Державний вищий навчальний заклад
“Прикарпатський національний університет імені Василя Стефаника”
Кафедра інформаційних технологій

УДК 004

ДИПЛОМНИЙ ПРОЕКТ

Тема Розробка вебсервісу для бронювання ресурсів

Спеціальність 121 Інженерія програмного забезпечення
код і назва спеціальності

ПОЯСНЮВАЛЬНА ЗАПИСКА
ДП.ІПЗ-23.ПЗ
(позначення)

Рецензент
ст. викл. Савка І. Я.
(посада) (підпис) (дата) (розшифровка підпису)

Студент
ІПЗ-41 Ратушний О.Л.
(шифр групи) (підпис) (дата) (розшифровка підпису)

Нормоконтролер
ст. викл. Савка І. Я.
(посада) (підпис) (дата) (розшифровка підпису)

Керівник дипломного проекту
доц. Ткачук В.М.
(посада) (підпис) (дата) (розшифровка підпису)

Допускається до захисту

Завідувач кафедри

Козленко М.І.
(посада) (підпис) (дата) (розшифровка підпису)

2020
(рік)

Державний вищий навчальний заклад
«Прикарпатський національний університет імені Василя Стефаника»
Факультет математики та інформатики Кафедра інформаційних технологій
Спеціальність 121 Інженерія програмного забезпечення

ЗАТВЕРДЖУЮ:
Завідувач кафедри Козленко М.І.
студенту Ратушному Олексію
Любомировичу
„_____”_____ 20___р.

ЗАВДАННЯ
НА ВИКОНАННЯ ДИПЛОМНОГО ПРОЕКТУ

Студенту Ратушному Олексію Любомировичу
(прізвище, ім’я, по батькові студента)
1. Тема проекту Розробка вебсервісу для бронювання ресурсів
затверджена розпорядженням по факультету математики та інформатики від „25” жовтня 2019 р.№7
2. Термін здачі студентом закінченого проекту 22 травня 2020 р.
3. Вихідні дані до дипломного проекту категорії систем онлайн бронювання, технологія програмування серверної частини – .NET Core, технології програмування клієнтської частини – HTML, TypeScript, Angular,
4. Зміст пояснювальної записки (перелік питань, що їх належить опрацювати)
1. Аналіз предметної області
2. Моделювання програми
3. Програмна реалізація
4. Бізнес-план
5. Перелік графічного матеріалу (з точним забезпеченням обов’язкових креслень)
6. Дата видачі завдання _____

Керівник _____ Ткачук В.М.
(підпис) (розшифровка підпису)

Завдання прийняв до виконання _____ Ратушний О.Л.
(підпис) (розшифровка підпису)

КАЛЕНДАРНИЙ ПЛАН

Номер і назва етапів дипломного проекту	Термін виконання етапів проекту	Примітка
1. Аналіз предметної області	22.10.2019	виконав
2. Моделювання системи	17.01.2020	виконав
3. Програмна реалізація	05.03.2020	виконав
4. Бізнес план	07.05.2020	виконав
5. Оформлення пояснювальної записки	18.05.2020	виконав

Студент _____Ратушний О.Л.

(підпис) (розшифровка підпису)

Керівник проекту _____Ткачук В. М.

(підпис) (розшифровка підпису)

РЕФЕРАТ

Пояснювальна записка: 56 сторінок (без додатків), 38 рисунків, 20 джерел, 1 додатку на 1 сторінці.

Ключові слова: ОНЛАЙН БРОНЮВАННЯ, ВЕБСАЙТ, C#, ANGULAR, ENTITY FRAMEWORK CORE, MS SQL SERVER.

Об’єктом дослідження є вебсайт для онлайн бронювання.

Мета роботи: створення вебсайту для бронювання, що буде універсальним для будь-якого типу товару.

Стислий опис тексту пояснювальної записки:

Протягом виконання дипломного проекту було досліджено предметну область та оглянено існуючі системи. Визначено вимоги до програмного забезпечення. Проведено моделювання системи та бази даних. Реалізовано вебсайт для бронювання, що буде універсальним для будь-якого ресурсу. Проведено економічне обгрунтування розробки.

ABSTRACT

Explanatory note: 56 pages (without appendix), 38 figures, 20 references, 1 appendix on 1 page.

Key words: ONLINE BOOKING, WEBSITE, C#, ANGULAR, ENTITY FRAMEWORK CORE, MS SQL SERVER.

Object of study: a website for online booking.

Brief description of the text of the explanatory note:

During the implementation of the diploma project the subject area was researched and the existing systems were inspected. Software requirements are defined. The system and database are modeled. Implemented a website for booking, which will be universal for any resource. The economic substantiation of development is carried out.

ЗМІСТ

ВСТУП

В сучасному світі люди можуть купити чи замовити багато різних речей, як квитки на якусь подію, книги, одяг і так далі. Все частіше люди роблять це онлайн і часто виникає потребі не одразу купити чи замовити певний продукт, а просто забронювати його. Особливо часто бронювати потрібно квитки на якусь поїздку чи подію або наперед забронювати певний товар, що швидко розпродається.

Зараз все більше речей переходить в онлайн, тому актуальність розробки програми, що дозволяє бронювати якусь сутність тільки зростає. Це необхідно не тільки споживачу товару, але в першу чергу тому, хто пропонує товар.

Створення вебсайту для бронювання, що буде універсальним для будь-якого типу товару є метою даної роботи.

Щоб досягти мети треба виконати такі завдання:

- здійснити аналіз предметної області;
- скласти вимоги до програми;
- створити зручний інтерфейс для користувача;
- розробити базу даних.

Об’єктом дослідження є системи бронювання.

Предметом дослідження є сайт для бронювання.

Роботу виконано на базі .NET Core 2.3 [1] та Angular [2], база даних - Microsoft SQL Server з використання T-SQL процедур та функцій [3].

Практичним результатом даної роботи є вебсайт, що можна використовувати для бронювання будь-яких продуктів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Онлайн бронювання

Із сучасним темпом життя все більше послуг та товарів пропонують через інтернет, адже це зручно як клієнту, так і продавцю. Часто виникає потреба не одразу придбати той чи інший товар, а для початку просто забронювати його для подальшого придбання.

Онлайн-бронювання – бронювання, що відбуваються через мережу Інтернет, в інтерактивному режимі. Найбільш часто бронювання застосовують до номерів в готелях, проїзних квитків, квитків на різного виду події чи взяття речей на прокат [4]. Але насправді застосовувати це можна до будь-якої сутності.

Загально відомі рішення базуються на бронюванні певної специфічної сутності і є не самостійними рішенням, а частиною вебсервісу відповідних компанії. Відповідно кожна компанія використовує своє специфічне рішення. Загальним принципом таких систем є:

1. Пошук доступних пропозицій за наданими критеріями.
2. Занесення контактних та платіжних даних.
3. Здійснення платежу.
4. Одержання документального підтверджує бронювання.

Зазвичай сервіси бронювання розробляються окрему компанію та її специфіку (рис. 1.1). Наприклад, відомий сайт Booking.com[5] - це система інтернет-бронювання житла. Суть цієї системи полягає тільки в бронюванні житла по всьому світі.

Інколи компанія може займатись будь-якою іншою діяльністю і надавати можливість бронювання їх ресурсів, як додатковий функціонал. Прикладом такого сайту є booking.uz.gov.ua [6]. В таких випадках системи бронювання написані тільки під конкретний продукт (Рис. 1.2).

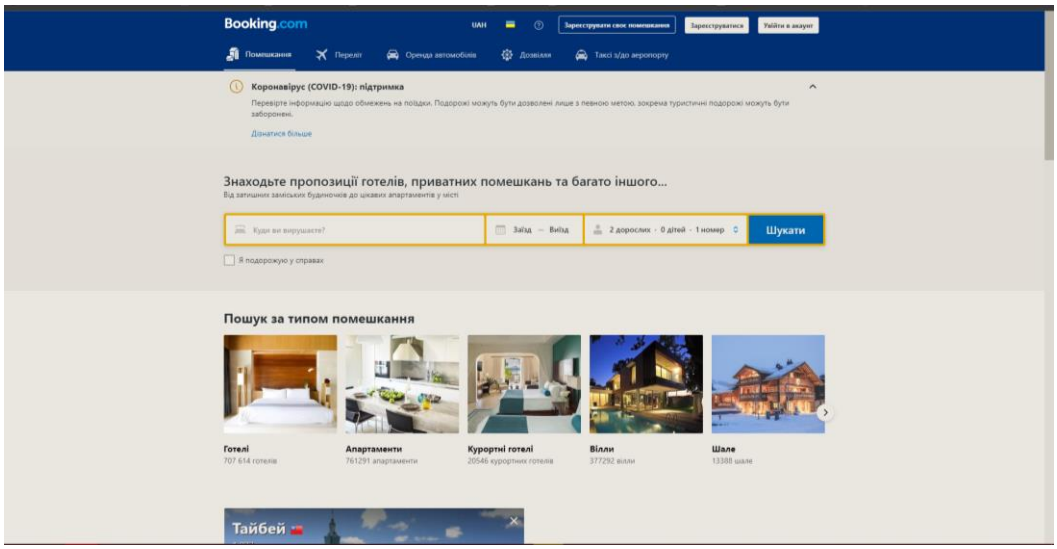


Рис. 1.1 – Сайт booking.com

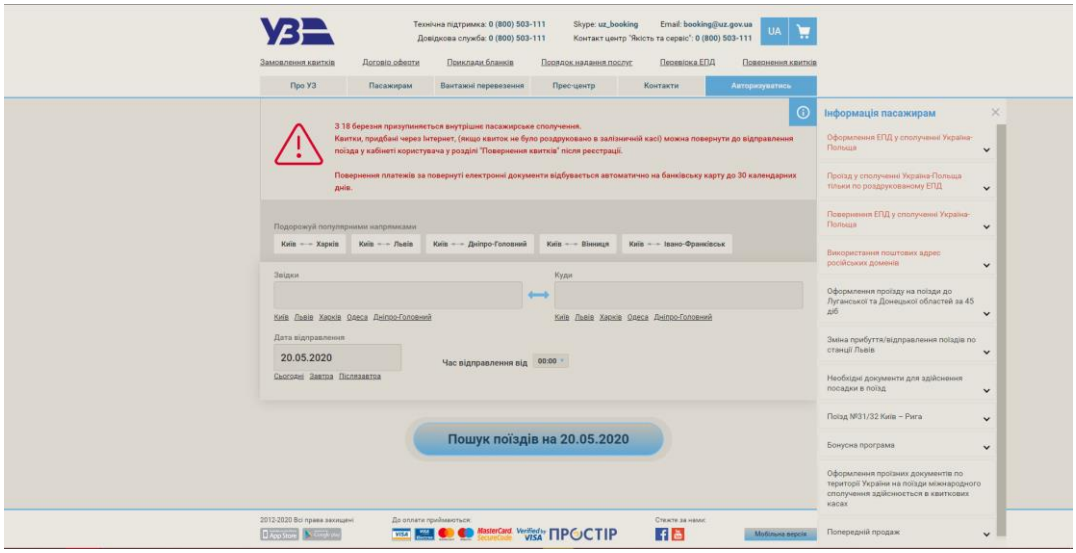


Рис. 1.2 – Сайт booking.uz.gov.ua

Відмінною рисою та перевагою продукту, що розробляється в межах даної роботи – уніфікованість системи. В даній роботі розроблено уніфікований підхід до бронювання різноманітних ресурсів і не має прив'язки до оплати, так як це може бути і сервіс для внутрішнього користування іншими компаніями. Дане рішення позиціонує себе як сервіс для компаній які хочуть швидко і ефективно розгорнути систему бронювання. Також такий функціонал також буде в нагоді

компаніям з офісами для того, щоб ефективно управляти тими чи іншими приміщеннями. Дана система надає можливість створення різнотипних сутностей для бронювання та виставлення правил бронювання для них.

1.2 Специфікація вимог

Перед розробкою програмного забезпечення було написано специфікацію вимог до програмного забезпечення.

1.2.1 Призначення

Сайт, що потрібно розробити, призначений для онлайн бронювання, без прив’язки до конкретної сутності.

1.2.2 Угоди, прийняті в документах

При розробці даної програми прийнято дотримуватись таких угод:

- дотримання стилю коду мов програмування;
- залишати коментарі до коду, де це необхідно на розсуд програміста;
- дотримання принципу багаторівневої архітектури;
- розділення програми на серверну частину і частину користувача;
- використовувати в дизайні тони блакитного, червоного та зеленого кольорів;
- використання шрифту та одного стилю.

1.2.3 Передбачувана аудиторія

Специфікація вимог до програмного забезпечення розрахована для програмістів та дизайнерів, що розроблятимуть програму.

Документ складається наступних розділів:

- введення;
- загальний опис;
- функції системи;
- вимоги до зовнішніх інтерфейсів;
- нефункціональні вимоги.

1.2.4 Границі проекту

Сайт для онлайн бронювання призначений для бронювання продуктів чи послуг будь-якого типу. Основними користувачами є компанії, що хочуть надати можливість бронювати їх послуги.

Програма не має обмежень, пов’язаних і статтю людини, національністю чи родом діяльності. Вік юзера, що зможе здійснювати бронювання обмежується згідно політики того, хто саме і які саме послуги дозволяє бронювати.

Мова, яку підтримує сайт – англійську.

1.2.5 Загальний погляд на продукт

Дана програма надає можливість створення різнотипних сутностей для бронювання та виставлення правил бронювання для них. Продукт розробляється «з нуля» і є новим.

1.2.6 Особливості продукту

Особливістю продукту є можливість його застосування до будь-якої сутності, товару чи послуг будь-якої організації.

Основними функціями вебсайту є:

- бронювання ресурсів;
- скасування бронювання;
- менеджмент ресурсів та правил;
- реєстрація та авторизація користувачів;
- виведення статистики для адміністраторів;
- менеджмент користувачів;
- групування ресурсів в ієрархічну систему з директоріями.

1.2.7 Класи і характеристики користувачів

Користувачі програми поділяються на такі класи:

- адміністратор – керує ресурсами та правилам для них, а також директоріями для ресурсів, створює, блокує, підтверджує реєстрацію для користувачів;
- користувач - може бронювати ресурс, скасовувати бронювання, змінювати ім’я та переглядати історію бронювання.

1.2.8 Операційне середовище

Апаратними засобами для використання сайту є пристрій, що містить браузер та доступ до інтернету.

Дані зберігаються у базі даних MS SQL Server.

1.2.9 Обмеження дизайну та реалізації

Програма розробляється з використанням гіпертекстової розмітки сторінки HTML, мов програмування JavaScript, C Sharp та скриптової мови Sass.

Для дизайну використовується “Bootstrap” з елементами матеріал дизайну.

1.2.10 Припущення і залежності

Можливі зміни у відображеннях компонентів сайту на пристроях з різними розміром екрану.

1.2.11 Опис і пріоритети

Розробка даного програмного забезпечення ґрунтується на принципах об’єктно-орієнтовного програмування.

Дані користувача повинні бути захищенні хешуванням.

Першим чином розробляється бекенд програми, потім авторизація, сторінка бронювання ресурсів, кабінет користувача, головна сторінка та кабінет адміністратора.

1.2.12 Послідовність «вплив-реакція»

Програма чітко реагує на натиски кнопок та переході між сторінками. Інтерфейс користувача дозволяє чітко прослідкувати послідовність між його діями та реакціями програми.

1.2.13 Функціональні вимоги

При взаємодії з користувача виглядом програми, функції зі сторони бекенду повинні відповідати функціям на стороні клієнта. Обов’язкова наявність можливості бронювати ресурс та скасувати бронювання. Перегляд статистики для адміністраторів.

1.2.14 Інтерфейси користувача

Дизайн інтерфейсу користувача розроблено згідно стандарту поєднання кольорів. Дії користувача та їх результати супроводжуються підказками та повідомленнями.

1.2.15 Програмні інтерфейси

Реалізація відбувається з використанням програмних інтерфейсів:

- .NET Core 2.1.
- EF core 6.
- Angular 5.
- JSON.

1.2.16 Інтерфейси обладнання

Інтерфейси комп’ютерних та мобільних пристроїв. З використання браузера який підтримує ECMAScript 2017 для повного функціонування всіх компонентів.

1.2.17 Інтерфейси зв'язку і комунікацій

Зв’язок із користувачем відбувається через оповіщення та підказки для орієнтації в інтерфейсі.

1.2.18 Вимоги до продуктивності

Швидкість відповіді на запит користувача повинна бути в межах 1-3 секунд в залежності від якості інтернет з’єднання.

1.2.19 Вимоги до збереженості даних

Зберігання даних користувача – база даних MS SQL Server. Пароль користувача повинен зберігатись у хешованому вигляді. Пошта кожного користувача унікальна. Користувач не може видалити інформацію про бронювання із бази даних. Ресурси, правила і директорії видаляти може лише адміністратор.

1.2.20 Критерії якості програмного забезпечення

Продукт повинен бути якісним і відповідати таким критеріям:

- Надійність.
- Ефективність.
- Зручність.
- Переносність.
- Функційність.
- Супроводжуваність.

1.2.21 Вимоги до безпеки системи

Код програми не може бути доступний користувачу. Обробка цінних даних відбувається тільки на бекенді. Права користувача та адміністратора чітко розділені. Дані програми не можуть бути у відкритому доступі чи у доступі не зареєстрованих користувачів. Не авторизований користувач має право лише переглядати стан бронювання ресурсів, при чому без відображення хто і з яким поясненням забронював ресурс, також користувач не бачить деталей чужого бронювання, адміністратор бачить всі дані всіх бронювання.

2 МОДЕЛЮВАННЯ ПРОГРАМИ

2.1 Загальний опис та модель роботи системи

Дане програмне забезпечення надає можливість гнучкого бронювання різноманітних сутностей. Архітектура даної системи дозволяє надавати послуги бронювання стороннім клієнта, що дозволяє розширити ринки збуту системи. Для адміністраторів та модераторів надається можливість налаштовувати правила бронювання. Також є можливість групувати ресурси за ієрархічними категоріями, що дозволяє більш ефективно використовувати дану систему. Гнучка система правил дозволяє ефективно контролювати бронювання.

Діаграма можливостей [7] різних користувачів в системі зображена на рисунку 2.1.

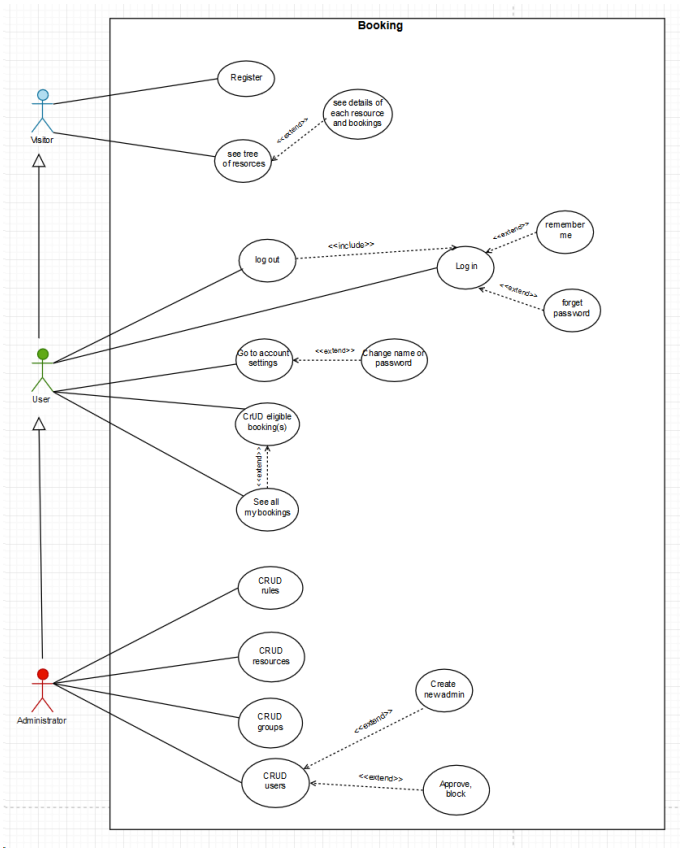


Рис 2.1 – Use case діаграма системи

Не авторизовані користувачі будуть мати можливість переглянути бронювання, але без інформування про користувача, який забронював ресурс. Також адміністратор буде мати можливість переглянути статистику бронювань.

Авторизований користувач зможе забронювати ресурс скасувати своє бронювання та переглянути існуючі бронювання, а ще буде мати можливість зміни свій логін та пароль.

Адміністратор буде мати наступні можливості:

- створювати користувача та видаляти користувача;
- підвереджувати та відхиляти реєстрацію користувача;
- блокувати та розблоковувати користувача;
- перегляди бронювання всіх користувачів та окремо кожного;
- переглядати бронювання ресурсів;
- створювати, редагувати та видаляти правила;
- створювати, редагувати та видаляти ресурси;
- створювати та видаляти директорії також і редагувати їх;
- всі можливості користувача.

Відповідно всі зареєстровані користувачі поділяються на дві ролі:

- користувач;
- адміністратор.

Також користувач може бути не авторизованим, тому з точки зору функціоналу є ще третя роль - гість.

2.2 Аутентифікація та авторизація

Авторизація та аутентифікація користувача реалізована з використанням двох JWT [8] токенів:

- Authorization token – авторизаційний токен, це набір зашифрованих даних які ідентифікують користувача;
- Refresh token [9] – токен оновлення, одноразовий токен який зберігається на сервері і дозволяє ідентифікувати користувача при завершені дії авторизаційного токена.

Авторизаційний токен має тривалість дії, це зумовлено питанням безпеки користувача. Суть методу в компромісі між постійною авторизацією на сервері, що є достатньо затратною задачею, і використанням одного і того ж токену, що дозволило б третій стороні перехопити його і виконувати дії від імені користувача. На рисунку 2.2 показано схему роботи авторизації з використанням двох токенів.

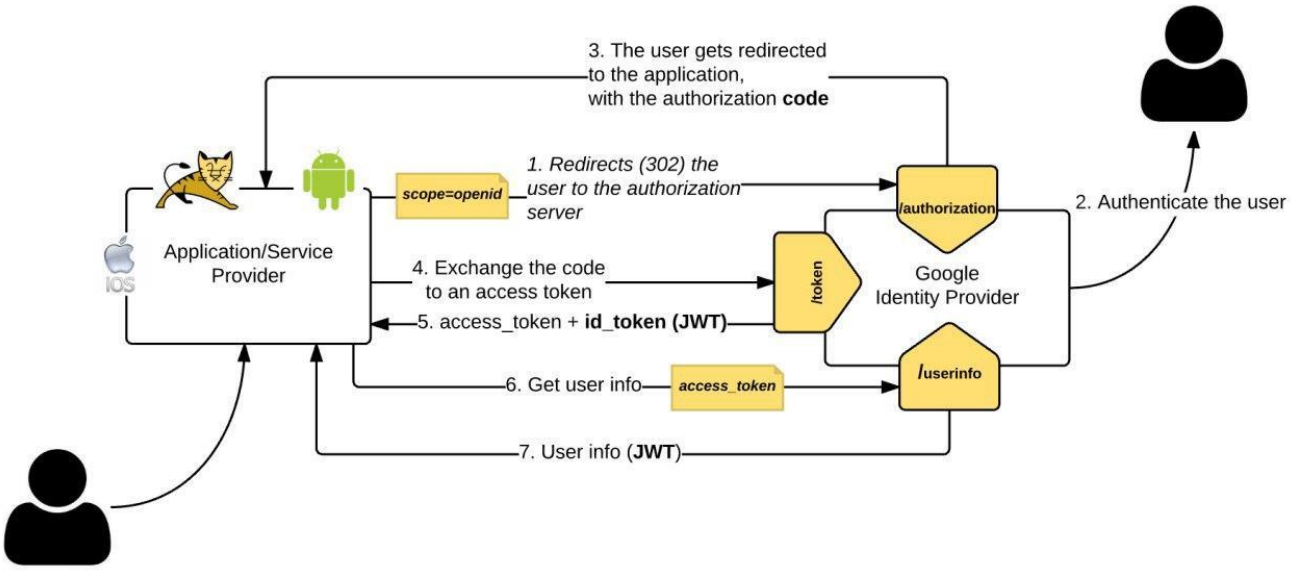


Рис 2.2 – Діаграма роботи авторизації на основі JWT токенів

В загальному дана система працює за наступним принципом:

1. Користувач вводить логіні і пароль та відправляє дані на сервер.
2. Сервер перевіряє дані і, якщо дані коректні, видає авторизаційний токен та токен оновлення і записує токен оновлення в базу даних.
3. Авторизаційний токен використовується при кожному запиті на сервер.

4. Після завершення дії авторизаційного токену, сервер інформує систему, що токен некоректний.
5. Клієнт відправляє токен оновлення, який сервер порівнює з існуючим в базі даних. Якщо знаходиться збіг, то система видає користувачу нову пару токенів, видаляє старий токен оновлення з бази даних та заносить новий взамін.

2.3 Модель бази даних

Передбачається, що користувачі зможуть реєструватись, а адміністратор буде підтверджувати реєстрацію. На даний момент дане рішення вважається достатнім. В майбутньому дану систему можна буде інтегрувати з зовнішніми сервісами авторизації, як SAML [10] SSO [11] або іншими сторонніми провайдерами авторизації.

База даних проектувалась з урахуванням того, що система буде розгорнута на технологічному стеку компанії Microsoft, тому при моделюванні були враховані сутності які створюються компонентами технологій які використовуються в системі.

Всі постійні дані зберігаються в базі даних з використанням схеми, зображеної на рисунку 2.3, що складається із наступних таблиць:

- AspNetUser – інформація про користувача;
- AspNetUserTokens – токени користувачів;
- AspNetRoles – список ролей користувачів;
- AspNetUserLogins – список логінів користувачів;
- AspNetRoleClaims – список клеймів для ролей;
- AspNetUserClaims – список клеймів для користувачів;
- Resources – список ресурсів, які можна бронювати;
- Bookings – історія бронювання;

- Rules – список правил для ресурсів;
- Folders – список категорій ресурсів.

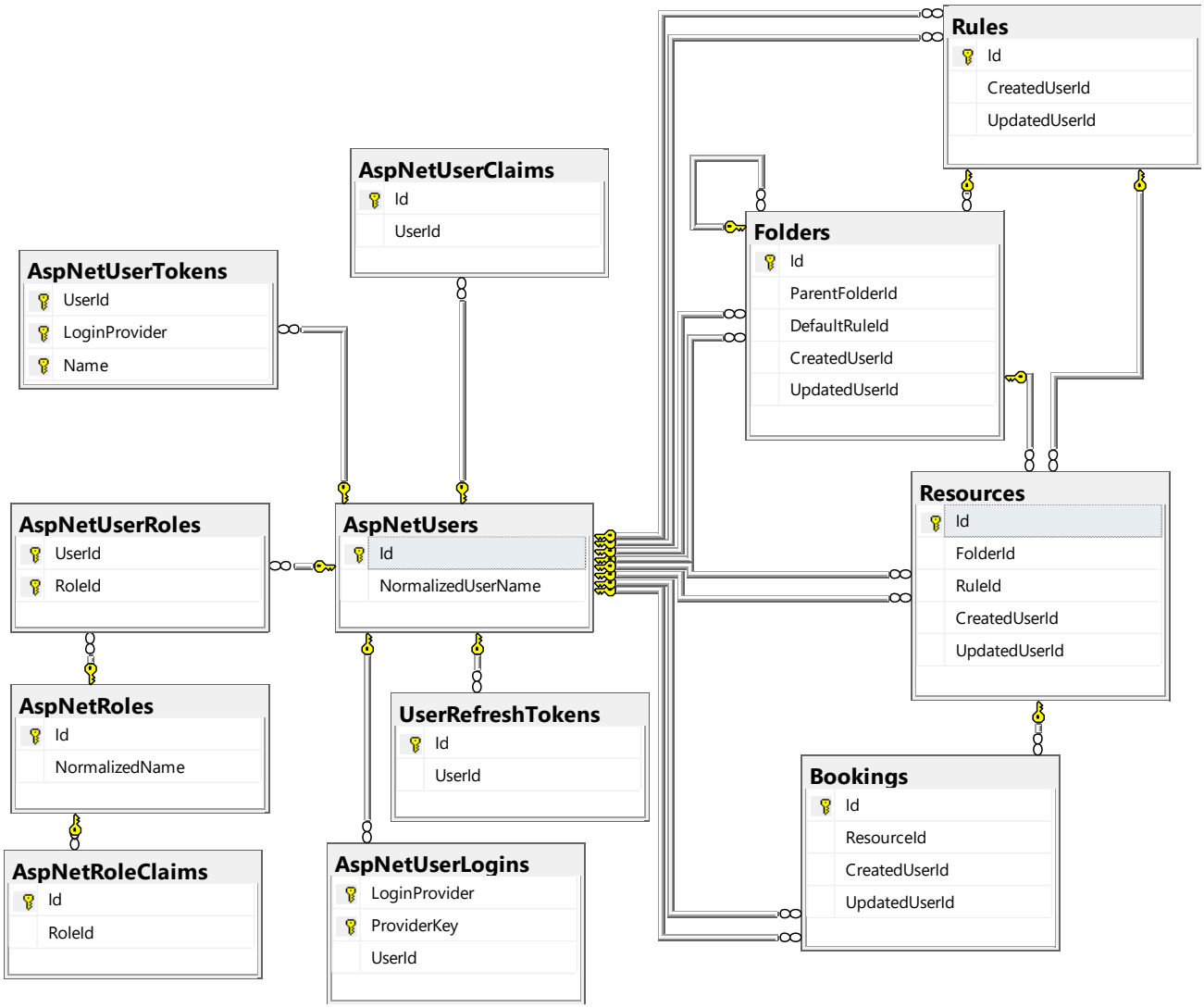


Рис. 2.3 – Схема бази даних системи бронювання

Таблиця які починаються на «AspNet» надаються системою Asp .Net Identity [12] і були створені автоматично.

Дана система працює з бронюванням, але бронювання самі по собі не існують. Бронюється якась сутність, наприклад готель, літак, велосипед, тощо. Для уніфікації системи сутність, яку бронюють іменується ресурсом. Кожен ресурс містить опис, ідентифікатор, та номер правила бронювання. Враховуючи

все різноманіття ресурсів, можливість їхньої класифікації, було вирішено створити окрему сутність правила бронювання. Її структуру можна побачити на рисунку 2.4.

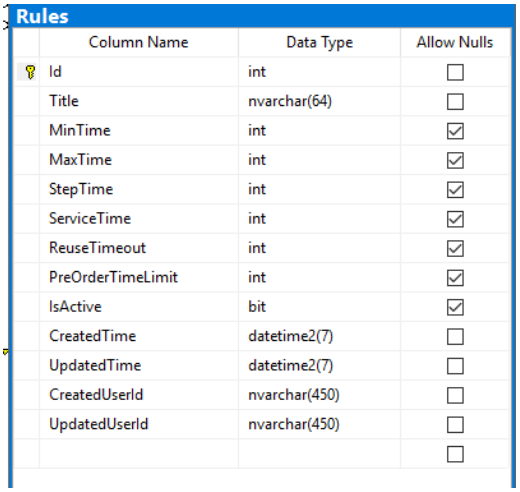


Рис. 2.4 – Схема таблиці правил бронювання

Дана сутність містить наступні політики (правила) бронювання:

- MinTime – мінімальний час бронювання;
- MaxTime – максимальний час бронювання;
- StepTime – крок бронювання, більшість ресурсів можна забронювати на певні дискретну тривалість часу. Наприклад, готель можна забронювати на день чи декілька, але не на 8 годин 3 хвилин і 17 секунд. Для того, щоб уникнути можливих проблем з тривалістю бронювання було вирішено ввести дану політику обмеження;
- ServiceTime – час обслуговування, деякі ресурси такі, як готель чи літак після використання мають бути підготовленні до наступного використання. Враховуючи уніфікацію даної системи було введено дану політику обмеження;

- ReuseTime – час, протягом якого той же користувач не має права повторно бронювати ресурс. Дана політика введена для уникнення спекуляцій, які є небажаними для менеджерів ресурсів;
- PreOrderTimeLimit – час, за який як максимум можна забронювати ресурс. Більшість ресурсів мають певне обмеження по часу бронювання на перед. Логічно, що не можливо за бронювати готель через 100 років. Відповідно для уникнення подібних ситуацій було введено дану політику обмеження.

Окрім правил для зручнішої класифікації ресурсів було введено поняття директорій. Як групи ресурсів та інших директорій що дозволяє створювати ієрархічну систему ресурсів. Детальну схему даної сутності можна побачити на рисунку 2.5.

Folders

Column Name	Data Type	Allow Nulls
Id	int	☐
Title	nvarchar(64)	☐
ParentFolderId	int	☑
DefaultRuleId	int	☑
IsActive	bit	☑
CreatedTime	datetime2(7)	☐
UpdatedTime	datetime2(7)	☐
CreatedUserId	nvarchar(450)	☐
UpdatedUserId	nvarchar(450)	☐
		☐

Рис. 2.5 – Схема сутності директорій ресурсів

Важливою деталлю є те, що директорії дозволяє мати стандартне правило, що є зручним при частому створенні ресурсів. Також директорія містить посилання на батьківську директорію, але це при умові, що директорія не є кореневою.

Запис про бронювання, окрім часу початку та кінця, і користувача який забронював, також містить запис про час скасування, якщо користувач скасував бронювання (Рис. 2.6).

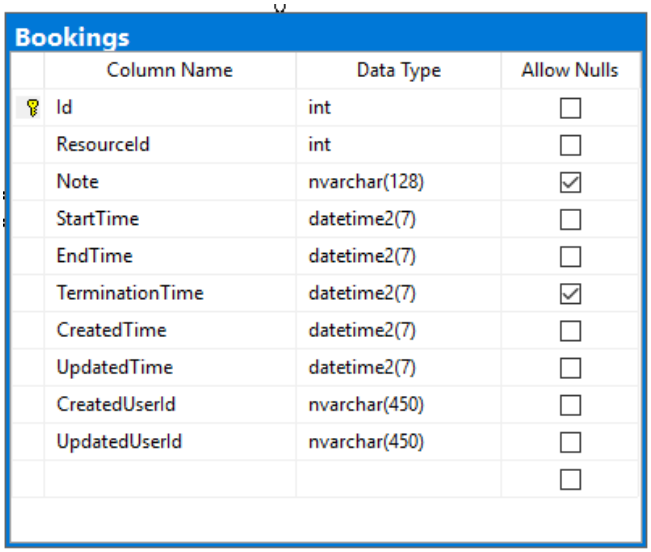


Рис. 2.6 – Схема сутності бронювання

Даний запис був введений з розрахунком на те, що люба з сторін має мати можливість скасувати бронювання. Також дана сутність містить нотатку яку залишив користувач. Це є зручним рішення для власника ресурсу.

Всі таблиці за винятком таблиць пов’язаних з користувачами мають поля, що потрібні для аудиту:

- CreateTime/UpdatTime – запису часової мітки створення/зміни запису;
- CreateUserId/UpdateUserId – запису ідентифікатора користувача який створив/змінив запис.

Також деякі таблиці мають поле IsActive, яке вказує чи запис активний, адміністратор, як правило має право бачити і взаємодіяти з неактивними записами, користувач, звісно, не може взаємодіяти з такими записами.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Загальний опис реалізації

Система реалізована у вигляді веб сервісу на базі ASP .Net Core 2.1 MVC, як фреймворку для бекенду, Angular 5, як фреймворку для фронтенд частини (веб аплікації), MS SQL, як системи управління базами даних. Даний стек технологій був обраний через простоту реалізації веб-сервісів та якісну документацію. Також даний стек є класичним рішенням для подібних систем.

Система складається з трьох проектів (Рис. 3.1):

- фронтенд і бекенд частина;
- модульні тести;
- інтеграційні тести.

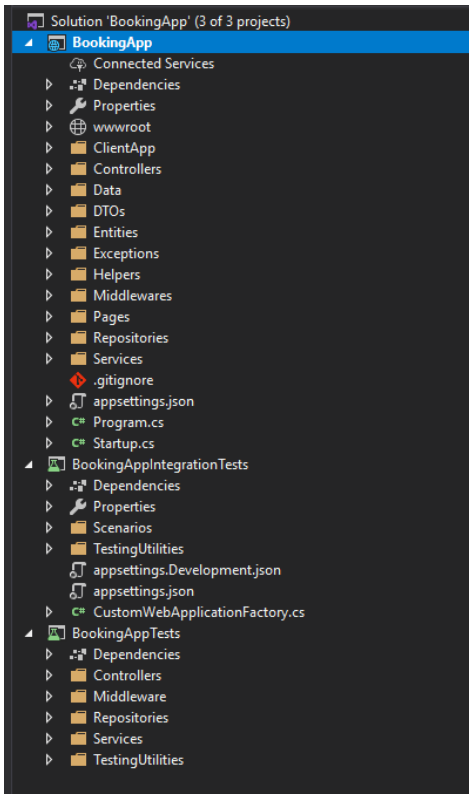


Рис. 3.1 – Структуру коду проекту

Фронтенд і бекенд частина розділені так як дані частини реалізовані з використанням різних стеків технологій. Фронтенд частина розташована в директорії ClientApp. А бекенд частина погрупована по наступним директоріям:

- Controlers – колекція контролерів (ендпоінтів), які надають точки взаємодії фронтенд частина та сторонніх систем з даним рішенням;
- Data – містить моделі бази даних і код SQL процедур [13] і функцій, контекст бази даних та логіку ініціалізації бази даних;
- DTOs – містить проміжні моделі[14], які є моделями даних, які приймає та вертає бекенд;
- Entities – містить моделі статистики, винесено окремо так, як вони не існують в межах бази даних і є самостійними сутностями;
- Exceptions – містить набір класів виключень, які є специфічними для бекенду;
- Helpers – містить допоміжні класи такі обробник помилок SQL процедур, завантажувач SQL процедур в базу даних, загально системні перелічення такі як ролі чи типи статистики, тощо;
- Midlewares – містить логіку проміжного обробника, який опрацьовує виключення при обробці запиту;
- Pages – містить стандартний шаблон сторінки помилки для розробників;
- Repositories – містить репозиторії, класи доступу до даних бази даних;
- Services – містить сервіси, які виконують логіку бекенду.

Окрім цього в класі Startup описано логіку запуску системи та ініціалізації компонентів які необхідні для системи.

Фронтенд частина в свою чергу теж має поділ який відображено на рисунку 3.2. Структура проекту має наступні ключові частини:

- директорія «e2e» містить тести інтерфейсу вебаплікації;
- директорія «src» в «app» містить код вебаплікації;

- директорія «admin» містить компоненти, які доступні лише адміністратору;
- директорія «cabinet» містить компоненти, які доступні зареєстрованому користувачу;
- директорія «material» містить опис компонентів системи Material[15], які використовуються у вебаплікації;
- директорія «models» містить моделі даних, якими оперує вебчастина;
- директорія «services» містить сервіси, які надають доступ до сервера та додаткову логіку таку, як загальна система нотифікацій та механізм авторизації;
- директорія «site» містить компоненти, які доступні всім, хто переглядає сторінку;
- директорія «tests» містить модульні тести вебаплікації;
- в файлах «app.component.*» описаний кореневий компонент аплікації, саме в ньому вказається, що буде в заголовку та знизу в усіх сторінка аплікації;
- в файлі «app.module.ts» реєструються всі компоненти та сервіси, які є в веб аплікації;
- в файлі «globals.ts» описані глобальні константи;
- директорія «asserts» містить статичні ресурси вебаплікації такі, як шрифти та зображення. Файли стилів знаходяться разом з компонентами тому їх немає в даній директорії;
- директорія «enviroment» містить перелік можливих середовищ запуска вебаплікації та логіки зв’язаної з цим;
- файл «index.html» містить базовий шаблон розмітки сторінки, в який вставляються частини веб аплікації.

Решта файлів є системним і створеними системою для коректної роботи програмного забезпечення.

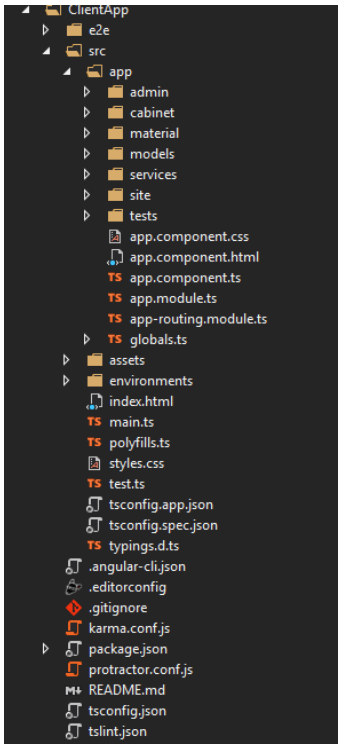


Рис. 3.2 – Структура фронтенд частини проекту

3.2 Вебаплікація

Веб частина аплікації складається з наступних основних компонентів:

- основна сторінка;
- кабінет користувача;
- кабінет адміністратора;
- сторінка бронювання.

3.2.1 Основна сторінка

Основна сторінка містить список всіх бронювань в дерево подібному вигляді. Вигляд основної сторінки для адміністратора можна побачити нижче, на рисунку 3.3.

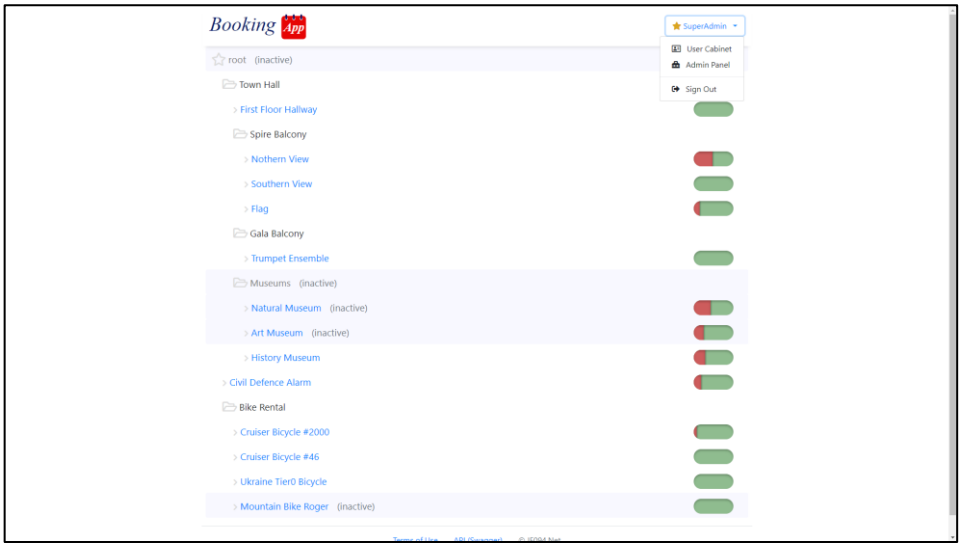


Рис. 3.3 – Вигляд основної сторінки для адміністратора

Також дана сторінка показує рівень зайнятості кожного ресурсу. Для розробників знизу є лінк на опис API взаємодії з бекенд частиною проекту. Дозволяє інтегрувати дану аплікацію в інші сервіси. Адміністратор додатково може бачити не активні на даний момент ресурси.

Зареєстрований або не зареєстрований користувач буде бачити тільки активні ресурси (Рис. 3.4). Кабінет користувача доступний як для адміністратора так і для зареєстрованого користувача.

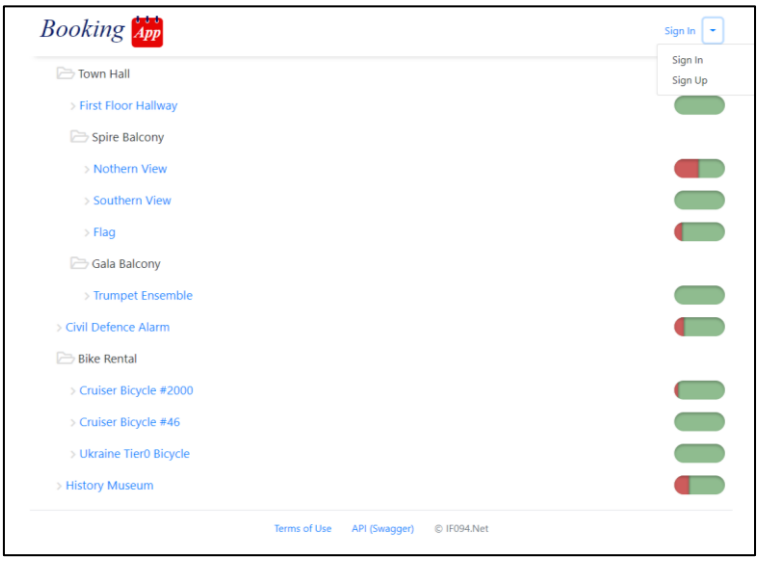


Рис. 3.4 – Вигляд основної сторінки для не зареєстрованого користувача

3.2.2 Кабінет користувача

Кабінет користувача дозволяє зміну логіну/паролю та перегляд власні бронювання (Рис 3.5).

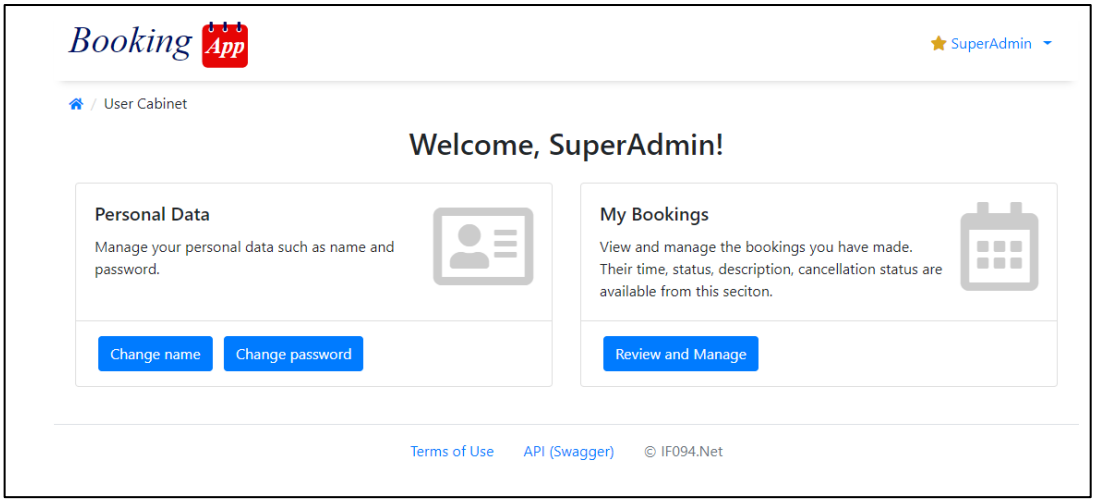


Рис. 3.5 – Кабінет користувача

Для зміни логіну достатньо ввести новий логін, а для зміни паролю потрібно ввести старий пароль і двічі новий (Рис. 3.6).

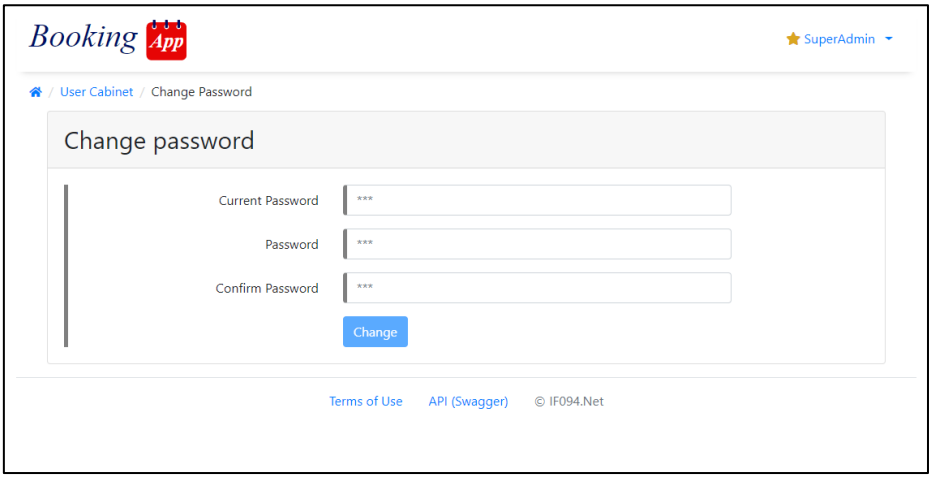


Рис. 3.6 – Сторінка зміни паролю

Перегляд власних бронювань (Рис. 3.7) дозволяє переглянути історію та стан бронювань, також є можливість відреагувати бронювання.

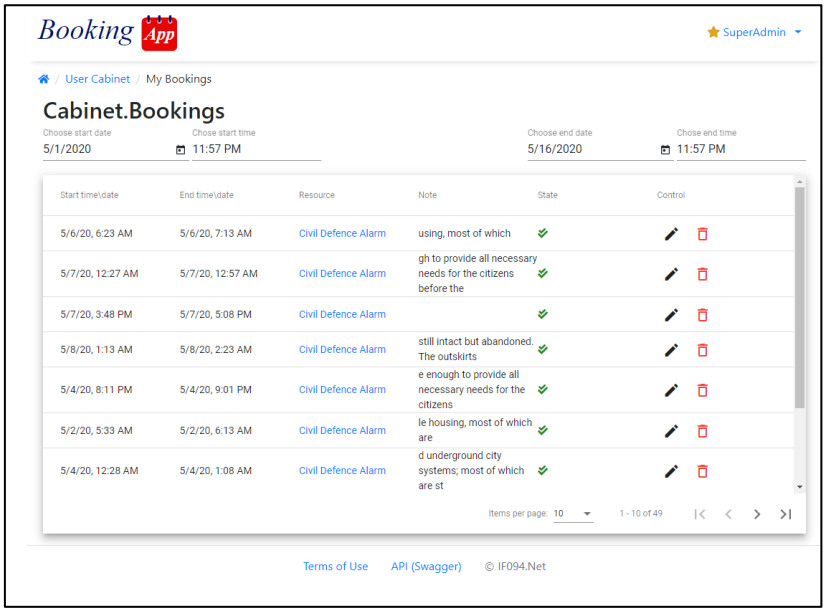


Рис. 3.7 – Список власних бронювань

3.2.3 Кабінет адміністратора

Для адміністраторів в аплікації передбачено окреме меню для керування бронювань, ресурсів, правил бронювань так користувачів. Також дане меню надає можливість переглянути статистику. Вигляд кабінету адміністратора зображено на рисунку 3.8.

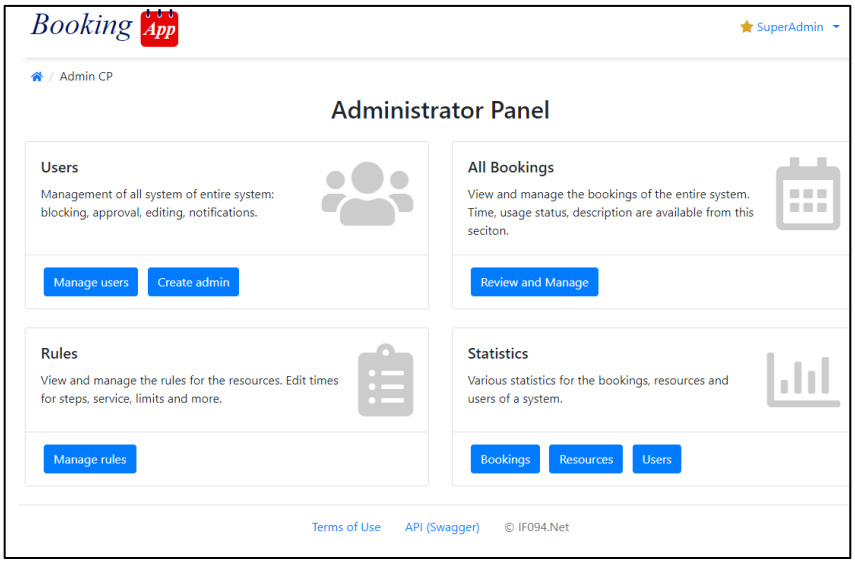


Рис. 3.8 – Вигляд кабінету адміністратора

Всі пункти в даному вікні поділені на чотири категорії:

- менеджмент користувача – включає, як менеджмент існуючими користувачами, так і можливість створення нового адміністратора;
- перегляд і керування всіма бронювань в системі;
- менеджмент правил бронювань;
- перегляд статистики по бронюванням, статистиці та користувачам.

Менеджмент користувачів дозволяє підтверджувати або відхиляти реєстрацію нових користувачів, блокувати або знімати блокування з існуючих користувачів (Рис. 3.9).

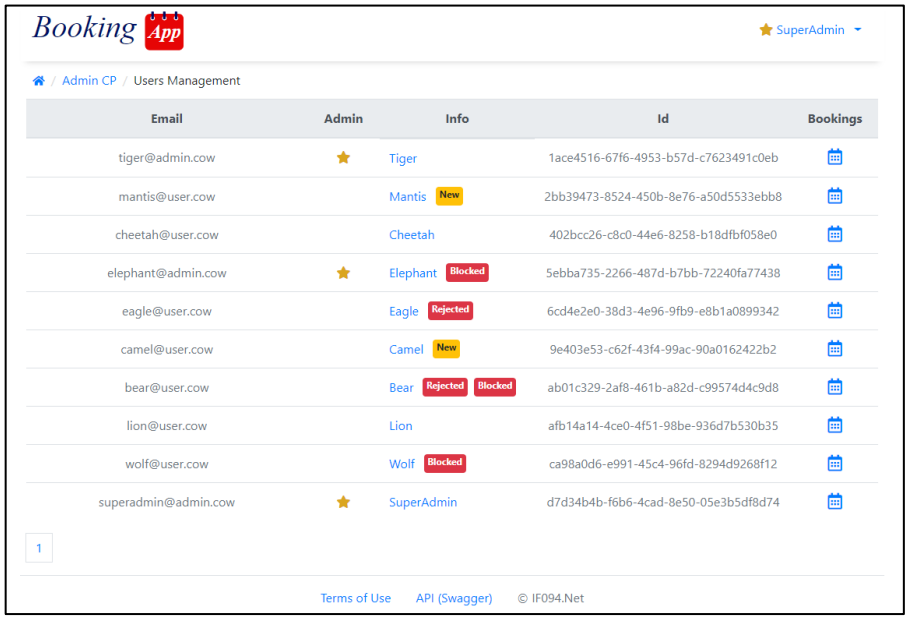


Рис. 3.9 – Сторінка керування існуючими користувачами

Можна також натиснути на логін користувача і відкрити додаткову сторінку керування відповідним користувачем. Сторінку меню керування користувачем зображено на рисунку 3.10.

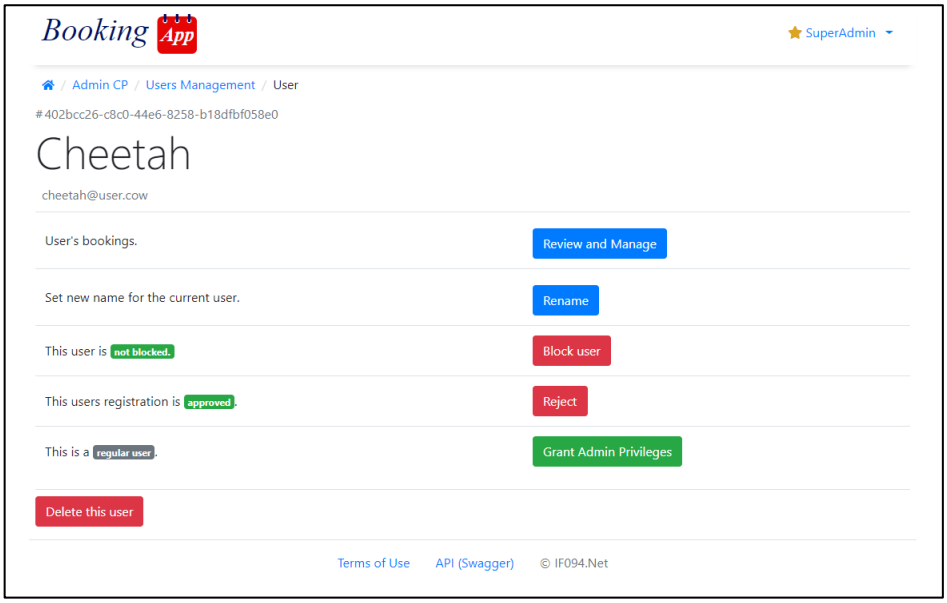


Рис. 3.10 – Вигляд меню керування користувачем

Дана сторінку надає наступні можливості:

- перегляд бронювання користувача;
- перейменування користувача;
- блокування та розблокування користувача;
- підтвердження або скасування реєстрації користувача;
- надання або видалення користувача з ролі адміністраторів;
- видалення користувача.

В кабінеті адміністратора також є пункт перегляду всіх бронювань який дозволяє переглянути всі існуючі бронювання в порядку їх створення (Рис. 3.11).

Дана сторінка дозволяє побачити інформацію про кожне бронювання та відредагувати або скасувати його. Відображається ідентифікатор бронювання, час початку/закінчення, ресурс, який бронюється, опис, який залишив користувач, користувач який забронював, поточний стан бронювання, та дії.

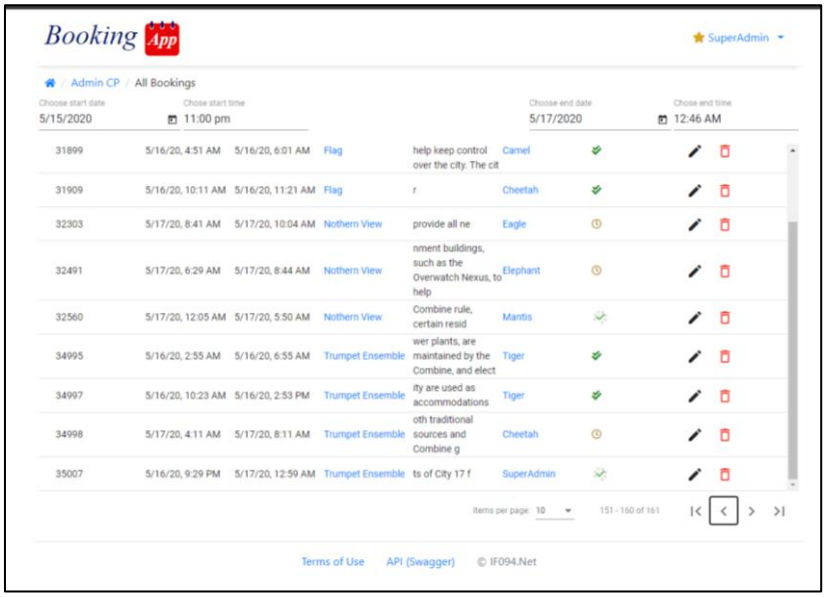


Рис. 3.11 – Сторінка історії всього бронювання

Редагування бронювання відбувається в окремому діалоговому вікні (Рис. 3.12).

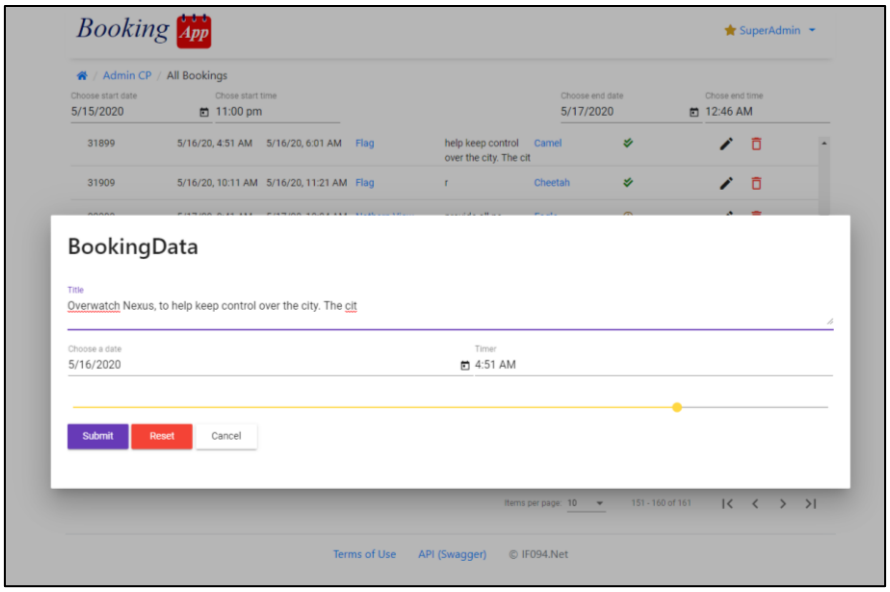


Рис. 3.12 – Вікно редагування бронювання

Дане вікно дозволяє зміни опис бронювання, час та дату бронювання та тривалість бронювання.

Також в кабінеті адміністратора є можливість налаштування правил бронювання (Рис. 3.13).

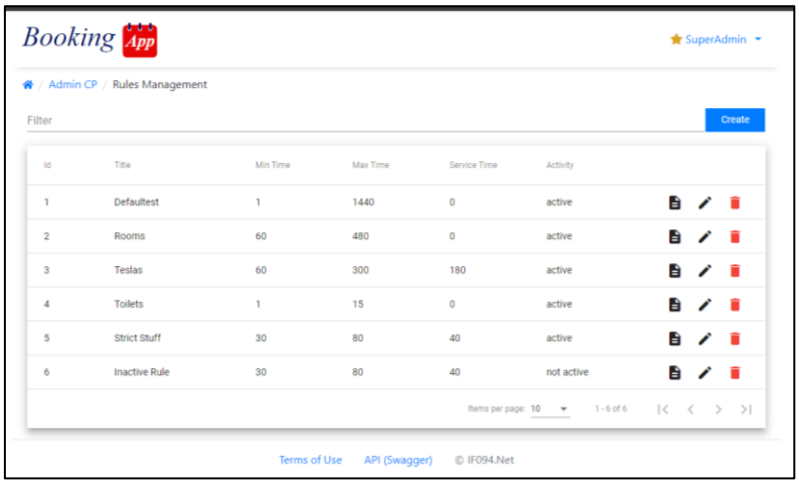


Рис. 3.13 – Налаштування правил бронювання

На даній сторінці можна переглянути список правил та основні параметри, також є кнопка додаткової інформації (Рис. 3.14).

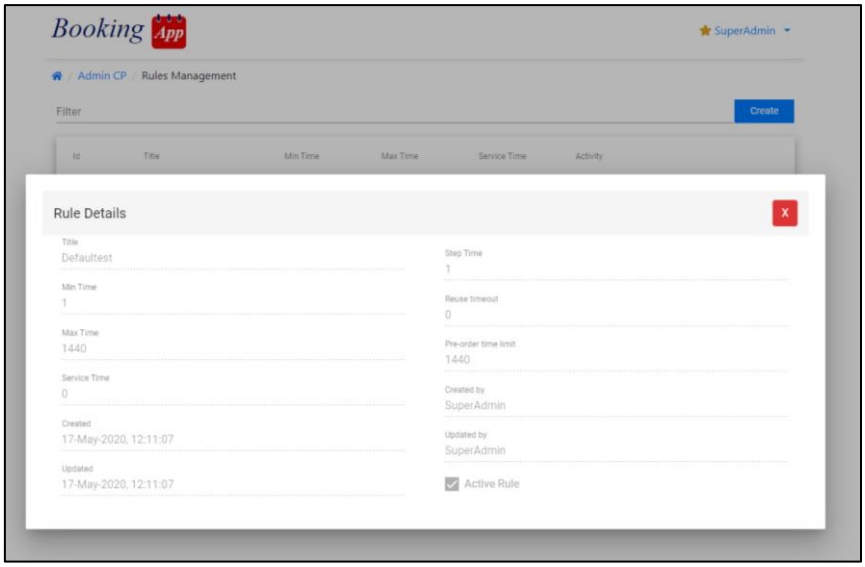


Рис. 3.14 – Додаткова інформація про бронювання

У вікні додаткової інформації можна переглянути всю інформацію про вибране правило. Також є кнопка для редагування правила (Рис. 3.15).

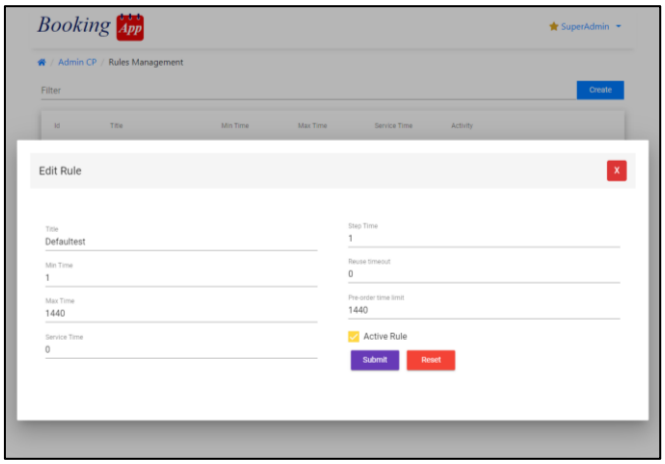


Рис. 3.15 – Редагування правила

У даному вікні є редагування параметрів правила. Всі параметри вказуються в хвилинах. Також є можливість деактивувати правило, що відповідно деактивує всі ресурси, які використовують дане правило.

Також в кабінеті адміністратора є можливість перегляду статистики за бронюванням, ресурсами та за користувачами. Всі три сторінки є приблизно однаковими за виглядом та функціоналом.

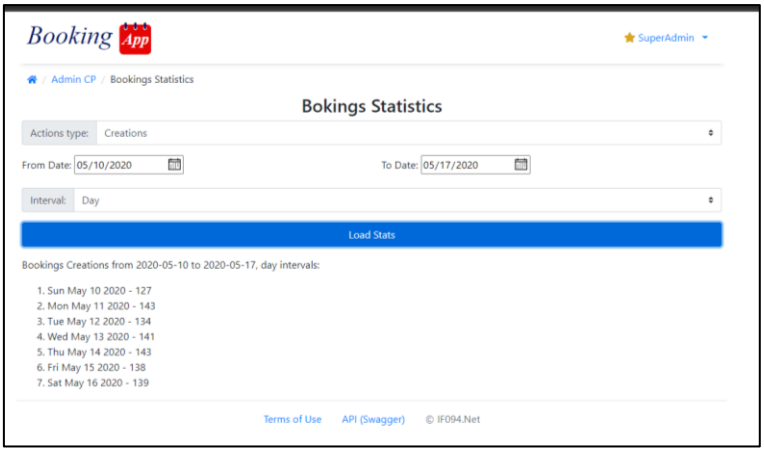


Рис. 3.16 – Сторінка статистики бронювання

3.2.4 Сторінка бронювання

Сторінка бронювання надає можливість всім зареєстрованим користувачам бронювати ресурси, не зареєстровані користувачі можуть тільки бачити стан ресурсу (Рис. 3.17).

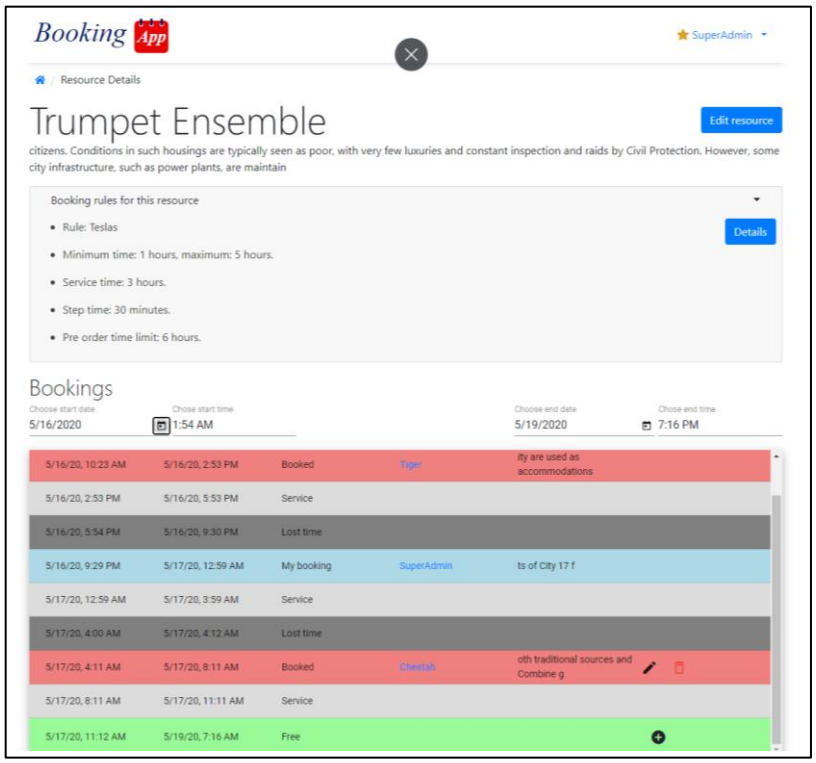


Рис. 3.17 – Сторінка бронювання

Дана сторінка дозволяє побачити стан бронювання певного ресурсу. Зверху опис ресурсу та правила бронювання, яке включає мінімальну та максимальну тривалість бронювання, крок тривалості бронювання, тривалість обслуговування, час, за який можна що найбільше забронювати ресурс.

Знизу таблиця, яка показує стан бронювання ресурсу на вибраний період часу:

- червоним показуються чужі бронювання. При чому адміністратор бачить чиї, а інші користувачі можуть бачити тільки факт зайнятості ресурсу у певний період часу;
- синім показані власні бронювання;
- сірим час обслуговування;
- чорним адміністратор бачить втрачений час, тобто час коли ресурс в минулому був вільний і не зайнятий користувачами;
- зеленим показано періоди які можна забронювати.

Адміністратор має право редагувати чужі бронювання, але лиш ті, які в майбутньому або активні. Також адміністратор, як і користувач, має право бронювати ресурс.

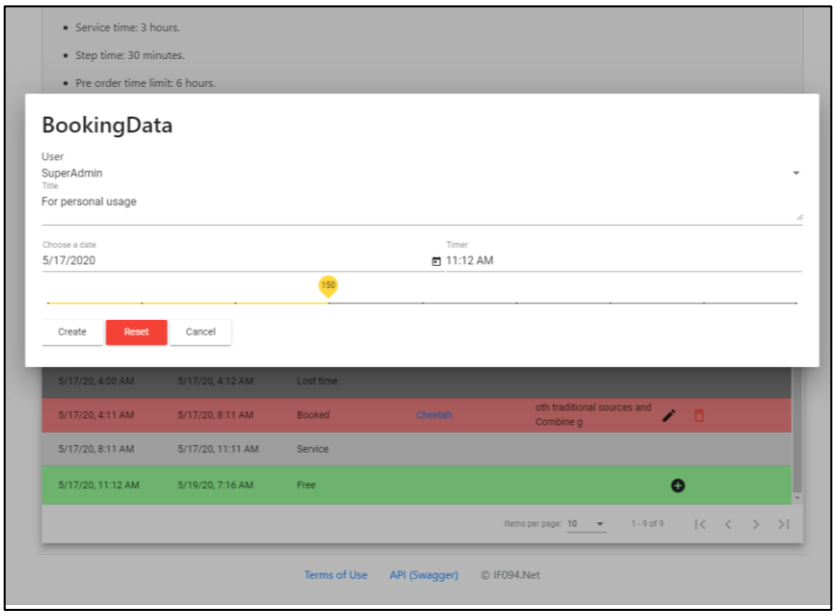


Рис. 3.18 – Вікно бронювання ресурсу

Адміністратор також має право бронювати ресурс не на себе, а на іншого користувача, що є корисним коли, наприклад, бронювання відбувається по телефону. Вибір тривалості бронювання відбувається за допомогою повзунка що є зручним рішенням та гнучким рішенням.

3.4 Реалізація API сервісу бронювання

Для API було вибрано C# Asp .Net Core 2.1, даний фреймворк є сучасним, високо продуктивним рішенням, яке має багатий набір готових функцій та бібліотек. В загальному API поділений на три шари:

- шар контролерів;
- шар сервісів;
- шар репозиторіїв.

Кожен шар має свою задачу і використовує або нижчий шар, або той же шар для виконання задач.

Задача контролерів полягає у перевірці авторизації так у виконанні перетворенні внутрішнього представлення даних в результуючий об’єкт та/або код результату. На рисунку 3.19 можна побачити приклад методу створення бронювання в контролері бронювання.

```
/// <summary>
/// Create new booking. POST: api/BookingsControler
/// </summary>
/// <param name="item">New <see cref="Booking"/> data</param>
/// <returns>Http response code</returns>
/// <response code="201">Success created</response>
/// <response code="400">Invalid argument</response>
/// <response code="404">Resources or rule not found</response>
/// <response code="500">Internal server error</response>
[HttpPost]
[ProducesResponseType(201)]
[ProducesResponseType(400)]
[ProducesResponseType(404)]
[ProducesResponseType(500)]
[Authorize(Roles = RoleTypes.User)]
0 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
public async Task<IActionResult> Create([FromBody] BookingCreateDTO item)
{
    if (!ModelState.IsValid)
        return BadRequest(ModelState);

    Booking model = dtoMapper.Map<Booking>(item);
    model.Note = item.Note;
    if (!IsAdmin)
        model.CreatedUserId = UserId;
    else
        model.CreatedUserId = item.CreatedUserId;

    await bookingService.CreateAsync(model);

    return new CreatedResult(
        $"api/booking/{model.Id}",
        dtoMapper.Map<BookingOwnerDTO>(await bookingService.GetAsync(model.Id))
        );
}
```

Рис. 3.19 – Ендпоінт створення бронювання

В загальному методи класів контролерів, які часто називаються ендпоінтами, мають наступні загальний алгоритм:

1. Перевірка цілісності вхідних даних, для чого використовується метод Model.IsValid [16]. Якщо даний метод вертає false, то метод відповідно вертає результат BadRequest [17], що сигналізує користувача про не коректні вхідні дані. Якщо перевірка пройшла успішно то переходим до наступного пункту.
2. Перетворення вхідних даних у внутрішнє представлення.
3. Виклик відповідних сервісів, які виконують основну логіку задачі. Контролер в ідеалі має викликати один метод відповідного сервісу, та часто внаслідок тих чи інших технічних причин може бути декілька викликів. Сервіс повертає результат виконання задачі.
4. Контролер, якщо сервіс повернув якісь дані, виконує перетворення внутрішнього представлення в модель результату.
5. Відправляє результат кінцевому користувачу.

Деякі методи контролера залежно від ролі користувача можуть вертати різні моделі (Рис. 3.20).

```
71
72          // Filtered access: Guest/User/Admin
73          /// <summary>
74          /// Get <see cref="Booking"/> info. GET: api/BookingControler/{bookingId}
75          /// </summary>
76          /// <param name="bookingId">Id of exist <see cref="Booking"/></param>
77          /// <returns>Status code</returns>
78          /// <response code="200">Success</response>
79          /// <response code="404">Not found</response>
80          /// <response code="500">Internal server error</response>
81          [HttpGet("{bookingId}")]
82          [ProducesResponseType(200)]
83          [ProducesResponseType(404)]
84          [ProducesResponseType(500)]
            3 references | 0 changes | 0 authors, 0 changes | 0 requests | 0 exceptions
85          public async Task<IActionResult> Details([FromRoute] int bookingId)
86          {
87              var model = await bookingService.GetAsync(bookingId);
88
89              if (model == null)
90                  throw new Exceptions.EntryNotFoundException($"Booking with id {bookingId} not found");
91
92              if (IsAdmin)
93              {
94                  var dtos = dtoMapper.Map<BookingAdminDTO>(model);
95                  return Ok(dtos);
96              }
97              else if(model.CreatedUserId == UserId)
98              {
99                  var dtos = dtoMapper.Map<BookingOwnerDTO>(model);
100                 return Ok(dtos);
101             }
102             else
103             {
104                 var dtos = dtoMapper.Map<BookingMinimalDTO>(model);
105                 return Ok(dtos);
106             }
107         }
```

Рис. 3.20 – Метод отримання інформації про бронювання

Метод отримання інформації про бронювання залежно від того чи користувач адміністратор повертає різні дані про бронювання.

Сервіси в свою чергу містять основну логіку, але часто вся задача зводиться до отримання даних з бази даних, тому методи сервісів можуть бути просто звертанням до репозиторію для отримання відповідних даних (3.21).

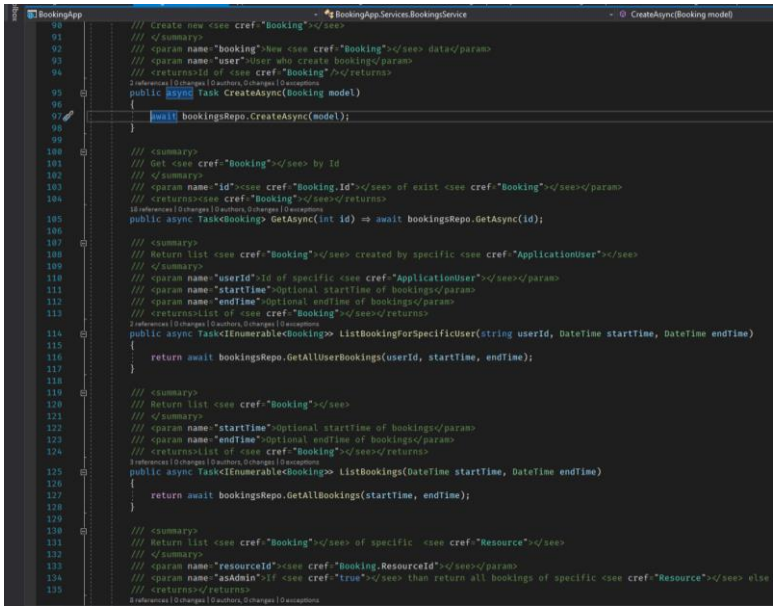


Рис. 3.21 – Приклад методів у сервісі бронювання

Останнім шаром в бекенд аплікації є репозиторій. Даний шар генерує запит до бази даних і повертає результат. В наслідок того, що EF Core 6 є дуже розвинутою, ORM[18] метод в репозиторіях теж часто короткі.

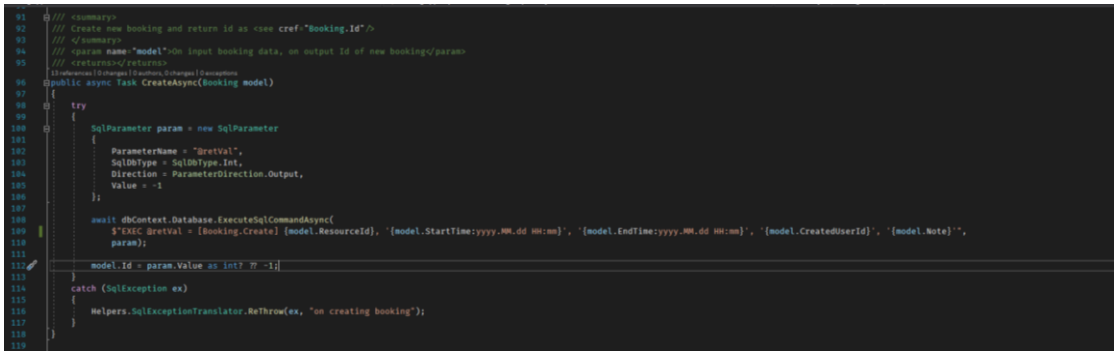
```
/// <summary>
/// Get all <see cref="Booking"></see> of specific <see cref="Resource"></see>
/// </summary>
/// <param name="resource">Exist <see cref="Resource.Id"></see></param>
/// <returns>List of all <see cref="Booking"></see> of specific <see cref="Resource"></see></returns>
public async Task<IEnumerable<Booking>> GetBookingsOfResource(int resourceId, DateTime startTime, DateTime endTime) => await ActualBookings
    .Where(b => ((b.ResourceId == resourceId) && ((b.TerminationTime ?? b.EndTime) <= endTime && b.StartTime >= startTime)))
    .ToListAsync();
```

Рис. 3.22 – Приклад методів в репозиторії

Деякі методи внаслідок своєї специфіки використовують не запити до бази даних, а виклик store procedure [13] це механізм SQL, який дозволяє виконати набір дій на стороні бази даних, зараз даний метод не рекомендується внаслідок

складності підтримки та оновлення таких процедур, але у випадках коли дані є дуже тісно між собою пов’язані, це може бути дуже ефективним механізмом.

На рисунку 3.23 зображено створення бронювання з використання процедури бази даних. Треба пам’ятати, що для створення бронювання потрібно перевірити чи ресурс, який бронюється в заданий час вільний і врахувати правила ресурсу при цьому. Тут прослідковується тісна зв’язаність даних. При яких процедура бази даних буде ефективніша.



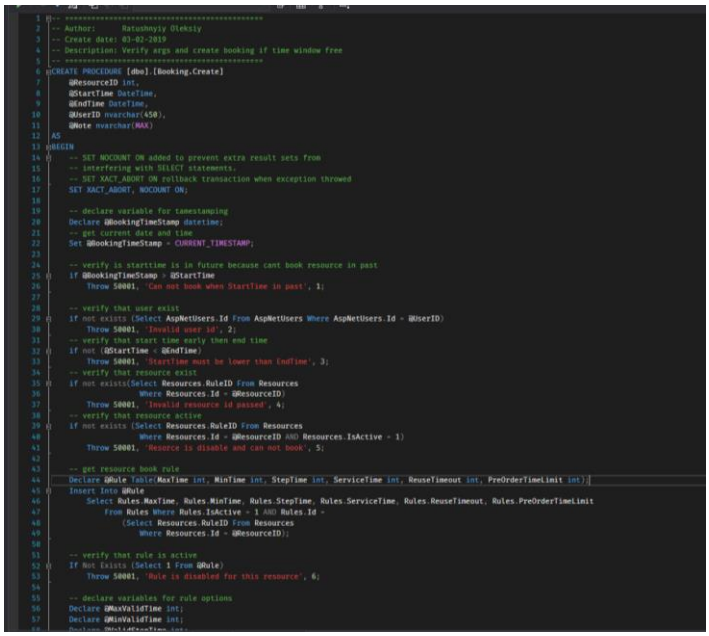
```
/// <summary>
/// Create new booking and return id as <see cref="Booking.Id"/>
/// </summary>
/// <param name="model">On input booking data, on output Id of new booking</param>
/// <returns></returns>
13 references | 0 changes | 0 authors, 0 changes | 0 exceptions
public async Task CreateAsync(Booking model)
{
    try
    {
        SqlParameter param = new SqlParameter
        {
            ParameterName = "@retVal",
            SqlDbType = SqlDbType.Int,
            Direction = ParameterDirection.Output,
            Value = -1
        };

        await dbContext.Database.ExecuteSqlCommandAsync(
            $"EXEC @retVal = [Booking.Create] {model.ResourceId}, '{model.StartTime:yyyy.MM.dd HH:mm}', '{model.EndTime:yyyy.MM.dd HH:mm}', '{model.CreatedUserId}', '{model.Note}'",
            param);

        model.Id = param.Value as int? ?? -1;
    }
    catch (SqlException ex)
    {
        Helpers.SqlExceptionTranslator.ReThrow(ex, "on creating booking");
    }
}
```

Рис. 3.23 – Виклику процедури бази даних в репозиторії

На рисунку 3.24 показано частину SQL процедури створення бронювання.



```
-- =============================================
-- Author:      RatushNyiy Oleksiy
-- Create date: 03-02-2019
-- Description: Verify args and create booking if time window free
-- =============================================
CREATE PROCEDURE [dbo].[Booking.Create]
    @ResourceID int,
    @StartTime DateTime,
    @EndTime DateTime,
    @UserID nvarchar(450),
    @Note nvarchar(MAX)
AS
BEGIN
    -- SET NOCOUNT ON added to prevent extra result sets from
    -- interfering with SELECT statements.
    -- SET XACT_ABORT ON rollback transaction when exception throwed
    SET XACT_ABORT, NOCOUNT ON;

    -- declare variable for timestamping
    Declare @BookingTimeStamp datetime;
    -- get current date and time
    Set @BookingTimeStamp = CURRENT_TIMESTAMP;

    -- verify is starttime is in future because cant book resource in past
    if @BookingTimeStamp > @StartTime
        Throw 50001, 'Can not book when StartTime in past', 1;

    -- verify that user exist
    if not exists (Select AspNetUsers.Id From AspNetUsers Where AspNetUsers.Id = @UserID)
        Throw 50001, 'Invalid user id', 2;
    -- verify that start time early then end time
    if not (@StartTime < @EndTime)
        Throw 50001, 'StartTime must be lower than EndTime', 3;
    -- verify that resource exist
    if not exists(Select Resources.RuleID From Resources
                    Where Resources.Id = @ResourceID)
        Throw 50001, 'Invalid resource id passed', 4;
    -- verify that resource active
    if not exists (Select Resources.RuleID From Resources
                    Where Resources.Id = @ResourceID AND Resources.IsActive = 1)
        Throw 50001, 'Resource is disable and can not book', 5;

    -- get resource book rule
    Declare @Rule Table(MaxTime int, MinTime int, StepTime int, ServiceTime int, ReuseTimeout int, PreOrderTimeLimit int)
    Insert Into @Rule
        Select Rules.MaxTime, Rules.MinTime, Rules.StepTime, Rules.ServiceTime, Rules.ReuseTimeout, Rules.PreOrderTimeLimit
            From Rules Where Rules.IsActive = 1 AND Rules.Id =
                (Select Resources.RuleID From Resources
                    Where Resources.Id = @ResourceID);

    -- verify that rule is active
    If Not Exists (Select 1 From @Rule)
        Throw 50001, 'Rule is disabled for this resource', 6;

    -- declare variables for rule options
    Declare @MaxValidTime int;
    Declare @MinValidTime int;
```

Рис. 3.24 – Перевірка аргументів в SQL процедурі створення бронювання в базі даних

Дана частина виконує перевірку вхідних аргументів та перевіряє наявність і активність ресурсу, який бронюється та правила для даного ресурсу.

Після перевірки аргументів відбувається перевірка діапазону часу відносно правила ресурсу, який намагаються забронювати, код даної перевірки можна побачити на рисунку 3.25.

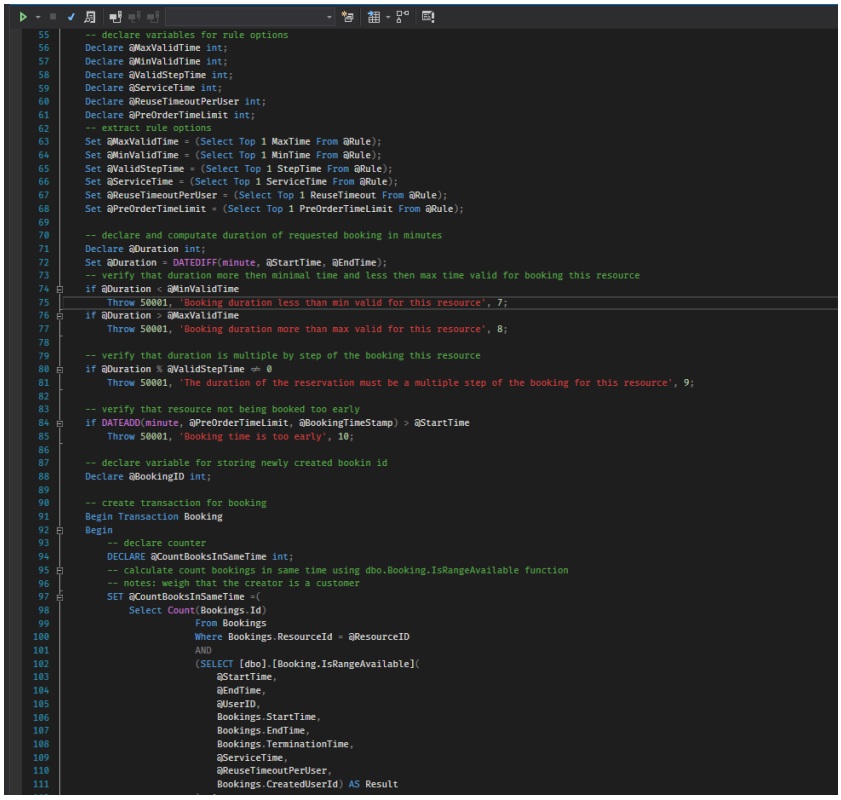


Рис. 3.25 – Логіка перевірки умов правила ресурсу в SQL процедури створення бронювання

Якщо якась з перевірок провалилась, SQL процедура генерує виключення, його перехоплює код на бекенду і відповідним чином реагує на виключення. Останньою перевіркою є перевірка на перетин часового діапазону з існуючими бронюваннями, код якої наведений на рисунку 3.26.

Для виконання перевірки відбувається виклик для кожного бронювання користувальницької SQL функції, яка реалізовує логікою перевірки, якщо перевірка виконалась успішно, то відбувається створення бронювання і

процедура завершує своє виконання. Цікавою деталлю цього етапу є те, що він виконується в межах транзакції. Це зроблено для гарантування атомарності створення бронювання.

```
    -- declare variable for storing newly created bookin id
    Declare @BookingID int;

    -- create transaction for booking
    Begin Transaction Booking
    Begin
        -- declare counter
        DECLARE @CountBooksInSameTime int;
        -- calculate count bookings in same time using dbo.Booking.IsRangeAvailable function
        -- notes: weigh that the creator is a customer
        SET @CountBooksInSameTime =(
            Select Count(Bookings.Id)
                        From Bookings
                        Where Bookings.ResourceId = @ResourceID
                        AND
                        (SELECT [dbo].[Booking.IsRangeAvailable](
                            @StartTime,
                            @EndTime,
                            @UserID,
                            Bookings.StartTime,
                            Bookings.EndTime,
                            Bookings.TerminationTime,
                            @ServiceTime,
                            @ReuseTimeoutPerUser,
                            Bookings.CreatedUserId) AS Result
                        )= 1
                );
        -- verify is no booking in same time
        if (@CountBooksInSameTime > 0)
            Throw 50001, 'Time range alredy booked', 11;
        -- declare temp table variable for store insert result
        Declare @InsertResult table(Id int);
        -- insert booking to bookings table and get his id
        Insert Into Bookings(ResourceID, StartTime, EndTime, CreatedTime, UpdatedTime, CreatedUserID, UpdatedUserID, Note)
        Output INSERTED.Id Into @InsertResult
        Values(@ResourceID, @StartTime, @EndTime, @BookingTimeStamp, @BookingTimeStamp, @UserID, @UserID, @Note);
        -- extract id from temp result table
        Set @BookingID = (Select Top 1 Id From @InsertResult);
        -- commit changes
    End;
    Commit Transaction Booking;
    -- return id of newly created booking
    Return Select Top 1 * From Bookings Where Bookings.Id = @BookingID
END
```

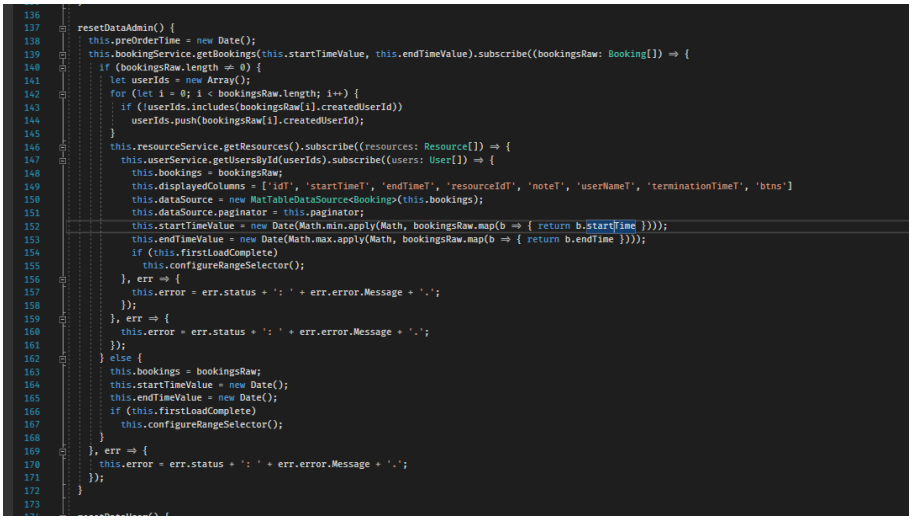
Рис. 3.26 – Перевірка на перетин часового діапазону з існуючими бронюваннями і додавання бронювання в SQL процедурі

На прикладі бронювання ресурсу яскраво видно шарову архітектуру даного програмного забезпечення. На фронтенд частині дані бронювання візуалізуються, але там теж є логіка зв’язана з відображенням даних.

Фронтенд частина ділиться на сервіси, які реалізовують логіку та з’єднання з ендпоінтами бекенду, і компоненти, які реалізовують логіку відображення і саме відображення даних.

Сервіси займаються підготовкою даних до відправки на бекенд і виконують відправку. Після чого повертається об’єкт Observable [19], який дозволяє додати дію, яка виконається після отримання відповіді від бекенду. На рисунку 3.27 яскраво бачимо з 28 по 36 рядки включно як відбувається

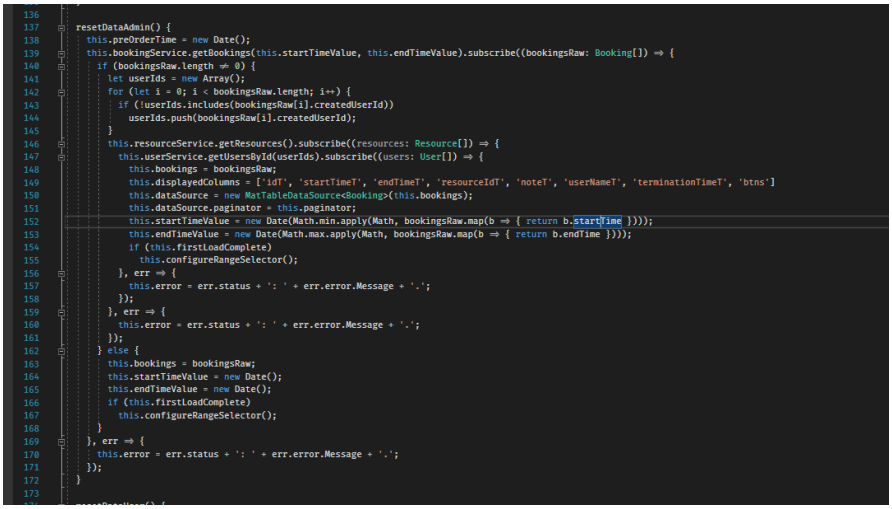
підготовка даних до відправки, а в 37 відбувається сама відправка і повернення об’єкту.



```
resetDataAdmin() {
  this.preOrderTime = new Date();
  this.bookingService.getBookings(this.startTimeValue, this.endTimeValue).subscribe((bookingsRaw: Booking[]) => {
    if (bookingsRaw.length != 0) {
      let userIds = new Array();
      for (let i = 0; i < bookingsRaw.length; i++) {
        if (!userIds.includes(bookingsRaw[i].createdUserId))
          userIds.push(bookingsRaw[i].createdUserId);
      }
      this.resourceService.getResources().subscribe((resources: Resource[]) => {
        this.userService.getUsersById(userIds).subscribe((users: User[]) => {
          this.bookings = bookingsRaw;
          this.displayedColumns = ['idT', 'startTimeT', 'endTimeT', 'resourceIdT', 'noteT', 'userNameT', 'terminationTimeT', 'btns']
          this.dataSource = new MatTableDataSource<Booking>(this.bookings);
          this.dataSource.paginator = this.paginator;
          this.startTimeValue = new Date(Math.min.apply(Math, bookingsRaw.map(b => { return b.startTime })));
          this.endTimeValue = new Date(Math.max.apply(Math, bookingsRaw.map(b => { return b.endTime })));
          if (this.firstLoadComplete)
            this.configureRangeSelector();
        }, err => {
          this.error = err.status + ': ' + err.error.Message + '.';
        });
      }, err => {
        this.error = err.status + ': ' + err.error.Message + '.';
      });
    } else {
      this.bookings = bookingsRaw;
      this.startTimeValue = new Date();
      this.endTimeValue = new Date();
      if (this.firstLoadComplete)
        this.configureRangeSelector();
    }
  }, err => {
    this.error = err.status + ': ' + err.error.Message + '.';
  });
}
```

Рис. 2.27 – Частина сервісу бронювання на фронтенді

Даний об’єкт використовується в інших сервісах або в компонентах для реєстрації дії, яку потрібно виконати з даними після їх отримання. На рисунку 2.28 можна побачити в 139 рядку виклик методу сервісу та реєстрації обробника у вигляді анонімної функції, яка в свою чергу виконує завантаження додаткових даних і підготовку списку бронювання до відображення яке описано у вигляді шаблону html розмітки.



```
resetDataAdmin() {
  this.preOrderTime = new Date();
  this.bookingService.getBookings(this.startTimeValue, this.endTimeValue).subscribe((bookingsRaw: Booking[]) => {
    if (bookingsRaw.length != 0) {
      let userIds = new Array();
      for (let i = 0; i < bookingsRaw.length; i++) {
        if (!userIds.includes(bookingsRaw[i].createdUserId))
          userIds.push(bookingsRaw[i].createdUserId);
      }
      this.resourceService.getResources().subscribe((resources: Resource[]) => {
        this.userService.getUsersById(userIds).subscribe((users: User[]) => {
          this.bookings = bookingsRaw;
          this.displayedColumns = ['idT', 'startTimeT', 'endTimeT', 'resourceIdT', 'noteT', 'userNameT', 'terminationTimeT', 'btns']
          this.dataSource = new MatTableDataSource<Booking>(this.bookings);
          this.dataSource.paginator = this.paginator;
          this.startTimeValue = new Date(Math.min.apply(Math, bookingsRaw.map(b => { return b.startTime })));
          this.endTimeValue = new Date(Math.max.apply(Math, bookingsRaw.map(b => { return b.endTime })));
          if (this.firstLoadComplete)
            this.configureRangeSelector();
        }, err => {
          this.error = err.status + ': ' + err.error.Message + '.';
        });
      }, err => {
        this.error = err.status + ': ' + err.error.Message + '.';
      });
    } else {
      this.bookings = bookingsRaw;
      this.startTimeValue = new Date();
      this.endTimeValue = new Date();
      if (this.firstLoadComplete)
        this.configureRangeSelector();
    }
  }, err => {
    this.error = err.status + ': ' + err.error.Message + '.';
  });
}
```

Рис. 3.28 – Частина компоненту бронювання яка виконує логіку для перегляду бронювання адміністратором

На рисунку 3.29 можна побачити частину шаблону, який показує таблицю бронювання для адміністратора.

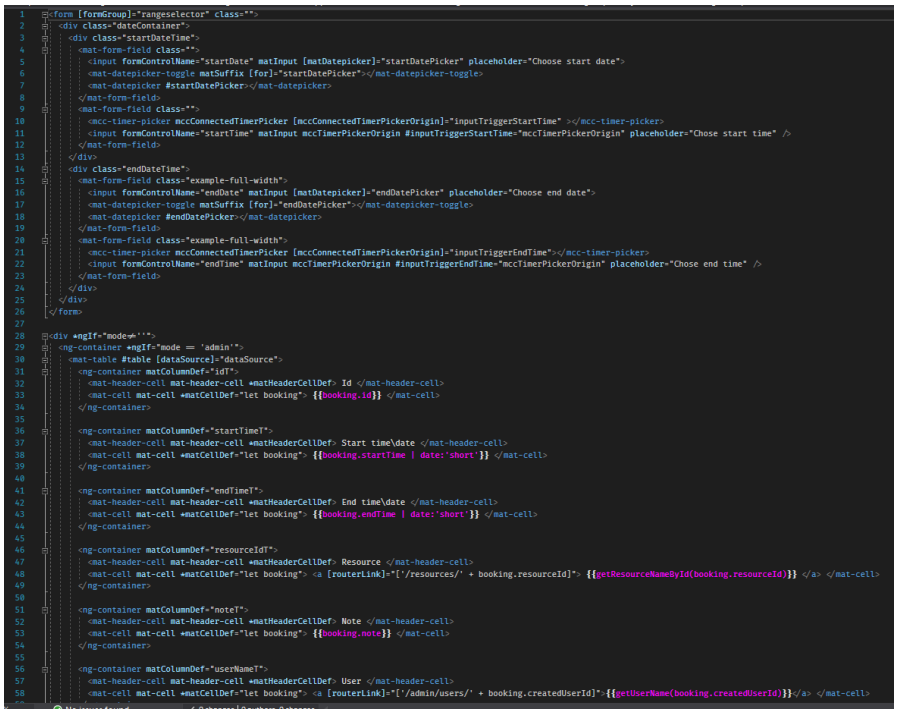


Рис. 3.29 – Частина html шаблону для відображення бронювань

Дані шаблони окрім самого html містять директиви, які дозволяють правильно зв’язати дані з розміткою. Що в свою чергу дозволяє гнучко писати комплексні сторінки з складною обробкою даних. Як вище було сказано, окрім самого html, на рисунку можна побачити директиви, які дозволяють дану сторінку гнучко генерувати, не залежно від кількості бронювання.

В додатку А знаходиться посилання на репозиторій розміщений у GitHub з кодом проекту.

4 БІЗНЕС-ПЛАН

4.1 Резюме проекту

Програмне забезпечення надаватиме послуги корпоративним клієнтам та фізичним особам з організації контролю доступу до різноманітних ресурсів.

Користувачами ресурсів буду корпоративні клієнти та фізичні особи, в яких є потреба в організації доступу до певних ресурсів, будь то приміщення в офісних будівлях чи комп'ютерні системи спільного доступу.

Орієнтовними початковими витратами є 30 тисяч гривень.

На розробку програмного продукту залучено особисті кошти.

Планова виручка становить 120 тисяч гривень за перший рік роботи, та у середньому 11 тисяч гривень за місяць.

Рентабельність проекту становить 1.2.

4.2 Маркетинг

Потенційним клієнтам буде запропоновано сервіс, що спростить рутині задачі по впорядкуванню доступу до спільних ресурсів.

Програмне забезпечення буде задовільнять такі потреби клієнтів як:

- ефективне використання ресурсів які будуть бронюватись;
- економія часу;
- зручний менеджмент об'єктів, які бронюються;
- статистичні дані про бронювання;
- не потрібність встановлення програмного забезпечення;
- інтеграція з системами аутентифікації та авторизації клієнтів;
- мобільність;
- готовий вебсервіс для адміністрування.

Програма надаватиме функціонал для різних сфер діяльності людини і підійде клієнтам, які мають велику кількість офісів і хочуть оптимізувати їхнє використання без скорочення штату працівників.

Адміністраторам системи буде надано готовий вебсервіс для адміністрування, що з спростить інтеграцію системи в існуючу екосистему програмного забезпечення компанії.

Дане програмне рішення буде постійно підтримуватись і клієнти буду отримувати оновлення та підтримку в режимі онлайн.

Потенційним недоліками можуть бути:

- не достатня система розділення користувачів на групи та ролі;
- відсутність експорту статистики в документи з вебінтерфейсу адміністрування.

В майбутньому планується ввести підтримку авторизації та аутентифікації використовуючи протокол LDAP[20], Kerberos [21] та технології SSO(SAML2).

Ринок збуту даного програмного забезпечення немає географічних обмежень. Клієнт може працювати абсолютному будь-якому місці світу так, як програмний продукт легко встановлюється та інтегрується з існуючим програмним забезпеченням.

Відмінною рисою даних систем є висока гнучкість і ефективна інтеграція з різноманітним сторонніми сервісами.

На початкове просування сервісу та рекламу програмного комплексу буде інвестовано приблизно 20% загального бюджету.

Рекламуючи програмний комплекс необхідно буде вказати загальну інформацію з приверненням уваги на його особливість та переваги відносно самописних рішень.

Розміщення реклами можливе в соціальних мережах та різноманітних сайтах і буде відбуватись не лише на початку запуску продажів продукту, а й на протязі всього циклу підтримки. На дану статтю видатків буде відведено 5-10% від щомісячного прибутку.

Прибуток буде приходити від продажів клієнтам та підтримці продукту. Оскільки прибуток буде залежати прямо від кількості продажів, прогнози яких можна побачити на рисунку 4.1.

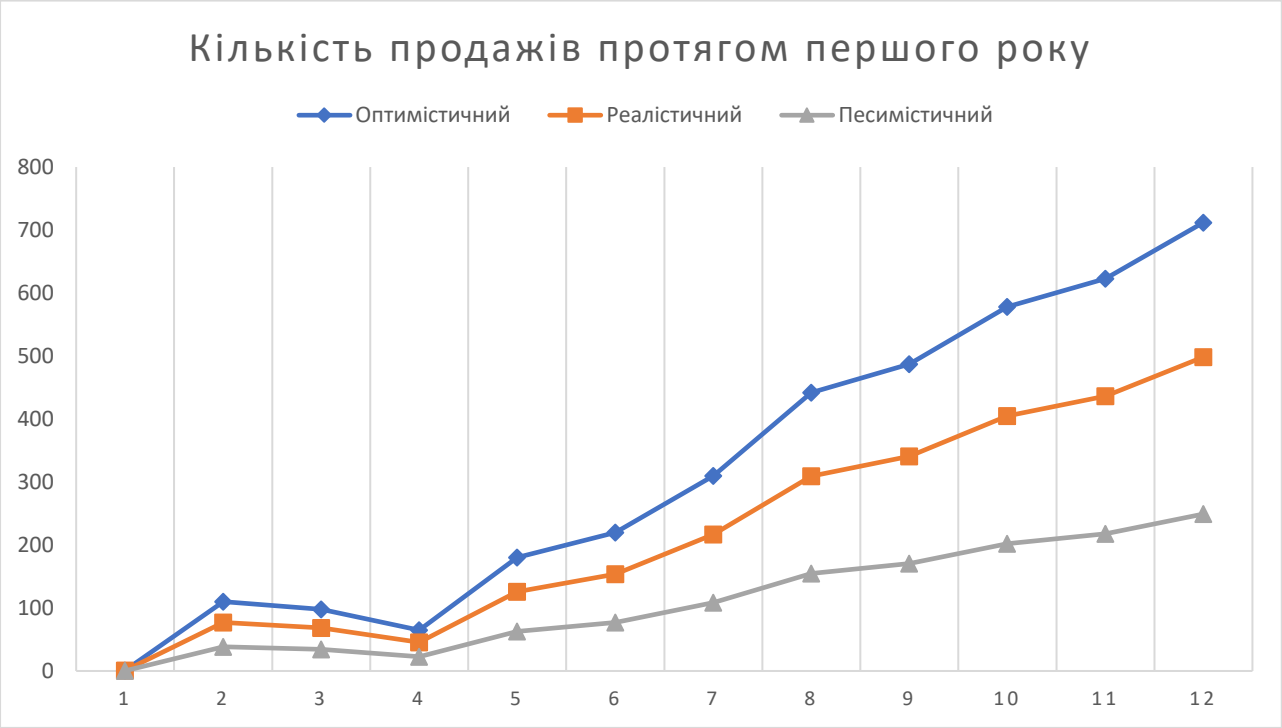


Рис. 4.1 – Прогнози кількості продажів

Згідно вище наведених прогнозів протягом року кількість копій буде наступна:

- при оптимістичному варіанті – 3826 проданих копій;
- при реалістичному варіанті – 2678 проданих копій;
- при песимістичному варіанті – 1339 проданих копій.

4.3 Нормативно правові моменти

Юридичний статус бізнесу - товариство з обмеженою відповідальністю.

Перевагами є у спрощеному способі реєстрації, у мінімальному фінансовому ризику для учасників, оскільки в ТОВ [22] більш захищені майнові

права, у можливості розширення бізнесу і залучення в нього інвестицій, у можливості використання спрощеної системи оподаткування.

У майбутньому можлива зміна форми бізнесу.

На початковому етапі дозволи на такого роду діяльність не потрібна.

4.4 Бюджет проекту

До виробничих затрат належить одне приміщення з доступом до електроенергії і комп'ютер для розробки програмного забезпечення.

За попередньою оцінкою відносно складності системи, на розробку програмного забезпечення потрібно 151 година, якщо розробка буде виконуватись однією людиною.

На оплату години роботи програміста початківця в середньому необхідно 67,5 грн/год, отже, в межах реалізації даного програмного забезпечення на заплату потрібно виділити 13 162,5 гривень.

Орієнтовна сума даних витрат, до отримання перших прибутків, становить 1914 грн/міс.

Амортизаційні відповідають 18% від витрат на розробку програмного забезпечення. На випадок надзвичайної ситуації регулярно відкладається 7% прибутку, як будуть використанні при такій ситуації.

Тривалість організаційного періоду бізнесу рівна двом місяцям. Фінансування проекту відбувається в до початку роботи в повному обсязі.

На початковому етапі впровадження програмного забезпечення розмір податків становитиме до 10% від мінімальної заробітної плати, що рівно 1070 гривні на місяць.

4.5 План очікуваних прибутків

Очікуваний прибуток від реалізації проекту за перший рік становить в середньому 12 000 грн/місяць, залежності від справдження прогнозів.

Сума постійних витрат становить приблизно 2 400 грн. На початковому етапі запуску програми витрати на просування та рекламу також постійно присутні і становлять 10% від загального прибутку.

В підсумку протягом першого року від початку реалізації програмного продукту постійні витрати становлять 45% від загального прибутку.

Постійний прибуток становить 55% від загального прибутку та дорівнює 6050 гривень на місяць в середньому за перший рік реалізації програмного продукту та поступово зростатиме. Розрахунок показників проекту. Загальна сума чистого прибутку за перший на перший рік становить 66 000 гривень.

Рентабельність продукту: 66 000 / 30 000 – 1 = 1,2. Таким чином впровадження даного програмного продукту є економічно вигідним.

ВИСНОВКИ

Метою даного проекту було створення вебсервісу для бронюванням. Для реалізації поставленої мети було зроблено аналіз предметної області дипломного проекту. Після чого було визначено набір вимог до програмного забезпечення та розроблено моделі можливостей системи і база даних.

Для реалізації було обрано мови програмування C# та TypeScript з використанням фреймворку Asp .Net Core та Angular. Базу даних було розроблена з використанням ORM Entiry Framework Core.

Було розроблено сайт для виконання бронювання та менеджменту ресурсами, правилами ресурсів та користувачами.

Функціонал даного програмного забезпечення буде в майбутньому розширюватись та вдосконалюватиметься, а саме запланованими є наступні завдання:

- підтримка сторонніх провайдерів авторизації;
- html розмітка для опису ресурсу бронювання;
- правила на основі днів тижня та певних дат;
- гнучка статистика по групі ресурсів;
- вивід графіків статистики для більшої наочності даних;
- обмеження видимості бронювання користувачами на основі приналежності останніх до певних ролей та груп з використанням механізму ACL[23].

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

REFERENCES

1. Pro ASP.NET Core MVC 2.

   Authors: Freeman, Adam. ISBN 978-1-4842-3150-0

   (дата звернення: 10.01.2020).

2. ASP.NET Core 2 and Angular 5: Full-Stack Web Development with .NET Core and Angular Paperback – November 24, 2017. Authors: Valerio De Sanctis. ISBN: 978-1788293600

   (дата звернення: 10.01.2020).

3. T-SQL Fundamentals 3rd Edition. August 13, 2016. Author Itzik Ben-Gan. ISBN: 978-1509302000

   (дата звернення: 11.01.2020).

4. Онлайн-бронювання. URL:

   https://uk.wikipedia.org/wiki/Онлайн-бронювання

   (дата звернення: 11.01.2020).

5. Booking.com. URL:

   https://www.booking.com/

   (дата звернення: 20.01.2020).

6. Онлайн резервування та придбання квитків – Укрзалізниця. URL:

   https://booking.uz.gov.ua/

   (дата звернення: 25.01.2020).

7. Сценарій використання. URL:

https://uk.wikipedia.org/wiki/Сценарій_використання

(дата звернення: 02.02.2020).

8. JSON Web Token. URL:

https://uk.wikipedia.org/wiki/JSON_Web_Token

(дата звернення: 13.02.2020).

9. Refresh Tokens: When to Use Them and How They Interact with JWTs. URL:

https://auth0.com/blog/refresh-tokens-what-are-they-and-when-to-use-them/

(дата звернення: 15.02.2020).

10. Security Assertion Markup Language a Complete Guide - 2020 Edition. Author: Gerardus Blokdyk. ISBN: 9781867324508

(дата звернення: 20.02.2020).

11. Single sign-on. URL:

https://en.wikipedia.org/wiki/Single_sign-on

(дата звернення 20.02.2020)

12. Introduction to Identity on ASP.NET Core. URL:

https://docs.microsoft.com/en-us/aspnet/core/security/authentication/identity?view=aspnetcore-3.1&tabs=visual-studio

(дата звернення: 01.03.2020).

13. SQL Stored Procedures. URL:

https://www.w3schools.com/sql/sql_stored_procedures.asp

(дата звернення: 04.03.2020).

14. Create Data Transfer Objects (DTOs). URL:

https://docs.microsoft.com/en-us/aspnet/web-api/overview/data/using-web-api-with-entity-framework/part-5

(дата звернення: 08.03.2020).

15. Angular Material UI component library. URL:

https://material.angular.io/

(дата звернення: 12.03.2020).

16. Model validation in ASP.NET Core MVC and Razor Pages. URL:

https://docs.microsoft.com/en-us/aspnet/core/mvc/models/validation?view=aspnetcore-3.1

(дата звернення: 01.04.2020).

17. ControllerBase.BadRequest Method. URL:

https://docs.microsoft.com/en-us/dotnet/api/microsoft.aspnetcore.mvc.controllerbase.badrequest?view=aspnetcore-3.1

(дата звернення: 01.04.2020).

18. Об'єктно-реляційне відображення. URL:

https:// uk.wikipedia.org/wiki/Об’єктно-реляційне_відображення

(дата звернення: 10.04.2020).

19. Observables in Angular. URL:

https://angular.io/guide/observables-in-angular

(дата звернення: 15.04.2020).

20. The ABCs of LDAP: How to Install, Run, and Administer LDAP Services 1st Edition. Author o Reinhard E. Voglmaier. ISBN: 978-1138453630

(дата звернення: 02.05.2020).

21. Kerberos (протокол). URL:
https://uk.wikipedia.org/wiki/Kerberos_(протокол)
(дата звернення: 02.05.2020).

22. Товариство з обмеженою відповідальністю. URL:
uk.wikipedia.org/wiki/Товариство_з_обмеженою_відповідальністю
(дата звернення: 12.05.2020).

23. Access control list. URL:
https://uk.wikipedia.org/wiki/Access_control_list
(дата звернення: 03.05.2020).

ДОДАТОК А

GitHub: https://github.com/Natsuki0Subaru/BookingApp