Хмельницький національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

КВАЛІФІКАЦІЙНА РОБОТА

Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника

Назва теми

Рівень вищої освіти Перший(бакалаврський)

Галузь знань 12 «Інформаційні технології»

Спеціальність 121«Інженерія програмного забезпечення»

Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

Шифр КвРІПЗ.200123.20.06.ПЗ

Виконав студент ІІІ курсу група ІПЗс-20-1 _____ В.А. Качур
Підпис Ініціали, прізвище

Керівник д-р фіз.-мат. наук, проф. _____ Л. П. Бедратюк
Науковий ступінь, звання Підпис Ініціали, прізвище

Нормоконтролер канд. пед. наук, доцент _____ Н.І. Праворська
Підпис Ініціали, прізвище

До захисту допускаю:
Завідувач кафедри інженерії програмного забезпечення _____ Л. П. Бедратюк
Підпис Ініціали, прізвище

26 травня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій
Кафедра Інженерії програмного забезпечення
Рівень вищої освіти Перший (бакалаврський)
Галузь знань 12 «Інформаційні технології»
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма Освітньо-професійна програма «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри ІПЗ
Л. П. Бедратюк
05 02 2023 р.

ЗАВДАННЯ
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ)

Качуру Володимиру Андрійовичу
Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника

Керівник проекту (роботи) Бедратюк Леонід Петрович, д-р фіз.-мат. наук, проф.
Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 05.02.2023 р. № 6
2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.
3. Вихідні дані до проекту (роботи) Матеріали переддипломної практики

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити) Дослідження предметної області та постановка задачі, проектування програмного забезпечення, програмна реалізація, тестування програмного забезпечення

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень) Презентаційні матеріали (слайди, 22шт.)

6. Консультанти розділів дипломного проекту

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийня
Нормоконтроль	Праворська Н.І., канд. пед. наук, доцент кафедри ІПЗ	23.05.2023р.	23.05.2023р.
Антиплагіат	Гурман І. В., доцент кафедри ІПЗ	25.05.2023р.	25.05.2023р.

7. Дата видачі завдання « 05 » лютого 2023р.

КАЛЕНДАРНИЙ ПЛАН

Назва етапів (розділів) дипломного проекту (роботи)	Строк виконання етапів проекту (роботи)	Примітка
1 Ознайомлення з тематикою дипломного проектування (ДП), визначення та узгодження індивідуальних там ДП	01.12 – 30.12.2022	
2 Дослідження предметної області, в якій планується використання програмного засобу (ПЗ), визначення задач та вимог, розробка технічного завдання	02.01 – 31.01.2023	
3 Проектування програмного забезпечення	01.02 – 28.02.2023	
4 Програмна реалізація	01.03 – 10.04.2023	
5 Тестування програмного забезпечення	11.04 – 30.04.2023	
6 Написання вступу, загальних висновків, оформлення джерел посилання та додатків. Оформлення пояснювальної записки ДП згідно вимог стандартів	01.05 – 25.05.2023	
7 Попередній захист ДП	Травень 2023 (згідно графіка)	
8 Перевірка ДП на плагіат, нормконтроль, отримання відгуків та рецензій. Брошурування (зшиття) пояснювальної записки	26.05 – 30.05.2023	
9 Підготовка до захисту та захист ДП	з 01.06.2023	

Студент _____ В.А. Качур
Підпис Ініціали, прізвище

Керівник проекту (роботи) _____ Л.П. Бедратюк
Підпис Ініціали, прізвище

АНОТАЦІЯ

Тема кваліфікаційної роботи: Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника.

Автор роботи: Качур Володимир Андрійович.

Керівник проекту: Бедратюк Леонід Петрович.

Пояснювальна записка: 104 с.,43 рис.,5 табл., 4дод., 19 джерел.

Графічна частина: 22 слайдів.

ЗАБУДОВА, E-COMMERCE, JAVASCRIPT, REST, SQL, LARAVEL, VUE.JS, VUEX, SCSS, MVC.

Метою проекту є розробка веб-застосунку, який забезпечує покращення взаємодії між компанією-забудовником та її клієнтами, що в свою чергу призведе до збільшення ефективності роботи та підвищення рівня задоволення клієнтів.

У процесі кваліфікаційної роботи було проведено аналіз предметної області, виявлено причини частого виникнення проблем при пошуку подібних послуг без використання онлайн-сервісів, проаналізовано існуючі альтернативні програмні продукти, а також порівняно архітектурні рішення та патерни для розробки веб-додатків. Крім того, були визначені модулі системи та здійснена їх програмна реалізація.

Для розробки веб-застосунку був використаний фронтенд JavaScript фреймворк Vue.js, бекенд PHP фреймворк Laravel, та база даних SQL.

В результаті було реалізовано веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника.

22.05.2023р.
Дата

Підпис

ВІДОМІСТЬ ДОКУМЕНТІВ

№ рядка	Формат	Позначення документа	Найменування документа	К-сть аркушів	№ екз.	Примітка
			Текстові документи			
1	А4	КвРІПЗ.200123.20.06.ПЗ	Пояснювальна записка	104		
2	А4		Завдання на дипломний	1		
			проект			
3	А4		Анотація	1		
			Графічні документи			
4	А4		Презентаційні матеріали	22		

Змн.	Арк.	№ докум.	Підпис	Дата	КвРІПЗ.200123.20.06.ПЗ		
Виконав		Качур В.А.			Веб-застосунок для автоматизації клієто-орієнтованих бізнес-процесів компанії-забудовника Відомість документів	Літ.	Арк. 4 / Акрушів 104
Керівник		Бедратюк Л.П.				ХНУ, ІПЗс-20-1	
Н. Контр.		Праворська Н.І.					
Зав. Каф.		Бедратюк Л.П.					

ЗМІСТ

ВСТУП

Зважаючи на нинішню політичну та економічну ситуацію в Україні, розвиток будівельної галузі є важливим елементом відновлення економіки та інфраструктури країни. Крім того, будівельна галузь є однією з основних галузей національної економіки, яка прямо чи опосередковано забезпечує зайнятість великої кількості людей в Україні.

У сучасному світі, компанії, що займаються будівництвом, стикаються зі складними процесами управління проектами, відстеженням виконання робіт та спілкуванням зі своїми клієнтами. Ці процеси можуть бути вкрай складними та забирати велику кількість часу та зусиль.

З метою оптимізації цих процесів, можна розглянути розробку веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника. Такий веб-застосунок може включати в себе функціональні можливості для ефективного відстеження виконання робіт, управління проектами, а також спілкування з клієнтами.

Для розробки веб-застосунку можна використовувати сучасні технології та інструменти, такі як фреймворки для веб-розробки, бази даних, а також інші технічні засоби, що дозволяють реалізувати бізнес-процеси компанії в онлайн-режимі.

Мета проекту - розробка веб-застосунку, який забезпечує покращення взаємодії між компанією-забудовником та її клієнтами, що в свою чергу призведе до збільшення ефективності роботи та підвищення рівня задоволення клієнтів.

Отже, в сучасних умовах, коли більшість клієнтів шукає швидкі та зручні способи замовлення будівельних послуг, розробка веб-застосунків для автоматизації клієнто-орієнтованих бізнес-процесів є необхідністю. Такі веб-застосунки можуть допомогти забудовникам забезпечити зручний та швидкий доступ до інформації про будівельний процес та статус виконання робіт, що, в свою чергу, забезпечить підвищення рівня задоволення клієнтів та збільшення обсягів замовлень.

Для досягнення поставленої мети були сформовані завдання на кваліфікаційну роботу:

− дослідити предметну область будівельного бізнесу та виявити потреби потенційних користувачів веб-застосунку;

− провести аналіз існуючих рішень;

− розробити технічне завдання;

− розробити архітектуру веб-застосунку та бази даних;

− обрати технології для розробки;

− розробити зручний та привітний інтерфейс для користувачів;

− розробити веб-застосунок;

− протестувати готовий веб-застосунок.

1 ДОСЛІДЖЕННЯ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАДАЧІ

1.1 Змістовний аналіз предметної області, її структурних та функціональних особливостей

Предметною областю є розробка веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника. Основна мета застосунку – полегшити взаємодію і спростити бізнес-процеси з клієнтами, що пов'язані зі зведенням житлових будівель.

Основні структурні елементи веб-застосунку включають в себе функції для взаємодії з клієнтами, такі як онлайн-заявки на купівлю житла, перегляд документів, повідомлення про стан будівництва та різні опції спілкування з представниками компанії. До інших структурних елементів веб-застосунку входять модулі для збереження та обробки даних клієнтів, різні інструменти для управління проектами та системи звітності для внутрішнього використання.

Функціональні особливості веб-застосунку включають в себе можливість залишати онлайн-заявки, які дозволять клієнтам купувати житло в будь-який час з будь-якої точки світу. Застосунок також надає можливість перегляду статусу будівництва в режимі реального часу, зокрема зміни у планах будівництва, терміни закінчення робіт та інші деталі проекту.

Окрім цього, веб-застосунок має систему збереження даних клієнтів та операцій з ними. Ця система надає можливість зберігати та обробляти персональні дані клієнтів, такі як контактна інформація та інші деталі.

Отже, об’єктом розробки є діяльність компанії забудовника, та її взаємодія з клієнтами

Таким чином, проведено аналіз обраної предметної області, визначено суть понять, таких як: компанія-забудовник, будівництво і виділено види організацій, які надають подібні послуги. Визначено і частково описано об’єкт розробки.

1.2 Аналіз наявного програмно-технічного забезпечення предметної області

Проведено детальний аналіз наявного програмного забезпечення, що дозволило створити список веб-сайтів, які користуються популярністю серед користувачів:

- “F7”;
- “AVILA Building Group”;
- “Me2Me”;
- “КомфортБудПлюс”;
- “IDM Group”.

Нижче наведено опис властивостей усіх згаданих програмних продуктів, з урахуванням їх функціональних та нефункціональних особливостей, зовнішнього вигляду, переваг та недоліків, отриманий з особистого досвіду використання та відгуків користувачів.

F7 – популярна компанія забудовник з привабливим веб-сайтом.

Серед інших вирізняється чудовим сучасним дизайном, великою кількістю складних анімацій та приємною темною темою (рисунки 1.1 та 1.2). Присутній обширний набір фотогалерей зі станами будівництва будинків, готових квартир та можливих варіантів готових інтер’єрів. Цікавою особливістю є можливість обирати перегляд планувань поверхів клікаючи безпосередньо на поверх будівлі на зображенні. Присутня можливість задати питання забудовнику через онлайн-форму на сайті та переглянути всі документи будови, що серйозно скорочує та спрощує об’єм роботи працівників відділу роботи з клієнтами.

Та з точки зору користувача є перелік недоліків, прикладом яких є завелика довжина головної сторінки, занадто великі фотографії будинків для вибору перегляду планувань – що виглядає занадто незручним для користувачів ноутбуків та пристроїв з невеликими розмірами екрану.

Основним недоліком можна виділити те, що на сайті взагалі відсутня інформація про статус наявності у продажу тої чи іншої квартири, що суттєво збільшує кількість роботи працівників відділу роботи з клієнтами через постійні відповіді на запитання чи є ще в наявності та чи інша житлова площа.

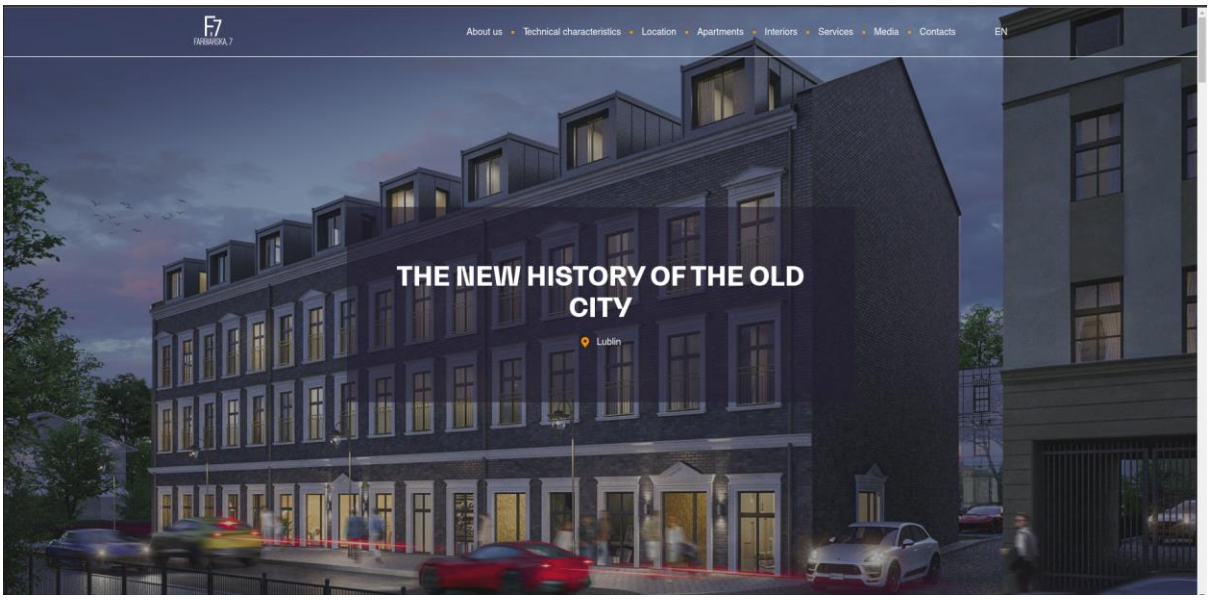


Рисунок 1.1 – Інтерфейс сайту “F7” - https://f7-lublin.pl/

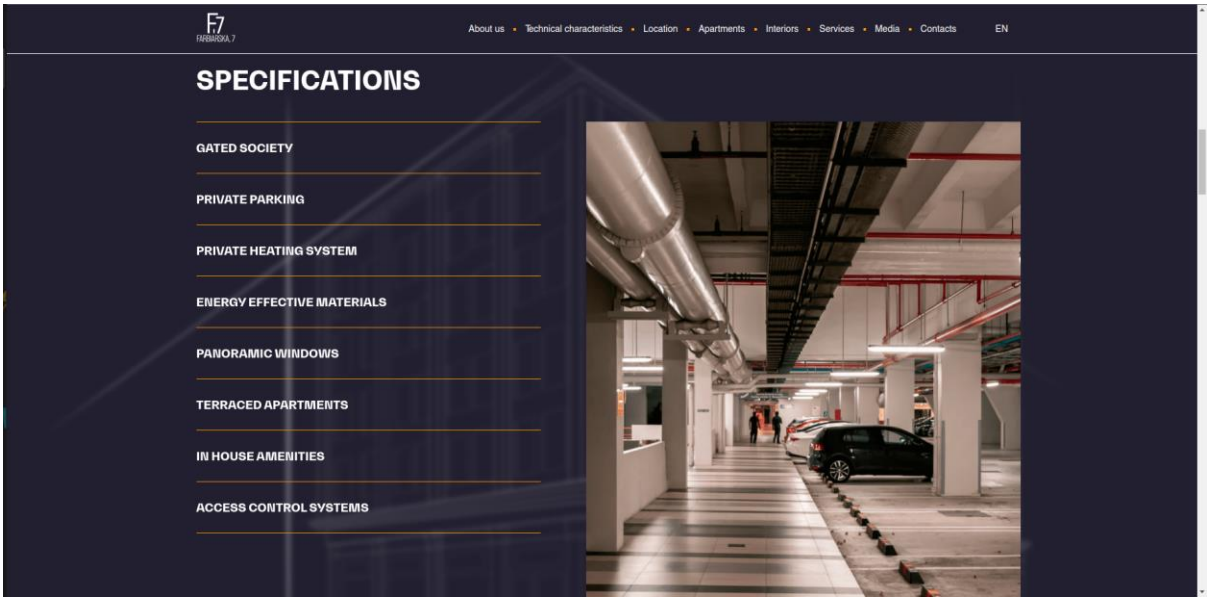


Рисунок 1.2 – Інтерфейс сайту “F7” - https://f7-lublin.pl/

Наступним розглянемо веб сайт компанії AVILA Building Group. Містить приємні для ока світлі кольори (рисунок 1.3), основною перевагою можна виділити обширно описане портфоліо компанії з повним переліком усіх зданих об’єктів (рисунок 1.4), що є дуже важливим аспектом у формуванні довіри

потенційних клієнтів, адже при перегляді портфоліо людина може пригадати що бачила ці будинки та може сформувати перше враження. Також присутня можливість онлайн-звязку із представниками відділу продажів, для уточнення будь яких питань потенційних клієнтів.

На жаль сайт містить досить велику кількість недоліків, серед яких можна виділити: застарілий дизайн та велика кількість текстового контенту на відносно невеликій площі модулів, що заплутує потенційного клієнта при перегляді сайту. Також відсутня можливість зміни локалізації, що складає негативне враження та відштовхує потенційних іноземних клієнтів.

Основним недоліком можна виділити повну відсутність будь якої інформації про стан будівництва об’єктів які зараз знаходяться на етапі будови, що має прямий вплив на кількість роботи з людьми у відділі роботи з клієнтами.

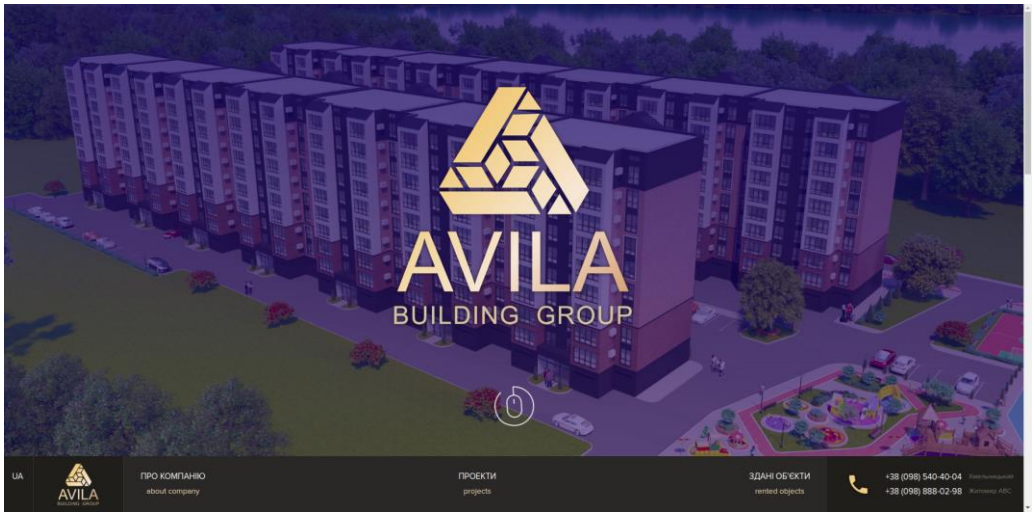


Рисунок 1.3 – Інтерфейс сайту AVILA Building Group - https://avila.com.ua/

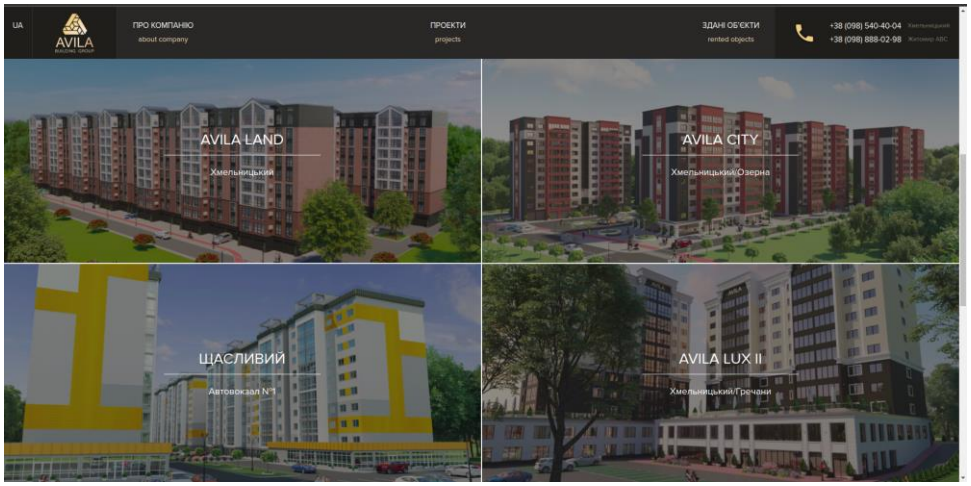


Рисунок 1.4 – Портфоліо AVILA Building Group - https://avila.com.ua/

Наступним розглянемо сайт компанії Me2Me. Одразу можна підкреслити чудовий мінімалістичний дизайн в світлих тонах з великою кількістю привабливих анімацій (рисунок 1.5), наявна велика кількість інформативного контенту закріпленого до відповідних зображень, одразу очевидно що компанія дуже добре попрацювала над інформативністю та якістю контенту. Також перевагою є те що наявна детальна інформація про будинки із усіма статусами та галереями.

Найсуттєвішими перевагами можна виділити те, що сайт містить окремий модуль часто задаваних питань (FAQ), що суттєво зменшує навантаження на відділ роботи із клієнтами, так як відповіді на більшість питань за якими могли б звертатись до працівників компанії потенційні клієнти можуть дізнатись прямо на сайті. Також унікальність даного сайту полягає в тому що потенційний клієнт може прямо на сторінці цікавого йому будинку забронювати його собі, обравши цікаву йому конфігурацію інтер'єру та екстер'єру внісши попередню передплату, і найцікавішим моментом є те що передплату можна здійснити навіть криптовалютою, що є надзвичайно сучасним та зручним рішенням (рисунок 1.6).

З недоліків можна виділити лише відсутність правових документів на сайті, адже при оформлені передплати користувач повинен ознайомитись із політикою приватності, правилами використання та політикою повернення, а за неявності цих документів компаніє порушує права клієнтів.

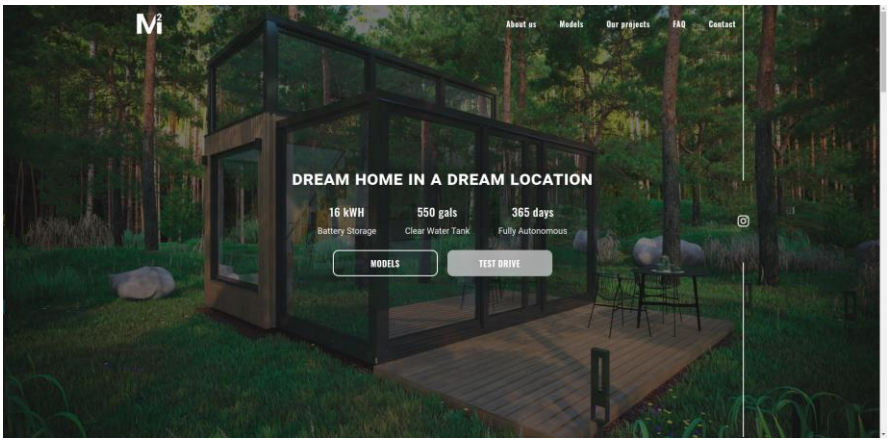


Рисунок 1.5 – Інтерфейс сайту Me2Me - https://me-2.me/

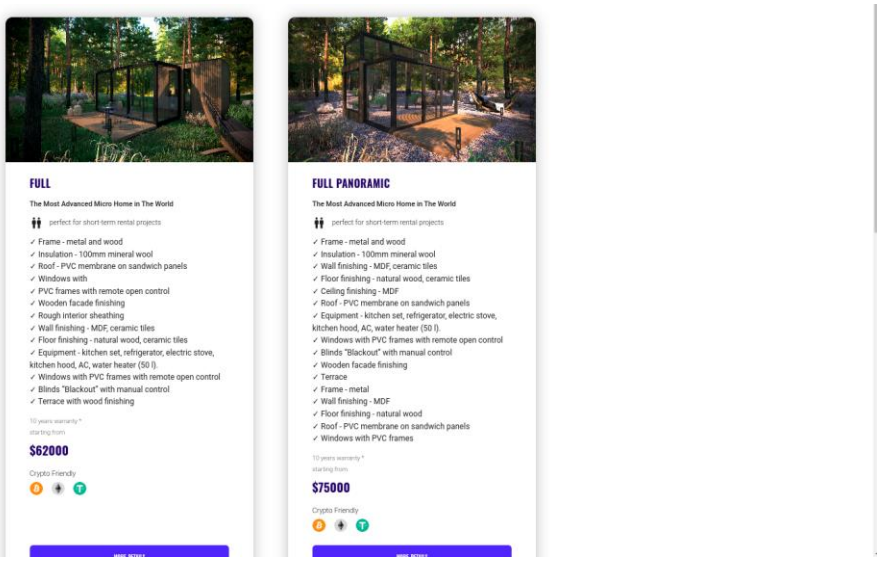


Рисунок 1.6 – Бронювання та передплата будинку - https://me-2.me/

Наступним оглянемо сайт компанії КомфортБудПлюс. Містить приємні для ока світлі кольори (рисунок 1.7), основною перевагою можна виділити обширно описане портфоліо компанії з повним переліком усіх зданих об’єктів (рисунок 1.8), що є дуже важливим аспектом у формуванні довіри потенційних клієнтів, адже при перегляді портфоліо людина може пригадати що бачила ці будинки та може сформувати перше враження. Також присутня можливість онлайн-звязку із представниками відділу продажів, для уточнення будь яких питань потенційних клієнтів.

Сайт містить кілька недоліків, серед яких можна виокремити застарілий дизайн та велика кількість текстового контенту яка розкидана в різних напрямках екрану, що заплутує потенційного клієнта при перегляді сайту. Також відсутня можливість зміни локалізації, що складає негативне враження та відштовхує потенційних іноземних клієнтів.

Основним недоліком можна виділити погану адаптацію веб сайту до мобільних пристроїв, що є дуже відлякуючим фактором для першого враження у потенційного клієнту, адже у сучасному світі 80% переглядів веб сайтів відбуваються саме з мобільних пристроїв.

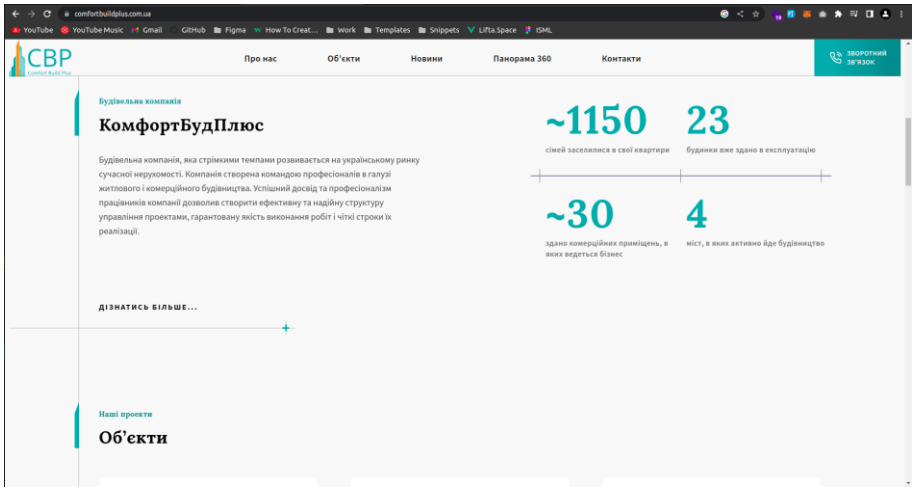


Рисунок 1.7 – Інтерфейс сайту КомфортБудПлюс - https://comfortbuildplus.com.ua/

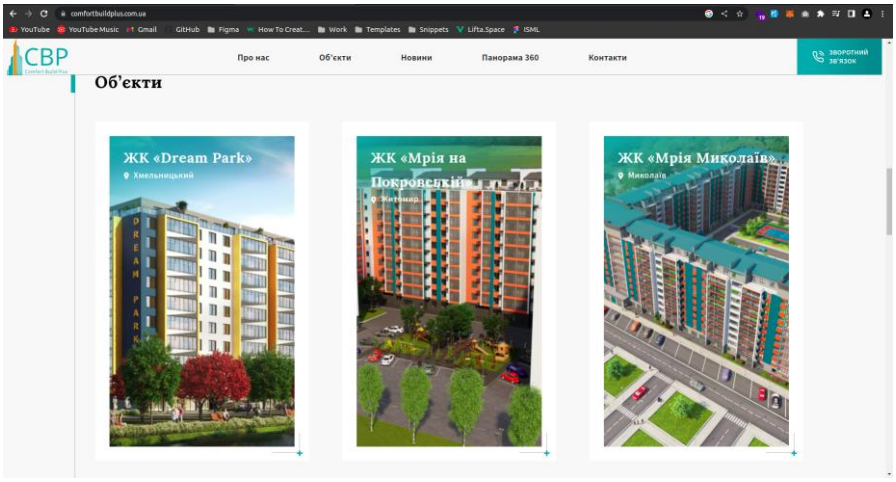


Рисунок 1.8 – Портфоліо сайту КомфортБудПлюс - https://comfortbuildplus.com.ua/

Останнім розгянемо веб-сайт компанії IDM Group. Можна підкреслити мінімалістичний дизайн в світлих тонах з великою кількістю крупних зображень (рисунок 1.9), наявна велика кількість інформативного контенту закріпленого до відповідних зображень. Також перевагою є те що наявна детальна інформація про будинки із можливістю переходу на окремий веб-сайт конкретної будівлі.

Найсуттєвішими перевагами можна виділити те, що сайт містить окремий модуль новин, що суттєво зменшує навантаження на відділ роботи із клієнтами, так як відповіді на більшість питань за якими могли б звертатись до працівників компанії потенційні клієнти можуть дізнатись прямо на сайті. Також унікальність даного сайту полягає в тому що замість монолітного сайту зі всіма будівлями, компанія розробила платформу яка перенаправляє людей між

окремими сайтами різних будівель, що є надзвичайно сучасним та розумним рішенням, адже при правильному використанні спеціальних інструментів, таких як Google Analytics, компанія може відслідковувати які будівлі привертають більше уваги потенційних клієнтів.

З недоліків можна виділити лише відсутність можливості онлайн звязку із відділом продажів, що суттєво ускладнює комунікацію клієнта із компанією.

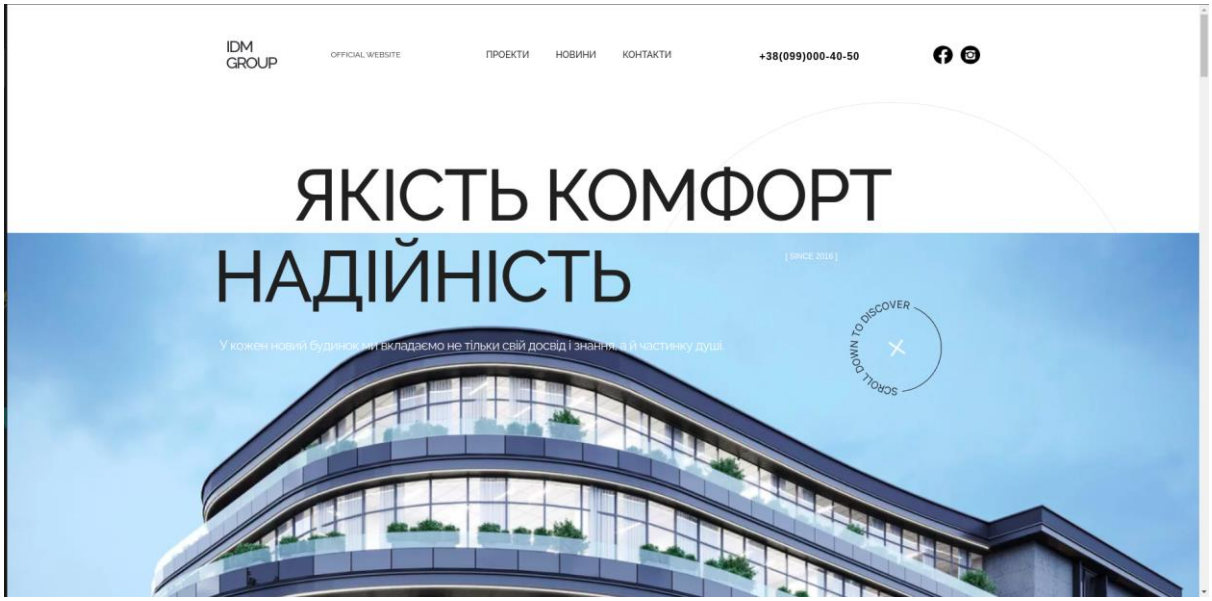


Рисунок 1.9 – Інтерфейс сайту IDM Group - https://idmgroup.com.ua/

Після детального аналізу всіх веб- за стосунків можна здійснити порівняльний огляд всіх розглянутих сайтів за різними особливостями. Основною властивістю сайтів є їх дизайн, а саме можливість швидко зафіксувати увагу та справити гарне перше враження на користувача. Далі надважливим є оцінка якості інтерфейсу, адже це головне, на що звертає увагу користувач. Також важливо порівняти такі функції, як присутність портфоліо, та наявність системи бронювання, щоб користувач міг одразу сформувати своє замовлення. Крім того важливим є те, чи може користувач звязатися із відділом продажів чи та переглянути статус будівництва.

В таблиці 1.1 зроблено порівняння сайтів за цими властивостями.

Таблиця 1.1 – Порівняльна таблиця застосунків

Назва застосунку	Дизайн	Інтерфейс	Портфоліо	Система бронювання	Онлайн зв'язок	Статус будівництва
F7	8/10	8/10	Відсутнє	Відсутня	Присутній	Присутній
AVILA Building Group	5/10	6/10	Присутнє	Відсутня	Присутній	Відсутній
Me2Me	10/10	9/10	Відсутнє	Присутня	Присутній	Відсутній
Комфорт БудПлюс	6/10	7/10	Присутнє	Відсутня	Присутній	Присутній
IDM Group	9/10	9/10	Присутнє	Відсутня	Відсутній	Відсутній

Таким чином, дослідивши кілька існуючих рішень для компаній забудовників, можна зробити підсумки. Більшість з цих веб сайтів мають зрозумілий для користувачів, однак не зовсім гарний та сучасний інтерфейс, який на сьогодні вже потребує оновлення. Також, із з’ясованого, переважна більшість сайтів дозволяє огляд всіх об’єктів які були побудовані компанією. З негативних моментів можна виділити те, що більше, ніж в половини випадків зустрічається відсутність статусів будови. Крім того, на більшості сайтів немає системи бронювання, користувач лише може переглянути планування та фото. Отже, проведений аналіз існуючих аналогів та їх особливостей дає можливість визначити основні характеристики, функціональні та нефункціональні потреби розроблюваного застосунку, та його релевантності.

1.3 Визначення вимог до програмного продукту

Після дослідження предметної області та аналізу існуючих програмних продуктів було розроблено технічне завдання та список вимог, які повинні бути враховані при розробці веб-сайту для клієнтів компанії-забудовника.

Цей веб-сайт має мати кілька особливостей, що роблять його унікальним порівняно з іншими аналогами. Найважливішою з них є те, що платформа має бути універсальною та складатися з двох частин: клієнтської та адміністраторської.

Крім того, веб-сайт повинен мати можливість перегляду галерей, портфоліо, наявних у продажі планувань квартир, статусів будівництва, можливості онлайн-звернення до відділу продажу для отримання будь-якої інформації про об’єкт, модуля конфігурування готових інтер’єрів квартир та можливості бронювання. Крім того, веб-сайт повинен підтримувати різні мови, оскільки його будуть використовувати користувачі з різних країн.

З урахуванням цих особливостей можна скласти список функціональних можливостей, які повинні бути доступні на веб-сайті.

Для звичайного користувача:

- перегляд інформаційних блоків про компанію;
- перегляд портфоліо компанії;
- перегляд галерей будівництва;
- окремі сторінки під різні будівлі з їх наявними плануваннями та статусами будови;
- можливість звернутись онлайн до відділу продажів;
- можливість переглянути юридичні документи компанії;
- можливість конфігурації готового інтер’єру квартири;
- можливість бронювання покупки квартири;
- можливість перегляду веб-сайту на різних мовах;

Для адміністратора:

− авторизація;

− створення ролей адміністраторів з відповідними доступами до редагування вмісту сайту;

− можливість створення та редагування блоків про компанію;

− можливість додавання об’єктів у портфоліо компанії;

− можливість завантаження фото/відео у галереї на веб-сайті;

− можливість створення та редагування окремих сторінок будівель з усією необхідною інформацією;

− можливість оновлювати статуси будови;

− можливість завантаження планувань будови;

− можливість опрацювання усіх вхідних онлайн-звернень з веб-сайту;

− можливість загрузки та оновлення необхідних юридичних документів;

− можливість створення модулів конфігурацій готових квартир;

− система обліку усіх вхідних заявок на бронювання;

− можливість редагування усього контенту сайту на різних мовах.

Також сформовано наступні вимоги до інтерфейсу та інші вимоги:

− респонсивна верстка, чутлива до мобільних пристроїв;

− світлий дизайн із темними текстами для пом'якшення зорового контакту із сторінкою;

− анімації на контент для приваблення уваги користувача;

− відкриття зображень у галереях на повний екран при кліку на зображення для детального перегляду усіх деталей які цікавлять користувача;

− інтуїтивно зрозуміла послідовність та розміщення блоків для уникнення невідповідностей та дезорієнтації користувача;ы

− високий рівень оптимізації веб сайту як по швидкісним показникам, так і по SEO параметрам.

Уніфікована мова моделювання UML використовується для чіткого формулювання вимог до програмного забезпечення. Діаграми UML

використовуються для візуального зображення системи, включаючи її сутність, такі як ролі, актори, класи, дії та інші елементи, що допомагають узгодженому розумінню або документуванню інформації про системи. Ці діаграми складаються з схематичних представлень елементів програми.

Для нашої системи ми складаємо описи акторів (користувачів системи) та перераховуємо всі можливі дії (варіанти використання), які вони можуть виконувати.

У таблиці 1.2 наведені актори розроблюваного програмного застосунку.

Таблиця 1.2 – Опис акторів розроблюваного програмного застосунку

Актор	Короткий опис
Користувач	Може переглядати клієнтську частину веб-застосунку для пошуку необхідної інформації та залишення заявок на бронювання або зв'язку із відділом продажів.
Адміністратор	Має змогу редагувати веб-застосунок з адмін панелі та опрацьовувати вхідні заявки із зв'язку з відділом продажів та бронювання будинку.

В таблиці 1.3 наведені способи використання розроблюваного веб-застосунку.

Таблиця 1.3 – Опис способів використання розроблюваного веб-застосунку.

Актор	Короткий опис
Користувач	Може переглянути головну сторінку, сторінку про компанію, перейти на сторінку зв'язку з відділом

	продажів, перейти у каталог доступних будинків, перейти на сторінку одного будинку, переглянути необхідну інформацію, переглянути галереї будови, обрати комплектацію готової оселі, при необхідності забронювати будинок.
Адміністратор	Може авторизуватися, після чого йому стає доступна панель керування сайтом, яка включає в себе зміну перекладів сайту, змінних, наповнювати галереї, створювати будинки та їх комплектації, опрацьовувати вхідні заявки на бронювання будинку та зв'язку із відділом продажів.

Визначивши акторів системи та варіанти використання, було побудовано діаграму варіантів використання (рисунок 1.10).

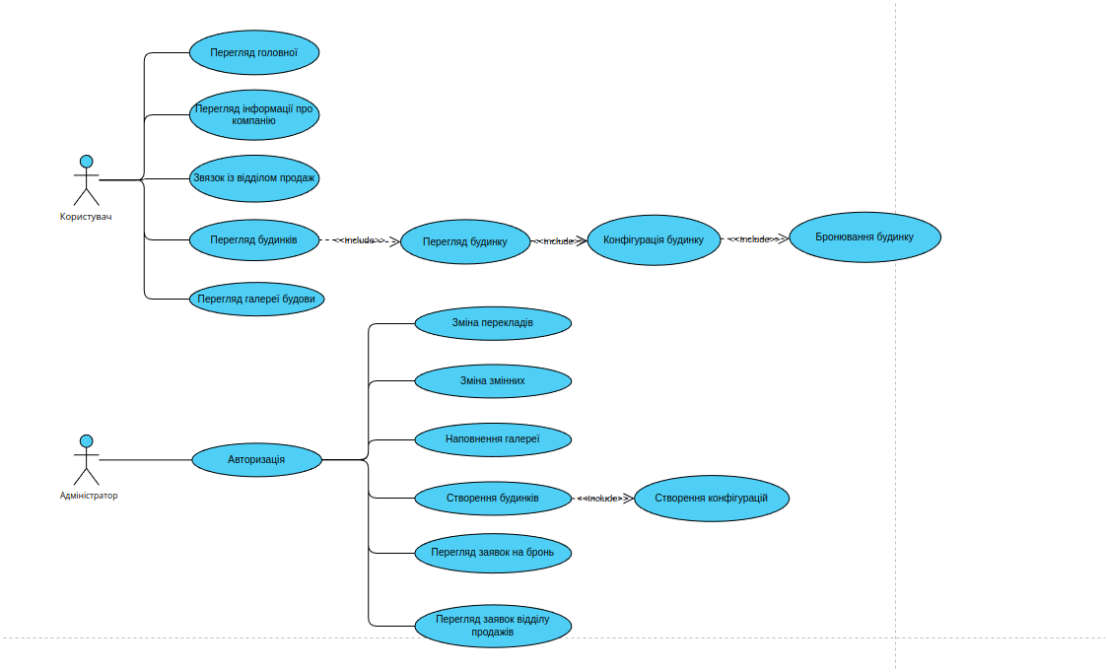


Рисунок 1.10 – Діаграма варіантів використання

1.4 Постановка задачі

Після завершення аналізу вимог до програмного продукту було створено технічне завдання, яке можна знайти у додатку А і використовувати як основу

для подальшої розробки веб-застосунку для автоматизації бізнес процесів компанії забудовника.

При розробці кваліфікаційної роботи "Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника" потрібно вирішити такі задачі:

- Розробити простий та приваблий інтерфейс користувача для зручного користування веб-застосунком.
- Розробити систему наповнення веб-застосунку контентом та створення каталогу будівель.
- Забезпечити можливість зворотнього зв’язку з відділом продажів та бронювання будинків користвуачами веб-застосунку.
- Розробити систему управління заявками на бронювання та зв’язку із відділом продажів.

2 ПРОЕКТУВАННЯ ВЕБ-ЗАСТОСУНКА

2.1 Аналіз та вибір архітектури веб-застосунка

Більшість веб-сайтів, які щодня можна знайти в Інтернеті, ґрунтуються на клієнт-серверній архітектурі. Цей вид архітектури має два основні компоненти – клієнт і сервер. Основний принцип і головна перевага цього підходу полягають у тому, що більшість інформації знаходиться на сервері, що дозволяє клієнту працювати швидше. Клієнтом може бути будь-яке програмне забезпечення з зовнішнім інтерфейсом, яке надсилає запити для передачі та отримання даних віддаленому серверу.

Сервер – це апаратне забезпечення, яке отримує запити від клієнтів, обробляє їх і надсилає дані у форматі, зрозумілому клієнту. Він також виконує різноманітні завдання, пов'язані з обчисленнями і логічними операціями. Сервер зазвичай є потужною комп'ютерною системою, яка може обробляти багато запитів одночасно. Запити обробляються послідовно, а відповіді надсилаються клієнтам для максимально швидкого функціонування. Сервери зазвичай працюють постійно, оскільки клієнти можуть надсилати запити в будь-який час.

Є випадки, коли програми-клієнти та програми-сервери розміщені на одному комп'ютері, це називається локальним сервером. Цей підхід часто використовується при розробці програмного забезпечення. Проте в більшості випадків клієнт і сервер знаходяться на різних комп'ютерах та взаємодіють через Інтернет за допомогою протоколу передачі даних HTTP – HyperText Transfer Protocol. HTML, що означає HyperText Markup Language, використовується для представлення даних, які сервер надсилає клієнтові, а клієнт зберігає та відображає їх користувачу на екрані.

Пізніше був розроблений розширений протокол передачі даних HTTPS (HyperText Transfer Protocol Secure). Він використовує SSL (Secure Sockets Layer) для шифрування даних, які передаються між клієнтом і сервером. Це шифрування забезпечує захист конфіденційної інформації та створює безпечне

з'єднання. HTTPS зараз широко використовується на більшості веб-сайтів, щоб захистити персональну інформацію користувачів.

З ростом кількості застосунків, які використовують клієнт-серверну архітектуру, почали з'являтися різні архітектурні рішення для створення та керування серверними API. Один з найпоширеніших шаблонів – REST (Representational State Transfer) – передача стану представлення. REST дозволяє обробляти запити відповідно до методу, яким вони були надіслані. Цей метод вказує, яку дію потрібно виконати – отримання, оновлення, видалення або створення даних. REST також надає клієнтам інформацію про статус виконання запиту за допомогою трьохзначного коду. Наприклад, успішний запит поверне код "200", а невдалий запит може повернути код "500" або "404", залежно від типу помилки.

Крім того, REST не накладає обмежень на розмір та формат передаваних даних і ресурсів.

2.2 Опис структури даних та моделі бази даних

Переважна більшість сучасних веб-сайтів зберігає інформацію про користувачів та інші важливі дані в різних типах сховищ, відомих як бази даних. Бази даних забезпечують надійне зберігання даних і забезпечують доступ до них за потреби. Вибір відповідної бази даних на етапі проектування програмного забезпечення є критичним, оскільки це впливає на швидкість роботи системи, можливість майбутньої підтримки програмного продукту та інші важливі фактори.

База даних (БД) - це структурована колекція інформації, організована у вигляді таблиць, де дані зберігаються, оновлюються та використовуються для подальшої обробки. Бази даних використовуються для збереження великої кількості даних, які пов'язані між собою, і надають можливість ефективно виконувати операції з цими даними. Бази даних використовуються для різних цілей, і основні з них включають:

Зберігання даних: БД дозволяють зберігати великі обсяги структурованих даних, таких як текст, числа, дати, зображення і відео. Вони дозволяють зручний доступ до цих даних, що спрощує їх збереження та організацію.

Організація даних: Бази даних надають зручну структуру для організації даних у вигляді таблиць, де кожен стовпець відповідає певному типу даних, а кожний рядок містить конкретну інформацію. Це дозволяє легко орієнтуватися в даних і швидко здійснювати пошук та фільтрацію.

- Операції з даними: Бази даних надають можливість виконувати різні операції з даними, такі як додавання, видалення, оновлення та пошук. Це дозволяє легко змінювати, оновлювати та отримувати інформацію з бази даних.
- Забезпечення цілісності даних: БД мають механізми для забезпечення цілісності даних, що означає, що дані зберігаються і оновлюються в безпечний і послідовний спосіб. Це включає валідацію даних, захист від несанкціонованого доступу та забезпечення конфіденційності.
- Поділ даних: Бази даних дозволяють одночасний доступ до даних кільком користувачам або застосункам. Це особливо важливо в організаціях, де багато людей мають різний рівень доступу до даних.
- Аналітика та звітність: Бази даних надають засоби для аналізу даних та створення звітів і статистичних звітів. Це дозволяє виявляти тенденції, знаходити корисну інформацію та приймати обґрунтовані рішення на основі даних.

Бази даних є важливим інструментом для багатьох галузей, таких як бізнес, наука, організація даних, соціальні мережі та багато інших. Вони дозволяють ефективно управляти, зберігати та обробляти великі обсяги даних, що є ключовим для успішної роботи багатьох сучасних систем і застосунків.

У сучасний період існує два основних типи систем управління даними: реляційні та нереляційні бази даних.

Реляційні бази даних є найпоширенішими, оскільки вони структурують збережену інформацію у вигляді таблиць. Кожна таблиця складається з рядків, що містять поля або ключі, за допомогою яких можна отримати дані про певний

об'єкт або сутність. Ці поля містять атрибути об'єктів. Стовпці таблиці описують характеристики цих атрибутів. Кожне поле має чітко визначений тип даних і відображає лише одну властивість об'єкта, таку як електронна пошта, назва книги, номер телефону і так далі.

У реляційній базі даних кожна таблиця повинна мати особливий атрибут, який служить унікальним ідентифікатором. Цей атрибут називають первинним ключем.

Реляційні бази даних мають свої переваги і недоліки. До переваг відносяться:

− просте розширення бази даних при збільшенні обсягу даних;

− наявність багатьох спеціальних програм, які називаються СКБД (системи керування базами даних) і надають адміністратору бази даних різноманітні корисні функції, такі як очищення таблиць, сортування та пошук даних та багато іншого;

− чітко визначена структура бази даних, що надає чітке уявлення про предметну область та сутності;

− зрозумілий спосіб представлення даних користувачам за допомогою таблиць та зв'язків.

Серед недоліків реляційних баз даних можна виокремити такі фактори:

− у великих проектах велика кількість таблиць та зв'язків можуть ускладнити розуміння структури бази даних і призвести до заплутаності;

− швидкість запису та витягування даних не є найвищою;

− бази даних вимагають значних фізичних ресурсів, оскільки обсяг пам'яті зростає зі збільшенням обсягу даних, що може призвести до погіршення продуктивності;

− не завжди можна належним чином описати предметну область за допомогою таблиць;

Реляційні бази даних повинні відповідати певним вимогам:

− атомарність: кожен атрибут повинен мати лише одне значення, а не множину значень;

− цілісність: всі частини бази даних повинні бути пов'язані в одне ціле;

− ізольованість: сутності бази даних не повинні залежати одна від одної.

− довговічність: забезпечення збереження та відновлення даних після збоїв у роботі бази даних після внесення або зміни запису.

Для спрощення роботи з базами даних застосовують різні системи керування базами даних (СКБД), такі як SQLLite, MySQL, PostgreSQL, MariaDB і інші. Ці системи дозволяють працювати з даними безпосередньо в базі, забезпечуючи можливість створення, пошуку, редагування та видалення конкретних даних з бази даних, забезпечуючи безпечний доступ.

Крім реляційних баз даних існують також нереляційні бази даних, які часто називаються NoSQL. Вони пропонують зовсім іншу концепцію створення та доступу до бази даних. Особливістю цих баз даних є відсутність структури в табличному вигляді, що надає більш гнучкі можливості управління даними. Це особливо корисно для роботи з великими розподіленими наборами даних. NoSQL бази даних характеризуються масштабованістю, високою продуктивністю та гнучкістю. Ці бази даних стають все більш популярними, оскільки вони використовуються в за стосунках з великим обсягом даних, який постійно зростає і вимагає обробки в реальному часі.

Існують різні типи NoSQL баз даних, і серед них можна виділити чотири основних типи.

− бази даних документів: цей тип баз даних зберігає дані у вигляді документів, схожих на об'єкти JSON (JavaScript Object Notation). Кожен документ містить пари полів і значень, де значення можуть бути різних типів, таких як рядки, числа, логічні значення, масиви або об'єкти. Система ієрархічно зберігає дані у вигляді дерева, що сприяє швидкому пошуку по всій базі даних. Цей тип бази даних корисний для зберігання впорядкованих даних та випадків, коли немає потреби у великій кількості зв'язків між даними або збору статистики. Деякими прикладами СКБД цього типу є MongoDB, eXist, CouchDB, MarkLogic;

− ключ-значення бази даних: у цьому типі баз даних дані зберігаються у вигляді пар ключ-значення. Кожен запис має унікальний ключ, за допомогою якого можна отримувати доступ до відповідного значення. Цей тип бази даних ефективний для швидкого пошуку за ключем, але не підтримує складні запити і зв'язки між даними. Прикладами СКБД цього типу є Redis, Riak, Amazon DynamoDB;

− сімейство стовпчикових баз даних: цей тип баз даних орієнтований на зберігання даних у вигляді стовпців замість рядків. Дані групуються за стовпцями, а не за рядками, що робить їх ефективними для роботи з великими обсягами даних та виконання складних аналітичних запитів. Деякими прикладами СКБД цього типу є Apache Cassandra, HBase, ScyllaDB;

− графові бази даних: цей тип баз даних орієнтований на зберігання та обробку зв'язків між об'єктами. Дані представлені у вигляді графів, де вершини представляють об'єкти, а ребра відображають зв'язки між ними. Графові бази даних дозволяють ефективно виконувати складні запити, такі як пошук шляху між об'єктами або аналіз мережі. Деякими прикладами СКБД цього типу є Neo4j, Amazon Neptune, JanusGraph.

Ці різні типи NoSQL баз даних надають гнучкість та ефективність у різних сценаріях роботи з даними, в залежності від потреб проекту.

MySQL є однією з найпопулярніших реляційних систем управління базами даних (СУБД). Це вільна та відкрита програма, яка дозволяє зберігати, керувати та отримувати доступ до великих обсягів даних. MySQL використовує мову запитів SQL (Structured Query Language) для взаємодії з базою даних.

Завдяки своїй простоті та широкому колу функцій, MySQL знайшла застосування у багатьох веб-застосунках, включаючи веб-сайти, блоги, електронну комерцію та багато іншого. Вона пропонує надійне зберігання та швидкий доступ до даних, забезпечуючи високу продуктивність та ефективність.

MySQL підтримує розподілені бази даних, багатопоточну обробку запитів та масштабованість, що дозволяє використовувати її у великих проектах

з великою кількістю користувачів та обсягами даних. Крім того, вона має широку спільноту розробників і підтримується багатьма платформами, такими як Windows, Linux та macOS.

MySQL – це популярна реляційна система управління базами даних (СУБД) з великою кількістю особливостей. Ось деякі з них:

− простота використання: MySQL має простий інтерфейс та зрозумілу структуру, що робить його дружнім для новачків. Він пропонує чіткі команди та синтаксис мови запитів SQL, що дозволяє легко взаємодіяти з базою даних;

− висока продуктивність: MySQL працює дуже швидко і ефективно навіть з великими обсягами даних. Він має оптимізований двигун запитів, який дозволяє швидко виконувати складні запити і оптимізувати роботу з базою даних;

− масштабованість: MySQL може легко масштабуватися для використання у великих проектах. Він підтримує розподілені бази даних, що дозволяє розподілити навантаження між багатьма серверами. Це робить його ідеальним вибором для веб-застосунків з великою кількістю користувачів;

− надійність і безпека: MySQL забезпечує надійне збереження даних і має вбудовані механізми резервного копіювання та відновлення. Він також пропонує різні рівні захисту даних, включаючи автентифікацію, шифрування та контроль доступу до бази даних;

− підтримка: MySQL має велику спільноту розробників та активну підтримку з боку Open Source Community. Це означає, що ви можете швидко знайти документацію, плагіни, оновлення та відповіді на свої запитання в разі потреби;

− кросплатформеність: MySQL підтримує різні операційні системи, включаючи Windows, Linux і macOS. Це дозволяє вам використовувати його на будь-якій платформі залежно від ваших потреб і вимог.

Загалом, MySQL є потужним інструментом для організації та управління базами даних, забезпечуючи надійність, швидкість та гнучкість у використанні.

Проаналізувавши переваги і недоліки різних типів баз даних, було вирішено розробляти завдання кваліфікаційної роботи з використанням реляційної бази даних MySQL.

Беручи за основу створену попередньо діаграму варіантів використання (див. рисунок 1.14), було визначено колекції даних, які будемо зберігати в базі даних.

Таким чином було створено велику кількість колекцій даних, побудована структура бази даних (рисунок 2.1).

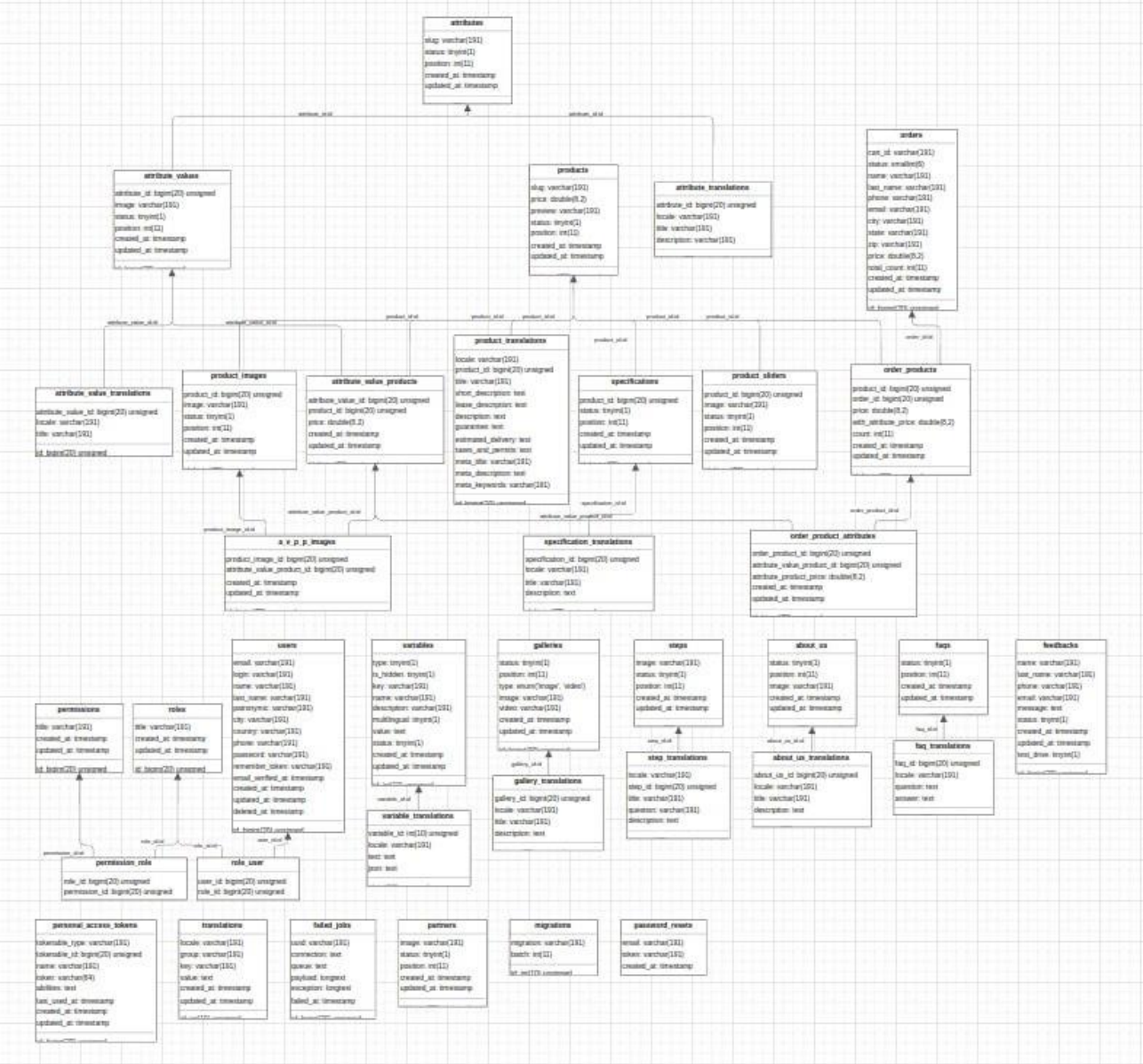


Рисунок 2.1 – Структура бази даних

В результаті, отримано значну кількість полів, які є необхідними для належної функціональності розроблюваного веб-сайту.

Варто відзначити, що деякі колекції мають повторювані поля, такі як UserId. Це здійснено з метою зв'язку з відповідними колекціями, що стосуються користувача та інформації про будинки.

Отже, була розроблена логічна модель бази даних, і згідно з нею можна перейти до проектування фізичної моделі.

2.3 Проектування серверної частини веб-застосунка

Серверна складова є важливою частиною програмного продукту, оскільки вона відповідає за основні функції системи. Сервер виконує завдання зв'язку між базою даних і тим, що відображається користувачеві на екрані, тобто клієнтською частиною.

Для створення та керування запитами використовується REST API – програмний інтерфейс, який надає можливість взаємодії з сервером через стандартні HTTP запити.

REST API (Representational State Transfer Application Programming Interface) – це архітектурний стиль для створення веб-сервісів, який дозволяє взаємодіяти з сервером за допомогою стандартних HTTP запитів, таких як GET, POST, PUT та DELETE. REST API використовує принципи роботи з ресурсами, які ідентифікуються унікальними URL-адресами.

За допомогою REST API можна отримувати, створювати, оновлювати та видаляти дані на сервері. Клієнтська програма може відправляти запити до сервера, вказуючи метод запиту (GET для отримання даних, POST для створення, PUT для оновлення, DELETE для видалення) та URL-адресу ресурсу, з яким потрібно взаємодіяти.

REST API передає дані у форматі, який часто є JSON (JavaScript Object Notation) або XML (eXtensible Markup Language), і зазвичай повертає відповідь у вигляді статусного коду та вмісту даних.

Цей підхід до побудови веб-сервісів є широко використовуваним у сучасному розробці програмного забезпечення, оскільки дозволяє розподіленій системі бути масштабованою, гнучкою та легкодоступною для різних типів клієнтів.

Важливим аспектом є спосіб представлення цих запитів та відповідей. Ці формати визначаються згідно зі стандартним протоколом HTTP.

На сьогоднішній день можна виділити два основні та найпоширеніші підходи до розробки серверів: монолітний та мікросервісний. Розглянемо переваги та недоліки кожного з цих підходів.

Монолітна архітектура веб-застосунків є традиційним підходом до розробки, де весь функціонал застосунку об'єднаний в єдиному компоненті або "моноліті". У такій архітектурі весь код, логіка та база даних знаходяться разом і виконуються в одному процесі.

Монолітна архітектура веб-застосунків може бути організована на різних рівнях, що визначають функціональність та відповідальність окремих компонентів системи. Зазвичай розрізняють три основних рівня: рівень презентації (presentation layer), рівень бізнес-логіки (business logic layer) та рівень доступу до даних (data access layer).

Рівень презентації (presentation layer): Цей рівень відповідає за відображення інтерфейсу користувача та взаємодію з ним. Він містить компоненти, такі як HTML-шаблони, CSS-стилі, JavaScript-скрипти, які відповідають за структуру та зовнішній вигляд веб-сторінок. Рівень презентації забезпечує взаємодію з користувачем шляхом обробки HTTP-запитів та надсилання відповідей.

Рівень бізнес-логіки (business logic layer): Цей рівень містить логіку застосунку, яка відповідає за обробку та управління даними. Він включає в себе функції та процедури, що виконують бізнес-логіку, такі як обробка та перевірка даних, взаємодія з базою даних, обчислення та логіка роботи застосунку. Рівень бізнес-логіки вирішує конкретні завдання та бізнес-правила, пов'язані з функціональністю застосунку.

Рівень доступу до даних (data access layer): Цей рівень відповідає за збереження та доступ до даних у базі даних. Він включає в себе методи та функції для звернення до бази даних, виконання запитів, збереження та витягування даних. Рівень доступу до даних дозволяє взаємодіяти з базою даних та забезпечує доступ до неї з рівнів презентації та бізнес-логіки.

Ці рівні монолітної архітектури допомагають організувати функціональність та відповідальність різних компонентів застосунку та представлено на рисунку 2.2.

Рисунок 2.2 – Монолітна архітектура

Розміщення логіки на відповідних рівнях сприяє модульності та покращує розуміння коду.

Основною перевагою монолітної архітектури є її простота. Всі компоненти застосунку розташовані в одному місці, що полегшує розробку, тестування та випуск нових функцій. Також, спільний доступ до бази даних дозволяє зручно організовувати зв'язки та запити між різними модулями.

Однак, монолітна архітектура може мати свої недоліки. З часом, коли додаток зростає, стає складніше розуміти та підтримувати весь код, оскільки зміни в одній частині можуть мати неочікувані наслідки в інших частинах. Також, масштабування застосунку може бути викликом, оскільки всі компоненти повинні бути масштабованими разом.

Загалом, монолітна архітектура є простим та поширеним підходом до розробки веб-застосунків, особливо на ранніх етапах. Вона підходить для невеликих та середніх проектів, де простота розробки та швидкість випуску нових функцій мають велике значення.

Мікросервісна архітектура представляє собою альтернативний спосіб розробки серверних застосунків.

Мікросервісна архітектура є підходом до розробки веб-застосунків, в якому програмне забезпечення розбивається на невеликі, самостійні та незалежні компоненти, відомі як мікросервіси. Кожен мікросервіс функціонує як окремий процес та виконує обмежену функціональність.

У мікросервісній архітектурі, кожен мікросервіс спеціалізується на виконанні конкретного завдання або функції, таких як автентифікація користувачів, обробка платежів, керування товарним каталогом тощо. Кожен мікросервіс може бути розроблений, масштабований та випускатись незалежно від інших компонентів системи.

Ці мікросервіси комунікують між собою за допомогою мережевих протоколів, таких як HTTP або AMQP, і взаємодіють з іншими мікросервісами для обміну даними та виконання комплексних операцій. Кожен мікросервіс може мати свою власну базу даних або використовувати спільну базу даних з іншими сервісами.

Структура мікросервісної архітектури представлена на рисунку 2.3.

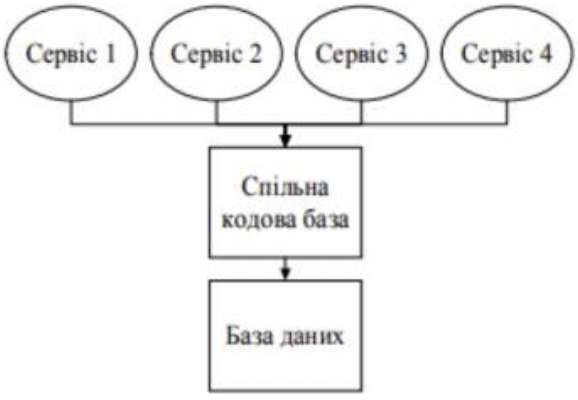


Рисунок 2.3 – Мікросервісна архітектура

Основна перевага мікросервісної архітектури полягає в її гнучкості та масштабованості. Оскільки кожен мікросервіс функціонує незалежно, розробники можуть розгортати, масштабувати та керувати окремими компонентами системи незалежно один від одного. Це дозволяє ефективно розподіляти ресурси та легко впроваджувати зміни без необхідності перебудови всієї системи.

Однак, мікросервісна архітектура також має свої виклики. Управління багатьма незалежними мікросервісами може бути складним завданням, а також потребує додаткової уваги до моніторингу, логування та безпеки. Крім того, взаємодія між мікросервісами через мережу може вплинути на продуктивність системи.

Загалом, мікросервісна архітектура дозволяє розробникам будувати гнучкі та масштабовані веб-застосунки, розбиваючи їх на невеликі та зручно керовані компоненти. Цей підхід дозволяє швидко впроваджувати зміни та розширювати функціональність системи з мінімальним впливом на решту компонентів.

Таким чином, можна зробити висновок, що монолітна архітектура є доцільним варіантом для простих та невеликих програмних систем, які не вимагають великих обчислювальних ресурсів. У той же час, мікросервісна архітектура підходить для складних систем з багатьма незалежними модулями.

Таким чином, у даному дипломному проекті було обрано монолітну архітектуру, оскільки вона відповідає середній складності проекту та обмеженому часу розробки. Модулі в цій системі мають взаємозв'язки, тому розбиття їх на окремі сервіси було б неефективним і зайняло багато часу для розгортання та тестування.

2.4 Проектування інтерфейсу користувача

Для розробки дизайну проекту було прийнято рішення використовувати метод прототипування.

Прототипізація у контексті веб-застосунків – це процес створення прототипів або моделей веб- застосунків для визначення їхнього функціоналу, інтерфейсу та взаємодії з користувачем. Прототип є проміжним результатом, який демонструє основні характеристики та вигляд майбутнього веб-застосунку. Він може бути представлений у вигляді набору скріншотів, проводити навчання через макети або навіть мати обмежений функціонал, який дозволяє протестувати основні функції системи.

Прототипізація дозволяє команді розробників та зацікавленим сторонам отримати уявлення про те, як буде виглядати та працювати веб-застосунок задовго до його повного розроблення. Це дозволяє виявити та виправити можливі проблеми, врахувати побажання користувачів та забезпечити оптимальну взаємодію між користувачем і додатком. Прототипізація допомагає зберегти час, зусилля та ресурси, оскільки дозволяє виявити потенційні проблеми та внести необхідні зміни на ранніх етапах розробки.

Спочатку потрібно описати верхню частину веб-сторінки, яка іменується «header». Він повинен в собі містити логотип, навігаційне меню переходу по сторінкам сайту, посилання на соціальні мережі власників веб-застосунку (рисунок 2.4).

Рисунок 2.4 – “Header”

Далі необхідно провести прототипування загальних сторінок розроблюваного веб-застосунку.

Розпочнемо з головної сторінки Тут будуть розміщені наступні елементи:

− заголовок;

− CTA (Call to Action) - елементи або кнопки, які спонукають користувачів виконати певну дію ;

− інформаційні блоки;

– форма зворотнього зв'язку із відділом продажів.

Макет головної сторінки представлено на рисунку 2.5.

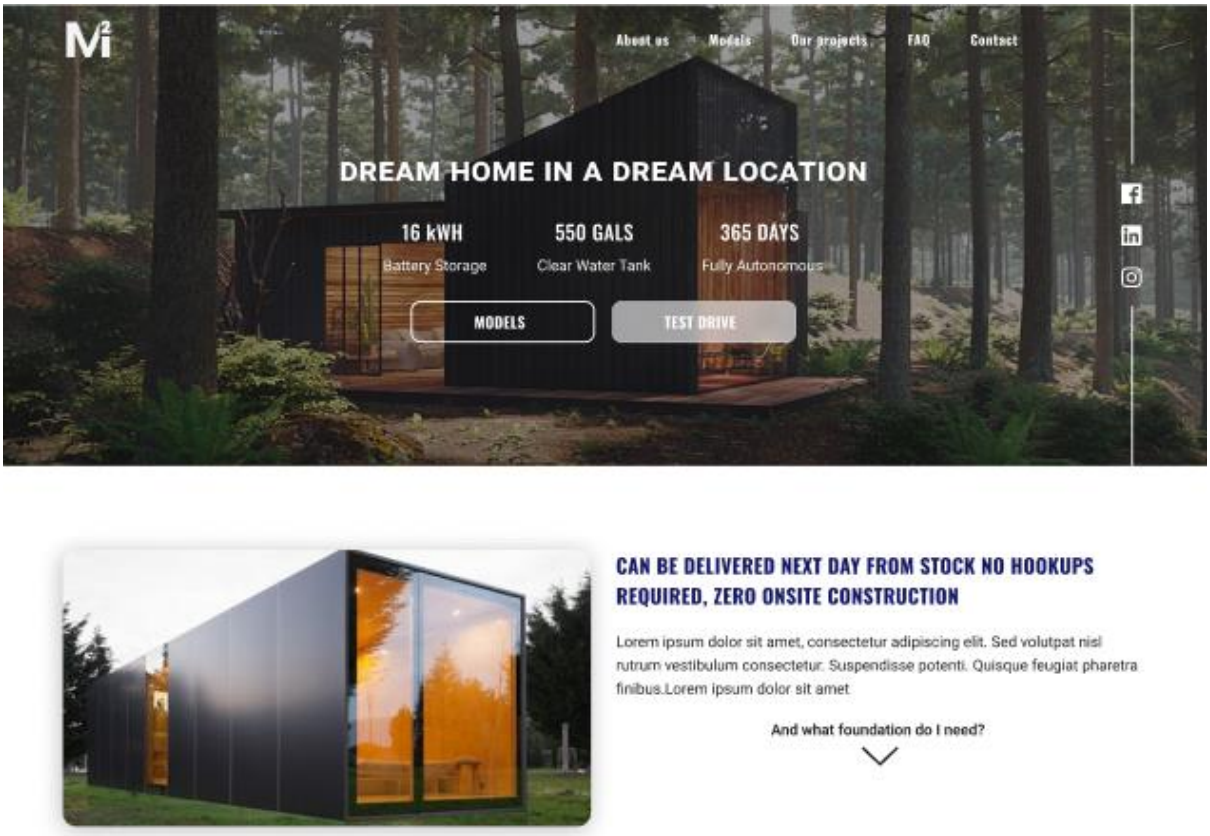


Рисунок 2.5 – Макет головної сторінки

Наступним кроком буде проектування макету для сторінки про компанію яка буде містити в собі текстові блоки з зображеннями. Макет сторінки з детальної інформацією представлений на рисунку 2.6.

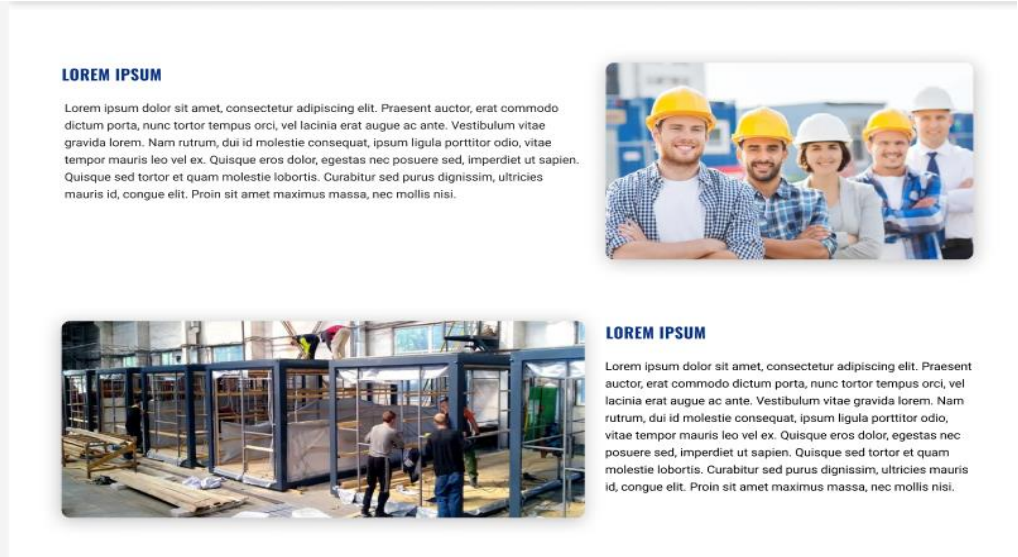


Рисунок 2.6 – Макет сторінки про компанію.

Далі змоделюємо сторінку каталогу будинків. Реалізовано у вигляді карток із зображеннями та детальною інформацією з можливістю перейти у сторінку конкретного будинку. Макет сторінки корзини зображений на рисунку 2.7.

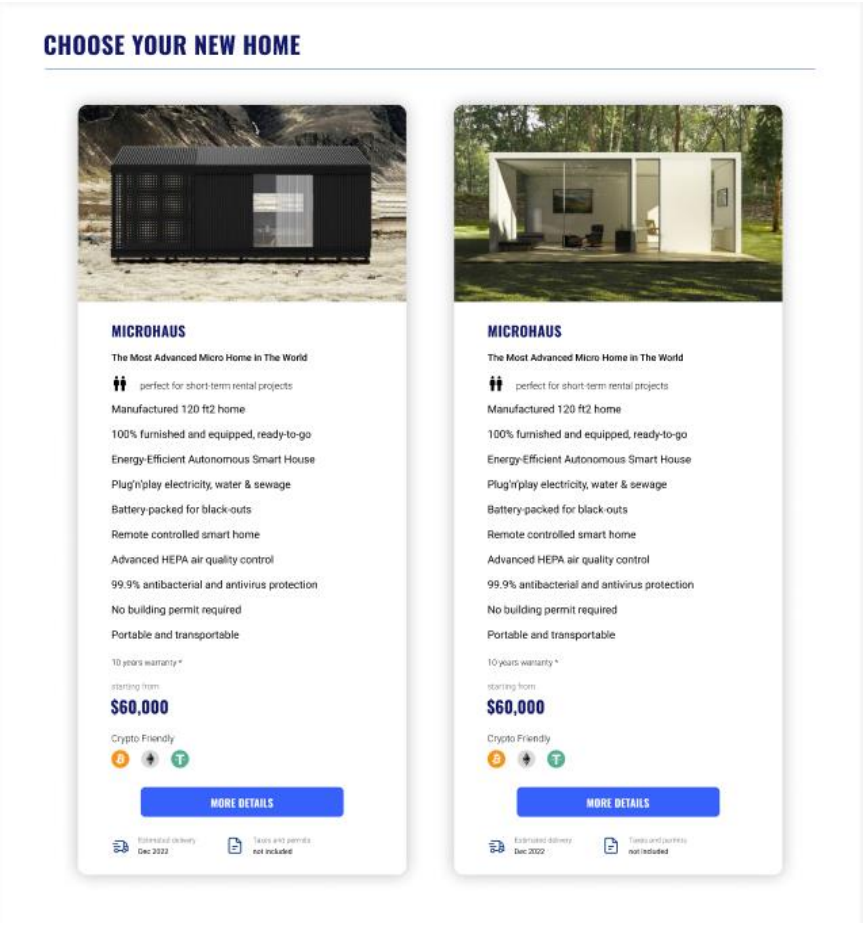


Рисунок 2.7 – Макет каталогу будинків.

Наступними проведено моделювання макетів сторінок для перегляду будинку і галереї. Також сторінка має містити селектори атрибутів - готових комплектацій будинків та короткі описи з можливістю повного читання інформаційних блоків про даний будинок та можливістю при потребі забронювати даний будинок. Макет сторінки будинку зображений на рисунку 2.8.

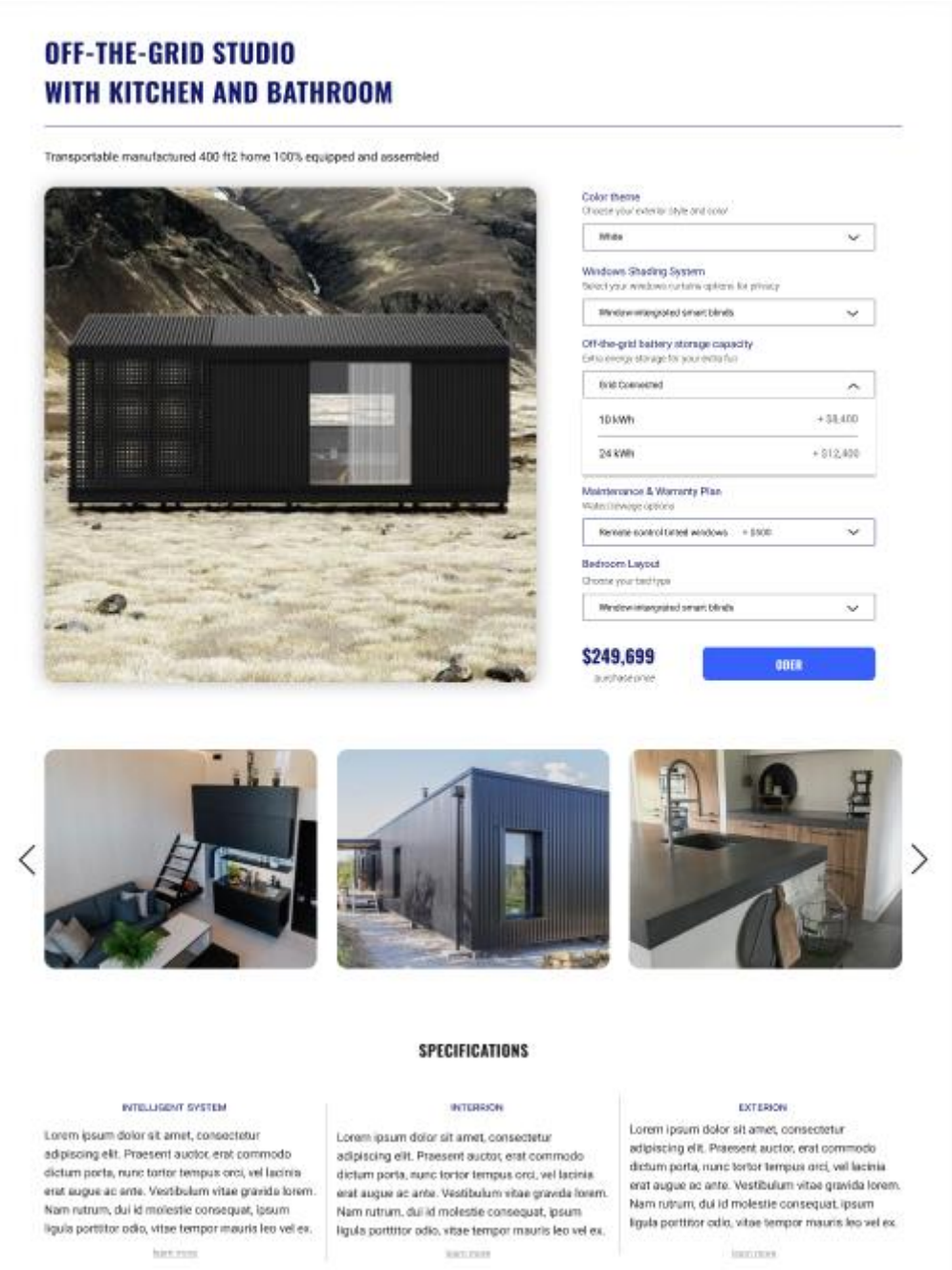


Рисунок 2.8 – Макет сторінки будинку.

Далі змоделюємо сторінку часто задаваних питань у вигляді акордеонів. Макет сторінки корзини зображений на рисунку 2.9.

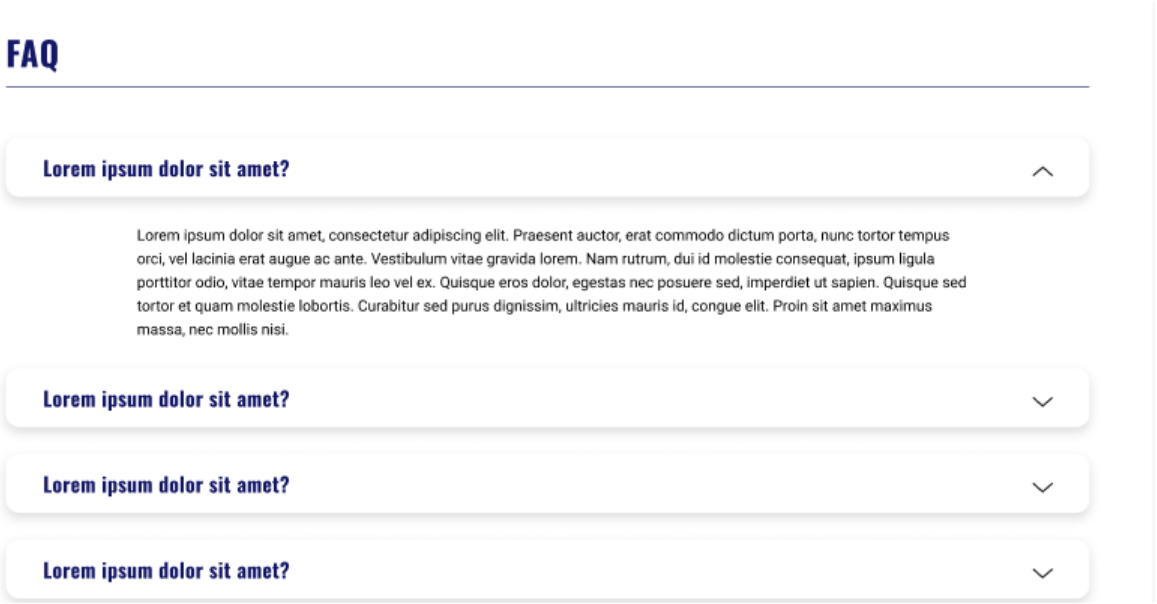


Рисунок 2.9 – Макет сторінки часто задаваниз питань.

Останніми спроектуємо екрани адмін панелі адміністраторів сайту, у якій буде міститись іфнормація про бронювання, зворотній зв'язок та дані про будинки. Макети сторінок зображені на рисунку 2.10-2.11.

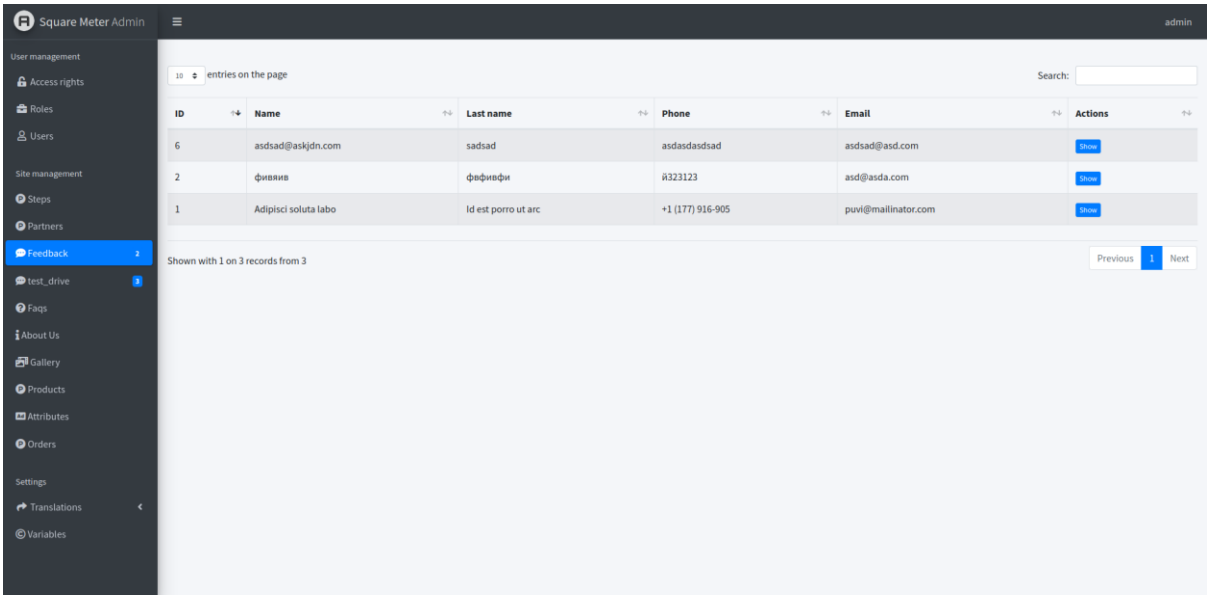


Рисунок 2.10 – Макет вигляду адмін панелі.

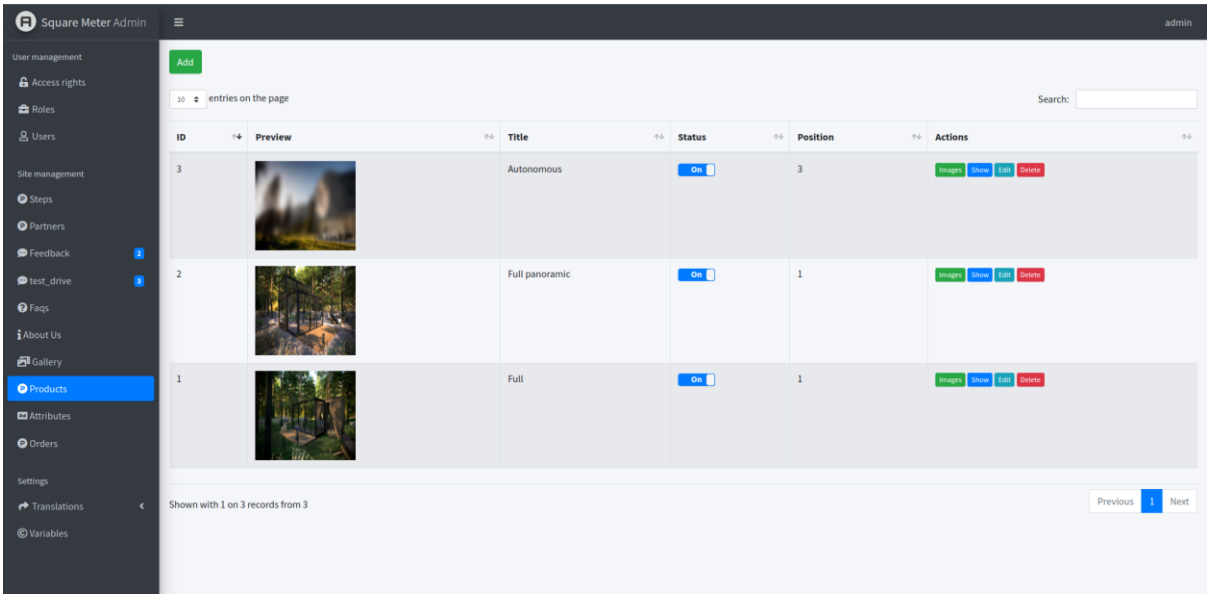


Рисунок 2.11 – Макет редагування будинків.

2.5 Аналіз та вибір технологій і методів реалізації веб-застосунка

У сучасній розробці веб-застосунків необхідно використовувати сучасні методи програмування. Однією з основних мов програмування для розробки є об'єктно-орієнтована JavaScript, яка має найбільшу популярність у всьому світі. JavaScript-код широко застосовується на різних платформах з відкритим вихідним кодом. Веб-сайти будуються з використанням різних технологій, таких як фреймворки, бібліотеки, плагіни, патерни та інші. Такі комбінації технологій часто називаються стеками. Для розробки кваліфікаційної роботи обрано стек Vue.js + Laravel, з використанням сучасних CSS препроцесорів SASS та HTML5.

Vue.js – це прогресивний фронтенд фреймворк, який дозволяє розробникам будувати інтерактивні інтерфейси користувача. Він використовує парадигму компонентної розробки, що дозволяє розбити веб-застосунок на незалежні компоненти, які можуть бути повторно використані.

Використання Vue.js в нашому проекті дозволило нам забезпечити чистоту коду та зручність управління станом застосунку. За допомогою Vue.js ми змогли побудувати компонентну структуру нашого застосунку, розділивши

його на окремі компоненти, такі як заголовок, бічне меню, форми тощо. Це дозволило нам забезпечити модульність і легкість розширення функціональності застосунку.

HTML є стандартною мовою розмітки для створення структури веб-сторінок. Ми використали HTML для визначення структури нашого застосунку, таких елементів як заголовки, кнопки, форми та таблиці. HTML надає нам можливість створювати семантичну розмітку, що полегшує розуміння коду і покращує доступність застосунку.

SASS (Syntactically Awesome Style Sheets) – це препроцесор CSS, який дозволяє використовувати покращений синтаксис для написання стилів. Використання SASS дало нам можливість створювати більш структурований і повторно використовуваний CSS-код. Ми використали SASS для стилізації компонентів та забезпечення їх консистентності в усьому застосунку.

Vuex є становим керувальним засобом для Vue.js застосунків. Він дозволяє ефективно управляти станом застосунку та забезпечує односторонню потоку даних. Vuex використовує концепцію централізованого сховища даних (store), в якому зберігаються всі дані застосунку.

Ми використали Vuex для керування станом застосунку і обміну даними між компонентами. Це дозволило нам ефективно синхронізувати дані між компонентами, що сприяло покращенню продуктивності та забезпеченню консистентності даних у всьому застосунку.

Laravel – це популярний PHP фреймворк, який забезпечує швидку та елегантну розробку веб-застосунків. Laravel надає набір інструментів для роботи з базами даних, маршрутизації, аутентифікації, кешування та багатьох інших функцій.

Ми використали Laravel для реалізації бекенду нашого веб-застосунку. Фреймворк надав зручні інструменти для створення API, обробки запитів від фронтенду та взаємодії з базою даних. Laravel також має велику спільноту розробників та багато готових рішень, що дозволяє прискорити процес розробки та забезпечити надійність застосунку.

Для зменшення затрат часу також було обрано готовий шаблон адмін-панелі з відкритим кодом та доступний за MIT ліцензією.

Отже - Vue.js та Laravel – це два потужних і популярних інструменти, які можуть працювати разом для створення сучасних веб-застосунків. Ось деякі переваги використання Vue.js разом з Laravel:

− простота і зручність розробки: Vue.js має чистий та легко зрозумілий синтаксис, що спрощує розробку фронтенду. Його компонентна структура дозволяє розбити додаток на незалежні компоненти, що полегшує розуміння та підтримку коду. Laravel, з свого боку, надає зручні інструменти для розробки бекенду, що дозволяє швидко створювати потужні API та обробляти запити від фронтенду;

− відмінна реактивність: Vue.js має унікальну можливість реактивної оновлення веб-сторінки. Завдяки використанню властивостей та директив Vue.js, зміни в даних автоматично відображаються на сторінці без необхідності оновлення її повністю. Це забезпечує швидку та зручну роботу з даними на фронтенді;

− продуктивність та швидкодія: Vue.js має легкий та ефективний віртуальний DOM (Document Object Model), що забезпечує високу продуктивність та швидкодію застосунку. Це особливо важливо для веб-застосунків з багатою динамічною взаємодією. Laravel також використовує ефективні методи оптимізації та кешування, що сприяє швидкості та ефективності бекенду;

− розширюваність: Комбінація Vue.js та Laravel дозволяє розширювати функціональність застосунку легко та ефективно. Інтеграція зовнішніх бібліотек та компонентів Vue.js у Laravel дозволяє швидко додавати нові функції та можливості до застосунку. Крім того, Laravel надає багато вбудованих функціональних можливостей, таких як автентифікація, маршрутизація, кешування, що полегшує розширення та розвиток застосунку;

− активна спільнота та ресурси: Як Vue.js, так і Laravel мають великі та активні спільноти розробників. Це означає, що ви можете знайти безліч

ресурсів, документацію, плагіни та готові рішення для вирішення різних завдань. Розробники з усього світу активно співпрацюють та допомагають один одному, що сприяє швидкому розвитку та покращенню застосунків.

Використання Vue.js разом з Laravel надає потужні та ефективні засоби для розробки як фронтенду, так і бекенду веб- застосунку. Це дозволяє забезпечити швидку розробку, високу продуктивність та розширюваність застосунку, а також використовувати широкий спектр ресурсів та підтримку спільноти для полегшення роботи розробників.

Висновки до розділу

У цьому розділі було здійснено проектування системи та вибрано оптимальний варіант архітектури, що відповідає предметній області.

На підставі вимог та поставлених задач, був розроблений інтерфейс користувача. Структура бази даних була спроектована, а також була розроблена архітектура системи та її компоненти для подальшої розробки модулів, таких як:

- Система наповнення веб-застосунку контентом та створення каталогу будівель.
- Модулі зворотнього зв’язку з відділом продажів та бронювання будинків користвуачами веб-застосунку.
- Система управління заявками на бронювання та зв’язку із відділом продажів.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Розробка бази даних

Для створення бази даних використовуючи MySQL, знадобиться доступ до MySQL-сервера та встановлений MySQL-клієнт. Ось кроки, які потрібно виконати для створення бази даних:

− запустити MySQL-клієнт або відкрити командний рядок;

− підключитись до MySQL-сервера за допомогою команди mysql -u ваше_ім'я_користувача -p. Вас буде запрошено ввести пароль для користувача;

− після успішного підключення можна виконувати команди MySQL;

− для створення нової бази даних потрібно виконати наступну команду: CREATE DATABASE ім'я_бази_даних;

Наприклад, якщо потрібно створити базу даних з назвою "mydatabase", команда буде виглядати так: CREATE DATABASE mydatabase;

− після виконання команди база даних буде створена. Можна перевірити це, виконавши команду SHOW DATABASES;. Після чого ми побачимо список доступних баз даних, включаючи ту, яку ми створили.

Для створення таблиць зі зв'язками в базі даних за допомогою MySQL знадобиться використати команди SQL. Створюються таблиці за допомогою команди CREATE TABLE. Ось приклад створення таблиці з двома стовпцями:

```
CREATE TABLE apartment ( id INT PRIMARY KEY, name VARCHAR(50) );
```

У цьому прикладі створюється таблиця "apartment" зі стовпцями "id" типу INT (ціле число) та "name" типу VARCHAR(50) (рядок довжиною до 50 символів). При необхідності можна додати інші стовпці, використовуючи відповідні типи даних.

Якщо потрібно встановити зв'язки між таблицями, потрібно використовувати зовнішні ключі. Наприклад, треба створити зв'язок між таблицями "apartment" і "attribute", в якому один атрибут може належати до

однієї будівлі, але будівля може мати багато атрибутів, можна виконати таку команду:

```
CREATE TABLE apartment ( id INT PRIMARY KEY, name VARCHAR(50) ); CREATE TABLE attribute ( id INT PRIMARY KEY, name VARCHAR(50), apartment_id INT, FOREIGN KEY (apartment_id) REFERENCES apartment(id) );
```

У цьому прикладі створюються дві таблиці - "apartment" і "attribute". У таблиці "attribute" створюється стовпець "apartment_id", який використовується як зовнішній ключ, посилаючись на стовпець "id" таблиці "apartment". Це створює зв'язок між двома таблицями.

Це лише основні кроки створення таблиць зі зв'язками в базі даних за допомогою MySQL. Для більш складних сценаріїв існує багато інших можливостей та конструкцій SQL, які можуть бути використані для оптимізації та налаштування бази даних.

3.2 Розробка програмних модулів

Веб-застосунок складається з двох частин - серверної та клієнтської. Розпочнемо із серверної частини та розглянемо структуру проекту на рисунку 3.1.

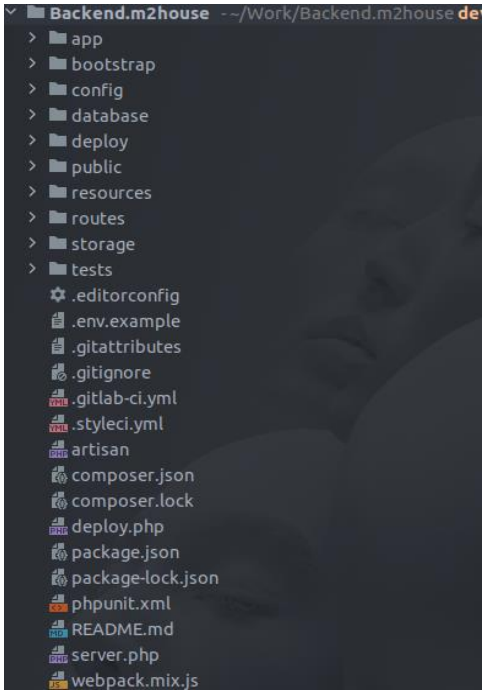


Рисунок 3.1 – Структура серверної частини проекту.

Директорія застосунку містить основний код вашого застосунку. Майже всі класи веб-застосунку будуть розташовані саме тут.

Директорія запуску містить файл app.php, який ініціалізує фреймворк. У цій директорії також знаходиться директорія кешу, яка містить згенеровані фреймворком файли для оптимізації продуктивності, такі як файли кешу маршрутів та сервісів. Зазвичай не потрібно змінювати жодних файлів у цій директорії.

Директорія конфігурації, як і сама назва вказує, містить всі файли конфігурації веб-застосунку.

Директорія бази даних містить міграції бази даних, фабрики моделей та початкові дані (seeds).

Директорія публікацій містить файл index.php, який є точкою входу для всіх запитів, що надходять до вашого застосунку, а також налаштовує автозавантаження. У цій директорії також знаходяться ресурси, такі як зображення, JavaScript і CSS.

Директорія ресурсів містить ваші представлення (views), а також нескомпільовані активи, такі як CSS або JavaScript.

Директорія маршрутів містить всі визначення маршрутів вашого застосунку. За замовчуванням, Laravel включає декілька файлів маршрутів: web.php, api.php, console.php і channels.php.

Файл web.php містить маршрути, які RouteServiceProvider розміщує в групі проміжного програмного забезпечення (middleware) web, яка забезпечує стан сесії, захист від CSRF і шифрування файлів cookie.

Якщо додаток не пропонує безстатевого, RESTful API, то, ймовірно, всі ваші маршрути будуть визначені у файлі web.php.

Файл api.php містить маршрути, які RouteServiceProvider розміщує в групі проміжного програмного забезпечення api. Ці маршрути призначені для безстатевих запитів, тому запити, що надходять до застосунку через ці маршрути, мають бути автентифіковані за допомогою токенів і не матимуть доступу до сесійного стану.

Приклад маршрутів із api.php:

```
Route::get('steps', [StepController::class, 'index']);
Route::get('partners', [PartnerController::class, 'index']);
Route::post('feedback', [FeedbackController::class, 'store']);
Route::get('faqs', [FaqController::class, 'index']);
Route::get('about_us', [AboutUsController::class, 'index']);
Route::get('translations', [TranslationController::class, 'index']);
Route::get('gallery', [GalleryController::class, 'index']);

Route::group(['namespace' => 'Ajax', 'prefix' => 'ajax'], function () {
  Route::get('attribute-values', [AttributeValueController::class, 'index']);
});

Route::get('products', [ProductController::class, 'index']);
Route::get('products/{products:slug}', [ProductController::class, 'show']);
Route::post('products/{product:slug}/images', [ProductController::class, 'images']);

Route::group(['prefix' => 'order/cart'], function () {
  Route::get('/', [OrderController::class, 'cart']);
  Route::post('products/{product_id}', [OrderController::class, 'add']);
  Route::post('submit', [OrderController::class, 'makeOrder']);
});
```

Файл console.php дозволяє визначати команди консолі на основі замикань (closures). Кожне замикання пов'язується з екземпляром команди, що дозволяє

простим способом взаємодіяти з методами введення-виведення (IO) кожної команди. Незважаючи на те, що цей файл не визначає маршрути HTTP, він визначає вхідні точки (маршрути) на основі консолі в застосунку.

Файл channels.php дозволяє реєструвати всі канали трансляції подій, які підтримує додаток.

Директорія зберігання містить ваші журнали, скомпільовані шаблони Blade, сесії на основі файлів, файлові кеші та інші файли, згенеровані фреймворком. Ця директорія розділена на директорії app, framework і logs. Директорія app може використовуватись для зберігання будь-яких файлів, згенерованих вашим додатком. Директорія framework використовується для зберігання файлів, згенерованих фреймворком і кешів. Нарешті, директорія logs містить файли журналу вашого застосунку.

Директорія storage/app/public може використовуватись для зберігання файлів, створених користувачами, таких як аватари профілю, які повинні бути доступні публічно. Можна створити символічне посилання на цю директорію, яке вказує на public/storage. Це посилання можна створити за допомогою команди Artisan php artisan storage:link.

Директорія тестів містить ваші автоматизовані тести. Laravel надає приклади тестів PHPUnit для модульних тестів і функціональних тестів. Кожен клас тесту повинен закінчуватись словом Test. Можна запускати тести за допомогою команд phpunit або php vendor/bin/phpunit. Або, якщо ж є бажання отримати більш детальні й красиві результати тестування, можна запускати тести за допомогою команди Artisan php artisan test.

Далі перейдемо до клієнтської частини застосунку, структура проекту представлена на рисунку 3.2.

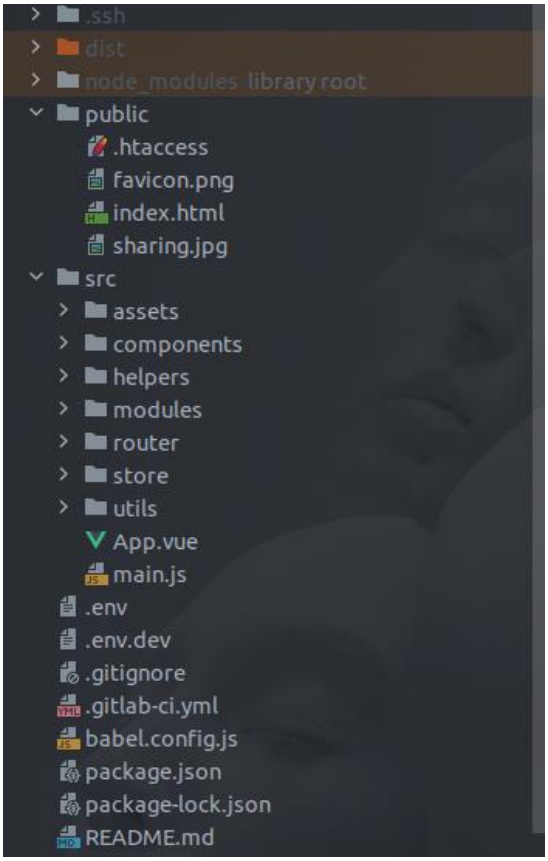


Рисунок 3.2 – Структура клієнтської частини проекту.

Папка public містить статичні файли які може використовувати додаток у початковій точці входу, такі як favicon, sharing зображення для мета даних та .htaccess для того щоб на сервері сайт сам пересилав клієнта на ssl та відкривав сайт на безпечному протоколі https, та основним файлом є index.html у середину якого динамічно генерується контент сайту в залежності від сторінки.

У папці srs знаходяться усі необхідні проекту директорії та файли, розпочнемо із папки assets, вона містить усі необхідні сайту файли такі як: шрифти, зображення, та глобальні стилі проекту, наприклад обгортки сайту, контейнери, стилі для скидування стандартних стилів браузера, анімації і так далі.

Наступною розглянемо папку components, вона містить усі атомарні компоненти які можуть використовуватись на різних сторінках сайту, для того щоб не дублювати код та оптимізувати швидкість та вагу сайту вони зберігаються у цій папці, звідки підключаються одним тегом який має назву

папки потрібного компоненту. Усі компоненти необхідно підключати у модулі за допомогою команди import, але для занадто часто використовуваних компонентів можна зробити вийнятки та ініціалізувати їх глобально у файлі main.js – після того вони стають глобально доступними у всьому проекті без попереднього імпорту.

Далі розглянемо папку modules – у ній зберігаються модулі проекту (сторінки), кожна сторінка це папка у якій набивається верстка, скрипти та за необхідності підключаються атомарні компоненти.

У папці helpers знаходяться необхідні змінні, наприклад переклади сайту у JSON форматі, якщо вони статичні.

У папці router знаходиться маршрутизація сторінок веб сайту, де вказується внутрішня назва модуля, шлях до необхідного модуля, та зовнішній url шлях для користувача, приклад маршрутизації веб-застосунку:

```
import Vue from 'vue'
import VueRouter from 'vue-router'

Vue.use(VueRouter);

const routes = [
 {
  path: '/:lang?',
  name: 'home',
  component: () => import('../modules/home/index.vue')
 },
 {
  path: '/:lang?/projects',
  name: 'projects',
  component: () => import('../modules/projects/index.vue')
 },
 {
  path: '/:lang?/about',
  name: 'about',
  component: () => import('../modules/about/index.vue')
 },
 {
  path: '/:lang?/contact',
  name: 'contact',
  component: () => import('../modules/contact/index.vue')
 },
 {
  path: '/:lang?/faq',
  name: 'faq',
```

```
  component: () => import('../modules/faq/index.vue')
 },
 {
  path: '/:lang?/models',
  name: 'models',
  component: () => import('../modules/models/index.vue')
 },
 {
  path: '/:lang?/models/:slug',
  name: 'product',
  component: () => import('../modules/product/index.vue')
 },
 {
  path: '/:lang?/order',
  name: 'order',
  component: () => import('../modules/order/index.vue')
 },
];

const router = new VueRouter({
 mode: 'history',
 base: process.env.BASE_URL,
 routes,
});

export default router
```

Наступною та самою важливою є папка store, її вміст виконує комунікацію із серверною частиною веб-застосунку, структура зображена на рисунку 3.3.

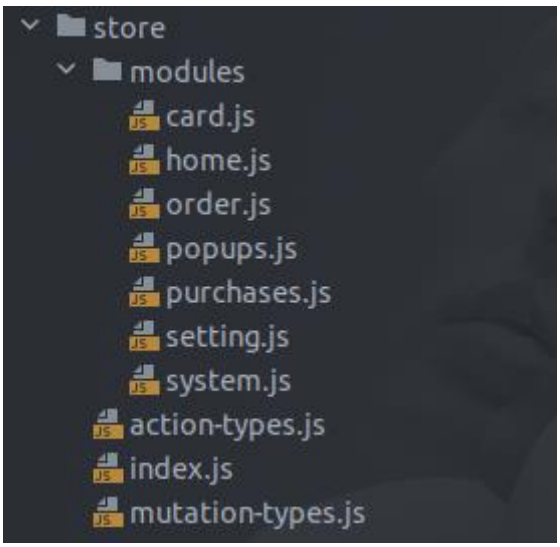


Рисунок 3.3 – Структура папки store.

У файлі index.js формується загальний vuex store веб-застосунку у який підключаються комунікаційні модулі із вкладеної папки modules у яких знаходяться описані state, actions, mutations, getters: state – це місце у якому зберігаються змінні які містять значення отримані із серверу, action - це метод який робить запит на сервер та отримує response – відповідь, у разі помилки запиту обробляє та видає статус та код помилки для подальшого опрацювання, у разі успіху передає отриману відповідь у mutation – засіб для модифікування state, адже він є константним об’єктом а константу можна тільки модифікувати, та getter – засіб через який можна отримати значення змінної у модулі/компоненті проекту для подальшого опрацювання.

Приклад описаного store файлу:

```
const state = {
 steps: null,
 };
const getters = {
 steps: state => state.steps,
 };
const actions = {
 [GET_STEPS]: async ({commit}) => {
   try {
     const response = await $http.get(`v1/steps`);
     commit(SET_STEPS, response.data.data);
   } catch (e) {
     throw e;
   }
 },
}
const mutations = {
 [SET_STEPS](state, list) {
   state.steps = list;
 },
}
export default {
 state,
 getters,
 actions,
 mutations,
};
```

Далі у папці utils знаходяться утиліти проекту для конфігурації http запитів, імпорт плагінів та конфігурація локалізацій.

App.vue та main.js – кореневі vue та js файли які збирають усі модулі та їх скрипти та білдять змонтовані сторінки у файл index.html на сервері.

3.3 Керівництво користувача

При вході на клієнтську частину користувач потрапляє на головну сторінку, де може прочитати інформацію про будинки, залишити заявку у відділ продажів та користуватися навігаційним меню сайту для переходу по сторінкам каталогу, про компанію, галерею та контакти (рисунки 3.4-3.5)

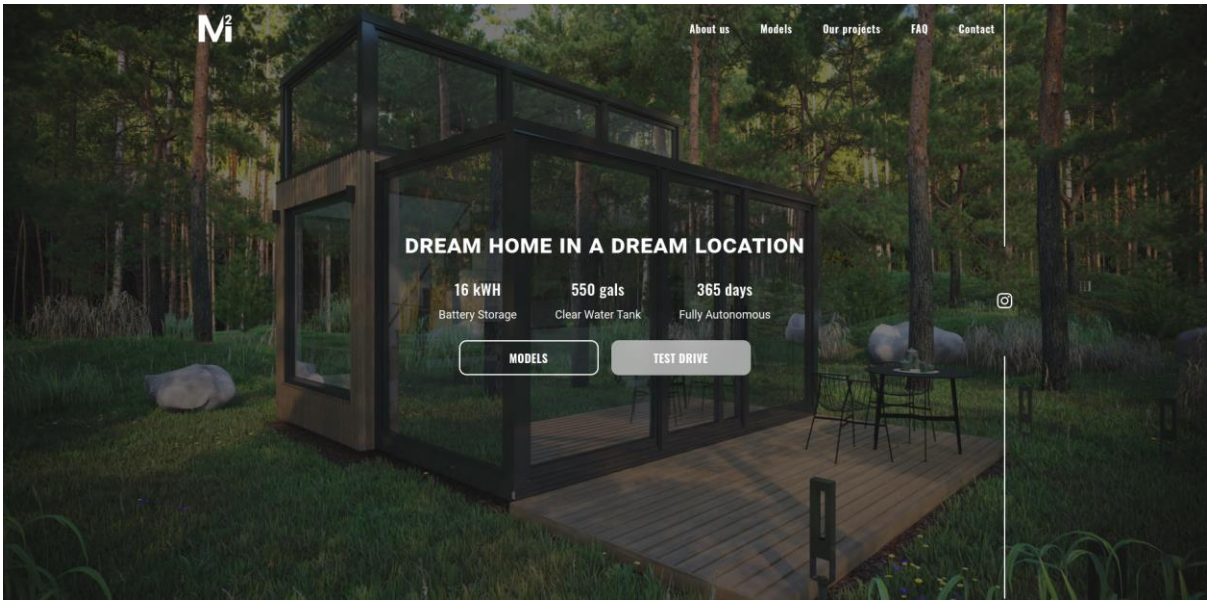


Рисунок 3.4 – Головна сторінка.

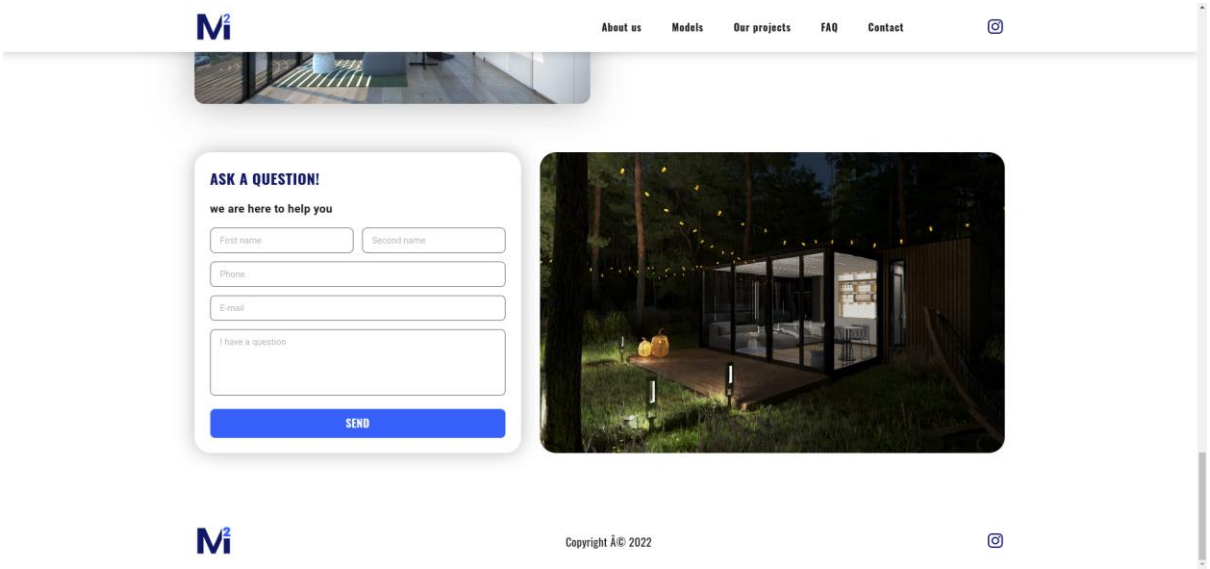


Рисунок 3.5 – Форма зв'язку із відділом продажів.

При переході на сторінку про компанію користувач отримує інформаційні блоки з інформацією представлені на рисунку 3.6.

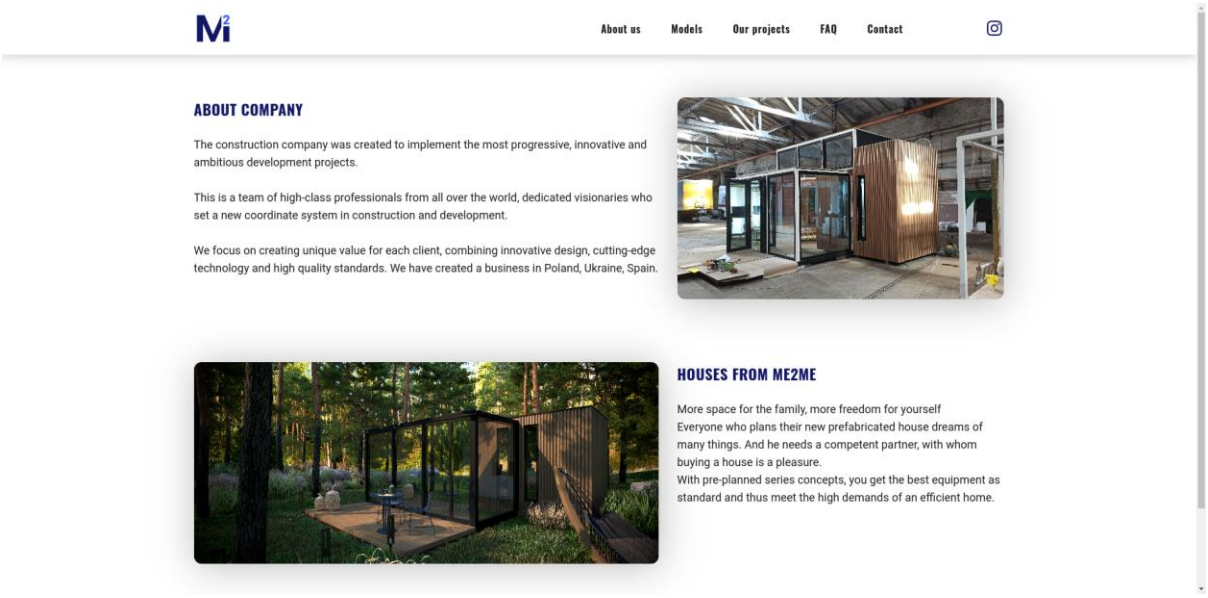


Рисунок 3.6 – Сторінка про компанію.

При переході у сторінку models користувач побачить картки доступних у продажі будинків із фотографіями та необхідною інформацією та можливістю переходу на сторінку конкретного будинку, зображено на рисунку 3.7.

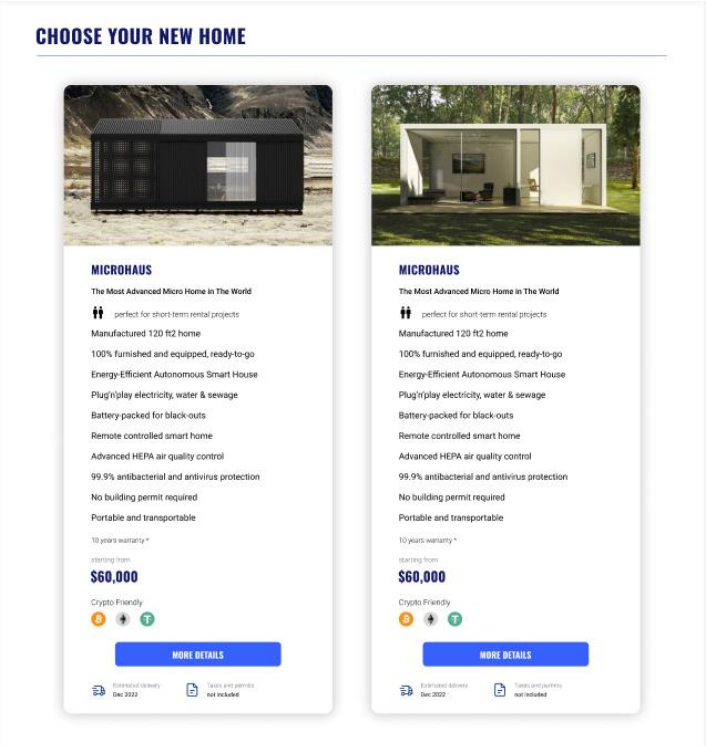


Рисунок 3.7 – Сторінка каталогу будинків.

Після переходу на сторінку будинку користувач бачить галерею будинку, має можливість читати текстові описи, також обирати конфігурацію готового будинку для бронювання, зображено на рисунку 3.8.

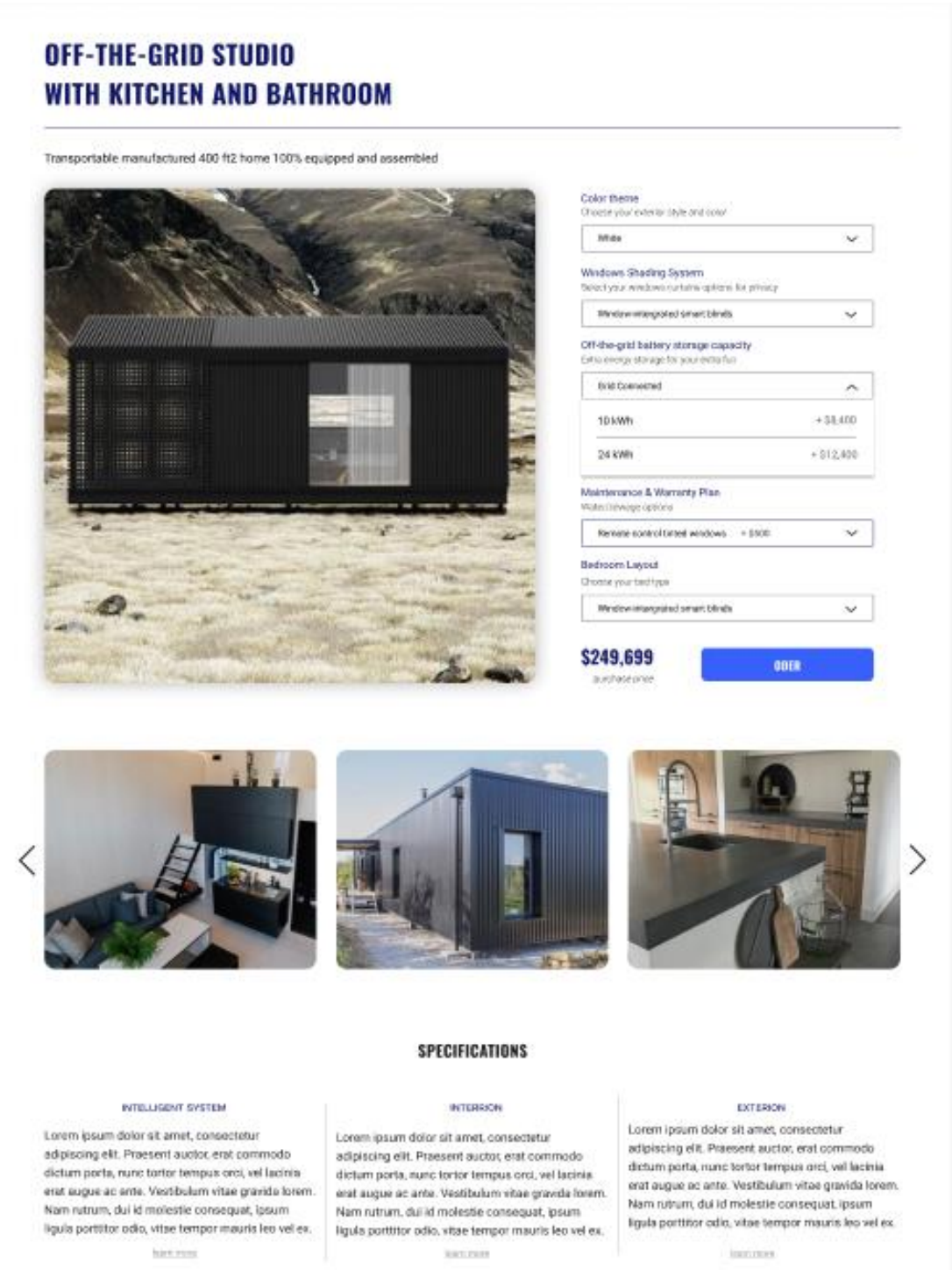


Рисунок 3.8 – Сторінка будинку.

Після вибору конфігурації та переходу на сторінку бронювання користувачу пропонується заповнити контактну форму для подальшого опрацювання клієнта адміністраторами сайту, зображено на рисунку 3.9.

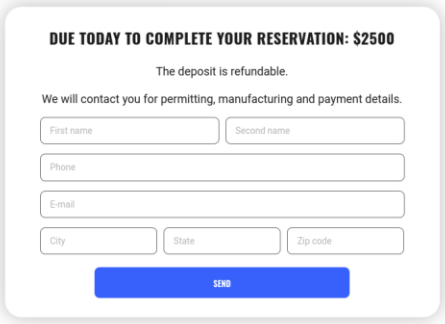


Рисунок 3.9 – Сторінка бронювання.

І на останок користувач може переглянути галерею будови та подивитись часто задавані запитання для мінімізації витрати часу відділу продажів (рисунок 3.10 - 3.11).

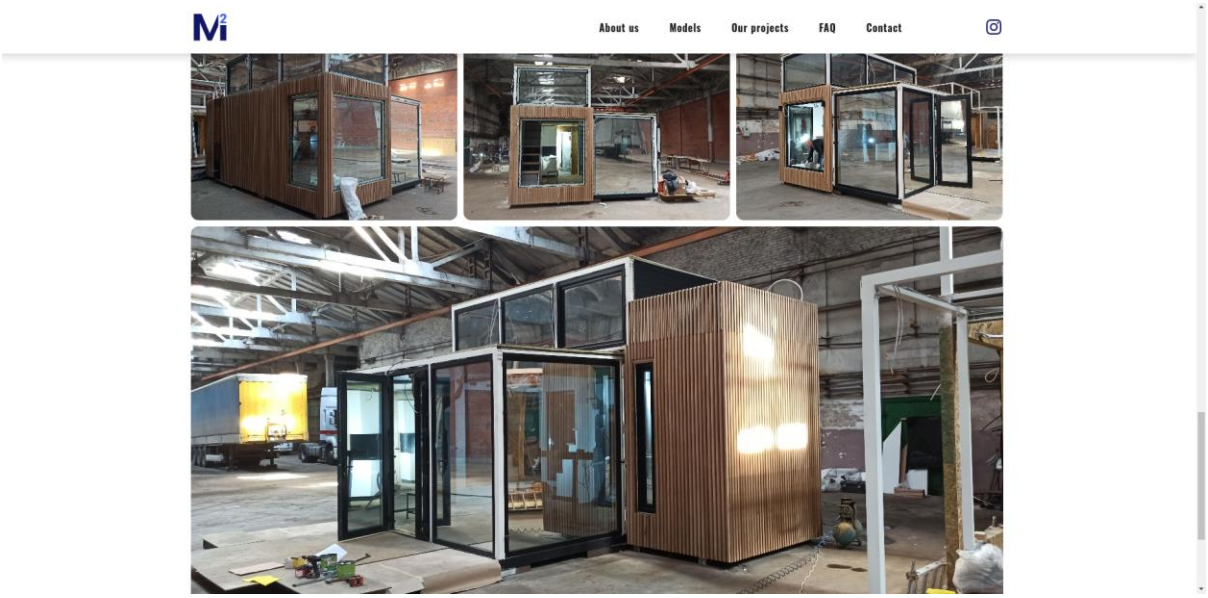


Рисунок 3.10 – Сторінка галереї.

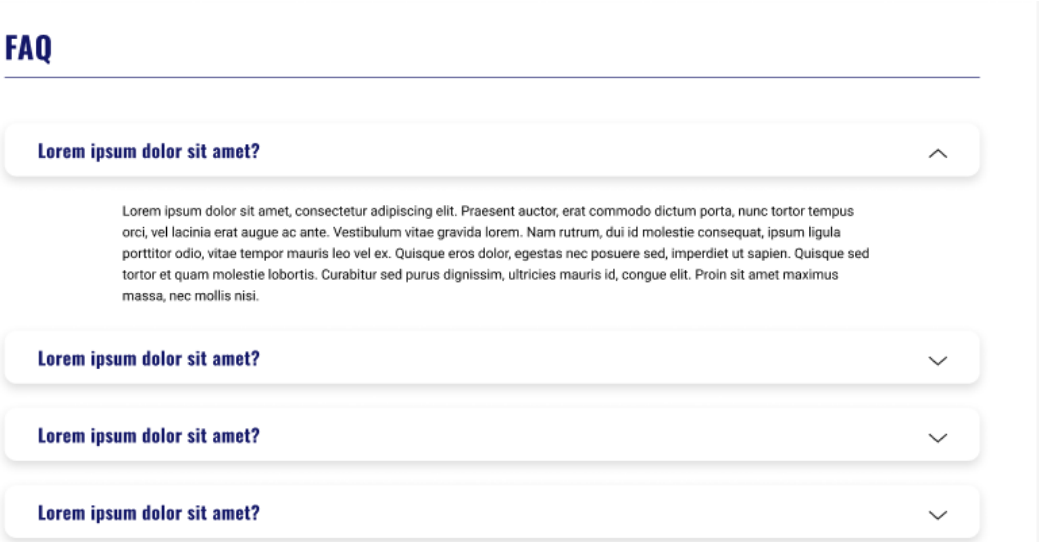


Рисунок 3.11 – Сторінка часто задаваних питань.

При вході у адмін панель адміністратору потрібно авторизуватись, зображено на рисунку 3.12.

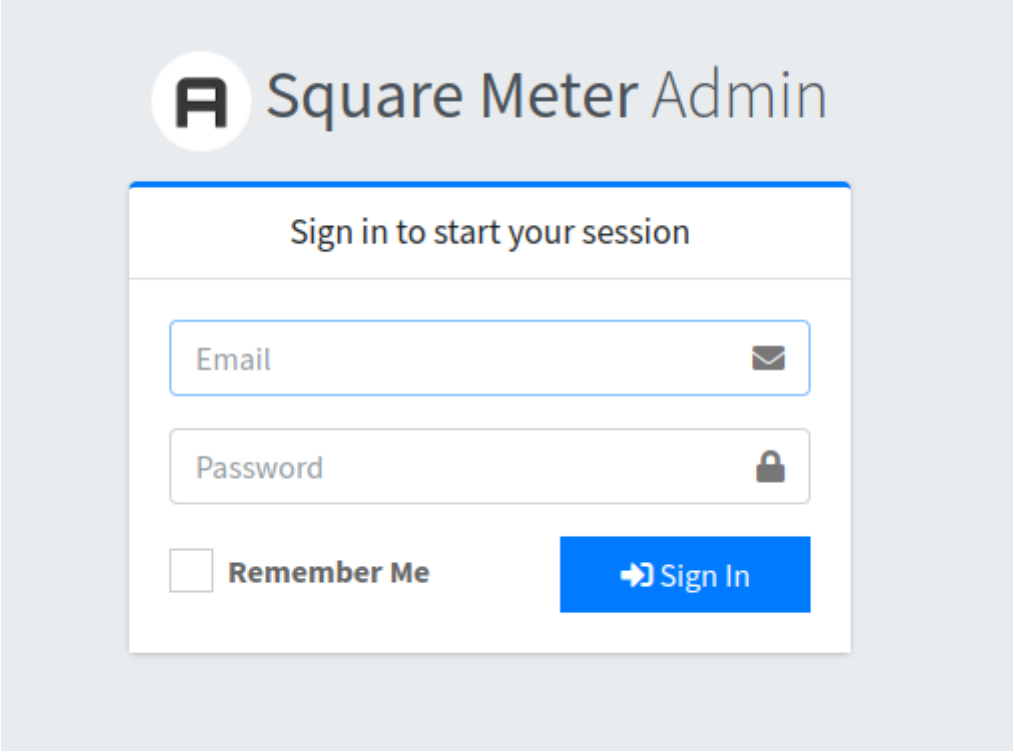


Рисунок 3.12 – Сторінка часто задаваних питань.

У разі успішної авторизації адміністратор потрапляє у панель керування сайтом яка у всіх розділах є шаблонною тому для прикладу приведено список будинків із можливостями створення нового будинку, завантаження зображень,

попереднього перегляду збереженої інформації, редагування вже існуючих даних та видалення будинків, зображено на рисунку 3.13

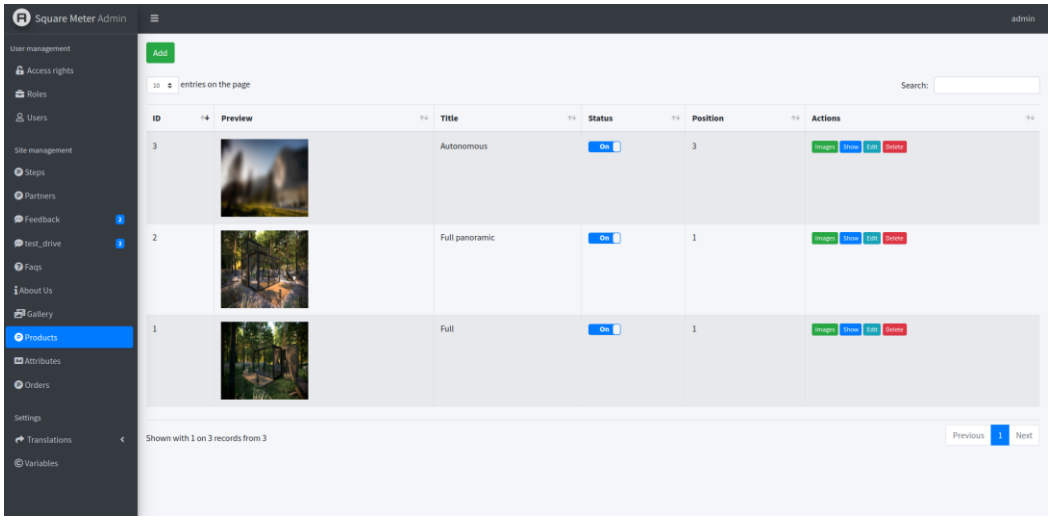


Рисунок 3.13 – Сторінка списку будівель.

При переході на створення або редагування записів відкривається мульти-форма яка є приблизно однаковою у всіх розділах адмін панелі зображена на рисунку 3.14.

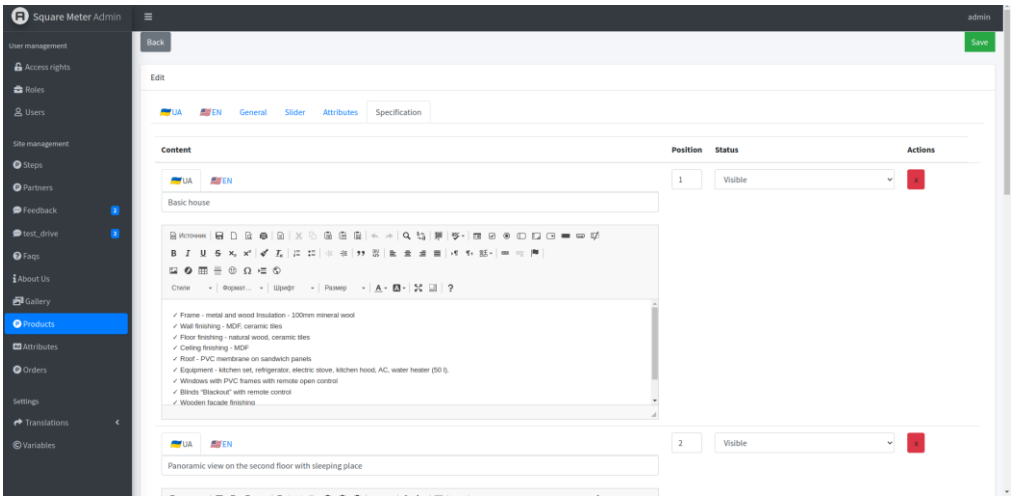


Рисунок 3.14 – Форма створення будівель.

3.4 Технічні характеристики веб-застосунку

Для коректної роботи веб-застосунку, потрібно щоб сервер, на якому він знаходиться задовольняв наступні вимоги:

- 1 Гб внутрішньої пам’яті;
- 2 Гб оперативної пам’яті;
- 4 – ядерний процессор;
- доступ до мережі «Інтернет» із вихідною швидкістю мінімум 20 Мбіт/с.

3.5 Розгортання та встановлення системи

Для розгортання клієнтської та серверної частини веб-застосунку потрібно мати встановлені Node.js, Composer, PHP та MySQL.

Для підготовки до розгортання клієнтської частини потрібно виконати такі команди:

- cd project_folder;
- npm install.

Після створення папки node modules проект готовий до запуску, використовується команда npm run serve, для сворення dist папки для подальшого завантаження на сервер використовується команда npm run build.

Для підготовки до розгортання серверної частини потрібно виконати наступне:

- cd project_folder;
- npm install;
- composer install;
- створити пусту базу даних;
- прописати у файлі .env назву створеної бази;
- запустити команду php artisan migrate для створення таблиць та зв'язків;
- php artisan key:generate;
- php artisan storage:link;

Після виконаних дій потрібно відкрити одночасно два термінали і для локального запуску необхідно в одному запустити команду php artisan serve для запуску локального сервера та у другому npm run watch для запуску вотчера для перегляду веб сайту з можливыстю hot - reload.

Для переносу на сервер на самому сервері потрібно встановити composer та node_modules, після чого локально виконати команду npm run production та перенести всі локальні файли на сервер, після чого у .env підключити серверну базу даних.

У даному розділі було описано процес створення бази даних та його підключення до веб-застосунку. Також проведено декомпозицію системи на модулі та розглянуто детально кожен з них. Були визначені технічні характеристики веб-застосунку. Окрім цього, розглянуто алгоритм інсталяції та експлуатації розробленого програмного продукту.

3.6 Тестування веб-застосунку

Тестування веб-застосунків є важливою складовою розробки програмного продукту, оскільки воно допомагає переконатися в якості та коректності функціоналу застосунку перед його впровадженням. Основна мета тестування полягає в виявленні помилок, неправильних робочих процесів та інших проблем, які можуть виникнути під час використання веб-застосунку.

Існує кілька видів тестування веб-застосунків:

− модульне тестування: перевіряє окремі модулі або функції програмного коду на правильність роботи. Це дозволяє виявити проблеми в окремих частинах коду та впевнитися, що вони працюють належним чином;

− інтеграційне тестування: перевіряє взаємодію між різними компонентами веб-застосунку, щоб переконатися, що вони працюють разом правильно та передають дані між собою коректно;

− функціональне тестування: перевіряє, чи виконує веб-застосунок очікувані функції та завдання. Це включає тестування різних сценаріїв

використання, введення та виведення даних, обробки помилок та інших аспектів, пов'язаних з функціональністю;

− візуальне тестування: перевіряє зовнішній вигляд та коректність відображення веб-сторінок. Використовуються інструменти, які автоматично перевіряють, чи відображаються елементи на сторінці правильно, чи зберігаються стилі та компонування;

− продуктивнісне тестування: оцінює продуктивність веб-застосунку шляхом вимірювання часу відгуку, швидкості завантаження сторінок та інших параметрів. Це дозволяє виявити можливі проблеми з продуктивністю та вчасно їх виправити.

Також існують інші типи тестування, такі як безпекове тестування (перевірка на вразливості та захист від атак), тестування сумісності (перевірка на різних платформах та браузерах), тестування навантаження (вимірювання реакції системи на великі навантаження) та інші. Вибір конкретних типів тестування залежить від потреб та вимог проекту.

Проведемо модульне тестування контроллерів передачі данних на клієнтську частину за допомогою Unit test для перевірки коректності відпрацювання функцій.

```
use PHPUnit\Framework\TestCase;
use App\Models\Product;
use App\Http\Resources\ProductResource;

class ProductControllerTest extends TestCase
{
   public function testShowReturnsProductResourceIfStatusIsTrue()
   {
      // Arrange
      $product = new Product();
      $product->status = true;
      $product->sliders = ['slider1', 'slider2'];
      $product->specifications = ['spec1', 'spec2'];

      $controller = new ProductController();

      // Act
      $result = $controller->show($product);
```

```
        // Assert
        $this->assertInstanceOf(ProductResource::class, $result);
        $this->assertEquals($product->sliders, $result->sliders);
        $this->assertEquals($product->specifications, $result->specifications);
    }

    public function testShowReturnsErrorNotFoundIfStatusIsFalse()
    {
        // Arrange
        $product = new Product();
        $product->status = false;

        $controller = new ProductController();

        // Act
        $result = $controller->show($product);

        // Assert
        $this->assertEquals('Not Found', $result);
    }
}
```

У цьому прикладі ми створюємо два тести. Перший тест перевіряє, що якщо значення status об'єкта Product дорівнює true, то функція show() повертає екземпляр класу ProductResource з відповідними значеннями sliders та specifications. Другий тест перевіряє, що якщо значення status дорівнює false, то функція show() повертає повідомлення про помилку "Not Found".

Цей Unit тест допоможе переконатися, що функція show() поводиться правильно в різних сценаріях та повертає очікувані результати, продемонстровано на рисунку 3.15.

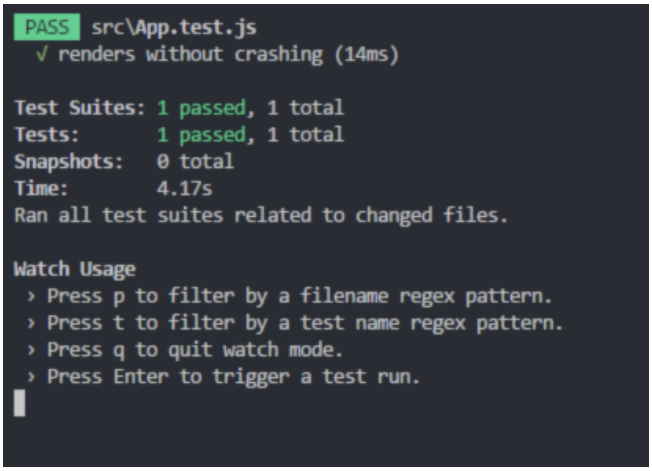


Рисунок 3.15 – Результати тестування.

Також обов’язковим є тестування функції збереження бронювання із всіма обраними атрибутами, адже будь яка помилка може призвести до фінансових втрат компанії.

```
use Illuminate\Foundation\Testing\DatabaseTransactions;
use Tests\TestCase;
use App\Models\Product;
use App\Http\Controllers\YourController; // Замініть на ваш контролер

class YourControllerTest extends TestCase
{
    use DatabaseTransactions;

    public function testAddProductToCart()
    {
        // Створення тестових даних
        $product = Product::factory()->create();
        $request = [
            'attribute_values' => [1, 2], // Припустимі значення атрибутів
            'count' => 2,
        ];

        // Виклик функції для тестування
        $controller = new YourController(); // Замініть на ваш контролер
        $response = $controller->add($request, $product->id);

        // Перевірка результату
        $this->assertNotNull($response);
        $this->assertInstanceOf(OrderResource::class, $response);
        // Додаткові перевірки, залежно від очікуваного результату

        // Перевірка в базі даних, чи створено відповідний запис
        $this->assertDatabaseHas('order_products', [
            'product_id' => $product->id,
            'count' => $request['count'],
        ]);
    }
}
```

Після тестування можемо переконатись що функція працює справно, рисунок 3.16.

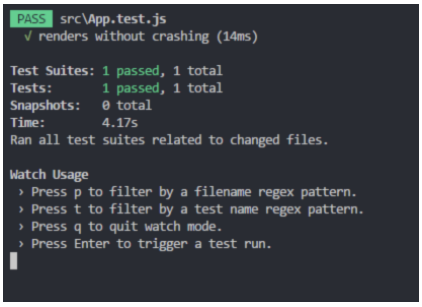


Рисунок 3.16 – Результати тестування збереження броні.

Отже, в даному розділі було розглянуто та описано основні підходи до тестування веб-застосунків.

Результати модульного тестування демонструють, що всі функції працюють так, як очікувалося.

Висновки до розділу

У даному розділі було описано процес створення бази даних та його підключення до веб-застосунку. Також проведено декомпозицію системи на модулі та розглянуто детально кожен з них. Були визначені технічні характеристики веб-застосунку. Окрім цього, розглянуто алгоритм інсталяції та експлуатації розробленого програмного продукту. Після проведення тестування можна зробити висновок, що програмний засіб успішно пройшов всі необхідні тести, відповідає вимогам, викладеним у технічному завданні, і готовий до розгортання та використання.

На підставі вимог та поставлених задач, був розроблений веб за стосунок, який містить модулі, такі як:

– Система наповнення веб-застосунку контентом та створення каталогу будівель.

– Модулі зворотнього зв’язку з відділом продажів та бронювання будинків користвуачами веб-застосунку.

– Система управління заявками на бронювання та зв’язку із відділом продажів.

ВИСНОВКИ

Під час розробки кваліфікаційної роботи було проведено аналіз сфери застосування, доведена актуальність даної розробки та виявлені можливі проблеми, з якими стикаються люди, які не використовують веб-застосунки у сфері будівництва. Також було вивчено наявні рішення в цій галузі. В результаті аналізу подібних програмних засобів було складено список їх переваг та недоліків, а також визначено функціональні та нефункціональні вимоги до майбутнього програмного продукту. На основі цих вимог було сформульовано технічне завдання та розглянуто варіанти використання застосунку.

Наступним кроком було приступлено до проектування веб-застосунку. На даному етапі було проведено аналіз переваг і недоліків поширених патернів архітектур, що зазвичай використовуються для розробки веб-застосунків. В результаті було визначено, що найбільш підходящою для цієї програмної системи є клієнт-серверна архітектура з використанням патерну MVC.

Після цього була розроблена архітектура бази даних, визначено поля, таблиці та зв'язки між ними. Враховуючи спроектовану архітектуру веб-застосунку та бази даних, було прийнято рішення про вибір технологій і інструментів, які найкраще відповідають цьому типу архітектури. Зокрема, були обрані фреймворки Vue.js, Laravel, препроцесор SASS, та реляційна база даних MySQL. Одним з ключових факторів при виборі була швидкість та надійність працездатності, яку забезпечують ці технології.

Наступним етапом роботи було розбиття програмної системи на модулі та проектування взаємодії між ними. Крім того, було проведено проектування інтерфейсу для веб-застосунку.

На основі розробленої архітектури та інтерфейсу було створено базу даних і реалізовано серверну та клієнтську частини системи, а також механізм взаємодії між ними.

Останнім етапом було проведення тестування, під час якого виявлено і виправлено всі несправності щодо функціональності та зовнішнього вигляду.

Були здійснені модульні та функціональні тести, і звіт був складений на основі результатів тестових випробувань. Проведені тести підтвердили, що система виконує всі функції, описані у технічному завданні, та має зручний для користувача інтерфейс, який не містить зайвих елементів.

В результаті роботи було розроблено веб-застосунок який задовольняє поставлені задачі:

- Розроблено простий та приваблий інтерфейс користувача для зручного користування веб-застосунком.
- Розроблено систему наповнення веб-застосунку контентом та створення каталогу будівель.
- Забезпечено можливість зворотнього зв’язку з відділом продажів та бронювання будинків користвуачами веб-застосунку.
- Розроблено систему управління заявками на бронювання та зв’язку із відділом продажів.

Таким чином, поставлене завдання на кваліфікаційній роботі було повністю виконано. Розроблений веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника працює бездоганно та ефективно. Ця робота дозволила набути практичних та теоретичних навичок у всіх етапах розробки програмного забезпечення та закріпити отримані знання.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Дипломний проект: методичні вказівки щодо його виконання для студентів спеціальності 121 «Інженерія програмного забезпечення» / Бедратюк Л. П., Радельчук Г. І., Форкун Ю. В., Яшина О. М. Хмельницький : ХНУ, 2020. 77с .
2. Vue.js – URL: https://vuejs.org/ . (дата звернення – 14.04.2023).
3. Vuex – URL: https://vuex.vuejs.org/ . (дата звернення – 01.04.2023).
4. TompsonLaura. Creating SPAWEB-Applications – London.: DiaSoft. 2016. – 672 с.(дата звернення – 28.03.2023).
5. Lindstorn, Steve. CSS Refactoring – New York: O'Relly Media, 2016. – 159с. . (дата звернення – 02.04.2023).
6. React – URL: https://uk.reactjs.org/ . (дата звернення – 10.04.2023).
7. Node.js чи Java – URL: https://medium.com/webbdev/js-9ca13173577b/ . (дата звернення – 12.04.2023).
8. Сервер. Створення сервера – URL: https://metanit.com/web/nodejs/3.1.php/ . (дата звернення – 17.04.2023).
9. F7 – URL: https://f7-lublin.pl/. (дата звернення – 19. 04.2023).
10. Me 2 Me – URL: https://me-2.me/. (дата звернення – 18.04.2023).
11. AVILA Building group – URL: https://avila.com.ua/ . (дата звернення – 18.03.2023).
12. КомфортБудПлюс – URL: https://comfortbuildplus.com.ua/ . (дата звернення – 19.04.2023).
13. IDM Group – URL: https://idmgroup.com.ua/ . (дата звернення – 22.04.2023).
14. Модульне тестування / QaLight. – URL: https://qalight.ua/baza-znaniy/modulne-testuvannya/ . (дата звернення – 08.05.2023).
15. Уніфікована мова програмування UML / Портал знань. – URL: http://www.znannya.org/?view=uml . (дата звернення – 11.05.2023).

16. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE – URL: https://www.guru99.com/unit-testing-guide.html (дата звернення – 11.05.2023).

17. NOSQL – переваги та недоліки нереляційних баз даних / QAinfo. – URL: https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).

18. Vue Router – маршрутизація веб-застосунку – URL: https://router.vuejs.org/ . (дата звернення – 05.05.2023).

19. Lodash – джаваскрипт утиліти – URL: https:/lodash.org/ . (дата звернення – 05.05.2023).

20. MySQL – серверна реляційна база даних – URL: https:/mysql.com/ . (дата звернення – 06.05.2023).

21. Laravel – php фреймворк – URL: https:/laravel.com/ . (дата звернення – 04.05.2023).

22. GitHub – хмарне кодове середовище – URL: https://github.com/ . (дата звернення – 07.05.2023).

23. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE – URL: https://www.guru99.com/unit-testing-guide.html (дата звернення – 11.05.2023).

24. NOSQL – переваги та недоліки нереляційних баз даних / QAinfo. – URL: https://www.quality-assurance-group.com/nosql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).

25. W3SHOOLS – html документація – URL: https://w3shools.com/ . (дата звернення – 04.05.2023).

26. SemRush – сео-оптимізація веб-застосунку – URL: https://www.semrush.com/blog/javascript-seo/ . (дата звернення – 08.05.2023).

27. Lindstorn, Steve. CSS Refactoring – New York: O'Relly Media, 2016. – 159с. . (дата звернення – 02.04.2023).

28. Figma – документація прототипування – URL: https://help.figma.com/ . (дата звернення – 04.05.2023).

29. GitHub – deploy документація – URL: https://docs.github.com/en/actions/deployment/ . (дата звернення – 10.05.2023).

30. Unit Testing Tutorial: What is, Types, Tools & Test EXAMPLE – URL: https://www.guru99.com/unit-testing-guide.html (дата звернення – 11.05.2023).

31. Сервер. Створення сервера – URL: https://metanit.com/web/nodejs/3.1.php/ . (дата звернення – 17.04.2023).

32. Vuex – архітектура проекту – URL: https://www.vuex.vuejs.org/guide/structure.html/ . (дата звернення – 04.05.2023).

33. Vuex – документація про геттери – URL: https://www.vuex.vuejs.org/guide/getters.html/. (дата звернення – 04.05.2023).

34. Vuex – документація про мутації – URL: https://www.vuex.vuejs.org/guide/mutations.html/. (дата звернення – 04.05.2023).

35. Vuex – документація про виклики – URL: https://www.vuex.vuejs.org/guide/actions.html/. (дата звернення – 04.05.2023).

36. GitHub – deploy документація – URL: https://docs.github.com/en/actions/deployment/ . (дата звернення – 10.05.2023).

37. SQL – переваги баз даних . – URL: https://www.quality-assurance-group.com/sql-perevagy-ta-nedoliky-nerelyatsijnyh-baz-danyh/ . (дата звернення – 04.05.2023).

38. PHP – php документація – URL: https://www.php.net/ . (дата звернення – 07.05.2023).

39. SASS – css препроцесор – URL: https://www.sass-lang.com/ . (дата звернення – 04.05.2023).

40. Webpack – документація проектного збірника – URL: https://www.webpack.js.org/ . (дата звернення – 04.05.2023).

ДОДАТОК А
(обов’язковий)

ТЕХНІЧНЕ ЗАВДАННЯ

Введення

Робота виконується в рамках проекту розробки веб застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника. Технічне завдання розроблено у відповідності до стандарту ГОСТ 19.201–78.

1 Підстава для розробки

Підставою для розробки є «Завдання на кваліфікаційну роботу», затверджене завідувачем кафедри інженерії програмного забезпечення. Найменування розробки: Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника.

2 Призначення розробки

2.1 Функціональне призначення

Функціональним призначенням застосунку є клієнтська частина для перегляду інформації про будинки та їх бронювання, та серверна частина з можливістю редагування вмісту сайту.

2.2 Експлуатаційне призначення

Програма повинна експлуатуватися на будь-яких пристроях, на яких є браузер. Кінцевим користувачем застосунку може виступати будь-яка особа.

3 Вимоги до програми

3.1 Вимоги до функціональних характеристик

Для звичайного користувача:

- перегляд інформаційних блоків про компанію;
- перегляд портфоліо компанії;
- перегляд галерей будівництва;
- окремі сторінки під різні будівлі з їх наявними плануваннями та статусами будови;
- можливість звернутись онлайн до відділу продажів;
- можливість переглянути юридичні документи компанії;
- можливість конфігурації готового інтер’єру квартири;
- можливість бронювання покупки квартири;
- можливість перегляду веб-сайту на різних мовах;

Для адміністратора:

- авторизація;
- створення ролей адміністраторів з відповідними доступами до редагування вмісту сайту;
- можливість створення та редагування блоків про компанію;
- можливість додавання об’єктів у портфоліо компанії;
- можливість завантаження фото/відео у галереї на веб-сайті;
- можливість створення та редагування окремих сторінок будівель з усією необхідною інформацією;
- можливість оновлювати статуси будови;
- можливість завантаження планувань будови;
- можливість опрацювання усіх вхідних онлайн-звернень з веб-сайту;
- можливість загрузки та оновлення необхідних юридичних документів;
- можливість створення модулів конфігурацій готових квартир;
- система обліку усіх вхідних заявок на бронювання;
- можливість редагування усього контенту сайту на різних мовах.

3.2 Вимоги до надійності

Веб-застосунок повинен забезпечувати такі вимоги до надійності:

- обробляти невірні дії користувача і попереджати його про можливі наслідки;
- можливість самостійно відновлюватись у разі збою;
- можливість резервного копіювання бази даних.

3.3 Умови експлуатації та вимоги до технічних засобів

Веб-застосунок повинен працювати на всіх пристроях, які мають браузер та стабільний доступ до мережі «Інтернет»: смартфонах, планшетах та комп’ютерах. Браузер може бути будь-яким: Safari, GoogleChrome, Opera, MozillaFirefox, тощо.

Для роботи платформи без збоїв, потрібно, щоб пристрій, на якому вона запускається задовольняв наступні мінімальні вимоги:

- 1 гб внутрішньої пам’яті;
- 2 Гб оперативної пам’яті;
- 4-ядерний процесор;
- доступ до мережі «Інтернет» зі швидкістю мінімум 20 Мбіт/с.

3.4 Вимоги до інформаційної та програмної сумісності

Для розробки веб-застосунку був використаний фронтенд JavaScript фреймворк Vue.js, бекенд PHP фреймворк Laravel, та база даних SQL.

3.5 Спеціальні вимоги

Програма повинна мати зручний та зрозумілий зовнішній інтерфейс користувача.

4 Вимоги до програмної документації

У момент здачі проекту замовнику надається наступний набір документів:

- текст програми;
- опис програми;
- технічне завдання;
- керівництво користувача.

5 Стадії та етапи розробки

Стадії та етапи розробки веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

Стадія розробки	Етапи робіт	Зміст робіт
1	2	3
Технічне завдання 02.01.23 – 31.01.23	Обґрунтування необхідності розробки програми	Коротка характеристика програмного забезпечення; підстава і призначення розробки; вимоги до програмної системи і документація; стадії і етапи розробки програми; порядок контролю і приймання
Ескізний проект 01.02.23 – 26.02.23	Розробка ескізного проекту	Попередня розробка структури вхідних і вихідних даних; уточнення середовища програмування; розробка і опис загальної алгоритмічної структури

Кінець таблиці А.1

1	2	3
Технічний проект 29.02.23 – 19.03.23	Розробка технічного проекту	Уточнення структури вхідних і вихідних даних; розробка докладного алгоритму; розробка структури програми
Робочий проект 20.03.23 – 15.04.23	Розробка програмного забезпечення	Реалізація програмного забезпечення; відлагодження; проведення попереднього тестування
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням
Тестування системи 23.04.23 – 30.04.23	Проведення тестування програмного забезпечення	Розробка методики тестування; проведення основних тестів; коректування програмного забезпечення
Впровадження	Підготовка і передача програми	Підготовка і розгортання програмного забезпечення
Розробка програмної документації 16.04.23 – 22.04.23	Розробка документації до програмного забезпечення	Розробка необхідної документації, передбаченої технічним завданням

6 Порядок контролю та приймання

Контроль здійснюється кінцевими користувачами системи, підключеними на етапі тестування застосунку.

ДОДАТОК Б
(обов’язковий)

ДІАГРАМИ

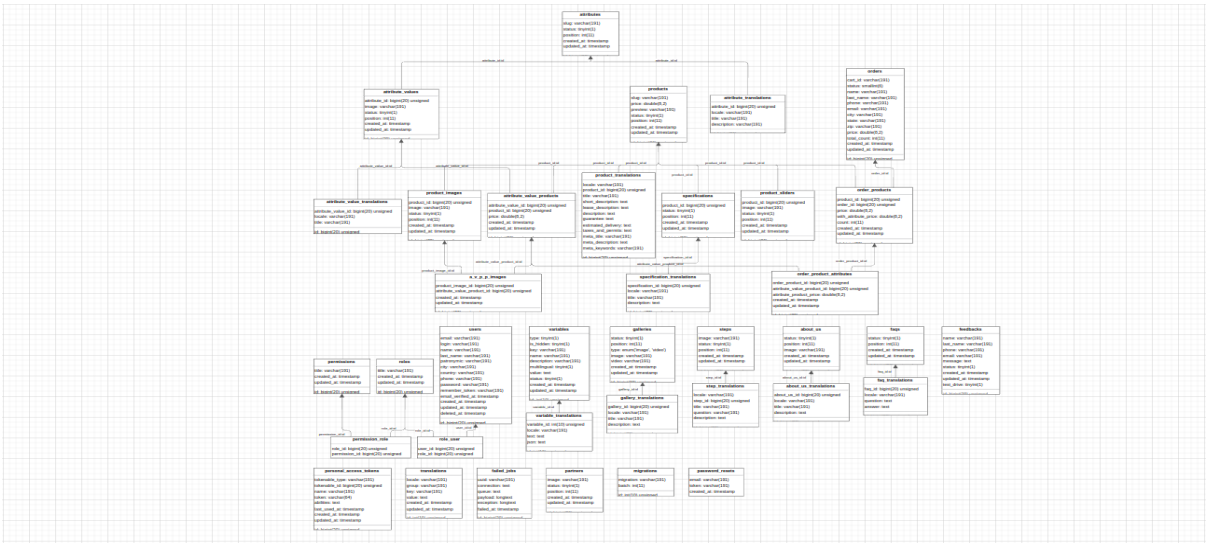


Рисунок Б.1 – Схема бази даних

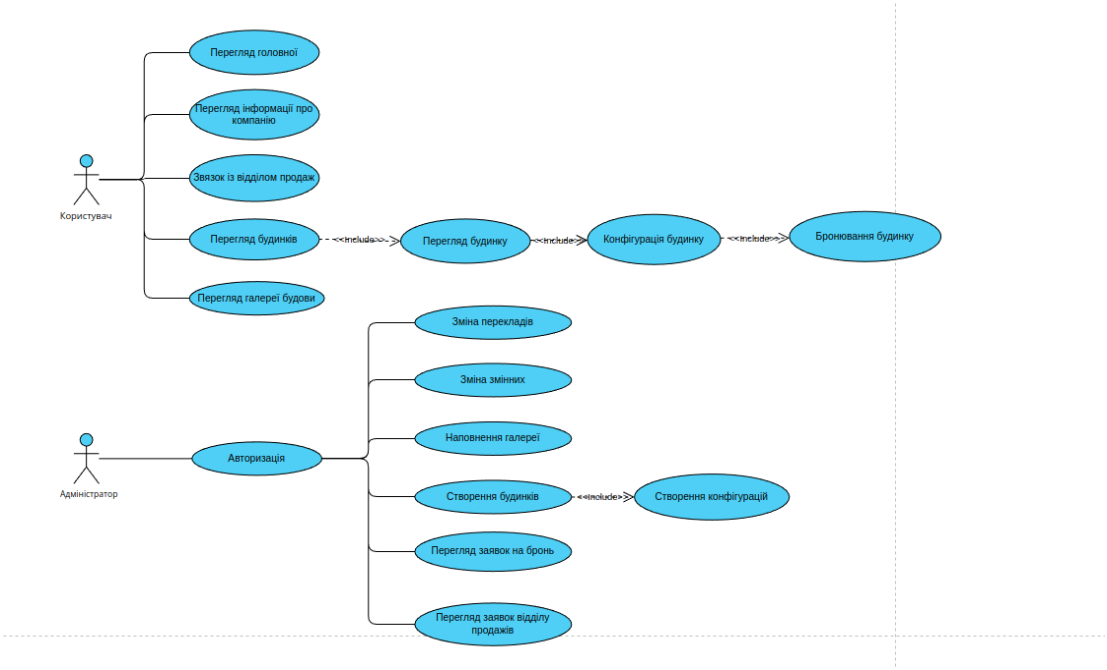


Рисунок Б.2 – Діаграма варіантів використання

ДОДАТОК В
(обов’язковий)

КОД (ЛІСТИНГ) ПРОГРАМИ

App.vue

```
<template>
 <div id="app" class="app">
   <div class="app-container">
     <div class="wrapper">
       <div class="wrapper-top">
         <main-header/>
         <transition name="component-fade" mode="out-in">
           <router-view/>
         </transition>
       </div>
       <div class="wrapper-bottom">
         <main-footer/>
       </div>
     </div>
   </div>
   <hidden/>
   <transition name="component-fade" mode="out-in">
     <div class="global-loader" v-show="globalLoader">
       <div class="Loader"></div>
     </div>
   </transition>
 </div>
</template>

<style src="@/assets/scss/main.scss" lang="scss"></style>

<script>
import {mapGetters, mapActions, mapMutations} from 'vuex';
import Hidden from '@/components/hidden/index.vue';
import MainHeader from '@/components/header/index.vue';
import MainFooter from '@/components/footer/index.vue';
import {isMobile, isIPad} from '@/helpers/variables/index'
import { nanoid } from 'nanoid'

export default {
 name: 'app',
 components: {
   Hidden,
   MainHeader,
   MainFooter
 },
 computed: {
   ...mapGetters({
     isAuthenticated: `auth/isAuthenticated`,
     variables: `setting/variables`,
     globalLoader: 'system/globalLoader',
   })
 },
 created() {
   this.fetchVariables()
   localStorage.getItem('cart_id') ? '' : localStorage.setItem('cart_id',  nanoid())
 },
 mounted() {
   if (!isMobile.any()) {
     document.querySelector('body').classList.add('hover');
   }
   if (!isIPad.iOS()) {
     document.querySelector('body').classList.add('iPad');
   }
 },
 methods: {
   ...mapMutations({
   }),
   ...mapActions({
     fetchVariables: 'setting/GET_VARIABLES'
   }),
```

```
 }
}
</script>
```

main.js

```
//utils
import './utils/plugins'
import './utils/translations'
import './utils/global-components'
//
import moment from 'moment';

import Vue from 'vue'
import App from './App.vue'
import router from './router'
import store from './store'
import VueHead from 'vue-head'
// moment
Vue.prototype.moment = moment


Vue.use(VueHead)
//vue configs
Vue.config.productionTip = false;

//set token after refresh page
if (localStorage.getItem('user_token')) {
 store.commit(`auth/SET_TOKEN`, localStorage.getItem('user_token'));
}

const app = new Vue({
 router,
 store,
 render: h => h(App),
 mounted() {
   document.dispatchEvent(new Event('render-event'))
 }
});

Vue.initI18nManager().then(() => {
 app.$mount('#app')
});
```

house.vue

```
<template>
 <div class="product">
   <div class="container">
     <div class="product-wrap" v-if="product">
       <h1 data-aos="fade-up" data-aos-duration="500">{{product.title}}</h1>
       <div data-aos="fade-up" data-aos-duration="700" class="line"/>
       <span data-aos="fade-up" data-aos-duration="700" class="short-desc">
         {{ product.shortDescription }}
       </span>
       <div class="product-wrapper">
         <div class="product-image" data-aos-offset="200" data-aos="fade-up-right" data-aos-duration="800">
           <img :src="(productImages && productImages.images.length) ? productImages.images[0] : product.preview"
:alt="product.title">
         </div>
         <div class="product-desc" data-aos-offset="200" data-aos="fade-up-left" data-aos-duration="800">
           <div class="product-desc-item" v-for="item in product.attributes.data" :key="item.id">
             <attributes @change-value="changeAttribute" :data="item"/>
           </div>
           <div class="product-order">
             <div class="product-order-price">
```

```
            <span>${{(productImages && productImages.price) ? productImages.price : product.price}}</span>
            <p>Purchase price</p>
          </div>
          <div class="product-order-btn">
            <main-button :progress="cartLoading" @change-event="order" :label="'order'"/>
          </div>
        </div>
      </div>
    </div>
    <div class="product-slider-wrap" data-aos="fade-up" data-aos-duration="700" v-if="product.sliders.data.length">
      <div class="product-slider-prev" v-if="product.sliders.data.length > 3" @click="carouselNav($event, 'backward')">
        <svg width="27" height="48" viewBox="0 0 27 48" fill="none" xmlns="http://www.w3.org/2000/svg">
          <path d="M25.1893 2L3 24.3284L25.1893 46" stroke="#1A1A32" stroke-width="3"/>
        </svg>
      </div>
      <div class="product-slider-next" v-if="product.sliders.data.length > 3" @click="carouselNav($event, 'forward')">
        <svg width="27" height="48" viewBox="0 0 27 48" fill="none" xmlns="http://www.w3.org/2000/svg">
          <path d="M1.81071 2L24 24.3284L1.81071 46" stroke="#1A1A32" stroke-width="3"/>
        </svg>
      </div>
      <carousel :per-page-custom=[[200,1],[680,2],[1100,3]] :loop="true"
                :scrollPerPage="false"
                :autoplay="false"
                :speed="1000"
                :autoplayHoverPause="false"
                :mouse-drag="true"
                :easing="'ease-in-out'"
                ref="carousel" data-ref="carousel" :paginationEnabled="false">
        <slide v-for="slide in product.sliders.data" :key="slide.id">
          <div class="product-slider-slide">
            <div class="img-wrap">
              <img :src="slide.image" :alt="product.title">
            </div>
          </div>
        </slide>
      </carousel>
    </div>
    <div class="product-specifications" data-aos="fade-up" data-aos-duration="700" v-if="product.specifications &&
product.specifications.data.length">
      <h2>Specifications</h2>
      <div class="product-specifications-wrap">
        <div class="product-specifications-item-wrap" v-for="item in product.specifications.data" :key="item.id">
          <div class="product-specifications-item">
            <h3>{{item.title}}</h3>
            <div class="product-specifications-item-desc" v-html="item.description"/>
            <a v-if="item.description.length > 80" @click="toggleSpecification(true, item)">learn more</a>
          </div>
        </div>
      </div>
    </div>
  </div>
  <div class="product-wrap" style="align-items:center;" v-else>
    <small-loader/>
  </div>
  <transition name="component-fade" mode="out-in">
    <specification @close="toggleSpecification(false, null)" v-if="selectedSpecification" :data="selectedSpecification"/>
  </transition>
 </div>
</div>
</template>
<style src="./index.scss" lang="scss"></style>
<script src="./index.js"></script>
```

house.js

```
import {mapActions, mapGetters, mapMutations} from "vuex";
import AOS from 'aos';
import 'aos/dist/aos.css';
import { Carousel, Slide } from 'vue-carousel';
//sections
```

```
import attributes from "@/modules/product/attributes/index.vue";
import specification from "@/modules/product/specification/index.vue";

export default {
 name: "product",
 components: {
  attributes,
  Carousel,
  Slide,
  specification
 },
 data() {
  return {
   selectedSpecification: null,
  }
 },
 watch: {
  '$route.params.slug'(newVal) {
   if (newVal) {
    this.initCart()
    this.resetProduct()
    this.fetchProduct(this.$route.params.slug).then(()=>{
     if(this.product.attributes.data.length){
      let queryAttributes = this.product.attributes.data.reduce((acc, el) => {
       acc[el.slug] = el.values.data[0].id
       return acc
      }, {})
      this.$router.push({query: Object.assign({}, this.$route.query, queryAttributes)}).catch(() => {
       console.log()
      });
     }
    }).catch(()=>{
     console.log()
    })
   }
  },
  '$route.query': {
   immediate: true,
   handler(newVal) {
    if(Object.keys(newVal).length){
     this.fetchProductImages({slug: this.$route.params.slug, payload:this.$route.query})
    }
   }
  },

 },
 created() {
  this.initCart()
  this.resetProduct()
  this.fetchProduct(this.$route.params.slug).then(()=>{
   if(this.product.attributes.data.length){
    let queryAttributes = this.product.attributes.data.reduce((acc, el) => {
     acc[el.slug] = el.values.data[0].id
     return acc
    }, {})
    this.$router.push({query: Object.assign({}, this.$route.query, queryAttributes)}).catch(() => {
     console.log()
    });
   }
  }).catch(()=>{
   console.log()
  })
 },
 mounted() {
  AOS.init({
   disable: function () {
    const maxWidth = 700;
    return window.innerWidth <= maxWidth;
   }
  });
 },
```

```
 computed: {
   ...mapGetters({
     product: 'category/product',
     productImages: 'category/productImages',
     cartLoading: 'card/cartLoading'
   }),
 },

 methods: {
   ...mapActions({
     fetchProduct: 'category/GET_PRODUCT_DATA',
     fetchProductImages: 'category/GET_PRODUCT_IMAGES',
     initCart: 'card/GET_CART',
     addOrder: 'card/ADD_ITEM_TO_CART'
   }),
   ...mapMutations({
     resetProduct: 'category/RESET_PRODUCT',
     fixBody: 'system/FIX_BODY'
   }),
   changeAttribute(attribute) {
     this.$router.push({
       query: Object.assign({},
         this.$route.query,
         {[attribute.slug]:attribute.id })}
     ).catch(() => {
       console.log()
     });
   },
   carouselNav($event, direction) {
     const carousel = this.$refs['carousel'];
     carousel.advancePage(direction);
   },
   toggleSpecification(status, data){
     if(status){
       this.fixBody(status)
       this.selectedSpecification = data
     }
     else{
       this.fixBody(status)
       this.selectedSpecification = data
     }
   },
   order(){
     this.addOrder({id: this.product.id, attribute_values: this.$route.query}).then(()=>{
        this.$router.push({name: 'order'}).catch(()=>{
          console.log()
        })
     }).catch(error => {
       this.$toasted.error(error)
     })

   },
 }
}
```

house.scss

```
@import "../../assets/scss/vars";

.product {
 &-wrap {
   padding-top: 160px;
   padding-bottom: 100px;
   width: 100%;
   display: flex;
   align-items: flex-start;
   justify-content: flex-start;
   flex-direction: column;
```

```
  h1 {
    font-family: 'Oswald', sans-serif;
    font-style: normal;
    font-weight: 700;
    font-size: 40px;
    line-height: 59px;
    letter-spacing: 0.02em;
    text-transform: uppercase;
    color: $dark-blue;
    margin-bottom: 24px;
    max-width: 570px;
  }

  .line {
    width: 100%;
    height: 1px;
    background: $dark-blue;
    margin-bottom: 30px;
  }

  .short-desc {
    font-style: normal;
    font-weight: 400;
    font-size: 18px;
    line-height: 28px;
    color: $text;
    margin-bottom: 30px;
  }
}

&-wrapper {
  width: 100%;
  display: flex;
  align-items: flex-start;
  justify-content: space-between;
}

&-image {
  width: 740px;
  height: 745px;
  border-radius: 16px;
  overflow: hidden;
  filter: drop-shadow(0px 0px 30px rgba(0, 0, 0, 0.3));

  img {
    width: 100%;
    height: 100%;
    object-fit: cover;
  }
}

&-desc {
  max-width: calc(100% - 810px);
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: flex-start;
  flex-direction: column;

  &-item {
    width: 100%;
  }

  &-item + &-item {
    margin-top: 18px;
  }
}
```

```
&-order {
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: space-between;
  margin-top: 40px;

  &-price {
    display: flex;
    align-items: center;
    justify-content: center;
    flex-direction: column;

    span {
      font-family: 'Oswald', sans-serif;
      font-style: normal;
      font-weight: 700;
      font-size: 28px;
      line-height: 41px;
      text-align: right;
      font-variant: small-caps;
      color: $dark-blue;
      margin-bottom: 5px;
    }

    p {
      font-style: normal;
      font-weight: 300;
      font-size: 14px;
      line-height: 16px;
      text-transform: lowercase;
      color: $text;
    }
  }

  &-btn {
    max-width: 270px;
    width: 100%;

    a {
      width: 100%;
      text-decoration: none;
      cursor: pointer;
      display: inline-block;
    }
  }
}

&-slider {
  &-wrap {
    margin-left: -15px;
    width: calc(100% + 30px);
    margin-top: 60px;
    position: relative;
  }

  &-next, &-prev {
    cursor: pointer;
    display: inline-block;
    position: absolute;
    bottom: 50%;
    transform: translateY(50%);
    z-index: 3;

    svg {
      width: 22px;
      height: 44px;
```

```
      path {
        stroke: #1A1A32;
        transition: stroke 0.3s ease-in-out;
      }
    }

    &:hover {
      svg {
        path {
          stroke: $blue;
        }
      }
    }
  }

  &-next {
    right: -36px;
  }

  &-prev {
    left: -36px;
  }

  &-slide {
    width: 100%;
    padding-left: 15px;
    padding-right: 15px;

    .img-wrap {
      width: 100%;
      height: 330px;
      border-radius: 16px;
      overflow: hidden;

      img {
        width: 100%;
        height: 100%;
        object-fit: cover;
      }
    }
  }
}

&-specifications {
  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;
  margin-top: 60px;

  h2 {
    font-family: 'Oswald', sans-serif;
    font-style: normal;
    font-weight: 700;
    font-size: 24px;
    line-height: 36px;
    letter-spacing: 0.02em;
    text-transform: uppercase;
    color: $dark-blue;
    margin-bottom: 40px;
  }

  &-wrap {
    display: flex;
    flex-wrap: wrap;
    width: calc(100% + 30px);
    margin-left: -15px;
```

```
}

&-item {
  &-wrap {
    position: relative;
    max-width: 33.333%;
    flex: 0 0 33.333%;
    width: 100%;
    padding-left: 15px;
    padding-right: 15px;
    margin-bottom: 20px;

    &:before {
      position: absolute;
      height: 100%;
      width: 1px;
      background: #B9B9B9;
      left: 0;
      top: 0;
      bottom: 0;
      content: '';
    }
    &:nth-child(3n+1){
      &:before {
        display: none;
      }
    }
  }

  width: 100%;
  display: flex;
  align-items: center;
  justify-content: center;
  flex-direction: column;

  h3 {
    font-style: normal;
    font-weight: 400;
    font-size: 18px;
    line-height: 21px;
    text-align: center;
    font-variant: small-caps;
    color: $dark-blue;
  }

  &-desc {
    margin-bottom: 16px;
    margin-top: 20px;
    font-style: normal;
    font-weight: 400;
    font-size: 18px;
    line-height: 28px;
    color: $text;
    display: -webkit-box;
    -webkit-line-clamp: 6;
    -webkit-box-orient: vertical;
    overflow: hidden;
  }

  a {
    display: inline-block;
    cursor: pointer;
    font-style: normal;
    font-weight: 400;
    font-size: 14px;
    line-height: 16px;
    text-decoration-line: underline;
    color: #8F8F8F;
  }
```

```
  }
 }
}

@media screen and (max-width: 1280px){
 .product{
  &-image{
   width: 600px;
   height: 650px;
  }
  &-slider{
   &-next, &-prev{
    bottom: -100px;
    transform: translateY(0);
   }
   &-prev{
    left: 15px;
   }
   &-next{
    right: 15px;
   }
   &-wrap{
    margin-bottom: 50px;
   }
  }
  &-desc{
   max-width: calc(100% - 650px);
  }
  &-order-btn{
   width: auto;
  }
 }
}
@media screen and (max-width: 1180px){
 .product{
  &-wrap{
   padding-top: 80px;
   padding-bottom: 60px;
  }
 }
}
@media screen and (max-width: 960px){
 .product{
  &-image{
   width: 350px;
   height: 400px;
  }
  &-desc{
   max-width: calc(100% - 400px);
  }
 }
}

@media screen and (max-width: 680px){
 .product{
  &-wrap{
   h1{
    font-size: 24px;
    line-height: 32px;
    margin-bottom: 10px;
   }
   .line{
    margin-bottom: 15px;
   }
   .short-desc{
    font-size: 14px;
    line-height: 18px;
    margin-bottom: 15px;
   }
  }
  &-image{
```

```
    width: 100%;
    border-radius: 0;
    margin-bottom: 20px;
   }
   &-desc{
    max-width: none;
   }
   &-wrapper{
    flex-direction: column;
    align-items: center;
    justify-content: center;
   }
   &-specifications{
    &-wrap{
     margin-left: 0;
     width: 100%;
    }
    &-item{
     &-wrap{
      max-width: 50%;
      flex: 0 0 50%;
      &:nth-child(3n+1){
       &:before {
        display: block;
       }
      }
      &:nth-child(2n-1){
       &:before{
        display: none;
       }
      }
     }
    }
   }
  }
}
@media screen and (max-width: 450px){
 .product{
   &-specifications{
    &-item{
     &-wrap{
      flex: 0 0 100%;
      max-width: 100%;
      &:before{
       display: none;
      }
     }
    }
   }
  }
}
```

productController.php

```
<?php

namespace App\Api\v1\Http\Controllers;

use App\Api\v1\Http\Resources\ProductResource;
use App\Models\AttributeValueProduct;
use App\Models\AVPPImages;
use App\Models\Product;
use App\Models\ProductImage;
use Illuminate\Http\Request;

class ProductController extends BaseController
{
   public function index()
```

```
    {
        $products = Product::with('sliders', 'specifications')->prepared()->get();

        return ProductResource::collection($products);
    }

    public function show(Product $products)
    {
        if (!$products->status) {
            return $this->errorNotFound();
        }

        $products->load('sliders', 'specifications');

        return ProductResource::make($products);
    }

    public function images(Request $request, Product $product)
    {
        $attribute_value = $request->get('attribute_values', []);

        if (empty($attribute_value)) {
            return response()->json([]);
        }

        $attribute_value_products = AttributeValueProduct::whereIn('id', $attribute_value)
            ->where('product_id', $product->id)
            ->get();

        $price = 0;
        foreach ($attribute_value_products as $item) {
            $price += $item->price;
        }

        $permissions = AVPPImages::with('productImage')->whereHas('productImage', function ($q) use ($product) {
            $q->where('product_id', $product->id);
        })
            ->get();

        $image_attributes = [];

        foreach ($permissions as $permission) {
            $image_attributes[$permission->product_image_id][] = $permission->attribute_value_product_id;
        }

        $attribute_value = $attribute_value_products->pluck('id')->toArray();

        $image_keys = [];

        foreach ($image_attributes as $key => $image_attribute) {
            if (!empty(array_intersect($attribute_value, $image_attribute))) {
                array_push($image_keys, $key);
            }

            //if all items all sended array of attribute exist in permission image
//            if (count(array_intersect($attribute_value, $image_attribute)) === count($attribute_value)) {
//                array_push($image_keys, $key);
//            }
        }

        $images = ProductImage::whereIn('id', $image_keys)->get();

        $image_links = [];
```

```
        foreach ($images as $item) {
            $image_links[] = file_url($item->image);
        }

        return response()->json([
            'images' => $image_links,
            'price'  => $price + $product->price
        ]);
    }
}
```

baseController.php

```
<?php

namespace App\Api\v1\Http\Controllers;

use Illuminate\Http\Request;
use Illuminate\Http\JsonResponse;
use Illuminate\Routing\Controller as LaravelBaseController;
use Illuminate\Foundation\Bus\DispatchesJobs;
use Illuminate\Foundation\Validation\ValidatesRequests;
use Illuminate\Foundation\Auth\Access\AuthorizesRequests;

abstract class BaseController extends LaravelBaseController
{
    use AuthorizesRequests, DispatchesJobs, ValidatesRequests;

    /**
     * HTTP header status code.
     *
     * @var int
     */
    protected $statusCode = 200;
    /**
     * Illuminate\Http\Request instance.
     *
     * @var Request
     */
    protected $request;

    /**
     * Getter for statusCode.
     *
     * @return int
     */
    protected function getStatusCode()
    {
        return $this->statusCode;
    }
    /**
     * Setter for statusCode.
     *
     * @param int $statusCode Value to set
     *
     * @return self
     */
    protected function setStatusCode($statusCode)
    {
        $this->statusCode = $statusCode;
        return $this;
    }

    /**
     * Response with validation error.
     *
     * @param array $messages
     *
```

```
     * @return JsonResponse
     */
    protected function errorValidation(array $messages = [])
    {
        return $this->setStatusCode(422)->respondWithArray(['errors' => $messages]);
    }

    /**
     * Respond with success.
     *
     * @param $message
     * @return JsonResponse
     *
     */
    protected function respondWithSuccess($message = 'Success')
    {
        return $this->setStatusCode(200)->respondWithArray(['message' => $message]);
    }

    /**
     * Respond with a given array of items.
     *
     * @param array $array
     * @param array $headers
     *
     * @return JsonResponse
     */
    protected function respondWithArray(array $array, array $headers = [])
    {
        return response()->json($array, $this->statusCode, $headers);
    }
    /**
     * Response with the current error.
     *
     * @param string $message
     *
     * @return mixed
     */
    protected function respondWithError($message)
    {
        return $this->respondWithArray([
            'message'   => $message,
            'error' => [
                'http_code' => $this->statusCode,
            ],
        ]);
    }
    /**
     * Generate a Response with a 403 HTTP header and a given message.
     *
     * @param $message
     *
     * @return JsonResponse
     */
    protected function errorForbidden($message = 'Forbidden')
    {
        return $this->setStatusCode(403)->respondWithError($message);
    }
    /**
     * Generate a Response with a 500 HTTP header and a given message.
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorInternalError($message = 'Internal Error')
    {
        return $this->setStatusCode(500)->respondWithError($message);
    }
    /**
     * Generate a Response with a 404 HTTP header and a given message.
```

```
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorNotFound($message = 'Resource Not Found')
    {
        return $this->setStatusCode(404)->respondWithError($message);
    }
    /**
     * Generate a Response with a 401 HTTP header and a given message.
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorUnauthorized($message = 'Unauthorized')
    {
        return $this->setStatusCode(401)->respondWithError($message);
    }
    /**
     * Generate a Response with a 400 HTTP header and a given message.
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorWrongArgs($message = 'Wrong Arguments')
    {
        return $this->setStatusCode(400)->respondWithError($message);
    }

    /**
     * Generate a Response with a 418 HTTP header and a given message.
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorAuth($message = 'Ці облікові дані не відповідають нашим записам.')
    {
        return $this->setStatusCode(418)->respondWithError($message);
    }

    /**
     * Generate a Response with a 403 HTTP header and a given message.
     *
     * @param string $message
     *
     * @return JsonResponse
     */
    protected function errorLocked($message = 'Your account is locked')
    {
        return $this->setStatusCode(403)->respondWithError($message);
    }

    /**
     * Get the token array structure.
     *
     * @param  string $token
     *
     * @return \Illuminate\Http\JsonResponse
     */
    protected function respondWithToken($token)
    {
        return response()->json([
                'access_token' => $token,
                'token_type' => 'bearer',
                'expires_in' => auth('api')->factory()->getTTL() * 60
        ]);
```

ДОДАТОК Г
(обов’язковий)

ПРЕЗЕНТАЦІЙНІ МАТЕРІАЛИ

Хмельницький Національний університет
Факультет інформаційних технологій
Кафедра інженерії програмного забезпечення

Кваліфікаційна робота на тему: “Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника”

Виконав студент Качур Володимир Андрійович
Керівник: Бедратюк Л.П. д-р фіз.-мат. наук, проф.

Актуальність теми

Зважаючи на нинішню політичну та економічну ситуацію в Україні, розвиток будівельної галузі є важливим елементом відновлення економіки та інфраструктури країни. Крім того, будівельна галузь є однією з основних галузей національної економіки, яка прямо чи опосередковано забезпечує зайнятість великої кількості людей в Україні.

У сучасному світі, компанії, що займаються будівництвом, стикаються зі складними процесами управління проектами, відстеженням виконання робіт та спілкуванням зі своїми клієнтами. Ці процеси можуть бути вкрай складними та забирати велику кількість часу та зусиль.

З метою оптимізації цих процесів, розроблено веб-застосунку для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника. Такий веб-застосунок може включає в себе функціональні можливості для ефективного відстеження виконання робіт, управління проектами, а також спілкування з клієнтами.

Мета та завдання проекту

Мета проекту - розробка веб-застосунку, який забезпечує покращення взаємодії між компанією-забудовником та її клієнтами, що в свою чергу призведе до збільшення ефективності роботи та підвищення рівня задоволення клієнтів.

Для досягнення поставленої мети необхідно виконати наступні завдання:

- дослідити предметну область будівельного бізнесу та виявити потреби потенційних користувачів веб-застосунку;
- провести аналіз існуючих рішень;
- розробити технічне завдання;
- розробити архітектуру веб-застосунку та бази даних;
- обрати технології для розробки;
- розробити зручний та привітний інтерфейс для користувачів;
- розробити веб-застосунок;
- протестувати готовий веб-застосунок.

Порівняння наявного ПЗ

Основною перевагою наявних сайтів можна виділити обширно описане портфоліо компаній з повним переліком усіх зданих об’єктів, що є дуже важливим аспектом у формуванні довіри потенційних клієнтів, адже при перегляді портфоліо людина може пригадати що бачила ці будинки та може сформувати перше враження. Також присутня можливість онлайн - звязку із представниками відділу продажів, для уточнення будь яких питань потенційних клієнтів.

На жаль сайти містять досить велику кількість недоліків, серед яких можна виділити: застарілий дизайн та велика кількість текстового контенту на відносно невеликій площі модулів, що заплутує потенційного клієнта при перегляді сайту. Також відсутня можливість зміни локалізації, що складає негативне враження та відштовхує потенційних іноземних клієнтів.

Основними недоліками можна виділити повну відсутність будь якої інформації про стан будівництва об’єктів які зараз знаходяться на етапі будови, що має прямий вплив на кількість роботи з людьми у відділі роботи з клієнтами.

Порівняння наявного ПЗ

Назва додатку	Дизайн	Інтерфейс	Портфоліо	Система бронювання	Онлайн зв'язок	Статус будівництва
AVILA Building Group	5/10	6/10	Присутнє	Відсутня	Присутній	Відсутній
КомфортБудПлюс	6/10	7/10	Присутнє	Відсутня	Присутній	Присутній
IDM Group	9/10	9/10	Присутнє	Відсутня	Відсутній	Відсутній

Функціональні вимоги

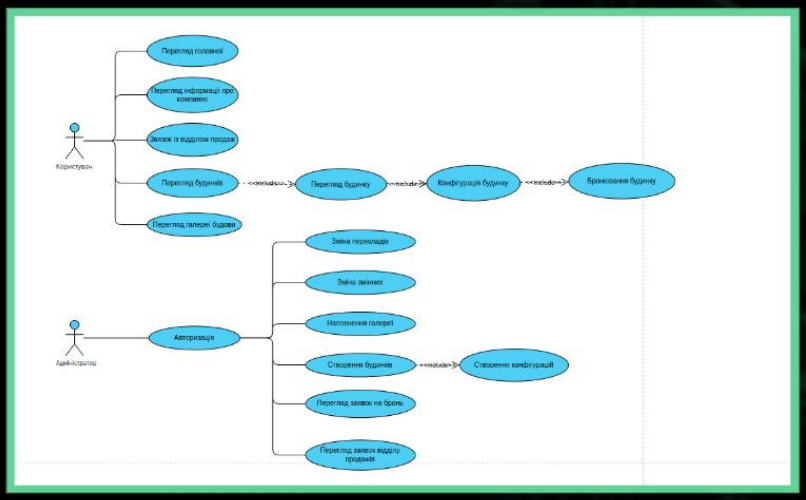
Для звичайного користувача:

- перегляд інформаційних блоків про компанію
- перегляд портфоліо компанії
- перегляд галерей будівництва
- окремі сторінки під різні будівлі з їх наявними плануваннями та статусами будови
- можливість звернутись онлайн до відділу продажів
- можливість переглянути юридичні документи компанії
- можливість конфігурації готового інтер'єру квартири
- можливість бронювання покупки квартири
- можливість перегляду веб-сайту на різних мовах

Для адміністратора:

- авторизація
- створення ролей адміністраторів з відповідними доступами до редагування вмісту сайту
- можливість створення та редагування блоків про компанію
- можливість додавання об'єктів у портфоліо компанії
- можливість завантаження фото/відео у галереї на веб-сайті
- можливість створення та редагування окремих сторінок будівель з усією необхідною інформацією
- можливість оновлювати статуси будови
- можливість завантаження планувань будови
- можливість опрацювання усіх вхідних онлайн-звернень з веб-сайту
- можливість загрузки та оновлення необхідних юридичних документів
- можливість створення модулів конфігурацій готових квартир
- система обліку усіх вхідних заявок на бронювання
- можливість редагування усього контенту сайту на різних мовах.

Діаграма варіантів використання



Використані технології

Архітектура рівня доступа до даних

Клієнт серверна архітектура

Архітектура рівня представлення

MVC - Model View Controller

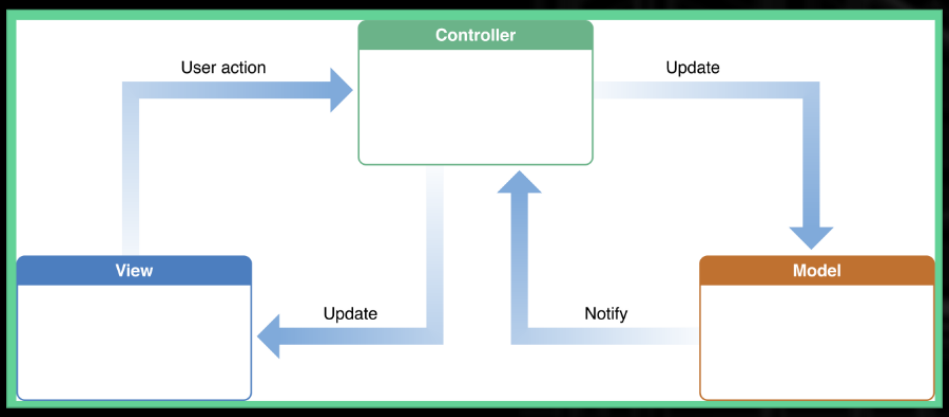
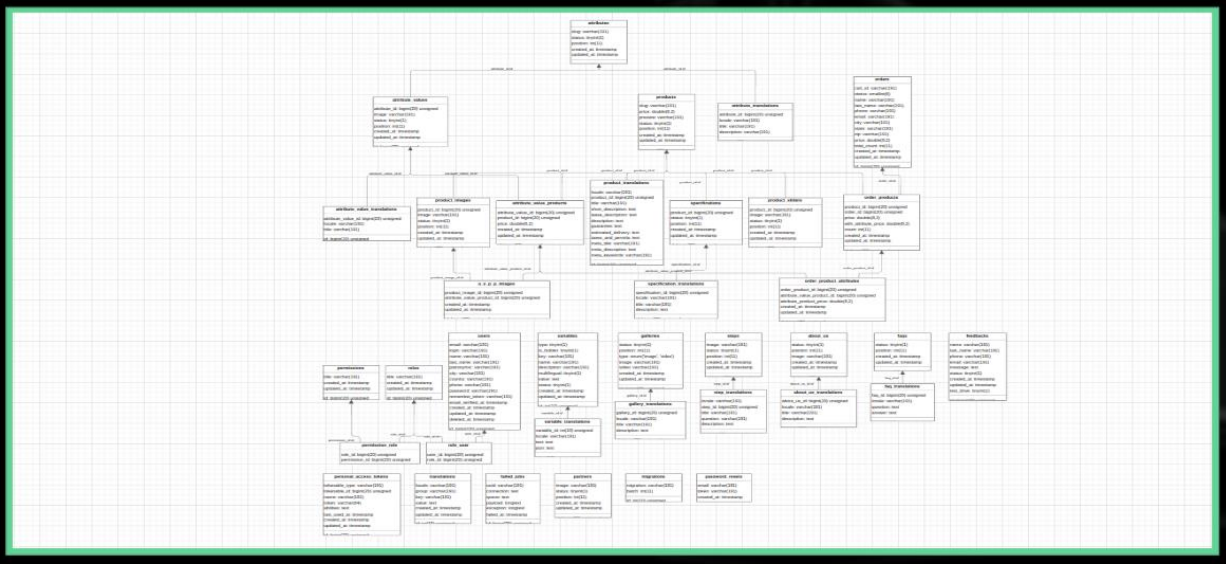
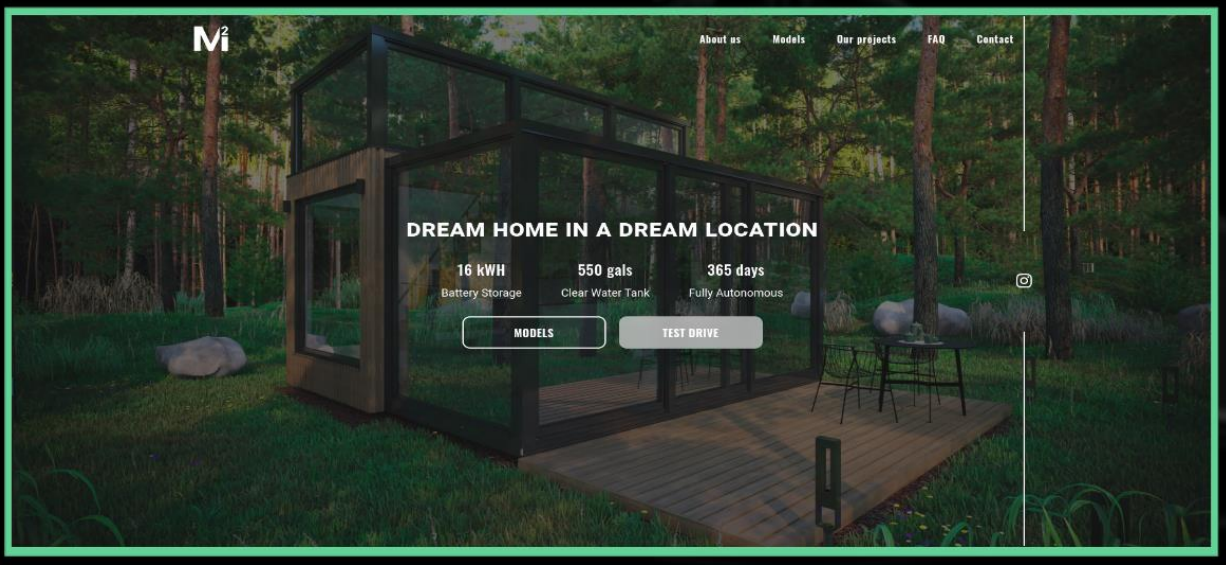


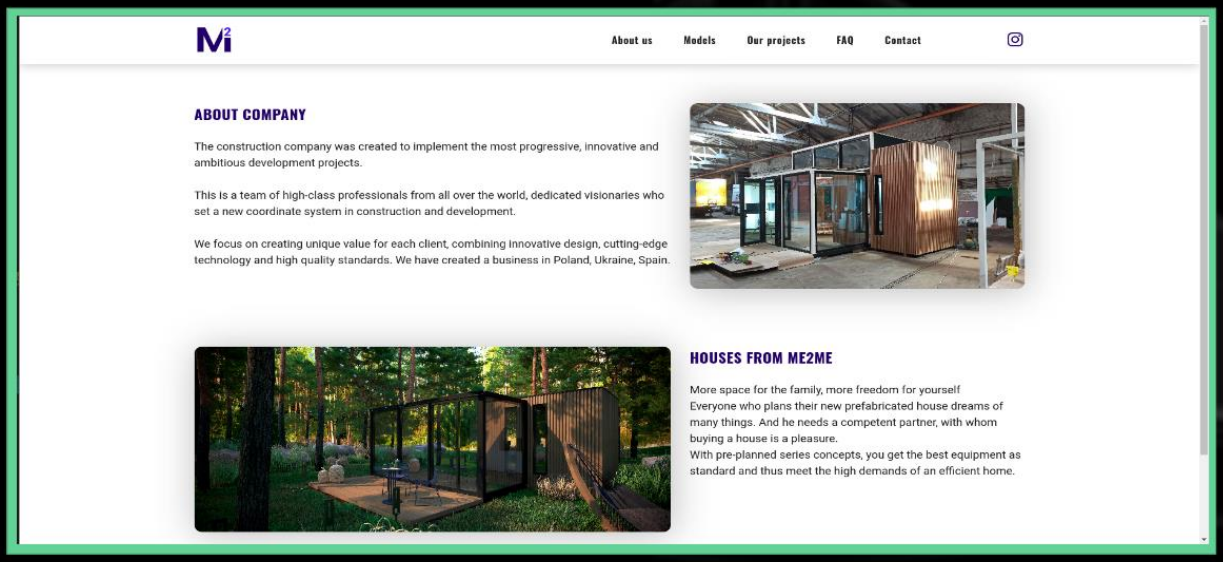
Схема бази даних



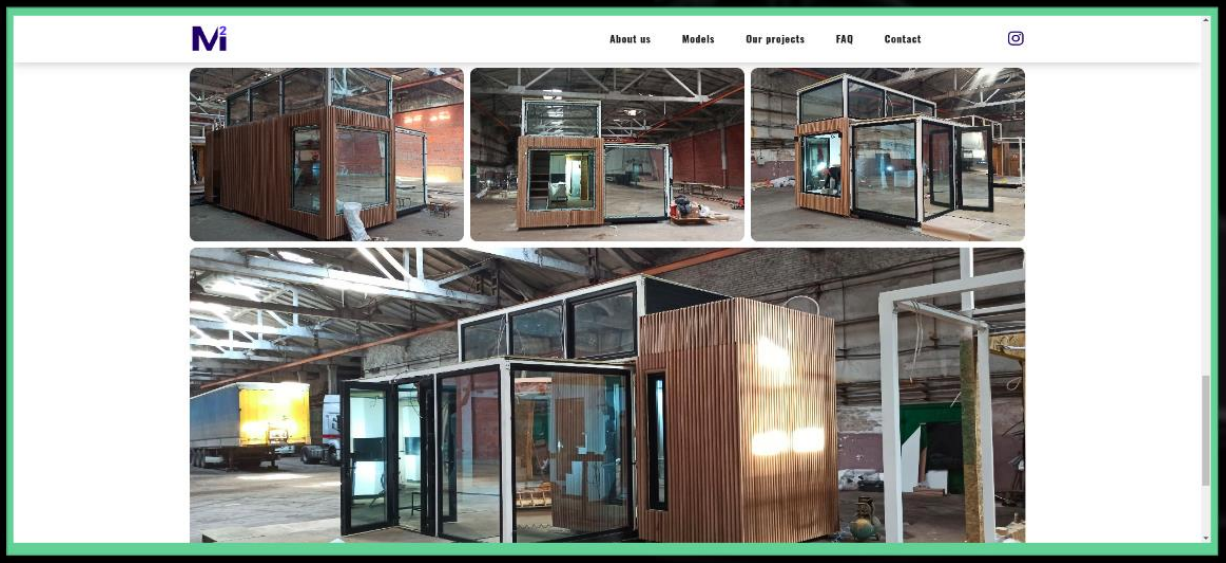
Реалізація проекту. Головна сторінка



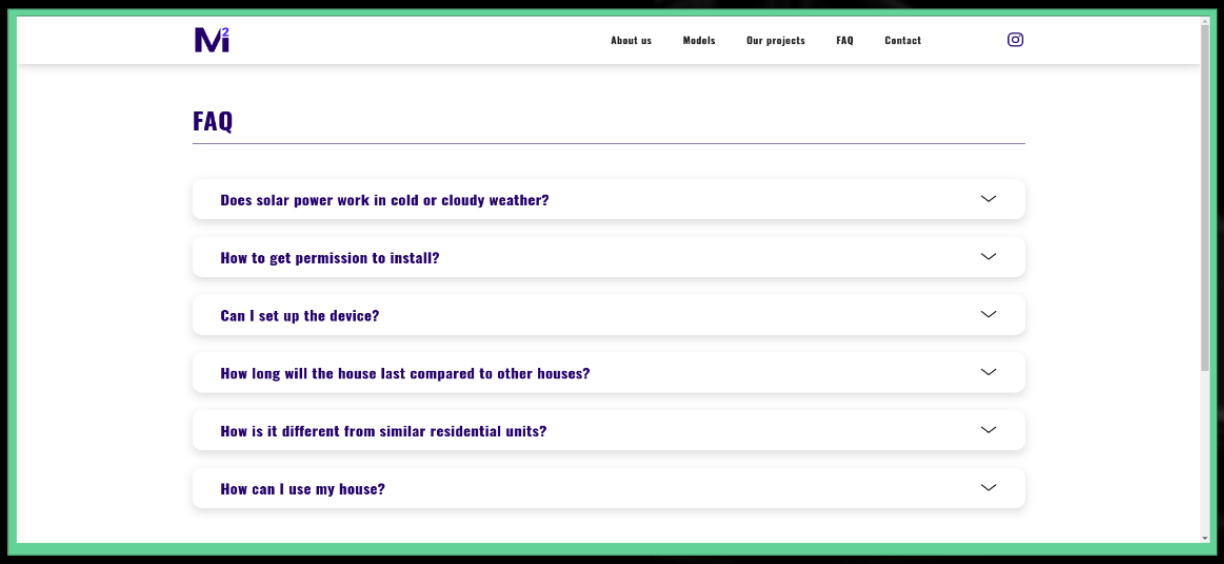
Реалізація проекту. Інформаційна сторінка про компанію



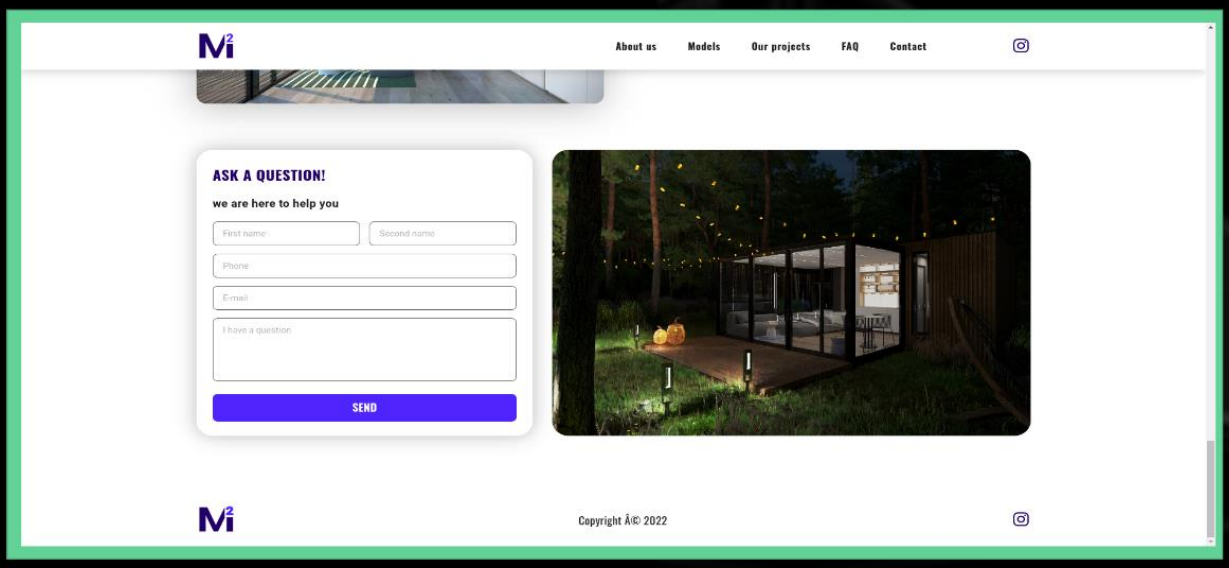
Реалізація проекту. Сторінка галереї робіт



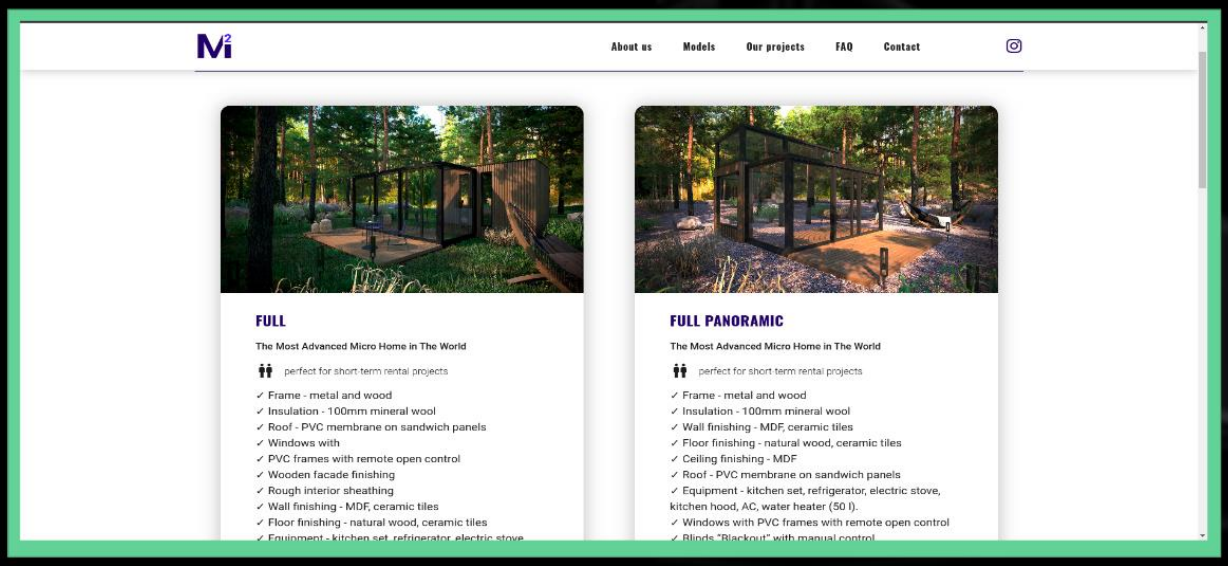
Реалізація проекту. Сторінка часто заданих питань



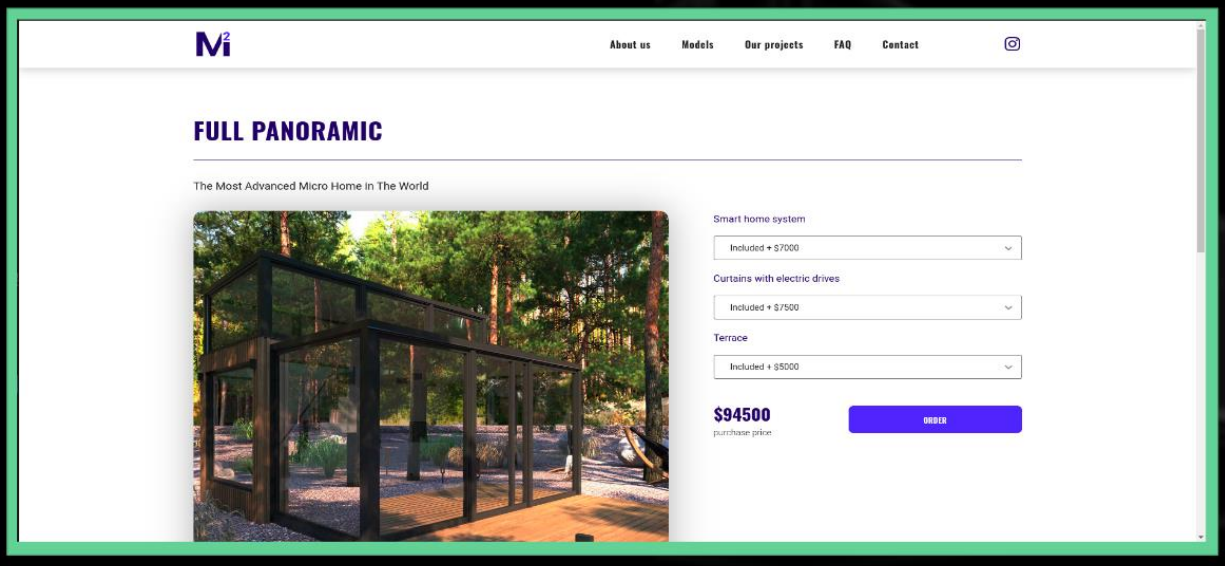
Реалізація проекту. Сторінка контактів із відділом продажів



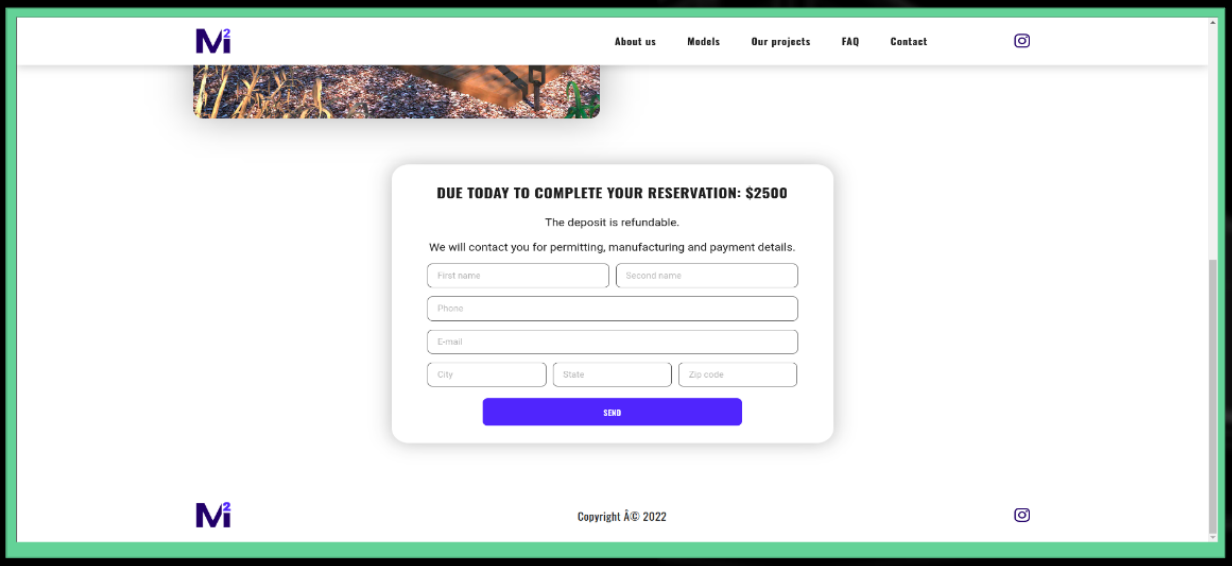
Реалізація проекту. Сторінка будинків у продажу



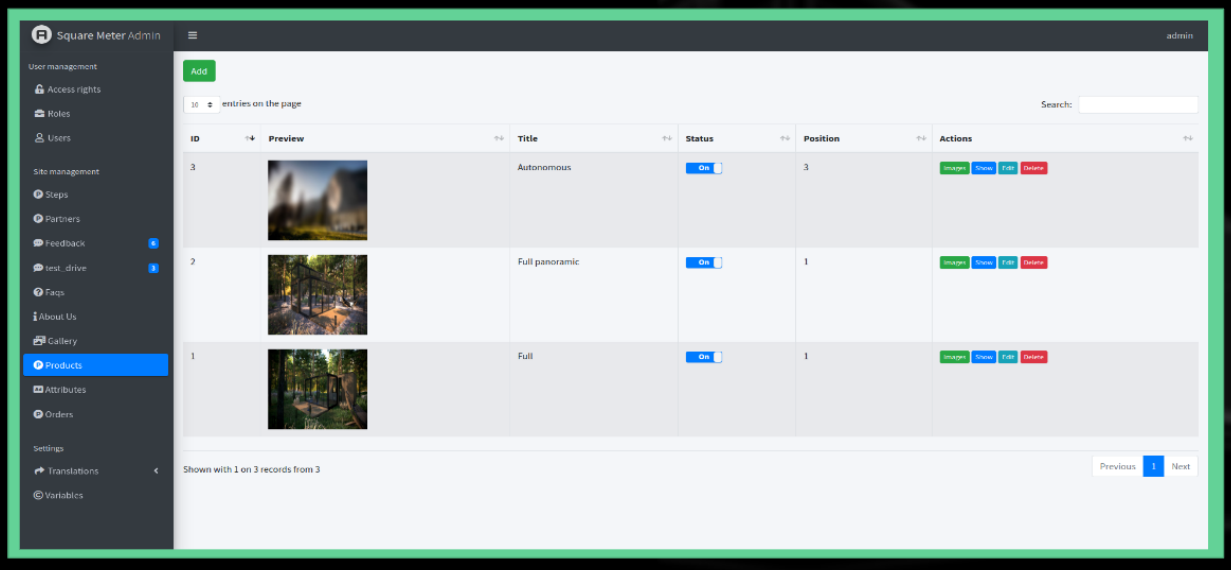
Реалізація проекту. Сторінка будинку



Реалізація проекту. Сторінка бронювання будинку



Реалізація проекту. Адмін панель



Висновки

Спроектовано інтерфейс, логіку та архітектуру веб-застосунку, базу даних і метод роботи API.

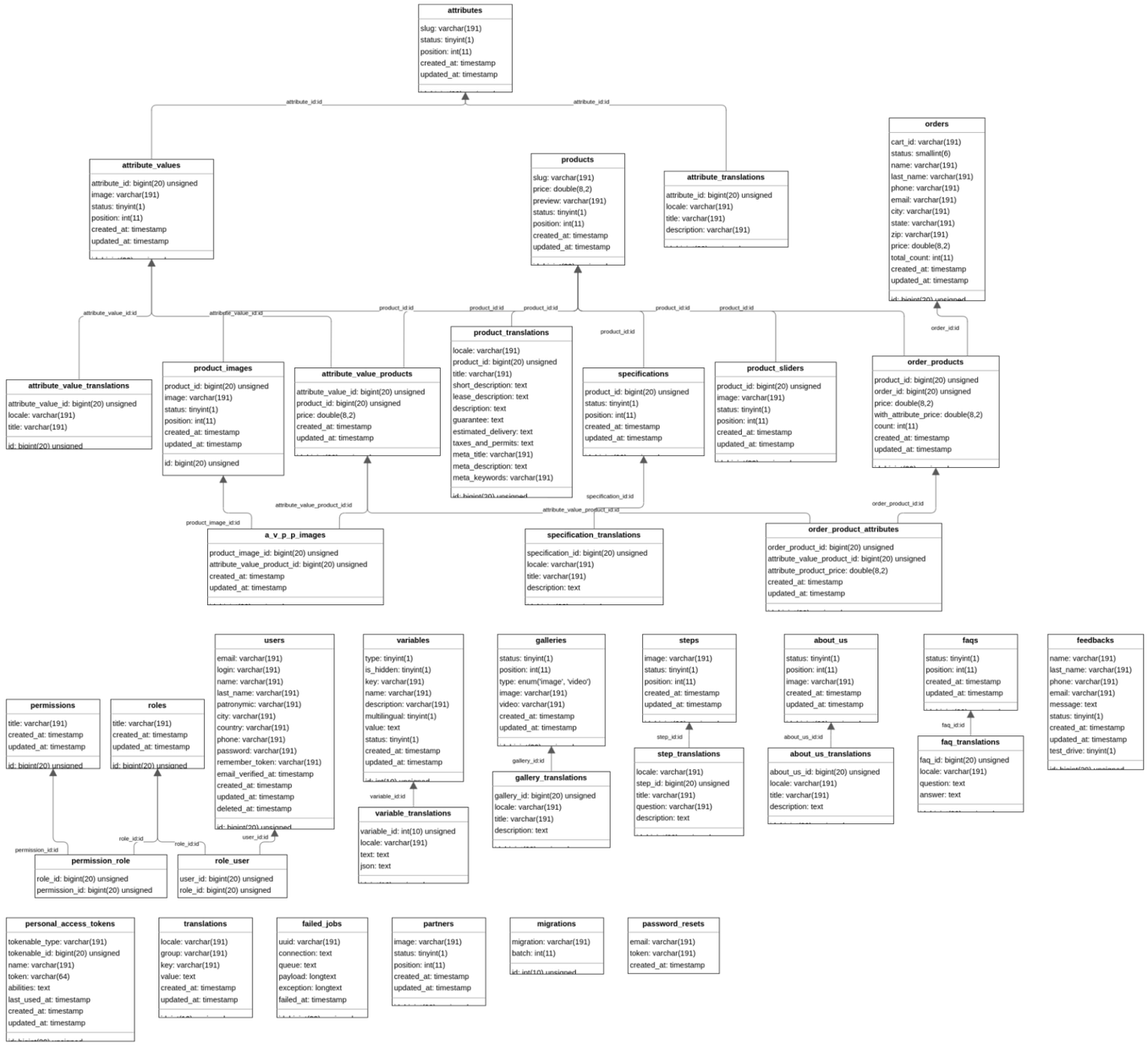
Створений веб-сайт відповідає поставленому завданню та усім визначеним вимогам.

Програмний продукт успішно сконструйований та готовий до використання. Судячи з цього, можна зробити висновок, що даний програмний продукт є цілком працездатним та справним.

Дякую за увагу

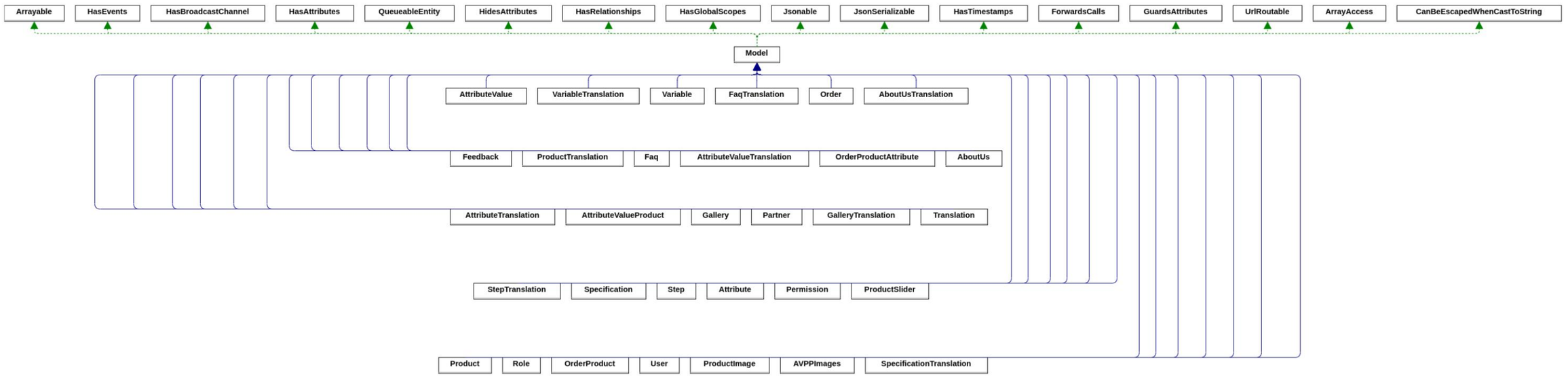
ГРАФІЧНА ЧАСТИНА

Рисунок 1 - Структура бази даних



					КвРІПЗ.200123.01.06.Е8			
					Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника - структура бази даних	Літера	Маса	Масштаб
Зм.	Арк.	№докум.	Підпис	Дата				
Розробив		Качур В.А.		22.05.23		Аркуш 1	Аркушів 3	
Керівник		Бедратюк Л.П.		22.05.23				
Консульт.						ХНУ, ІПЗс-20-1		
Н.Контр.		Праворська Н.І		22.05.23				
Зав.каф.		Бедратюк Л.П.		22.05.23				

Рисунок 2 - Структура моделей даних



					КвРІП3.200123.01.06.Е8			
Зм.	Арк.	№докум.	Підпис	Дата	Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника - структура моделей даних	Літера	Маса	Масштаб
Розробив		Качур В.А.		22.05.23				
Керівник		Бедратюк Л.П.		22.05.23		Аркуш 2	Аркушів 3	
Консульт.								
Н.Контр.		Праворська Н.І		22.05.23		ХНУ, ІПЗс-20-1		
Зав.каф.		Бедратюк Л.П.		22.05.23				

Рисунок 3 - Діаграма варіантів використання

					КвРІП3.200123.01.06.Е8			
					Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника - діаграма варіантів використання	Літера	Маса	Масштаб
Зм.	Арк.	№докум.	Підпис	Дата				
Розробив		Качур В.А.		22.05.23		Аркуш 3	Аркушів 3	
Керівник		Бедратюк Л.П.		22.05.23				
Консульт.						ХНУ, ІПЗс-20-1		
Н.Контр.		Праворська Н.І		22.05.23				
Зав.каф.		Бедратюк Л.П.		22.05.23				

Завідувачу кафедри
інженерії програмного забезпечення
проф. Бедратюку Л. П.
студента групи _____
Качура В. А.
Прізвище, ініціали

ЗАЯВА

Прошу закріпити за мною тему кваліфікаційної роботи освітнього ступеня «бакалавр» за спеціальністю 121 «Інженерія програмного забезпечення»: Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компаній-забудовників.

(керівник роботи – Бедратюк Леонід Петрович)
Прізвище, ім'я, по батькові

05.02.2023
Дата

Підпис студента

Завідувачу кафедри інженерії програмного забезпечення проф. Бедратюку Л. П.

здобувача вищої освіти
Качура В.А.
Прізвище, ініціали

факультет ІТ, 3 курс, група ІПЗс-20-1

ЗАЯВА

З правилами чинного Положення «Про дотримання академічної доброчесності в Хмельницькому національному університеті» від 26.09.2020 (зі змінами від 26.11.2020), згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений. Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

25.05.23
дата

підпис

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ДЕКЛАРАЦІЯ УЧАСНИКА ОСВІТНЬОГО ПРОЦЕСУ

щодо дотримання академічної доброчесності

Цією декларацією я, Качур Володимир Андрійович

Прізвище, ім'я, по батькові

здобувач вищої освіти (шифр та назва спец-ті, курс, академічна група)/ науково-педагогічний працівник (назва кафедри)

назва факультету

підтверджую, що ознайомився (- лась) з Положенням про систему забезпечення академічної доброчесності в Хмельницькому національному університеті та Кодексом академічної доброчесності і **зобов'язуюсь** дотримуватися їх вимог під час освітнього процесу, проведення наукової діяльності, виконання організаційно-адміністративних функцій тощо.

Усвідомлюю, що у разі порушення мною принципів академічної доброчесності нестиму відповідальність перед академічною спільнотою ХНУ згідно з нормами, визначеними Положенням систему забезпечення академічної доброчесності в Хмельницькому національному університеті, законодавства України.

«05» лютого 2023 р.

Підпис

Thu May 25 18:39:42 EEST 2023, Хіврич Володимир Русланович, Хмельницький національний університет, ХНУ

Anti-Plagiarism v-15.257

Максимальне співпадіння з одним документом 22.0%

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 11%**

ID: 114045 Назва: БКР Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника Додано в БД: 2023-05-25 Автора: Бедратюк Л.П. Керівники: Качур В.А. Консультанти: Опоненти:	Документ		Сумарний збіг по Базі Даних	
	Символи	Лексеми	Символи	Лексеми
	71194	626	19540 (27%)	172 (27%)

Джерело плагіату

ID	Опис	Наявність плагіату в документі	
		Символи	Лексеми
111874	Назва: Звіт з переддипломної практики бакалавр на тему: Веб застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника Додано в БД: 2023-03-02 Автора: Качур В.А. Керівники: Бедратюк Л.П. Консультанти: Опоненти:	15406 (22.0%)	124 (20.0%)
104085	Назва: Інтернет-платформа «Агентство подорожей» Додано в БД: 2022-05-27 Автора: В.В. Собко Керівники: Н.І. Праворська Консультанти: Опоненти:	4767 (7.0%)	69 (11.0%)

UNICHECK

Ім'я користувача:
Кафедра ІПЗ

Дата перевірки:
25.05.2023 19:38:39 EEST

Дата звіту:
25.05.2023 20:00:12 EEST

ID перевірки:
1015257491

Тип перевірки:
Doc vs Internet + Library

ID користувача:
100005589

Назва документа: Качур_ІПЗс-20-1_Диплом-1.1

Кількість сторінок: 69 Кількість слів: 10418 Кількість символів: 83153 Розмір файлу: 24.26 MB ID файлу: 101493

Виявлено модифікації тексту (можуть впливати на відсоток схожості)

10.9%
Схожість

Найбільша схожість: 7.92% з джерелом з Бібліотеки (ID файлу: 1011233803)

3.4% Джерела з Інтернету 7 Сторінка 71

8.27% Джерела з Бібліотеки 118 Сторінка 77

0% Цитат

Вилучення цитат вимкнене

Вилучення списку бібліографічних посилань вимкнене

0%
Вилучень

Немає вилучених джерел

Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

Підозріле форматування 14 сторінок

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ
КАФЕДРИ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ
ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника
Автор: Качур Володимир Андрійович
Спеціальність: 121 – Інженерія програмного забезпечення
Освітня програма: Освітньо-професійна програма «Інженерія програмного забезпечення»
Науковий керівник: Бедратюк Леонід Петрович, д-р фіз-мат. наук, професор
Після аналізу звіту подібності зроблено такий висновок:

№	Висновок	Позначка про відповідність
1	Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту.	**відповідає**
2	Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи	
3	Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат.	
4	Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту.	
5	Інше:	

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) у тексті дипломної роботи системами перевірки на плагіат виявлено схожість з деякими документами в частині загальновживаних обов'язкових словосполучень у стандартних бланках (титулка, бланк завдання), у структурі змісту, назвах розділів/підрозділів тощо та в назвах публікацій у переліку джерел посилання;

2) в якості запозичень системою було зафіксовано деякі послідовності вихідного коду і посилання на бібліотеки, які є стандартними мовними конструкціями програмування та не можуть розглядатися як об'єкт авторських прав і, відповідно, їх порушення;

3) усі запозичення є фрагментарними або мають належним чином оформленні посилання;

4) виявлені 22 відсотки запозичень стосуються звіту з практики і не є плагіатом.

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів ідентичності/ схожості, складає 22 % і адресується до звіту з переддипломної практики, який містить у собі частину кваліфікаційної роботи, що, з урахуванням наведених обґрунтувань, відповідає характеру теми і свідчить на користь дипломного проекту.

Керівник _____ Л. П. Бедратюк

Гарант ОП _____ Л. П. Бедратюк

Завідувач кафедри _____ Л. П. Бедратюк

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ
освітнього ступеня «Бакалавр»

Дипломник Качур Володимир Андрійович

Тема Веб-застосунок для автоматизації клієнто-орієнтованих бізнес-процесів компанії-забудовника.

Спеціальність 121 – Інженерія програмного забезпечення

Обсяг кваліфікаційної роботи:

Кількість листів креслень 0; кількість сторінок записки 70

1.Короткий зміст пояснювальної записки та прийнятих рішень У кваліфікаційній роботі було досліджено і проаналізовано предметну область будівельного бізнесу та виявлено потреби потенційних користувачів програмного продукту. Був проведений аналіз відомих рішень, визначено їх переваги і недоліки, доведено актуальність розробки нового програмного забезпечення. Було розроблено технічне завдання, архітектуру веб-застосунку та бази даних, вибрано технології для розробки, розроблено зручний для користувачів інтерфейс, розроблено веб-застосунок, а також протестовано готовий веб-застосунок.

2. Висновок про відповідність проекту поставленому завданню Кваліфікаційна робота виконана відповідно до поставленого завдання та з дотриманням вимог.

3. Характеристика виконання кожного розділу проекту, ступінь використання останніх досягнень науки і техніки та передових методів роботи У вступі доведено актуальність теми, визначено мету та завдання кваліфікаційної роботи. У першому розділі проведено аналіз предметної області, розглянуто відомі рішення та визначені функціональні і нефункціональні вимоги до розроблюваного веб-застосунку. У другому розділі проведено аналіз сучасних архітектур, розглянуто їх переваги і недоліки та визначено, що система буде відповідати монолітній архітектурі та моделі клієнт-сервер. У третьому розділі підготовлено всі залежності для написання коду та виконано практичну розробку програмних модулів і описано їх особливості, в результаті чого створено програмний продукт. В четвертому розділі було виконано тестування системи у відповідності до функціональних вимог, в результаті чого було підтверджено правильну роботу веб-застосунку.

4. Позитивні сторони проекту Тематика кваліфікаційної роботи є актуальною, оскільки користувачі потребують простої і доступної програмної системи, що надавала б перевагу над іншими подібними рішеннями в галузі будівництва. Також було застосовано сучасні технології для побудови веб-застосунку та актуальні архітектурні рішеня.

5. Негативні сторони проекту Робота реалізована без технології SSR, тому не має можливості динамічної зміни SEO складової сторінок, що може негативно впливати

на SMM складову для адміністраторів веб-застосунку.

6. Оцінка графічного оформлення та пояснювальної записки проекту Графічне оформлення виконано відповідно до теми кваліфікаційної роботи. Пояснювальна записка оформлена згідно вимог чинних стандартів.

7. Відгук про кваліфікаційну роботу в цілому Кваліфікаційна робота заслуговує позитивної оцінки. Викладення матеріалу пояснювальної записки є структурованим, послідовним та чітким, що дозволяє зрозуміти викладений матеріал у рамках тематики кваліфікаційного проекту. Графічний матеріал надає можливість наочно побачити деталі проектування веб-застосунку.

8. Інші зауваження

9. Оцінка кваліфікаційної роботи Кваліфікаційна робота виконана у повному обсязі, відповідає поставленій задачі та заслуговує на оцінку «добре».

РЕЦЕНЗЕНТ Бобровнікова Кіра Юліївна, кандидат технічних наук, доцент кафедри комп'ютерної інженерії та інформаційних систем (КІІС) ХНУ.

“ 26 ” травня 2023 р. _____
(підпис)