакалаврська кваліфікаційна робота
(бакалаврська кваліфікаційна робота / магістерська кваліфікаційна робота)

Підрозділ кафедра комп'ютерних наук
(кафедра, факультет, навчальна група)

Коефіцієнт подібності текстових запозичень, виявлених у роботі системою StrikePlagiarism 1,09 %

Висновок щодо перевірки кваліфікаційної роботи (відмітити потрібне)

✔ Запозичення, виявлені у роботі, є законними і не містять ознак плагіату, фабрикації, фальсифікації. Роботу прийняти до захисту

- У роботі не виявлено ознак плагіату, фабрикації, фальсифікації, але надмірна кількість текстових запозичень та/або наявність типових розрахунків не дозволяють прийняти рішення про оригінальність та самостійність її виконання. Роботу направити на доопрацювання.

- У роботі виявлено ознаки плагіату та/або текстових маніпуляцій як спроб укриття плагіату, фабрикації, фальсифікації, що суперечить вимогам законодавства та нормам академічної доброчесності. Робота до захисту не приймається.

Експертна комісія:

Яровий А.А., зав. каф. КН _____
(прізвище, ініціали, посада) (підпис)

Колесницький О.К., проф. каф КН _____
(прізвище, ініціали, посада) (підпис)

Особа, відповідальна за перевірку _____ Озеранський В.С.
(підпис) (прізвище, ініціали)

З висновком експертної комісії ознайомлений(-на)

Керівник _____ Озеранський В.С.
(підпис) (прізвище, ініціали, посада)

Здобувач _____ Шевченко О.О.
(підпис) (прізвище, ініціали)

ДОДАТОК Б (ОБОВ'ЯЗКОВИЙ)
ЛІСТИНГ ПРОГРАМИ

```
function validateError(formData) {
    const errors = [];

    if (!formData.name || formData.name.trim() === '') {
      errors.push("Будь ласка, введіть ім'я");
    }

    if (!formData.email || !/^\S+@\S+\.\S+$/.test(formData.email)) {
      errors.push("Будь ласка, введіть коректний email");
    }

    if (!formData.message || formData.message.trim() === '') {
      errors.push("Будь ласка, введіть повідомлення");
    }

    return errors;
  }

  async function sendToZapier(formData) {
      const zapierWebhookURL = 'https://hooks.zapier.com/hooks/catch/xxxxxx/yyyyyyy/';

    try {
      const response = await fetch(zapierWebhookURL, {
        method: 'POST',
        headers: {
          'Content-Type': 'application/json'
```

```
      },
      body: JSON.stringify(formData)
    });

    if (!response.ok) {
      throw new Error(`Помилка відправки: ${response.statusText}`);
    }

    return true;
  } catch (error) {
    console.error("Помилка при відправці до Zapier:", error);
    return false;
  }
}

async function handleSubmit(event) {
  event.preventDefault();

  const formData = {
    name: document.querySelector("#name").value,
    email: document.querySelector("#email").value,
    message: document.querySelector("#message").value
  };

  const validationErrors = validateError(formData);

  if (validationErrors.length > 0) {
    alert('Помилки:\n' + validationErrors.join('\n'));
    return;
  }
```

```
  const success = await sendToZapier(formData);

  if (success) {
    alert('Дякуємо! Ваше повідомлення відправлено.');
    event.target.reset();
  } else {
    alert('Сталася помилка при відправці. Спробуйте ще раз.');
  }
}

  document.querySelector("#contact-form").addEventListener('submit',
handleSubmit);

document.querySelector('#contact-form').addEventListener('submit', async
function(event) {

  event.preventDefault();


  const formData = {

    name: document.querySelector('#name').value,

    phone: document.querySelector('#phone').value,

    message: document.querySelector('#message').value

  };


  try {

    await fetch('https://hooks.zapier.com/hooks/catch/123456/abcde', {
```

```
            method: 'POST',
            headers: {
                'Content-Type': 'application/json'
            },
            body: JSON.stringify(formData)
        });

        alert("Дякуємо! Ваше повідомлення відправлено.");
    } catch(error) {
        console.error("Помилка при відправці форми:", error);
        alert("Щось пішло не так. Спробуйте пізніше.");
    }
});

function validateError(formData) {
  const errors = [];
  if (!formData.name || formData.name.trim() === '') {
    errors.push("Будь ласка, введіть ім'я");
  }
  if (!formData.email || !/^\S+@\S+\.\S+$/.test(formData.email)) {
    errors.push("Будь ласка, введіть коректний email");
  }

  if (!formData.message || formData.message.trim() === '') {
    errors.push("Будь ласка, введіть повідомлення");
```

```
  }

  return errors;
}
const validationErrors = validateError(formData);
if (validationErrors.length > 0) {
  alert('Помилки:\n' + validationErrors.join('\n'));
  return;
}
```

ДОДАТОК В (ОБОВ'ЯЗКОВИЙ)

ІЛЮСТРАТИВНА ЧАСТИНА

«РОЗРОБКА WEB-РЕСУРСУ ДЛЯ РОБОТИ З КЛІЄНТАМИ»

Виконала: студентка 4 курсу, групи 3КН-21б
спеціальності 122 Комп'ютерні науки
(шифр і назва напряму підготовки, спеціальності)

Шевченко О.О.
(прізвище та ініціали)

Керівник: доцент каф. КН

Озеранський В.С.
(прізвище та ініціали)

«3» червня 2025 р.

Вінниця ВНТУ – 2025 рік

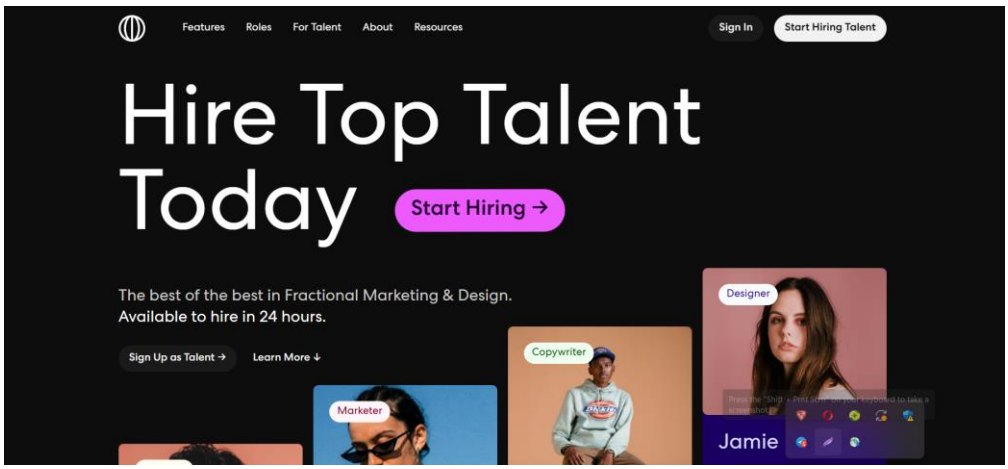


Рисунок В.1 - приклад сайту створений у Webflow

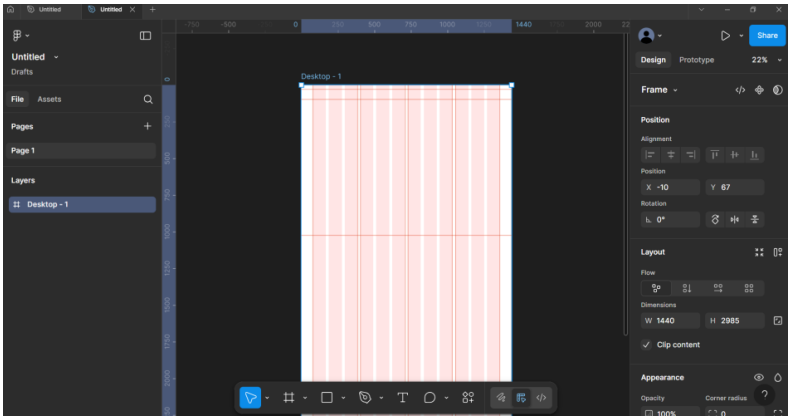


Рисунок В.2. - Інтерфейс програми Figma

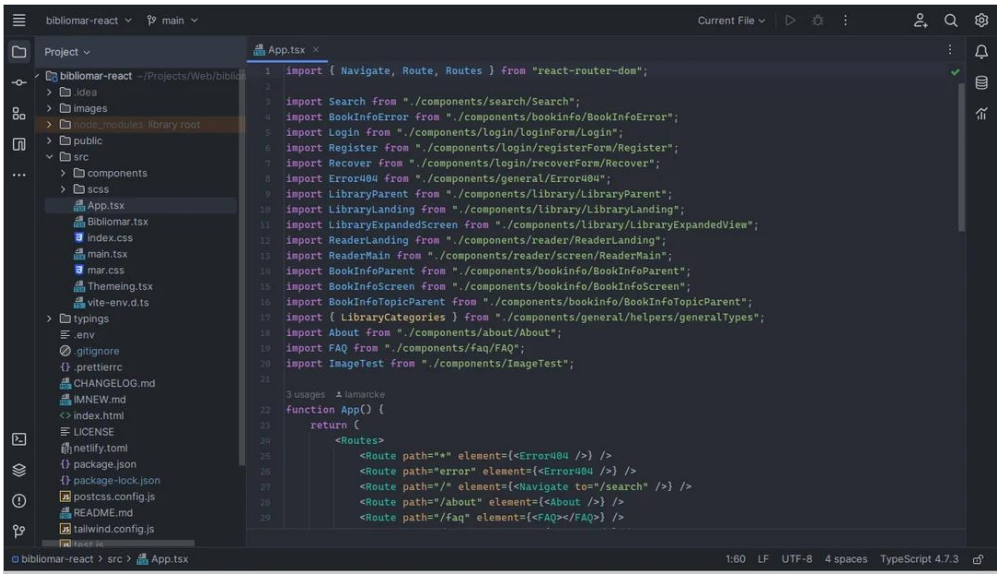


Рисунок В.7 - Інтерфейс WebStorm

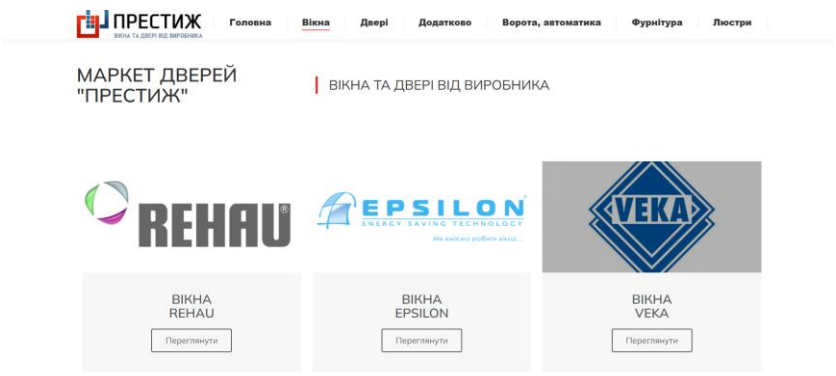


Рисунок В.9 - Головна сторінка сайту

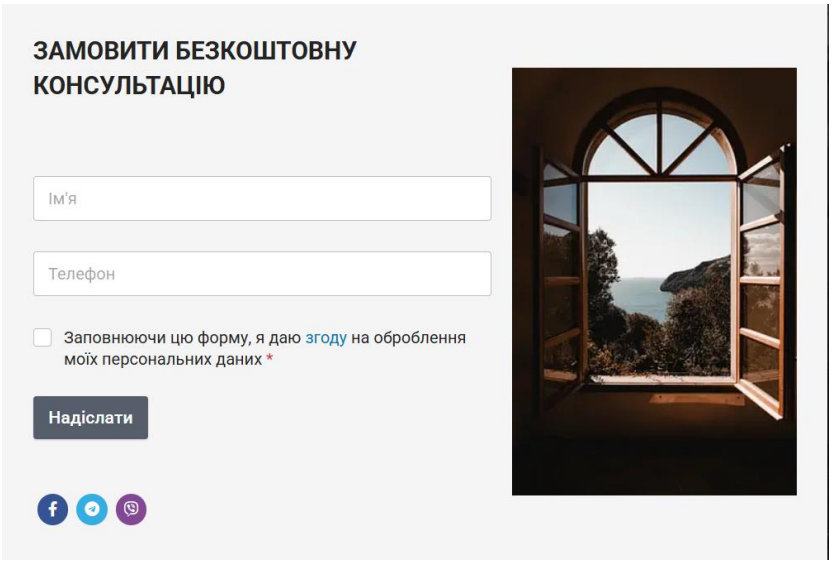


Рисунок В.11 - Форма сайту “Еко-Вікна”

Рисунок В.14 - Головна сторінка сайту “salamandravikna”

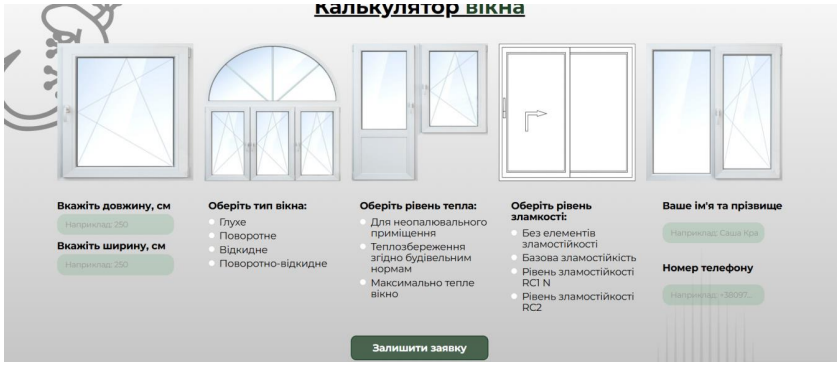


Рисунок В.15 - Форма зворотного зв’язку 1

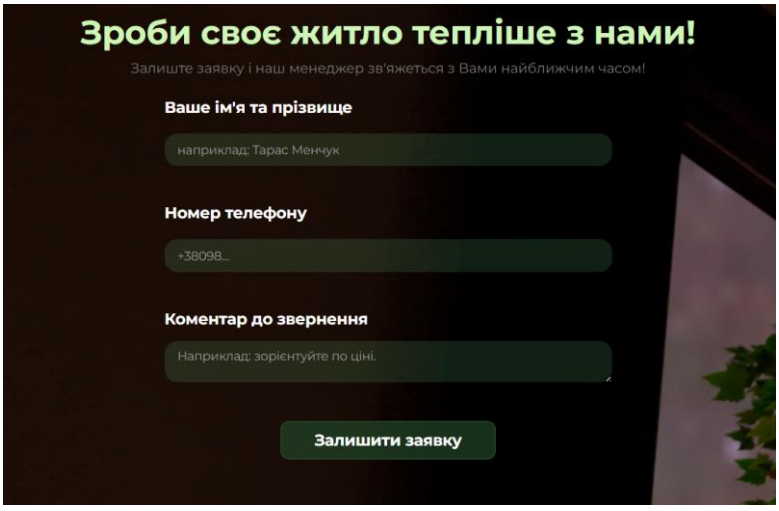


Рисунок В.16 - Форма зворотного зв’язку 2

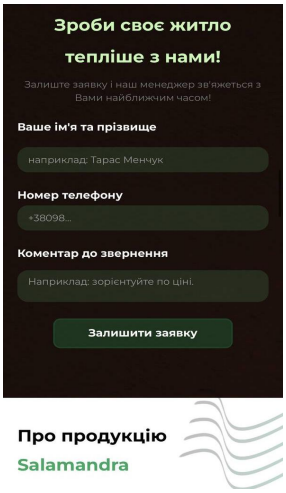


Рисунок В.17 - Головна сторінка сайту “salamandravikna” (мобільна версія)

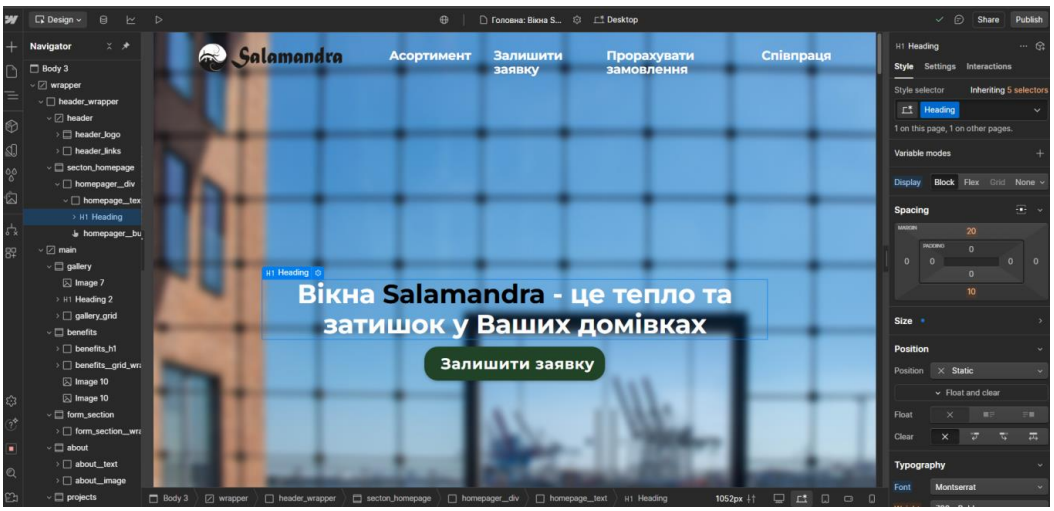


Рисунок В.18 - Інтерфейс Webflow

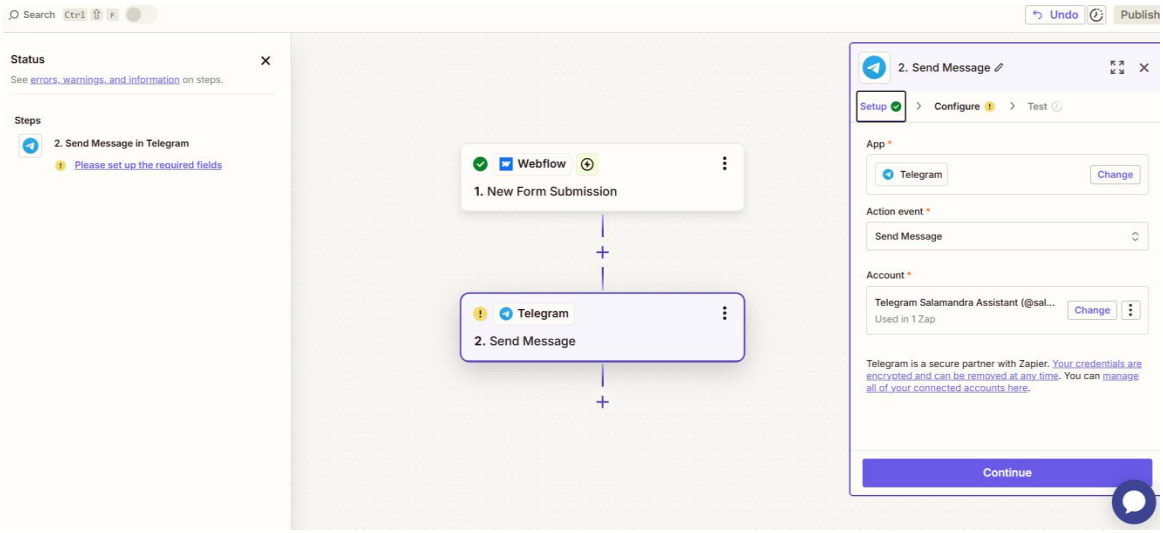


Рисунок В.19 - Інтерфейс плагіну Zapier

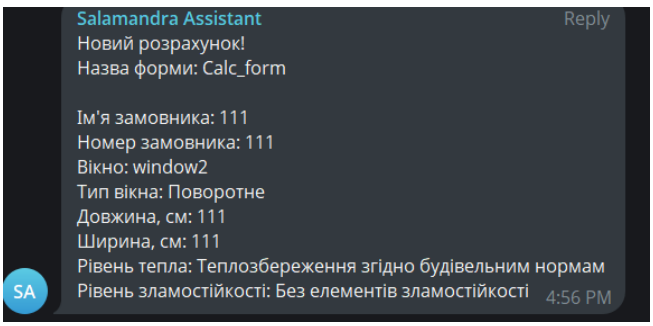


Рисунок В.21 - Приклад заявки з сайту

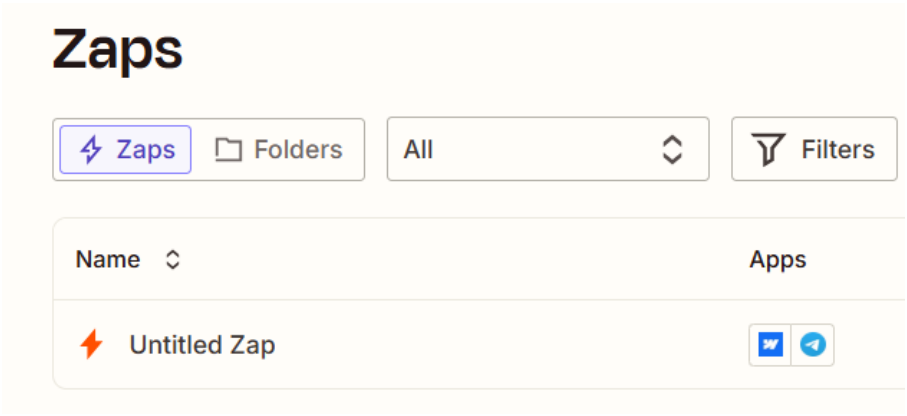


Рис В.23 - Виконання тригеру

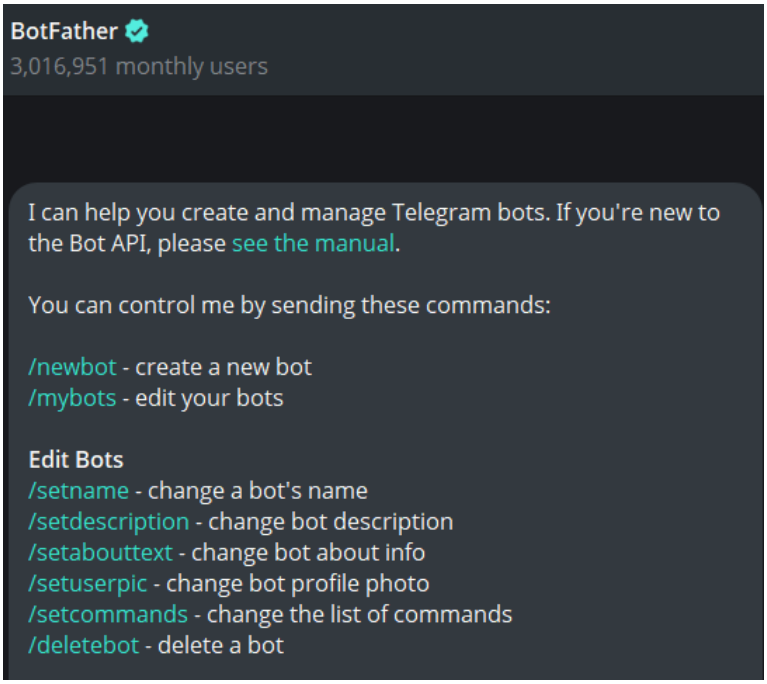
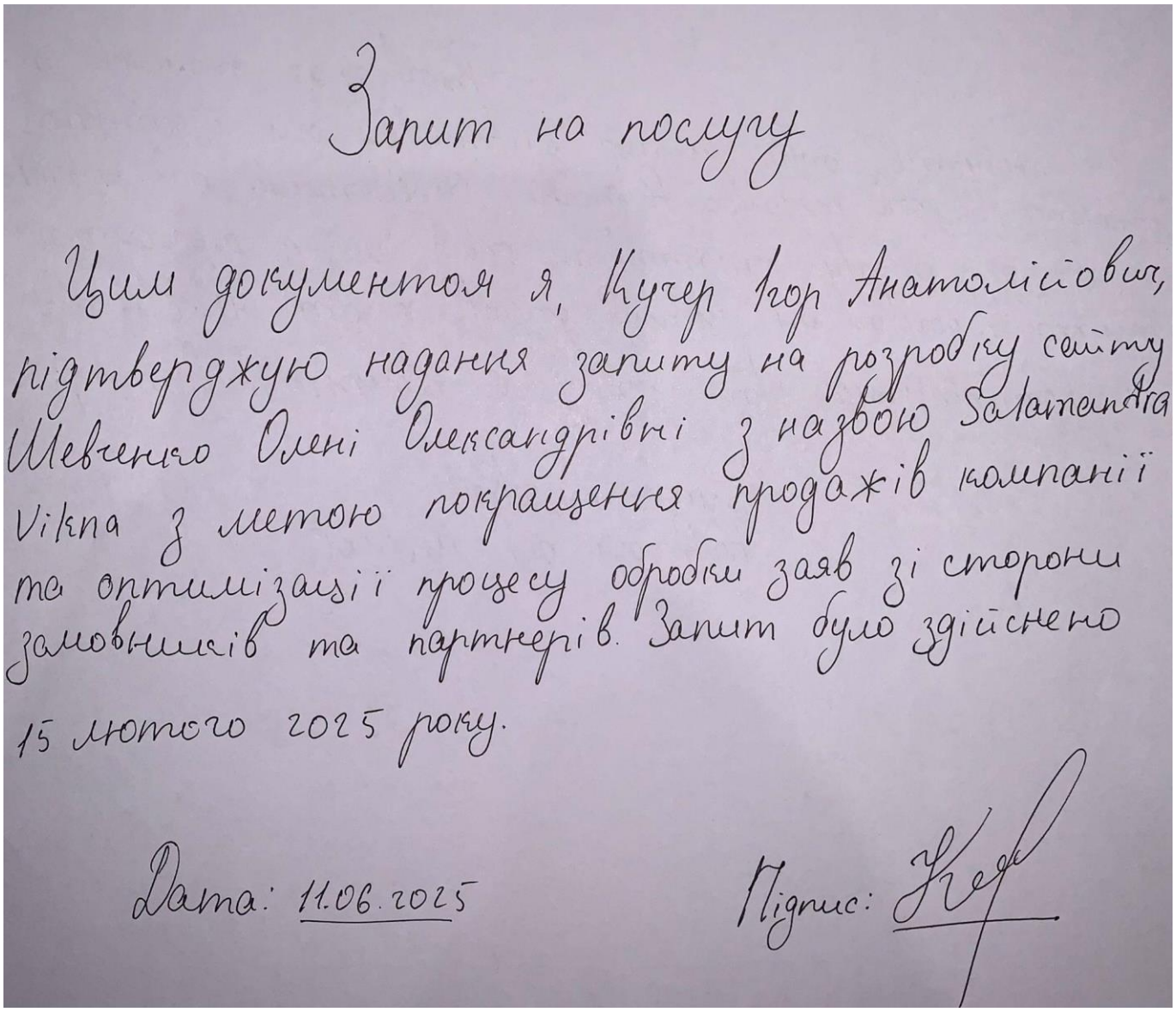


Рисунок В.24 - Створення Телеграм-бота

ДОДАТОК Г (ДОВІДНИКОВИЙ)

ЗАПИТ НА РОЗРОБКУ САЙТУ ВІД ЗАМОВНИКА



Запит на послугу

Цим документом я, Кучер Ігор Анатолійович, підтверджую надання запиту на розробку сайту Шевченко Олені Олександрівні з назвою Salamandra Vikna з метою покращення продажів компанії та оптимізації процесу обробки заяв зі сторони замовників та партнерів. Запит було здійснено 15 лютого 2025 року.

Дата: 11.06.2025

Підпис: Кучер