МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ
Факультет кібербезпеки, комп'ютерної та програмної інженерії
Кафедра інженерії програмного забезпечення

ДОПУСТИТИ ДО ЗАХИСТУ
Завідувач кафедри

"____"_____2023 р.

КВАЛІФІКАЦІЙНА РОБОТА
(ПОЯСНЮВАЛЬНА ЗАПИСКА)

ВИПУСКНИКА ОСВІТНЬОГО СТУПЕНЯ
МАГІСТРА

Тема: Методика та застосунок вдосконалення процесу розробки веб-сайтів

Виконавець: Набок Ростислав Сергійович

Керівник: к.т.н доцент Радішевський Микола Федорович

Нормоконтролер: ст. в. Гололобов Дмитро Олександрович

Київ 2023

НАЦІОНАЛЬНИЙ АВІАЦІЙНИЙ УНІВЕРСИТЕТ

Факультет кібербезпеки та програмної інженерії
Кафедра інженерії програмного забезпечення
Освітній ступінь магістр
Спеціальність 121 Інженерія програмного забезпечення
Освітньо-професійна програма «Програмне забезпечення систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри

"___" _____ 2023 р

ЗАВДАННЯ
на виконання кваліфікаційної роботи студента
Набок Ростислава Сергійовича

1. Тема кваліфікаційної роботи: «Методика та застосунок вдосконалення процесу розробки веб-сайтів» затверджена наказом ректора від 29.09.2023 р. № 1994/ст.

2. Термін виконання проекту: з 02.10.2022 р. по 31.12.2023 р.

3. Вихідні дані до роботи : програмний продукт розробити за допомогою фреймворку React.js та мови програмування JavaScript

4. Зміст пояснювальної записки:
   1. Аналіз існуючої системи вдосконалення розробки вебсайтів.
   2. Методика вдосконалення процесу розробки вебсайтів.
   3. Структура застосунку вебсайт білдеру.
   4. Прототип застосунку та перевірка методики вдосконалення розробки вебсайтів.

5. Перелік обов'язкових слайдів презентації:
   1. Аналіз існуючої системи вдосконалення розробки вебсайтів.
   2. Методика вдосконалення процесу розробки вебсайтів.
   3. Функціональні можливості програми.
   2. Інтерфейс системи.
   4. Схема роботи програми.
   5. Демонстрація роботи програми.
   6. Демонстрація роботи модулів програми.

6. Календарний план-графік

№ пор.	Завдання	Термін виконання	Відмітка про виконання
1.	Розробка та затвердження графіка роботи..	25-31.10	
2.	Підготовка та написання 1 розділу. Відсилка керівнику	01-07.11	
3.	Підготовка та написання 2 розділу. Відсилка керівнику	08-14.11	
4.	Підготовка та написання 3 та 4 розділу	15-21.11	
5.	Редагування та друк пояснювальної записки, графічного матеріалу Відправка ПЗ для перевірки на плагіат одним файлом.	13-19.12	
6.	Проходження нормо-контролю, перепліт пояснювальної записки. Отримання відгуку керівника. Підготовка презентації та тексту доповіді.	20-26.12	
7.	Передзахист	11-17.12	
8.	Отримання рецензії	18-24.12	
9.	Здати секретарю ДЕК		
10.	Захист дипломної роботи перед ЕК	25-31.12	

Дата видачі завдання 02.10.2023 р.

Керівник дипломної роботи: к.т.н доцент Микола РАДІШЕВСЬКИЙ

Завдання прийняв до виконання: Ростислав НАБОК

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи «Методика та застосунок вдосконалення процесу розробки веб-сайтів»: 79 сторінки, 34 рисунка, 12 таблиці.

ВДОСКОНАЛЕННЯ ПРОЦЕСУ, НОВІ МЕТОДИКИ, ДИНАМІЧНІ ВЕБ-САЙТИ, БІЛДЕР САЙТІВ, РОЗШИРЕННЯ ФУНКЦІОНАЛУ.

Об’єкт дослідження - дослідження існуючої методики розробки веб-сайтів.

Мета дипломної роботи - підвищення швидкості розробки веб-сайтів шляхом використання сучасних інструментів та можливостей вебсайт білдерів.

Метод дослідження - аналіз процесу розробки програмного продукту за допомогою якого бізнес може швидко та зручно створювати динамічні веб-сайти.

В процесі роботи, був зроблений глибокий аналіз ринку конкурентів та технологій в результаті чого виявилося, що на розробку вебсайту “з нуля” компанії витрачають дуже багато часу та грошей, а зробити це за допомогою конструкторів сайту це задача не для кожного.

Результати роботи можуть бути використані при розробці програмних засобів призначених для вдосконалення, пришвидшення розробки веб-сайтів.

Дослідження проводилися під управлінням ОС Windows 10. Розробка програми проводилася з використанням мови програмування JavaScript, бібліотеки React.js (https://react.dev), бібліотеки Builder.io (https://www.builder.io) та інші.

ABSTRACT

Explanatory note for the qualifying work 'Methodology and Application of Improving the Process of Web Development': 79 pages, 34 figures, 12 tables.

PROCESS IMPROVEMENT, NEW METHODOLOGIES, DYNAMIC WEBSITES, WEBSITE BUILDERS, FUNCTIONALITY EXTENSION.

Research Object - The application of the website development process improvement.

The objective of the diploma thesis is to enhance the speed of website development by utilizing modern tools and capabilities of website builders.

Research Method - Developing a software product that allows businesses to quickly and conveniently create dynamic websites.

During the work, a deep analysis of the market, competitors, and technologies was conducted, revealing that companies spend a significant amount of time and money on developing websites "from scratch," while doing it with website builders is not a task for everyone.

The results of the work can be utilized in the development of software tools intended for enhancing and expediting website development.

The research was conducted under the management of the Windows 10 operating system. The program development was carried out using the JavaScript programming language, React.js library (https://react.dev), Builder.io library (https://www.builder.io), and others.

ЗМІСТ

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ 8

ВСТУП 9

РОЗДІЛ 1. АНАЛІЗ ІСНУЮЧОЇ СИСТЕМИ ВДОСКОНАЛЕННЯ РОЗРОБКИ ВЕБСАЙТІВ 10

1.1 Аналіз стану розробки веб-сайтів в сучасному світі 10

1.2 Процес та стадії розробки сучасних веб-сайтів та проблеми з якими стикаються розробники. 11

1.3 Способи створення веб-сайтів 12

1.4 Таблиця середніх показників для веб-сайту, створеного за допомогою вебсайт-конструктора 14

1.5 Аналіз сучасних засобів та програм для пришвидшення розробки веб-сайтів 17

1.6 Аналіз ринку сучасних веб-сайт білдерів 18

1.7 Аналіз проблеми 25

1.8 Висновок 25

РОЗДІЛ 2. МЕТОДИКА ВДОСКОНАЛЕННЯ ПРОЦЕСУ РОЗРОБКИ ВЕБСАЙТУ 26

2.1 Проблеми в розробці сайтів з використанням вебсайт білдерів 26

2.2 Методики вирішення кожної проблеми та популярні інструменти для вирішення кожної з проблем 31

РОЗДІЛ 3. ЕКСПЕРЕМЕНТАЛЬНА ЧАСТИНА, СТРУКТУРА РОЗРОБЛЮВАНОЇ ПРОГРАМИ, ЇЇ ОПИС ТА ФУНКЦІЇ, ПЕРЕВІРКА МЕТОДИКИ 35

3.1 Основні вимоги до вебсайту та приклади з дизайну деяких вузлів системи: 37

3.2 Основні сторінки адмін панелі: 43

3.3 Основні сторінки публічного вебсайту: 46

3.4 Розробка адмін панелі використовуючи фреймворк react-admin: 51

3.5 Розробка публічного вебсайту використовуючи фреймворк next.js: 66
РОЗДІЛ 4. ПЕРЕВІРКА МЕТОДИКИ ВДОСКОНАЛЕННЯ РОЗРОБКИ ВЕБСАЙТІВ 78
4.1 Перевірка методики 78
ВИСНОВОК 80
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 81

ПЕРЕЛІК ПРИЙНЯТИХ СКОРОЧЕНЬ

JS – JavaScript

SSR – Server-side Rendering

SEO – Search Engine Optimization

UX – User Experience

HTTPS – Hypertext transfer protocol secure

CSS – Cascading Style Sheets

API – Application Programming Interface

URL – Uniform Resource Locator

ВСТУП

У сучасному цифровому світі веб-сайти стали невід'ємною частиною нашого повсякденного життя. Вони використовуються для комунікації, інформування, розваг та ведення бізнесу. Зараз існує безліч інструментів і технологій для розробки веб-сайтів, що вимагають від фахівців у цій галузі постійного вдосконалення.

Однією з ключових задач веб-розробників є забезпечення високої якості та ефективності веб-сайтів, щоб задовольнити потреби користувачів і досягти бізнес-цілей. Для досягнення цієї мети необхідно постійно вивчати нові технології, методи та практики розробки веб-сайтів.

Аналіз ринку веб-розробки показує, що важливою складовою успішності є здатність веб-розробника адаптуватися до змінних умов та вдосконалювати свої навички. Практика показує, що через різноманітні фактори (нові технології, зміни в вимогах ринку, необхідність робити перерви у роботі) фахівці в галузі веб-розробки можуть втрачати свою ефективність та актуальність.

Саме тому, актуальною задачею є розробка методики та застосування спеціалізованих інструментів для покращення процесу розробки веб-сайтів та постійного підвищення кваліфікації веб-розробників. У цьому контексті, можливість створення і використання спеціалізованих тренажерів чи інших засобів для тренування та вдосконалення навичок веб-розробників набуває великого значення.

РОЗДІЛ 1.
АНАЛІЗ ІСНУЮЧОЇ СИСТЕМИ ВДОСКОНАЛЕННЯ РОЗРОБКИ ВЕБСАЙТІВ

У сучасному цифровому світі, веб-сайти відіграють важливу роль як засіб комунікації, інформування, реклами та обслуговування користувачів. Вони є важливим інструментом для бізнесу, громадських організацій, освітніх установ, та інших сфер діяльності. Процес розробки веб-сайтів є складним та багатоаспектним, і його якість впливає на користувачів, бізнес та репутацію веб-розробників.

1.1 Аналіз стану розробки веб-сайтів в сучасному світі

Аналіз сучасного стану розробки веб-сайтів показує наявність ряду проблем, які потребують уваги та вдосконалення. Найбільш важливі з них включають:

1. Різноманітність технологій та інструментів: Сучасний ландшафт веб-розробки пропонує безліч технологій і фреймворків, що можуть бути використані для створення веб-сайтів. Наприклад, вибір мови програмування, такої як JavaScript, Python, PHP або Ruby, залежить від конкретних потреб проекту. Кожна мова має свої переваги та обмеження, і вибір повинен бути обґрунтованим.
2. Забезпечення безпеки: Кібербезпека стає все більшою проблемою веб-розробки, оскільки інтернет постійно під загрозою зловмисників. Важливо здійснювати регулярні аудити та пентестування, щоб виявляти і усувати потенційні вразливості веб-сайтів. Потрібно також використовувати актуальні версії фреймворків та бібліотек, а також розробляти безпечний код.
3. Мобільна адаптація: Зростання популярності мобільних пристроїв вимагає створення веб-сайтів, які будуть коректно відображатися на різних розмірах екранів. Це означає використання адаптивного дизайну та розробку мобільних додатків, які забезпечують оптимальний досвід користувача на мобільних пристроях.

4. Оптимізація продуктивності: Важливість оптимізації продуктивності полягає в тому, щоб зменшити час завантаження сторінок та зробити веб-сайт більш ефективним. Це включає у себе мінімізацію зображень, кешування статичного контенту, компресію файлів і роботу з CDN-системами для зменшення завантаження сервера.
5. Навчання та професійний розвиток: Сфера веб-розробки швидко розвивається, і фахівці мають постійно навчатися новим технологіям і методам роботи. Онлайн-курси, воркшопи, конференції та книги можуть допомогти веб-розробникам вдосконалювати свої навички і залишатися актуальними в галузі.

Ці проблеми веб-розробки стають все більш актуальними в сучасному світі, де веб-сайти є важливим інструментом для багатьох видів діяльності. Розвиток методик і інструментів для вдосконалення процесу розробки веб-сайтів є необхідним для забезпечення якості та безпеки веб-продуктів, а також для підтримки професійного росту фахівців у цій галузі.

1.2 Процес та стадії розробки сучасних веб-сайтів та проблеми з якими стикаються розробники.

Розробка сучасних вебсайтів - це складний та багатоетапний процес, який включає в себе кілька ключових етапів. Ось загальна схема розробки вебсайту:

Визначення цілей та вимог: команда розробки спільно з клієнтом визначає цілі та завдання вебсайту, а також збирає всі необхідні вимоги.

Проектування: дизайнери вирішують структуру та вигляд вебсайту, створюють макети та прототипи. Визначається архітектура інформації, навігаційна структура та інші ключові елементи.

Розробка: розробники використовують різноманітні технології (HTML, CSS, JavaScript, фреймворки) для створення фронтенду вебсайту. На бекенді використовуються мови програмування (наприклад, Python, PHP, Java) та бази даних для обробки запитів користувачів та зберігання даних.

Тестування: проводяться різні види тестування, такі як тестування функціональності, тестування безпеки та тестування відмовостійкості. Виправляються помилки та недоліки, забезпечуючи високу якість вебсайту.

Впровадження: розроблений вебсайт впроваджується на серверах та стає доступним для користувачів.

Підтримка та оновлення: забезпечується подальша технічна підтримка, а також можливість внесення оновлень та доповнень.

Проте, під час розробки вебсайтів, люди можуть зіткнутися з рядом проблем:

Недостатність вимог: неясні або неповні вимоги від клієнта можуть призвести до непорозумінь у процесі розробки.

Зміни вимог під час розробки: зміни вимог серед виробництва можуть призвести до збільшення обсягу роботи та затримок у графіку.

Проблеми з безпекою: недоліки в безпеці можуть виникнути, що призводить до можливих загроз для конфіденційності та цілісності даних.

Розбіжності відомостей між командами: недостатня комунікація між різними командами (дизайн, розробка, тестування) може спричинити розбіжності в розумінні завдань та вимог.

Проблеми з оптимізацією швидкості та продуктивності: несправні або неоптимізовані елементи можуть призвести до повільної роботи вебсайту, що негативно впливає на користувацький досвід.

Сумісність із браузерами: різні браузери можуть відрізнятися у відображенні та виконанні коду, що може призвести до проблем зі сумісністю.

Вирішення цих проблем вимагає уважності до деталей, ефективного управління проектом та спільної роботи всіх учасників команди розробки.

1.3 Способи створення веб-сайтів

Існує декілька способів створення вебсайтів. Кожен з них має свої переваги та недоліки тому кожен замовник старається знайти ідеальну середину в таких категоріях як варстість, швидкість розробки, наповненість, функціонал, переваги

над конкурентами. Від обраного способу розробки буде залежати кінцевий результат тому треба підходити до цього максимально серйозно, зважаючи на всі індивідуальні характеристики. Способи розробки веб-сайтів можна поділити на 4 групи, їх можна побачити на рисунку 1.1.

Рис. 1.1 – способи створення сайтів

Очевидно, що найбільш швидкий спосіб це конструктор сайтів, він також є достатньо гнучким, проте в цьому способі є декілька проблем які я описав в пункті 1.5. Моя задача та ціль - оптимізувати процес розробки під конкретний домен (обрано домен “Нерухомість”) використовуючи білдер (конструктор) технології.

1.4 Таблиця середніх показників для веб-сайту, створеного за допомогою вебсайт-конструктора (Табл. 1.1, 1.2, 1.3, 1.4, 1.5, 1.6, 1.7, 1.8, 1.9). [1]

Таблиця 1.1

Дизайн і UX

Показник	***Обов'язково мати***	***Бажано мати***
Дизайн і UX		
Загальний вигляд	+	
Адаптивний дизайн	+	
Інтуїтивна навігація		+
Швидкість завантаження		+

Таблиця 1.2

Функціональність

Функціональність		
Форми зворотнього зв'язку		+
Реєстрація та авторизація		+
Можливість пошуку		+
Інтеграція з соцмережами		+
Адмін панель для управління		+

Підтримка SEO для вебсайтів є критично важливою. Вона забезпечує підняття вебсайту в пошукових результатах, збільшуючи його видимість та рейтинг. Ефективна SEO привертає цільову аудиторію, спрямовуючи користувачів, які шукають конкретну інформацію чи продукти. Високий рейтинг збільшує довіру користувачів і підвищує авторитет вебсайту. Відсутність SEO може призвести до

втрати конкурентної переваги перед іншими вебсайтами. Зокрема, SEO сприяє підвищенню конверсій, дозволяючи вебсайту бути ефективнішим та привабливим для своєї аудиторії.

Таблиця 1.3

Контент і SEO

Контент і SEO		
Якісний контент	+	
Оптимізація для пошукових систем	+	
Мапа сайту	+	

Таблиця 1.4

Аналітика і відстеження

Аналітика і відстеження		
Google Analytics	+	
Відстеження конверсій		+

Безпека вебсайтів є невід'ємною частиною їхньої експлуатації та успішної роботи. Забезпечення безпеки включає в себе різноманітні заходи та стратегії для захисту від різних видів загроз і зловмисних дій.

Таблиця 1.5

Безпека

Безпека		
HTTPS	+	
Захист від SQL-ін'єкцій	+	
Захист від Cross-Site Scripting (XSS)	+	

Таблиця 1.6

Сумісність

Сумісність		
Сумісність із різними браузерами	+	

Таблиця 1.7

Підтримка

Підтримка		
Технічна підтримка		+
Оновлення та патчі	+	

Таблиця 1.8

Маркетинг та звітність

Маркетинг та звітність		
Інтеграція із соцмережами для реклами		+
Звіти про відвідуваність		+

Таблиця 1.9

Локалізація

Локалізація		
Підтримка кількох мов	+	

Ця таблиця надає загальний огляд показників, які важливі для веб-сайту. Обов'язкові показники вказують на ключові аспекти, які необхідно мати для ефективного та успішного веб-сайту, тоді як бажано мати вказує на елементи, які

можуть покращити користувацький досвід та функціональність сайту. Зважаючи на все це я хочу створити свій власний конструктор який зможе “закрити” більшість бажаних пунктів.

1.5 Аналіз сучасних засобів та програм для пришвидшення розробки веб-сайтів

Аналіз програм, які допомагають пришвидшити процес розробки веб-сайтів, важливий для збільшення продуктивності та якості роботи веб-розробників. Ось кілька популярних програм та інструментів, які використовуються для цієї мети:

Integrated Development Environments (IDEs):

Visual Studio Code (VS Code): VS Code є надзвичайно популярним та легким текстовим редактором, який має широкий вибір розширень та плагінів для веб-розробки. Він підтримує багато мов програмування та має вбудовані інструменти для роботи з Git.

WebStorm: Це спеціалізована IDE для розробки веб-сайтів, яка надає інтегровану підтримку для JavaScript, HTML, CSS та інших веб-технологій. Вона має потужні інструменти для автодоповнення коду, налагодження та оптимізації.

Фреймворки та бібліотеки:

React: Для швидкого розробки користувацького інтерфейсу веб-сайтів, React є популярним вибором. Він дозволяє створювати компоненти, які можна повторно використовувати та легко оновлювати.

Bootstrap: Цей CSS-фреймворк містить готові стилі та компоненти, що значно спрощує процес дизайну та розміщення елементів на сторінці.

Системи керування версіями:

Git: Git є найпопулярнішою системою керування версіями. Він дозволяє веб-розробникам працювати над проектами разом, відстежувати зміни в коді та відновлювати попередні версії.

Препроцесори CSS:

Sass/SCSS: Sass дозволяє використовувати змінні, вкладені стилі та інші покращення для CSS, що полегшує розробку та підтримку стилів на веб-сайтах.

Автоматизація завдань:

Gulp ma Grunt: Ці інструменти дозволяють автоматизувати рутинні завдання в розробці, такі як компіляція Sass, оптимізація зображень, об'єднання та стиснення файлів.

Менеджери пакетів:

npm (Node Package Manager): npm дозволяє легко установлювати та керувати бібліотеками та пакетами, які використовуються в проекті.

Інструменти для тестування:

Jest: Для тестування JavaScript коду, Jest надає можливості для автоматизованого тестування функціональності та візуального порівняння.

Ці програми та інструменти допомагають веб-розробникам раціоналізувати процес розробки, вдосконалюючи продуктивність та якість роботи. Вони дозволяють автоматизувати багато рутинних завдань та надають зручні засоби для співпраці, тестування та оптимізації веб-сайтів.

1.6 Аналіз ринку сучасних веб-сайт білдерів [2]

Аналіз білдерів веб-сайтів допоможе краще розібратися у їх можливостях та знайти найкращий інструмент для конкретних потреб. Вебсайт-білдери - це інструменти, призначені для швидкого та простого створення вебсайтів без необхідності глибоких технічних навичок програмування. Вони надають користувачам готові шаблони та інтуїтивно зрозумілі інтерфейси для дизайну та

редагування вмісту. Ось огляд деяких популярних білдерів веб-сайтів, включаючи їх функції, переваги та недоліки:

Wix (https://uk.wix.com/):

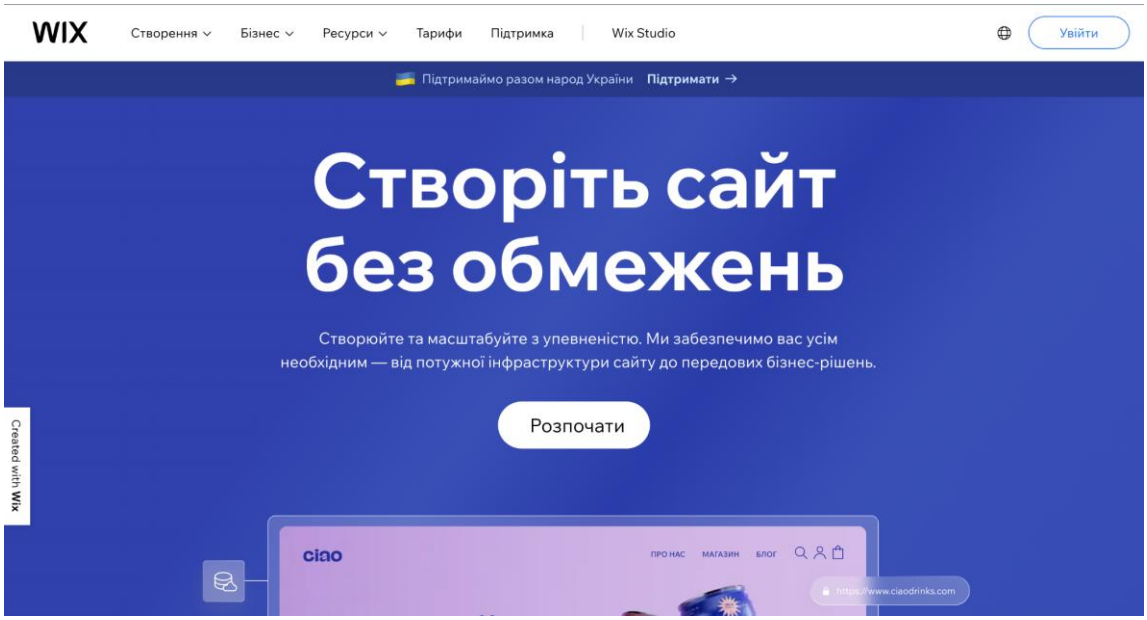

Рис. 1.2 – головна сторінка сайту Wix

Опис: Wix (Рис. 1.2) є одним із найпопулярніших білдерів веб-сайтів, призначеним для різних видів користувачів, включаючи особисті веб-сайти, бізнес-проекти та онлайн-магазини.

Функції:

- Шаблони: Wix має велику кількість готових шаблонів для різних галузей та потреб. Ви можете вибрати шаблон, який найкраще відповідає вашому проекту та внести зміни в ньому, щоб відповідати вашим потребам.
- Візуальний редактор: Wix надає візуальний редактор, який дозволяє редагувати сторінки за допомогою перетягування та розміщення елементів. Це робить процес створення веб-сайту дуже інтуїтивним.
- Додаткові функціональність: Wix має велику кількість додаткових функцій та плагінів, які дозволяють додавати різноманітну функціональність до веб-сайту, таку як блоги, форми зворотного зв'язку, галереї, онлайн-магазини тощо.

- SEO-інструменти: Wix надає інструменти для оптимізації веб-сайту для пошукових систем. Ви можете налаштовувати метатеги, URL-адреси та інші параметри, щоб підвищити видимість вашого сайту в пошукових системах.
- Безкоштовний хостинг: Wix надає безкоштовний хостинг для веб-сайтів, створених на їхній платформі. Це включає безкоштовне доменне ім'я, яке виглядає як "ім'явашогосайту.wix.com".

Переваги: Простий для використання, можливість безкоштовного створення веб-сайту, готові шаблони та велика спільнота користувачів.

Недоліки: Відсутність повного доступу до коду, обмежена можливість переносити вміст на інші платформи.

Squarespace (https://www.squarespace.com/):

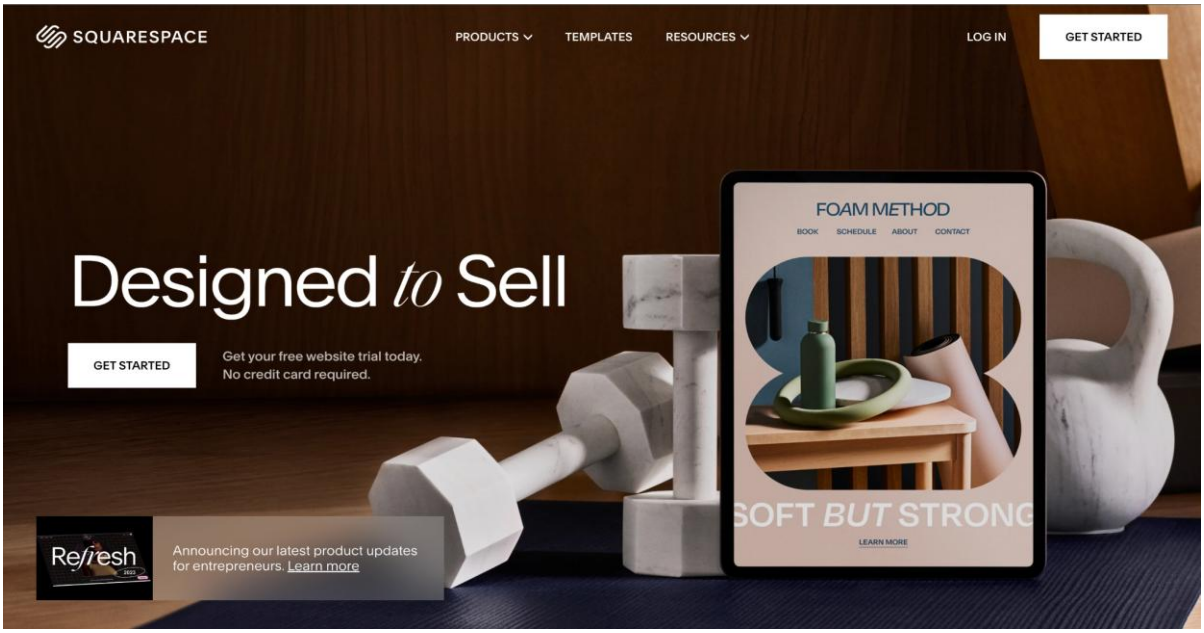

Рис. 1.3 – головна сторінка сайту Squarespace

Опис: Squarespace (Рис. 1.3) відомий своїми стильними та мінімалістськими дизайнами, спрямованими на креативних професіоналів та художників.

Функції:

- Готові шаблони: Squarespace пропонує вражаючий вибір готових дизайнів, які легко налаштовувати під свої потреби. Шаблони підходять для різних видів проектів, включаючи особисті веб-сайти, портфоліо, магазини і блоги.
- Візуальний редактор: Спеціальний візуальний редактор Squarespace дозволяє вам редагувати вміст та дизайн свого сайту за допомогою перетягування та розміщення елементів. Він дозволяє докладно налаштовувати сторінки без кодування.
- Інтеграція з соціальними мережами: Squarespace дозволяє підключити свої соціальні медіа профілі та відображати їх на веб-сайті. Ви також можете додавати соціальні кнопки та інші елементи для взаємодії зі своїми відвідувачами.
- Аналітика: Платформа надає інструменти для відстеження відвідуваності та аналізу поведінки ваших користувачів на веб-сайті.
- Безкоштовний хостинг: Squarespace включає безкоштовний хостинг та безкоштовне доменне ім'я на платформі. Ви також можете підключити власний домен до сайту.

Переваги: Вражаючі дизайни, простота використання, включена вартість хостингу та домену.

Недоліки: Обмежена кількість шаблонів порівняно з іншими білдерами, обмежена можливість налаштування дизайну.

Weebly (https://www.weebly.com):

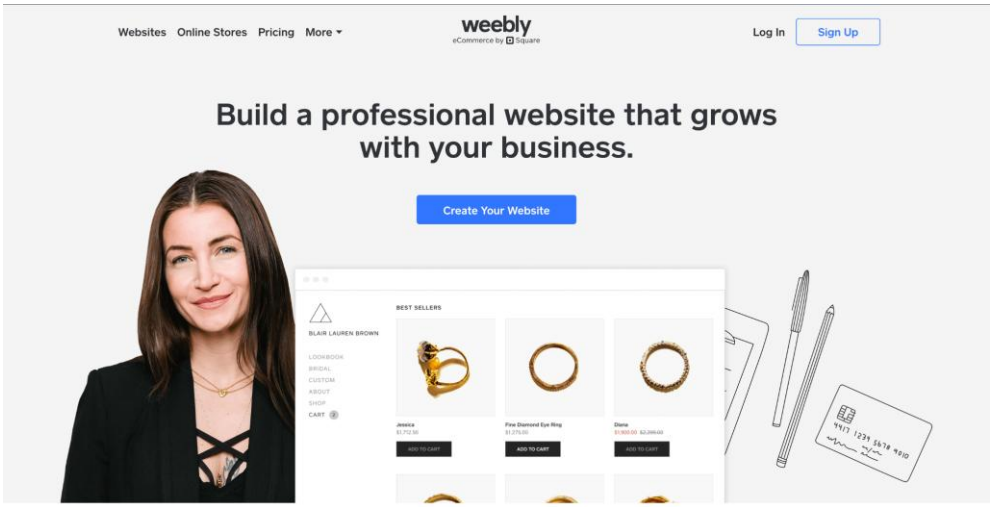

Рис. 1.4 – головна сторінка сайту Weebly

Опис: Weebly (Рис. 1.4) - інтуїтивно зрозумілий білдер веб-сайтів, що підходить для різних видів проектів, включаючи особисті блоги, бізнес-сайти та онлайн-магазини.

Функції:

- Готові шаблони: Weebly надає великий вибір готових дизайнів, які можна налаштовувати за своїми потребами. Вони включають стильні та сучасні варіанти для різних галузей.
- Візуальний редактор: Weebly пропонує візуальний редактор, який дозволяє перетягувати та розміщувати елементи на сторінці, редагувати текст та зображення. Це робить процес створення веб-сайту дуже інтуїтивним.
- Можливість додавання сторінок та функціональності: Ви можете додавати нові сторінки, блоги, галереї та іншу функціональність до свого сайту за допомогою вбудованих інструментів.
- Аналітика: Weebly надає інструменти для аналізу відвідуваності та поведінки користувачів на веб-сайті.

Переваги: Легка використання, безкоштовний план доступний, магазин додатків для розширення функціональності.

Недоліки: Обмежена можливість налаштування дизайну порівняно з іншими білдерами.

WordPress.com (https://wordpress.com):

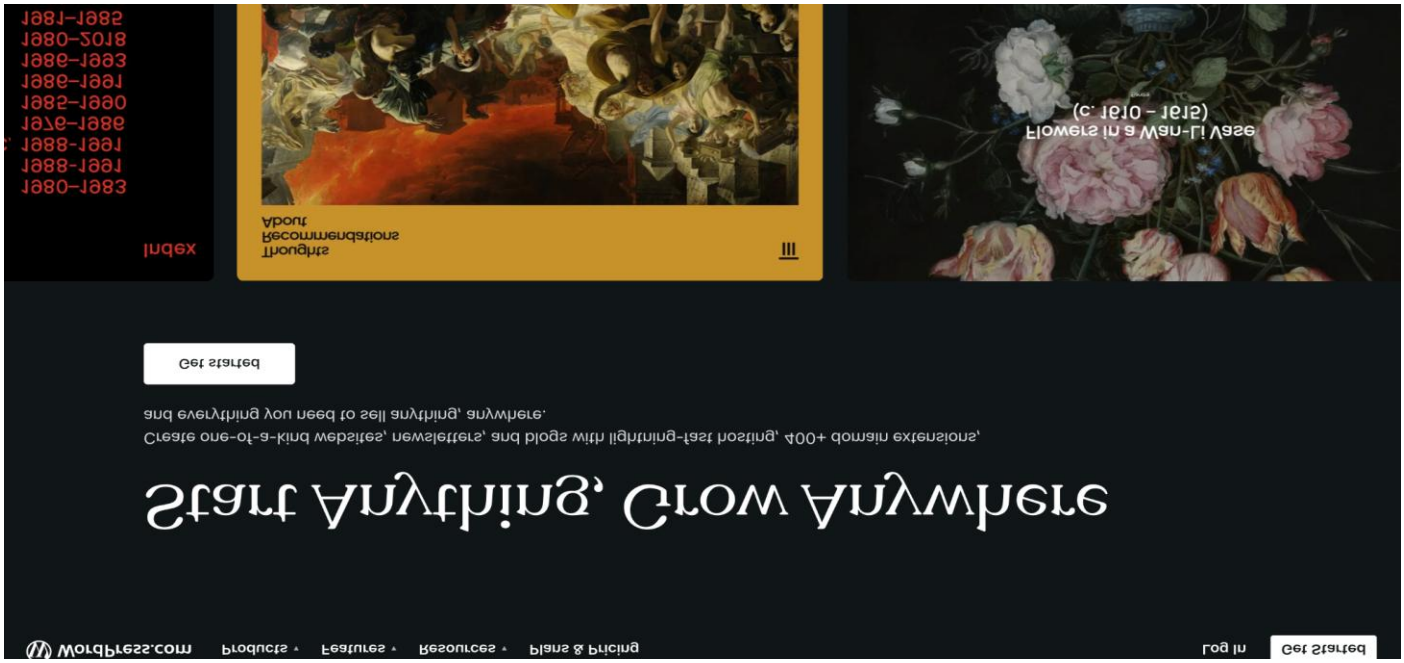

Рис. 1.5 – головна сторінка сайту WordPress

Опис: WordPress.com (Рис. 1.5) - це хостингова платформа для веб-сайтів на базі WordPress, яка надає можливість створювати та налаштовувати сайти за допомогою візуального редактора.

Функції:

- Велика спільнота: WordPress має велику спільноту користувачів та розробників, яка підтримує безліч тем, плагінів та розширень для різних потреб.
- Великий вибір тем: Є безліч готових тем, які допомагають створити веб-сайт різного стилю та спрямування.
- Плагіни та розширення: WordPress надає можливість встановлення різних плагінів для розширення функціональності, таких як SEO, аналітика, соціальні мережі та багато інших.
- Повний доступ до коду: WordPress дає можливість повного доступу до коду, що дозволяє розробникам налаштовувати та розширювати сайт за їхніми потребами.

- Багато мов і перекладів: WordPress підтримує багато мов та має широкий спектр перекладів для глобальних аудиторій.

Переваги: Широкий вибір дизайнів та функціональності, повний доступ до коду (при підписці на платний план).

Недоліки: Навчання може бути складним для початківців, платні плани можуть бути дорожчими порівняно з іншими білдерами.

Shopify (https://www.shopify.com):

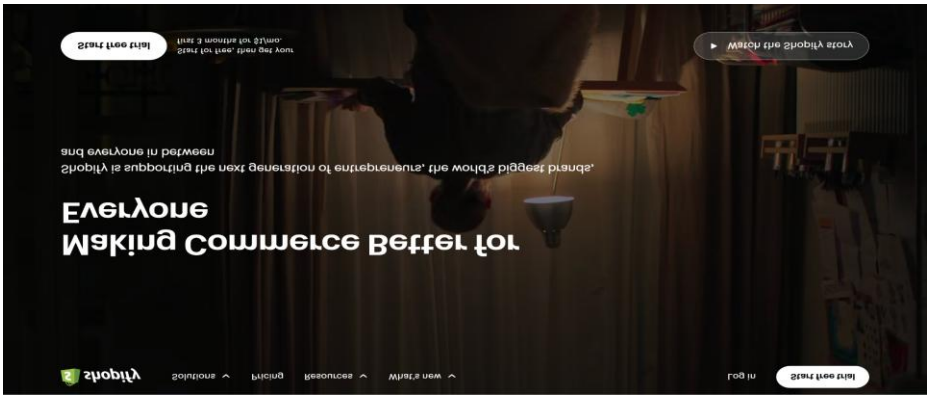

Рис. 1.6 – головна сторінка сайту Shopify

Опис: Shopify (Рис. 1.6) - це спеціалізований білдер для створення онлайн-магазинів.

Функції:

- Готові теми та дизайн: Shopify пропонує великий вибір готових тем та шаблонів для різних видів товарів і брендів. Ви можете вибрати тему, яка відповідає вашим потребам та бренду, і налаштувати її за своїми уподобаннями.
- Елементи керування товарами: Ви можете додавати та редагувати товари, встановлювати ціни, ведення обліку запасів та встановлювати податки.
- Платіжні системи та обробка замовлень: Shopify інтегрується з різними платіжними системами і надає засоби обробки замовлень. Ви можете приймати кредитні картки, PayPal та інші платіжні методи.
- Аналітика та звітність: Shopify надає інструменти для аналізу продажів, відвідуваності та іншої статистики для оптимізації бізнесу.

- Можливості SEO: Ви можете оптимізувати свій магазин для пошукових систем, включаючи налаштування метатегів та SEO-френдлі URL-адрес.

Переваги: Надійна платформа для електронної комерції, великий магазин додатків для розширення функціональності.

Недоліки: Перелік шаблонів може бути обмеженим, плани також можуть бути платними.

1.7 Аналіз проблеми

Як бачимо, більшість білдерів мають обмежений функціонал, обмежену або урізану кількість шаблонів, дорогу підписку або ж вони складні у використанні. Деякі з них не заточені під конкретну предметну область або домен тому бізнесу не так легко взаємодіяти з цими інструментами, підлаштовувати компоненти ибо логіку під себе. В цій дипломній роботі я хочу представити власний білдер на React.js/Next.js (JavaScript бібліотеки) та бібліотеці Builder.io який можна гнучко налаштувати під домен “Нерухомість, оренда нерухомості”, використовувати заготовлені шаблони та темплейти. Це допоможе бізнесу та компаніям створювати вебсайти швидко, конкретно під цілі. Також буде додана адмін панель в якій можна зручно налаштовувати вебсайт, додавати темплейти, створювати сторінки в пару кліків та “будувати” сайт з компонентів для себе.

1.8 Висновок

Отже, проаналізувавши ринок та доступні вебсайт білдери я можу зробити висновок, що за допомогою них можна створити вебсайт який дуже сильно залежить від можливостей платформи, з урізаним функціоналом тому як раз під конкретний домен буде дуже доцільно зробити свій власний білдер.

РОЗДІЛ 2.
МЕТОДИКА ВДОСКОНАЛЕННЯ ПРОЦЕСУ РОЗРОБКИ ВЕБСАЙТУ

2.1 Проблеми в розробці сайтів з використанням вебсайт білдерів

Використання вебсайт-білдерів може спростити процес розробки вебсайту, але, як і в будь-якому іншому виді розробки, можуть виникати проблеми. Ось кілька загальних проблем та можливі варіанти їх вирішення:

Обмежена гнучкість та кастомізація:

Проблема обмеженої гнучкості та кастомізації в вебсайт-білдерах випливає з обмежень функціоналу, шаблонного оформлення та обмежених можливостей кастомізації. Багато таких інструментів пропонують обмежений набір дизайнів та стилів, обмежуючи тим самим творчість користувачів. Також, відсутність

розширених кодових редакторів може ускладнювати глибокі зміни в структурі веб-сайту. Крім того, залежність від конкретного постачальника може ускладнити перенесення веб-сайту на іншу платформу. Інтеграційні обмеження та обмеження в можливостях замовництва також можуть впливати на спроби користувачів створювати більш складні та індивідуалізовані веб-проекти. Для тих, хто шукає більшу гнучкість, альтернативні інструменти або професійні веб-розробники можуть бути більш відповідними виборами.

Швидкість та оптимізація:

Проблема обмеженої швидкості та неоптимізованості в вебсайт-білдерах виникає через обмежену швидкість завантаження та нестабільну роботу веб-сайтів. Багато інструментів можуть стикається з обмеженнями у видачі оптимізованого коду, що призводить до сповільнення завантаження сторінок. Недостатня можливість оптимізації графіки та ресурсів також може впливати на продуктивність веб-сайту. Крім того, відсутність ефективних інструментів кешування та стискання може збільшувати час завантаження сторінок. Залежність від конкретного хостингу та інфраструктури також може впливати на швидкість роботи веб-сайту, обмежуючи можливості оптимізації сервера та ресурсів. Для користувачів, які прагнуть покращити швидкість своїх веб-сайтів, розглядання альтернативних інструментів або застосування додаткових методів оптимізації може бути важливим аспектом.

Сумісність із сторонніми сервісами:

Проблема обмеженої сумісності зі сторонніми сервісами в вебсайт-білдерах виникає через обмежені можливості інтеграції з різноманітними зовнішніми сервісами та додатками. Багато інструментів можуть не надавати достатньої гнучкості для взаємодії з різними API і сервісами, що може обмежити можливості користувачів розширювати функціоналість свого веб-сайту через сторонні додатки. Неявна або обмежена підтримка різних форматів даних та протоколів може стати перешкодою для ефективної інтеграції.

Залежність від власних екосистем або обмежений набір інтеграцій може ускладнювати роботу користувачів, які прагнуть використовувати конкретні зовнішні сервіси для покращення функціоналу свого веб-сайту. Це може бути особливо проблематично для тих, хто планує використовувати спеціалізовані інструменти або сервіси для конкретних потреб, таких як аналітика, електронна комерція чи соціальна мережа.

Для користувачів, які потребують великої сумісності із сторонніми сервісами, важливо обирати вебсайт-білдер, який надає широкі можливості інтеграції та підтримку різноманітних екосистем для задоволення їхніх унікальних потреб.

Обмежені можливості SEO:

Проблема обмежених можливостей SEO в вебсайт-білдерах виникає через обмеженість інструментів оптимізації для пошукових систем. Багато вебсайт-білдерів можуть не надавати достатніх можливостей для повноцінного SEO, що може впливати на видимість та ранжування веб-сайту в пошукових системах.

Обмеженість у редагуванні технічних аспектів, таких як заголовки, мета-теги, чи файл robots.txt, може ускладнювати користувачам оптимізацію свого веб-сайту для кращого індексування пошуковими системами. Також, відсутність можливості налаштовувати швидкість завантаження сторінок чи оптимізовувати структуру URL може обмежувати повний контроль над технічними аспектами, які впливають на SEO.

Деякі вебсайт-білдери можуть автоматично генерувати URL або використовувати стандартні шаблони, що не завжди є оптимальним для ключових слів та структури сайту. Це може ускладнити оптимізацію для конкретних ключових фраз та стратегій SEO.

Для тих, кому важливий успішний SEO, важливо враховувати можливості оптимізації, які пропонує обрана платформа для створення веб-сайту, а також розглядати можливість використання додаткових інструментів чи платформ, які забезпечують більш широкі можливості SEO.

Вартість та власні ресурси:

В контексті вебсайт-білдерів виникають проблеми з вартістю та власними ресурсами. Вартість використання може бути високою, особливо для розширених функцій. Обмеженість на розширення та обмеження власних ресурсів може ускладнювати масштабування веб-проектів. Залежність від постачальника може впливати на вартість та доступність ресурсів. Втрата контролю над даними стає актуальною при використанні платформи. Для деяких користувачів, самостійна веб-розробка або використання інших платформ може бути вибором для більшого контролю та гнучкості.

Обмежена масштабованість:

Проблема обмеженої масштабованості в вебсайт-білдерах виникає внаслідок обмежень у здатності розширювати та адаптувати веб-сайт з ростом потреб та об'єму контенту. Багато вебсайт-білдерів можуть стикатися з обмеженнями щодо кількості сторінок, об'єму даних, чи інших параметрів, що може ускладнювати розвиток великих або розширених веб-проектів.

Це може бути особливо проблематичним для бізнесів або проектів, які очікують значний ріст та потребують постійного оновлення та розширення. Обмежена кількість шаблонів чи можливостей для додавання додаткових функцій може ускладнювати адаптацію веб-сайту до змін потреб користувачів чи бізнес-вимог.

Для тих, кому важлива масштабованість, розглядання альтернативних рішень, таких як використання більш гнучких платформ для розробки веб-сайтів чи найм професіоналів для створення індивідуальних рішень, може бути розумним вибором.

Недоліки щодо безпеки:

Проблеми безпеки в вебсайт-білдерах можуть виникати з кількох причин. По-перше, багато вебсайт-білдерів працюють на основі загальних шаблонів, що може робити їх вразливими до атак, орієнтованих на конкретні платформи. Нестача

індивідуального підходу може створювати однотипність, що полегшує зловмисникам знаходження слабких місць.

Деякі вебсайт-білдери можуть мати обмежені можливості з точки зору захисту від кібератак. Вони можуть не надавати широкі можливості налаштування параметрів безпеки, що робить важчим забезпечення надійності веб-сайту. Крім того, оновлення безпеки та захист від нових загроз може бути обмеженим у випадку вебсайт-білдерів, оскільки це часто залежить від самого постачальника.

Залежність від вебсайт-білдера може також означати, що ви не маєте повного контролю над доступом до своїх даних. Це може створювати ризики щодо конфіденційності і безпеки важливих інформаційних ресурсів.

Для забезпечення оптимального рівня безпеки, користувачам може бути вигідно ретельно розглядати політику безпеки обраного вебсайт-білдера, враховувати можливості шифрування, налаштування прав доступу та перевірку оновлень безпеки.

Проблеми із сумісністю браузерів:

Проблеми сумісності з браузерами в вебсайт-білдерах можуть виникнути через різні фактори. Оскільки вебсайт-білдери зазвичай використовують власні технології та кодову базу, сумісність із всіма браузерами не завжди гарантована.

Багато вебсайт-білдерів можуть оптимізовуватися для певних популярних браузерів, таких як Google Chrome або Mozilla Firefox, і це може викликати проблеми в інших менш популярних браузерах. Різноманіття версій браузерів також може вносити свої відмінності відображення та функціональність створених за допомогою вебсайт-білдера сайтів.

Потрібно враховувати, що вебсайт-білдери можуть використовувати специфічні технології, які можуть бути несумісні з усіма браузерами. Нестача можливостей налаштування або відсутність перевірки сумісності може призводити до того, що веб-сайт виглядає та працює не так, як очікувалося, на певних браузерах.

Для забезпечення максимальної сумісності з браузерами важливо тестувати веб-сайт на різних платформах та браузерах і враховувати особливості їх роботи при використанні конкретного вебсайт-білдера.

Вартість додаткових функцій:

Вартість додаткових функцій в вебсайт-білдерах може стати проблемою для користувачів, які прагнуть розширити функціонал свого веб-сайту за межі базового набору можливостей. Багато вебсайт-білдерів пропонують основні функції на стартовому рівні, але додаткові, більш потужні або спеціалізовані опції часто доступні за додаткову вартість.

Вартість додаткових функцій може швидко наростати, особливо якщо вам потрібно декілька розширень чи опцій для задоволення ваших конкретних потреб. Також, ціни на додаткові функції можуть варіюватися від одного вебсайт-білдера до іншого, що може створювати додаткові труднощі при бюджетуванні.

Деякі користувачі можуть виявити, що для отримання повноцінного набору функцій, який вони бажають, вони вимушені перейти на вищий тарифний план або навіть використовувати додаткові платні додатки. Це може вплинути на загальну вартість використання вебсайт-білдера та зробити його менш привабливим для тих, хто прагне економії коштів.

При виборі вебсайт-білдера важливо уважно розглядати вартість додаткових функцій та перевіряти, чи вони відповідають вашим потребам та бюджету.

2.2 Методики вирішення кожної проблеми та популярні інструменти для вирішення кожної з проблем

Зважаючи на те, що вирішення проблем у розробці веб-сайтів з використанням вебсайт-білдерів може бути різноманітним, ось докладніші методики для кожної з наведених раніше проблем:

Обмежена гнучкість та кастомізація:

Методи вирішення: В ряді випадків можливо внести додаткові налаштування або використовувати власний CSS/JS код. Але, в будь якому з цих випадків бізнес не

зможе налаштувати все швидко, без розробників. Не буде вистачати цільових компонентів для вебсайту які потрібні під конкретну бізнес задачу, таку як вебсайт для оренди нерухомості.

Швидкість та оптимізація:

Методи вирішення: Оптимізація зображень, використання кешування, обмеження використання великої кількості вбудованих елементів. Вибір шаблону, який добре оптимізований. За оптимізацію та швидкість буде відповідати фреймворк повер бібліотеки React.js - Next.js. Основний слоган звучить так “Next.js використовується деякими з найбільших світових компаній, Next.js дає змогу створювати повний стек веб-додатків, розширюючи найновіші функції React та інтегруючи потужні інструменти JavaScript на основі Rust для найшвидшої збірки.” Next.js — це популярний фреймворк для розробки веб-додатків на базі React. Next.js — потужний фреймворк для розробки веб-додатків на React, відомий своєю універсальністю, вбудованою підтримкою Server-Side Rendering, розширеною системою маршрутизації та зручною обробкою даних на сервері. Його автоматична оптимізація зображень, підтримка HMR та інтеграція з TypeScript роблять його привабливим вибором для розробників, спрямованих на ефективний розвиток сучасних веб-додатків. Фішки Next.js на скріншоті головного вебсайту фреймворку на рисунку 2.1.

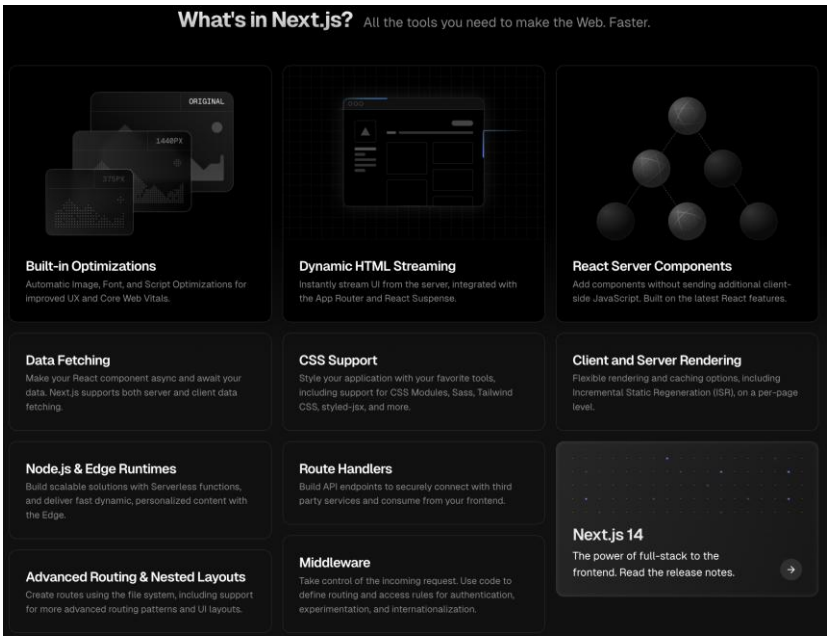

Рис. 2.1 – основні функції Next.js

Сумісність із сторонніми сервісами:

Next.js демонструє високу сумісність із різноманітними сервісами, що робить його привабливим вибором для розробки веб-додатків, які вимагають інтеграції з іншими зовнішніми сервісами та інструментами.

Однією з ключових переваг Next.js є його здатність працювати як із статичним, так із динамічним контентом, що дозволяє легко інтегрувати його з різними Content Management Systems (CMS), такими як WordPress або Drupal. Така гнучкість спрощує спільну роботу розробників та редакторів контенту.

Засоби маршрутизації Next.js дають можливість легко інтегрувати додаток зі сторонніми сервісами API. Це особливо корисно для розробки веб-додатків, які використовують дані зовнішніх джерел, таких як соціальні мережі, сервіси оплати, або будь-які інші API. Також, Next.js має вбудовану підтримку для роботи з серверними функціями, що дає змогу легко інтегрувати серверний логіку, наприклад, з серверними функціями в облаках, такими як AWS Lambda чи Vercel Functions. До того ж, Next.js добре взаємодіє з іншими популярними бібліотеками та фреймворками, такими як Redux для керування станом додатку, або Apollo Client для роботи з GraphQL. Це забезпечує розробникам гнучкість у виборі та інтеграції додаткових інструментів у свої проекти.

Узагалі, Next.js робить акцент на простоті інтеграції, що робить його високо сумісним з різноманітними сервісами та забезпечує ефективну розробку та масштабування веб-додатків.

Обмежені можливості SEO:

Використання Server-Side Rendering. Server-Side Rendering (SSR) в Next.js представляє собою механізм, який дозволяє виконувати рендеринг React-компонентів на сервері перед відправленням вмісту на клієнтський браузер. Це забезпечує кілька ключових переваг для веб-додатків. Це відбувається на сервері, що полегшує рендеринг сторінки та забезпечує її повноцінність ще до того, як вона

буде відправлена на клієнтський браузер. SSR вирішує кілька ключових проблем, зокрема, сприяє поліпшенню індексації пошуковими системами, забезпечує швидке завантаження сторінок та дозволяє кешувати результати, що сприяє покращенню продуктивності. У Next.js використання SSR досить просте завдяки вбудованим інструментам. Зокрема, використання компонента getServerSideProps дозволяє визначити дані для серверного рендерингу сторінки, а результати вже відправляються клієнтському браузеру разом із заповненим контентом. Це робить SSR в Next.js потужним інструментом для розробки веб-додатків з високою продуктивністю та ефективністю в області SEO.

Вартість та власні ресурси:

Вартість створення вебсайту для конкретного домену буде мінімальною, все налаштоване “з коробки” та не потребує роботи програмістів. В зручному та зрозумілому інтерфейсі потрібно вибрати компоненти для вебсайту, розмістити їх в секціях та зберегти зміни.

Обмежена масштабованість:

Все налаштоване “з коробки”, основні функції вебсайту вже включені в систему тому що вона розроблялась конкретно під один домен. Додавання нових компонентів не буде займати багато часу і вони не будуть заважати роботі інших вузлів системи.

Безпека:

Використання протоколів https, захищені бази даних, зашифровані паролі, налаштування додаткового захисту серверів мають зупинити атаки та надійно захищати дані користувачів.

Проблеми із сумісністю браузерів:

Next.js вирішує проблеми з сумісністю всіх браузерів за допомогою ряду підходів, спрямованих на забезпечення оптимальної роботи та вигляду веб-додатку на різних браузерах. Одним з таких підходів є автоматична префіксація CSS, коли

Next.js автоматично додає вендорні префікси до CSS-стилів для забезпечення сумісності із старішими браузерами.

Також, використання Babel для транспіляції JavaScript коду дозволяє Next.js конвертувати сучасний код в формат, який може бути виконаний в різних браузерах. Використання поліфілів допомагає підтримувати стандартні JavaScript функції в тих браузерах, які можуть не мати повноцінної підтримки. Важливим елементом є статичний аналіз коду, який дозволяє виявляти можливі проблеми, пов'язані із сумісністю браузерів, та надавати рекомендації щодо їх вирішення. Ці стратегії дозволяють автоматично адаптувати веб-додаток до різних умов та забезпечувати надійний та сприятливий досвід для користувачів незалежно від їхнього вибору браузера.

Вартість додаткових функцій:

Так як система вже розроблена під один домен то додаткові функції можуть не знадобитись взагалі. Але, в будь якому випадку додавання новий функцій в систему не буде займати багато часу, а це значить що їх вартість буде невеликою.

Ці методи вирішення можуть бути адаптовані в залежності від конкретної ситуації та вимог проекту. У процесі дипломної роботи важливо глибоко дослідити кожну проблему, розглянути доступні варіанти вирішення та обґрунтувати вибір конкретних методів тому всі ці методи вирішення будуть використані в моєму продукті. У наступному розділі буде описано кожен метод більш детально.

РОЗДІЛ 3.
ЕКСПЕРЕМЕНТАЛЬНА ЧАСТИНА, СТРУКТУРА РОЗРОБЛЮВАНОЇ ПРОГРАМИ, ЇЇ ОПИС ТА ФУНКЦІЇ, ПЕРЕВІРКА МЕТОДИКИ

Проаналізувавши ринок, зваживши всі перечислені методики та всі варіанти вирішення я прийняв рішення розробити свій проект. Публічний вебсайт допоможе бізнесу та компаніям які займаються нерухомістю швидко створювати вебсайти для своїх клієнтів, а з допомогою адмін панелі агенти нерухомості або власники

нерухомості зможуть зручно корегувати контент вебсайту. У таблиці 3.1 можна побачити актуальний список технологій які будуть використані у розробці.

Таблиця 3.1

Список тохнологій

Призначення	**Технологія**
Основна мова програмування Front-end частини	JavaScript (React.js)
Типізація	TypeScript
SSR та CEO	Next.js
Підтримка декількох мов	next-i18next
Бібліотека UI компонентів	Material UI
Білдер функціонал	builder.io
Дизайн	Figma
Валідація	zod
Запити на сервер	react-query

Кінець таблиці 3.1

Продовження таблиці 3.1

Форми	react-hook-form
Адмін панель	react-admin
Карта	Google map API
Основна мова програмування для Back-end	PHP (Laravel)

База даних	MySQL + MongoDB

3.1 Основні вимоги до вебсайту та приклади з дизайну деяких вузлів системи:

В інтернет-світі, де кожен клік визначає враження користувача, створення вдачного вебсайту стає ключовим завданням для будь-якого бізнесу. У цьому блоку ми розглянемо основні вимоги до вебсайту, який виконує роль білдера темплейтів для оренди нерухомості, схожого на популярний сервіс, такий як booking.com.

Вибір нерухомості — це важливий етап планування подорожі, та вирішальний для комфортного перебування. Від користувача вимагається не лише зручність в навігації, але й можливість індивідуалізації та адаптації обраного варіанту під його потреби та уподобання.

Мій блок "Основні вимоги до вебсайту" пропонує глибокий розгляд того, як важливо забезпечити максимальний рівень зручності для користувачів, які обирають місце для проживання. Ми розглянемо ключові аспекти, включаючи дизайн, функціональність, інтерактивність та можливості кастомізації темплейтів, щоб кожен відвідувач міг знайти ідеальну пропозицію, адаптовану до його унікальних потреб.

Підтримка декількох з основних мов

У світі глобальних можливостей та різноманіття користувачів, важливою складовою вебсайту є його доступність для широкого кола аудиторії. Підтримка мов — це не просто технічний аспект, але й стратегічна можливість розширити географію аудиторії та забезпечити зручність для користувачів різних національностей. У цьому підблоці ми розглянемо важливість і переваги мультиязикової підтримки, а також технічні аспекти впровадження цієї функціональності. Ми покажемо, як створення вебсайту, який може працювати на різних мовах, не лише розширює аудиторію, але й покращує загальний користувацький досвід, роблячи його максимально інклюзивним та зручним для

кожного відвідувача. Підтримка мов на публічному вебсайті буде розроблена у вигляді модального вікна яке можна побачити на рисунку 3.1.

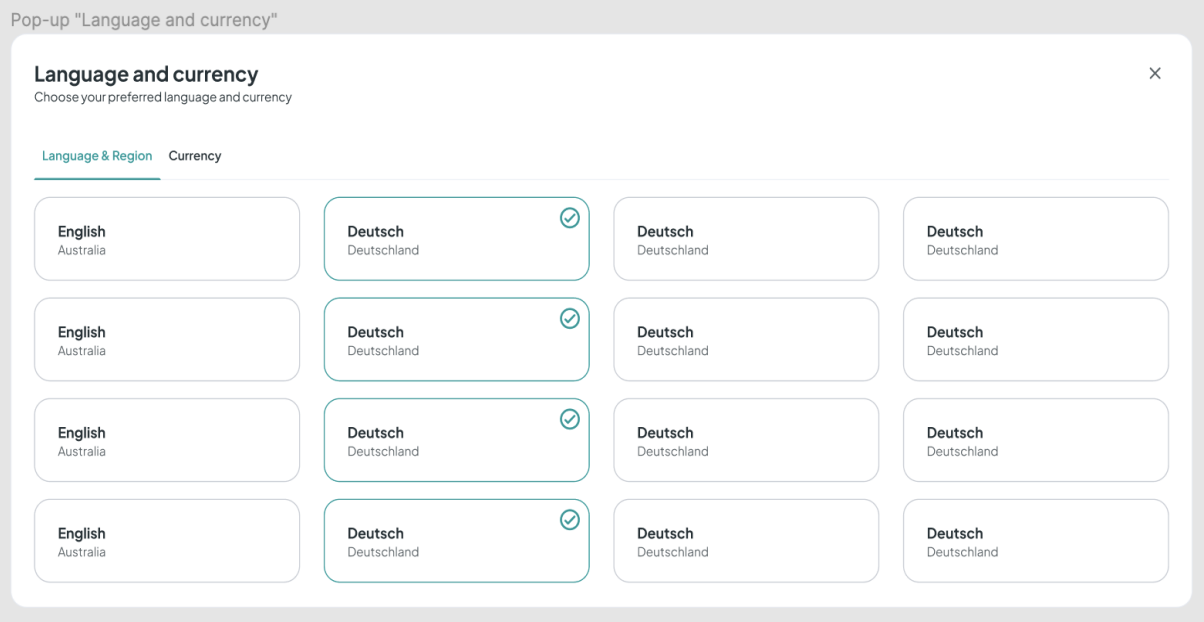

Рис. 3.1 - дизайн модального вікна мов які підтримуються на вебсайті

Гнучкі фільтри

Буде надано можливість швидко та ефективно фільтрувати і знаходити найвідповідніші пропозиції стає ключовим аспектом успіху будь-якого вебсайту. Гнучкі фільтри створюють можливість персоналізації пошуку відповідно до конкретних потреб користувача. Ми дослідимо, як створення різноманітних фільтрів, таких як тип нерухомості, цінові діапазони, зручності та інші, сприяє покращенню користувацького досвіду та допомагає зробити пошук ідеального місця для проживання швидким та ефективним. Дизайн фільтрів які будуть розроблені в модільному вікні можна побачити на рисунку 3.2

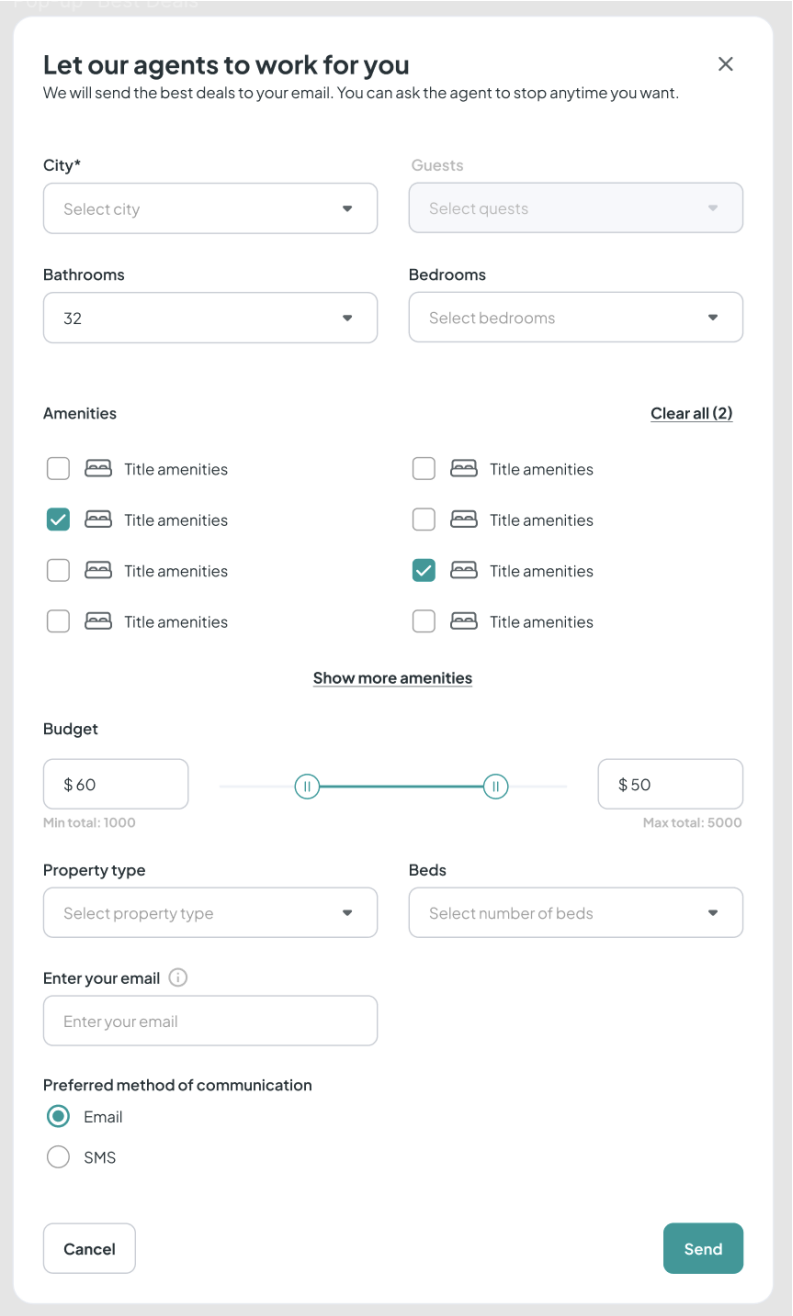

Рис. 3.2 - дизайн модального вікна фільтрів які підтримуються на вебсайті

Підтримка декілької валют

В умовах глобального ринку та різношерстості користувачів, можливість використовувати кілька валют на вебсайті для оренди нерухомості є ключовою характеристикою, яка сприяє зручності та гнучкості. У цій дипломній роботі я дослідив важливість цієї функціональності та її позитивний вплив на користувацький досвід. Забезпечення можливості обирання різних валют

користувачами сприяє міжнародному розширенню та зручності для подорожуючих з усього світу. Ми розглянемо, як інтеграція системи підтримки декілької валют допомагає користувачам отримувати інформацію та розраховувати вартість оренди в зручний для них спосіб. Цей підблок дозволить нам розглянути технічні аспекти впровадження підтримки декілької валют, а також виокремить стратегічні переваги, які це принесе вашому вебсайту в сфері оренди нерухомості, роблячи його більш привабливим та доступним для глобальної аудиторії. До того ж, кожна компанія зможе налаштовувати кількість валют для свого вебсайту окремо. Модальне вікно з валютами можна побачити на рисунку 3.3.1.

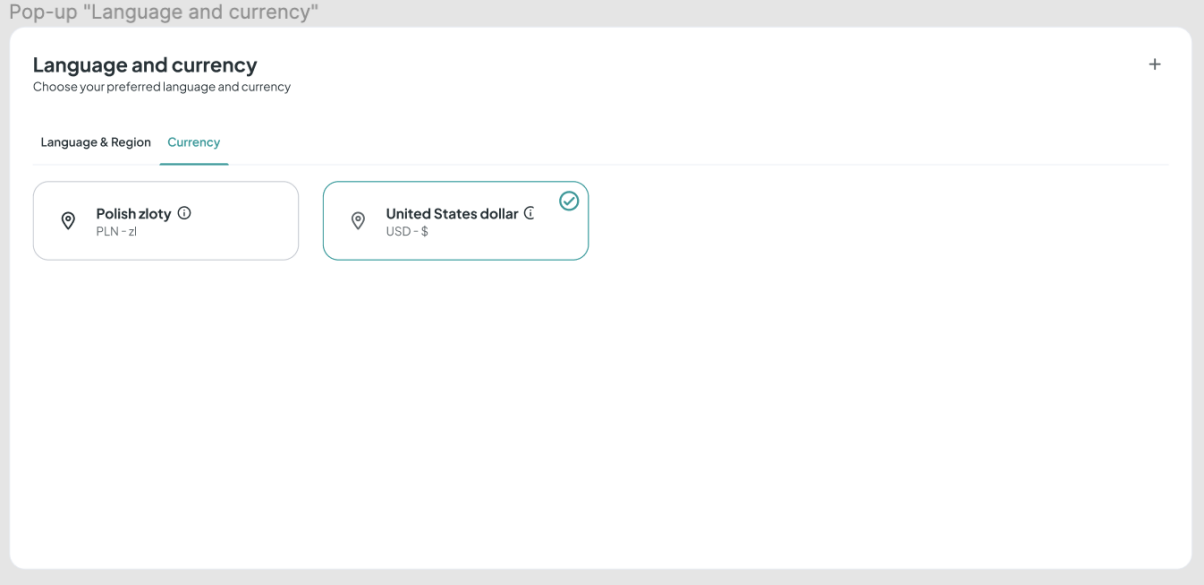

Рис. 3.3 - дизайн модального вікна вибору валюти яка підтримується на вебсайті

Мапа нерухомості

Додавання інтерактивної карти з відображенням лістінгів нерухомості стає потужним інструментом для зручності користувачів та покращення їхнього досвіду на вебсайті. Інтерактивна карта дозволяє користувачам візуально оцінити розташування доступних лістінгів, забезпечуючи їм повну картину регіону чи міста. Ми розглянемо, як це може полегшити процес вибору нерухомості, забезпечуючи детальний погляд на її місцезнаходження та оточення. Додавання мапи стає важливим кроком для забезпечення повноти та зручності взаємодії з вашим вебсайтом в галузі оренди нерухомості. Дизайн мапи можн побачити на рисунку 3.4.

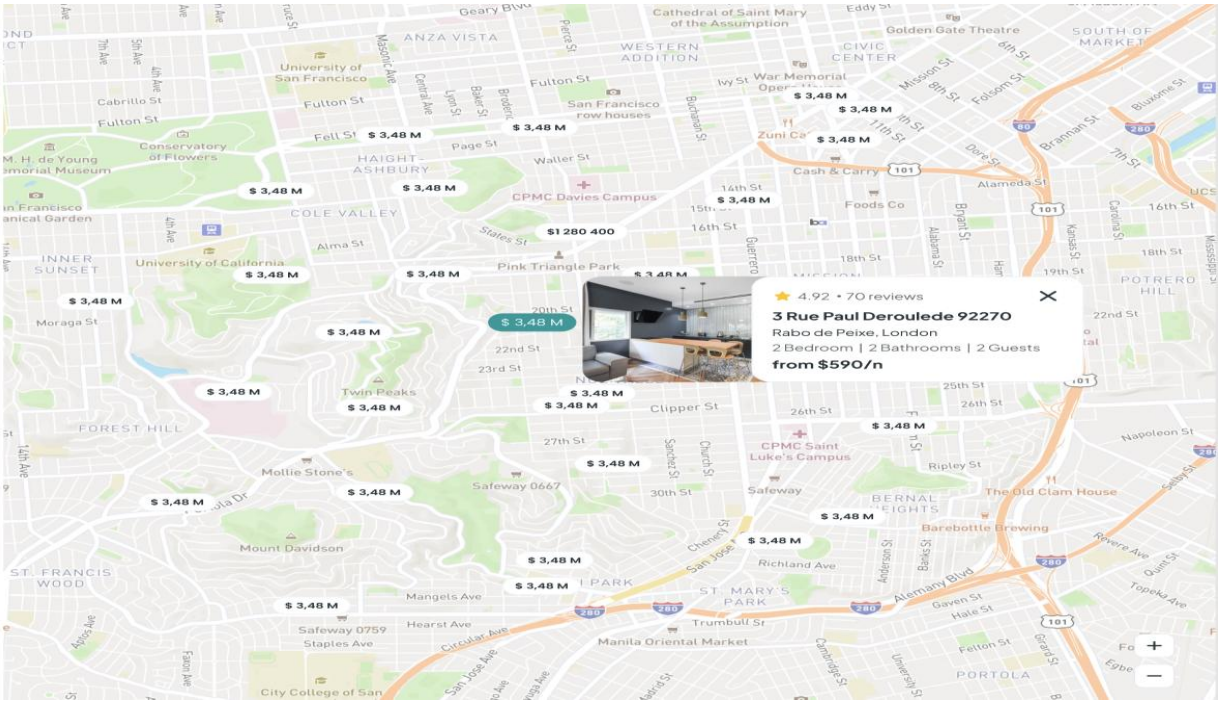

Рис. 3.4 - дизайн карти

Відгуки відвідувачів

Відгуки відвідувачів відіграють важливу роль у вирішенні користувачами питання про надійність та якість надання послуг на вебсайті з оренди нерухомості. Відгуки стають джерелом реальних вражень користувачів, що може допомогти іншим визначити, чи відповідає конкретний лістінг їхнім потребам і очікуванням. Дизайн блоку відгуків можна побачити на рисунку 3.5.

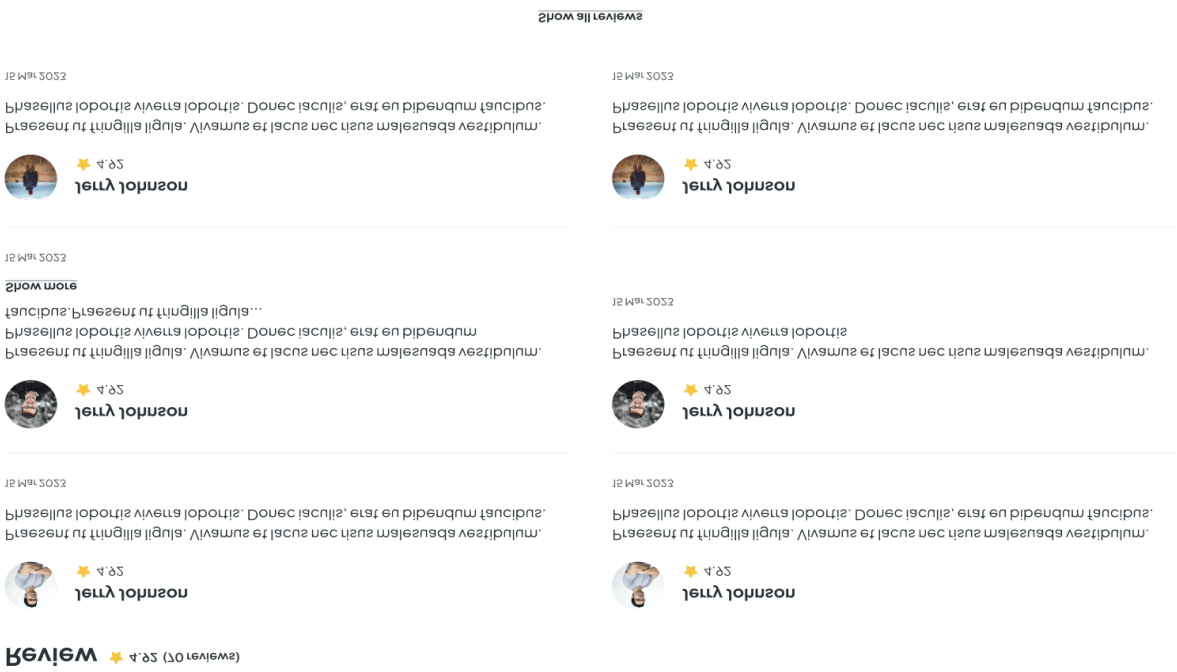

Рис. 3.5 - дизайн блоку відгуків

Галерея картинок

Галерея фотографій лістингів стає важливим інструментом для привертання уваги користувачів та демонстрації привабливості нерухомості. Візуальний контент грає визначальну роль у виборі користувачів, тому галерея фотографій дозволяє надати повний та привабливий огляд нерухомості. Дизайн галереї можна побачити на рисунку 3.6.

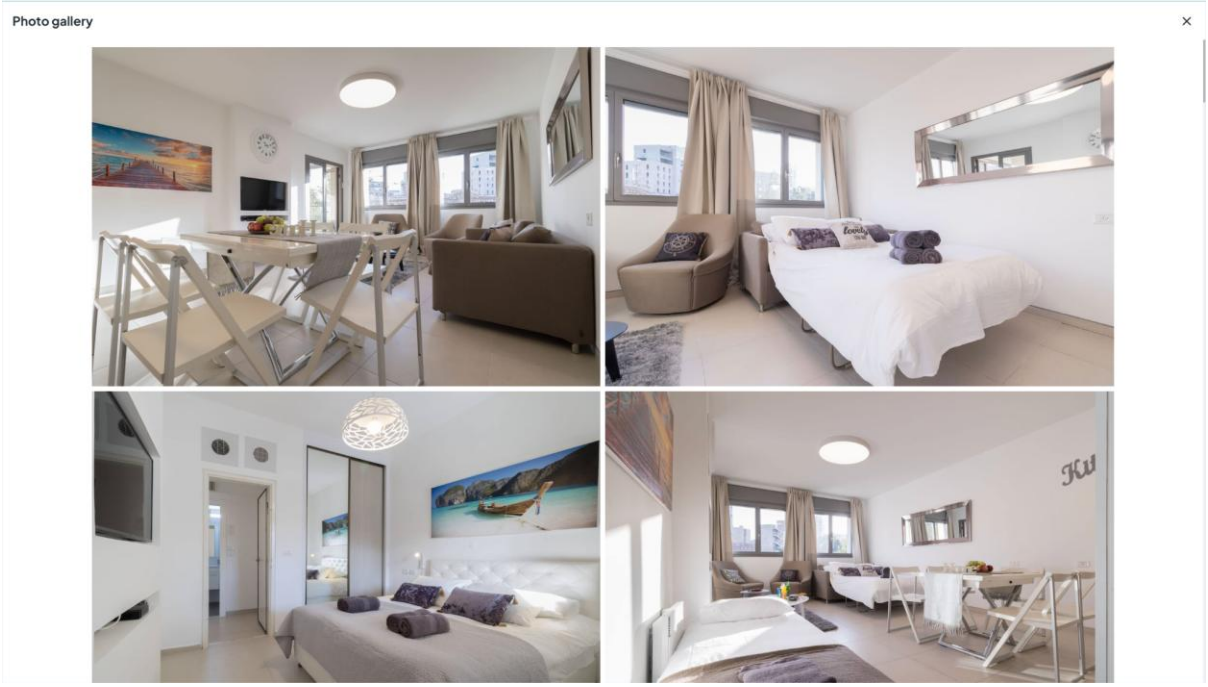

Рис. 3.6 - дизайн галереї

Форма оплати та додавання купонів на знижку

Форма оплати та використання купону є ключовими елементами для зручності та привабливості для користувачів вашого вебсайту з оренди нерухомості. Форма оплати повинна бути простою, безпечною та відповідати різним вимогам користувачів. Я буду використовувати такий метод оплати як Stripe щоб забезпечити надійність та ефективність процесу оплати. Використання купонів може стати потужним інструментом маркетингу та залучення нових користувачів. Ми розглянемо стратегії надання та використання купонів, а також технічні аспекти їх обробки на вебсайті. З формою оплати можна ознайомитись на рисунку 3.7 Форму введення купонів можна побачити на рисунку 3.8.

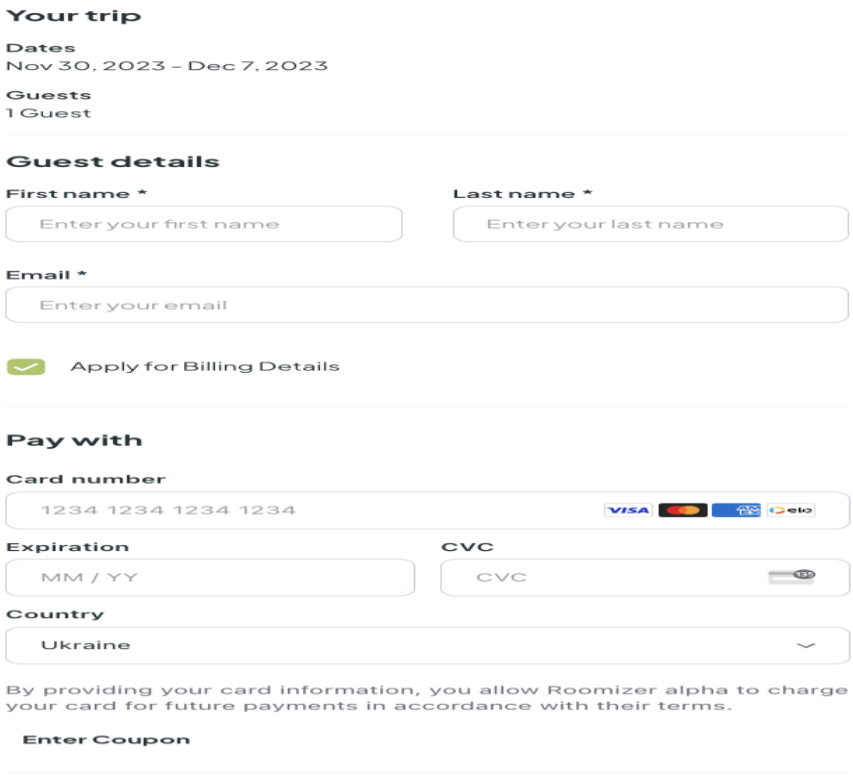

Рис. 3.7 - дизайн форми оплати

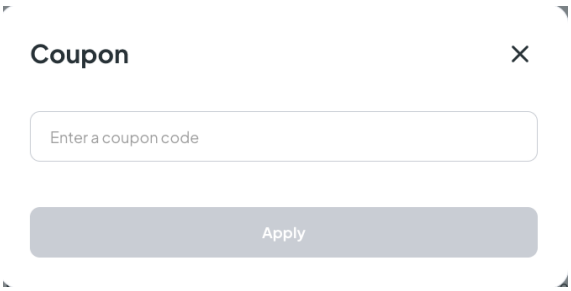

Рис. 3.8 - дизайн форми оплати

3.2 Основні сторінки адмін панелі:

В адмін панелі можна гнучко налаштовувати вебсайт під свої вимоги. Розглянемо декілька з основних сторінок адмін панелі та її основні функції.

1. Сторінка створення вебсайту. Це перша сторінка яка буде зустрічати користувача адмін панелі. Без налаштування вебсайту неожна перейти до наповнення. Тут адміністратор має додати домен, деякі мета заголовки для покращення роботи пошукових роботів, завантажити логотипи та обрати основний колір вебсайту. Цю сторінку можна побачити на рисунку 3.9.

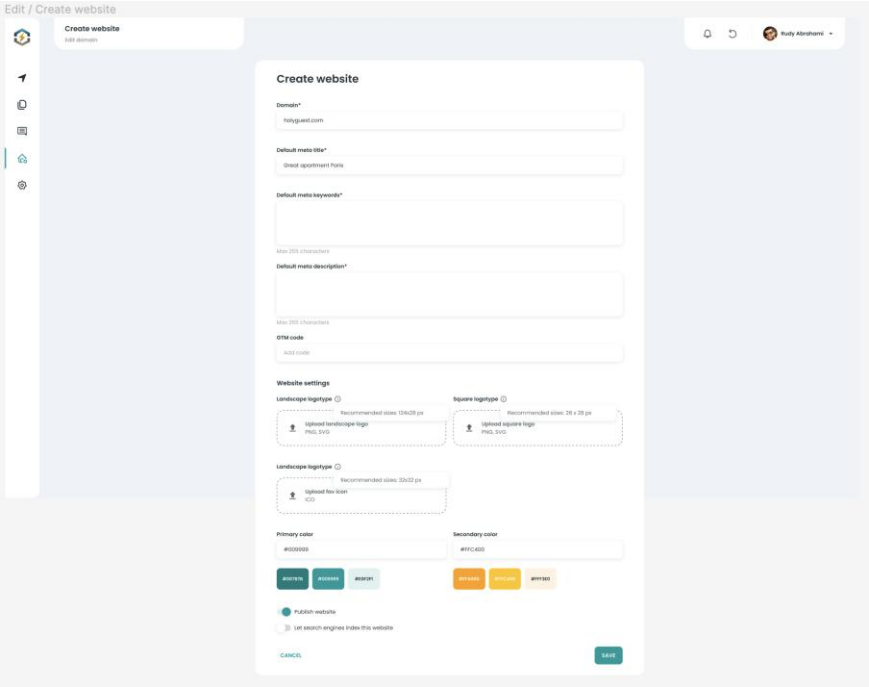

Рис. 3.9 - дизайн сторінки створення вебсайту

2. Сторінка навігаторів це сторінка це користувач (власник, адміністратор) може обрати та налаштувати меню публічного вебсайту, а також додати свої власні айтеми які будуть вести на сторонні ресурси. Дизайн цієї сторінки можна побачити на рисунку 3.10.

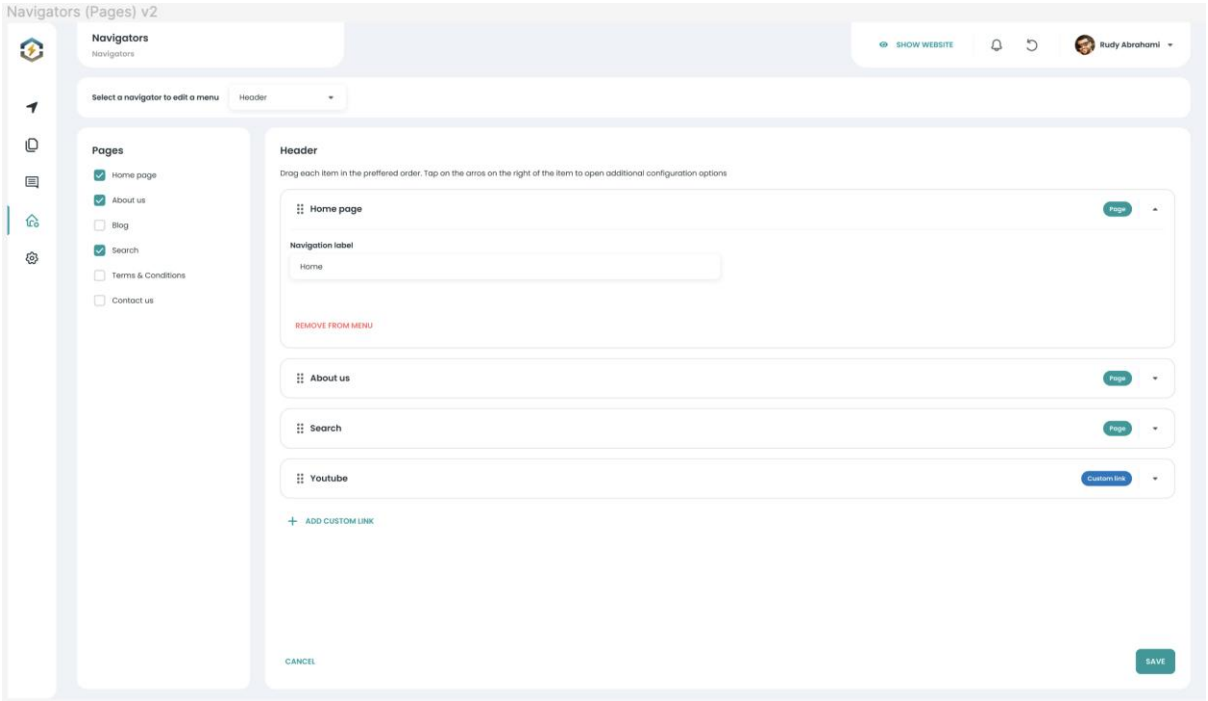

Рис. 3.10 - дизайн сторінки навігаторів

3. Кастомізація сторінок публічного вебсайту. Тут адміністратор може кастомізувати свої сторінки, ті які використовують білдер - можуть бути гнучко налаштовані тут. Для того щоб зрозуміти яка сторінка повністю білдер сторінка було вирішено використовувати іконки з червоними та зеленими кольорами. Цю сторінку можна побачити на рисунку 3.11. Форму ж створення своєї сторінки можна побачити на рисунку 3.12.

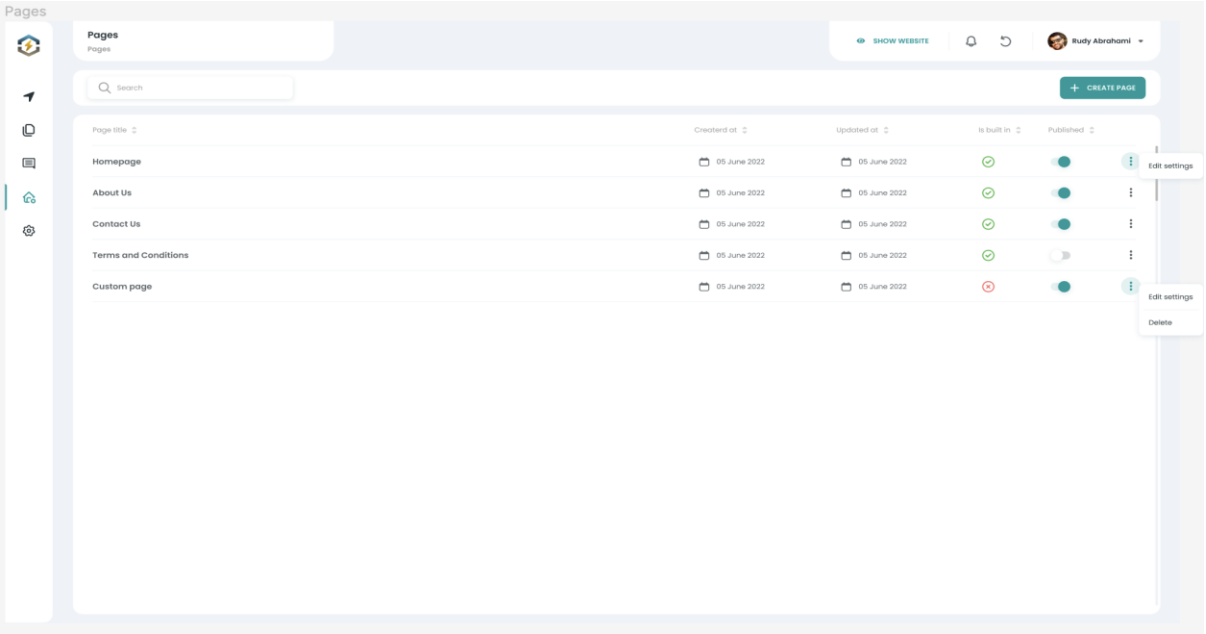

Рис. 3.11 - дизайн сторінки налаштувань сторінок публічного вебсайту

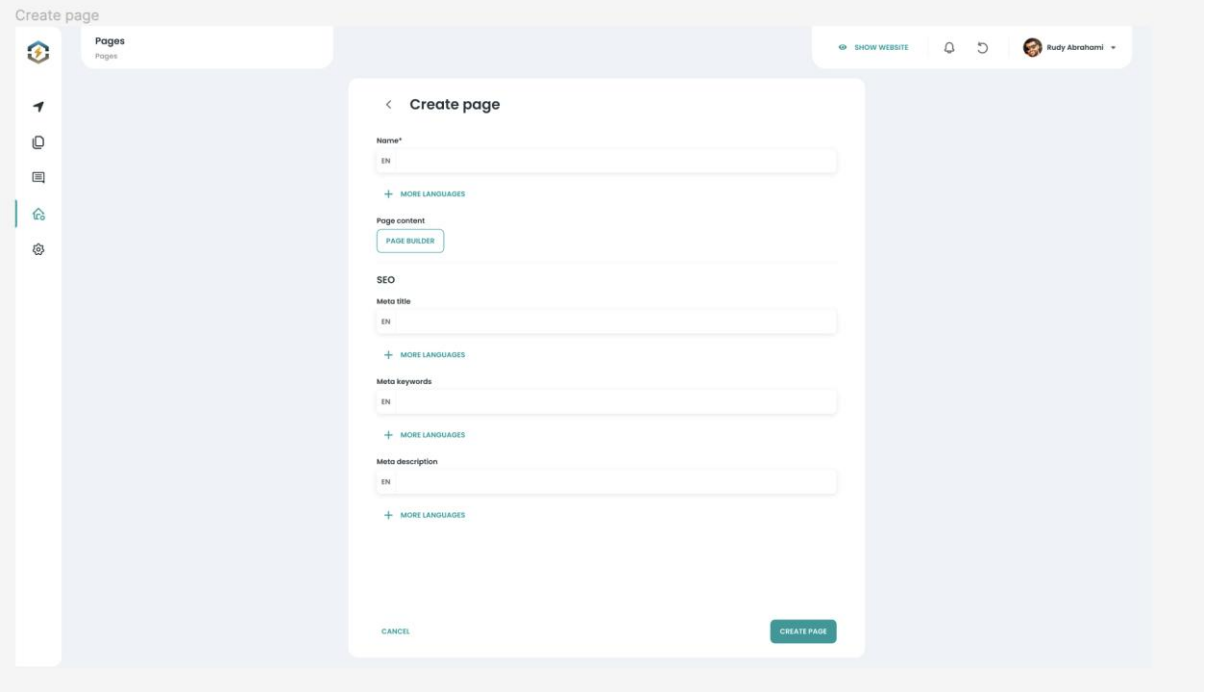

Рис. 3.12 - дизайн сторінки створення сторінки

3.3 Основні сторінки публічного вебсайту:

Перед тим як ознайомитись з переліком сторінок публічного вебсайту слід зазначити що є 2 типи сторінок.

1. Сторінки з використання білдеру. Ці сторінки можна кастомізувати в адмін панелі. Кожному власнику бізнесу можна налаштовувати текст на різних мовах, передвигати секції на сторінках, обирати додаткові налаштування. Приклади сторінок з адмін панелі будуть надані в наступному блоці.
2. Сторінки які мають статичний вигляд та дизайн. Ці сторінки розроблені статично, їх не можна змінювати в білдері, не можна обирати варіант вигляду, змінювати текст або якісь функції.

Тепер пропоную ознайомитись з основними сторінками вебсайту.

Основна сторінка (рис. 3.9):

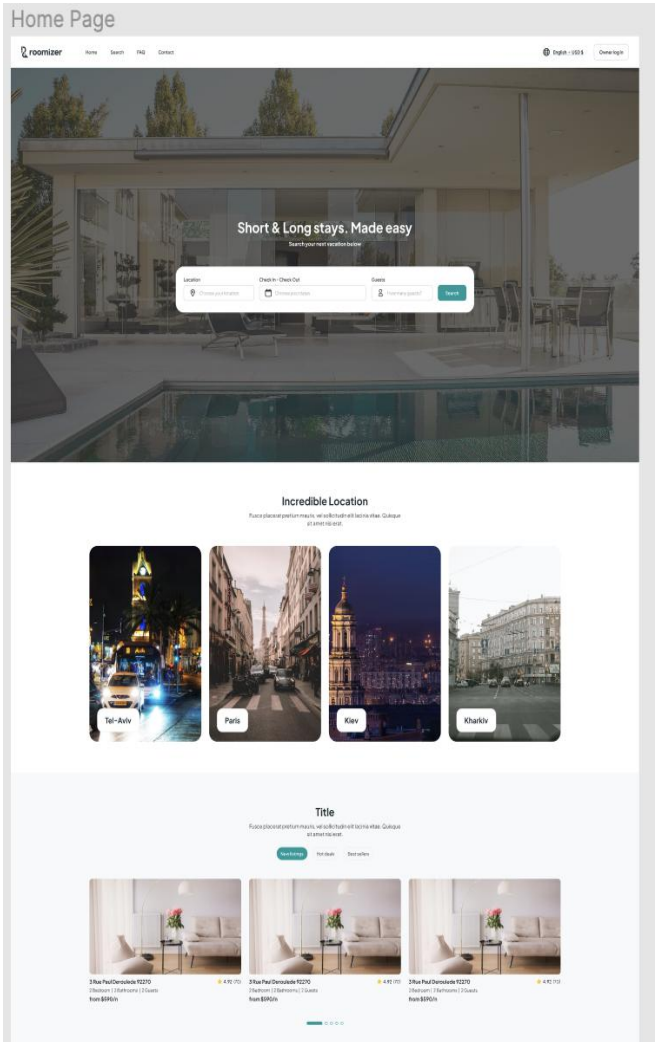

Рис. 3.9 - дизайн головної сторінки

Ціль:

1. Привернення уваги та перші враження.
2. Швидкий доступ до основних функцій сайту.

Ключові елементи:

1. Інтерактивний головний банер з привабливими фотографіями нерухомості.
2. Пошукове поле з основними фільтрами.
3. Рекомендації та популярні лістинги для залучення уваги.
4. Меню навігації для швидкого доступу.

Сторінка пошуку (рис. 3.10):

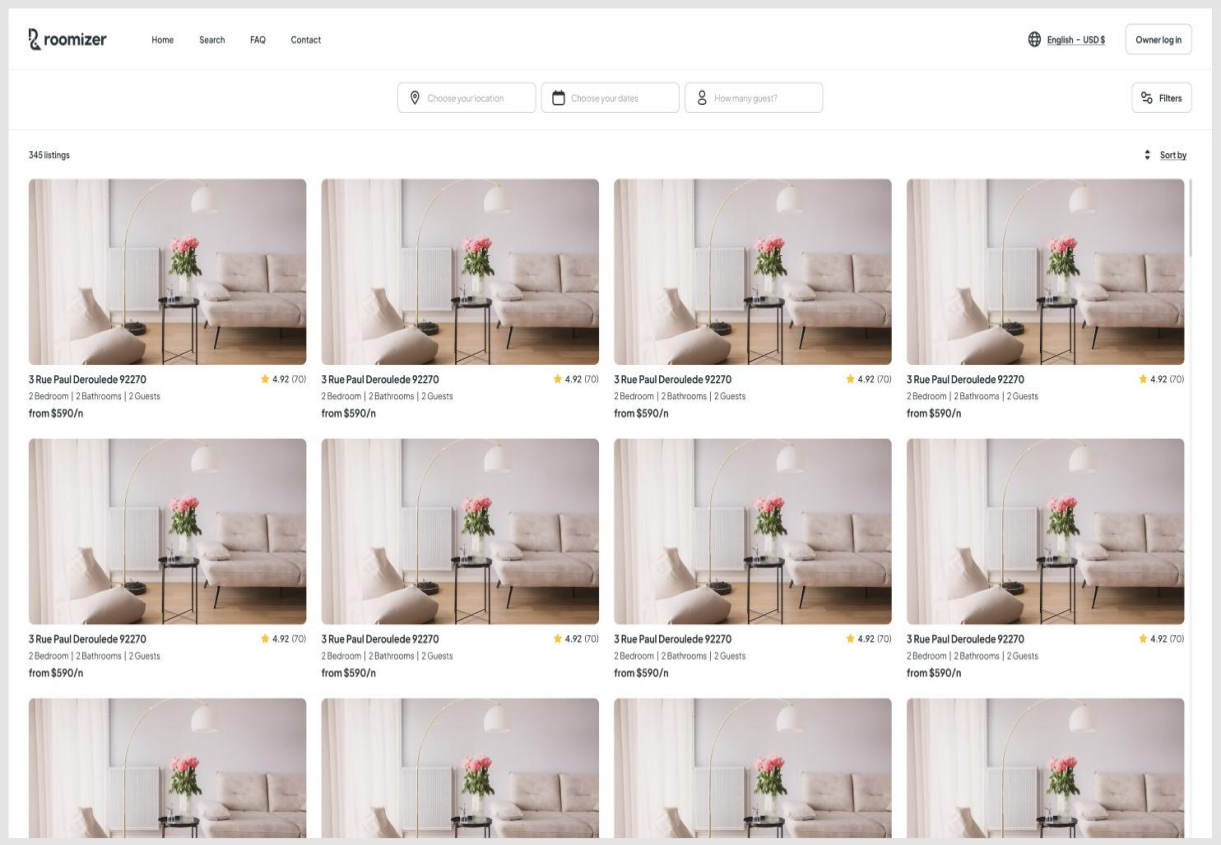

Рис. 3.10 - дизайн сторінки пошуку

Ціль:

1. Ефективний та зручний пошук житла за різними параметрами.
2. Інтерактивна мапа яка допоможе користувачу швидко знайти житло по місцевості.

Ключові елементи:

1. Розширений пошук з фільтрами за типом нерухомості, ціновим діапазоном, зручностями тощо.
2. Інтерактивна карта з лістингами для візуального відображення розташування.
3. Сортування результатів за різними критеріями.

Сторінка зв'язку (рис. 3.11):

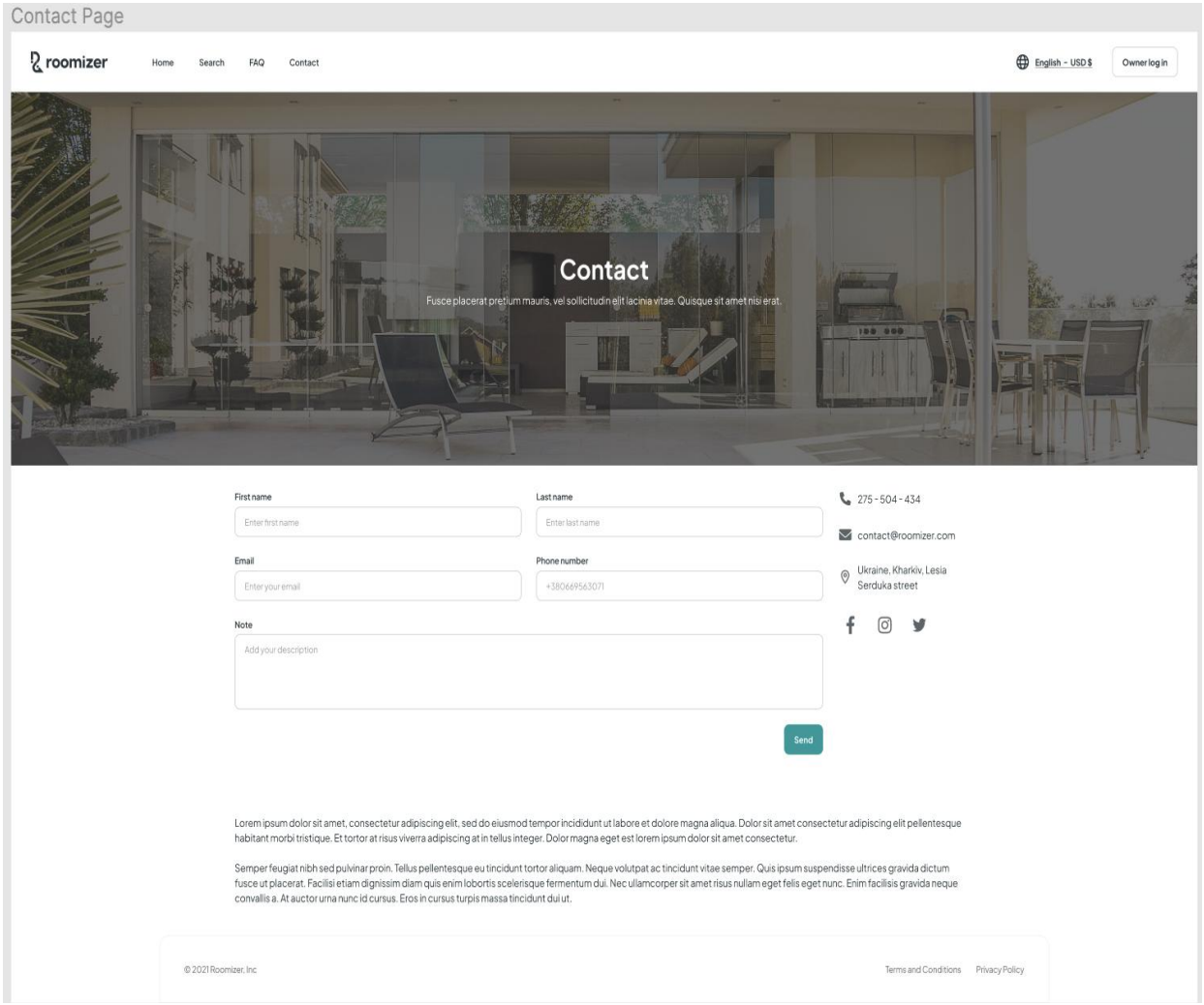

Рис. 3.11 - дизайн сторінки контактів

Ціль:

1. Забезпечення зручного способу зв'язку та отримання додаткової інформації.

Ключові елементи:

1. Форма зв'язку для пошукових запитань або отримання консультації.
2. Контактна інформація та робочий графік.

Сторінка обраного лістингу (рис. 3.12):

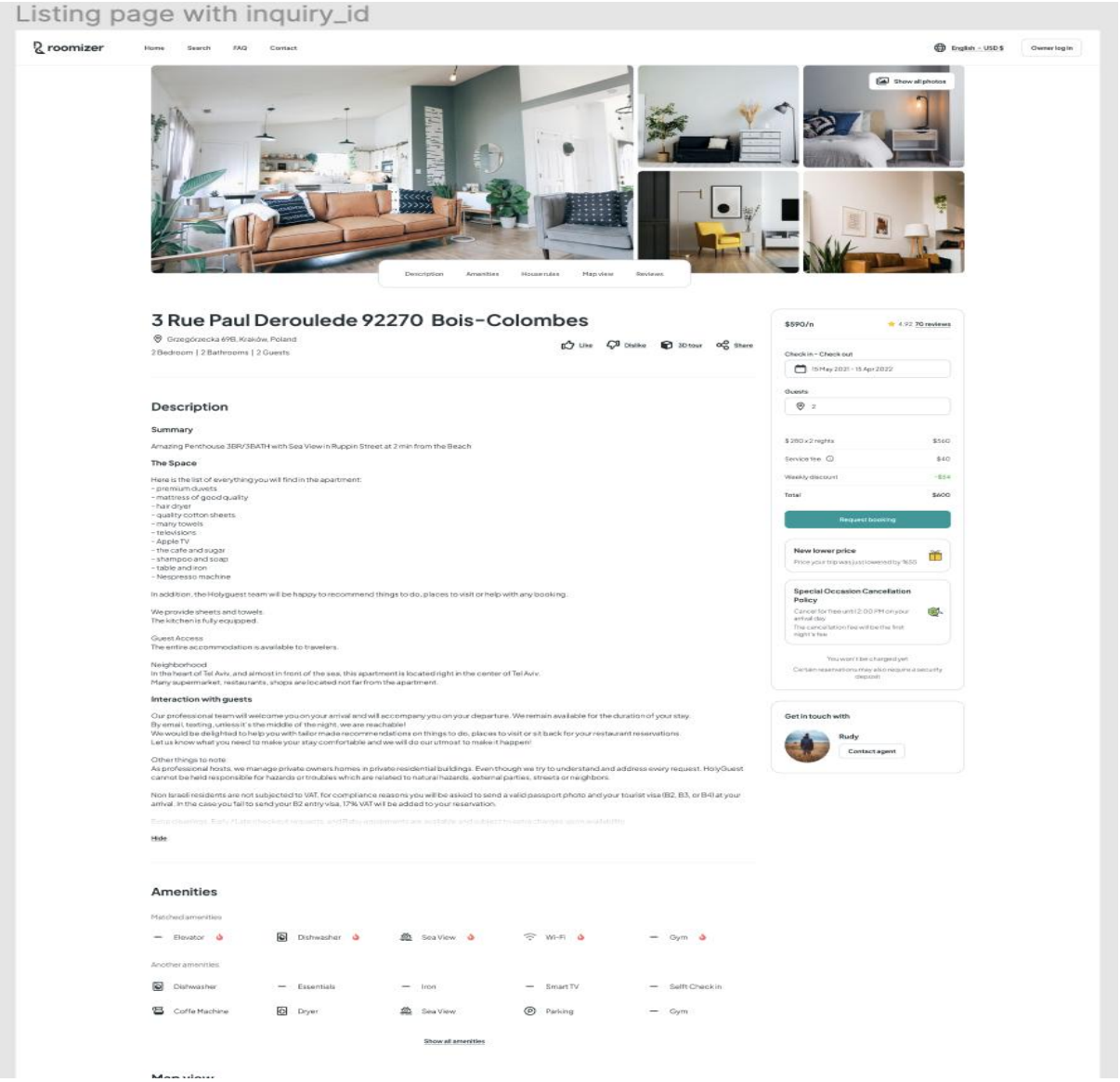

Рис. 3.12 - дизайн сторінки лістингу

Ціль:

1. Детальне вивчення обраного житла та здійснення бронювань.

Ключові елементи:

1. Галерея фотографій для детального перегляду.
2. Інформація про зручності та опис нерухомості.
3. Форма бронювання та вибір опцій.
4. Відгуки та рейтинги користувачів.

Кожна сторінка повинна бути максимально інтуїтивною та користувацьки орієнтованою. Забезпечте гармонійний дизайн та логіку взаємодії, щоб користувачі могли легко знаходити інформацію та здійснювати необхідні дії.

Сторінка успішної оплати (рис 3.13):

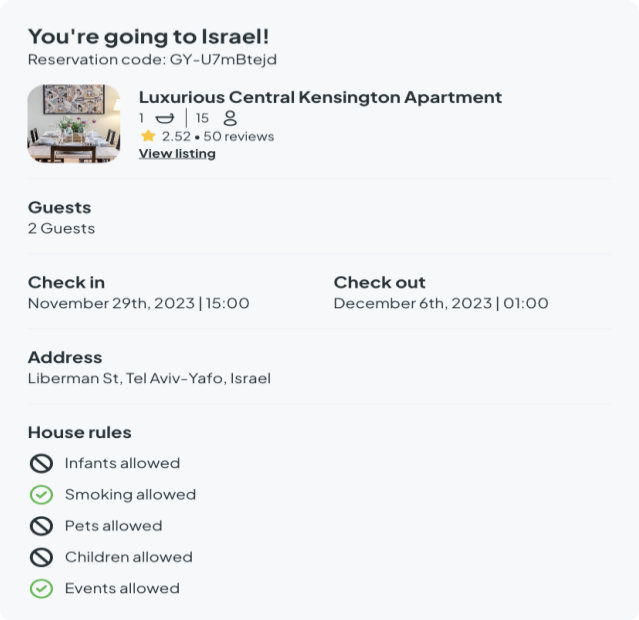

Рис. 3.13 - дизайн сторінки підтвердження оплати

Ціль:

1. Підтвердження успішного завершення оплати та забезпечення додаткової інформації.

Ключові елементи:

1. Повідомлення про успішну оплату та вдячність за користування сервісом.
2. Деталі оплати та підтвердження бронювання.
3. Посилання на сторінку з деталями лістингу та інші корисні ресурси.

Сторінка успішної оплати повинна передавати позитивні емоції та забезпечувати впевненість у користувача щодо вибору та оплати обраного лістингу. Вона також може містити рекомендації або спеціальні пропозиції для подальших дій користувача на сайті.

3.4 Розробка адмін панелі використовуючи фреймворк react-admin:

У світі веб-розробки створення адміністративної панелі, що є ефективним і зручним інструментом для управління вебсайтом, є важливим завданням. React-Admin надає нам потужні інструменти для швидкої та ефективної розробки інтерфейсу адміністратора, дозволяючи фокусуватися на функціональності та досконалості замість базових деталей. У цьому блоці я опишу основні вузли системи з прикладами коду, скріншотами готового результату.

Етап 1: Встановлення та Конфігурація

1. Встановлення React-Admin:

В цьому пункті я створюю пусту папку для майбутньої адмін панелі, вікриваю консоль та вставляю скрипт який створить мені стартовий проект.

```
yarn add react-admin ra-data-json-server prop-types
```

2. Створення основного компонента:

Створюємо базовий компонент для того аби запустити проект.

Код 3.1

Вхідна точка в проект

```
// admin/App.js
import React from 'react';
import { Admin, Resource, ListGuesser } from 'react-admin';
import jsonServerProvider from 'ra-data-json-server';

const dataProvider = jsonServerProvider('тут ми будемо вписувати
бекенд АПІ');
const App = () => (
    <Admin dataProvider={dataProvider}>
        <Resource name="users" list={ListGuesser} />
    </Admin>
);

export default App;
```

Етап 2: Визначення Ресурсів та Компонентів

1. Створення Компонентів для Ресурсів:

В цьому боці потрібно описати справжні ресурси які будуть використані в проекті. Важливо відмітити, що назви ресурсів повинні бути ідентичні назвам ендпоінтів на сервері. В цьому файлі я також використав захист від роботів та ботів. Встановивши та підключивши *react-google-recaptcha-v*
я вирішив цю проблему. Додаємо основні ресурси нашого проекту які можна побачити в таблиці 3.2.

Таблиця 3.2

Основні ресурси вебсайту

Назва ресурсу	**Функції**
/navigators	Отримати навішатори такі як хедер та футер з назвами меню блоків та іхніми посиланнями.
/website	Отримати повну інформацію про вебсайт, ключі капчі, налашьування, колір вебсайту та іншу інформацію.
/pages	Отримати сторінки вебсайту, контент, мета теги та мови.

```
import { ReactElement, useMemo } from 'react';

import Layout from 'components/layouts/Layout';
import { NavigatorsPage } from 'customPages/NavigatorsPage';
import { useGoogleReCaptcha } from 'react-google-recaptcha-v3';
import { Route } from 'react-router-dom';

import { Admin, CustomRoutes, Resource } from '@roomizer/theme';

import * as pages from 'constants/pages';
import * as resources from 'constants/resources';

import { providersCreator } from './providersCreator';
import pagesComponents from './resources/pages';

const ListPlaceholder = (): null => null; // to show custom page
in menu

export function App(): ReactElement {
```

```
  const { executeRecaptcha } = useGoogleReCaptcha();

  const providers = useMemo(() => providersCreator({
executeRecaptcha }), [executeRecaptcha]);

  return (
    <Admin {...providers} layout={Layout}>
      <Resource
        name={resources.NAVIGATORS_RESOURCE}
        list={ListPlaceholder}
        options={{ iconName: 'navigators' }}
      />
      <Resource
        name={resources.PAGES_RESOURCE}
        options={{ iconName: 'pages' }}
        {...pagesComponents}
      />
      <Resource name={resources.SITES_RESOURCE} options={{
iconName: 'settings' }} {...websites} />

      <CustomRoutes>
        <Route path={`/${resources.NAVIGATORS_RESOURCE}`}
element={<NavigatorsPage />} />
      </CustomRoutes>
    </Admin>
  );
}

export default App;
```

2. Створення теми та додавання основних стилей проекту.

Тут я підключаю додаткову бібліотеку Material UI використовуючи скрипт.

```
yarn add @mui/material @emotion/react @emotion/styled
```

Після цього треба налаштувати тему для проекту та підлючити її. В наступному блоці коду я покажу основні моменти з налаштування та підключення. Створюючи тему можна налаштувати індивідуальні відступи, типографію, кольори а також індивідуальні налаштування (`customThemeProps`) можна змержити.

```
const theme = createTheme(
  merge(
    {},
    {
      spacing: SPACING,
      typography,
      palette,
    },
    customThemeProps,
  ),
);
```

Для прикладу, ось такі індивідуальні налаштування теми я використовую в адмін панелі. Це можуть бути різні налаштування для варіантів типографії, кольорів та іншого.

```
import { createTheme, alpha } from '@mui/material/styles';

import { SPACING } from './constants';
import palette from './palette';
import { TCustomThemeProps } from './types';
import typography from './typography';

const theme = createTheme({
  palette,
  spacing: SPACING,
  typography,
});

export const customThemeProps: TCustomThemeProps = {
  dropShadow: {
    compensationWidth: 8, // use it when overflow + boxShadow
    compensations: {
      '40%': {
        x: 10,
        y: 8,
      },
    },
    '40%': '0px 2px 10px rgba(183, 197, 205, 0.4)', // Drop Shadow
40% by figma
    '70%': '0px 2px 10px rgba(183, 197, 205, 0.7)', // Drop Shadow
```

```
70% by figma
    '100%': '0px 2px 10px rgba(183, 197, 205, 1)', // for box-
shadow animation. We'll use 'opacity' to get '40%' and '70%'.
https://tobiasahlin.com/blog/how-to-animate-box-shadow/
  },
  typography: {
    buttonUnderline: {
      fontFamily: theme.typography.fontFamily,
      fontSize: theme.typography.pxToRem(12),
      lineHeight: theme.typography.pxToRem(16),
      fontWeight: 700,
      textDecoration: 'underline',
    },
    small1: {
      fontFamily: theme.typography.fontFamily,
      fontSize: theme.typography.pxToRem(10),
      lineHeight: theme.typography.pxToRem(12),
      fontWeight: 500,
      color: theme.palette.text.light,
    },
    small2: {
      fontFamily: theme.typography.fontFamily,
      fontSize: theme.typography.pxToRem(10),
      lineHeight: theme.typography.pxToRem(12),
      fontWeight: 700,
    },
  },
  palette: {
    text: {
      light: '#5C656A',
    },
    info: {
      background: '#E1F5FE',
    },
    primary: {
      background: '#EFF2F7',
      border: '#C8CDD5',
    },
    other: {
      border: '#E7EBF2',
    },
    actions: {
      main: alpha(theme.palette.primary.main, 0.08),
      secondary: alpha(theme.palette.text.secondary, 0.08),
```

```
      error: alpha(theme.palette.error.dark, 0.08),
      unactive: '#BDBDBD',
      background: '#F7F8FB',
    },
    chart: {
      blueSteel: '#3B5998',
      mint: '#3EB489',
      yellowGreen: '#9ACD32',
      turquoise: '#009999',
      purple: '#816CE7',
      bittersweet: '#F88175',
      brickRed: '#C24451',
    },
  },
};
```

3. Налаштування fetchClient для проекту

FetchClient потрібен для додаткового налаштування запитів на сервер, це допоможе не дублювати код, гнучко маніпулювати запитами.

4. Створення форм

В цьому блоці я покажу одну з основних форм та найважливіших форм проекту. Це PagesForm в якій користувач може налаштувати сторінку. Поля та їх функції писані в таблиці 3.2.

Таблиця 3.2

Поля для форми сторінки

Інпут	**Функції**
title	Назва сторінки
meta-title	Мета тег, який використовується в HTML-документах для визначення заголовка сторінки, який буде відображений у вкладці браузера або на панелі завдань. Заголовок сторінки є важливим елементом для

	пошукової оптимізації (SEO) та зручності користувачів. Дані з цього інпуту будуть встановлені в заголовок сторінки та заголовок вкладки браузера. Він може бути корисним для визначення унікальної та інформативної назви для кожної сторінки вашого вебсайту. Заголовок важливий для SEO, оскільки часто використовується пошуковими двигунами при індексації та ранжуванні веб-сторінок.
meta-keywords	Вважається не дуже ефективним, оскільки багато пошукових двигунів ігнорує цей тег, а деякі навіть можуть використовувати його для визначення спаму. Замість цього, пошукова оптимізація зараз спрямована на використання вмісту сторінки, який надає значення та інформацію для користувачів, а також на використання інших елементів SEO однак ми будемо використовувати та підтримувати його
meta-descriptions	Дані з цього інпуту можуть бути використані пошуковими двигунами для відображення опису сторінки в результатах пошуку. Добре написаний та релевантний опис може покращити клацання на сторінку, оскільки користувачі можуть зрозуміти, чи відповідає контент їхнім потребам.

	Однак важливо враховувати, що Google та інші пошукові двигуни можуть вирішити використовувати власний опис сторінки, який вони вважають більш відповідним для конкретного запитання користувача. Тому важливо створювати унікальний та важливий опис для кожної сторінки вашого вебсайту
builder-input	Це кнопка по кліку на яку відкривається iframe де користувач може налаштовувати під себе вебсайт, додавати текст та багато іншого.

Код форми можна побачити нижче.

```
import { FC, useCallback, useMemo } from 'react';

import { zodResolver } from '@hookform/resolvers/zod';
import { Grid, Typography, Box, IconButton } from '@mui/material';
import get from 'lodash/get';
import { useQueryClient } from 'react-query';
import { useNavigate } from 'react-router-dom';
import { z, ZodSchema } from 'zod';

import {
  FormPaper,
  FormPaperClasses,
  GridWrapper,
  styled,
  StyledRootProps,
  useTranslate,
  ValidationForm,
  ValidationFormProps,
  LabelWrapper,
  useGetLocales,
  useMainLocale,
  useRecordContext,
  useResourceContext,
```

```
  RaMultiLanguageInput,
  zodIdSchema,
  getLocalizedItemsSchema,
  useGetLanguages,
  useSaveContext,
  IcoMoon,
} from 'react-admin';

import { PAGES_RESOURCE } from 'constants/resources';

import useGetSitePagesShowResponseRecordSchema from
'./api/hooks/useGetSitePagesShowResponseRecordSchema';
import { SOURCES } from './constants';
import { PagesToolbar } from './PagesToolbar';

import { BuilderInput } from '../../components/builder/BuilderInput';
import { useGetSitesList } from '../sites';

const PREFIX = 'PagesForm';

export const PagesFormClasses = {
  title: `${PREFIX}-title`,
};

const StyledPagesForm = styled('div', {
  name: 'StyledPagesForm',
  label: `Website-panel--${PREFIX}`,
})(({ theme }) => {
  return {
    [`&& .${FormPaperClasses.root}`]: {
      padding: theme.spacing(3, 6),
    },
  };
});

interface GetValidationSchemaParams {
  locales: string[];
  mainLocale: string;
}

const getValidationSchema = (params: GetValidationSchemaParams):
ZodSchema => {
  const { locales, mainLocale } = params;

  const textAreaMaxSymbols = 191;

  const localizedItemsSchema = getLocalizedItemsSchema({
    locales,
```

```
    mainLocale,
    schema: z.object({
      [SOURCES['title']]: z.string().min(1).max(textAreaMaxSymbols),
      [SOURCES['meta_title']]:
z.string().min(1).max(textAreaMaxSymbols),
      [SOURCES['meta_keywords']]:
z.string().min(1).max(textAreaMaxSymbols),
      [SOURCES['meta_description']]:
z.string().max(textAreaMaxSymbols).optional(),
      [SOURCES['content']]: z.string().optional(),
    }),
    optionalSchema: z
      .object({
        [SOURCES['title']]: z.string(),
        [SOURCES['meta_title']]: z.string(),
        [SOURCES['meta_keywords']]: z.string().max(textAreaMaxSymbols),
        [SOURCES['meta_description']]:
z.string().max(textAreaMaxSymbols),
        [SOURCES['content']]: z.string(),
      })
      .partial(),
  });

  return z
    .object({
      [SOURCES['content_files']]: z.array(z.object({ id: zodIdSchema
})).optional(),
    })
    .merge(localizedItemsSchema);
};

export type PagesFormProps = ValidationFormProps & StyledRootProps;

export const PagesForm: FC<PagesFormProps> = (props) => {
  const { sx, className, isEdit = false } = props;
  const resoure = useResourceContext(props);
  const record = useRecordContext(props) || {}; /* It may be undefined
*/
  const translate = useTranslate({ basePath: 'resources.site-pages' });
  const navigate = useNavigate();
  const queryClient = useQueryClient();
  const locales = useGetLocales();
  const mainLocale = useMainLocale();

  const { save } = useSaveContext();

  const { data: sites } = useGetSitesList();
  const languageId = get(sites, '0.company.default_language_id');
```

```
  const { data: languagesData } = useGetLanguages();

  const defaultLocale = useMemo(() => {
    const defaultLanguage = languagesData?.find((language) =>
language.id === languageId);

    return defaultLanguage?.locale;
  }, [languageId, languagesData]);

  const sitePageShowDataSchema =
useGetSitePagesShowResponseRecordSchema();

  const handleCloseForm = useCallback(() => {
    navigate(`/${PAGES_RESOURCE}`);
  }, [navigate]);

  const handleBuilderInputClick = useCallback(() => {
    // Refresh page data to get new signatures for images
    queryClient.invalidateQueries([resoure]);
  }, [queryClient, resoure]);

  if (!locales || !mainLocale || !sitePageShowDataSchema) return null;

  const isCreate = !isEdit;
  const parsedPageRecord = sitePageShowDataSchema.safeParse(record);

  const isCustomizable =
    isEdit && parsedPageRecord.success &&
!!parsedPageRecord.data['is_customizable'];

  const withContent: boolean = isCreate || isCustomizable;

  return (
    <StyledPagesForm sx={sx} className={className}>
      <GridWrapper container justifyContent="center">
        <Grid item xs={12} md={6} lg={6}>
          <FormPaper
            title={
              <Box sx={{ display: 'flex', alignItems: 'center' }}>
                <IconButton onClick={handleCloseForm}>
                  <IcoMoon icon="arrow-left" />
                </IconButton>
                <Typography variant="h2" sx={{ textTransform:
'capitalize', ml: 2 }}>
                  {isEdit ? translate('labels.edit_page') :
translate('labels.create_page')}
                </Typography>
```

```
              </Box>
            }
          >
            <ValidationForm
              toolbar={<PagesToolbar onClose={handleCloseForm}
isEdit={isEdit} />}
              resolver={zodResolver(getValidationSchema({ locales,
mainLocale }))}
              {...props}
            >
              <GridWrapper container>
                <Grid item xs={12}>
                  <RaMultiLanguageInput
                    source={SOURCES['title']}
                    label={translate('labels.name')}
                    isRequired
                  />
                </Grid>
                <Grid item xs={12}>
                  <RaMultiLanguageInput
                    source={SOURCES['meta_title']}
                    label={translate('labels.meta_title')}
                    isRequired
                  />
                </Grid>
                <Grid item xs={12}>
                  <RaMultiLanguageInput
                    source={SOURCES['meta_keywords']}
                    inputProps={{ multiline: true }}
                    label={translate('labels.meta_keywords')}
                    isRequired
                  />
                </Grid>
                <Grid item xs={12}>
                  <RaMultiLanguageInput
                    source={SOURCES['meta_description']}
                    inputProps={{ multiline: true }}
                    label={translate('labels.meta_description')}
                  />
                </Grid>
                {withContent && (
                  <Grid item xs={12}>
                    <LabelWrapper
label={translate('labels.page_content')}>
                      <BuilderInput
                        defaultLocale={defaultLocale}
                        source={SOURCES['content']}
                        sourceFiles={SOURCES['content_files']}
```

```
                      label={translate('labels.builder_input')}
                      onClick={handleBuilderInputClick}
                      onSave={save}
                      queryParams={{
                        host: get(sites, '0.host'),
                      }}
                    />
                  </LabelWrapper>
                </Grid>
              )}
            </GridWrapper>
          </ValidationForm>
        </FormPaper>
      </Grid>
    </GridWrapper>
  </StyledPagesForm>
 );
};
```

Етап 4: Запуск проекту

На цьому етапі я вже можу запустити проект та побачити основні листи та дані які приходять з серверу. Це останній крок, адмін панель готова. На наступних скріншотах можна побачити проект після запуску.

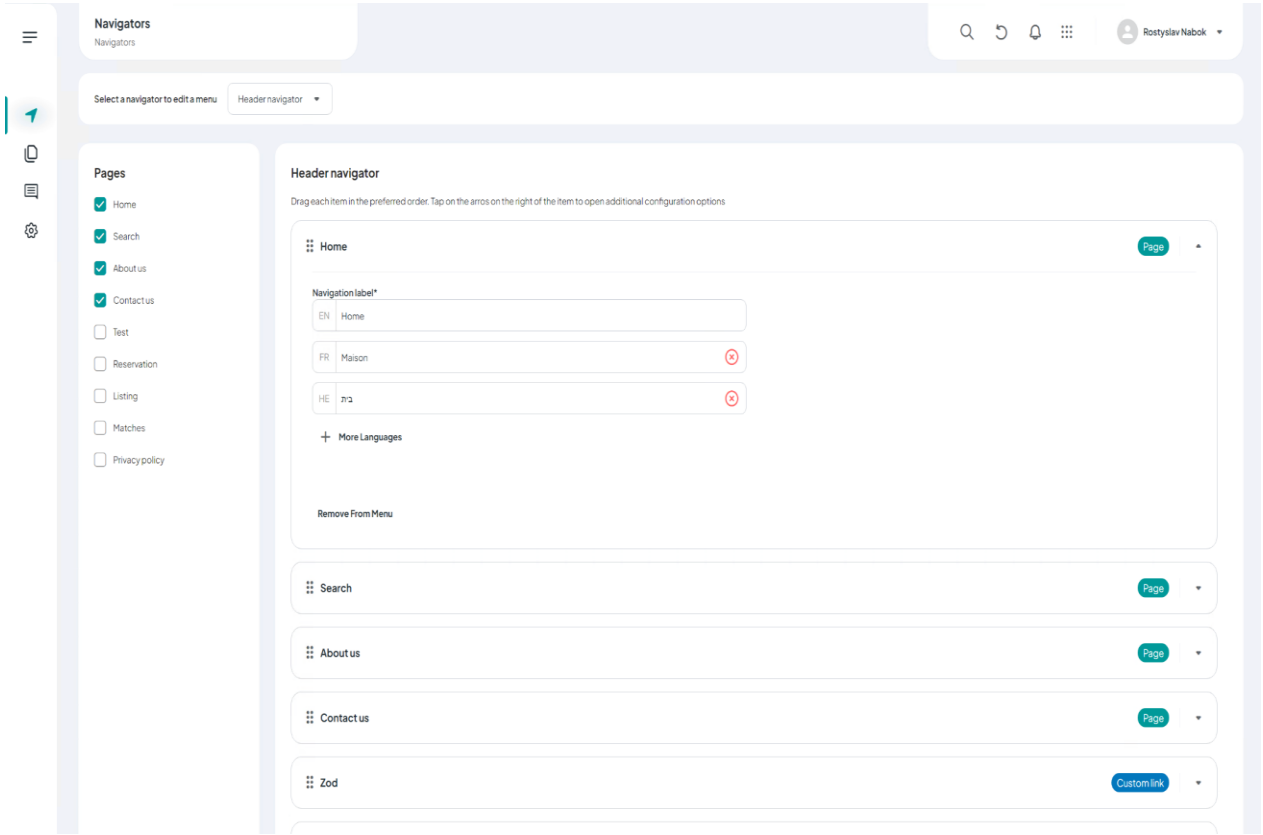

Рис. 3.14 - сторінка навігаторів

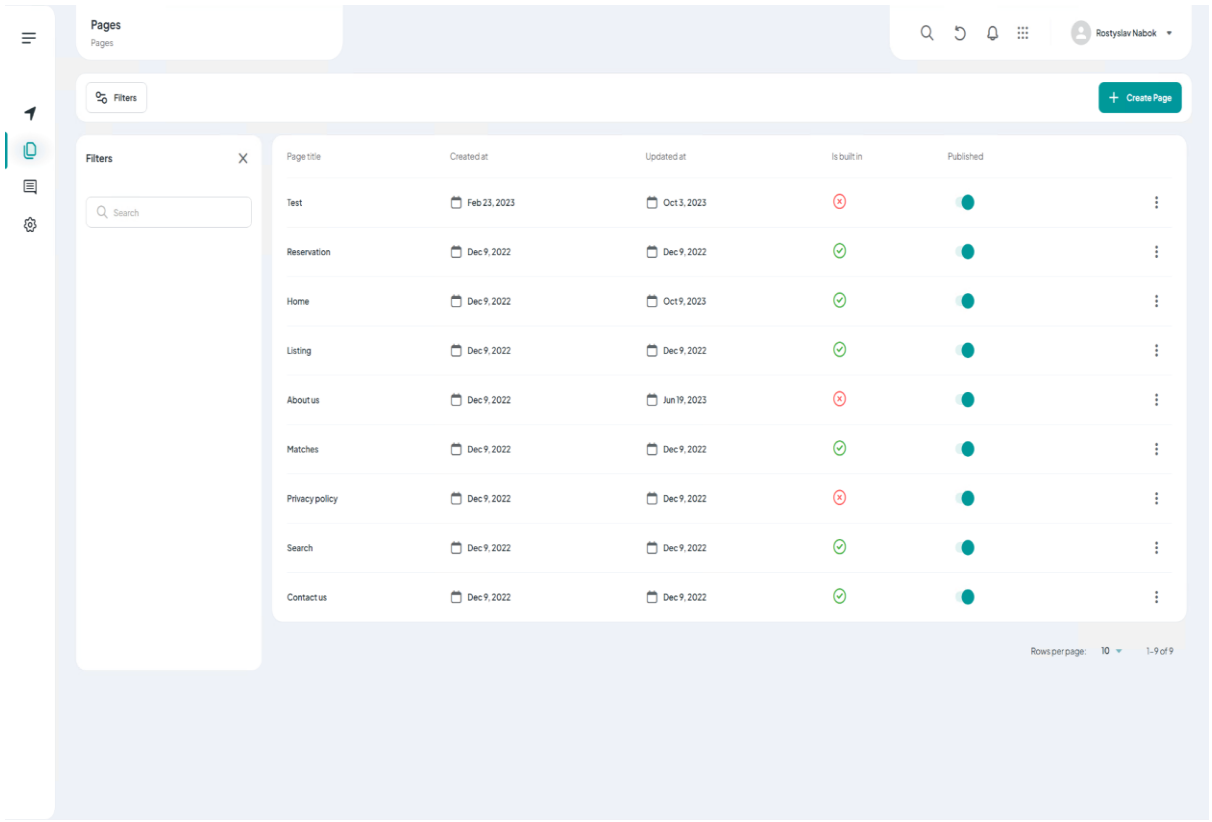

Рис. 3.15 - сторінка пейджів

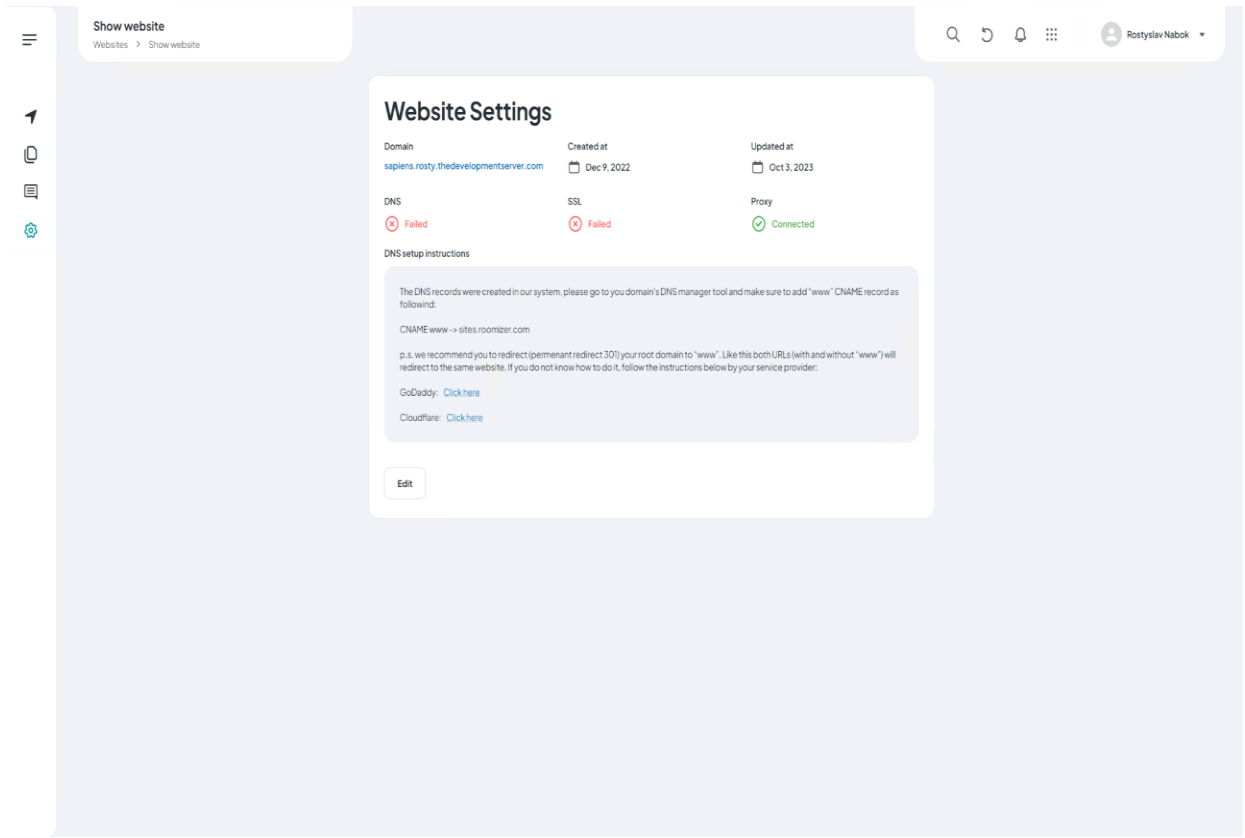

Рис. 3.16 - сторінка налаштуваня вебсайту

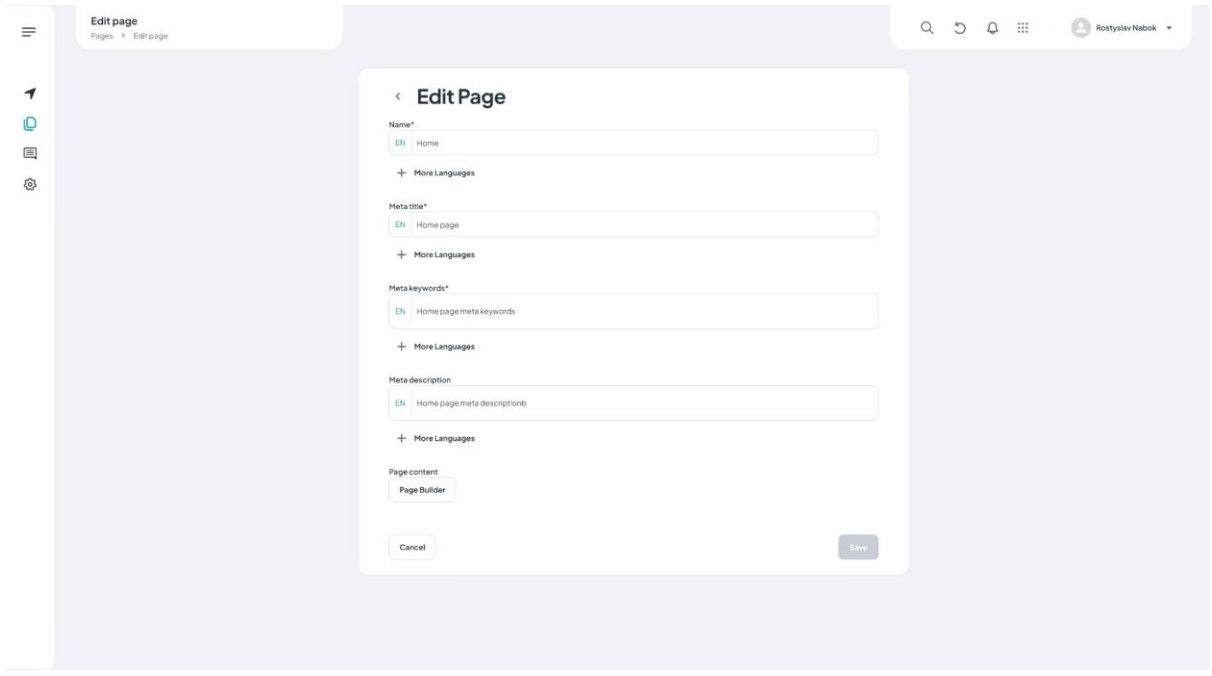

Рис. 3.17 - сторінка редагування пейджів

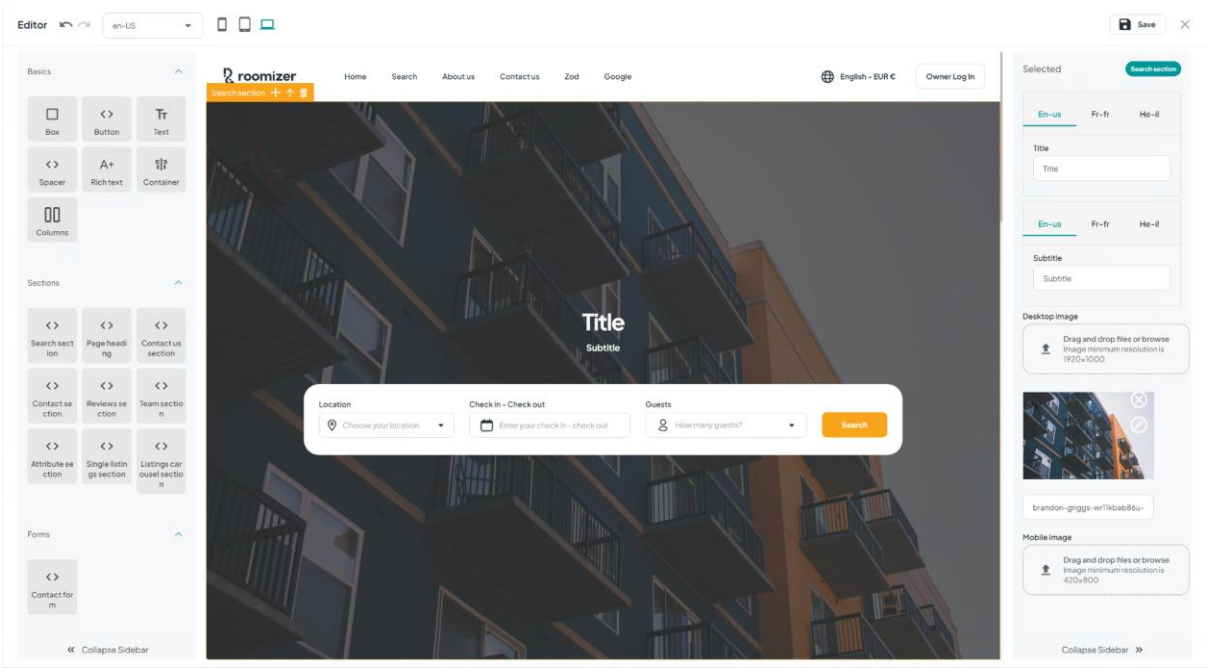

Рис. 3.18 - сторінка білдер налаштування

3.5 Розробка публічного вебсайту використовуючи фреймворк next.js:

Створення сайту з використанням фреймворка Next.js може значно полегшити розробку, особливо якщо мені потрібен сервер рендерінг або можливості статичного сайту. Нижче наведено загальний процес та приклади коду для створення сайту за допомогою Next.js:

Етап 1: Встановлення та Конфігурація

1. Встановлення Next.js:

```
npx create-next-app my-nextjs-app
```

Етап 2: Створення Компонентів та Сторінок

1. Створення сторінок:

Створивши базовий макет проекту я приступаю до налаштування, створення сторінок. У моєму проекті є 4 базові сторінки: домашня сторінка, сторінка пошуку, сторінка одного лістингу, сторінка оплати, сторінка подяки, сторінка контактів, загальна сторінка. В наступному блоці коду я створюю загальну сторінку, вона буде використовуватись для загального темплейту, він буде активуватись коли користувач обере ‘custom’ сторінку в адмін панелі та захоче її налаштуквати.

```
import { withServerSideProps } from 'helpers/withServerSideProps';
import { prefetchPage, useGetPage } from 'hooks/index';
import { prefetchCurrencies } from 'hooks/queries/useGetCurrencies';
import { prefetchLocales } from 'hooks/queries/useGetLocales';
import { prefetchPopularCities } from
'hooks/queries/useGetPopularCities';
import type { NextPage } from 'next';
import DefaultErrorPage from 'next/error';
import { NextSeoProvider } from 'providers/NextSeoProvider';
import { StandartTemplate } from 'templates/StandartTemplate';

export const IndexPage: NextPage = () => {
  const { error: pageError, data: page } = useGetPage();

  if (pageError) {
    return <DefaultErrorPage statusCode={404} />;
  }

  return (
    <>
      <NextSeoProvider
        title={page?.meta_title}
        description={page?.meta_description}
        additionalMetaTags={[{ name: 'keywords', content:
page?.meta_keywords || '' }]}
      />
      <StandartTemplate />
    </>
  );
};

export default IndexPage;

export const getServerSideProps = withServerSideProps({
  enableRequestsOnClientSideTransition: true,
  fetchers: [prefetchPage, prefetchPopularCities, prefetchCurrencies,
prefetchLocales],
});
```

В коді нижче я створюю сторінку пошуку, одну з найскладніших та найважливіших сторінок, це основа для сайту. Тут користувач може знайти лістинги які його цікавлять використовуючи карту та фільтри.

```
import { withServerSideProps } from 'helpers/withServerSideProps';
import { prefetchPage, useGetPage } from 'hooks/index';
import { prefetchCurrencies } from 'hooks/queries/useGetCurrencies';
import { prefetchLocales } from 'hooks/queries/useGetLocales';
import { prefetchPopularCities } from
'hooks/queries/useGetPopularCities';
import type { NextPage } from 'next';
import DefaultErrorPage from 'next/error';
import { NextSeoProvider } from 'providers/NextSeoProvider';
import { StandartTemplate } from 'templates/StandartTemplate';

export const IndexPage: NextPage = () => {
  const { error: pageError, data: page } = useGetPage();

  if (pageError) {
    return <DefaultErrorPage statusCode={404} />;
  }

  return (
    <>
      <NextSeoProvider
        title={page?.meta_title}
        description={page?.meta_description}
        additionalMetaTags={[{ name: 'keywords', content:
page?.meta_keywords || '' }]}
      />
      <StandartTemplate />
    </>
  );
};

export default IndexPage;

export const getServerSideProps = withServerSideProps({
  enableRequestsOnClientSideTransition: true,
  fetchers: [prefetchPage, prefetchPopularCities, prefetchCurrencies,
prefetchLocales],
});
```

Далі по плану приступаю до сторінки обраного лістингу, тут користувач може побчаити детальну інформацію про вибрану нерухомість, почитати опис, побачити всі функції, розглянути фото та перейти далі на сторінку оплати або бронювання. Код цієх сторінки можна побачити на нижче.

```
import { FooterClasses } from 'components/layouts/Footer/Footer';
import { MainLayout } from 'components/layouts/MainLayout';
import ListingPage from
'components/pageTemplates/ListingPage/ListingPage';
import ListContextProvider from 'contexts/ListContext';
import { withServerSideProps } from 'helpers/withServerSideProps';
import { prefetchFilterAttributes, prefetchListing, useGetListing } from
'hooks/index';
import { prefetchCurrencies } from 'hooks/queries/useGetCurrencies';
import { prefetchLocales } from 'hooks/queries/useGetLocales';
import { get } from 'lodash';
import type { NextPage } from 'next';
import { NextSeoProvider } from 'providers/NextSeoProvider';
import { MOBILE_FIXED_BOTTOM_SECTION_HEIGHT, styled } from
'theme/index';

import { isBrowser } from '@packages/helpers';

const PREFIX = 'ListingShow';

const StyledListingShow = styled('div', {
  name: `Styled${PREFIX}`,
  label: `Sapiens--${PREFIX}`,
})(({ theme }) => ({
  [`&& .${FooterClasses.root}`]: {
    [theme.breakpoints.down('md')]: {
      marginBottom: MOBILE_FIXED_BOTTOM_SECTION_HEIGHT,
    },
  },
}));

const ListingShow: NextPage<{ referer: string | null }> = (props) => {
  const { referer = '' } = props;
  const { data: listing, isSuccess } = useGetListing();

  const firstImage = get(listing, 'gallery.0.src');

  /* We use SSR 'referer' for initial render and change it with 'href'
soon */
  const url = isBrowser ? window.location.href : referer;
  return (
    <ListContextProvider>
      <NextSeoProvider
        title={listing?.title}
        openGraph={{
          images: [{ url: firstImage }],
        }}
      />
```

```
      <StyledListingShow>
        <MainLayout>
          {isSuccess && listing && <ListingPage data={listing} url={url
|| ''} />}
        </MainLayout>
      </StyledListingShow>
    </ListContextProvider>
  );
};

export default ListingShow;

export const getServerSideProps = withServerSideProps({
  fetchers: [prefetchListing, prefetchCurrencies, prefetchLocales,
prefetchFilterAttributes],
  handler: (_queryClient, context, props) => {
    const { req } = context;

    const { referer = null } = req.headers;

    return { props: { ...props, referer } };
  },
});
```

Наступний крок це розробка сторінки подяки, це відбувається після успішного бронювання. Це фінальна стрінка де можна знайти детальну інформацію про час заселення, точну адресу. Код цієї сторінки можна побачити нижче.

```
import { MainLayout, MainLayoutClasses } from
'components/layouts/MainLayout';
import { ThankYouTemplate } from
'components/pageTemplates/ThankYouTemplate';
import { withServerSideProps } from 'helpers/withServerSideProps';
import { prefetchCurrencies } from 'hooks/queries/useGetCurrencies';
import { prefetchLocales } from 'hooks/queries/useGetLocales';
import { prefetchReservation, useGetReservation } from
'hooks/queries/useGetReservation';
import { RESERVATION_RESPONSE_SOURCES } from
'hooks/queries/useMutationReservation';
import { useRouterWithLocale } from 'hooks/useRouterWithLocale';
import { get } from 'lodash';
import { NextPage } from 'next';
import { NextSeoProvider } from 'providers/NextSeoProvider';
import { styled } from 'theme/index';
```

```
const PREFIX = 'ThankYouPage';

const StyledMainLayout = styled(MainLayout, {
  name: `Styled${PREFIX}`,
  label: `Sapiens--${PREFIX}`,
})(() => ({
  [`&& .${MainLayoutClasses.main}`]: {
    alignItems: 'center',
    display: 'flex',
  },
}));

const ThankYouPage: NextPage = () => {
  const { query } = useRouterWithLocale();

  const reference = get(query,
RESERVATION_RESPONSE_SOURCES['reference']) as string;
  const { data: reservation, isSuccess } = useGetReservation(reference);

  return (
    <>
      <NextSeoProvider title={reservation?.reference} />
      <StyledMainLayout>
        {isSuccess && reservation && <ThankYouTemplate
reservationData={reservation} />}
      </StyledMainLayout>
    </>
  );
};

export default ThankYouPage;

export const getServerSideProps = withServerSideProps({
  fetchers: [prefetchReservation, prefetchLocales, prefetchCurrencies],
});
```

2. Створення основних компонентів:

Створивши базовий макет проекту я приступаю до налаштування, створення сторінок та основних компонентів. Публічний вебсайт це дуже комплексний та складний проект тому описати код кожного компонента буде дуже тяжко, але я покажу основні з них. Почну з розробки компонента заголовок. Код цього компоненту можна побачити нижче.

```
const Header: FC<HeaderProps> = (props) => {
  const { sx } = props;

  const { isEditing } = useBuilderState();

  return (
    <>
      {!isEditing && (
        // default app bar to hold space
        <AppBar
          component="div"
          position="static"
          color="transparent"
          sx={{
            border: 'none',
            boxShadow: 'none',
            height: { xs: MOBILE_HEADER_HEIGHT, md: HEADER_HEIGHT },
          }}
        />
      )}
      <AppBar
        position={isEditing ? 'static' : 'fixed'}
        color="transparent"
        sx={{
          bgcolor: 'background.default',
          color: 'text.primary',
          zIndex: 'modal',
          boxShadow: 'none',
          height: { xs: MOBILE_HEADER_HEIGHT, md: HEADER_HEIGHT },
          borderBottom: `1px solid ${theme.palette.primary.background}`,
          ...sx,
        }}
      >
        <MenuToolbar />
      </AppBar>
    </>
  );
};

export default Header;
```

Тепер хочу представити компонент якоря який буде використовуватись на сторінці лістинга, його зоінішній вигляд можна побачити на рисунку 3.19.

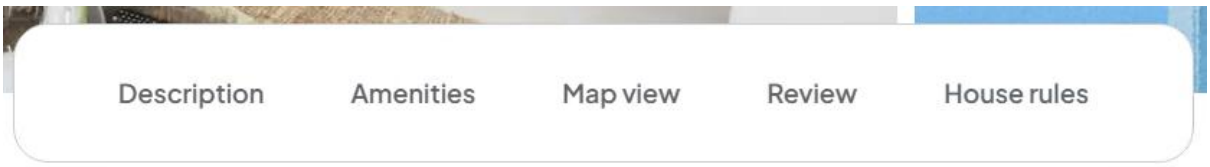

Рис. 3.19 - компонент якоря

Якорі в HTML - це спосіб створення посилань на певні частини того ж документа або інших документів в Інтернеті. Якорі дають можливість легко навігувати по великих документах та створювати внутрішні посилання. Їх можна використовувати для внутрішньої навігації або посилань на конкретні частини вмісту. У HTML, для створення якорів, використовується тег <a> з атрибутом href, в якому значення починається з символу #, а за ним слідує ім'я якоря. Місце для якоря вказується за допомогою атрибута id у відповідному елементі на сторінці.

```
import { FC } from 'react';

import { Box } from '@mui/material';

import { anchorMenuItems } from '../../../app/constants';
import { Link } from '../../links/Link';

export interface AnchorMenuProps {
  items?: Array<{
    label: string;
    href: string;
  }>;
}

export const AnchorMenu: FC<AnchorMenuProps> = (props) => {
  const { items = anchorMenuItems } = props;
  return (
    <Box
      sx={{
        display: 'flex',
        justifyContent: 'space-between',
        gap: 5,
        bgcolor: 'background.default',
        px: 6,
        border: 1,
        borderColor: 'primary.border',
        borderRadius: 4,
        height: 64,
```

```
        alignItems: 'center',
        overflow: 'auto',
      }}
    >
      {items.map((item) => {
        const { label, href } = item;
        return (
          <Link
            key={label}
            t={label}
            href={href}
            variant="body2"
            color="text.light"
            sx={{ textDecoration: 'none', ':hover': { color:
'primary.main', transition: '0.3s' } }}
          />
        );
      })}
    </Box>
  );
};

export default AnchorMenu;
```

Далі розглянемо компонент карту (рис. 3.20), це один з найважливіших компонентів системи, він максимально інтерактивний. Такі функції як зум, унікальні маркери та тултіпи допоможуть покращити користувацький досвід карти. Застосовуючи цей компонент користувач може швидко та гнучко побчаити адреса на карті, відкрити меню інформації про нерухомість та ознайомитись.

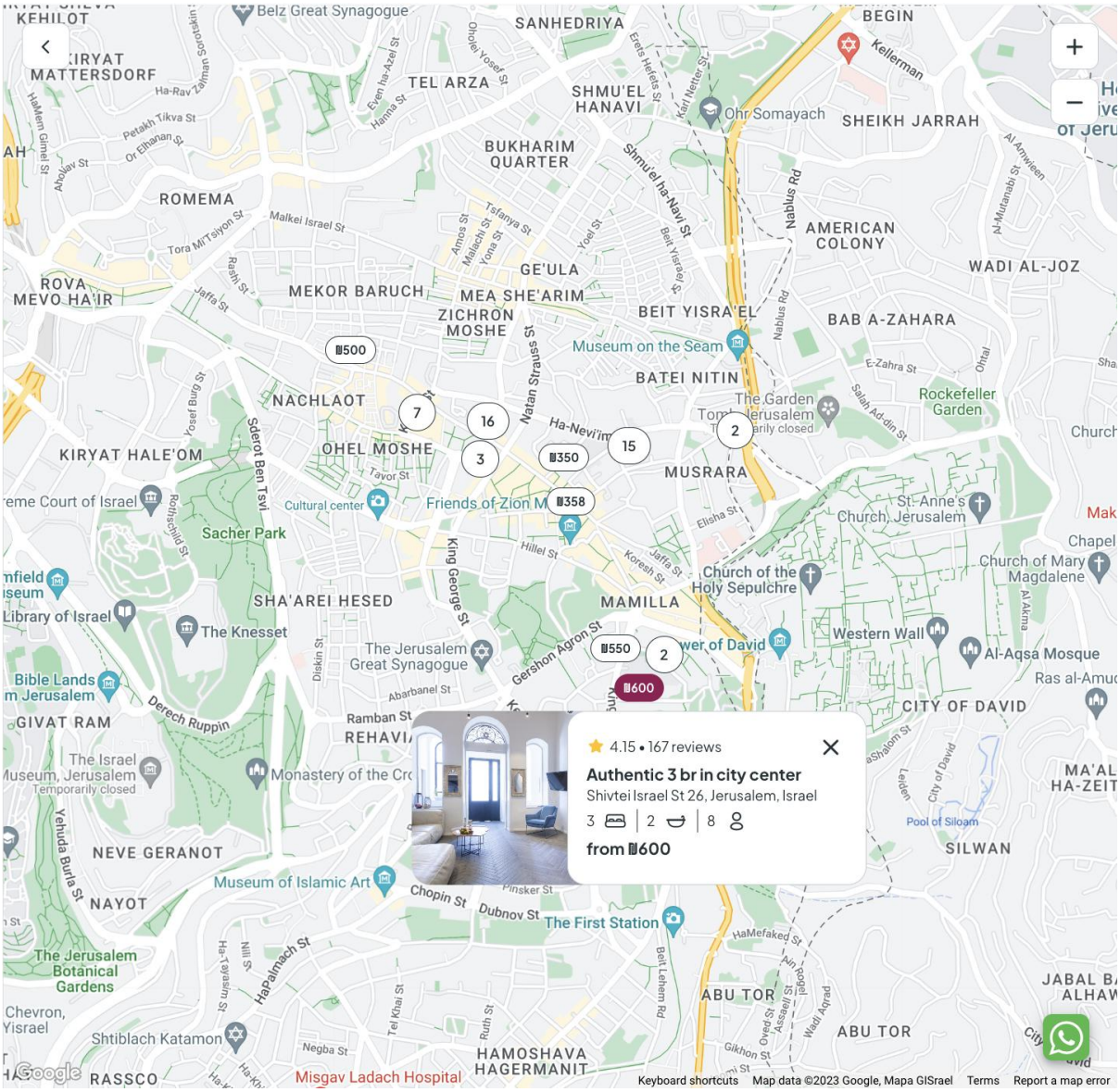

Рис. 3.20 - компонент “Map”

Код основного компонента (див. нижче) максимально декомпозований для зручного налаштування, перевикористання.

```
import { CSSProperties, FC, useMemo } from 'react';

import { GoogleMap, GoogleMapProps, LoadScript } from '@react-google-
maps/api';
import { DEFAULT_LOCALE, DEFAULT_MAP_ZOOM_VALUE } from 'app/constants';
import { Loader } from 'custom/Loader';
import { useCurrentLocale } from 'helpers/useCurrentLocale';
import { styled } from 'theme/index';

import { StyledRootProps } from '@packages/types';
```

```
import { ZoomControl } from './components/ZoomControl';
import { customMapStyle } from './mapStyles';

export const GOOGLE_MAPS_API_KEY = process.env.NX_GOOGLE_MAP_API_KEY;

const PREFIX = 'Map';

export const MapClasses = {
  mapContainer: `${PREFIX}-mapContainer`,
};

const StyledMap = styled('div', {
  name: `Styled${PREFIX}`,
  label: `Sapiens--${PREFIX}`,
})(() => ({
  width: '100%',
  height: '100%',
  display: 'flex',
  alignItems: 'center',
  justifyContent: 'center',

  [`& .${MapClasses.mapContainer}`]: {
    width: '100%',
    height: '100%',
  },
}));

export const MIN_ZOOM_VALUE = 2;

export type Map = google.maps.Map;
export type LatLng = google.maps.LatLng;
export type LatLngLiteral = google.maps.LatLngLiteral;
export type LatLngBounds = google.maps.LatLngBounds;
export type BoundsLiteral = google.maps.LatLngBoundsLiteral;
export type ClustersBounds = GeoJSON.BBox;
export type Zoom = number;

export type MapProps = GoogleMapProps &
  StyledRootProps & {
    mapPosition?: CSSProperties['position'];
    googleMapsApiKey: string;
    googleMapsMapId?: string;
  };

export const Map: FC<MapProps> = (props) => {
  const { sx, className, googleMapsApiKey, googleMapsMapId, children,
options, ...rest } = props;
```

```
  const { currentLocale } = useCurrentLocale();

  const language = useMemo(() => currentLocale?.locale ||
DEFAULT_LOCALE, [currentLocale?.locale]);

  return (
    <StyledMap sx={sx} className={className}>
      <LoadScript
        googleMapsApiKey={googleMapsApiKey}
        language={language}
        loadingElement={<Loader />}
      >
        <GoogleMap
          mapContainerClassName={MapClasses.mapContainer}
          options={{
            mapId: googleMapsMapId,
            fullscreenControl: false,
            streetViewControl: false,
            mapTypeControl: false,
            disableDefaultUI: true,
            /**
             * We should remove our styles because they can conflict
with
             * user styles from google cloud that we get via the map id
             */
            styles: googleMapsMapId ? undefined : customMapStyle,
            minZoom: MIN_ZOOM_VALUE,
            ...options,
          }}
          zoom={DEFAULT_MAP_ZOOM_VALUE}
          {...rest}
        >
          {children}
          <ZoomControl />
        </GoogleMap>
      </LoadScript>
    </StyledMap>
  );
};
```

РОЗДІЛ 4.
ПЕРЕВІРКА МЕТОДИКИ ВДОСКОНАЛЕННЯ РОЗРОБКИ ВЕБСАЙТІВ

4.1 Перевірка методики

В розділі "Перевірка Методики" я планую ретельно проаналізувати ефективність та результативність використання вебсайт-білдера, спеціально налаштованого для потреб "Пошуку та Оренди Нерухомості". Цей вебсайт-білдер надає можливість впровадження унікального та інноваційного підходу до створення вебсайтів для даної галузі.

Особливий акцент буде зроблено на адмін панелі, яка визначається своєю гнучкістю та налаштовуваністю. За допомогою адмін панелі користувачі можуть змінювати поведінку вебсайту відповідно до їхніх потреб та стратегій. Розглянемо можливості адмін панелі, які дозволяють легко налаштовувати вигляд, функціонал, та інші аспекти вебсайту без необхідності великих технічних навичок чи залучення програмістів.

Під час аналізу я спробую визначити, наскільки ефективно вебсайт-білдер і адмін панель відповідають вимогам та очікуванням, пов'язаним із створенням та управлінням веб-платформою для пошуку та оренди нерухомості.

Я запропонував своєму другу Ліору з Ізраїлю який працює агентом нерухомості спробувати мій проект на практиці з використанням реальних даних, реальних будинків. Перевірка методики включає у себе фазу тестування на практиці після віддавання проекту агентові. Під час цього етапу аналізується реальна робота вебсайту в живому середовищі. Оцінюються такі аспекти, як продуктивність, відповідність вимогам користувачів, ефективність адміністрування та інші показники. Проект було залито на хостинг https://isrentals.viciniapp.com. Я допомагав з налаштуваннями та додаванням даних в базу. Після успішного запуску я збирав дані по користувачам та помилкам які можуть бути на вебсайті за допомогою стороннього сервісу Sentry. Пройшов тиждень і я отримав баг репорт з однією помилкою яка була відразу виправлена. Сайт успішно працює і на сьогоднішній день. Ліор дуже задоволений проектом, клієнти також. Всі поставлені цілі виконані.

Я зробив вебсайт білдер який може бути проданий як SAS. Інший бізнес може використовувати даний проект під свої цілі, зі своїми даними тому це дуже прибуткове та далекоглядне рішення. Основні скріншоти програми можна побачити на рисунках.

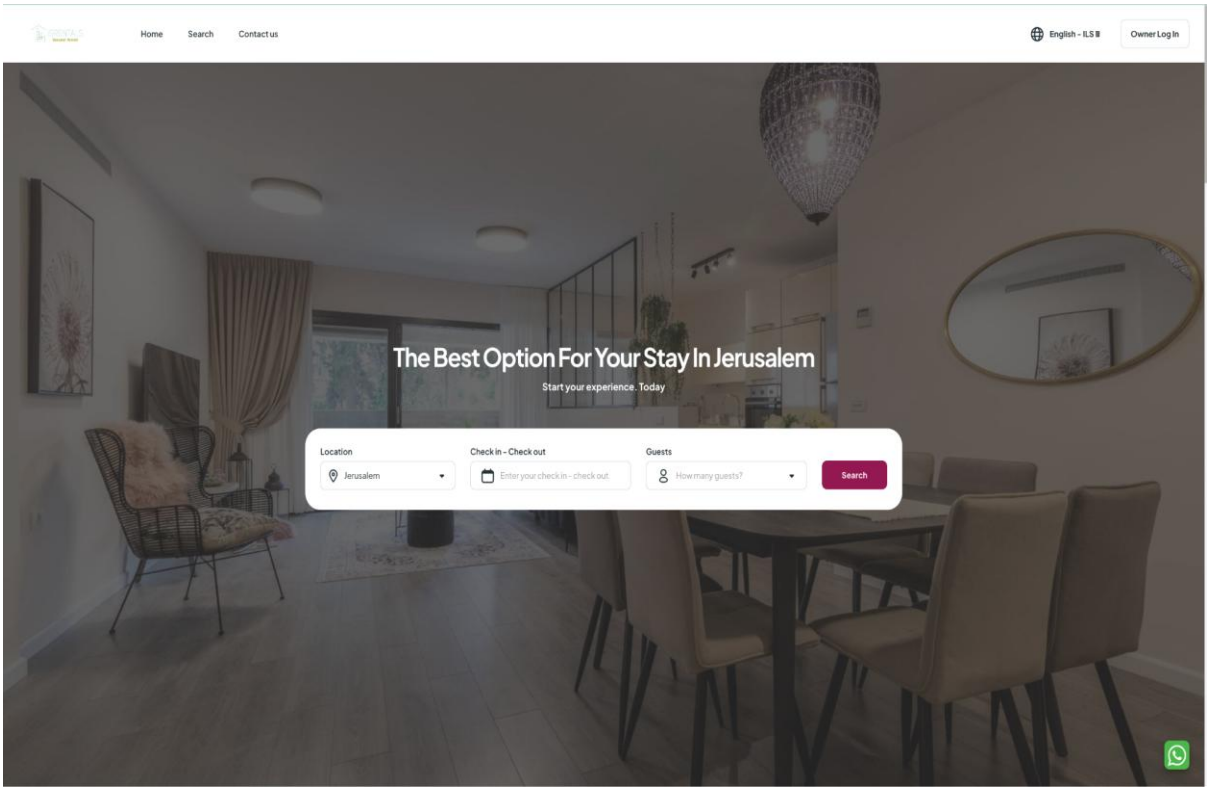

Рис. 3.21 - головна сторінка

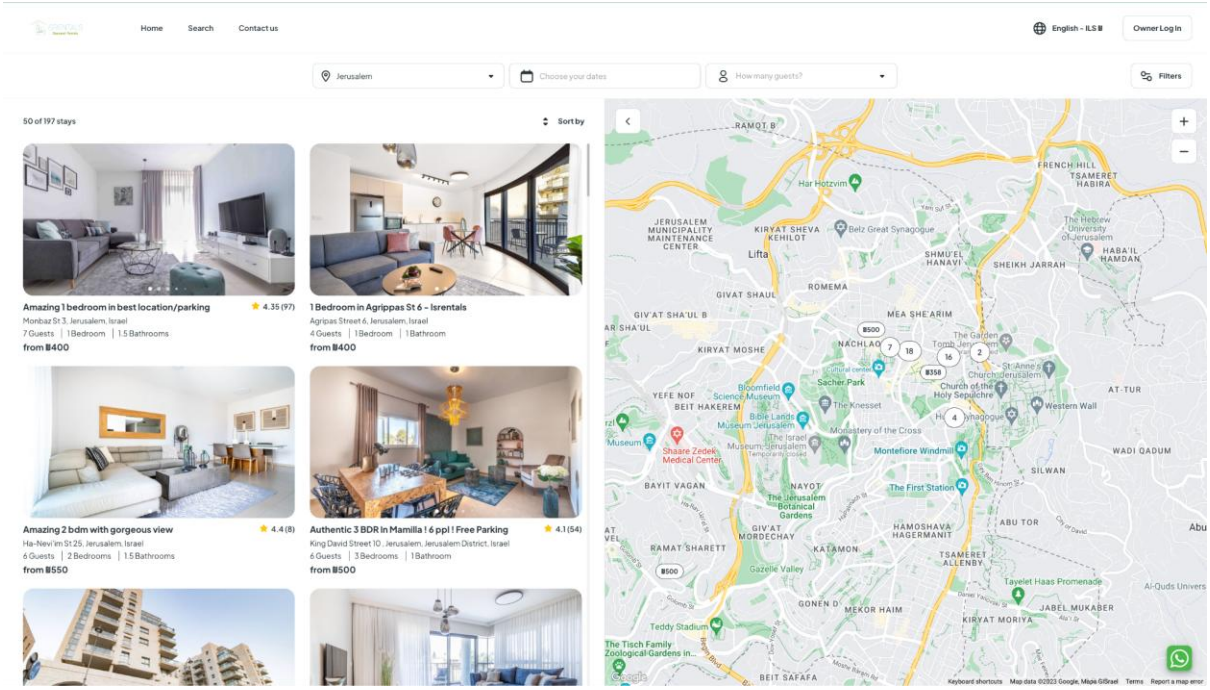

Рис. 3.22 - сторінка пошуку

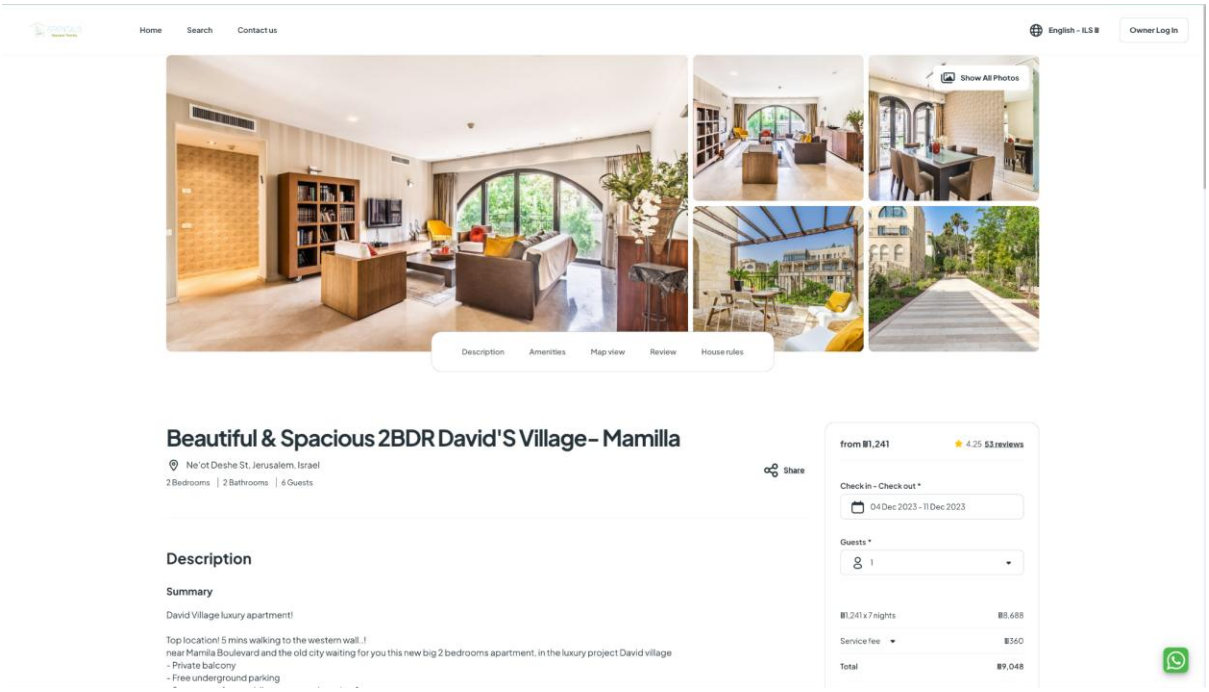

Рис. 3.21 - сторінка лістингу

ВИСНОВОК

Розробивши продукт-систему з адмін панелью та публічним сайтом, використавши його в реальному житті та отримавши позитивний відгук я можу сказати, що це було вдале рішення. Бізнес нерухомості може легко налаштовувати свої будинки, гнучко корегувати публічний вебсайт та отримувати прибутки.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. What Is the Best Website Builder for You? Let's Break It Down. Електронний ресурс: https://www.shutterstock.com/blog/best-website-builder-for-you?utm_source=GOOGLE&utm_campaign=CO%3DRU_LG%3DEN_BU%3DIMG_AD%3DDSA_TS%3Dlggeneric_RG%3DEUAF_AB%3DACQ_CH%3DSEM_OG%3DCONV_PB%3DGoogle&ds_cid=71700000063236480&ds_ag=DSA%20-%20Blog&kw=&ds_agid=5870
2. Best Website Builder. Електронний ресурс: https://www.forbes.com/advisor/business/software/best-website-builders/
3. Our top three website builder provider. Електронний ресурс: https://www.techradar.com/news/the-best-website-builder