Міністерство освіти і науки України
НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ПРОЕКТУВАННЯ БАЗИ ДАНИХ ТА РОЗРОБКА АВТОМАТИЗОВАНОЇ ІНФОРМАЦІЙНОЇ СИСТЕМИ У СФЕРІ МАРКЕТИНГОВОЇ КОМУНІКАЦІЇ

Текстова частина до курсової роботи

за спеціальністю «Інженерія програмного забезпечення» 121

Керівник курсової роботи

Яремко С.А.

_____

(підпис)

«____» _____ 2021 р.

Виконала студентка 3-го курсу

Білорус К.С.

«____» _____ 2021 р.

Київ 2021

Міністерство освіти і науки України

НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ «КИЄВО-МОГИЛЯНСЬКА АКАДЕМІЯ»
Кафедра інформатики факультету інформатики

ЗАТВЕРДЖУЮ
Викладач кафедри інформатики,
канд. фіз-мат. наук, доц.
_____ Гороховський С.С.
(підпис)
«____»_____2021 р.

ІНДИВІДУАЛЬНЕ ЗАВДАННЯ
на курсову роботу

студентці Білорус Катерині Сергіївні факультету інформатики 3-го курсу
ТЕМА: Проектування бази даних та розробка автоматизованої інформаційної системи у сфері маркетингової комунікації

Зміст ТЧ до курсової роботи:

Індивідуальне завдання
Вступ
1 Аналіз предметної області
2 Теоретичні відомості
3 Проектування системи
4 Опис реалізації
Висновки
Список використаних джерел
Додатки

Дата видачі «___» _____ 2021 р. Керівник _____
(підпис)
Завдання отримав _____
(підпис)

Тема: **Проектування бази даних та розробка автоматизованої інформаційної системи у сфері маркетингової комунікації**

Календарний план виконання роботи:

№ п/п	Назва етапу курсової роботи	Термін виконання етапу	Примітка
1.	Отримання завдання на курсову роботу.	25.10.2020	
2.	Огляд літератури за темою роботи.	24.12.2020	
3.	Проведення анкетування потенційних користувачів.	12.01.2021	
3.	Аналіз проведеного опитування, виявлення потреб користувачів.	20.01.2021	
4.	Дослідження аналогічних систем.	27.01.2021	
5.	Опис функціональних вимог до системи.	05.02.2021	
6.	Аналіз отриманих результатів з керівником.	20.02.2021	
7.	Розробка ER-моделі.	02.03.2021	
8.	Розробка реляційної моделі та створення бази даних в системі PostgreSQL.	20.03.2021	
9.	Розробка UI/UX.	04.04.2021	
10.	Створення каркасу проекту, підключеннія фреймворків Express, Next.js.	25.04.2021	
11.	Реалізація базового функціоналу системи	1.05.2021	
12.	Створення презентації та написання доповіді.	8.05.2021	
13.	Подання роботи на кафедру для перевірки на плагіат.	16.05.2021	
14.	Захист курсової роботи.	травень 2021	

Студентка **Білорус К.С.**

Керівник **Яремко С.А.**

«_____»_____

Зміст

Перелік термінів та умовних позначень

API – визначений набір методів для програми

АІС – автоматизована інформаційна система

Інтерфейс – засіб для взаємодії користувача з програмою

Роутер – програмне забезпечення або пристрій для керування маршрутизацією

Токен – інформація, що використовується для ідентифікації

Фреймворк – допоміжна програма для полегшення розробки

ВСТУП

Актуальність теми

Зі зростанням попиту на послуги організації заходів, якість продуків, які постачають маркетингові компанії, підвищується. Це відбувається за рахунок покращення якості кінцевого продукту з метою заохочити потенційних замовників обрати послуги саме цієї фірми. Для надання якомога кращих умов використовується більше людськиї ресурсів для пошуку найкращих варіантів та умов для організації. З розвитком економіки, зростає якість сервісів обслуговування та туризму, а отже, і конкуренція серед компаній, які організовують заходи. Кількість ресторанів, готелів стає все більшою і знайти серед них найкращий варіант постає проблемою.

Більшість компаній забезпечують ефективний пошук конкурентоспроможних пропозицій за допомогою збільшення кількості працівників, проте організація внутрішньої інформаційної системи може вирішити цю проблему без залучення додаткових зусиль.

Автоматизована система для збереження та обробки даних про існуючі потенційні місця проведення, ресторани та інші елементи сучасного публічного заходу, дозволяє швидко звужувати пошук потрібних компонентів організації виходячи з потреб замовника. Створення актуальних пропозицій стає більш легким та швидким, а отримані результати можна легко перенести у затверджений проект.

Збереження інформації у структурованому електронному вигляді забезпечує зощадження часу працівників. Відсутність друкованих звітів зменшує витрати на канцелярію та сприяє екологічному стану

навколишнього середовища. Інформація, що зберігається в системі, є приватною і захищеною від несанкціонованого доступу.

Мета і завдання дослідження

Аналіз потреб користувачів та недоліків сучасних інформаційних систем корпоративного користування в області організації заходів. Розробка автоматизованої інформаційної системи для івент-агенства, призначена для збереження актуальної інформації про компоненти організації заходу та спільної роботи працівників над проектами. Основним завданням АІС є зменшення часових витрат на створення пропозицій проведення та полегшення комунікації між співробітниками.

Об’єкт дослідження

Розробка архітектури інформаційної системи на основі системи керування базами даних PostgreSQL.

Методи дослідження

Анкетування цільової аудиторії для виявлення недоліків сучасних ситем та опису вимог для розробки. Проектування системи на основі отриманих даних після анкетування. Розробка застосунка мовою Node.js на базі фреймворків Express та Next.js.

Практичне значення одержаних результатів

Розроблена система буде впроваджена в роботу івент-агенства для автоматизації організації заходів та полегшення комунікації між працівниками.

Джерела дослідження

Під час розробки було досліджено наукову літературу по маркетингу та організації заходів, проведення анкетувань для досліджень потреб цільової аудиторії, дизайну інтерфейсу та архітектурі проектування баз даних. Проведено аналіз доступних аналогічних інформаційних систем та потреб групи кінцевих користувачів.

РОЗДІЛ 1. Аналіз предметної області

1.1. Аналіз сучасного стану питання та обґрунтування теми

Щодня кількість інформації, яка нас оточує, зростає з неймовірною швидкістю. Стає все складнішим ефективно використовувати дані, які зберігаються у системах. Від цього страждають не тільки окремі користувачі, але і великі компанії. Для дослідження цієї проблеми було обрано українську маркетингову компанію, що займається організацією заходів.

Основною діяльністю компанії є участь у тендерах на проведення заходів, таких як: майстер-класи, конференції, корпоративи для великих корпорацій. Умовою виграшу тендеру є виконання пропозицією наданих замовником умов та найнижча серед конкурентів. Участь у тендері обмежена невеликим проміжком часу, після якого відбувається підготовка до самого заходу у разі перемоги.

Роботу з інформацією в агенстві можна розділити на такі етапи:

а) **Збір інформації.** Одним із обов'язків старших менеджерів є отримання та оновлення актуальної інформації про місця та послуги, які можуть стати складовими проекту. До цих даних входить інформація про місця проведення (готелі, ресторани, конференц-зали і т.д.), ведучих, артистів, технічних спеціалістів тощо. Отримані дані є повністю приватними і надаються агенству за вимогою.

б) **Обробка інформації.** Під час створення пропозиції або організації заходу головним завданням є пошук місця проведення, ведучих і т.д., які задовольняють запит замовника події. Важливими також є такі фактори, як зручність розташування, ціна, місткість та інше.

в) **Рекламна кампанія.** Після розробки попереднього макету події, проводиться рекламна кампанія для залучення потенційних відвідувачів. Важливим є візуальні складові реклами: банери, інформаційні блоки у соціальних мережах.

Під час організації важливим є спільна робота декількох менеджерів над одним проектом та зручна комунікація між ними, замовником та постачальниками послуг.

На початку дослідження у системі дані зберігалися у вигляді дерева каталогів, де в листкових каталогах знаходилися дані в форматі таблиць MS Excel або медіафайли. Для того, щоб ефективно оперувати цією інформацією, потрібно мати визначений стандарт, де буде вказано, як називати нові файли, які дані мають зберігатися про об’єкти однакового або схожого типу. Цей взірець має бути завжди доступним для всіх співробітників. Незважаючи на ці умови, неможливо контролювати цілісність системи. Серед найбільших недоліків цього методу збереження інформації можна виділити:

– для нових працівників невідома структура системи, щоб знайти будь-яку інформацію потрібно оперувати лише назвами каталогів;
– немає обмежень щодо типів файлів, які зберігаються, та їх змісту;
– неможливо забезпечити захист інформації, легко перемістити, скопіювати, видалити всі файли разом;
– інформацію складно підтримувати актуальною, багато зазначених в системі закладів не функціонували.

1.2. Опитування потенційних користувачів

1.2.1. Анкетування

Було проведено інтерв'ю за порадами наведеними в книзі [1, с. 85-89] з представником кожного з відділів: менеджменту, дизайну, технічного забезпечення. На основі отриманих даних було створено анкету (додаток А). За цією анкетою було проведено опитування співробітників компанії: 8 менеджерів, 2 дизайнерів, 1 технічного спеціаліста. В результаті опитування були отримані результати, зображені на рисунку 1.1.

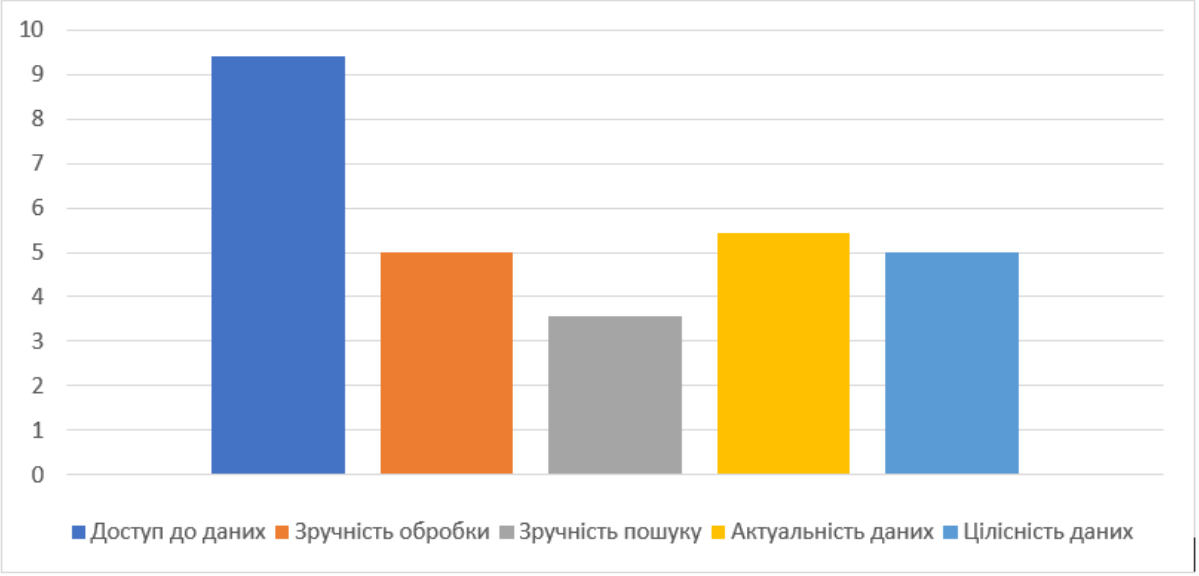


Рисунок 1.1 - Результати анкетування

На даних, отриманих з інтерв'ю та анкет, було виявлено, що найпопулярніші проблеми під час роботи з даними – це невпорядкованість інформації, дублювання інформації, відсутність можливості повернути попередній стан даних, несистематизованість назв файлів, великі витрати по часу для пошуку потрібної інформації.

Серед позитивних сторін було виокремлено простоту користування, можливість зберігати будь-який формат даних.

1.2.2. Тестування системи на завданнях

Одним з основних недоліків системи були великі витрати часу на пошукові операції. Головним завданням під час підготовки для замовника різних пропозицій проведення заходу є створення презентації у MS PowerPoint з вказанням числових та медіаданих щодо кожної з них. Тому для наочної перевірки було проведено незалежне тестування двох груп, кожна з яких складалася з 3 менеджерів. Кожному працівнику було надано окреме завдання.

1 група. Створити презентацію-пропозицію, яка складається з 5 слайдів, на яких буде розміщена інформація (назва, адреса, місткість, фото, короткий опис приміщення) про 5 різних варіантів місць проведення, назви яких були заздалегідь прописані. Основною метою було дізнатися, скільки часу займає знаходження заданої інформація в системі.

2 група. Створити презентацію-пропозицію, яка складається з 5 слайдів, на яких буде розміщена інформація (назва, адреса, місткість, фото, короткий опис приміщення) про 5 різних варіантів місць проведення, які підходили б під задані критерії (місткість, місцезнаходження та інші додаткові вимоги). За мету було обрано перевірку часу, який займає пошук інформації, яка б задовільняла певні фільтри, в системі.

Під час самого експерименту найбільше були помітні недоліки актуальної структури. Співробітники багато разів переглядали той самий каталог, плуталися в тому, де вже відбувався пошук та не могли знайти

навіть ту інформацію, про наявність якої точно знали. Також під час фільтрованого пошуку учасники тестування покладалися на власні знання та здогадки про те, які об’єкти будуть підходити під задані параметри. Єдиним інструментов слугувала пошукова стрічка в файловому провіднику.

В результаті перевірки було отримані такі дані:

Для 1 групи середній час створення презентації-звіту із заданими назвами об'єктів пошуку склав 12хв.

Для 2 групи, яка мала створити презентацію, що має підходити під певні критерії пошуку, середній час склав більше 30хв.

Важливо зазначити, що це час, затарчений тільки на пошук готової інформації, не враховуючи аналіз ринку, конкурентів, замовника та обробку даних, які не наявні в системі, що зазвичай входить в процес створення пропозиції.

1.3. Аналіз існуючих аналогів інформаційних систем

Було проведено аналіз подібних систем для організації заходів.

iDOevent – онлайн-конструктор для організації заходів [2]. Сервіс має вигляд онлайн-магазину, де замість товарів, в корзину можна додавати послуги.

Переваги:

– Легкий доступ. Не потрібна реєстрація, можна одразу передивитися параметризовані елементи.

– Великий вибір складових заходу. Сервіс пропонує замовити як майданчики для проведення, так і музикантів, декораторів, ведучіх тощо.

– Зручна фільтрація з великою кількістю параметрів.

– Для більшості послуг є календар зайнятості, де можна подивитись вільні дати та час.

– Гнучкий калькулятор кошторису.

– Дані постійно оновлюються і перебувають в актуальному стані.

– Готові варіанти пропозицій під різні формати заходів.

– Існує програма лояльності.

Недоліки:

– Відсутність можливості додавати свої варіанти послуг, які не наявні на платформі.

– Замовлення відбувається через посередника. Ціна є вищою за пропозицію напряму від постачальника послуг.

– Стала ціна, тому неможливо створити пропозицію на основі тих самих позицій, що була б краще за варіант конкурента.

– Сервіс працює на дуже обмеженій території.

– Неможливо створити декілька варіантів пропозицій і порівняти їх. В корзині на кожний момент часу може знаходитися тільки один проект.

– Немає можливості спільної роботи декількох менеджерів з різних акаунтів над одним замовленням.

Super Planner – мобільний додаток для розрахунку вартості заходу на основі заданих параметрів [3].

Переваги:

– Калькулятор для розрахунку розмірів приміщення, кількості столів для різних видів розсадки.

– Калькулятор для розрахунку потрібної кількості їжі для вказаного числа відвідувачів.

– Можливість розрахунку кінцевої вартості пропозиції.

– Багатоплатформенність. Підтримка Android та IOS.

Недоліки:

– Додаток є платним для всіх платформ. Немає безкоштовного пробного періоду.

– Немає можливості додавати і зберігати будь-яку інформацію. Для розрахунку потрібно заново вводити дані.

– Можна проводити розрахунок тільки для місця проведення та кейтерингу.

– Немає програми або веб-сайту для доступу з комп’ютера.

– Немає можливості зберігати та переглядати попередні проекти.

– Відсутня можливість реєстрації та спільної роботи над проектами.

– Погана підтримка, відсутність оновлень.

– Маленький функціонал.

У відкритому доступі не було знайдено жодної АІС, яка могла бути використана всередині компанії з власним інформаційним наповненням. Головним напрямком роботи компанії є участь у тендерах. Пропозиція має виконувати всі умови, вказані замовником, та мати найнижчу ціну, щоб здобути перемогу і можливість подальшої організації заходу. Обов’язковою має бути можливість спільного доступу до проектних даних компанії, захист унікальності збережних даних, можливість гнучкого оновлення інформації для підтримки актуальності. Жодна зі знайдених платформ не надавала потрібного для цього функціоналу.

РОЗДІЛ 2. Теоретичні відомості

Під час розробки програмного забезпечення від аналізу користувачів до інтеграції продукту постає велика кількість питання щодо вибору методології розробки, архітектури програми, апаратних та програмних засобів розробки. Не існує одного набору підходів да засобів, який забезпечував ефективну роботу над всіма проектами. Варто обирати ті компоненти розробки, які забезпечать найвищу швидкість та якість конкретного продукту.

2.1. Життєвий цикл програмного забезпечення

Згідно зі стандартом ISO/IEC/IEEE 12207:2017 [4, с. 19-21] життєвий цикл програмного забезпечення можна розділити на 4 процеси:

а) процес домовленості включає в себе виставлення та узгодження вимог зі сторони користувача та сторони розробника продукту;

б) організаційні процеси, що сприяють реалізації продукту мають під собою набір процесів, які потенційно нададуть необхідні для розробки ресурси та забезпечать якість кінцевого продукту;

в) процеси технічного управління описують розподіл технічних ресурсів, постановку вимог якості кінцевого продукту;

г) технічні процеси – це процеси, що стосуються безпосередньої розробки програмного забезпечення: бізнес-аналіз, створення архітектури та дизайну, розробка, інтегрування продукту, тестування та інше.

2.2. Архітектурна модель представлення 4+1

Архітектура застосунку описує компоненти застосунку та взаємодію між ними. Модель представлення 4+1 описує систему як взаємодію 4 різних представлень та ілюструє це на часткових прикладах [5]. Представлленя поділяються на наступні:

а) логічне представлення включає опис основних абстрактних класів та структури проекту у вигляді структурного дерева або діаграм класів;

б) представлення процесів описує процеси системи та взаємодію між ними для забезпечення таких нефункіональних вимог, як масштабованість, вискодоступність, безпека, можливість відновлення системи;

в) представлення реалізації демонструє розбиття системи на менші підсистеми такі, як бібліотеки та модулі, які надають більш конкретний функціонал;

г) фізичне представлення охоплює фізичні компоненти і описує їх як окремі ланки, які забезпечують певний функціонал (тестування, розгортання та підтримка застосунку) і варіанти їх зв’язку для забезпечення масштабованості та високодоступності.

Сценарії у свою чергу демонструють взаємодію всіх чотирьох компонентів між собою на прикладах часткових випадків роботи системи.

2.3. Предметно-орієнтоване проектування

Предметно-орієнтоване проектування (Domain-driven design) – це підхід, коли центральним об’єктом проектування є предметна область системи [6, с. 2-7]. Дизайн будується навколо моделей, що описують специфіку предметної області. Основні концепції:

а) обмежений контекст забезпечується використання декількох моделей, що описують предметну область, і чітким обмеженям їх взаємодії між собою в залежності від контексту;

б) постійна інтеграція включає в себе підтримку цілісності контексту за допомогою підтримки інтеграції фрагментів контектсу в одне ціле;

в) карта контекстів – це чіткий набір моделей та їх зв’язків, що утворюють між собою контексти, які в результаті дозволяють полегшити роботу над декількома моделями і зберігати цілісність їх контекстів.

2.4. Бібліотеки та фреймворки для розробки серверної частини застосунку на Node.js

Node.js – це JavaScript–оточення, що дозволяє обробляти запити користувача поза межами браузера [7].

2.4.1. Express

Express – фреймворк для додатків на базі Node.js, основним використанням якого є маршрутизація клієнтських запитів. Під час проектування системи для кожної моделі на основі її API створюється роутер, який ставить у відповідність шляхи, що відповідають запитам

клієнта, та функціями, які оброблюють цей запит на стороні сервера та повертають потрібну інформацію клієнту.

Express дозволяє застосовувати проміжне програмне забезпечення на рівнях додатку та маршрутизації, для обробки помилок та використовувати стороннє проміжне програмне забезпечення.

Body-parser – проміжне програмне забезпечення для синтаксичного аналізу тіла запиту [8].

Cookie-parser – проміжне програмне забезпечення для синтаксичного аналізу заголовків Cookie [9].

CORS – проміжне програмне забезпечення для регулювання спільного використання ресурсів з різних джерел [10].

2.4.2. Jsonwebtoken

Jsonwebtoken – реалізація JSON Web Token для Node.js [11]. JSON Web Token – маркер доступу, що використовується для авторизації. В заголовку вказується алогоритм, який шифрує підпис. Вміст токена зберігає в собі інформацію про користувача, якого було авторизовано, наприклад, ім’я та роль користувача. Підписом є конкатенація заголовку та вмісте токену, що шифрується за допомогою секретної стрічки на основі обраного алгоритму.

Express-jwt – проміжне програмне забезпечення авторизації для Express, що валідує JSON Web Tokens [12].

2.4.3. Pg

Pg – PostgreSQL клієнт, який дозволяє використовувати пул з’єднання, для роботи з PostgreSQL СКБД [13].

2.5. Бібліотеки та фреймворки для розробки клієнтської частини застосунку

2.5.1. React

React – JavaScript-бібліотека для створення користувацьких інтерфейсів [14]. Будівельними блоками є компоненти, які мають властивості та зберігають свій стан.

Next.js – надбудова над React для створення додатків, які візуалізуються на стороні сервера [15]. Маршрутизація в Next.js відбувається на основі файлової системи, де кожен піддомен є каталогом. Next.js також дозволяє реалізувати власний API не використовуючи сторонніх фреймворків.

2.5.2. Material UI

Material UI - фреймворк для дизайну користувацького інтерфейсу [16]. Пропонує власні компоненти React та набір компонентів для створення дизайну та теми всього додатку, які можна застосувати до всіх компонентів для підтримки цілісності оформлення сторінок.

РОЗДІЛ 3. Проектування системи

3.1. Опис груп користувачів

Власник-директор. Керує роботою компанії. Займається підбором кадрів та організацією комунікації між співробітниками, контролює роботу працівників. Слідкує за всіма процесами, що відбуваються всередині компанії. Приймає рішення про участь у тендерах та прийняття нових пропозицій, проведення рекламних кампаній. Здійснює комунікацію між замовником та співробітниками на етапі формування угоди. Розподіляє фінансування та доступ співробітників до інформації різних рівнів конфіденційності. Бере участь у презентації пропозицій перед замовником.

Креативний директор. Створює унікальні рішення щодо проведення заходів. Є посередником між замовником та дизайнерами, слідкує за роботою креативного відділу. Проводить перемовини з постачальниками послуг сфер маркетингу, реклами та організації шоу, які беруть участь у заході. Створює сценарій подій та спостерігає за ходом його виконання під час проведення.

Старший менеджер. Керує роботою молодших менеджерів. Веде комунікацію з замовником на етапі розробки пропозицій та безпосередньої організації заходу. Спілкується з підрядниками, будівельниками, менеджерами артистів та інших людей, які в перспективі будуть надавати послуги. Їздить у відрядження для становлення та підтримки зв’язків з керівниками готелів, ресторанів та інших закладів; отримання конфіденційної інформації такої як: плани будівель, схеми залів тощо. Вносить отриману інформацію в інформаційну систему. Адмініструє акаунти в соціальних

мережах та на адміністративних сервісах. Перевіряє та завантажує документацію для тендерів. Аналізує ринок та конкурентів, пропозиції суперників на теденрах. Перевіряє угоди та консультується з юристом. Створює технічне завдання для презентацій та презентує її перед замовником. Безпосередньо перебуває на місці проведення події та керує процесом організації. Слідкує за проведенням підготовки, вчасного виконання завдань підрядниками.

Молодший менеджер. Допомагає старшому менеджеру в організації заходів на місці подій. Вносить нові дані до інформаційної системи, слідкує за актуальністю старих даних. У разі необхідності дізнається поточну інформацію від постачальника послуг та оновляє її в системі. Створює презентації-пропозиції для замовників на основі технічного завдання наданого старшим менеджерам. Допомагає знаходити можливі варіанти місця проведення та інших послуг для пропозицій клієнтам. Створює анкети для відвідувачів заходів та переносить цю інформацію в електронний вигляд пілся проведення. Веде облік присутніх під час заходів.

Дизайнер. Створює унікальні дизайни для брендбуку компанії. За потреби створює тематичну символіку, шрифти, логотопи, декорації для заходу. Формує унікальні макети та стилі для презентацій з пропозиціями. Робить дизайн афіш, банерів, рекламних листівок, візитівок, тематичної продукції (футболки, сумки, блокноти з символікою заходу). Займається організацією поліграфії та рекламних кампаній у соціальних мережах.

Бухгалтер. Веде облік капіталу компанії та її прибутків. Збирає дані про витрати на кожен проект та формує звітність. Готує документи для подання заявок на тендери. Виплачує заробітну плату працівникам.

3.2. Технічні вимоги користувачів до функціональності системи

Власник-директор

- реєструвати користувачів тільки за корпоративною поштою з доменом компанії;
- змінювати права доступу для різних типів користувачів;
- переглядати та редагувати режими роботи з даними для ролей;
- створювати проекти і додавати до них складові (місце проведення, ведучі тощо);
- вести декілька проектів одночасно або створювати групи проектів;
- отримувати звіт, які проекти веде кожен з менеджерів.
- переглядати історію проектів;
- додавати, редагувати та переглядати інформацію про готелі;
- додавати, редагувати та переглядати інформацію про номери в кожному готелі;
- додавати, редагувати та переглядати інформацію про додаткові зручності (басейн, спа, спортивні майданчики, тренажерні зали тощо);
- додавати, редагувати та переглядати інформацію про ресторани;
- додавати, редагувати та переглядати інформацію про конференц-зали;
- додавати, редагувати та переглядати інформацію про ведучих, співаків, музичні гурти, аніматорів та інших артистів;

- додавати, редагувати та переглядати інформація про підрядників будівельники, технічні спеціалісти тощо);
- додавати, редагувати та переглядати інформацію про постачальників інших послуг: декоратори, кондитери, костюмерні.

Старший менеджер

- додавати, редагувати та переглядати середній цінник для номерів в готелі;
- додавати, редагувати та переглядати інформацію про додаткові зручності;
- додавати, редагувати та переглядати інформацію про ресторани.
- додавати, редагувати та переглядати інформацію про конференц-зали;
- додавати, редагувати та переглядати інформацію про ведучих, співаків, музичні гурти, аніматорів та інших артистів;
- додавати, редагувати та переглядати інформація про підрядників;
- додавати, редагувати та переглядати інформацію про постачальників інших послуг;
- додавати, редагувати та переглядати інформацію про складову проекту та беріатии у PDF-докумет для офлайн роботи;
- створювати макетів дописів для соціальних мереж;
- прив’язувати акаунт до корпоративної пошти і писати листи підрядникам одразу з внтутрішньої системи;
- комунікувати з іншими менеджерами всередині системи з пересилкою на пошту;
- пересилати інформацію про складові та проекти всередині системи.

Молодший менеджер

- додавати, редагувати та переглядати макети презентацій;
- копіювати всю інформацію, яка потрібна для слайду про складову заходу, однією дією;
- фільтрувати дані по параметрам (для готелів розташування, кількість зірок; для ресторанів розташування, середній цінник, кількість місць і т.д.).
- повнотекстовий пошук пошук по назві або імені.

Креативний директор та дизайнер

- додавати, редагувати та переглядати складові брендбуку компанії;
- додавати, редагувати та переглядати брендбуки компаній-клієнтів;
- додавати, редагувати та переглядати макети презентацій;
- додавати, редагувати та переглядати дизайни під кожний проект;
- додавати, редагувати та переглядати статистику про рекламні кампанії.

Бухгалтер

- отримати звіт по поточних проектах;
- отримати звіт по проектах за місяць;
- отримати звіт по проектах за рік;
- отримати звіт по кожному з менеджерів про проекти, якими він керує;
- отримати звіт по кожному з менеджерів про проекти, якими він керував за місяць;
- отримати звіт по кожному з менеджерів про проекти, якими він керував за рік;
- отримати звіт по найпопулярніших закладах, в яких проводлилися заходи;
- отримати по компаніях-замовниках про проекти, які вони замовляли;

- додавати, редагувати та переглядати документацію по тендерах на всі проекти.

3.3. Специфікація вимог

3.3.1. Вимоги до даних

Готелі

Готелі існують як окремі об’єкти, які можна переглядати та змінювати. Про них має зберігатися така інформація: назва, веб-сайт, телефон (якщо для різних відділів різні контактні номери, то зберігати всі з описом того, який це відділ, наприклад, рекламний або рецепція), електронна пошта (для кожного відділу відповідно), кількість зірок, адреса, фото, короткий опис, додаткова інформація. Готель також є складовою проектів, тому має відображатися як їх частина і якщо є, то відгук про досвід роботи з закладом від менеджерів, які керували проектом.

Готельні номери

Готельні номери є складовими частинами готелів і не можуть існувати окремо від них. Про них має зберігатися така інформація: назва номеру відповідно до прейскуранту, кількість осіб в номері, кількість кімнат, підвид номера(люкс/напівлюкс), площа приміщення, прейскурант, фото, короткий опис, додаткова інформація.

Конференц-зали

Конференц-зали можуть існувати як незалежні місця проведення або як частини готелю і відповідно відображатися в інформації про готель. Про них має зберігатися така інформація: назва, веб-сайт, телефон (якщо для різних

відділів різні контактні номери, то зберігати всі з відповідним описом), електронна пошта (відповідно до відділів), місткість, площа, схема розстановки, прейскурант, фото, короткий опис, технічне обладнання (інтерактивні дошки, магнітні дошки, мікрофони, комп'ютери, проектори тощо), додаткова інформація. Якщо, конференц зал є частиною готелю, то адресою має бути готель, в якому він знаходиться. Для кожного конференц-залу відображати проекти, в яких він брав участь і якщо є, то відгук про досвід роботи з закладом від менеджерів, які керували проектом.

Ресторани

Ресторани, як і конференц-зали, можуть бути окремими одницями або бути кладовими готелю і відповідно відображатися в інформації про готелях, в яких вони знаходяться. Про ресторани має зберігатися така інформація: назва, веб-сайт, телефон (для всіх відділів з описом), електронна пошта (відповідно до відділва), адреса, місткість, площа, банкетні послуги, схема залів, меню, фото, додаткова інформація. Якщо, ресторан є частиною готелю, то адресою має бути готель, в якому він знаходиться. Для кожного роесторану відображати проекти, в яких він брав участь і якщо є, то відгук про досвід роботи з закладом від менеджерів, які керували проектом.

Артисти

Про кожного артиста має зберігатися така інформація: ім'я або назва групи, місто перебування, можливість проведення подій в інших містах, умови перебування, прейскурант, контактна інформація, фото, додаткова інформація. Більшість артистів мають одного або декілька менеджерів. В цьому випадку зберігати контактну інформація як сукупність контактів менеджерів у вигляді: ім'я менеджера, номер, електронна пошта. Для

кожного артиста відображати проекти, в яких він брав участь і якщо є, то відгук про його роботу від менеджерів, які керували проектом.

Підрядники

Про кожного підрядника має зберігатися така інформація: ім’я, веб-сайт, телефон, електронна пошта, вид роботи, місто перебування, прейскурант, додаткова інформація. Для кожного підрядника відображати проекти, в яких він брав участь і якщо є, то відгук про його роботу від менеджерів, які керували проектом.

Інші постачальники послуг

Про кожного постачальника має зберігатися: назва компанія або ім’я працівника, веб-сайт, телефон, електронна пошта, місто перебування, прейскурант, приклади робіт, додаткова інформація. Для кожного постачальника відображати проекти, в яких він брав участь і якщо є, то відгук про його роботу від менеджерів, які керували проектом.

Проект

Для кожного проекту має бути зазначена така інформація: назва, дати проведення, менеджери, які вели проект, складові проекту, вартість, прибуток.

Працівники компанії

Кожен працівнк компанії має свою унікальну корпоративну пошту, через яку реєструється в системі АІС. Окрім цього в акаунті кожного працівника зберігається така інформація: унікальний табельний номер, ПІБ, адреса, посада (роль), стать, дата народження, зарплата (грн./місяць), номери

телефонів, дата прийняття на роботу та звільнення (якщо існує). Для кожної посади (керівник, креативний директор, дизайнер, старший менеджер, молодший менеджер, бухгалтер) є відповідна роль доступу до перегляду та зміни інформації в системі. Зміну ролей здійснює власник компанії, який має доступ до всіх даних та має право їх змінювати. До документації, окрім власника, має доступ бухгалтер. Старший менеджер може переглядати та змінювати інформацію про проекти. Інфомрацію про окремі складові та брендбук компанії можуть переглядати та змнінювати усі працівники. Креативний директор та дизайнер мають доступ до брендбуків інших компаній та до інформації про дизайни до кожного проекту.

3.3.2. Системна специфікація

Початковий розмір бази даних. Всі унікальні дані, які зберігаються у поточній системі (50ГБ з повтореннями у виглядів файлів та каталогів).

Темп росту бази даних. Темп росту напряму залежить від кількості нових проектів та нових компаній або окремих людей, з якими буде співпраця.

Типи інформаційного пошуку та їх розподіл по частоті використання. АІС має підтримувати параметризований пошук та пошук за ключовими словами.

Вимоги до роботи в мережі та спільного доступу. Система має розгортатися на сервері компанії. Працівники повинні мати безпереревний спільний доступ до перегляду та зміни інформації всередині локальної мережі. Після зміни інформації, оновлени дані мають відображатися без безпосередньої ініціації користувача.

Захист. Авторизація користувачів відбувається за корпоративною поштою та паролем, створеним під час реєстрації користувача в системі. Кожен користувач має свою роль і може переглядати тільки ту інформацію, яка дозволена на його рівні. До АІС немає доступу поза межами локальної мережі. Всі інформацію з бази даних неможливо скопіювати за один раз. Захист від XSS атак та SQL ін'єкцій.

Резервне копіювання. Усі дані, що зберігаються в базі, мають бути скопійовані до резервної копії, прихованої та захищеної паролем. Доступ до копії має лише власник. Копіювання відбувається щотижня.

3.4. Архітектура системи

Архітектуру системи було розроблено за допомогою патерну MVC з використанням підходу предметно-орієнтовано програмування (рис. 3.1). Контроллери відповідають рівню представлення і надають інтерфейс до бізнес-функцій, які обґрунтовані пердметною областю. Прикладний рівень представлений сервісами, які делегують обробку запиту на рівень домену. В свою чергу доменний рівень складається з моделей та репозиторіїв. Моделі уособлюють в собі бізнес-логіку застосування, а репозиторії відповідають за отримання інформації з бази даних та її передачу у зворотній бік.

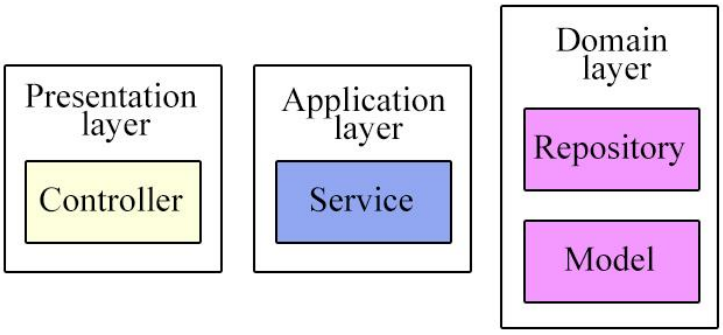


Рисунок 3.1 - розподілення рівнів застосунку

Процес обробки взаємодії користувача з інтерфейсом можна описати такими етапами (рис. 3.2):

а) користувач взаємодіє з компонентами користувацького інтерфейсу;

б) компонент надсилає запит серверу;

в) запит перехоплюється маршрутизаром, який прослуховує відповідний порт та викликає відповідний до API контролер;

г) запит проходить ланцюжок з проміжного програмного забезпечення;

д) контроллер отримує попередньо оброблений запит та викликає відповідну функцію сервісу його домену, після виконання надсилає відповідь з отриманими даними або помилкою;

е) сервіс проводить валідацію інформації та вирішує, як обробити запит та повертає дані або помилку;

ж) репозиторій звертається через з’єднання з пулу до бази даних для здійснення змін відповідно до запиту або отримання інформації та повертає її сервісу.

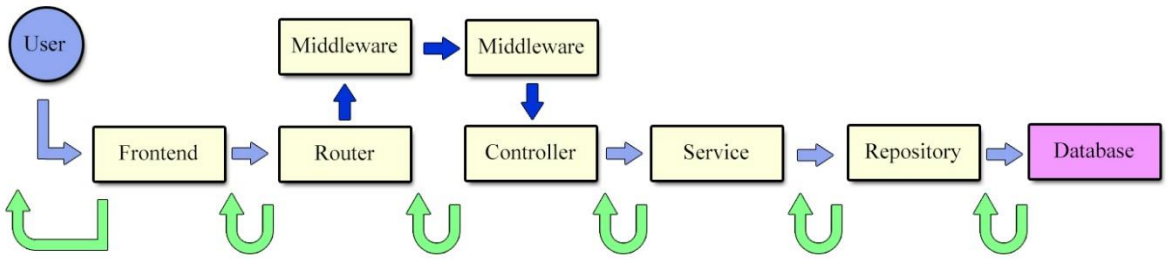


Рисунок 3.2 - обробка взаємодії користувача

3.5. Логічне проектування

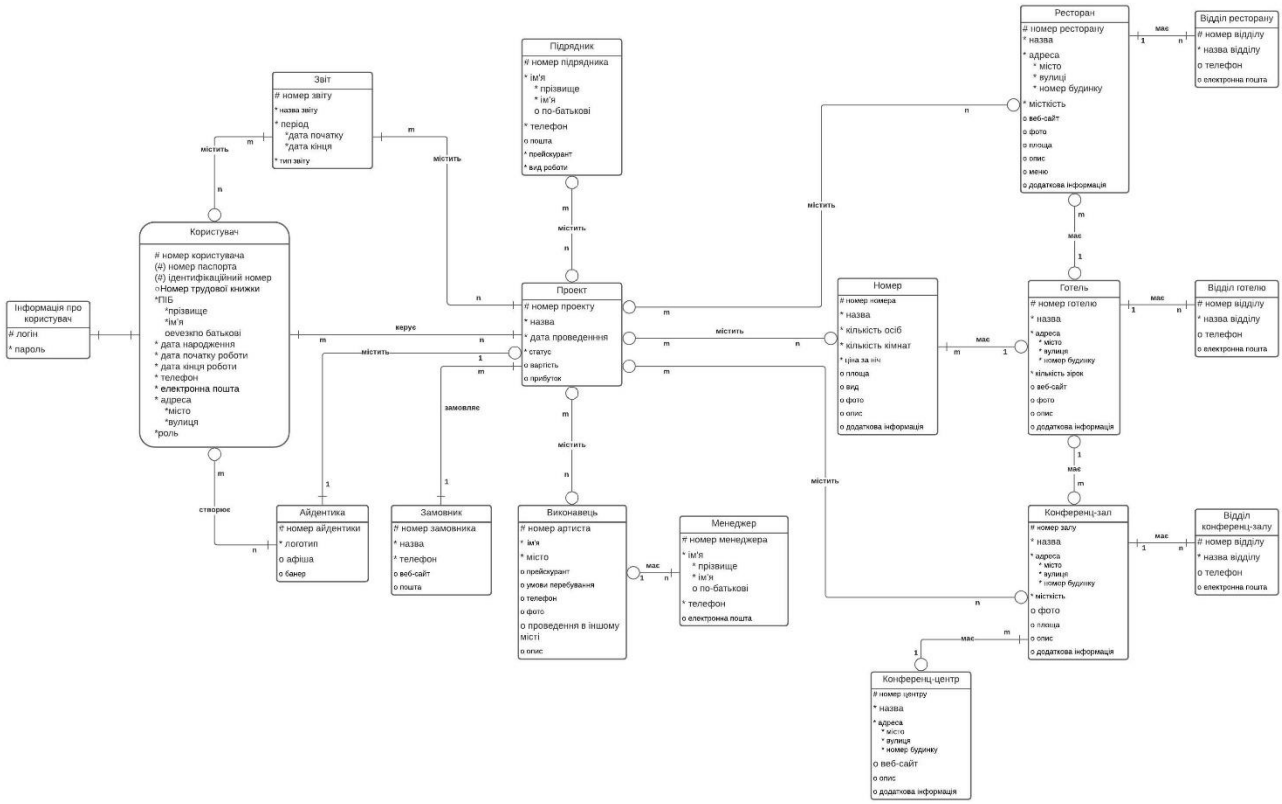


Рисунок 3.3 - модель «сутність-зв'язок»

На рисунку 3.3 зображено діаграму, яка відображає бізнес-логіку предметної області.

Проектування починається з основної моделі будь-якого застосунку - користувача. Інформацію про акаунт користувача виокремлено в окрему сутність зі зв’язком один до одного з користувачем з міркувань обмеження контексту, адже акаунт користувача існує тільки в межах додатку, в той час, як сама предметна область описує користувача тільки як працівника компанії.

Користувач може мати одну з чотирьох ролей: адміністратор (належить тільки керівнику компанії), менеджер, дизайнер, бухгалтер. Адміністратор

при створенні нового користувача можу наділяти його тільки однієї роллю. В залежності від ролі, яка контролюється коропоративними обмеженнями цілісності даних користувач може керувати проектом або створювати айдентику. Усі користувачі можуть бути складовими звіту.

Проект унікально ідентифікується своїм порядковим номером, може включати в себе всі компоненти, які беруть участь в організації та проведення заходу (проекту), а саме: номери готелів, ресторани, конференц-зали, виконавці, підрядники та айдантика.

Усі номери, які наявні в системі, мають порядковий номер, який їх унікально ідентифікує. Номери знаходяться в готелі і тільки одному. В свою чергу готель може містити багато різних номерів і також має свій порядковий номер.

Конференц-зал унікально ідентифікується порядковим номером і може знаходитися або в конференц-центрі, або в готелі, через це зв’язок з кожною з моделей є необов’язковим, проте корпоративними обмеженнями контролюється наявність одного і тільки одного з них.

Ресторан, як і конференц-зал, має власний порядковий номер та може знаходитися на території готелю і бути його частиною.

В моделі відсутні контакти для складових проекту, але наявний зв’язок з «відділами». Це явище спричинила відсутність однозначної контактної лінії зв’язку з компонентом. Один ресторан може мати відділ бронювання та відділ кейтерингу. В залежності від функції у проекті, ресторан буде мати різні контактні дані.

Звіт може містити або проекти, або користувачів, наявність хоча б одного зі зв’язків контролюється корпоративними обмеженнями та вказанням типу звіту.

Готелі та номери, ресторани, конференц-зали мають обов’язкові атрибути такі, як адреса або рейтинг, що дозволить здійснювати фільтрацію, а отже, і ефективний пошук по системі.

3.6. Проектування бази даних

На базі моделі «сутність-зв’язок» було спроектовано реляційну модель для побудови бази даних в PostgreSQL (рис. 3.4). Зв’язок між користувачем та акаунтом було перетворено на дві таблиці, де в акаунті зберігається посилання на користувача, щоб при авторизації можна було отримати інформацію про дані користувача, зокрема його роль.

Усі зв’язки один до багатьох було реалізовано додаванням зовнішнього ключа на сутність, з якої починався зв’язок.

Усі зв’язки багато до багатьох було спроектовано за допомогою додаткових реляцій, які мали як частинні первинні ключі зовнішні ключі на реляції, що беруть участь у зв’язку.

При перетворенні сутності конференц-залу на реляцію, було додано необов’язкові зовнішні ключі, що посилаються на реляції готелю та конфренес-центру.

Так само було спроектовано звіти, додавши два необов’зякових зовнішніх ключі, що посилаються на користувача та на проект.

Обмеження цілісності було спроектовано з огляду на предметну область. Для всіх відділів на видалення було накладено обмеження CASCADE при видаленні реляції, на яку посилається зовнішній ключ. Відділи готелю або ресторану не матимуть сенсу за відсутністю реляції, на яку посилаються, адже при проектування відділи було винесено в окремо модель через

неоднозначність контакної інформації та багатозначність атрибуту (декілька відділів належать одному готелю і т.д.).

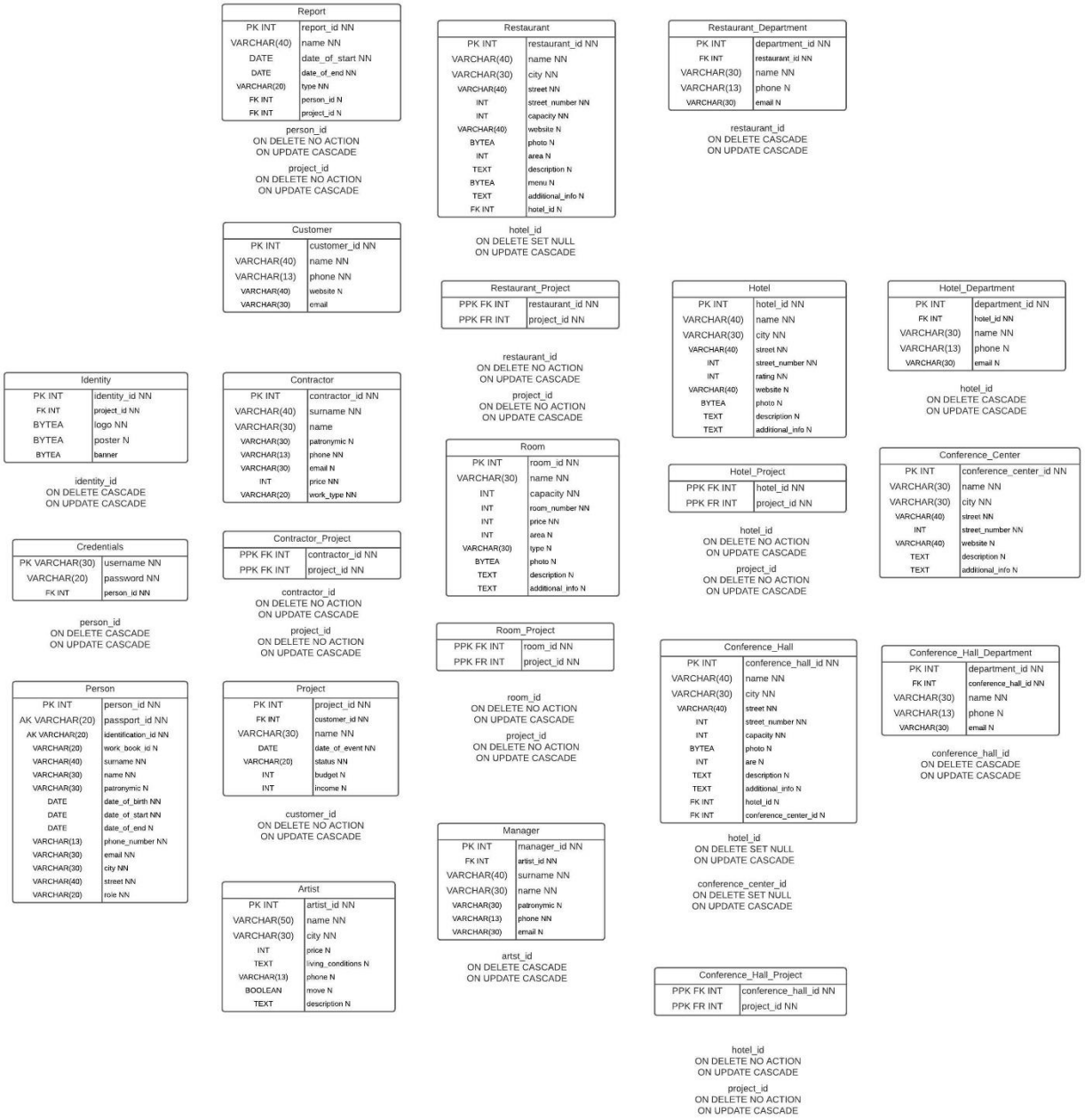


Рисунок 3.4 - реляційна модель

Для реляцій, які мали зв’язок один до одного і виокремлені через специфіку предметної області, було встановлено обмеження CASCADE для видалення.

Для реляцій, які мають необов’язковий зовнішній ключ, було накладено обмеженням SET NULL при видаленні.

Для доступу до зображень в системі було обрано зберігання в базі даних замість посилань на локальні шляхи. Це зумовлено вимогами до безпеки, адже локально зображення б зберігалися всі в одному місці і їх було легко разом скопіювати або видалити.

3.7. Дизайн інтерфейсу

Сторінка авторизації (рис. 3.5). Містить можливість тільки авторизуватися, немає можливості зареєструватися. Усі акаунти створює адміністратор, якого напряму додано в базу даних. Роль адміністратора виконує керівник компанії.

Рисунок 3.5 - макет сторінки авторизації

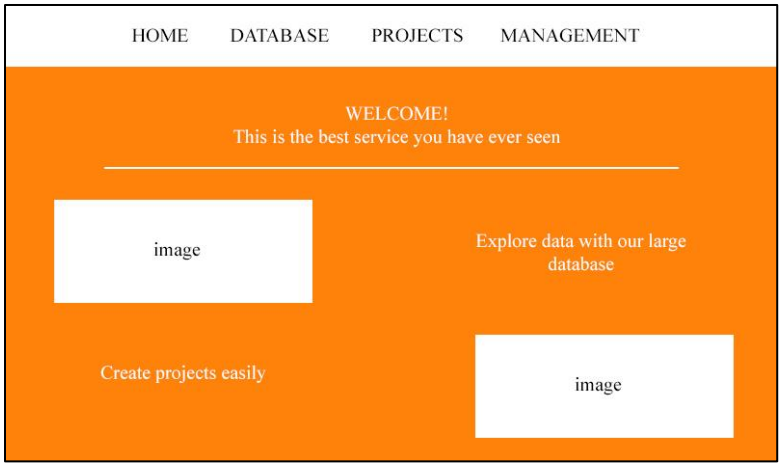


Рисунок 3.6 - макет домашньої сторінки

Домашня сторінка (рис. 3.6). Містить вітання та список можливостей сайту.

Сторінка бази даних (рис. 3.7). Відповідно до обраної категорії містить перелік моделей та набір фільтрів для пошуку.

Рисунок 3.7 - макет сторінки переліку готелів

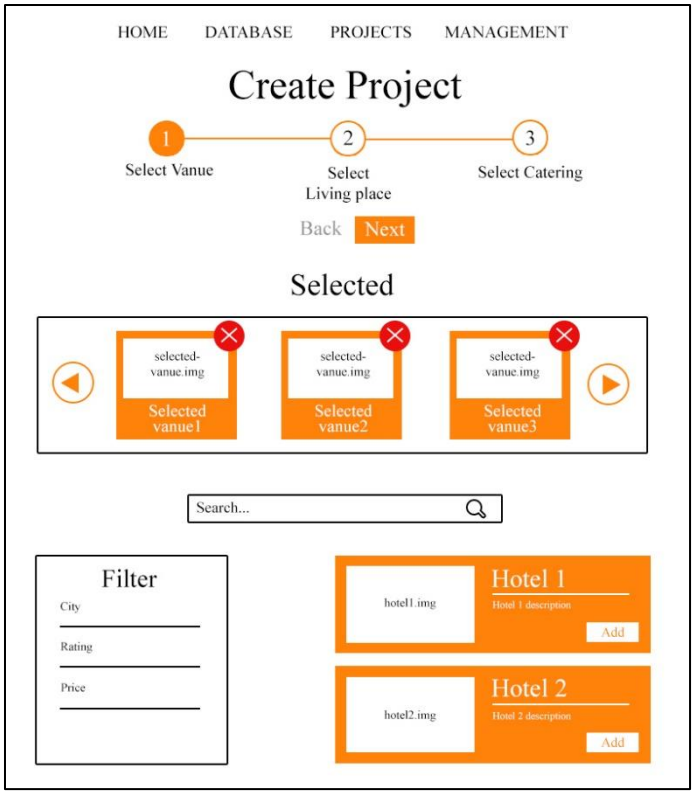


Рисунок 3.8 - сторінка створення проекту

Сторінка створення нового проекту (рис. 3.8). Проект створюється у декілька етапів. Кожен етап супроводжується вибором елементів з можливим фільтрованим пошуком, всі етапи, окрім першого, можна пропускати. В будь-який момент можна повернутися до попереднього етапу зі збереженням стану.

Сторінка управлінная користувачами (доступна тільки для адміністратора). Дозволяє переглядати користувачів та додавати нових.

Для створення теми продукту було обрано брендовий колір компанії – помаранчевий (#ff820a). Як контрастний колір було обрано білий та чорний для звичайного тексту.

РОЗДІЛ 4. Опис реалізації

4.1. Структура програми

Для реалізації клієнтської частини застосування було обрано фреймворк React, а саме його надбудову Next.js.

Next.js дозволяє розробити власний API, проте було обрано розділити реалізацію на серверну та клієнтську частину задля легшої масштабованості та можливості використовувати API окремо від клієнтського інтерфейсу.

4.1.1. Клієнтська частина застосунку

Процес маршрутизації в Next.js проходить набагато зручніше в порівнянні зі стандартним React. Маршрутизація відбувається на основі файловою системи, а саме файлів, що знаходяться в каталозі “pages”. Точкою входу є обов’язковий файл “index.js”, що за замовчуванням відповідає шляху “/”. Щоб перейти на іншу сторінку, потрібно як і в React, додати роутер з посиланням на наступну сторінку, де посиланням є відносний шлях до потрібний сторінки в дереві файлової системі.

На рисунку 4.1 зображено файлове дерево каталогу “pages”. На цьому прикладі можна побачити, що підкаталоги конференц-залу, готелю, проекту та ресторану відповідають відповідним піддоменам. Щоб з домашньої сторінки, яка реалізована у файлі “home” перейти на сторінку створення проекту, в Link потрібно вказати шлях “/project/create”, що відповідало б шляху при переході в цей файл у файловій системі. Так само, щоб зі сторінки окремого конференц-залу перейти на сторінку авторизації, потрібно вказати шлях “../index”. Окрім розташування у відповідних файлах реалізацій сторінок ніяких додаткових залежностей не потрібно вказувати для коректної

маршрутизації на відміну від стандартного React, де при додаванні нової сторінки до проекту, потрібно обов’зяково новостворений компонент додавати до маршрутизатора.

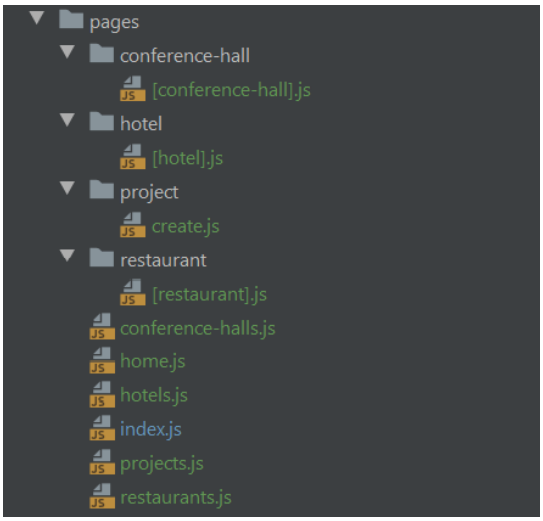


Рисунок 4.1 - каталог "pages"

Також на рисунку 4.1 можна побачити приклад сторінок з динамічною генерацією таких, як “[conference-hall].js”, “[hotel].js”, “[restaurant].js”. На динамічну генерації фказують квадртані дужки, що обрамлюють назву сторінки. Динамічна сторінка генерується на основі параметру, який переданий в шляху до неї, для того, щоб перехопити цей параметр потрібно підключити маршрутизатор (рис. 4.2). Як видно, далі цей параметр можна використовувати як звичайну змінну, наприклад, в запити до API.

```
import { useRouter } from 'next/router'

const router = useRouter();
const { rid } = router.query;
const { state, data, error } = api( url: `/api/restaurant/${rid}` );
```

Рисунок 4.2 - перехоплення параметру для динамічної генерації сторінки

Вище зазначалось, що під час реалізації було обрано повне розділення клієнтської та серверної частини, проте на рисунку 11 видно, що шлях до API має вигляд відносного, а не абсолютного. З версії 9.5 в Next.js [17] було представлено можливість співставляти шляхи запиту з кінцевою адресою за допомогою функції “rewrites”. Алгоритм схожий на поведінку фізичного маршрутизатора. На прикладі з рисунку 4.3 продемонстровано, як всі запити, для яких буде вказаний шлях “/api/:path”, де “:path” це параметр будуть направлені на адресу сервера з вдповідним шляхом до його API.

Рисунок 4.3 - вміст файлу "next.config.js"

4.1.2. Серверна частина застосунку

Для реалізації шаблону серверної частини було обрано мову Node.js через дуже зручну і легку маршрутизацію, яку можна впровадити в архітектуру застосунку.

Рівень бізнес-логіки представлений моделями (рис. 4.4) та сховщами (рис. 4.5). Моделі описують конкретні аспекти предметної області. Кожній моделі відповідає своє сховище, яке надає доступ до бази даних та здійснює

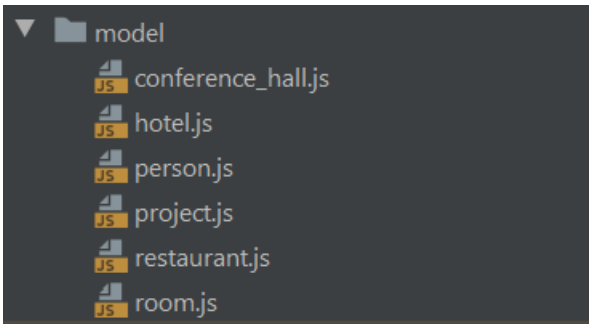


Рисунок 4.4 - вміст каталогу "model"

маніпуляції над інформацією, яка в ній збережена.

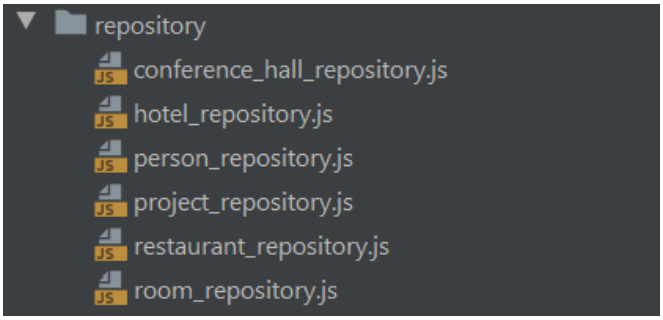


Рисунок 4.5 - вміст каталогу "repository"

Прикладний рівень реалізовано за допомогою сервісів (рис. 4.6). Кожній моделі відповідає свій сервіс. Він керує тим, яким чином буде відбуватися здобуття результату, який хоче отримати користувач. На рівні сервісу можуть використовуватися сервісі інших моделей, якщо цього потребує предметна область.

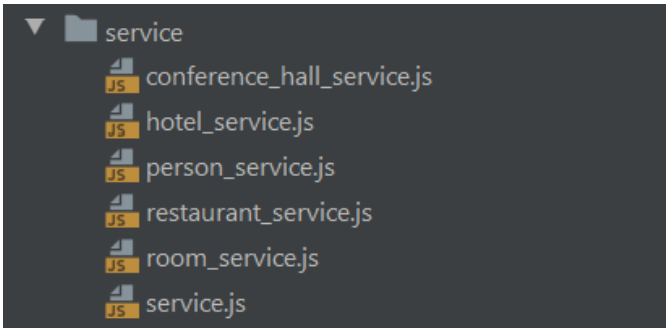


Рисунок 4.6 - вміст каталогу "service"

Контроллери реалізують рівень представлення (рис. 4.7). Вони отримують запит користувача і визначають, яка відповідь буде повернення після здійснення обрахунків сервісами. Кожний контроллер відповідає за надання інтерфейсу до API відповідної моделі предметної області.

В маршрутизаторах (рис. 4.8) зберігається відповідність між шляхом, вказаним у запиті користувача, та функцію контролера, яка має обробити цей запит та повернути відповідь. Кожний маршрутизатор відповідає одному з доменів предметної області.

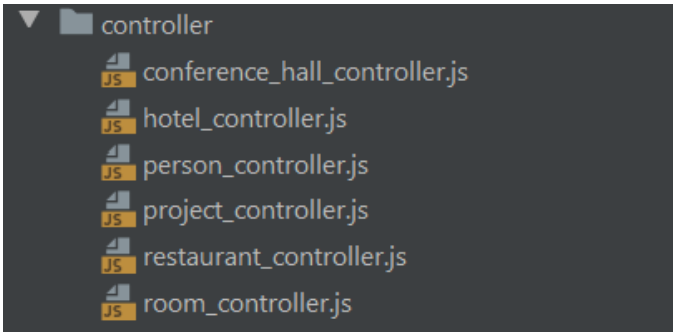


Рисунок 4.7 - вміст каталогу "controller"

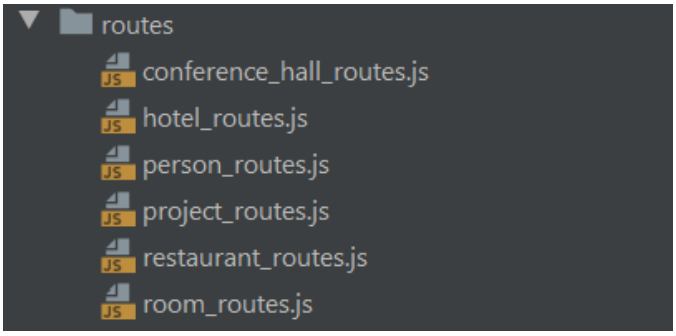


Рисунок 4.8 - вміст каталогу "routes"

Для зв’язку з базою даних використовується набір з’єднань, які встановляються на початку роботи сервера (рис. 4.9). Для кожного запита

обирається одне з вільних з’єднань, таким чином при новому запиті не потрібо кожний раз заново створювати зв’язок.

```
const Pool = require('pg').Pool;
const dbConfig = require("./config/db.config.js");

const pool = new Pool( options: {
    user: dbConfig.USER,
    host: dbConfig.HOST,
    database: dbConfig.DB,
    password: dbConfig.PASSWORD,
    port: dbConfig.PORT,
});

exports.pool = pool;
```

Рисунок 4.9 - створення пулу з'єднань

Авторизація реалізована за допомогою JSON Web Token. Задля кращої масштабованості було обрано реалізація клієнт-серверної архітектури без збереження стану. Клієнт має однозначно себе ідентифікувати в кожному запиті до серверу. При авторизації на головній сторінці додатку, після підтвердження логіну та пароля створюється унікальний токен, який включає в себе ім’я користувача та його роль в системі. Після цього токен шифрується за допомогою секретного ключа. Токени передаються в HttpOnly Cookie. Таким чином мініфікується можливість ппідробки токену, адже його неможливо прочитати або змінити його з браузера.

Доступ до всіх сторінок, окрім сторінки авторизації, є тільки у авторизованих користувачів, що мають свій токен, який валідується сервером і періодично оновлюється.

Для реалізації авторизації було вирішено не використовувати готові рішення такі, як OAuth 2.0 через потребу у наданні приватної інформації стороннім сервісам.

Безпека зберігання інформації реалізована за допомогою використання алгоритму шифрування Argon2 для зберігання паролей користувачів.

4.2. API

Для кожного домену предметної області на стороні сервера було створено CRUD функції (додатки Б, В). Додавання нового об’єкту здійснюється за допомогою POST методу з об’єктом моделі, який передається в тілі запиту. Для зчитування використано метод GET без парамтерів для отримання всіх об’єктів домену або з параметром, який ідентифікує конкретний об’єкт, який потрібно повернути. Редагування інформації реалізовано за домогою UPDATE методу. Параметри, які потрібно змінити та їх нове значення передаються в тілі запиту. Для видалння одного або усіх об’єктів використано метод DELETE парамтризований або без параметрів відповідно.

Після кожного запиту контролер повертає або інформацію, отриману після успішного виконання запиту, або код помилки та інформацію про неї.

Документацію API було згенеровано за допомогою Swagger.

4.3. Інтерфейс та користувацький досвід

Для зручного дизайну було використано компоненти фреймворку Material UI. Компоненти було обгорнуто у власну тему, створену на основі кольорів компанії. Розробка інтерфейсу проводилася з розрахунку зручного користування.

Сторінка авторизації (рис.4.10). На сторінці авторизації розміщено форму для вводу імені користувача та його паролю. З міркувань безпеки введені в поле паролю символи змінені.

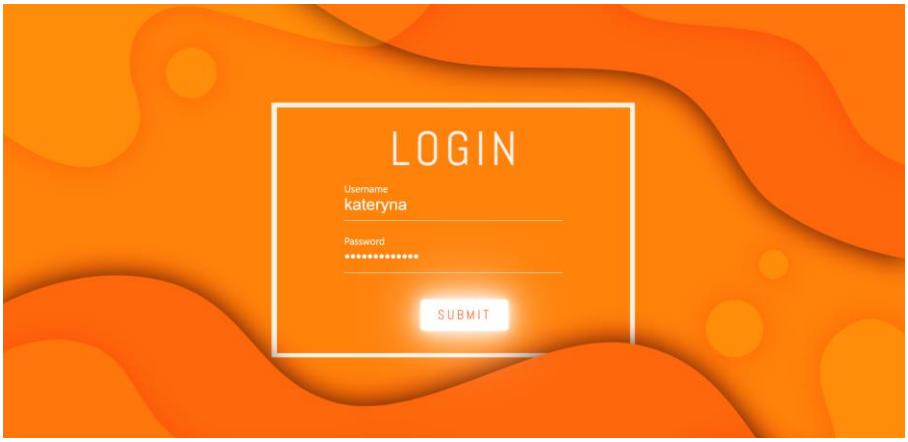


Рисунок 4.10 - сторінка авторизації

Домашня сторінка (рис. 4.11). Якщо пароль було введено правильно, користувач переходить на домашню сторінку з привітанням та посиланням на вкладинку з проектами. Також згори можна побачити меню з посиланнями на сторінки бази даних, проектів та менеджменту для користувача з роллю адміністратора.

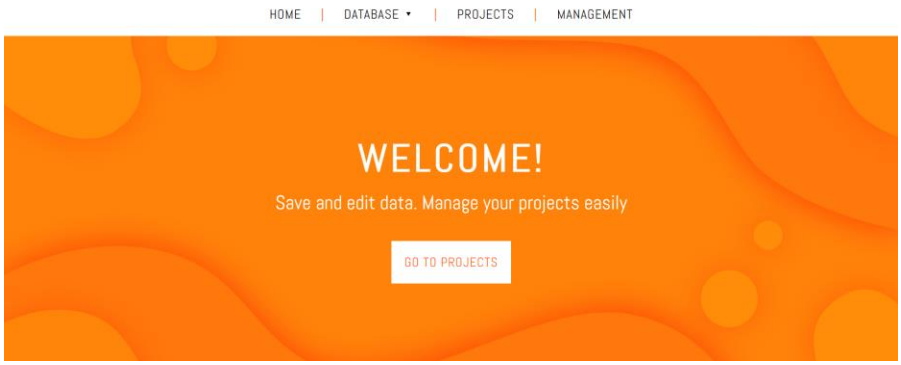


Рисунок 4.11 - домашня сторінка

Сторінки розділів бази даних (рис. 4.12). Вкладки бази даних містить в собі декілька категорій компонентів заходу: готелі, конференц-зали,

ресторани, артисти, підрядники. На сторінці кожної з цих категорій надається фільтр пошуку та перелік компонентів, що задовільняють заданій фільтрації. Кожен елемент списку відображає базову інформацію та має кнопку з посиланням на власну сторінку.

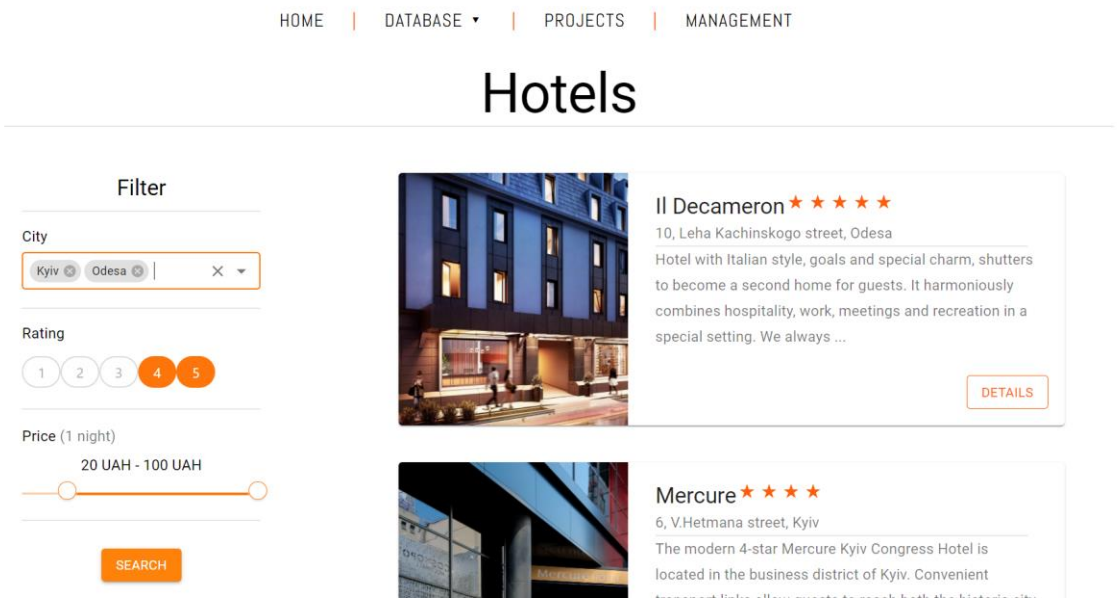


Рисунок 4.12 - сторінка категорії бази даних

Сторінка об’єкту (рис. 4.13). Ця сторінка має повну інформацію, яка доступна про об’єкт і також містить в собі посилання на інші об’єкти, які з ним пов’язані.

Сторінка створення проекту (рис. 4.14). Створення проекту скаладається з трьох етапів: вибору місця проведення, місця проживання та кейтерингу, тобто ресторану. Обов’язковим є тільки перший етап. На кожному етапі пропонується вибір компонентів з доступних для цього етапу категорій. На кожному етапі є можливість повернутися на попердній етап та змінити обрані компоненти без втрати інформації.

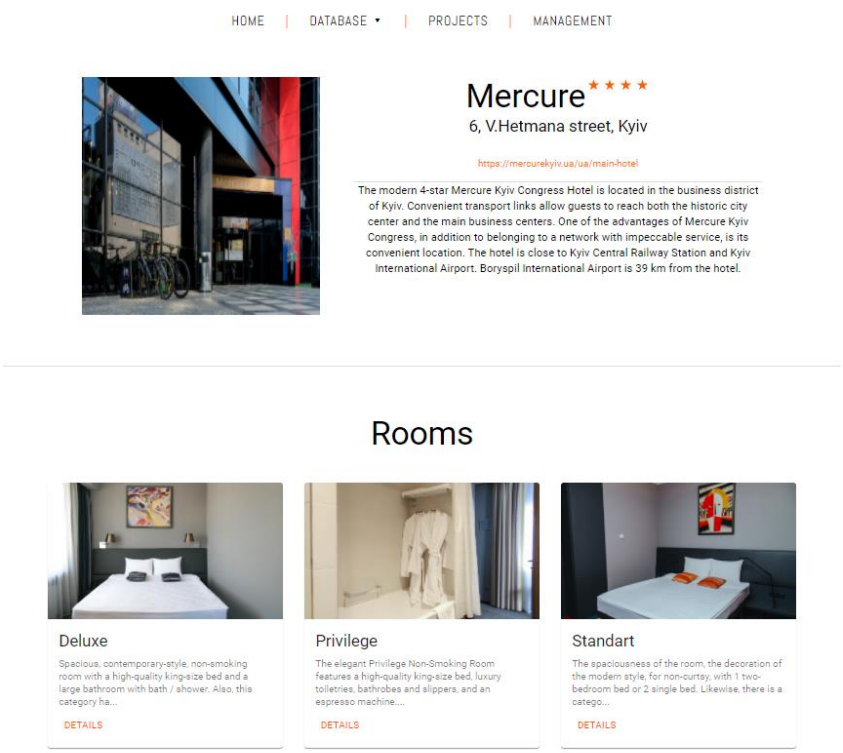


Рисунок 4.12 - сторінка об'єкту

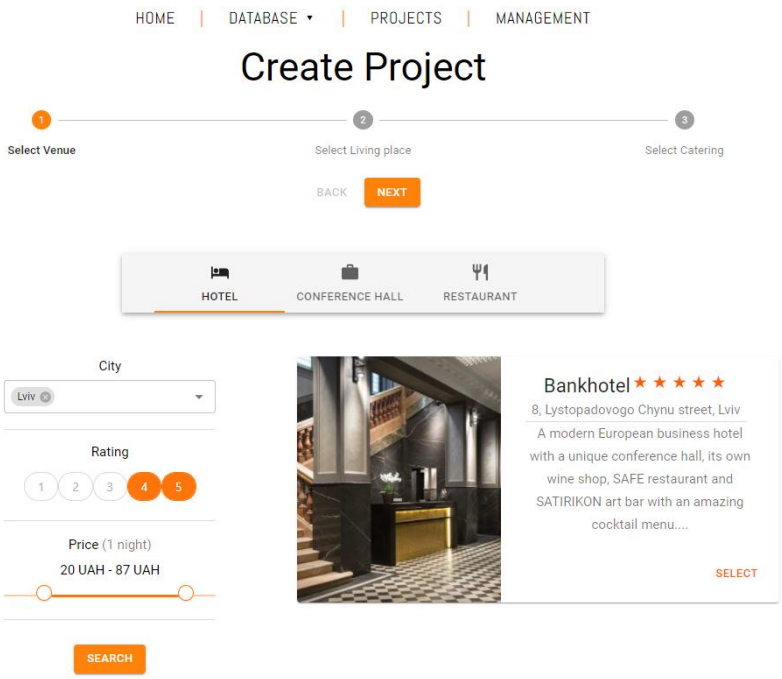


Рисунок 4.13 - сторінка створення проекту

Користувач переміщується між сторінками за допомогою посилань, які розміщенні на кожній з них. Також для зручності переміщення між основними комопнентами сайту було додано меню, яке видно згори на будь-якій сторінці, окрім авторизації. Всі переходи, які можливо зробити за допомогою інтерфейсу на сторінці можна зобразити у вигляді мапи (рис. 4.14).

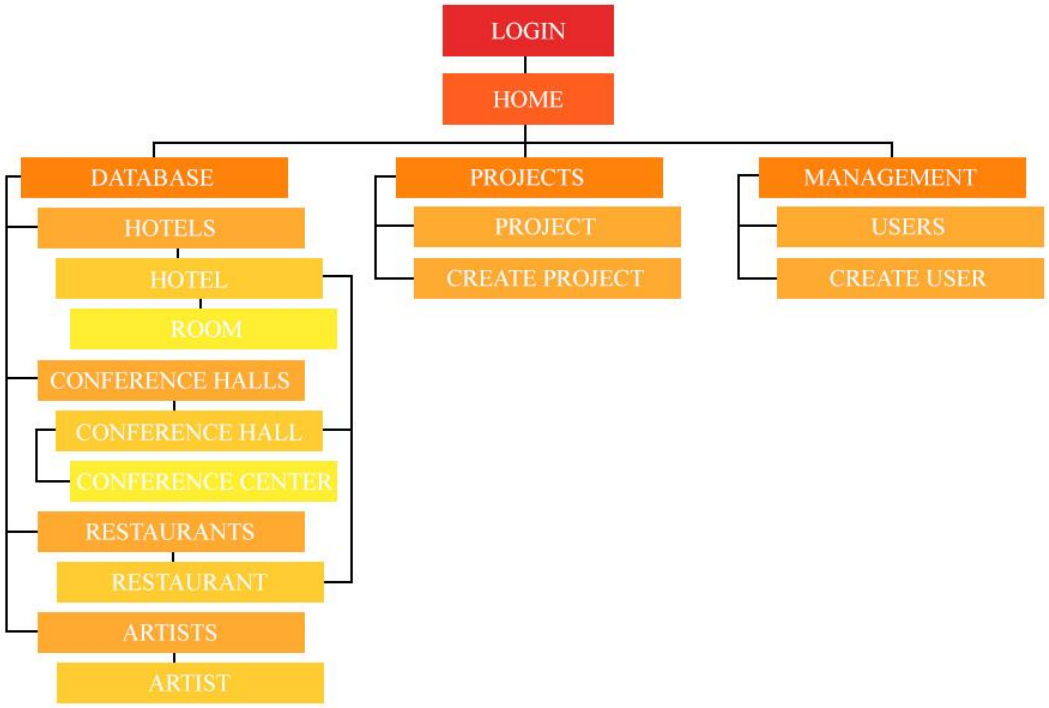


Рисунок 4.14 - мапа сайту

Висновки

Під час розробки було:

а) проведено аналіз предметної області, виокремлено особливості організації подій для розробки додатку, що буде ефективно виконувати потрібні функції;

б) досліджено методи опитування цільової аудиторії для ефективного проведення аналізу потреб користувачів, ;

в) проаналізовано аналогічні інформаційні системи, знайдено переваги та недоліки для подальшого усунення в розроблюваній системі;

г) проаналізовано різні методи проектування систем та засоби розробки для обрання найкращого з них для реалізації поставленого завдання;

д) спроектовано архітектуру системи, розроблено логічну та реляційну модель, створено прототипи сторінок;

е) реалізовано основний функціонал системи, які покриває базові вимоги користувачів.

Архітектура системи дозволяє повністю задовольнити функціональні вимоги всіх груп користувачів та дає можливість за вимогою надалі розширяти проект. В подальшому планується реалізувати можливість формувати звіти, основані на роботі менеджерів над проектами, швидкого копіювання відформатованої інформації до презентації. Також доповнити розділи бази даних на сайті такими пунктами, як дизайн проектів, брендбук компанії; покращити безпеку застосунку та його швидкодію.

Список використаних джерел

1. Hall E. Just Enough Research. New York : A Book Apart, 2013.
2. Онлайн-конструктор для організації заходів iDoEvent [Електронний ресурс]

   idoevent.ru/
3. Багатоплатформенний мобільний додаток для планування подій Super Planner [Електронний ресурс]

   apps.apple.com/us/app/super-planner-event-planning/id383727111
4. ISO/IEC/IEEE12207:2017(E). Systems and software engineering – Software life cycle processes.2017.
5. Kruchten P. Architectural Blueprints–The “4+1” View Model of Software Architecture. IEEE Software. 1995. November. P. 2–12.
6. Evans E. Domain-Driven Design References. Definitions and Pattern Summaries. Domain Language, Inc., 2015.
7. Офіційний сайт Node.js [Електронний ресурс]

   nodejs.org/uk/
8. Сторінка body-parser на сайті npm [Електронний ресурс]

   npmjs.com/package/body-parser
9. Сторінка cookie-parser на сайті npm [Електронний ресурс]

   npmjs.com/package/cookie-parser
10. Сторінка CORS на сайті npm [Електронний ресурс]

    npmjs.com/package/cors
11. Сторінка jsonwebtoken на сайті npm [Електронний ресурс]

    npmjs.com/package/jsonwebtoken
12. Сторінка express-jwt на сайті npm [Електронний ресурс]

    npmjs.com/package/express-jwt

13.Сторінка pg на сайті npm [Електронний ресурс]
npmjs.com/package/pg
14. Офіційний сайт React [Електронний ресурс]
uk.reactjs.org/
15.Офіційний сайт Next.js [Електронний ресурс]
nextjs.org/
16. Офіційний сайт Material UI [Електронний ресурс]
material-ui.com/
17. Огляд версії Next.js 9.5 [Електронний ресурс]
nextjs.org/blog/next-9-5

Додаток А
(обов'язковий)
Анкета-опитування працівників компанії

Частина 1. Актуальна структура зберігання даних в компанії.

Оцініть своє ставлення до наступних фактів від 1 до 10, де 1 – абсолютно не погоджуюсь, 10 – повністю погоджуюсь.

1. Я завжди маю доступ до даних. __
2. Мені зручно вносити нові та оновлювати старі дані. __
3. Мені зручно шукати дані за заданими характеристиками. __
4. Інформація завжди є актуальною на момент пошуку. __
5. Інформація коректна та не повторюється. __

Дайте розгорнуту відповідь на наступні запитання:

1. З якими операціями над даними Вам найчастіше доводиться мати справу?

_____

_____

2. Що Вам подобається в поточній інформаційній системі?

_____

_____

3. Що Вам НЕ подобається в поточній інформаційній системі?

_____

_____

Частина 2. Власний досвід користування інформаційними системами.

Дайте розгорнуту відповідь на наступні запитання:

1. Які критерії Ви вважаєте найважливішими у досвіді користування базою даних?

_____

_____

_____

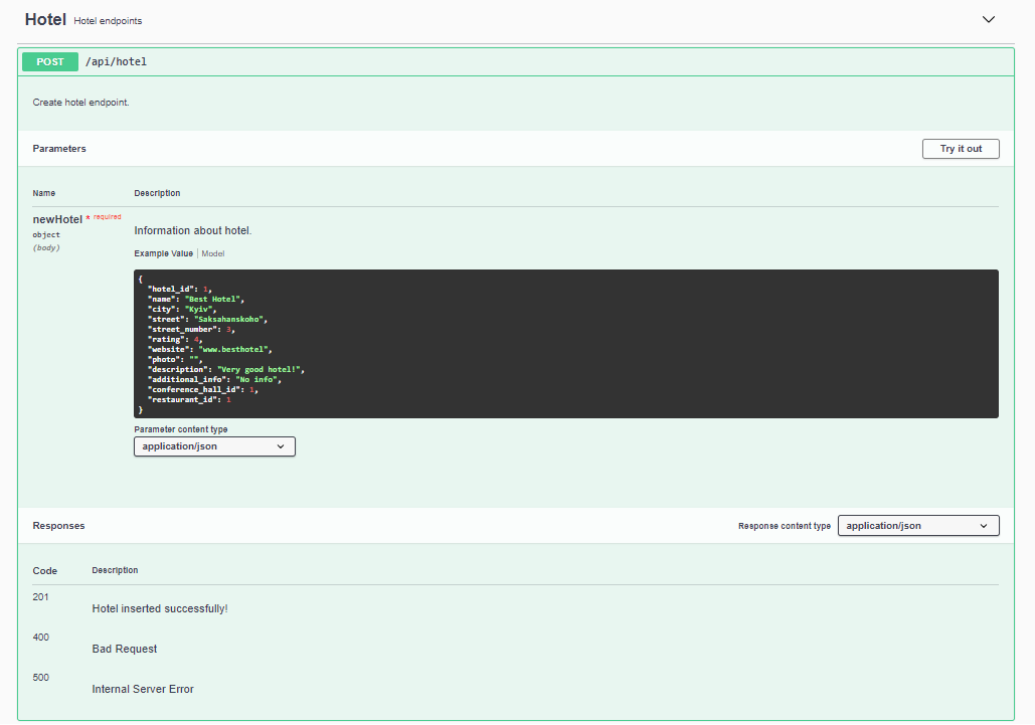
2. Який додатковий функціонал, окрім безпосередніх операцій з даними, для Вас важливий?

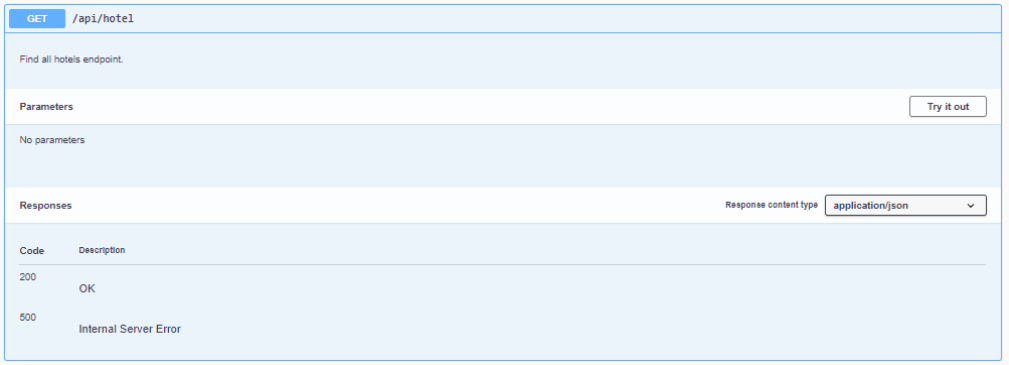
_____

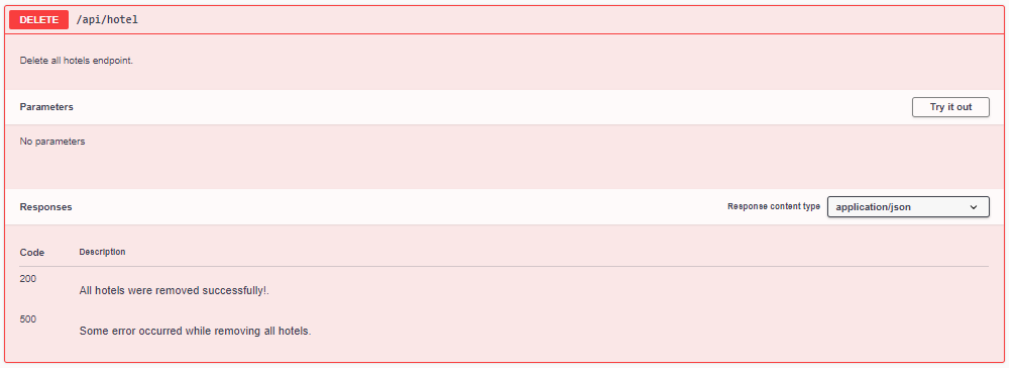
_____

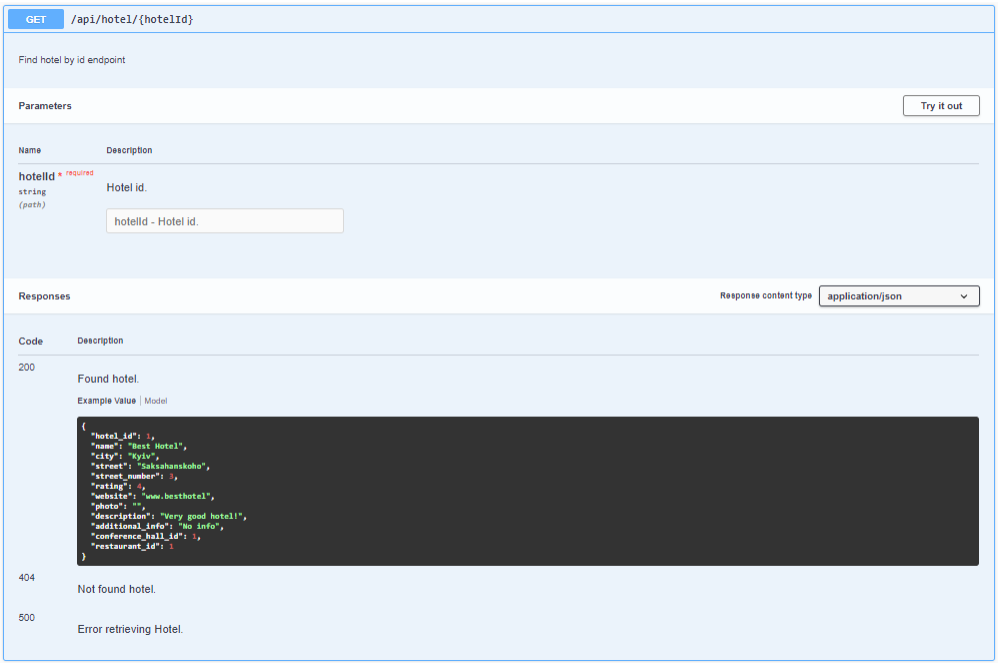
_____

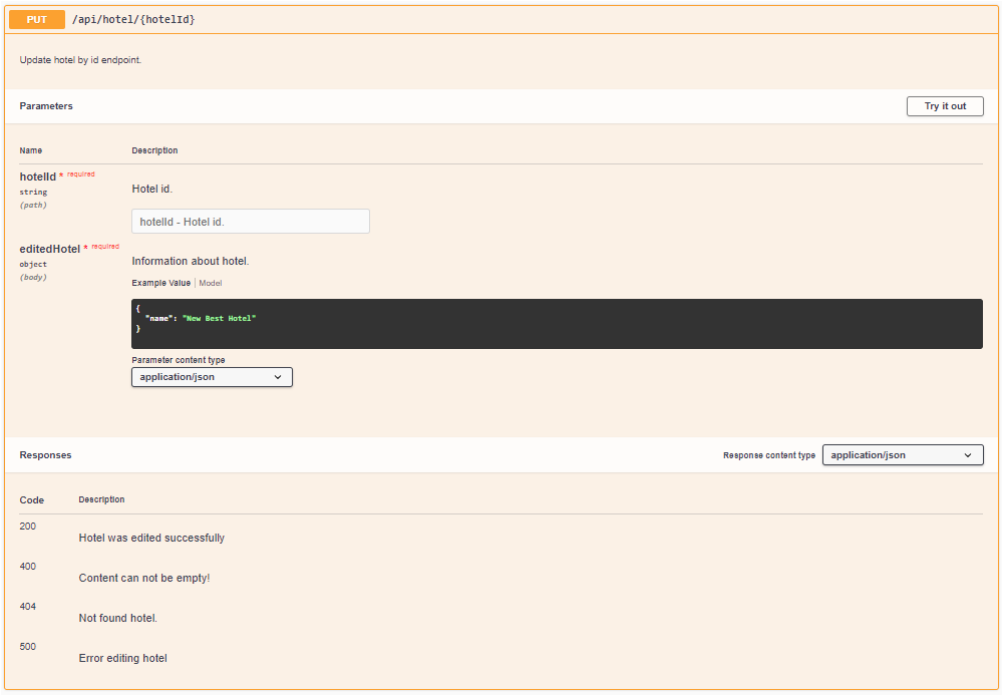
Додаток Б
(обов'язковий)
Документації API для готелів

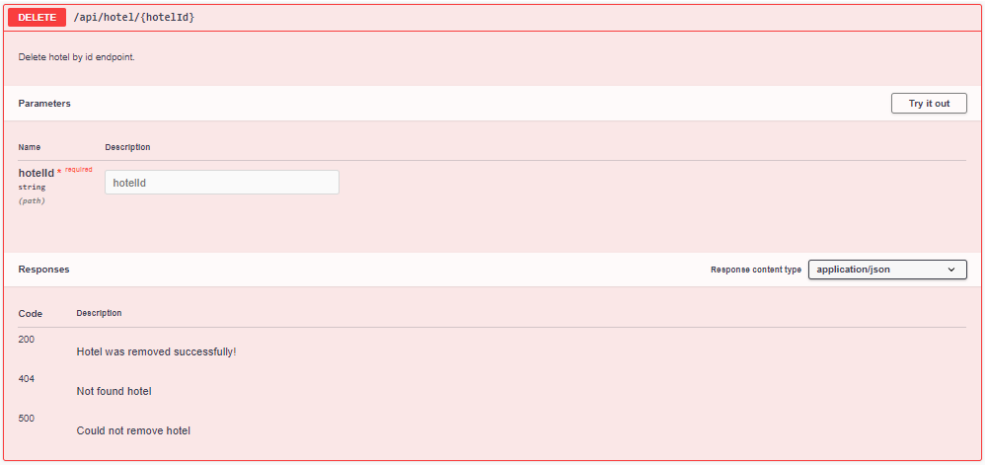




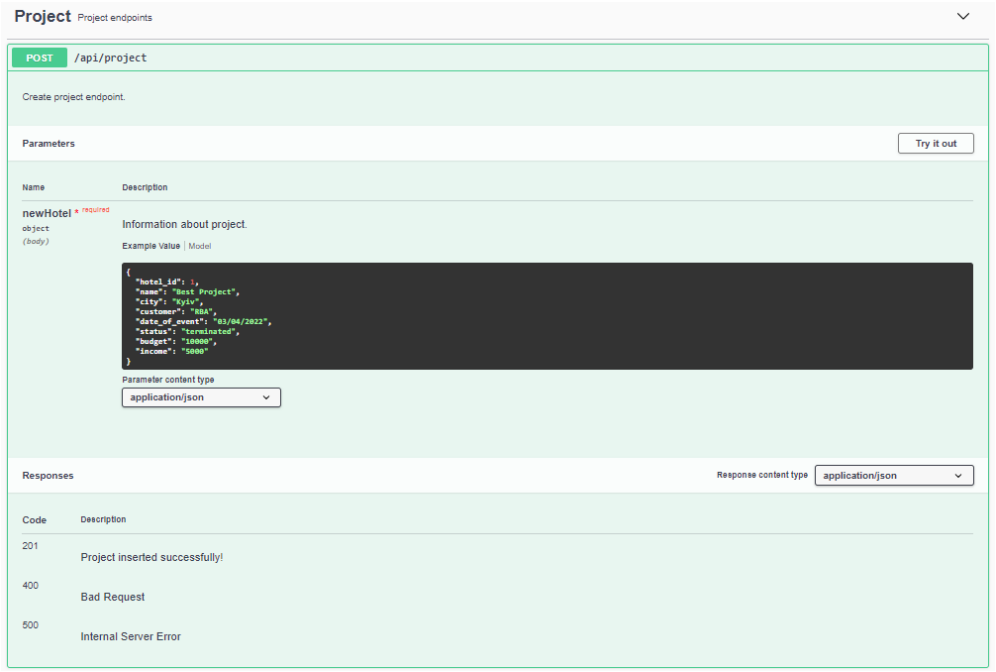

DELETE
/api/hotel
Delete all hotels endpoint.
Parameters
Try it out
No parameters
Responses
Response content type
application/json
Code
Description
200
All hotels were removed successfully!.
500
Some error occurred while removing all hotels.

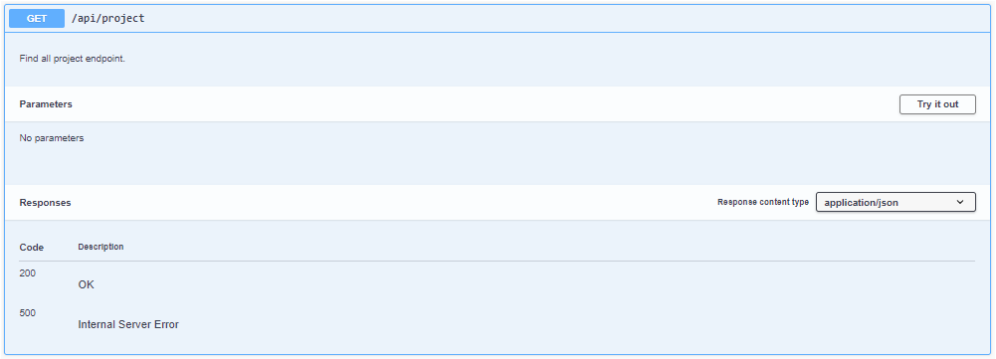

GET
/api/hotel/{hotelId}
Find hotel by id endpoint
Parameters
Try it out
Name
Description
hotelId
* required
string
(path)
Hotel id.
hotelId - Hotel id.
Responses
Response content type
application/json
Code
Description
200
Found hotel.
Example Value
Model
{
"hotel_id": 1,
"name": "Best Hotel",
"city": "Kyiv",
"street": "Saksahanskoho",
"street_number": 3,
"rating": 4,
"website": "www.besthotel",
"photo": "",
"description": "Very good hotel!",
"additional_info": "No info",
"conference_hall_id": 1,
"restaurant_id": 1
}
404
Not found hotel.
500
Error retrieving Hotel.

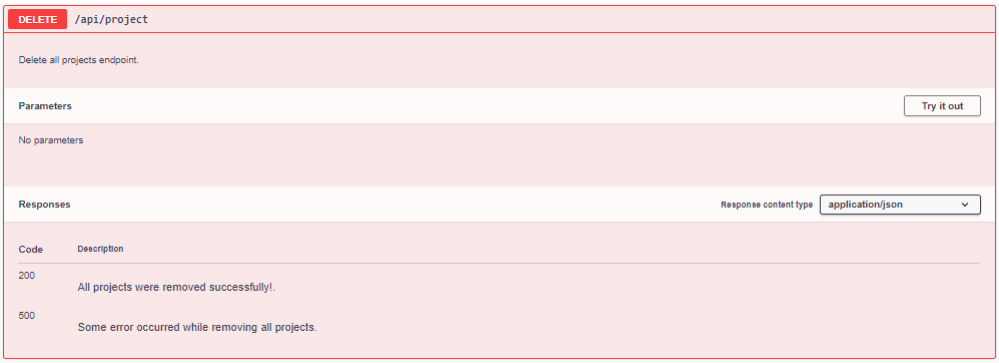

PUT
/api/hotel/{hotelId}
Update hotel by id endpoint.
Parameters
Try it out
Name
Description
hotelId * required
string
(path)
Hotel id.
hotelId - Hotel id.
editedHotel * required
object
(body)
Information about hotel.
Example Value | Model
{
"name": "New Best Hotel"
}
Parameter content type
application/json
Responses
Response content type
application/json
Code
Description
200
Hotel was edited successfully
400
Content can not be empty!
404
Not found hotel.
500
Error editing hotel

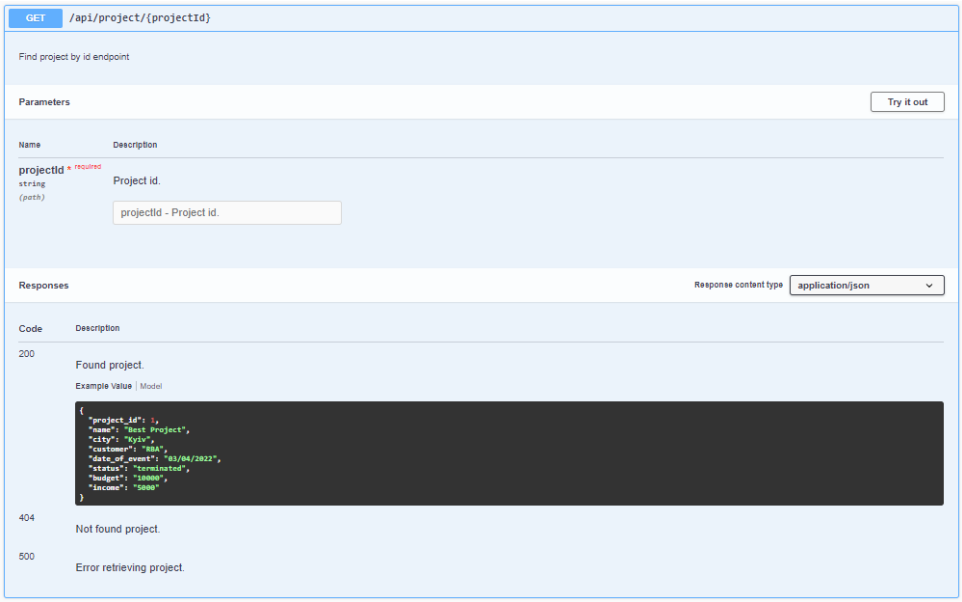

DELETE
/api/hotel/{hotelId}
Delete hotel by id endpoint.
Parameters
Try it out
Name
Description
hotelId * required
string
(path)
hotelId
Responses
Response content type
application/json
Code
Description
200
Hotel was removed successfully!
404
Not found hotel
500
Could not remove hotel

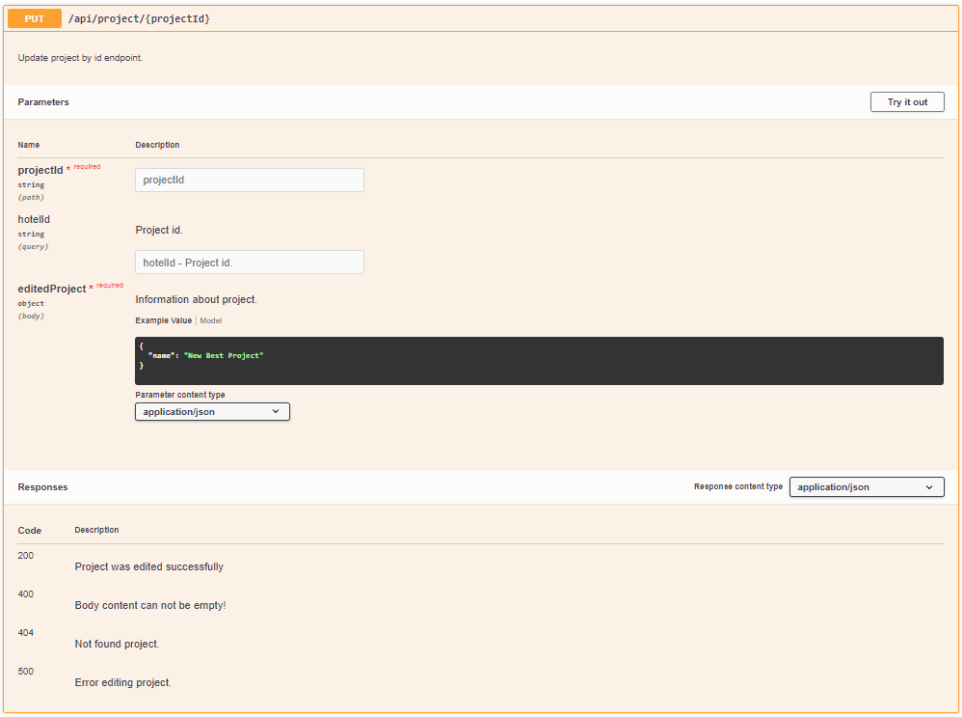
Додаток В
(обов'язковий)
Документації API для проектів

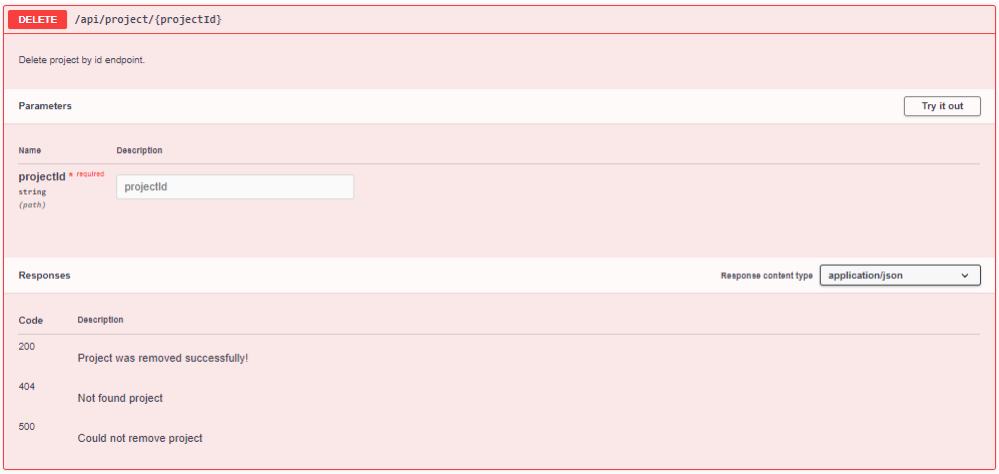







GET
/api/project/{projectId}
Find project by id endpoint
Parameters
Try it out
Name
Description
projectId * required
string
(path)
Project id.
projectId - Project id.
Responses
Response content type
application/json
Code
Description
200
Found project.
Example Value | Model
{
"project_id": 1,
"name": "Best Project",
"city": "Kyiv",
"customer": "RBA",
"date_of_event": "03/04/2022",
"status": "terminated",
"budget": "10000",
"income": "5000"
}
404
Not found project.
500
Error retrieving project.


PUT
/api/project/{projectId}
Update project by id endpoint.
Parameters
Try it out
Name
Description
projectId * required
string
(path)
projectId
hotelId
string
(query)
Project id.
hotelId - Project id.
editedProject * required
object
(body)
Information about project.
Example Value | Model
{
"name": "New Best Project"
}
Parameter content type
application/json
Responses
Response content type
application/json
Code
Description
200
Project was edited successfully
400
Body content can not be empty!
404
Not found project.
500
Error editing project.


DELETE
/api/project/{projectId}
Delete project by id endpoint.
Parameters
Try it out
Name
Description
projectId * required
string
(path)
projectId
Responses
Response content type
application/json
Code
Description
200
Project was removed successfully!
404
Not found project
500
Could not remove project