МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«До захисту допущено»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)
2024 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук,
освітньо-наукової програми «Інформатика»
на тему: «Інформаційна система керування веб-платформою проектно-виробничого архітектурно-планувального бюро»
здобувача групи ІНз-01с Тарана Дмитра Вадимовича

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело.

_____ Дмитро ТАРАН
(підпис)

Керівник,
старший викладач,
кандидат технічних наук Артем КОРОБОВ _____
(підпис)

Суми – 2024

Сумський державний університет
Центр заочної, дистанційної та вечірньої форм навчання
Кафедра комп'ютерних наук

«Затверджую»
В.о. завідувача кафедри
_____ Ігор ШЕЛЕХОВ
(підпис)

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня бакалавр

зі спеціальності 122 - Комп'ютерних наук, освітньо-наукової програми «Інформатика»
здобувача групи ІНз-01с Тарана Дмитра Вадимовича

1. Тема роботи: «Інформаційна система керування веб-платформою проектно-виробничого архітектурно-планувального бюро»
затверджую наказом по СумДУ від «26» квітня 2024 р. № 0438-VI

2. Термін здачі здобувачем кваліфікаційної роботи до «05» червня 2024 року

3. Вхідні дані до кваліфікаційної роботи

4. Зміст розрахунково-пояснювальної записки (перелік питань, що їх належить розробити)
1) Аналіз методів оптимізації роботи архітектурно-планувального бюро. 2)Вибір технологій та інструментів, які підходять для вирішення поставлених завдань. 3) Розробка інформаційної системи для архітектурно-планувального бюро. 4) Аналіз результатів.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)

6. Консультанти до проекту (роботи), із значенням розділів проекту, що стосується їх

Розділ	Консультант	Підпис, дата	
		Завдання видав	Завдання прийняв

7. Дата видачі завдання «____» _____ 20 ___ р.

Завдання прийняв до виконання _____ (підпис) Керівник _____ (підпис)

КАЛЕНДАРНИЙ ПЛАН

№ п/п	Назва етапів кваліфікаційної роботи	Термін виконання	Примітка
1	*Аналіз проблеми предметної області, постановка й формування завдань дослідження*		
2	*Огляд технологій, що використовуються для написання веб-додатків*		
3	*Розробка інформаційної системи для архітектурно-планувального бюро*		
4	*Аналіз отриманих результатів*		
5	*Оформлення пояснювальної записки до кваліфікаційної роботи*		

Здобувач вищої освіти _____ Керівник _____
(підпис) (підпис)

АНОТАЦІЯ

Записка: 43 стр., 19 рис., 13 джерел.

Обґрунтування актуальності теми роботи – тема роботи є актуальною, оскільки вона вирішує реальні бізнес-проблеми на підприємстві. В ході роботи були вирішені проблеми як з боку клієнтів, так і з боку керівників підприємства.

Об’єкт дослідження — процес взаємодії клієнтів з послугами архітектурно-планувального бюро та контролю даними проектів.

Предмет дослідження — методи розробки веб-додатку для задоволення потреб архітектурно-планувального бюро.

Мета роботи — створення зручного та ефективного додатку для вирішення бізнес-проблем взаємодії клієнтів з послугами архітектурно-планувального бюро та можливостями для управління проектами.

Методи дослідження — технології для створення ефективного та зручного веб-додатку для архітектурно-планувального бюро.

Результати — готовий веб-додаток для вирішення проблем взаємодії клієнтів з послугами архітектурно-планувального бюро, управлінням даними проектів з боку керівників бюро, збору та відображенню статистики продаж.

ДИСТАНЦІЙНЕ НАВЧАННЯ, ВЕБ-РОЗРОБКА, FRONTEND, BACKEND, JAVASCRIPT, REACT JS, NEST JS.

ЗМІСТ

ВСТУП

Актуальність. У сучасному світі, де швидкість і ефективність грають ключову роль у бізнесі, інформаційна система стає необхідним інструментом для оптимізації процесів. Розробка веб-платформи для архітектурно-планувальних бюро є актуальною, оскільки це дозволяє ефективно представляти їхні послуги та привертати клієнтів, а також надає можливість ефективно контролювати та аналізувати дані проектів. Використання сучасних технологій, таких як React.js, Nest.js та MongoDB, дозволяє створити потужну та надійну інформаційну систему. Ця робота допоможе відповісти на реальні потреби архітектурно-планувальних бюро в управлінні проектами та аналізі їх ефективності. Такий підхід не лише сприятиме підвищенню конкурентоспроможності, але й забезпечить краще задоволення потреб клієнтів.

Об’єкт дослідження. Процес взаємодії клієнтів з послугами архітектурно-планувального бюро та контролю даними проектів.

Предмет дослідження. Методи розробки веб-додатку для задоволення потреб архітектурно-планувального бюро.

Гіпотеза. Система суттєво полегшить взаємодію користувачів з послугами, значно підвищить ефективність роботи керівників бюро.

Новизна. Дана система пропонує новий підхід у функціонуванні архітектурно-планувального бюро, в якій головною частиною є зручний і мінімалістичний інтерфейс та швидкий і легко масштабований бекенд. Для розробки обох частин було використано актуальні та нові технології.

Структура. Дана кваліфікаційна робота складається зі вступу, аналізу предметної області, постановки задачі, вибору програмних засобів для вирішення поставленої мети, опису програмного забезпечення інформаційної системи, висновків, списку використаних джерел та додатків.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Дослідження актуальності веб-технологій

З кожним роком веб-розробка стає все більш важливою, оскільки наявність власної веб-платформи у підприємстві відіграє ключову роль у сучасному світі як для комерційних, так і для некомерційних цілей. Веб-технології швидко змінюються, і нові інструменти та фреймворки постійно з'являються для полегшення процесу розробки веб-додатків.

За останні десятиліття спостерігається експоненційне зростання кількості веб-ресурсів. Згідно з даними статистичних порталів, кількість веб-сайтів перевищує вже понад 1 мільярд, та цей показник продовжує зростати. Це свідчить про постійне збільшення популярності веб-технологій та необхідність вдосконалення існуючих інструментів для розробки веб-проектів. За даними Gartner, використання веб-технологій у бізнесі продовжує зростати. Більшість компаній нині активно використовують веб-платформи для підтримки своїх бізнес-процесів, спілкування з клієнтами та маркетингу. Існують авторитетні оцінки кількості сайтів у певних категоріях, наприклад таких як сайти електронної комерції. На момент написання кількість ecommerce веб-сайтів у всьому світі становить 26,5 мільйонів [1]. З 2019 року це число зросло в геометричній прогресії (на 188%).(рис 1.1)

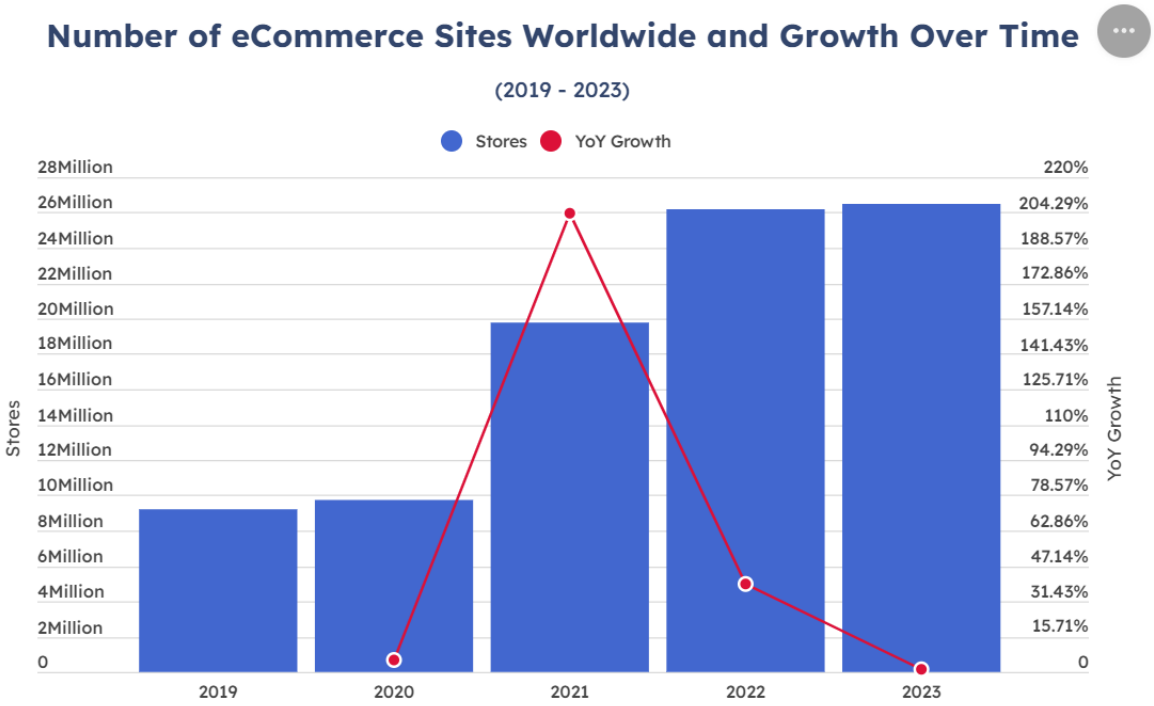

Рис 1.1 - Графік кількості комерційних веб-сайтів у світі з 2019 по 2023 р.

Із розвитком електронної комерції, веб-платформи стають все більш популярними як інструмент для розміщення та просування товарів та послуг. За даними Forbes [2], обсяг електронної комерції вже перевищив 6 трільйонів доларів у 2024 році, що демонструє великий попит на різноманітні веб-платформи. Також очікується, що обсяг продаж електронної комерції буде значно зростати. (рис 1.2)

Рис. 1.2 - Зростання обсягу продаж електронної комерції в світі з 2022 по 2027 роки. (у трильйонах)

Очікується, що в 2024 році 20,1% роздрібних покупок відбуватимуться онлайн.

З кожним роком дедалі менше людей роблять покупки у звичайних магазинах. Щоб залишатися попереду, компанії мають надати клієнтам ефективний спосіб купувати їх товари або послуги в Інтернеті. Замість того, щоб вкладати більше грошей у звичайні магазини, все більше підприємств думають про те, щоб направити цей бюджет на розробку та маркетинг веб-додатку. Здебільшого це обумовлено наступними чинниками:

- дають можливість привернути клієнтів з будь-якої точки світу, оскільки доступні вони через Інтернет. Це розширює потенційну аудиторію і збільшує можливість здійснення продажів;
- створення та підтримка зазвичай вимагає менших витрат порівняно з традиційним магазином. Немає необхідності в оренді приміщення, оплаті комунальних послуг, а також менше потреба в персоналі;
- працюють цілодобово, що дозволяє клієнтам здійснювати покупки в будь-який час, незалежно від географічного положення або робочого

графіка. Це збільшує зручність для покупців і може призвести до збільшення обсягу продажів;

- мають широкі можливості для маркетингу та реклами через цифрові канали, такі як соціальні медіа, пошукові системи та електронна пошта. Це дозволяє ефективно спрямовувати рекламні зусилля на цільову аудиторію і збільшувати обсяги продажів;
- дозволяють збирати великий обсяг даних про поведінку користувачів, їхній інтерес та переваги. Це дає можливість проводити аналіз ефективності та оптимізувати процеси продажу, пристосовуючи асортимент та маркетингові стратегії.

Отже можна з упевненістю сказати, що веб-технології продемонстрували свою актуальність і важливість у сучасному світі, що підтверджується рядом факторів. Перш за все, зростання кількості веб-ресурсів, що використовуються як для комерційних, так і для некомерційних цілей, свідчить про постійне зростання популярності веб-технологій.

Проаналізувавши переваги створення веб-додатків для компаній, можна сказати, що інвестування коштів у веб-технології, такі як розробка веб-платформ, має потенціал для високого віддачі завдяки глобальному доступу, зменшенню витрат, цілодобовій доступності, можливостям маркетингу та аналітиці. Це дає компаніям широкі можливості для розвитку, залучення нових клієнтів та збільшення обсягів продажів.

У цілому, веб-технології залишаються ключовим інструментом для розвитку сучасного бізнесу та суспільства в цілому, і їхня актуальність та важливість буде продовжувати зростати в майбутньому.

1.2 Аналіз існуючих веб-додатків архітектурно-планувального бюро

В сфері архітектурно-планувального бізнесу веб-додатки відіграють значну роль у спрощенні процесів проектування, візуалізації та спілкування з клієнтами. Однак, варто відзначити, що ринок таких веб-рішень є відносно обмеженим, і вибір відповідного інструменту може бути складним завданням. Кожен веб-додаток має свої переваги та недоліки, і важливо провести детальний аналіз існуючих рішень для визначення найбільш підходящого для конкретних потреб клієнтів. У даному розділі мною буде розглянуто декілька прикладів веб-сайтів архітектурно-планувальних бюро, я порівняю їх та проаналізую їхні переваги та недоліки.

Archline Design - ArchLine - це веб-додаток, який пропонує послуги з проектування та будівництва будинків. На сайті можна знайти різноманітні проекти будинків, від класичних до сучасних, та замовити індивідуальний проект за власними потребами.

Переваги:

1. **Якісний інтерфейс:** дизайн додатку виглядає дуже красиво і зручно, та є адаптивним, тобто якісно функціонує на різних девайсах, що робить взаємодію з сайтом дуже приємною та зручною.
2. **Онлайн консультація з дизайнером:** сайт надає можливість здійснити онлайн консультацію з дизайнером, що дозволяє клієнтам отримати професійну допомогу та поради щодо вибору проекту та виконання індивідуальних замовлень.
3. **Індивідуальний підхід:** Можливість замовлення індивідуального проекту дозволяє клієнтам реалізувати свої унікальні ідеї та потреби.

Недоліки:

1. **Брак детальної інформації:** Деяка інформація про проекти може бути обмеженою, що ускладнює вибір клієнтів.

2. **Відсутність української локалізації:** Сайт працює тільки на англійській та російській мові, що може стати перешкодою для українських користувачів, які бажають отримати інформацію або скористатися послугами на своїй рідній мові.
3. **Відсутність проектів приватних будинків і прибудов:** Сайт обмежений лише проектами квартир, тому відсутність інших видів проектів, таких як часні будинки або прибудови, може бути недоліком для клієнтів, які шукають інші типи будівельних проектів.

У висновку можна сказати, що ArchLine пропонує дуже якісний інтерфейс, широкий вибір проектів квартир та надає можливість отримати онлайн консультацію з дизайнером. Однак, обмеженість сайту лише проектами квартир, без інших видів будівельних проектів та відсутність української локалізації, може стати перешкодою для деяких клієнтів.

BorisStudio - це архітектурне бюро, яке спеціалізується на проектуванні, дизайні, ремонті та будівництві різноманітних об'єктів, включаючи житлові будинки, офісні приміщення, комерційні об'єкти та інші.

Переваги:

1. **Широкий вибір послуг:** Бюро надає різноманітні послуги, що охоплюють усі етапи проектування та будівництва, включаючи дизайн інтер'єру, архітектурне проектування, будівництво та ремонт.
2. **Партнерства зі світовими брендами:** Борис Студіо має партнерства з провідними компаніями у сфері будівництва та дизайну, включаючи Franke, Maytoni, Grohe та інші, що гарантує використання якісних матеріалів та обладнання у проектах.
3. **VR-тур:** Наявність VR-туру дозволяє клієнтам переглянути готові дизайни з першої особи, що надає можливість краще уявити собі простір та дизайн об'єкта перед початком будівництва чи ремонту.

Недоліки:

1. **Складний інтерфейс:** На сайті присутній дуже великий обсяг розділів та інформації, що може ускладнити навігацію та зробити процес замовлення послуг менш зручним для клієнтів. Зайві розділи та інформація можуть бути спрощені або зовсім видалені для полегшення взаємодії з веб-сайтом.
2. **Обмежене охоплення географією:** Бюро працює тільки у великих містах, таких як Київ, Харків та Львів, що може обмежити доступність послуг для жителів інших регіонів.

Отже BorisStudio - це веб-сайт архітектурного бюро з широким спектром послуг та індивідуальним підходом до кожного клієнта. Незважаючи на деякі обмеження, такі як складний інтерфейс та обмежене охоплення географією, наявність VR-туру дозволяє клієнтам отримати більш чітке уявлення про готові дизайни об'єктів, а партнерства зі світовими брендами гарантують використання якісних матеріалів та обладнання у всіх проектах.

Після аналізу існуючих веб-додатків архітектурно-планувального бюро можна зробити декілька важливих висновків. Веб-технології швидко розвиваються, що призводить до появи різноманітних інструментів для архітектурного проектування та планування. Всі додатки мають свої переваги та недоліки, але вони спрямовані на полегшення робочого процесу архітекторів та забезпечення клієнтам доступу до якісних послуг.

Перш за все, важливо відзначити, що існують веб-додатки, такі як "Archline", які надають можливість клієнтам замовити проекти будинків та скористатися різноманітними інструментами для візуалізації. Однак, важливо враховувати, що багато з цих додатків можуть бути обмежені у функціональності та можливостях адаптації під конкретні потреби клієнтів.

Деякі архітектурні бюро, такі як "Boris Studio", надають послуги не лише в дизайні та проектуванні, а й у будівництві та ремонті. Це важливо з точки зору комплексного обслуговування клієнтів, але такі бюро можуть

стикатися з викликами, пов'язаними зі складністю інтерфейсу та обмеженим охопленням географією.

У цілому, веб-додатки архітектурно-планувального бюро стають все більш важливими у сучасному світі. Вони допомагають спростити процес проектування та планування, забезпечуючи доступ до якісних послуг для клієнтів. Однак важливо продовжувати вдосконалювати ці додатки, забезпечуючи їхню високу функціональність та зручний інтерфейс для кінцевих користувачів.

1.3 Постановка задачі

Метою розробки цього додатку є створення інформаційної системи керування веб-платформою для архітектурно-планувального бюро. Головною метою є забезпечення зручного та ефективного інструмента для керування проектами будівництва та дизайну архітектурних об'єктів, а також для забезпечення зручного спілкування з клієнтами та аналізу статистичних даних.

Вимоги додатка:

- повинен мати зрозумілий та легкий у використанні інтерфейс для користувачів будь-якого рівня;
- повинен надавати можливості додавання, редагування, видалення проектів будівництва;
- наявність інструментів для аналізу статистичних даних, таких як витрати, продажі, розподіл проектів по категоріям та інші параметри.
- інтеграція засобів зв'язку для комунікації з клієнтами, включаючи можливість відправки повідомлень та організацію онлайн консультацій.

Задачі для досягнення поставленої мети:

1. Розробка функціональних вимог до додатку, включаючи визначення основних функцій та їхню взаємодію.
2. Проектування інтерфейсу користувача, з урахуванням зручності та ергономіки.
3. Розробка бекенду додатку для забезпечення зберігання та обробки даних.
4. Розробка функціоналу для аналізу та звітності.
5. Інтеграція засобів зв'язку для взаємодії з клієнтами.
6. Тестування та відлагодження додатку перед випуском його в експлуатацію.

2 ВИБІР МЕТОДІВ РІШЕННЯ

2.1 Клієнтська частина

Для розробки клієнтської частини веб-додатку мною було використано відкритий JavaScript фреймворк React.js, розроблений

компанією Facebook. Він використовується для побудови користувацьких інтерфейсів веб-додатків та мобільних додатків. Основною ідеєю цього фреймворку є створення компонентів, які відображають статичні частини інтерфейсу, і вони автоматично оновлюються при зміні даних. React.js є одним з найпопулярніших фреймворків для розробки користувацьких інтерфейсів у світі веб-розробки. За даними Stack Overflow trends[3], кількість питань пов'язаних з React.js становить 6,5%, що є найбільшим показником серед усіх фронтенд фреймворків(рис 2.1), що безпосередньо каже про те, що він є самим популярним фронтенд фреймворком.

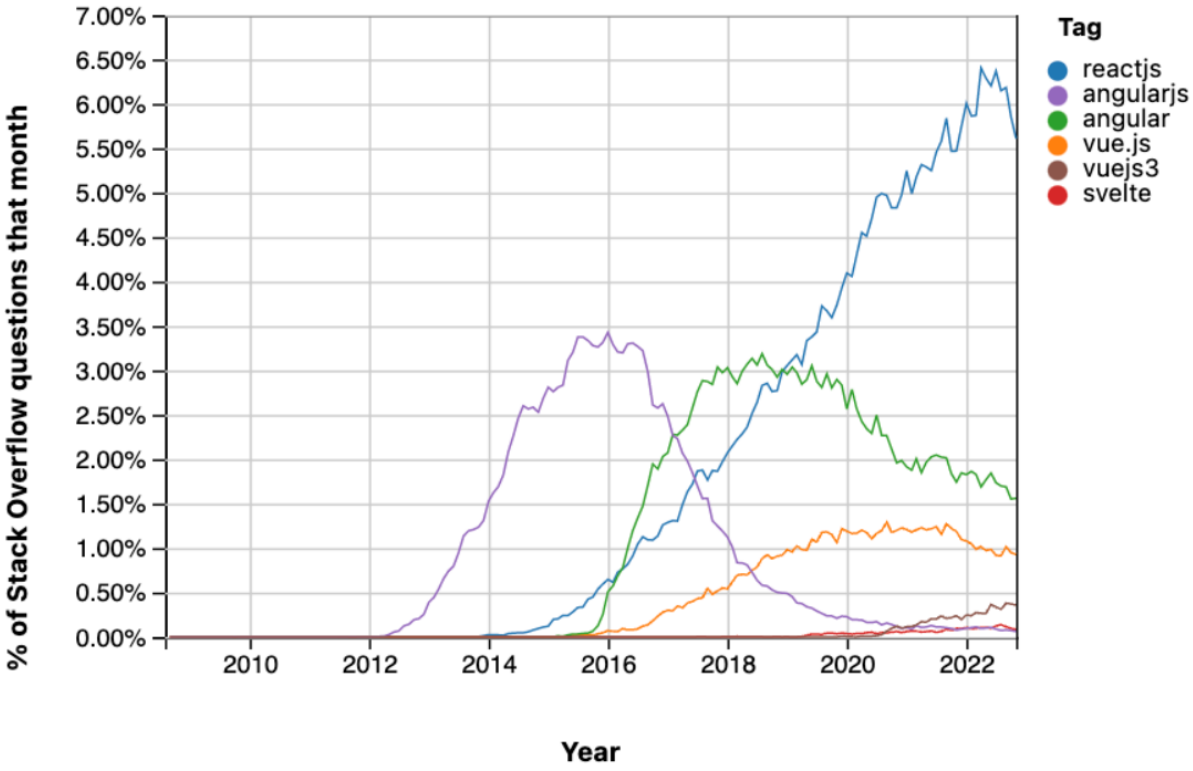

Рис. 2.1 - Кількість питань на StackOverflow серед фронтенд фреймворків

Розглянемо поверхнево кожен із них, визначимо їх плюси та мінуси, порівняємо їх між собою.

Svelte - це модерний фреймворк для розробки користувацьких інтерфейсів, який відрізняється від інших фреймворків, таких як React або Angular, тим, що працює на етапі компіляції, перетворюючи код

компонентів у високоефективний ванільний JavaScript під час збірки[4]. Основні переваги:

- генерує чистий та оптимізований JavaScript код без використання віртуального DOM, що дозволяє створювати швидкі та ефективні додатки;
- синтаксис досить легкий для вивчення, що робить його доступним навіть для початківців;
- компоненти в Svelte мають менше ваги порівняно з іншими фреймворками, що дозволяє створювати компактний та простий код;
- без використання віртуального DOM та інших накладних витрат, додатки мають високу швидкодію та реактивність.

Основні недоліки:

- у порівнянні з React або Angular, спільнота Svelte набагато менша, тому може бути складніше знайти допомогу чи розв'язання проблем;
- оскільки Svelte є новішим фреймворком, він може мати меншу кількість інтеграцій з іншими бібліотеками чи інструментами порівняно з більш популярними аналогами;
- у Svelte можуть бути менше функцій або розширень, порівняно з React або Angular, що може ускладнювати реалізацію певних завдань.

Vue.js - це прогресивний фреймворк JavaScript для розробки користувацьких інтерфейсів та односторінкових додатків. Він відрізняється легкістю використання та гнучкістю, що дозволяє розробникам швидко створювати складні веб-додатки. [5]

Основні переваги:

- має простий та легкий для вивчення синтаксис, що робить його доступним для початківців та досвідчених розробників;
- дозволяє використовувати як весь його функціонал, так і лише окремі його частини, що сприяє реалізації різних архітектурних рішень;

- має оптимальну продуктивність завдяки ефективному управлінню віртуальним DOM та швидкому оновленню компонентів.

Основні недоліки:

- у порівнянні з React або Angular, Vue.js має меншу популярність, що може ускладнювати пошук розробників або знаходження готових рішень;
- у деяких аспектах може мати обмежені можливості порівняно з іншими фреймворками, що може призвести до необхідності використання сторонніх бібліотек або розширень.

AngularJS - це відкритий фреймворк JavaScript, розроблений командою Google, призначений для створення односторінкових додатків. [6]

Основні переваги:

- має вбудований набір інструментів для створення односторінкових додатків, включаючи систему маршрутизації, сервіси, директиви та інші;
- підтримує модульну архітектуру, що дозволяє розбивати додаток на невеликі компоненти для полегшення управління та розвитку;
- використовує двостороннє зв'язування даних, що дозволяє автоматично оновлювати відображення даних при їхній зміні;
- оптимізований для створення односторінкових додатків (SPA), що забезпечує швидку та плавну навігацію.

Основні недоліки:

- може виявитися складним для вивчення, особливо для початківців, через велику кількість концепцій та понять;
- може генерувати великі об'єми коду, що може вплинути на продуктивність та швидкодію додатку;
- для роботи з AngularJS може знадобитися більше ресурсів та досвіду у порівнянні з іншими фреймворками.

React.js - це відкритий JavaScript фреймворк, розроблений компанією Facebook, призначений для розробки користувацьких інтерфейсів веб-додатків та односторінкових додатків.[7]

Основні переваги:

- використовує віртуальний DOM для ефективного оновлення сторінок, що робить його швидким та ефективним у порівнянні з іншими фреймворками;
- розробка за допомогою React базується на створенні та використанні компонентів, що робить код більш чистим та повторно використовуваним;
- підтримує створення односторінкових додатків, що забезпечує плавну та швидку навігацію без перезавантаження сторінок;
- використовує реактивний підхід, що дозволяє автоматично оновлювати вміст сторінок при зміні даних без необхідності вручному втручанні;
- має велике та активне співтовариство розробників, що забезпечує підтримку, оновлення та велику кількість ресурсів для навчання та розвитку.

Основні недоліки:

- для початку роботи з React може знадобитися деякий час для вивчення його концепцій та особливостей;
- для реалізації певних функцій може знадобитися використання додаткових бібліотек, що може збільшити обсяг проекту та складність його розробки.

Порівнявши вищеперелічені фреймворки, можна зробити висновок, що кожен з них має свої переваги та недоліки. Але я вирішив використати саме React.js, тому що він має компонентну структуру, що є дуже зручним підходом і добре підходить для мого проекту, має велике комьюніті, що

допоможе вирішити несподівані проблеми, також я вже маю певний досвід роботи з цим інструментом.

У якості UI бібліотеки для основного додатку я вирішив обрати Chakra UI, для адміністративної панелі - Material UI. Серед UI бібліотек самими популярними можна виділити Material UI, Ant Design, Chakra UI. Далі я напиши про кожну з них, визначу їх переваги на недоліки.

Material UI - це популярна бібліотека компонентів для React, яка надає інструменти для швидкої та простої розробки сучасних користувацьких інтерфейсів. Вона базується на дизайні Material Design від Google і надає широкий набір готових компонентів і стилів, які можна легко використовувати для створення естетично зручних та функціональних додатків.[8]

Основні переваги:

- компоненти Material UI мають чистий та сучасний вигляд, що дозволяє створювати привабливі та сучасні інтерфейси;
- має велику та активну спільноту розробників, яка підтримує розвиток бібліотеки та надає підтримку користувачам.

Основні недоліки:

- через велику кількість готових компонентів та стилів, обсяг використаного коду може бути великим, що може вплинути на завантаження та продуктивність додатку;
- у деяких випадках стилі компонентів Material UI можуть не повністю відповідати дизайну вашого проекту, що може потребувати додаткового налаштування.

Chakra UI - це сучасна бібліотека компонентів для React та React Native, яка дозволяє легко та швидко створювати стильні та функціональні користувацькі інтерфейси.[9]

Основні переваги:

- бібліотека має простий та зрозумілий API, що дозволяє легко використовувати її навіть початківцям у розробці;
- компоненти Chakra UI мають сучасний та естетичний дизайн, що дозволяє швидко створювати привабливі користувацькі інтерфейси;
- бібліотека підтримує рендеринг на сервері (SSR) та темний режим, що робить її універсальним рішенням для будь-якого проекту.

Основні недоліки:

- у порівнянні з іншими бібліотеками, Chakra UI може мати обмежений набір компонентів, що може вимагати використання сторонніх бібліотек або налаштування власних компонентів;
- у деяких випадках можуть виникати конфлікти стилів між компонентами бібліотеки та іншими стилізованими елементами додатку.

Ant Design - це повноцінна бібліотека компонентів для React та React Native, яка пропонує широкий набір готових компонентів і стилів для створення сучасних та ефективних користувацьких інтерфейсів.[10]

Основні переваги:

- бібліотека містить великий набір готових компонентів і функціональності, що дозволяє швидко створювати складні та функціональні додатки;
- пропонує великий вибір компонентів з різних категорій, що дозволяє вибирати найбільш підходящі рішення для конкретних задач.

Основні недоліки:

- через велику кількість компонентів і стилів, використання Ant Design може призвести до збільшення обсягу коду проекту;
- дизайн компонентів Ant Design може бути специфічним і не відповідати всім потребам або вимогам проекту.

Проаналізувавши найпопулярніші рішення, можна сказати, що обидві бібліотеки, Material UI і Chakra UI, підходять для вирішення поставлених

задач. Однак, їхні особливості та підходи можуть різнитися, що робить їх більш або менш підходящими для певного проекту.

Я вирішив використати Material UI для написання додатку для керування веб-системою саме тому, що ця бібліотека має широкий набір готових компонентів, які спеціально призначені для адміністративних інтерфейсів. Ці компоненти мають розширені можливості для керування даними, фільтрації, сортування і інші функції, які важливі саме для управління даними. Крім того, Material UI має стильний та сучасний дизайн, який може підійти для професійного вигляду адміністративної системи.

З іншого боку, я обрав Chakra UI для написання основного додатку архітектурного бюро через його гнучкість налаштування, лаконічність та швидкість. Ця бібліотека дозволяє швидко створювати і налаштовувати компоненти, що може бути особливо корисним у випадках, коли потрібно швидко реагувати на зміни вимог до додатку або експериментувати з дизайном. Крім того, Chakra UI має простий та зрозумілий API, який дозволяє швидко і ефективно працювати з компонентами.

Для управління станом на клієнтській частині додатку я використав бібліотеку Redux. Вона дозволяє зберігати стан додатку в одному централізованому місці та управляти змінами цього стану за допомогою чистих функцій, що називаються редукторами.

Основні задачі, які виконує Redux у цьому проекті:

- Управління станом додатку: Redux дозволяє зберігати всі дані додатку в централізованому місці, що робить керування станом простішим та прозорішим.
- Передача даних між компонентами: За допомогою Redux, дані можна передавати з одного компонента в інший безпосередньо, а замість цього вони зберігаються в Redux store і доступні для всіх компонентів.

Для демонстрації статистичних даних у вигляді графіків та діаграм я використав бібліотеку Nivo. Ця бібліотека - це набір гнучких компонентів

для візуалізації даних у веб-додатках. Вона дозволяє легко створювати різноманітні графіки, діаграми та інші візуальні елементи для відображення різних типів даних.

Основні переваги бібліотеки Nivo:

- Різноманітність візуалізацій: Nivo має широкий спектр компонентів для створення різноманітних візуалізацій, включаючи графіки ліній, стовпців, кругові діаграми, картограми, теплові карти та багато інших.
- Гнучкість налаштування: Кожен компонент Nivo має велику кількість параметрів налаштування, що дозволяє легко налаштувати вигляд та поведінку візуалізацій згідно з потребами проекту.
- Висока продуктивність та швидкість: Nivo побудована на основі бібліотеки D3.js та React, що забезпечує високу продуктивність візуалізацій та швидке рендеринг навіть при обробці великих обсягів даних.
- Простота використання: Nivo має простий та зрозумілий API, що робить її легкою у використанні.

2.1 Серверна частина

У цьому проекті я написав спільний бекенд як для основного додатку, так і для додатку керування системою. Для цього я обрав javascript фреймворк Nest.js, який дуже стрімко набирає популярність. Він займає 5 місце серед усіх бекенд фреймворків, та не так давно перегнав свого основного конкурента серед js фреймворків - ExpressJS, який довгі роки буд попереду.(рис 2.2)

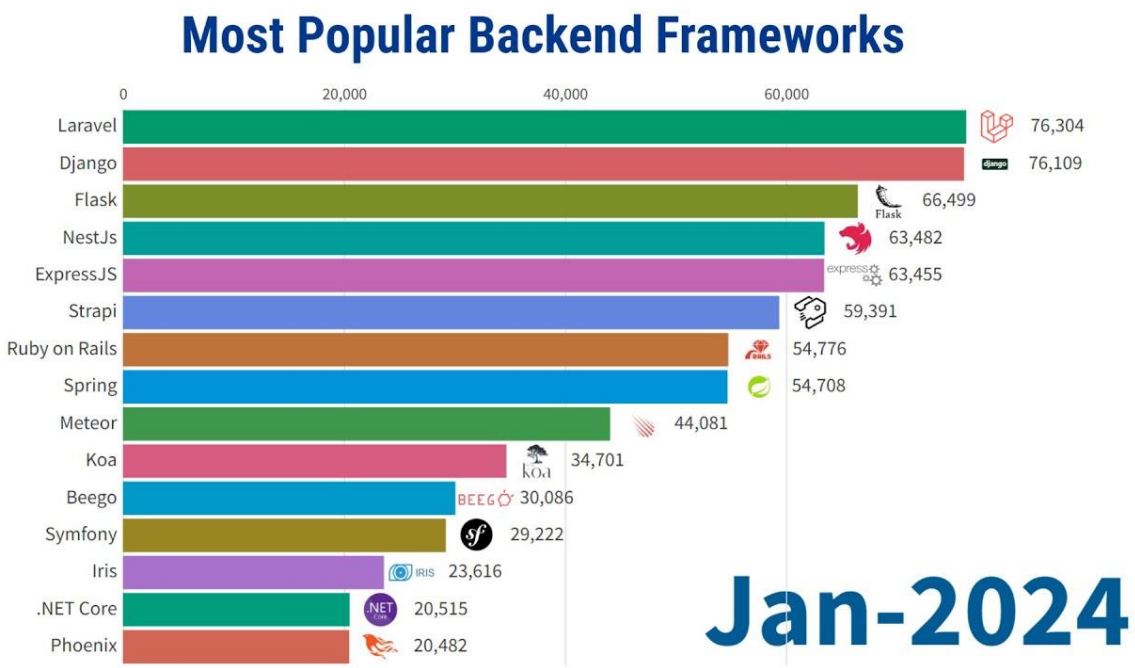

Рис. 2.1 - Кількість публічних репозитаріїв на Github серед бекенд фреймворків

Взагалі я використовую саме javascript фреймворк для бекенду через те, що це дозволяє мені писати і фронтенд і бекенд однією мовою програмування, що є дуже зручним. Також я раніше відкрив для себе Nest.js, який є дуже потужним інструментом для написання бекенду. Нижче я трохи напишу про обидва найпопулярніші javascript бекенд фреймворки, підкреслю їх плюси та мінуси.

ExpressJS - це популярний фреймворк для розробки серверної частини додатків, який відомий своєю простотою та мінімалістичним підходом до розробки веб-серверів. Також він має велику та активну спільноту розробників, що робить його дуже популярним і забезпечує доступ до безлічі ресурсів та бібліотек.[11]

Основні переваги:

- простота використання та навчання;
- велика кількість готових рішень та бібліотек;
- гнучкість у налаштуванні серверної частини.

Основні недоліки:

- відсутність вбудованих механізмів для підтримки структури проекту та організації коду;
- можливість виникнення складних рішень при роботі зі складними асинхронними операціями.

NestJs - це платформа для створення ефективних, масштабованих серверних програм Node.js. Основною його особливістю є модульність, тобто він пропонує модульну архітектуру, яка дозволяє організувати код у вигляді окремих модулів зі своїми контролерами, сервісами та провайдерами, що дозволяє підтримувати високий рівень структуризації та організації коду проекту. Також NestJS підтримує TypeScript з коробки, що дозволяє писати більш безпечний та розумінний код з підтримкою типів даних. [12]

Основні переваги:

- високий рівень структуризації коду та організації проекту;
- вбудована підтримка TypeScript;
- модульність та гнучкість у виборі технологій та інструментів.

Основні недоліки:

- потребує більшого часу для навчання та розуміння особливостей фреймворку.

Загалом, якщо порівнювати ці два інструменти, можна прийти к висновку, що Express.js може бути кращим вибором для простих або менш складних проектів, де пріоритетом є швидкість розробки та мінімалізм, в той час як Nest.js надає більшу структурованість та гнучкість для більш складних та масштабованих проектів. Особисто мені більше подобається модульна архітектура NestJS, в порівнянні з його головним конкурентом, цей фреймворк пропонує більш структурований підхід для написання коду, що робить його фаворитом.

У якості бази даних я вирішив використати NoSQL базу даних MongoDB. MongoDB, відрізняються від традиційних реляційних баз даних тим, що дозволяє зберігати дані у вигляді JSON-подібних документів, що робить роботу з даними більш інтуїтивною та гнучкою. Це дає можливість легко вставляти, оновлювати та видаляти дані без необхідності виконання складних операцій JOIN. Також NoSQL бази даних дозволяють легко масштабувати додаток горизонтально, додаючи нові сервери для розподіленого зберігання даних та обробки навантаження.[13] База даних MongoDB входить у четвірку найпопулярніших баз даних та є самою популярною NoSQL базою даних згідно порталу StackOverflow(рис 2.3)

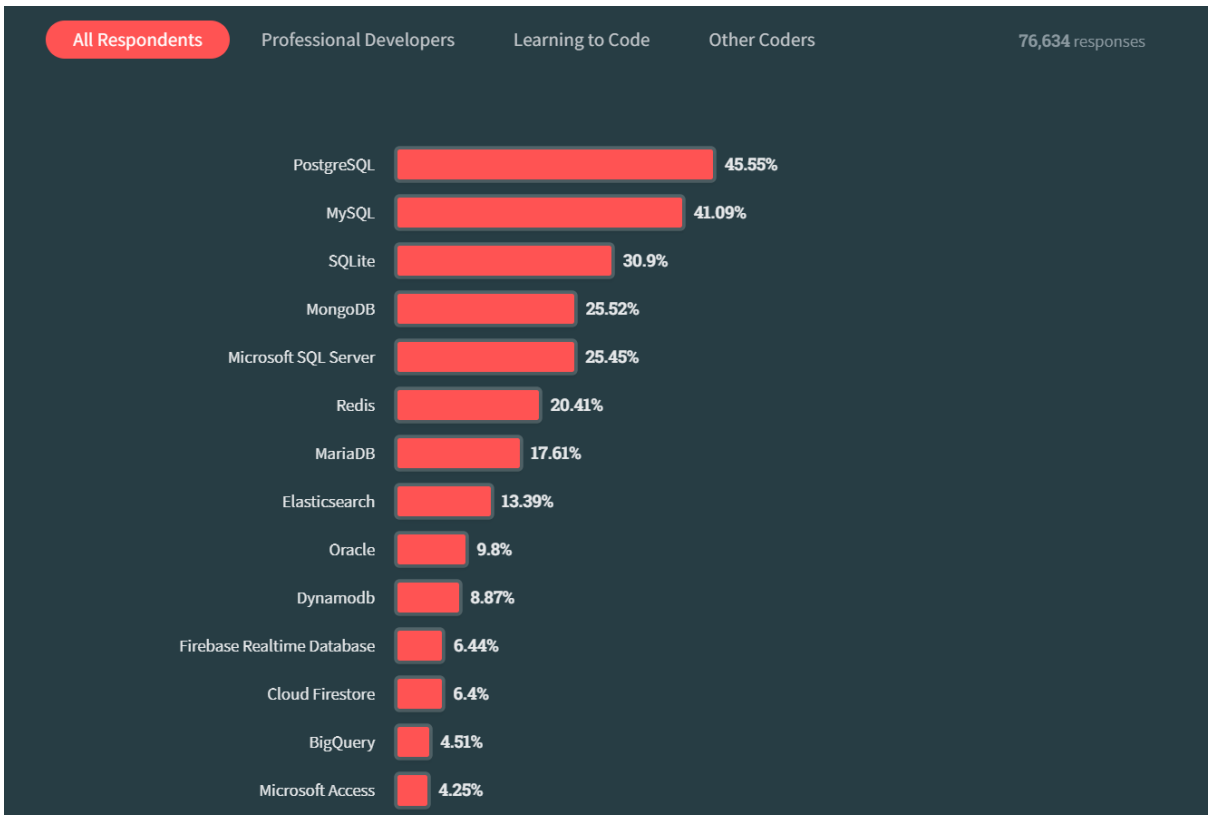

Рис. 2.1 - Рейтинг популярності баз даних серед розробників на StackOverflow

Отже, MongoDB є доцільним вибором для цього проекту завдяки своїй гнучкості, швидкості розробки, масштабованості, що допомогло мені

створити ефективну інформаційну систему для архітектурно-планувального бюро.

Для зв’язку клієнтської та серверної частини я використав інструмент вбудований в бібліотеку вже згадану вище - Redux, а саме RTK Query. Він використовується для створення сервісів для запитів на сервер. Він дозволяє створювати зручні інтерфейси, завдяки яким можна відстежувати стан запитів і при цьому не створювати багато одноманітного коду.

3 ПРАКТИЧНА РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Клієнтська частина основного додатку

Додаток складається з різних розділів, які представляють собою сторінки додатку, розберем кожний з них:

Розділ 1: Головна сторінка. Головна сторінка додатку складається з навігаційної панелі, банеру, списку послуг архітектурного-бюро, каталогу проектів, футеру.(рис 3.1-3.4)

Рис. 3.1 - Навігаційна панель

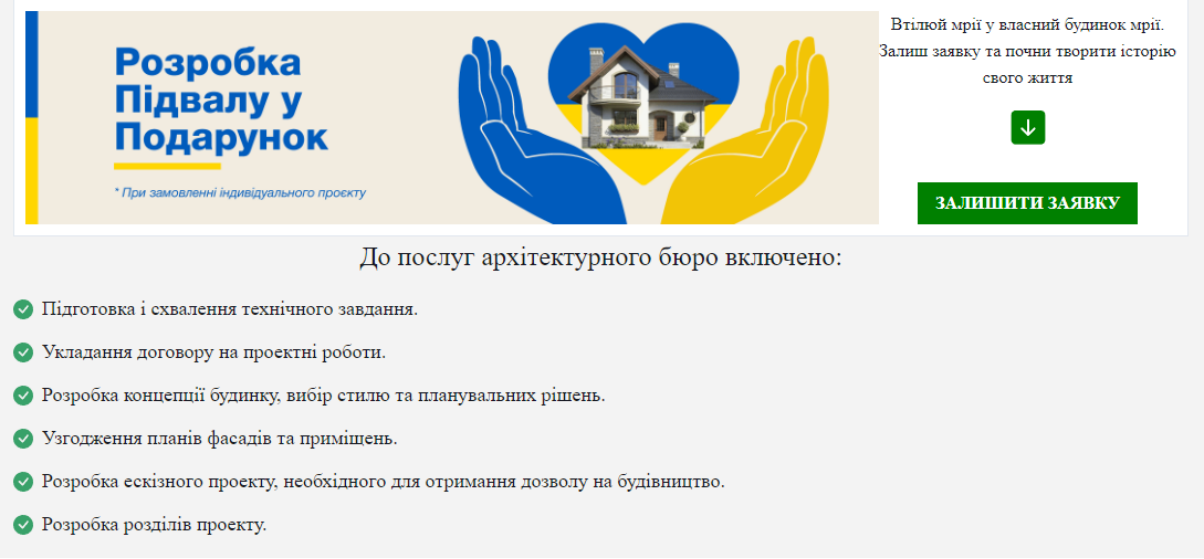

Рис. 3.2 - Банер та список послуг

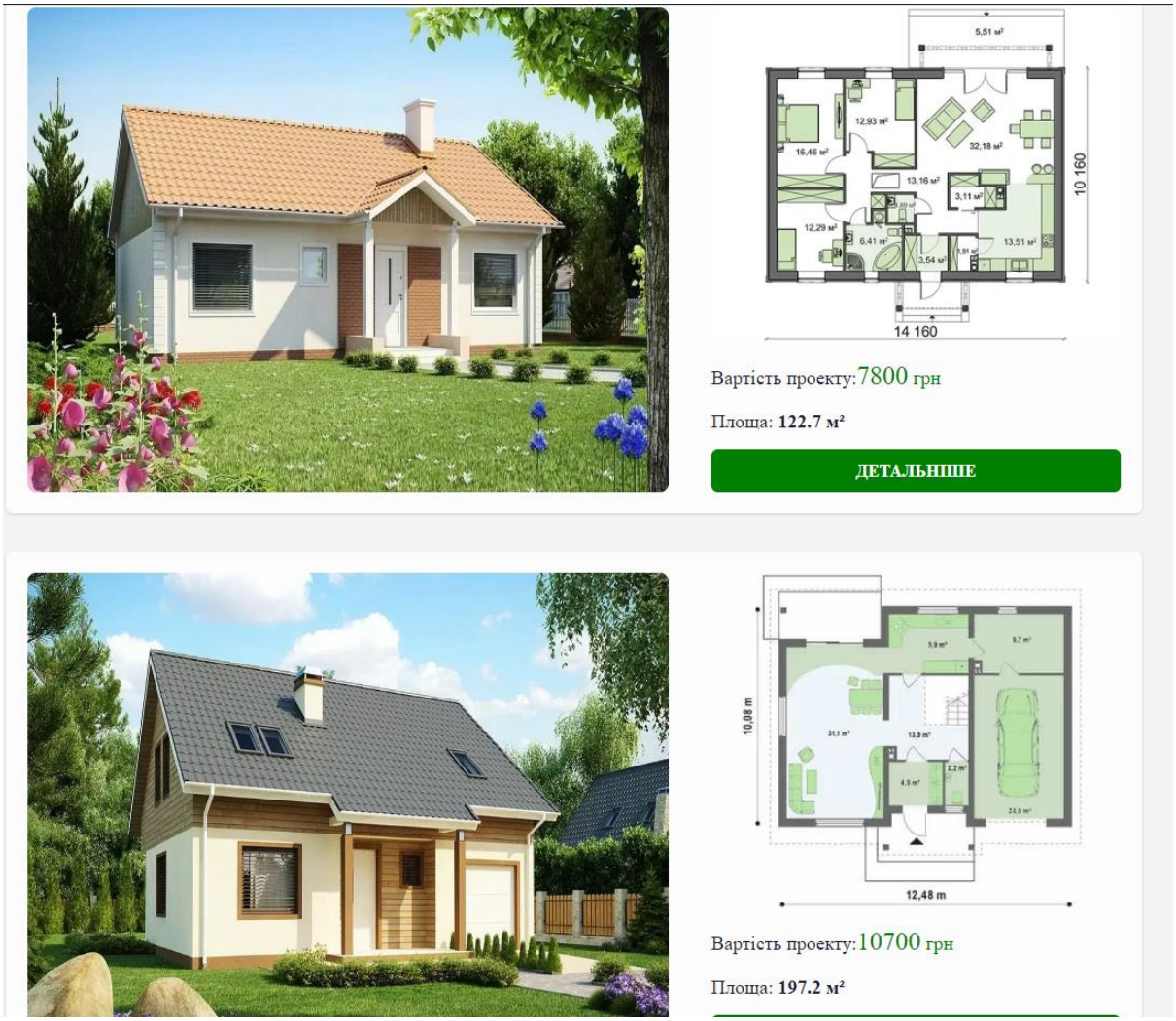

Рис. 3.3 - Каталог проектів

Рис. 3.4 - Футер

Розділ 2: Сторінка для подачі заявки на складання проекту. Після натискання на кнопку “Залишити заявку” на головній сторінці, користувача перекидає на сторінку для подачі заявки, в якій він може залишити заявку на складання свого власного проекту, а не с каталогу. В заявці потрібно вказати свої персональні дані, обрати площу будівлі, кількість поверхів, написати коментар. Також у формі є калькулятор, який буде автоматично обчислювати вартість проекту, в залежності від площі будинку. Сторінка складається з банеру з зображенням креслення одного з проектів та перевагами бюро, форми для заповнення заявки.(рис 3.5-3.6)

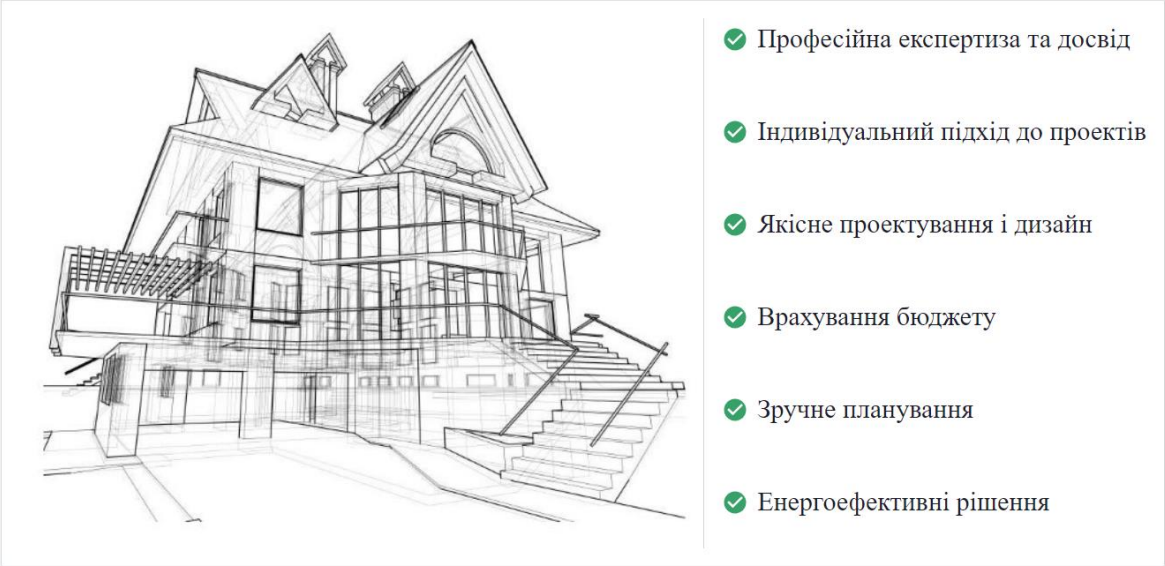

Рис. 3.5 - Банер та список переваг бюро

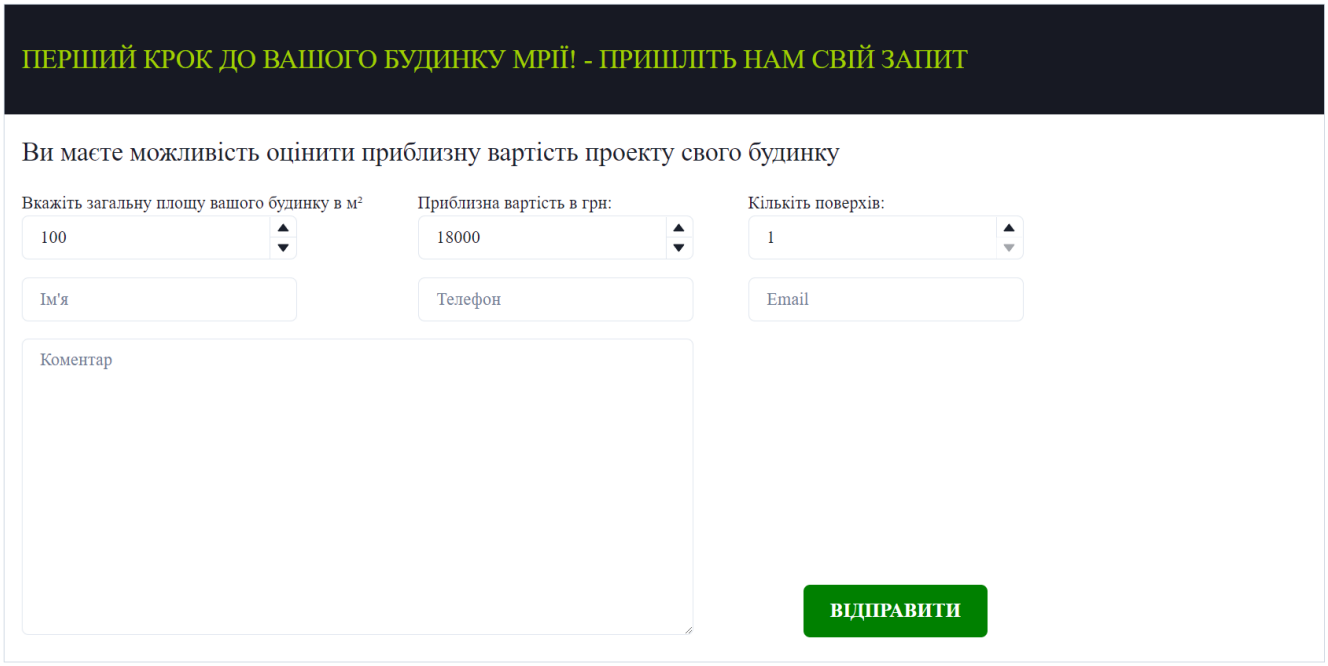

Рис. 3.6 - Форма для написання заявки

Розділ 3: Сторінка конкретного проекту з каталогу. При натисканні на кнопку “Детальніше” на будь-якому будинку з каталогу, відкривається його детальна інформація, в якій зазначено основну характеристику будівлі, а саме: кубатура, загальна площа, житлова площа, кількість поверхів, висота будинку, зображення екстер’єра будинку, зображення креслення, ціну та кнопку “Заказати”. (рис 3.7)

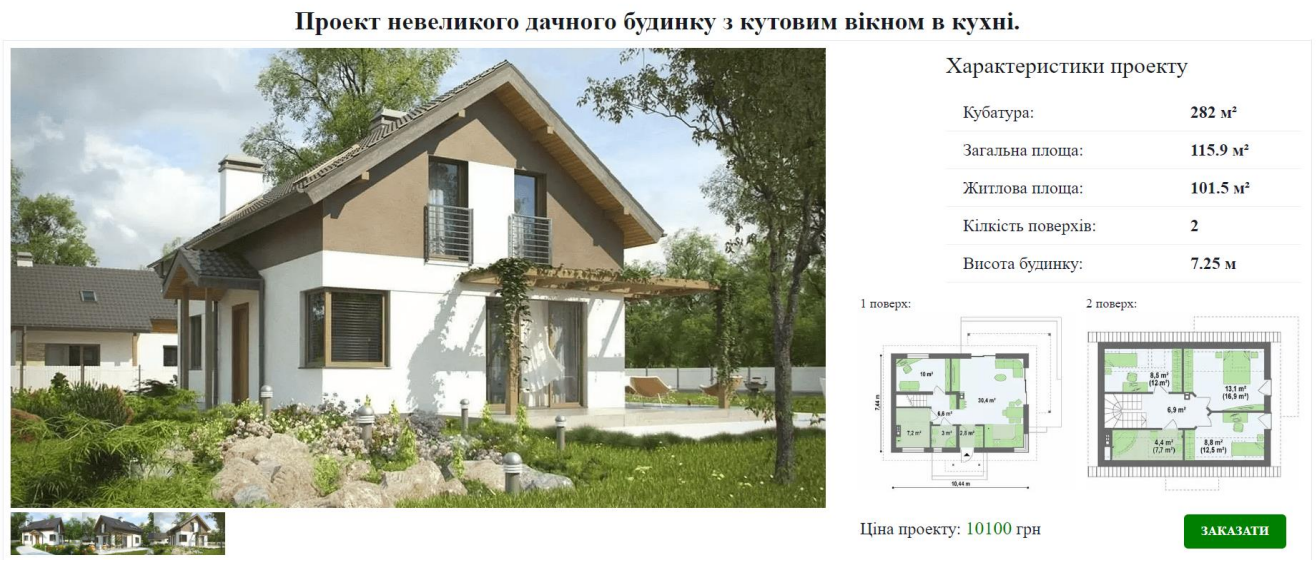

Рис. 3.7 - Детальна інформація будинку

Розділ 3: Сторінка заповнення заказу. Після натискання на кнопку “Заказати” на сторінці певного будинку, відкривається сторінка для заповнення заказу, на якій користувач може заповнити відповідні поля для відправки заявки на покупку певного проекту. А саме, користувач повинен написати свої персональні дані, обрати форму оплати, спосіб доставки, написати коментар та натиснути кнопку “Відправити”, після чого ця заявка відправиться у базу даних, та буде доступна для просмотру в адмін-додатку.(рис 3.8)

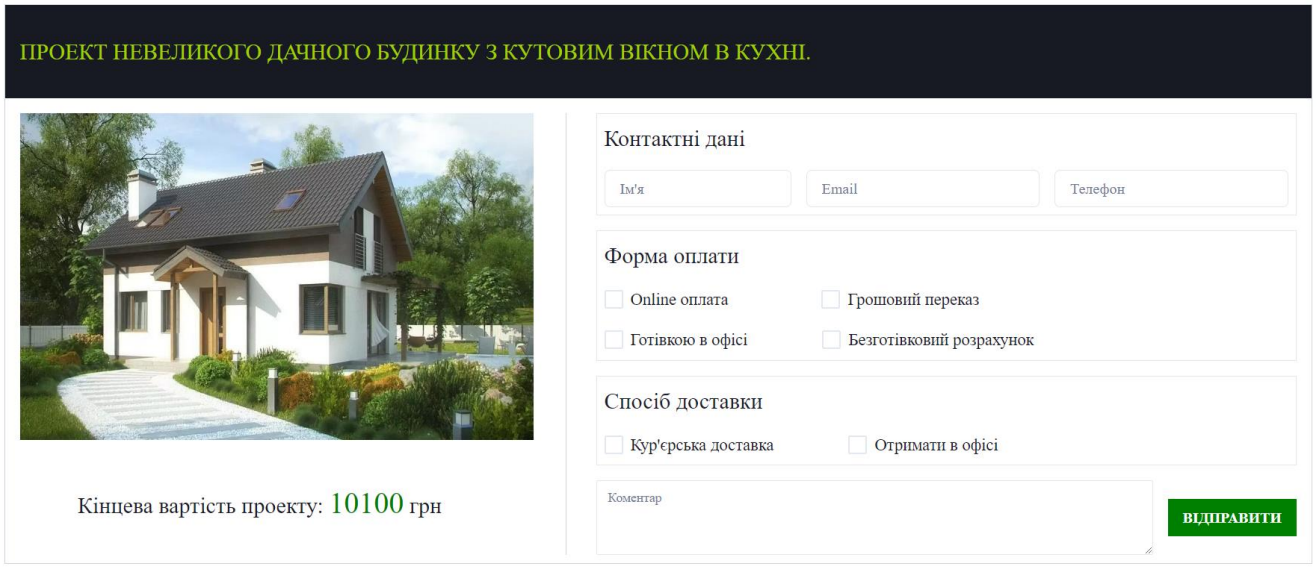

Рис. 3.8 - Сторінка заповнення заказу

3.2 Серверна частина основного додатку

Серверна частина основного додатку складається з 3 основних сутностей: houses, orders, requests. Кожна з цих сутностей відображає основні елементи системи, та має певні атрибути, які я перечислю нижче:

1. **Houses -** ця сутність представляє проекти на будівлі, які доступні в каталозі. Має такі атрибути:
    - name(string) - назва проекту.
    - price(number) - ціна проекту.
    - preview_images(string[]) - зображення будинків.
    - project_images(string[]) - зображення креслень будинків.
    - floor_count(number) - кількість поверхів будівлі.
    - building_footprint(number) - кубатура будівлі.
    - house_height(number) - висота будівлі.
    - general_square(number) - загальна площа будинку.
    - living_square(number) - житлова площа будинку.
2. **Orders -** ця сутність відповідає за закази на будинки з каталогу. Має такі атрибути:
    - email(string) - імейл користувача, який заказує проект.
    - name(string) - ім’я користувача, який заказує проект.
    - phone(string) - номер телефону користувача, який заказує проект.
    - cash_method, cashless_payment, money_transfer, online_payment(boolean) - методи оплати.
    - courier_delivery, in_office(boolean) - методи доставки.
    - comment(string) - коментар користувача.
    - house(objectId) - id будинку, який заказує користувач.
3. **Requests -** ця сутність представляє заявки на складання проекту. Має

такі атрибути:

- email(string) - імейл користувача, який пише заявку.
- name(string) - ім’я користувача, який пише заявку.
- phone(string) - номер телефону користувача, який пише заявку.
- square(number) - площа будівлі.
- floor_count(number) - кількість поверхів.
- comment(string) - коментар користувача.

3.3 Клієнтська частина додатку для керування системою

Клієнтська частина додатку для керування системою складається з різних розділів, які виконують свої певні задачі, розберем кожний з них:

Розділ 1: Закази на будинки. В цьому розділі міститься список заказів на будинки з каталогу, які зробили користувачі. Кожен елемент списку має особисті дані користувача, можливість подивитися будинок, на який зроблено заказ та видалити заказ. Список заказів зроблений за допомогою компоненту “DataGrid” з бібліотеки Material UI.(рис 3.9)

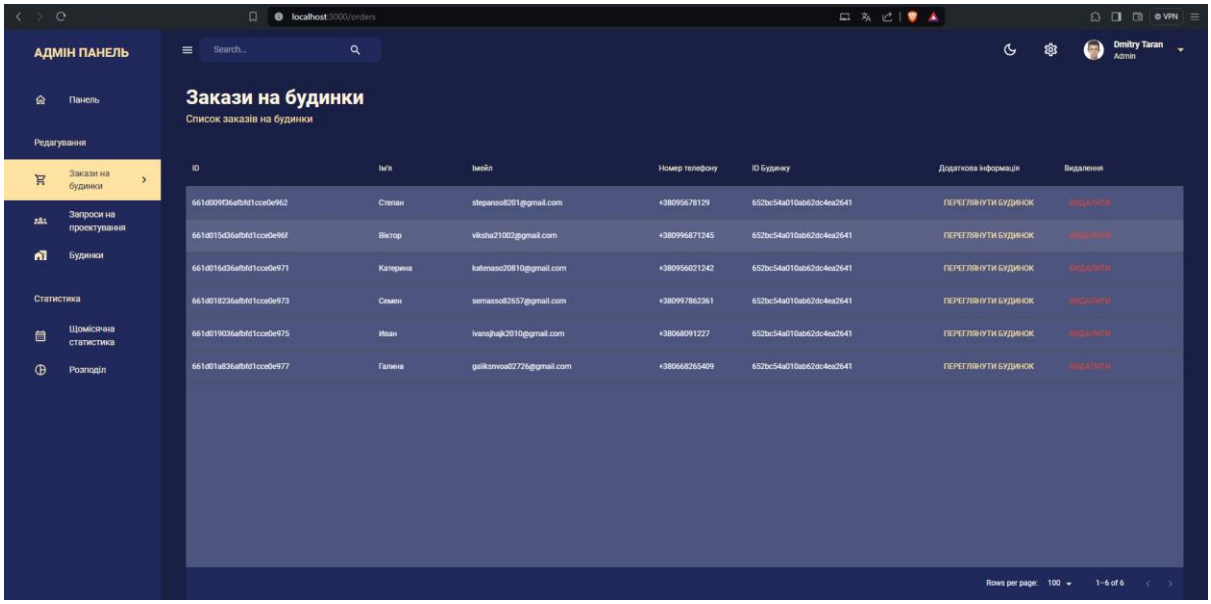

Рисунок 3.9 Закази на будинки

Розділ 2: Запроси на проектування. Цей розділ містить список заявок на реалізацію проектів, які залишили користувачі на сайті. Кожен елемент

списку має особисту інформацію користувача, площу будівлі, коментар та кнопку для видалення заявки. Кожна заявка оформлена у вигляді Card-компоненту з бібліотеки Material UI.

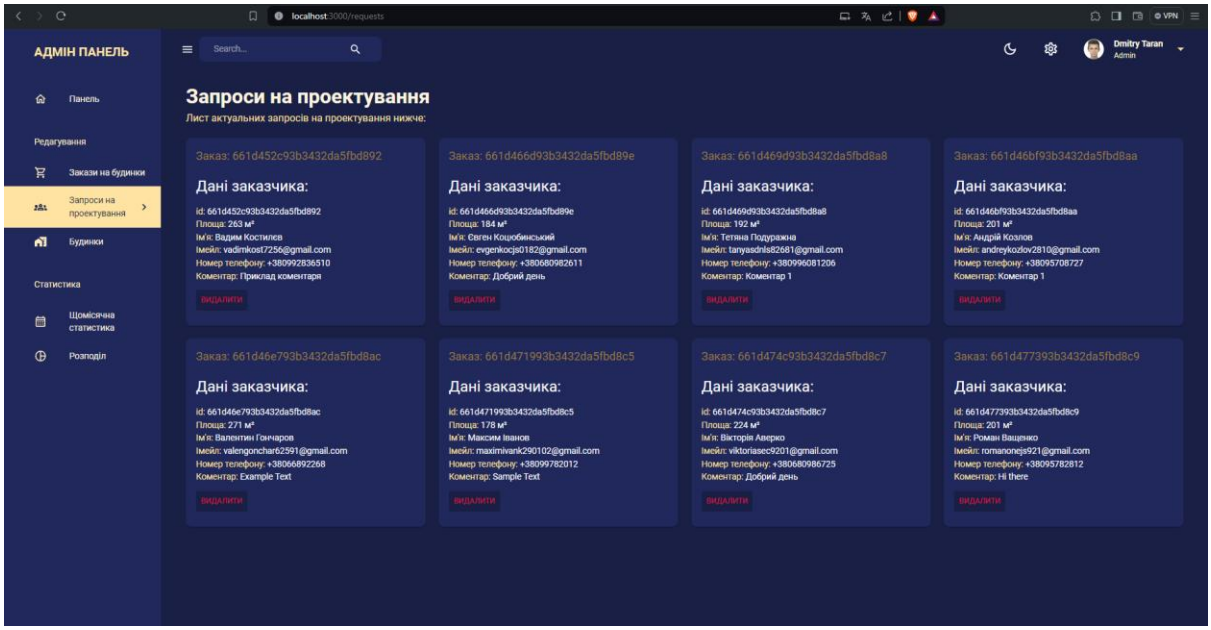

Рисунок 3.10 Запроси на проектування

Розділ 3: Список будинків. В цьому розділі містяться будинки, які відображаються у каталозі на основному сайті. Кожен елемент списку містить основну інформацію про будинок: ціну, кількість поверхів, кубатуру, висоту, загальну площу, житлову площу. Також є дві кнопки для редагування та видалення будинку. Кожний елемент списку зроблений за допомогою Card-компоненту з бібліотеки Material UI.

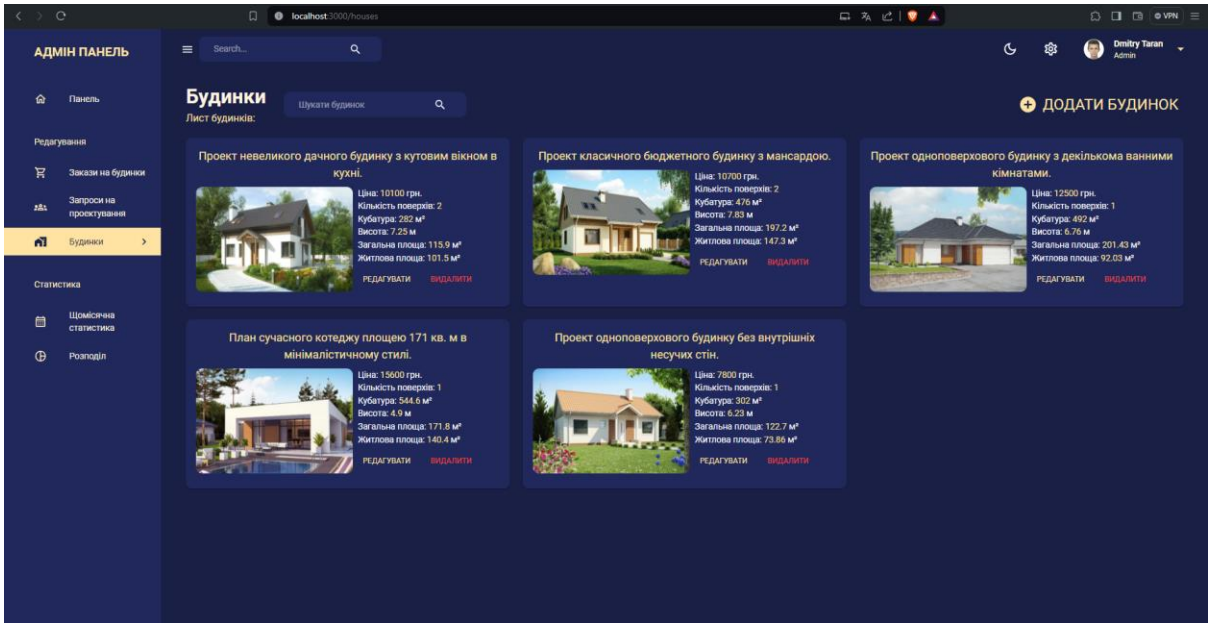

Рисунок 3.11 Список будинків

При натисканні на кнопку “Додати будинок” відкривається модальне вікно з відповідними полями для створення нового будинку, після заповнення котрих слід натиснути кнопку відповідну кнопку знизу для додавання будинку у каталог, після чого на сервер відправиться запит, та новий будинок відобразиться одразу на сайті завдяки бібліотеці Redux.

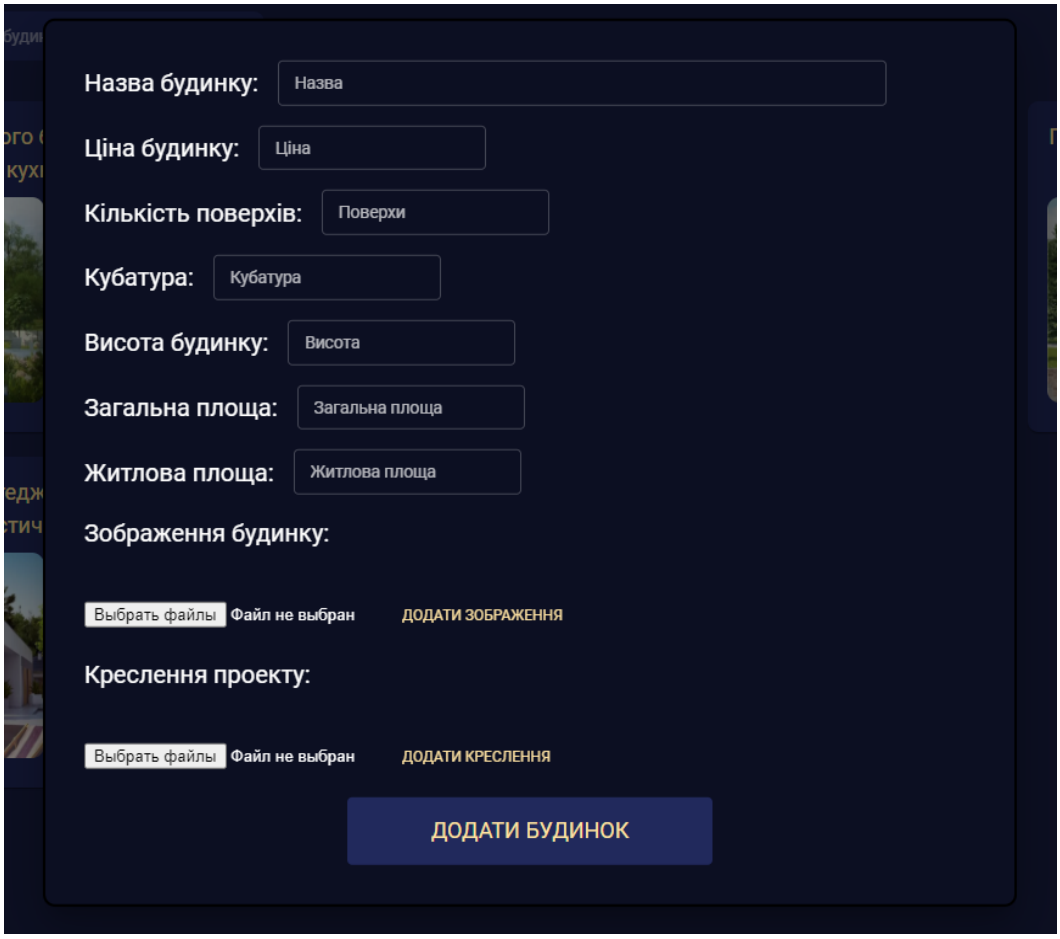

Рисунок 3.12 Модальне вікно для створення будинку

Після натискання на кнопку “Редагувати” на будинку, відкривається модальне вікно для редагування вже існуючого будинку в каталозі. Вікно складається з полів, які вже заповнені існуючими значеннями певного будинку, які треба замінити.

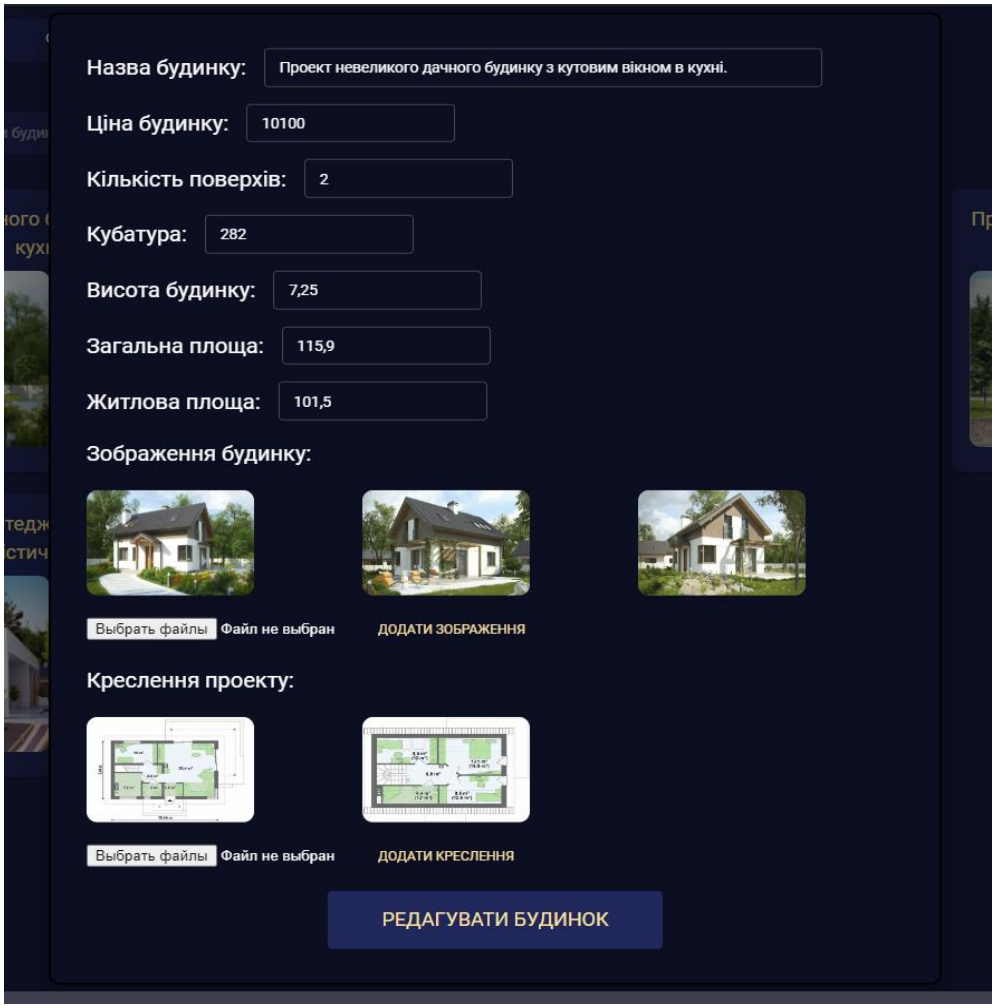

Рисунок 3.13 Модальне вікно для редагування будинку

Розділ 4: Щомісячна статистика. Цей розділ відповідає за відображення кількості заказів на будинки та запросів на проектування по місяцях за певний рік у виді графіку. Графік відображається завдяки компоненту ResponsiveLine з бібліотеки Nivo.

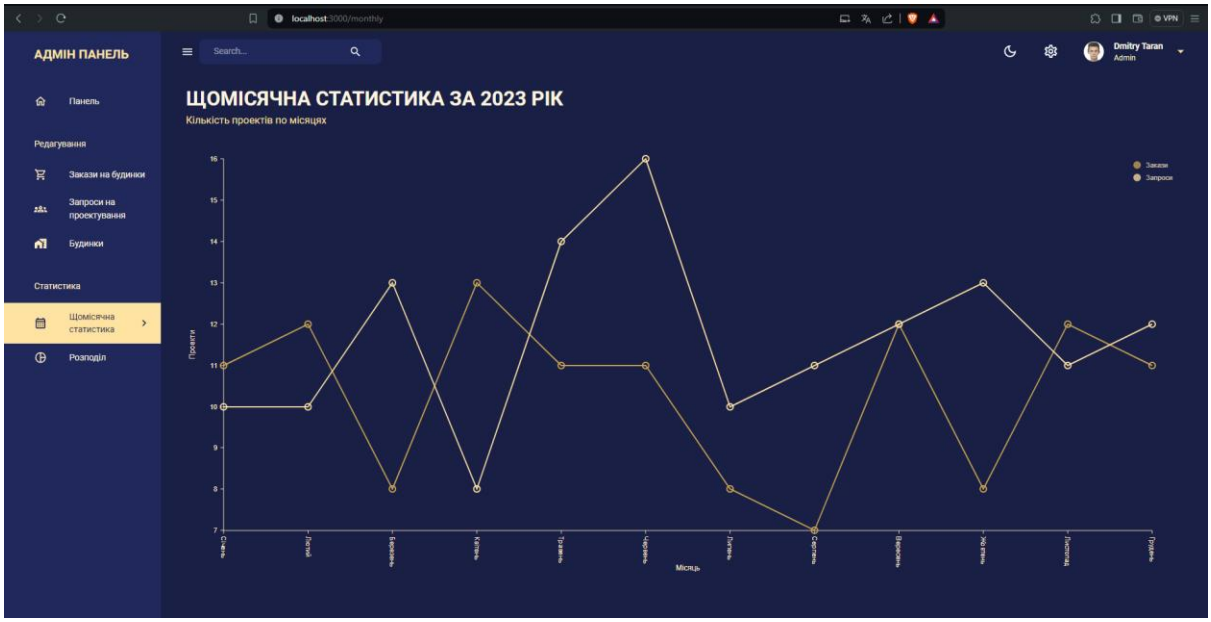

Рисунок 3.14 Щомісячна статистика

Розділ 5: Розподіл проектів за категоріями. В цьому розділі відображається розподіл виконаних проектів за категоріями, а саме: капітальні ремонти, реконструкції, поточні ремонти, нові будівництва. Реалізовано це за допомогою кругової діаграми, а саме компоненту ResponsivePie з бібліотеки Nivo.

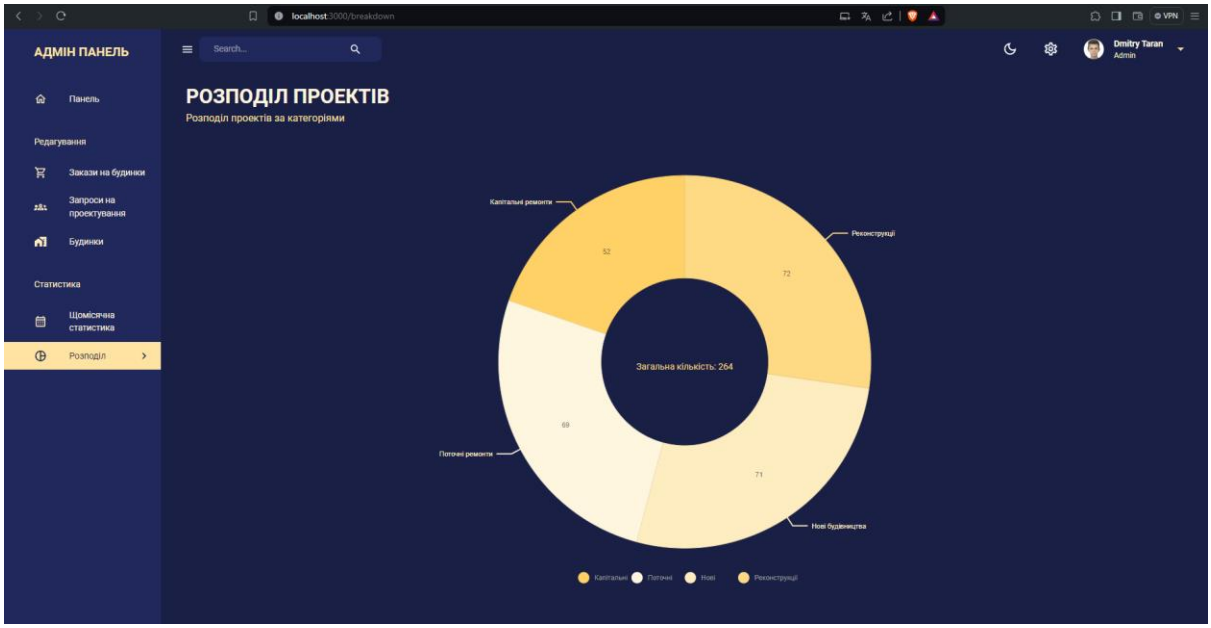

Рисунок 3.15 Розподіл проектів за категоріями

Розділ 6: Головна панель. Цей розділ демонструє загальну статистику поточного року, також виводить компоненти з інших розділів у стислому

форматі. Взагалі цей розділ містить: кількість клієнтів та заказів у поточному році, кількість капітальних ремонтів у поточному році, кількість заказів на будинки з каталогу та запросів на проектування у поточному місяці, графік кількості заказів по місяцях за попередній рік, список заказів на будинки та кругова діаграма розподілу проектів по категоріях.

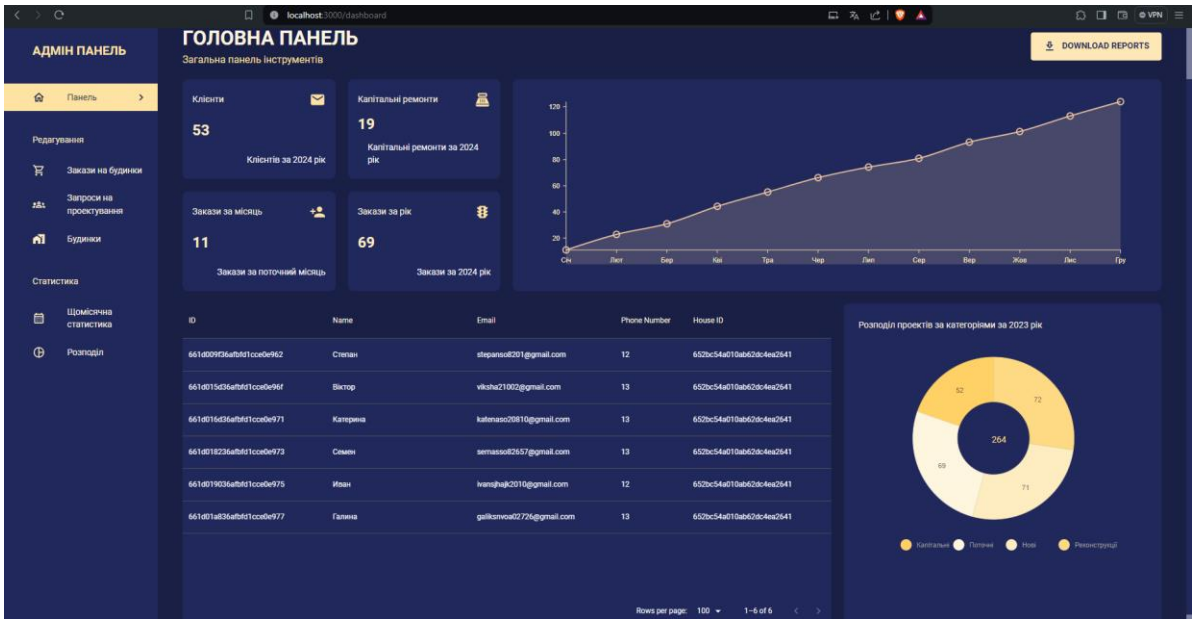

Рисунок 3.16 Головна панель

3.4 Серверна частина додатку для керування системою

Як я вже писав раніше, у цьому проекті серверний додаток є спільним, тому що створювати окремий немає сенсу, через те, що більшість сутностей використовуються як в основному додатку, так і в додатку для керування системою, також серверна архітектура написана за допомогою фреймворку Nest.js добре масштабується. Але все ж таки серверний додаток був трохи доповнений, була додана нова сутність Stats, яка містить у собі основні статистичні дані, має такі атрибути:

- totalCustomers(number) - кількість клієнтів за рік.
- overhauls(number) - кількість капітальних ремонтів.
- currentRepairs(number) - кількість поточних ремонтів.
- reconstructions(number) - кількість реконструкцій.

- newConstructions(number) - кількість нових будівництв.
- yearlySalesTotal(number) - сума проданих заказів та заявок на проектування.
- year(number) - рік, за який зібрана статистика.
- monthlyData(array) - массив об’єктів, який містить статистику за кожний місяць відповідного року, має такі атрибути:
    - month(string) - місяць, за який зібрана статистика.
    - requests(number) - кількість заявок на проектування.
    - orders(number) - кількість заказів на будинки.

Також у вже існуючому серверному додатку був доданий модуль Dashboard, у якому реалізована логіка виводу статистики як за певний рік, так і за певний місяць.

Посилання на код основного додатку - https://github.com/nevermind1727/university-architecture-bureau.

Посилання на код додатку для керування системою - https://github.com/nevermind1727/control-system-for-architecture-bureau.

ВИСНОВКИ

У цій роботі мною було створено інформаційну систему для

керування веб-платформою проектно-виробничого архітектурно-планувального бюро. Основною метою розробки було створення зручного та функціонального додатку, який дозволить користувачам здійснювати замовлення на складання архітектурних проектів, а також придбати вже існуючі проекти з каталогу.

Основна функціональність додатку включає в себе можливість створення та редагування архітектурних проектів, перегляд замовлень із каталогу, обробку заявок на складання проектів, а також аналіз статистики використання платформи. Додаток для адміністраторів дозволяє здійснювати управління системою, додавати нові проекти до каталогу, розглядати заявки користувачів та аналізувати статистику використання.

Результати дослідження показали, що створення інформаційної системи для керування веб-платформою проектно-виробничого архітектурно-планувального бюро є актуальним та необхідним. Застосування сучасних веб-технологій, таких як React.js, Nest.js, MongoDB, дозволило створити потужний та ефективний додаток, який відповідає вимогам сучасного ринку та забезпечує зручність користувачів та адміністраторів.

Отже, інформаційна система керування веб-платформою проектно-виробничого архітектурно-планувального бюро є важливим інструментом для оптимізації бізнес-процесів та підвищення ефективності роботи компанії. Її розробка відкриває широкі можливості для подальшого розвитку та розширення функціональності з урахуванням потреб користувачів та вимог ринку.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Кількість сайтів електронної комерції в усьому світі та їх

зростання з часом [Електронний ресурс] - Доступ до ресурсу: https://themeisle.com/blog/how-many-websites-are-there/.

2. Загальна статистика електронної комерції [Електронний ресурс] - Доступ до ресурсу: https://www.forbes.com/advisor/business/ecommerce-statistics/.

3. Stack Overflow Trends [Електронний ресурс] – Доступ до ресурсу: https://insights.stackoverflow.com/trends?tags=reactjs%2Cvue.js%2Cangular%2Csvelte%2Cangularjs%2Cvuejs3.

4. Документація фронтенд фреймворку Svelte [Електронний ресурс] – Доступу до ресурсу: https://svelte.dev/docs/introduction

5. Документація фронтенд фреймворку Vue [Електронний ресурс] – Доступ до ресурсу: https://vuejs.org/guide/introduction

6. Документація фронтенд фреймворку Angular [Електронний ресурс] – Доступ до ресурсу: https://angular.io/guide/what-is-angular.

7. Документація фронтенд фреймворку React [Електронний ресурс] – Доступ до ресурсу: https://legacy.reactjs.org/docs/getting-started.html

8. Документація UI бібліотеки Material UI [Електронний ресурс] – Доступ до ресурсу: https://mui.com/material-ui/.

9. Документація UI бібліотеки Chakra UI [Електронний ресурс] – Доступ до ресурсу: https://v2.chakra-ui.com/getting-started.

10. Документація UI бібліотеки Ant Design [Електронний ресурс] – Доступ до ресурсу: https://ant.design/docs/spec/introduce/.

11. Документація бекенд фреймворку ExpressJS [Електронний ресурс] - Доступ до ресурсу: https://expressjs.com/.

12. Документація бекенд фреймворку NestJS [Електронний ресурс] - Доступ до ресурсу: https://docs.nestjs.com/.

13. SQL vs NoSQL: Differences, Databases, and Decisions [Електронний ресурс]

- Доступ до ресурсу: https://ua.talend.com/resources/sql-vs-nosql/.