ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра інженерії програмного забезпечення

Пояснювальна записка

до бакалаврської кваліфікаційної роботина ступінь вищої освіти бакалавр

на тему: **«Розробка серверного додатку для організації роботи автомобільного аукціону з використанням технології ASP.NET Web API, мовою C#»**

Виконав: студент 4 курсу, групи ПД– 41

Спеціальності 121 Інженерія програмного забезпечення

(шифр і назва спеціальності)

Решетнік Н. О.

(прізвище та ініціали)

Керівник Трінтіна Н.А.

(прізвище та ініціали)

Рецензент _____

(прізвище та ініціали)

Нормоконтроль _____

(прізвище та ініціали)

Київ – 2023

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ
НАВЧАЛЬНО–НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

Кафедра - Інженерії програмного забезпечення
Ступінь вищої освіти -
"Бакалавр"
Спеціальність - 121 «Інженерія програмного забезпечення»

ЗАТВЕРДЖУЮ
Завідувач кафедри
інженерії програмного забезпечення

_____ Негоденко О.В.

"___" _____ 2023 року

ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ

Решетнік Нікіта Олександрович
(прізвище, ім'я, по батькові)

1. Тема роботи: «Розробка серверного додатку для організації роботи автомобільного аукціону з використанням технології ASP.NET Web API, мовою C#»

Керівник роботи к.т.н., доцент Трінтіна Н. А.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від "24" лютого 2023 року №22.

2. Строк подання студентом роботи "1" червня 2023 року
3. Вихідні дані по роботі:
   3.1. Офіційна документація Microsoft.
   3.2. Наукова-технічна література.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити):
   4.1. Аналіз та огляд існуючих додатків та інструментів для реалізації системи.
   4.2. Розробка структури серверного додатку для організації роботи автомобільного аукціону.
   4.3. Програмна реалізація додатку.
   4.4. Висновки

5. Перелік графічного матеріалу
   5.1. Титульний слайд
   5.2. Мета, об’єкт, предмет та наукова новизна дослідження
   5.3. Аналоги
   5.4. Технічні завдання
   5.5. Програмні засоби реалізації
   5.6. Схема бази даних додатку

Дата видачі завдання 25.02.2022

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів бакалаврської роботи	Строк виконання етапів роботи	Примітка
1	Ознайомлення з предметною галуззю	25.02.2023 - 01.03.2023	Виконано
2	Визначення етапів розробки додатку	02.03.2023 – 10.03.2023	Виконано
3	Розробка примітивного каркасу та архітектури додатку	11.03.2023 – 15.03.2023	Виконано
4	Проектування структури бази даних	16.04.2023 – 19.04.2023	Виконано
5	Розробка основної «бізнес-логіки» додатку	20.04.2023 – 02.05.2023	Виконано
6	Рефакторінг додатку	02.05.2023 – 04.05.2023	Виконано
7	Тестування додатку	05.05.2023 - 07.05.2023	Виконано
8	Оформлення пояснювальної записки	08.05.2023 – 18.05.2023	Виконано
9	Передзахист	19.05.2023 - 25.05.2023	Виконано
10	Захист роботи	01.06.2023	

Студент_____ Н.О. Решетнік
(підпис) *(прізвище та ініціали)*

Керівник роботи_____ Н.А. Трінтіна

РЕФЕРАТ

Текстова частина бакалаврської роботи с., рис., табл, джерел.

Мета роботи – спрощення процесу цифровізації сфери автомобільних аукціонів переходу до цифрової форми проведення торгів, що дозволить їм ефективніше взаємодіяти зі своїми клієнтами та знизити витрати на проведення аукціонів.

Об'єкт дослідження – розробка серверного додатку для автоматизації проведення автомобільних аукціонів.

Предмет дослідження – додаток для автоматизації проведення автомобільних аукціонів. Дослідження включає детальний аналіз вимог до функціональності цифрової платформи для автомобільних аукціонів, проектування архітектури та розробку функціоналу, включаючи взаємодію з базою даних. Окрім цього, в рамках дослідження можуть бути досліджені питання безпеки даних та захисту персональної інформації користувачів та бізнесу.

У роботі було проведено аналіз існуючих сервісів та предметної галузі, які використовуються для цифровізації роботи автомобільних аукціонів. Також було виконано проектування додатку з використанням діаграм UML, включаючи діаграму варіантів використання, діаграму архітектури додатку, та діаграму баз даниї. Додаток було розроблено з використанням мови C# та фреймворка ASP.NET з використанням СУБД PostgreSQL.

Сфера використання – будь-який бізнес з продажу автомобілів, які вже продають автомобілі за допомогою аукціонів, чи розглянули б таку можливість.

ЗМІСТ

ЗАВДАННЯ НА БАКАЛАВРСЬКУ РОБОТУ СТУДЕНТУ 2

КАЛЕНДАРНИЙ ПЛАН 3

РЕФЕРАТ 6

ЗМІСТ 7

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ 9

ВСТУП 10

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПЗ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ .. 12

1.1 Ознайомлення з предметною областю 12

1.1.1 Актуальність теми 12

1.1.2 Переваги формату аукціону порівняно з звичайними прямими продажами 13

1.2 Огляд існуючих додатків 13

1.2.1 Prozorro.Sale 13

1.2.2 uBiz 14

1.2.3 Copart 15

1.2.4 Menheim 16

1.3 Порівняння аналогів 17

1.4 Постановка технічного завдання 19

2. ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ 21

2.1 Засоби програмної реалізації 21

2.1.1 Середовище розробки JetBrains Rider 21

2.1.2 Мова програмування – C# 22

2.1.3 Платформа розробки – .NET 23

2.1.4 Фреймворк для побудування API – ASP.NET 24
2.1.5 Фреймворк для роботи з БД з коду (ORM) – EntityFramework 25
2.1.6 База даних – PostgreSQL 26
3. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ 28
3.1 Проектування діаграми використання 28
3.2 Проектування бази даних 30
3.3 Проектування внутрішньої архітектури додатку 32
3.4 Тестування програмного продукту 36
3.4.1 Юніт-тестування 37
3.4.2 Інтеграційне тестування 41
4. ВИСНОВКИ 52
5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ 53
6. ДОДАТОК А 55

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

ПЗ – програмне забезпечення

ПК – Персональний комп’ютер

БД – База даних

Back-end – серверна частина додатку

API (або інтерфейс програмування застосунків) - це набір правил та протоколів, що дозволяють двом або більше програмам взаємодіяти між собою та обмінюватися інформацією. В основному, API використовується для створення програмного забезпечення, що має можливість взаємодіяти з іншими додатками та сервісами через мережу Інтернет.

API ендпоінт - це кінцева точка (адреса) веб-сервера, за допомогою якої можна отримати доступ до даних або функцій, що пропонуються через певний API (Application Programming Interface). Вони використовуються для здійснення HTTP запитів до веб-сервера з деякими параметрами (зазвичай в форматі JSON або XML), відповідно до встановленого протоколу. За допомогою API ендпоінта можна отримати доступ до різноманітних функцій і даних, таких як статистика відвідувань, списки товарів, географічні координати та інше.

ВСТУП

Всі більша кількості компаній у всьому світі прагнуть впроваджувати новітні технології та переходити до цифрової форми проведення бізнесу. Одною з сфер, де це може бути актуально, є автомобільні аукціони. Зараз проведення аукціонів в традиційній формі вимагає значних витрат на забезпечення безпеки, оренду приміщення та інфраструктури, а також на управління бізнес-процесами. Тому зацікавленість до автомобільних онлайн аукціонів зростає кожного дня.

І тому метою цієї дипломної роботи розробка серверного додатку для забезпечення проведення автоматизованого аукціону. Одним із плюсів даного додатку що на справді він є тільки мозоком аукціону, а візуальна частина може бути зроблена будь-яка та за допомогою будь яких технологій що підтримують клієнт-серверну комунікацію за допомогою REST API, наприклад: веб-сервіс, мобільний додаток, додаток для персонального комп’ютера, або навіть інший бек-енд додаток.

Для реалізації поставленої мети потрібно вирішити наступні завдання:

- Проаналізувати вимоги, потреби, технічні можливості потенційної аудиторії даного програмного продукту, переваги та недоліки існуючих рішень
- Провести аналіз інструментів та програмних засобів реалізації, що покажуть найкращу продуктивність системи, під час застосування продукту в реальних умовах, та що надають найбільшу зручність під час написання продукту
- Спроектувати та розробити серверний додаток для організації проведення онлайн аукціонів
- Провести End-To-End тестування додатку на предмет недоліків, помилок та не валідної поведінки.

ПЗ було розроблено за допомогою мови програмування C#, фреймворку ASP.NET, СУБД PostgreSQL, та інших не великих бібліотек. C# - наразі одна із найпопулярніших мов програмування для створення ентерпрайз бізнес-додатків. Я обрав її для проекту тому що після аналізу вимог, стало зрозуміло що ця мова

буквально була створена для подібних задач. В тандемі з ASP.NET це неймовірно потужний інструмент для створення різноманітних додатків, в тому числі серверних API додатків, яким і є мій проект.

Фіналом цієї роботи стане додаток, який зможе спростити бізнес, зменшити витрати часу та грошей, автоматизувати процеси та компаніям які продають автомобілі та спростити процес їх росту.

1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПЗ ДЛЯ РЕАЛІЗАЦІЇ СИСТЕМИ

1.1 Ознайомлення з предметною областю

1.1.1 Актуальність теми

Тема цифровізації продажу автомобілів за допомогою онлайн-аукціонів є дуже актуальною та має великий потенціал для розвитку в Україні і у цього є певні причини. Наприклад те що сьогодні неймовірно популярна купівля вживаних авто зі всього світу, та продаж їх в Україні вже в форматі «під ключ» або на онлайн майданчиках продажу авто, трішки менш популярний варіант це продаж авто на так званих «авто-базарах». І не зважаючи на переваги які будуть описані нижче, більшість сервісів з продажу авто все ще не мають у себе подібного функціоналу, окрім окремих прецедентів як Prozorro.Sale, і то варто відзначити що вони організовані лише для державних активів та локалізовані регіонально тільки в Україні.

Що дасть наявність онлайн аукціонів Українцям? Наприклад може значно полегшити та прискорити процес купівлі-продажу автомобілів в Україні, зменшивши час і зусилля, які зазвичай затрачаються на відвідування авто-базарів та інших традиційних майданчиків. За допомогою онлайн-аукціонів можна також більш ефективно здійснювати експорт та імпорт автомобілів. Крім того, це може збільшити конкуренцію та знизити ціни на автомобілі в Україні, що сприятиме розвитку ринку автомобілів в цілому. Однак, для успішного впровадження онлайн-аукціонів необхідна підтримка від держави та спеціалізованих організацій, а також висока якість технічної інфраструктури та безпеки даних, що може стати викликом для компаній, які хочуть розпочати свою діяльність в цьому напрямку.

1.1.2 Переваги формату аукціону порівняно з звичайними прямими продажами

– Швидкий продаж: зазвичай аукціони проводять в чітко визначених термінах що обмежує час покупців на розмислення, і це змушує потенційних покупців більш активно брати участь в торгах. Через це хоч велика вірогідність що хоч один покупець на авто буде в кінці терміну аукціону, що дуже збільшує обіг продажу автомобілів
– Кращі ціни: цей аргумент може працювати як у сторону продавця, так і покупця. Аукціони дають шанс продати авто за більш привабливу ціну за рахунок конкуренції за товар, що може збільшити прибуток для продаця. Але також за відсутності великої конкуренції для покупця є можливість купити авто дешевше ніж він купляв би на авто-ринку чи з рук.
– Доступність великого кола покупців: за рахунок проведення аукціону онлайн, дуже підвищується охоплення потенційними клієнтами зі всієї України та можливо навіть світу.

1.2 Огляд існуючих додатків

Додатків які спеціалізовано займаються продажом авто за допомогою аукціонів майже відсутні. Більшість займаються купівлею авто з аукціонів за кордоном, та привозять їх в Україну або «під ключ» для замовника, або для продажів звичним способів за допомогою майданчиків для продажу. Тому порівняння буде з існуючими додатками у всьому світі.

1.2.1 Prozorro.Sale

Prozorro.Sale - це державна електронна платформа, що пропонує механізми продажу різних державних активів, зокрема автомобілів. Ця система розроблена для забезпечення прозорості, конкуренції та ефективності у процесі продажу майна

державних органів та підприємств. Крім того, Prozorro.Sale дозволяє здійснювати електронну реєстрацію на аукціони та контролювати весь процес торгів від початку до кінця. Це дозволяє максимально оптимізувати і спростити процес продажу майна та забезпечує повну відкритість та чесність у проведенні торгів.

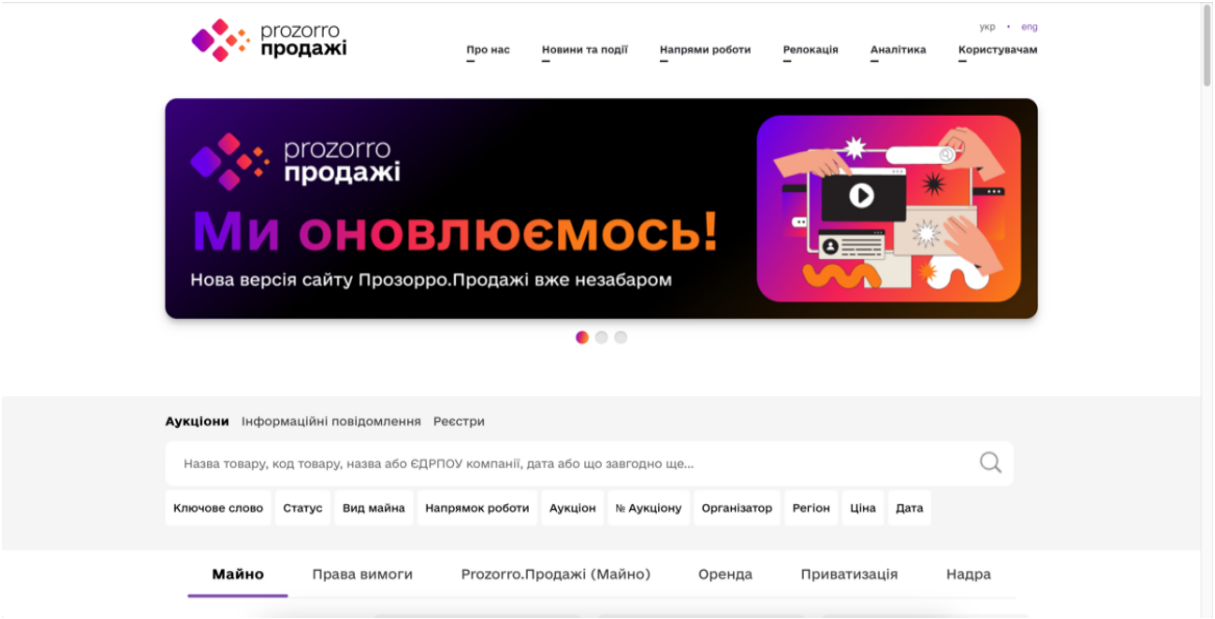


Рисунок 1.2.1 - Головна сторінка Prozorro.Sale

1.2.2 uBiz

UBiz - це онлайн-платформа та учасник платформи Prozorro для проведення електронних торгів з продажу різноманітного майна, включаючи автомобілі, нерухомість, обладнання та інше. Сервіс є одним з лідерів в Україні в сфері електронних торгів та має досить високу популярність серед продавців та покупців.

Сервіс дає можливість проводити електронні торги в режимі реального часу, що забезпечує прозорість та ефективність процесу. Продавці можуть додати свої лоти на платформу, а покупці можуть знайти багато пропозицій для покупки майна різного типу та цін. В той же час, UBiz надає розумні інструменти для зручної та швидкої покупки майна, що забезпечує зручність та ефективність процесу.

Однією з головних переваг UBiz є те, що сервіс має велику кількість

користувачів, що дозволяє продавцям максимально розширити аудиторію своїх лотів та забезпечити їх більш ефективну реалізацію. Крім того, UBiz забезпечує покупцям можливість отримати детальну інформацію про лоти, що сприяє ухваленню правильного рішення при покупці.

Однак, серед недоліків UBiz можна відзначити плату за використання сервісу, яка може бути значною для деяких користувачів. Крім того, процес проведення торгів може бути складним та потребувати певного рівня знань з цієї сфери.

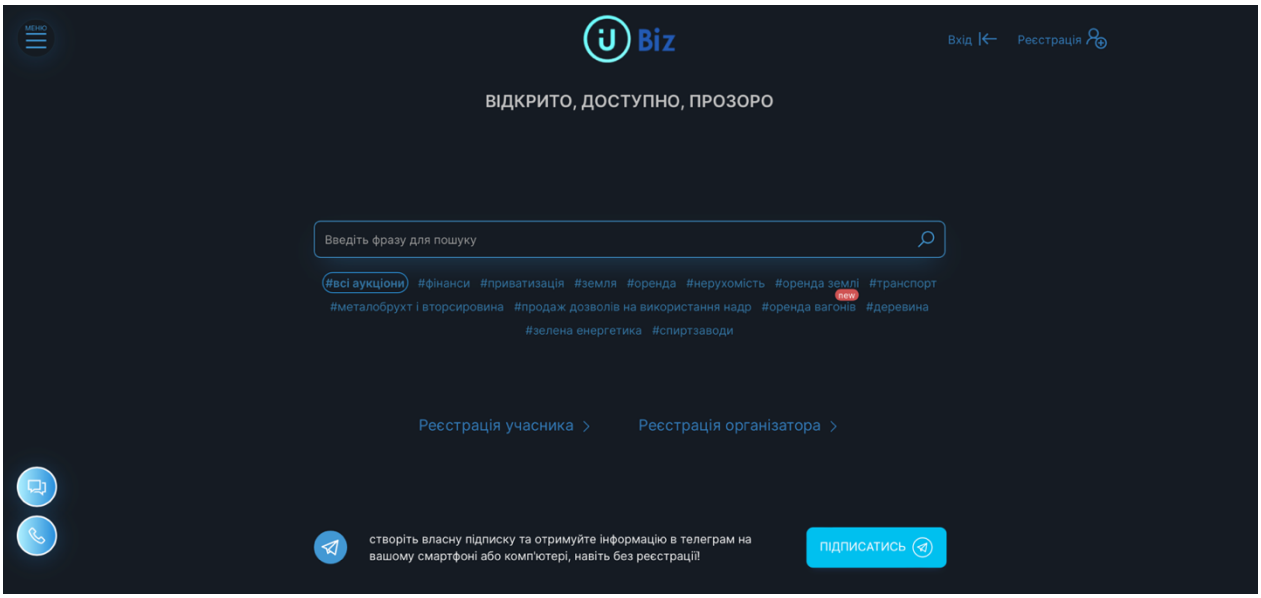


Рисунок 2.2.2 - Головна сторінка Ubiz.ua

1.2.3 Copart

Copart – це один з найбільших в світі інтернет-аукціонів підтриманих автомобілів. Компанія пропонує понад 125 тисяч лотів на тиждень, що включає в себе автомобілі, мотоцикли, судна та інші транспортні засоби. Copart забезпечує доступ до міжнародної бази даних з продажу автомобілів, що дає можливість придбати транспорт за конкурентною ціною зі всього світу.
Крім продажу підтриманих автомобілів, Copart також надає послуги з вилучення, транспортування та зберігання транспортних засобів. Компанія має розвинуту мережу філій по всьому світу, що дозволяє їм оперативно проводити операції з продажу автомобілів в різних країнах. Крім того, Copart пропонує можливість зареєструватися

як дилер та отримувати спеціальні пропозиції та знижки на придбання автомобілів на аукціоні.

Відома також як лідер у галузі переробки схемного металобрухту та утилізації автомобілів, Copart здійснює роботу зі збору, переробки та продажу матеріалів зі складів викидів.

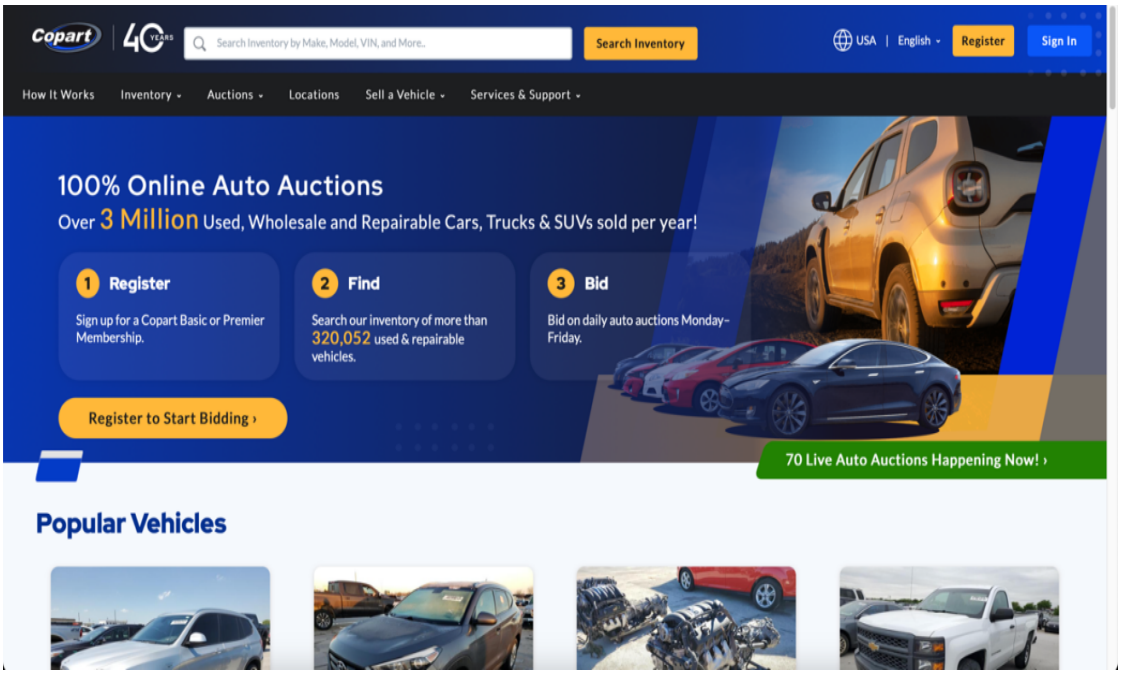


Рисунок 1.2.3 - Головна сторінка Copart

1.2.4 Menheim

Menheim – найбільший в світі аукціон підтриманих автомобілів, що належить до компанії Cox Automotive. Manheim надає послуги з продажу транспортних засобів на аукціонах в США, Канаді, Великобританії та Європі, включаючи інтернет-аукціони та аукціони у форматі "живого" присутності.
В світі Manheim відомий не тільки своїми аукціонами, а й додатковими сервісами, такими як страхування, фінансування та логістичні послуги. Крім того, Manheim дозволяє своїм клієнтам здійснювати продажі на власних майданчиках, а також використовувати платформу для продажу транспортних засобів у віртуальному режимі. Найбільші автомобільні дилерські мережі світу вважають Manheim важливим

партнером, який допомагає забезпечувати їхню потребу у продажу авто. Відмінні послуги та широкий географічний охоплюють Manheim є серед найбільш конкурентоспроможних гравців у сфері продажу транспортних засобів.

1.3 Порівняння аналогів

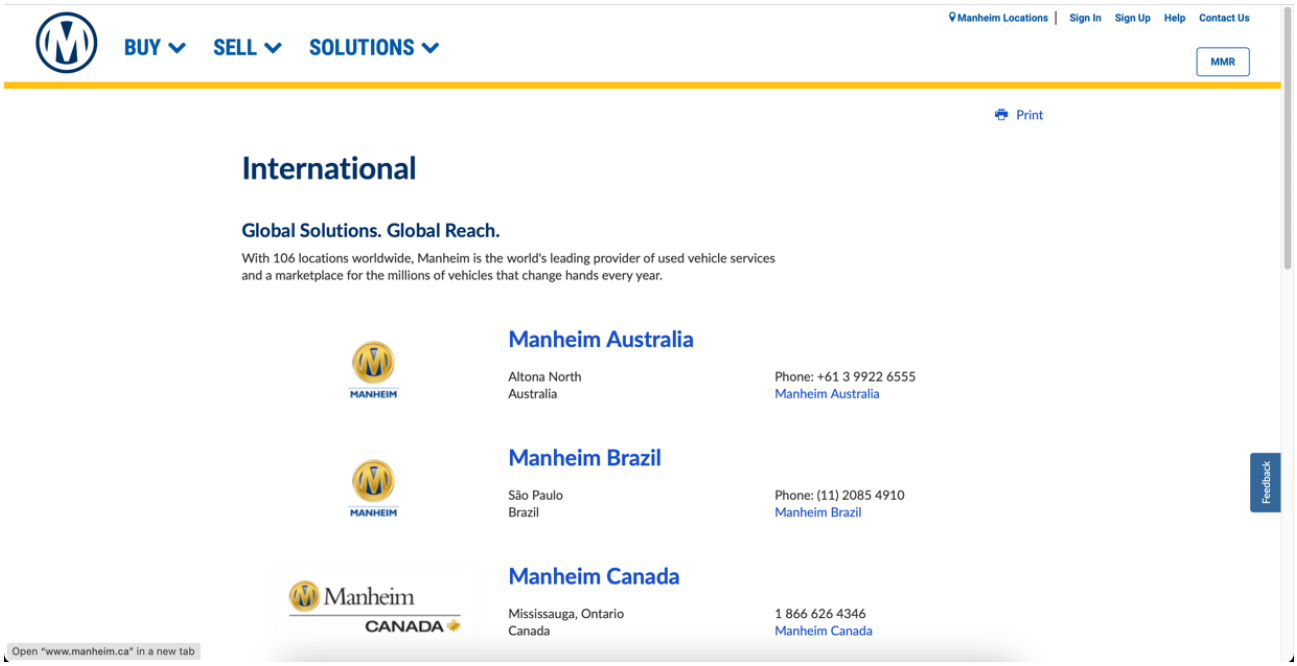


Рисунок 1.2.4 - Головна сторінка Menheim

Таблиця 1.3 – Порівняння додатків за головними пунктами

	Prozorro.Sale	Ubiz	Copart	Menheim
Регіон	Україна	Україна	США, Канада, Великобританія, Деякі країни Європи та Азії	США, Канада, Великобританія, Європа
Переваги	Забезпечує прозорість та конкуренцію	Забезпечує прозорість та конкуренцію	Великий вибір автомобілів, міжнародна присутність	Великий вибір автомобілів, міжнародна присутність
…	…	…	…	…

Продовження таблиці 1.3 – Порівняння додатків за головними пунктами

…	…	…	…	…
Недоліки	Організований лише для державних активів, обмежений регіонально, мала кількість пропозицій	Обмежений регіонально, висока конкуренція. великі гарантійні внески, велика присутність бізнесу, та мала звичайних фіз. осіб	Високі комісійні сбори та вартість доставки, складність доставки, обмежена доступність деяких аукціонів	Високі ціни та конкуренція, обмежений доступ для громадян не з США, складність доставки
Наявність мобільного додатку	+	+	+	+
Мови	Українська, Англійська	Українська, Англійська, російська	Багатомовна	Багатомовна
Розмір комісії	Відсутня для покупців, комісію оплачує продавець, та вона залежить від типу майна (0.5%-1%)	Відсутня для покупців, комісію оплачує продавець, та вона залежить від типу угоди (1%-2%)	$25-$900 якщо ціна до $15000, від $15000 комісія 7,5%	$300 + 2% якщо ціна від $0 до $2000, від $2000 комісія складатиме $400 + 2%

1.4 Постановка технічного завдання

Додаток для організації аукціону з продажу підтриманих авто має бути реалізований у форматі серверного API додатку з можливість інтегрування будь-якого клієнту, наприклад Веб-додаток, мобільний додаток, додаток для ПК. Серед конкретних вимог можна назвати наступні:

Має реалізовувати базовий функціонал який необхідний для будь-якого додатку-клієнта

- Функціонал реєстрації та авторизації користувача
- Функціонал підтвердження реєстрації
- Функціонал пагінації відповідей сервісу
- Слідувати протоколам та правилам REST API
- Обмежувати доступ за типом користувача
- Валідація запитів зі сторони додатків-клієнтів

Має реалізовувати функціонал аукціону зі сторони користувача

- Створення лотів аукціону
- Створення ставок на вже існуючи лоти
- Додавання автомобілів з їх даними для подальшого продажу
- Автоматизовану детермінацію переможця аукціону

Має реалізовувати функціонал аукціону зі сторони адміністратора

- Додавання/Отримання/Оновлення/Видалення інформації про автомобільні бренди, моделі та колір
- Додавання/Отримання/Оновлення/Видалення інформації про лоти та ставки
- Додавання/Отримання/Оновлення/Видалення інформації про користувачів

Має реалізовувати функціонал зберігання даних та доступу до них

- СУБД для зберігання всіх даних
- Надійне зберігання персональних даних користувачів
- Шифрування паролів
- Структура БД має бути достатньо нормалізована там мати розширювану структуру.

2. ВИБІР ЗАСОБІВ ПРОГРАМНОЇ РЕАЛІЗАЦІЇ

2.1 Засоби програмної реалізації

2.1.1 Середовище розробки JetBrains Rider

Рисунок 2.1.1.1 - Логотип JetBrains Rider

JetBrains Rider – це потужне інтегроване середовище розробки (IDE) для платформи .NET і різних мов програмування, включаючи C#, F# і VB.NET. Серед ключових переваг можна відзначити високу продуктивність та швидкість розробки завдяки вбудованим інструментам і підтримці мультиплатформенності. Rider пропонує повноцінну підтримку для платформи .NET Core, включаючи розробку під Linux і macOS що має вбудований дебагер з підтримкою майже всіх типів .NET-додатків, включаючи ASP.NET, Unity і Xamarin.

Інші важливі можливості Rider включають в себе розширену підтримку рефакторінгу коду, автоматичне завершення кодування, інтеграцію з іншими інструментами розробки, такими як інтерфейс системи контролю версій git або SVN, а також підтримку різних мов програмування в одному проекті. Також середовище розробки дозволяє працювати з базою даних прямо з того ж проекту, не вимагаючи для цього окремого додатку чи навіть встановки плагіну, що значно допомагає при роботі з сховищем даних при розробці.

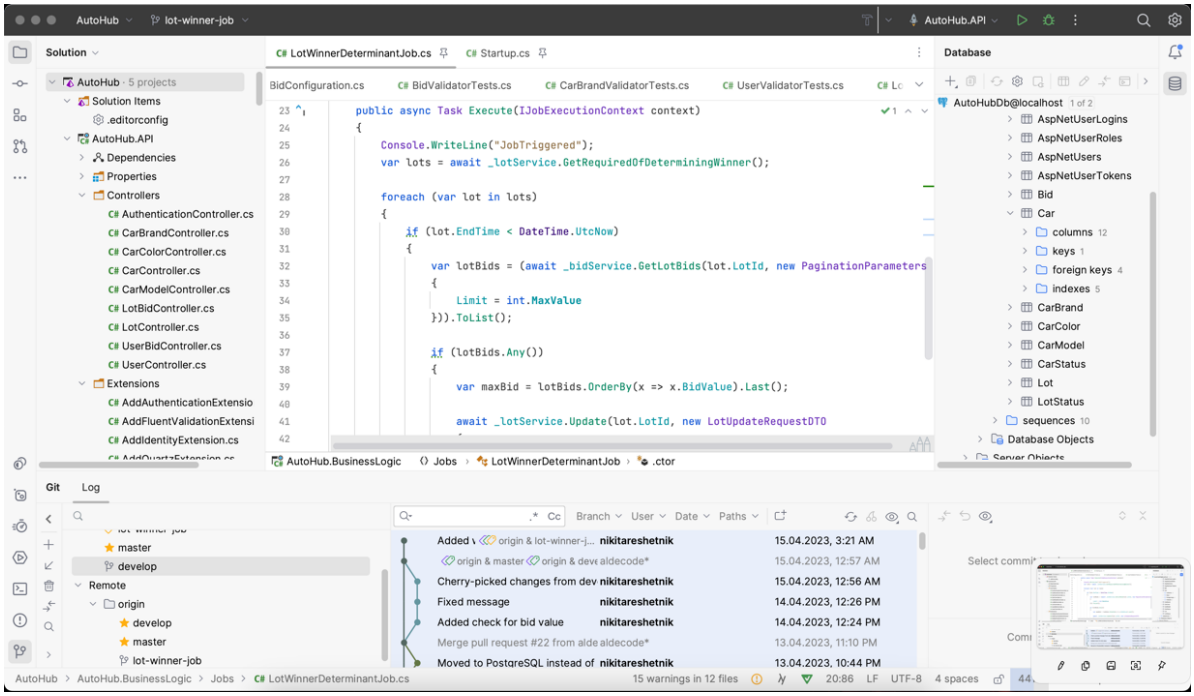


Рисунок 2.1.1.2 – Інтерфейс JetBrains Rider

2.1.2 Мова програмування – C#

Рисунок 2.1.2 - Логотип C#

C# (найчастіше вживане прочитання – «сі шарп») - це універсальна високорівнева об'єктно-орієнтована мова програмування, яка була розроблена компанією Microsoft, що тісно інтегрується з платформою .NET, яка одночасно є платформою та середовищем виконання. Тісна інтеграція з платформою є одною із

головних переваг, тому що дає можливість використовувати бібліотеки класів та компонентів .NET, що значно спрощує процес розробки програм, та зменшує кількість витраченого часу на написання коду. Ще однією перевагою C# є його безпека, так як мова програмування дозволяє перевіряти типи даних під час компіляції, що допомагає уникнути багів та помилок на етапі виконання програм. Крім того, C# підтримує поліморфізм та наслідування, що дозволяє створювати більш складні та функціональні програми з меншим обсягом коду. І на останок, C# має величезну активну спільноту та велику кількості детальної документації та підручників. Це дозволяє легко знайти відповіді на будь-які запитання та швидко розв'язувати проблеми під час розробки програм.

2.1.3 Платформа розробки – .NET

Рисунок 2.1.3 - Логотип .NET

.NET (найчастіше вживане прочитання – «дотнет») - це платформа з відкритим кодом для групи мов програмування яка об’єднує спільні бібліотеки, інструменти та фреймворки для розробки додатків різних видів та призначень.

В минулому, .NET найчастіше використовувалась для написання програм для систем Windows, але з подальшим розвитком, з’явилась мульти-платформена версія .NET Core, а пізніше після злиття .NET Framework та .NET Core стала просто .NET.

Платформа має широкий спектр задач які вона може вирішувати:

- Веб-додатки – ASP.NET MVC, Razor Pages, Blazor WASM
- API – ASP.NET Web API
- Мобільні додатки – Blazor, Xamarin, MAUI
- Компьютерні додатки – WinForms, WPF, UWP, Blazor, MAUI
- Штучний інтеллект – ML.NET, .NET for Apache Spark
- Розробка ігор – Unity
- Інтернет речей – ARM32, ARM34
- Хмарні технології – Azure

2.1.4 Фреймворк для побудування API – ASP.NET

Рисунок 2.1.4 - Логотип ASP.NET

ASP.NET є однією з найбільш популярних платформ для розробки веб-додатків на мові програмування C#. Основною перевагою ASP.NET є його висока продуктивність та можливість розробки масштабованих веб-додатків. Крім того, ASP.NET має багатий функціонал, який дозволяє розробникам швидко створювати веб-сайти та додатки з використанням готових компонентів та бібліотек. ASP.NET також має вбудовану підтримку безпеки, що дозволяє захистити веб-додатки від потенційних загроз, таких як атаки з використанням SQL-ін’єкцій або XSS. Крім того, ASP.NET дозволяє розробникам швидко створювати RESTful API та інтегруватись з іншими технологіями, що робить його однією з найбільш гнучких та розширюваних платформ для веб-розробки.

2.1.5 Фреймворк для роботи з БД з коду (ORM) – EntityFramework

Рисунок 2.1.5 - Логотип EntityFramework

Entity Framework (EF) є фреймворком для розробки програмного забезпечення на платформі .NET, який дозволяє розробникам працювати з базами даних з використанням об'єктно-орієнтованого підходу з багатьма існуючими СУБД, включаючи Microsoft SQL Server, Oracle, MySQL та PostgresSQL..

Основні переваги Entity Framework полягають у тому, що він дозволяє легко створювати моделі даних, управляти зв'язками між об'єктами, виконувати операції з базою даних і робити це все з використанням мови C#.

Більш того, Entity Framework підтримує різні підходи до розробки:

– Code First - це підхід, при якому база даних створюється на основі коду, написаного на C#. Потрібно визначити модель або сутність в класі C# та написати клас контексту для роботи з базою даних. Цей підхід найчастіше використовують програмісти на C# і я його буду використовувати у своєму проєкті.
– Database First - передбачає створення бази даних спочатку, а потім моделі бази даних EDMX. Файл .edmx містить інформацію про структуру бази даних, модель даних та її відображення, і може бути використаний у Visual Studio для генерації коду.
– Model First – передбачає першочергове створення моделі графіку EDMX, а

потім класи моделі C# створюються автоматично за лаштунками. База даних створюється з графіка EDMX. Цей підхід не потребує знання синтаксису SQL або C#.

Крім того, EF має ряд додаткових функцій, які роблять його ще більш привабливим для розробників, наприклад, забезпечує можливість використовувати LINQ для написання запитів до бази даних в зручному для програміста синтаксисі. LINQ to Entities дозволяє створювати складні запити, що пов'язують дані з різних таблиць та взаємодіють з різними асоціаціями. Entity Framework перетворює вирази LINQ to Entities на запити SQL, які може виконати конкретна база даних. Це забезпечує оптимальну продуктивність та ефективність роботи з базою даних. Крім того, Entity Framework дозволяє використовувати ORM-технології та забезпечує зручний механізм маппінгу об'єктів бази даних на об'єкти C#.

2.1.6 База даних – PostgreSQL

Рисунок 2.1.6 - Логотип PostgreSQL

PostgreSQL - це потужна об'єктно-реляційна система управління базами даних, яка є однією з найбільш популярних відкритих баз даних. Також вона має високу стабільність та безпеку, що робить її ідеальним вибором для великих та критичних за даними проектів. Одна з переваг цієї СУБД полягає у тому, що вона є дуже

розширюваною, дозволяючи розробникам створювати власні функції та типи даних. Ще однією важливою перевагою PostgreSQL є її висока продуктивність та швидкість роботи з даними. Більшість запитів до бази даних виконуються дуже швидко, завдяки чому вона стала популярною для використання в веб-проектах та додатках з великим обсягом даних.

Крім того, PostgreSQL має велику спільноту розробників та користувачів, яка надає багато документації та підтримки. Також ця система управління базами даних є безкоштовним та відкритим програмним забезпеченням, що робить його доступним для використання в будь-яких проектах, незалежно від їх розміру та складності.

3. РОЗРОБКА ТА ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Проектування діаграми використання

Діаграми використання допомагають визначити контекст і вимоги всієї системи або її частин, і показують взаємодію між системою та акторами. Де актор це – будь-яка сутність що взаємодіє з системою. Дана діаграма описує функції та область застосування системи, а додатки та актори на діаграмах описують, що робить система і як учасники її використовують.

Діаграми варіантів використання також можуть бути корисними під час збору вимог, фази аналізу та проектування, тестування тощо. Використання діаграм дозволяє визначити класи, які потрібні системі, визначити тести для системи та інші важливі етапи розробки системи. Однак, варто пам'ятати, що діаграми використання не описують, як саме система працює всередині.

В даному додатку Актором буде виступати додаток-клієнт, який буде викликати ендпоінти мого додатку. Окрім акторів треба визначити основні варіанти використання системи:

- Робота з користувачами
- Робота з аутентифікацією
- Робота з автомобілями
- Робота з брендами автомобілів
- Робота з моделями автомобілів
- Робота з кольорами
- Робота з лотами
- Робота з користувачами
- Робота з ставками користувачів
- Робота з ставками лотів

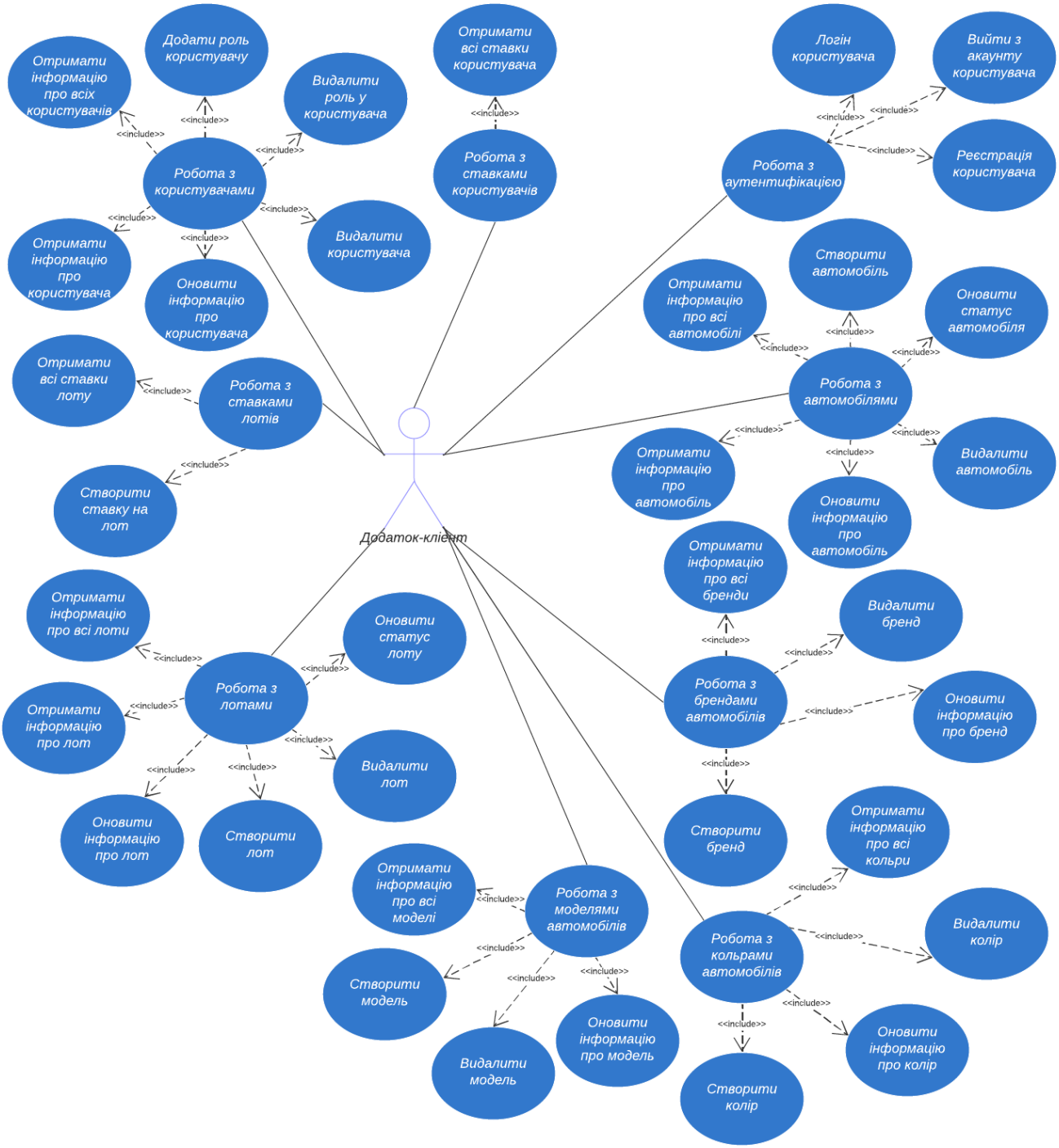


Рисунок 3.1 Діаграма використання додатку

В даній діаграмі ми можемо бачити всі основні функції додатку які зможе використовувати додаток-клієнт, в основному це робота з даними за допомогою ендпоінтів API.

3.2 Проектування бази даних

Діаграми бази даних є графічними інструментами, які допомагають візуалізувати структуру бази даних і її взаємозв'язки між ними. Вони дозволяють описати сутності та їх взаємозв'язки в базі даних, а також показати, як дані зберігаються та обробляються в системі. На основі діаграми використання ми можемо зробити наступну діаграму баз даних яка містить в собі всі сутності необхідні для взаємодії з додатком:

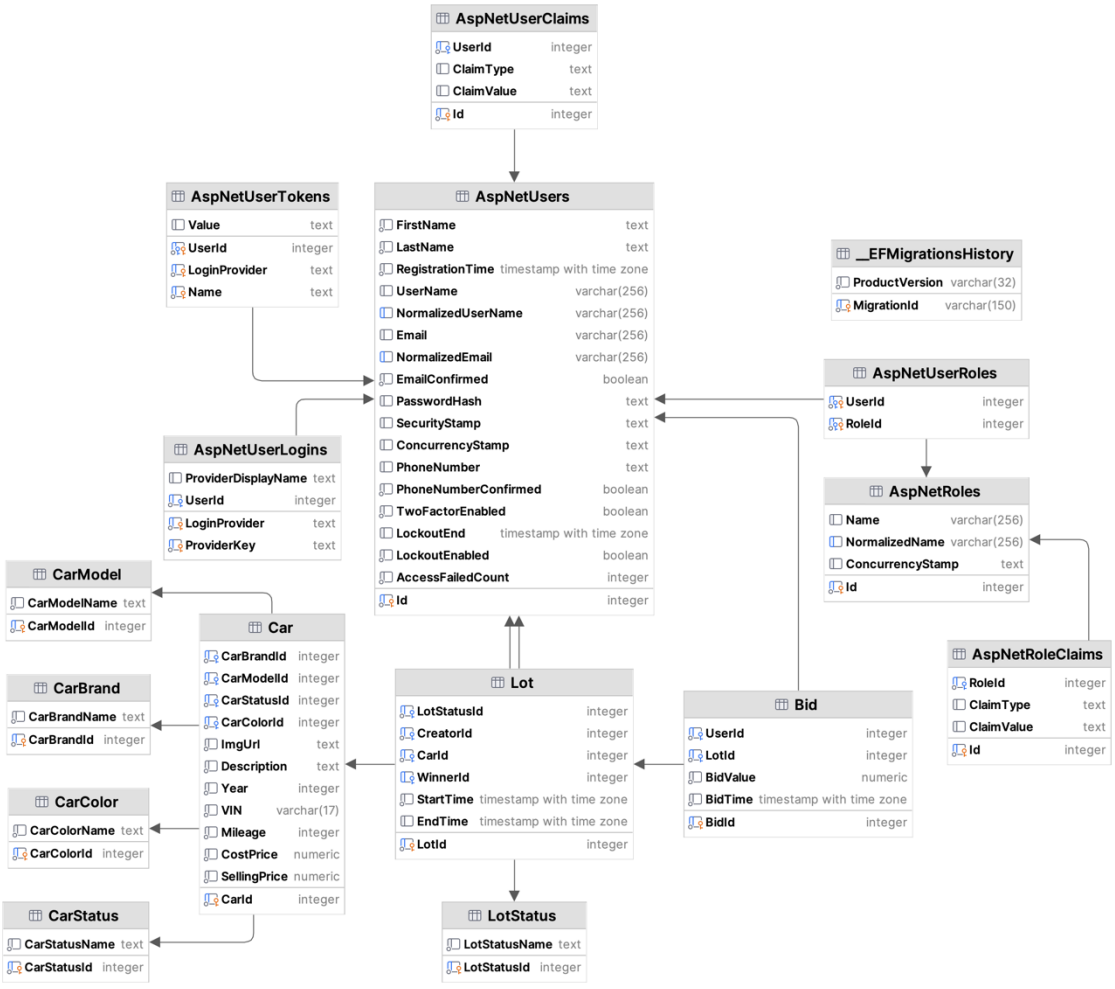


Рисунок 3.2 – Діаграма бази даних додатку

В даній діаграмі є декілька основних елементів, таких як таблиці, та зв’язки між ними. Таблиці є першим з основних об'єктів в базах даних. Вони представляють собою сукупність структурованих даних, організованих у вигляді рядків і стовпців. Кожна таблиця в базі даних містить певну кількість стовпців (також називають полями) та рядків (також називають записами або кортежами). Кожен стовпець має свій унікальний ідентифікатор (назву) і тип даних, який відповідає за те, який тип даних може бути збережений у цьому стовпці. Рядки таблиці містять фактичні значення даних, які відповідають за один запис у таблиці. Також при проектуванні бази даних, таблиці повинні бути розроблені таким чином, щоб вони були нормалізовані, наприклад не містити повторюваних даних та мати первинний ключ, які можуть призвести до проблем зі збереженням та отриманням даних. Також важливо визначити ключі, які дозволяють ідентифікувати унікальні записи в таблиці та забезпечувати взаємозв'язок між таблицями в базі даних.

Зв'язки в базах даних використовуються для встановлення взаємозв'язку між таблицями. Вони дозволяють об'єднувати дані з різних таблиць в один запит, що забезпечує ефективне зберігання та управління даними. Зв'язки бувають різних типів, таких як один до одного, один до багатьох та багато до багатьох, і вони можуть бути встановлені за допомогою первинних та зовнішніх ключів. Вірно налаштовані зв'язки дозволяють відносно легко отримувати потрібну інформацію з бази даних.

Зазвичай ці діаграми створюються на самому початку проектування системи, оскільки вона допомагає описати сутності які будуть використовуватись в системи, їх поля, та зв’язки один між одним. Вже з наявною діаграмою баз даних набагато зручніше створювати саму базу даних та код додатку, більш того додаток вже може використовувати тестові дані для тестування нового функціоналу. В більшості випадках ця діаграма обов’язкова оскільки в сучасному світі майже відсутні системи які не зберігають дані та не мають бази даних.

3.3 Проектування внутрішньої архітектури додатку

У сучасному світі, коли додатки виконують роль ключового елемента бізнесу, його надійність, стійкість, підтримуваність та потенціал до розширення є невід'ємною частиною його успішної роботи. За останні десять років підходи мінялись та розроблялись все більш новітні та різноманітні підходи для кожної з задач.

В даному додатку в якості архітектури додатку було використано багатошарову або N-Tier архітектуру, оскільки ця архітектура найдоцільніша в додатку даного розміру та функціоналу, за рахунок чого зберігається простота архітектури та реалізації, але при цьому вона надає необхідні переваги.

У розробці програмних додатків – багаторівнева архітектура, це клієнт-серверна архітектура в якій використовується підхід розділення відповідальності окремих частин додатків, які з’єднані між собою послідовно, та кожен компонент знає там має доступ тільки тих що поруч по ієрархії. А моєму додатку я обрав розділення архітектури на наступні «шари», всього їх 3:

1. Рівень API (Presentation Layer) – це самий перший та верхній рівень який представляє собою інтерфейс взаємодії з серверним додатком за допомогою API ендпоінтів. Його основна задача – прийняття запитів від додатків клієнтів на якісь дані або дії, та повертання інформації для подальшого відображення на додатку клієнті.
2. Рівень бізнес-логіки (Bussiness Logic Layer) – це сервісний рівень який в собі містить всю бізнес логіку додатку. Саме в цьому рівні відбуваються всі необхідні потрібні фільтрації, обчислення, та дії які були зініційовані клієнтом або самим додатком.
3. Рівень доступу до даних (Data Access Layer) – цей рівень відповідає за доступ та взаємодію з постійними сховищами даних. Саме він зберігає в собі логіку запитів до баз даних, та сутностей що в ній зберігаються.

Ця варіація рівнів є базовою, але не являє собою обмеження в кількості шарів, що означає те, що в залежності від потреб додатку, кількість шарів може збільшуватись або зменшуватись в майбутньому.

При застосуванні даного підходу є багато переваг, наприклад:

- Безпека: оскільки клієнт може отримати доступ до рівня даних лише через логічний рівень, це зменшує точку входу та захищає багато небезпечних системних функцій.
- Розподілена розробка: Також завдяки цьому підходу паралельна розробка додатку з іншими розробниками стає більш зручною, через меншу кількість конфліктів при змінах.
- Модульність та ізольованість: за допомогою модульності додатку, ми можемо відокремлювати різні частини додатку в різі рівні, що позбавляє нас змішування логіки відображення, та доступу к даним. Кожен рівень має свою відповідальність, ми в цьому впевненні, і тому на рівні абстракцій легше розуміти структуру проекту.
- Підтримка: Розподілена структура архітектури сприяє зручності підтримки та масштабованості системи. Кожен є окремим елементом проекту, що зменшує ризики помилок в додатку, та полегшує зміни технічного стеку.

Але має і свої недоліки:

- Часом надлишкова абстракція: при використанні цієї архітектури, можуть виникати випадки, коли логіка переходів між шарами є надлишковою, наприклад коли клієнт просить повернути просто всіх користувачів. В даному випадку нам не потрібна ніяка фільтрація, чи бізнес логіка, але запит все ще має пройти від найвищого до найнижчого рівня, навіть якщо на кожному рівні використовується тільки виклик наступного
- Відносно низька швидкість роботи: з минулого пункту випливає що через те що часто є випадки коли дані просто переходять від шару до шару без логіки, що в свою чергу потребує зайвих ресурсів

- Дуплікація коду: часом для підтримування абстракцій на всіх рівнях може виникнути ситуація в потребі дублювання коду, та не потрібних операцій (наприклад Маппінгу). Але цей пункт можливо легко вирішити загальним рівнем для всіх інших рівнів, який буде в собі утримувати всю загальну логіку (Domain Tier)

Окрім того в даному додатку було виділено ще два окремих компонента які було б не правильно називати шарами тому що вони не приймають участь в обробці запитів та в основному виконують допоміжну функцію:

- Компонент доменних сутностей: цей компонент містить в собі доменні або «повні моделі» даних, константи, класи виключень, та будь-що що має загальне призначення та не відноситься конкретно до одного з шарів. До цього компонента доступ має кожен шар, але не навпаки.
- Компонент юніт-тестування: цей компонент містить в собі юніт-тести для тестування окремих частин додатку, та може містити в собі якусь окрему логіку, або допоміжні класи які призначені виключно для юніт-тестування додатку.

Виходячи з інформації описаної вище ми можемо описати архітектурну побудову додатку діаграмою пакетів, яка відображена на малюнку :

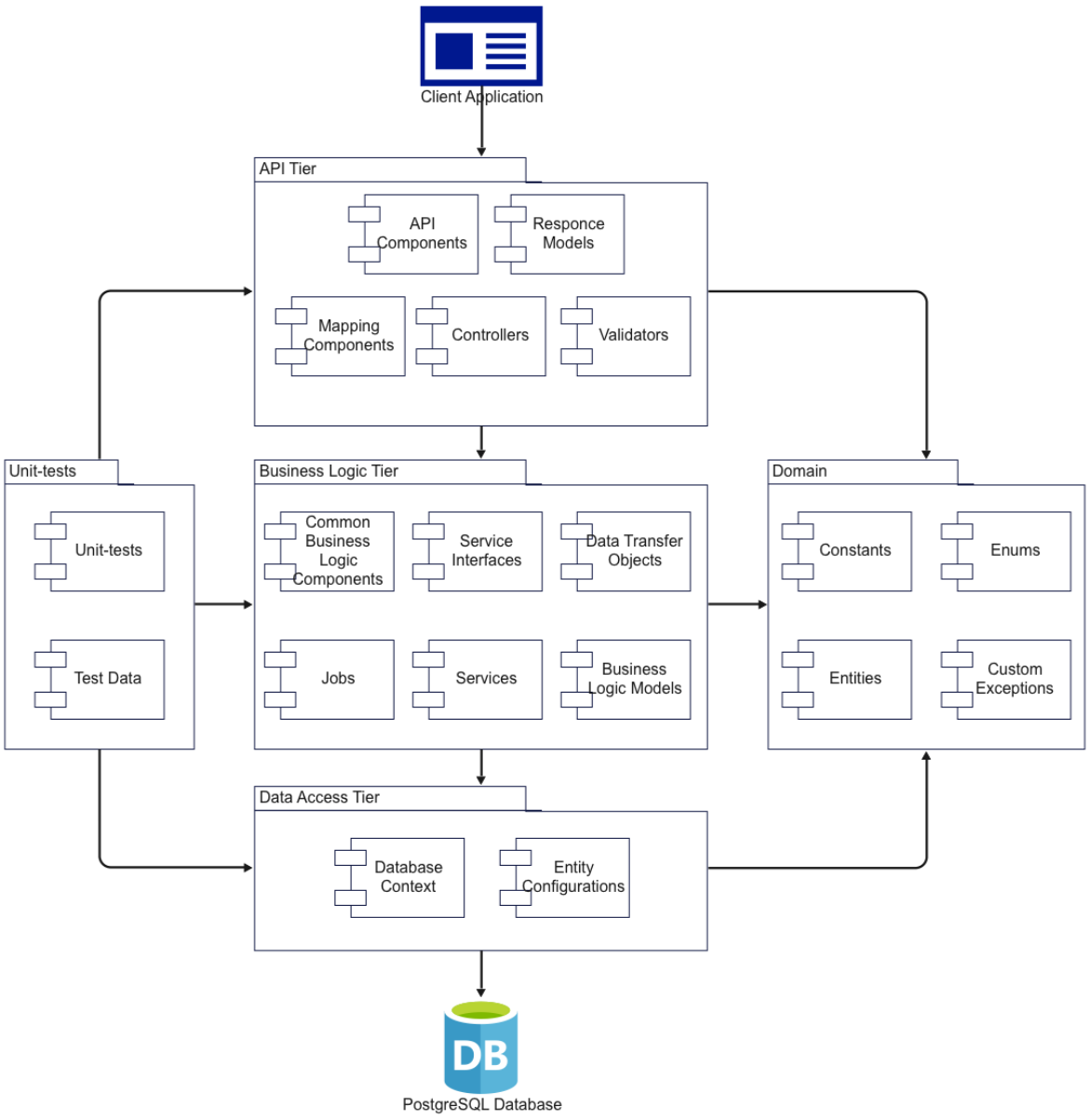


Рисунок 3.3 – Діаграма пакетів додатку

3.4 Тестування програмного продукту

Тестування програмного забезпечення (ПЗ) - це процес перевірки програмного продукту з метою виявлення помилок та відхилень від очікуваного функціоналу.

Цей процес є важливим етапом розробки, оскільки допомагає виявити помилки та проблеми у програмі ще до її випуску в маси. Це зменшує ризик виявлення проблем на етапі експлуатації додатку користувачами та забезпечує більш високу якість продукту. Крім того, тестування ПЗ допомагає забезпечити відповідність програмного продукту вимогам та специфікаціям, а також забезпечується зручна і проста в експлуатації інтерфейс.

Цей процес може включати різні типи тестів, такі як модульні, інтеграційні, системні та інші, які допомагають встановити якість ПЗ.

Модульне тестування – яке ще часто називають Юніт-тестуванням, являється перевіркою окремих функцій або модулів програмного продукту з метою виявлення та усунення помилок. Цей тип тестування допомагає встановити правильність роботи кожної окремої функції, які згодом знаходяться в складі більш складних модулів.

Інтеграційне тестування - це перевірка взаємодії між різними модулями програми для визначення правильності роботи системи в цілому. Цей тип тестування допомагає виявити помилки, які можуть виникнути в процесі взаємодії різних компонентів.

Функціональне тестування - це перевірка функціональності програмного продукту відповідно до вимог та специфікацій, в яких визначено його поведінку. Цей тип тестування допомагає забезпечити відповідність програмного продукту вимогам користувачів.

Системне тестування - це перевірка системи як цілого з метою виявлення проблем та помилок в поведінці системи в різних ситуаціях. Цей тип тестування допомагає визначити правильність роботи системи в різних умовах, а також виявити непередбачувані помилки.

Верифікаційне тестування - це перевірка відповідності програмного продукту визначеним вимогам та специфікаціям. Цей тип тестування допомагає визначити, чи виконує програмний продукт вимоги, встановлені на етапі розробки.

Стрес-тестування - це перевірка роботи програмного продукту в умовах тривалого навантаження. Цей тип тестування допомагає визначити, чи може програмний продукт працювати стабільно в умовах постійного навантаження.

Кожен з цих типів тестування має свою мету та спрямований на виявлення певних видів помилок та проблем. В залежності від вимог та потреб користувачів, розробники вибирають один або кілька типів тестування для забезпечення високої якості програмного продукту.

В даному додатку буде використовуватись автоматизоване тестування за допомогою юніт-тестів, та ручне інтеграційне тестування що являє собою ручний виклик кожного ендпоінта додатку та перевірка роботи на відповідність очікуваному результату.

3.4.1 Юніт-тестування

Юніт тестування в даному додатку реалізовано за допомогою одного з найпопулярніших фреймфорків – xUnit. xUnit є одним з найбільш популярних та безкоштовних фреймворків для тестування .NET-програм, і він використовується в багатьох великих проектах, в тому числі в проектах що використовуюит ASP.NET, Entity Framework та відкритих проектах .NET, таких як .NET Runtime і .NET SDK. Крім того, xUnit має активну спільноту, яка постійно розширює його можливості та підтримку. Один з головних принципів xUnit - це "Convention over configuration", що означає, що він має стандартні конвенції та правила, які спрощують налаштування тестів та в результаті пришвидшують їх написання. Одним із прикладом можно назвати те, що методи, що мають префікс "Test" в назві, будуть автоматично визнані як тести, і xUnit виконає їх під час запуску тестів. Такоє xUnit надає різноманітні можливості для налаштування тестів, включаючи налаштування параметрів тестів,

встановлення обмежень часу виконання тестів, обробку винятків та інше.

В даному випадку ми будемо використовувати його для написання юніт тестів для валідаторів які валідують інформацію що передає нам клієнт. Для повного покриття тестами валідаторів, достатньо всього 2 тести на кожен конкретний валідатор, один на позитивний результат, коли валідатор не знайшов помилки, і один на негативний коли валідатор побачив помилку в даних. Для показового прикладу візьмемо валідатор для моделі яку додаток отримує при створенні нового автомобіля приклад позитивного тесту можемо побачити на рисунку 3.4.2:

```
[Fact]
aldecode
public void CreateCarTestValidate_ValidModel_ShouldNotHaveError()
{
    //Arrange
    var model = new CarCreateRequest
    {
        CarBrand = "Audi",
        CarModel = "RS e-tron GT",
        CarColor = "Space Gray",
        CostPrice = 134500,
        SellingPrice = 141225,
        Mileage = 24631,
        VIN = "1HGCM66537A023172",
        Year = 2021,
        Description = "Description",
        ImgUrl = "audi.com/RS_etron_GT"
    };

    //Act
    var result = _createValidator.TestValidate(model);

    //Assert
    result.ShouldNotHaveValidationErrorFor(x => x.CostPrice);
    result.ShouldNotHaveValidationErrorFor(x => x.SellingPrice);
    result.ShouldNotHaveValidationErrorFor(x => x.Mileage);
    result.ShouldNotHaveValidationErrorFor(x => x.VIN);
    result.ShouldNotHaveValidationErrorFor(x => x.CarBrand);
    result.ShouldNotHaveValidationErrorFor(x => x.CarColor);
    result.ShouldNotHaveValidationErrorFor(x => x.CarModel);
    result.ShouldNotHaveValidationErrorFor(x => x.Year);
    result.ShouldNotHaveValidationErrorFor(x => x.Description);
    result.ShouldNotHaveValidationErrorFor(x => x.ImgUrl);
}
```

Рисунок 3.4.1.1 – Позитивний тест для валідатора

```
[Fact]
aldecode
public void CreateCarTestValidate_InvalidModel_ShouldHaveError()
{
    //Arrange
    var model = new CarCreateRequest
    {
        CarBrand = "",
        CarModel = "",
        CarColor = "",
        CostPrice = -240000,
        SellingPrice = -7101225,
        Mileage = -24631,
        VIN = "1HGCM66537A0231",
        Year = 1755,
        Description = "",
        ImgUrl = ""
    };

    //Act
    var result = _createValidator.TestValidate(model);

    //Assert
    result.ShouldHaveValidationErrorFor(x => x.CostPrice);
    result.ShouldHaveValidationErrorFor(x => x.SellingPrice);
    result.ShouldHaveValidationErrorFor(x => x.Mileage);
    result.ShouldHaveValidationErrorFor(x => x.VIN);
    result.ShouldHaveValidationErrorFor(x => x.CarBrand);
    result.ShouldHaveValidationErrorFor(x => x.CarColor);
    result.ShouldHaveValidationErrorFor(x => x.CarModel);
    result.ShouldHaveValidationErrorFor(x => x.Year);
}
```

Рисунок 3.4.1.2 – Негативний тест для валідатора

Даний додаток має всього 14 моделей даних що ми можемо отримати отримуємо від клієнта, та які потребують валідування, це означає що для повного покриття тестування валідаторів ми маємо написати всього 28 тестів щоб покрити цей функціонал на 100%.

Коли написання тестів завершено користуючись інтерфейсом Rider ми можемо запустити їх одною кнопкою та побачити результат:

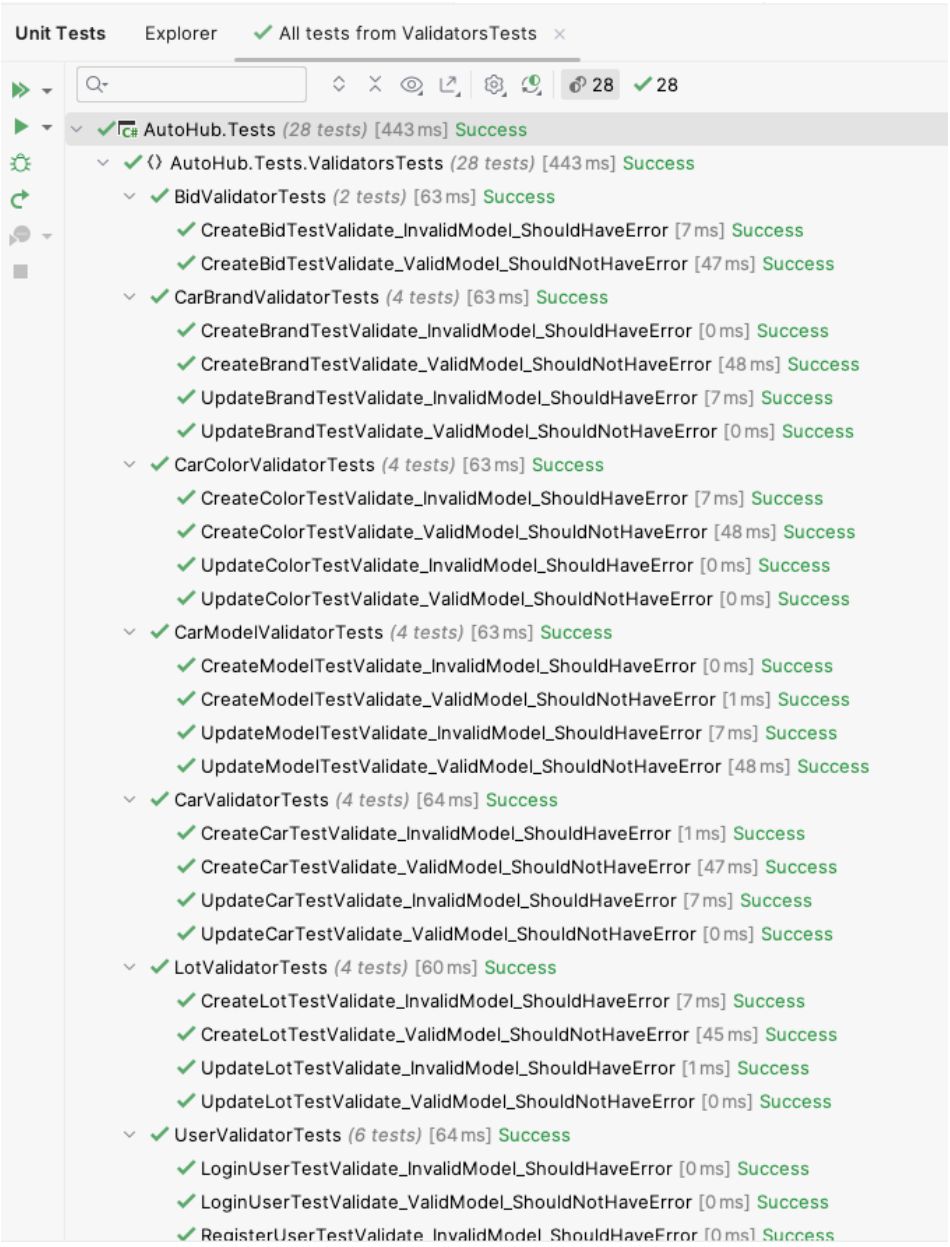


Рисунок 3.4.1.3 – Результат запуску тестів

На рисунку 3.4.1.3 ми можемо почати, що всі тести пройшли, і тепер ми можемо бути впевнені якщо в майбутньому будуть якісь зміни в даній частині проекту, то ми відразу зможемо відловити помилку на самому ранньому етапі, і вона точно не потрапить до фінальних користувачів.

3.4.2 Інтеграційне тестування

Інтеграційне тестування додатку ми будемо проводити вручну за допомогою веб-інтерфейсу для виклику ендпоінтів – Swagger UI.

Swagger UI - це безкоштовний інструмент для автоматичної генерації документації та підтримки взаємодії з RESTful API який може бути легко інтегрований в різні проекти які використовують різноманітні фреймворки, в тому числі ASP.NET Core.

Він дозволяє створювати документацію та налаштовувати API на основі відкритого стандарту OpenAPI (раніше відомий як Swagger Specification), який описує структуру та поведінку API, наприклад визначення заголовків запитів, опису параметрів, встановлення обмежень на запити та інше .

Також одним з його найголовніших плюсів це генерація веб-інтерфейсу для перегляду документації, та взаємодії з API, що дозволяє розробникам легко переглядати всі доступні ендпоінти, їх параметри та можливі відповіді в різних ситуаціях.

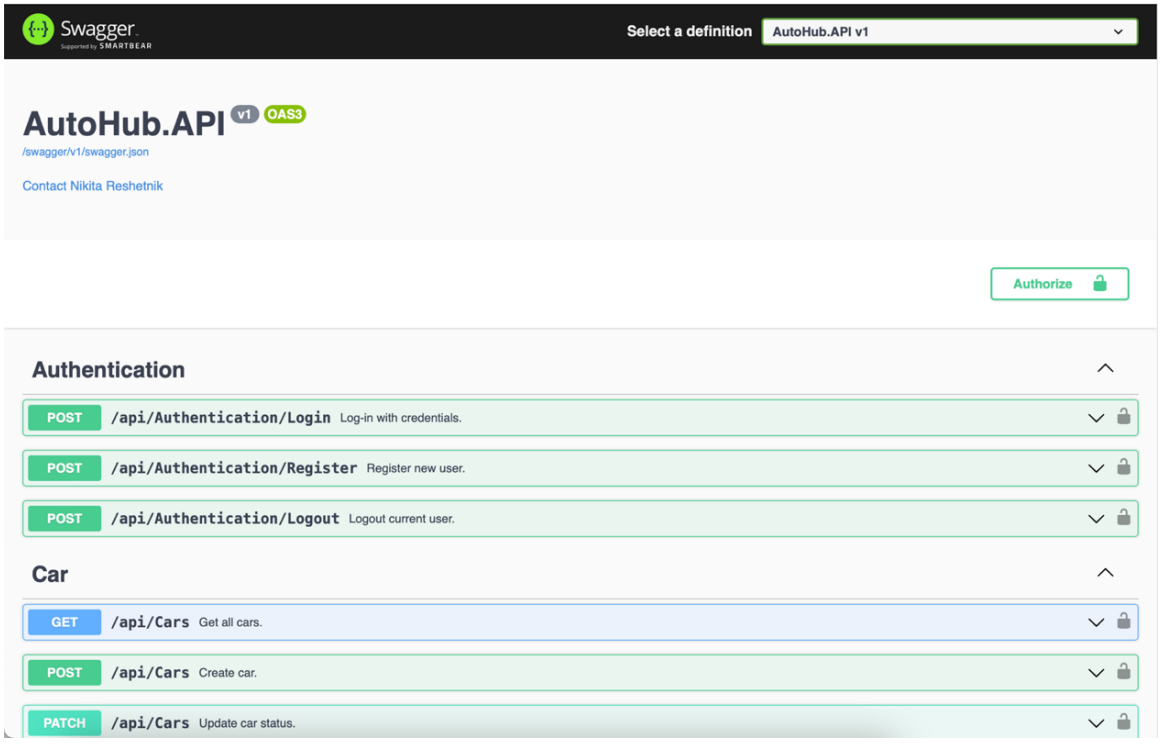


Рисунок 3.4.2.1 – Головна сторінка Swagger UI додатку

На рисунку 3.4.2.1 ми можемо побачити головну веб-сторінку нашого API додатку. На веб сторінці ми можемо побачити назву проекту, список доступних ендпоінтів розподілених по групам, кнопку авторизації та навіть посилання на контакти творця. Також Swagger UI підтримує версійність що дозволяє мати документацію для кожної версії додатку.

Більш того цей веб-інтерфейс дозволяє виконувати запити прямо з нього, що можна використати для тестування запитів API, включаючи всі RESTful методи - GET, POST, PUT, DELETE та інші. Це дозволяє легко протестувати API, без необхідності встановлювати додаткове програмне забезпечення, що значно спрощує процес тестування додатку.

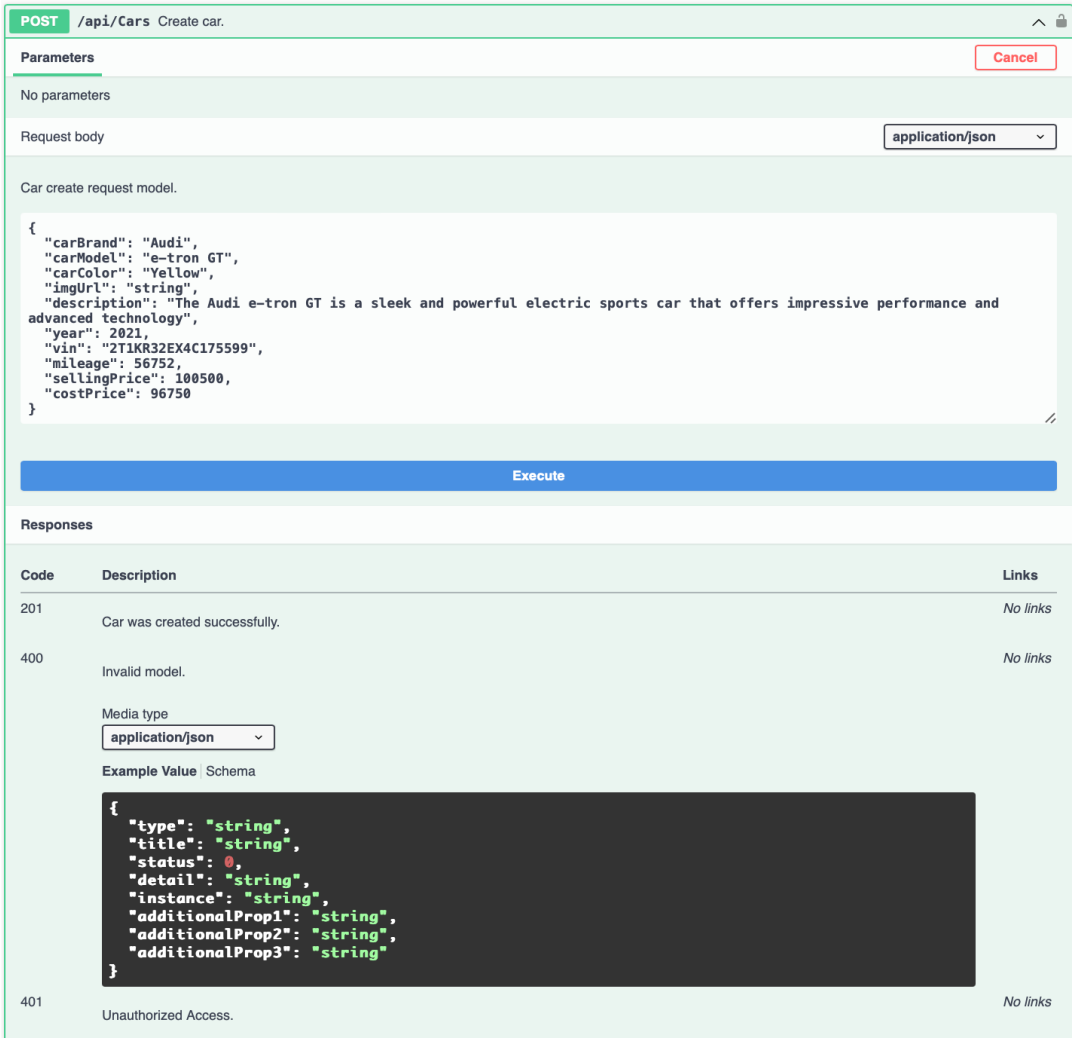


Рисунок 3.4.2.2 – Відкрита вкладинка запиту створення автомобіля

Також якщо відкрити одну із вкладинок (Рис. 3.4.2.2), ми побачимо інтерфейс для відправки запиту. А саме: поле для вводу тіла запиту, в основному це поле приймає JSON або XML формат об’єктів які перетворюються на об’єкти в коді, кнопка запуску запиту, і документація з описанням можливих відповідей та їх тіла та статус кодів. Також окрім тіла запиту, запит може потребувати інші типи параметрів, наприклад параметри URL як відображено на рисунку 3.4.2.3, або не потребувати їх взагалі.

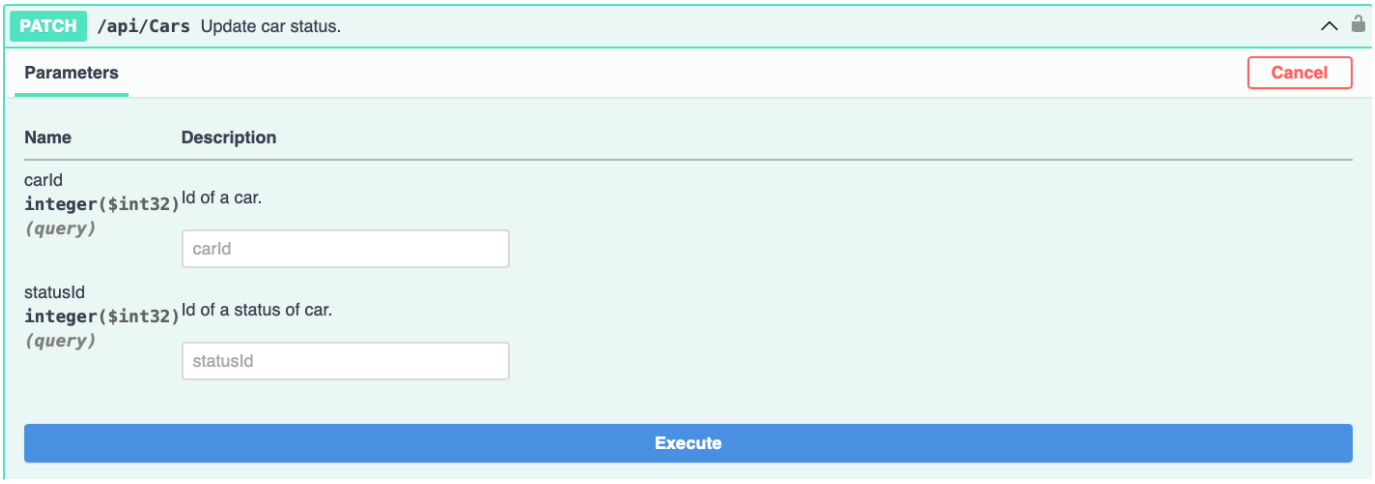


Рисунок 3.4.2.3 – Відкрита вкладинка запиту оновлення статусу автомобіля

Оскільки більшість дій ендпоінтів це CRUD (Create, Read, Update, Delete) операції, то мануальне тестування яке покриватиме більшість додатку можна спланувати наступною послідовністю дій:

1. Створення сутності за допомогою POST ендпоінту – перевірка що механізм створення сутності працює коректно.
2. GET запит на отримання сутності – перевірка що механізм отримання сутності працює коректно.
3. Оновлення сутності за допомогою UPDATE запиту – перевірка що механізм оновлення сутності працює коректно.
4. DELETE запит на видалення сутності – перевірка зо механізм видалення сутності працює коректно.

Для тестування виберемо сутність автомобіля оскільки ця сутність має велику кількість полів, але не має залежності від інших сутностей, що буде зручно для відображення прикладу тестування.

Для початку, оскільки додаток підтримує авторизацію користувачів, нам треба авторизуватись, для тесту ми будемо використовувати вже готового користувача-адміністратора який генерується автоматично, якщо аналогічно ще немає в базі даних. Для цього нам потрібно відправити POST запит на api/Authentication/Login ендпоінт, його приклад можна побачити на рисунку 3.4.2.4.

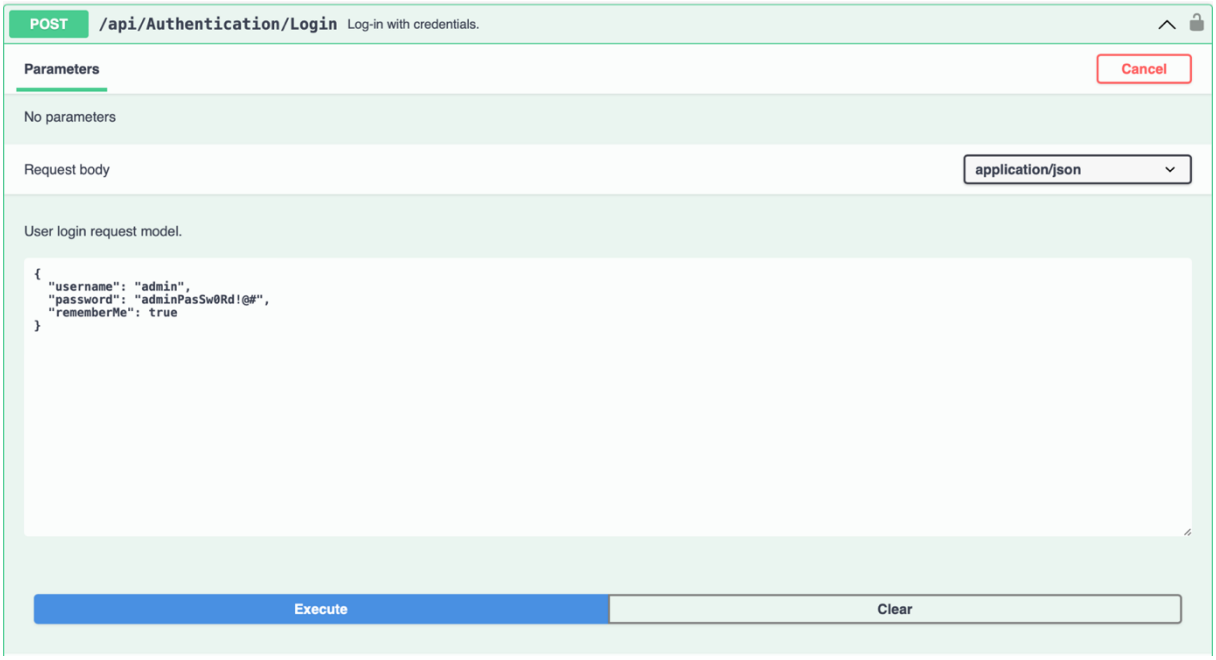


Рисунок 3.4.2.4 – Вкладинка запиту для авторизації

У відповідь ми отримуємо статус код 200 OK інформацію про користувача, та токен авторизації, якій знадобиться нам для доступу до більшості ендпоінтів (Рис. 3.4.2.5)

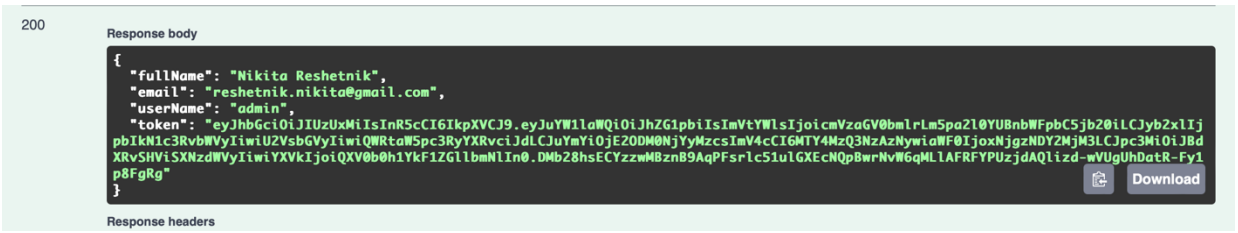


Рисунок 3.4.2.5 – Відповідь додатку на запит авторизації.

Токен який ми отримали це JSON Web Token (JWT) – стандарт відкритого ключа для безпечної передачі інформації який дозволяє зручну та безпечну авторизацію користувачів на сервері. Він складається з трьох частин: заголовок, тіло та підпис. Заголовок містить тип токена та алгоритм шифрування, тіло містить інформацію, яка може бути перевірена та декодована без секретного ключа, а підпис підтверджує, що токен був підписаний з секретним ключем сервера.

JWT токен використовуються для авторизації «Актора» в додатку та визначення рівня доступу для кожної окремої частини додатку. Коли «Актор» виконує вхід на сайт, сервер генерує JWT токен та повертає його до «Актора». Далі, кожен запит до сервера має включати JWT токен в заголовку запиту, щоб сервер міг перевірити, що запит приходить від авторизованого користувача.

Для перевірки токена сервер декодує його та перевіряє підпис. Якщо токен дійсний, сервер дозволяє доступ до запиту, якщо ні, сервер повертає помилку авторизації.

Використаємо цей токен для авторизації у додатку, натиснувши кнопку Authorize та вставивши токен (Рис. 3.4.2.6)

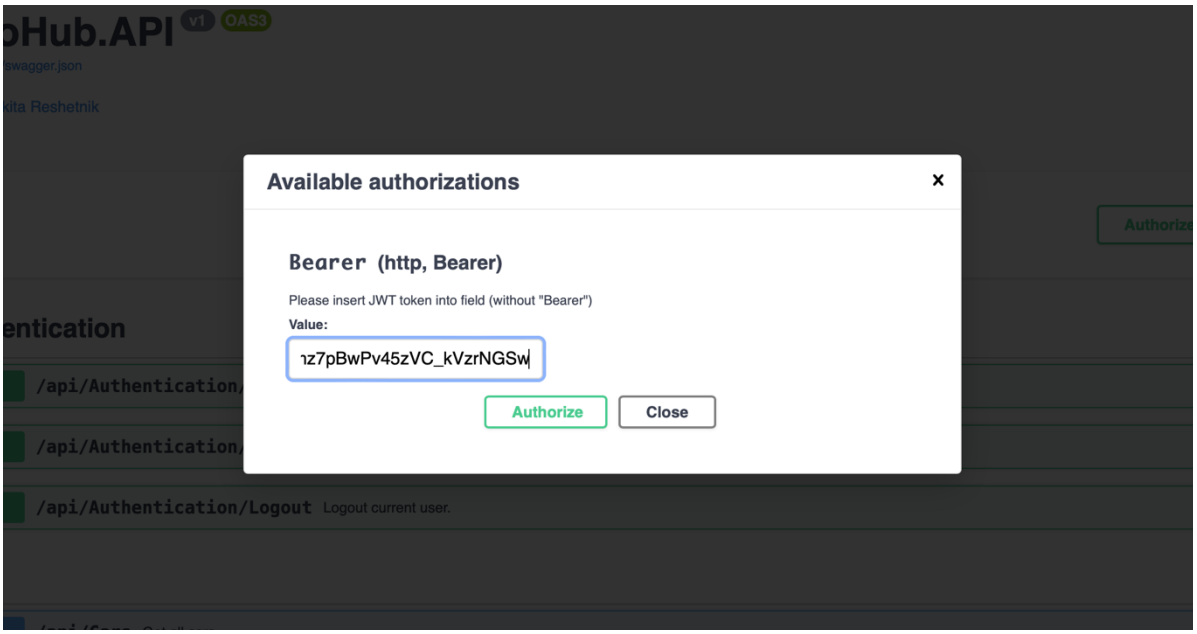


Рисунок 3.4.2.6 – Використання JWT токену

Тепер отримавши доступ до ендпоінтів ми можемо викликати будь-який з них, в нашому випадку це точки для взаємодії з автомобілями. Для початку перевіримо запит на створення нового автомобіля.

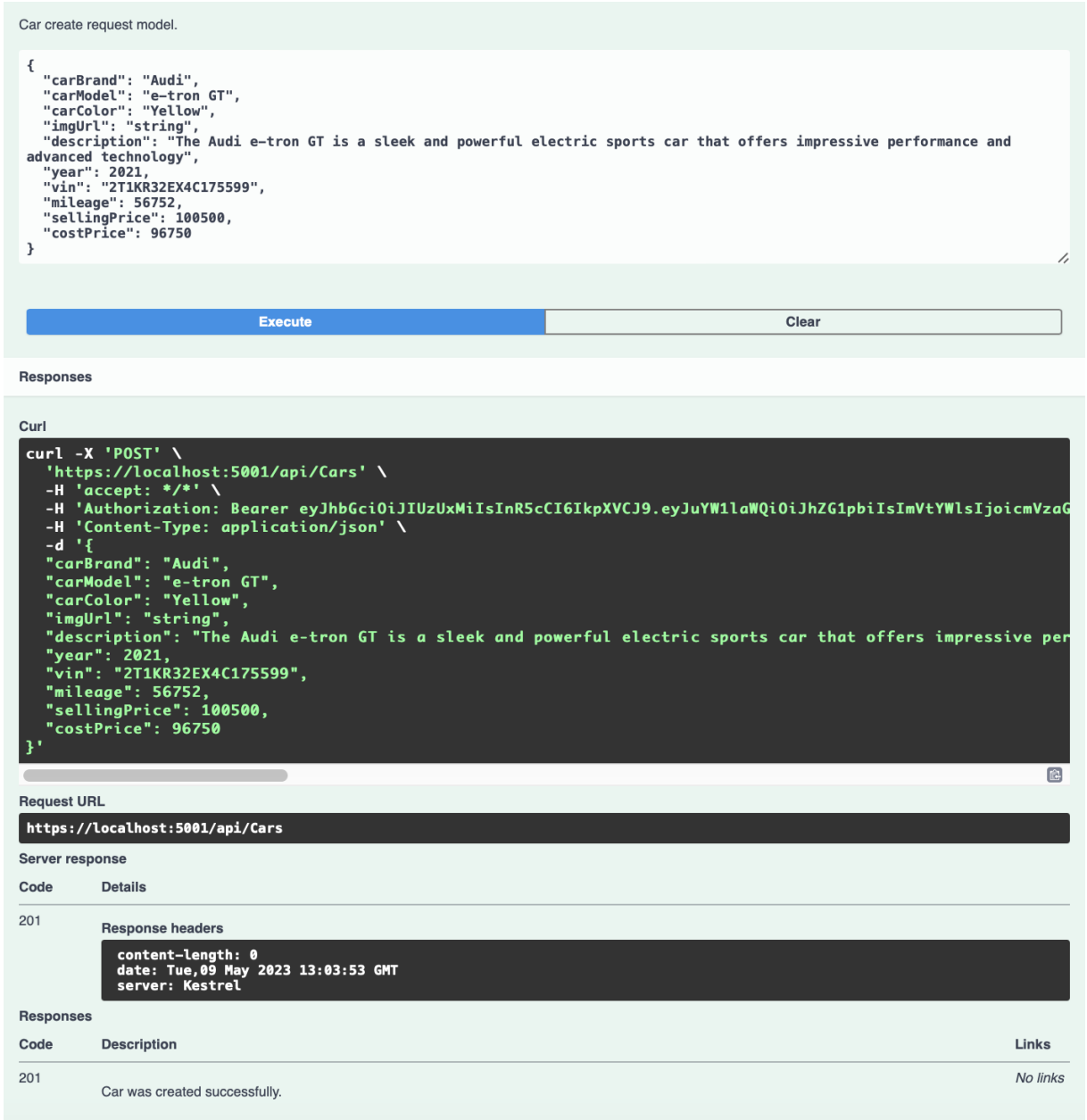


Рисунок 3.4.2.7 – Результат запиту створення автомобіля

На рисунку 3.4.2.7 можемо що наш додаток відповів на наш запит кодом 201, що каже про то що автомобіль був створений успішно, перевіримо це за допомогою запиту на отримання автомобілів.

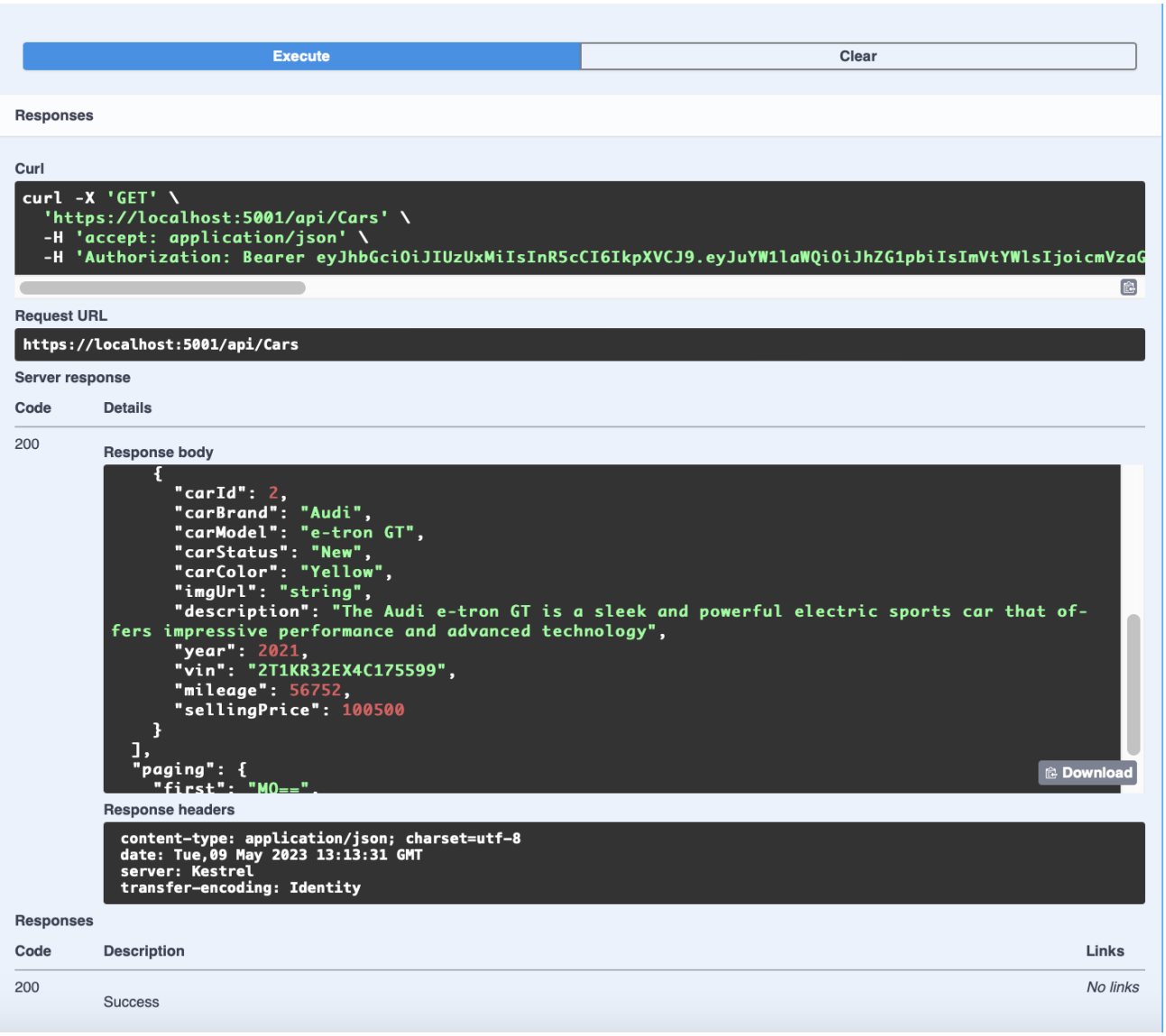


Рисунок 3.4.2.8 – Результат запиту на отримання автомобілів

На рисунку 3.4.2.8 можемо що наш додаток відповів на наш запит кодом 200, та відповіддю в форматі JSON з інформацією про щойно створений автомобіль, що підтверджує те що автомобіль був створений успішно.

Далі на черзі запит з оновлення інформації про автомобіль, для цього нам треба взяти унікальний ідентифікатор автомобіля що ми тільки що створили, в нашому випадку це 2, вставити це значення в поле CarId. І тепер вже ми можемо створити тіло запито на оновлення інформації. Спробуємо поміняти всі поля щоб цей автомобіль не був схожий на минули

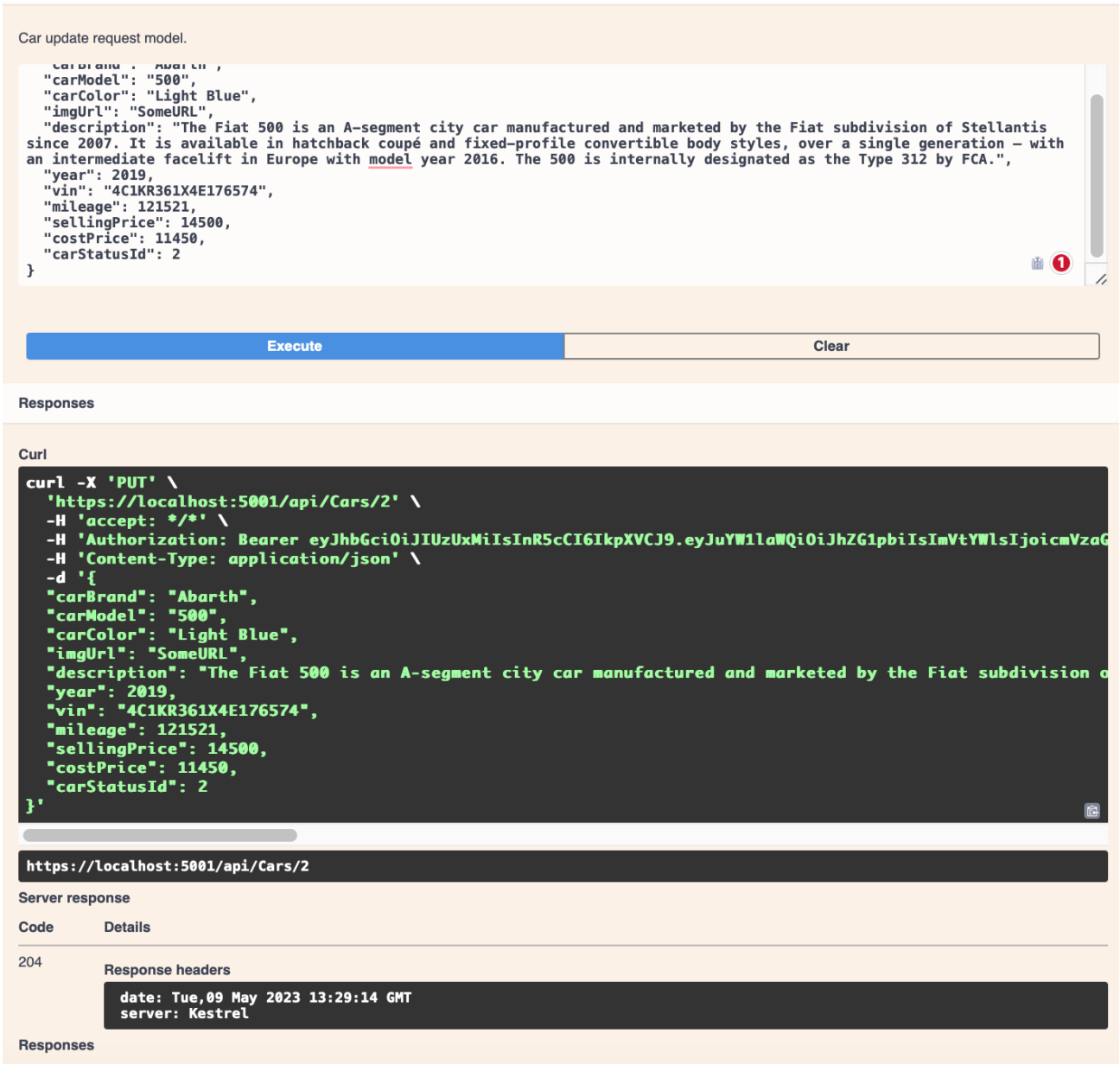


Рисунок 3.4.2.9 – Результат запиту на оновлення автомобіля

На рисунку 3.4.2.9 можемо що наш додаток відповів на наш запит кодом 204 що означає що операція пройшла успішно, і додатку немає тіла щоб нам повернути. Перевіримо це за допомогою запиту на отримання автомобіля але який повертає один автомобіль по його унікальному ідентифікатору, оскільки ми його вже знаємо.

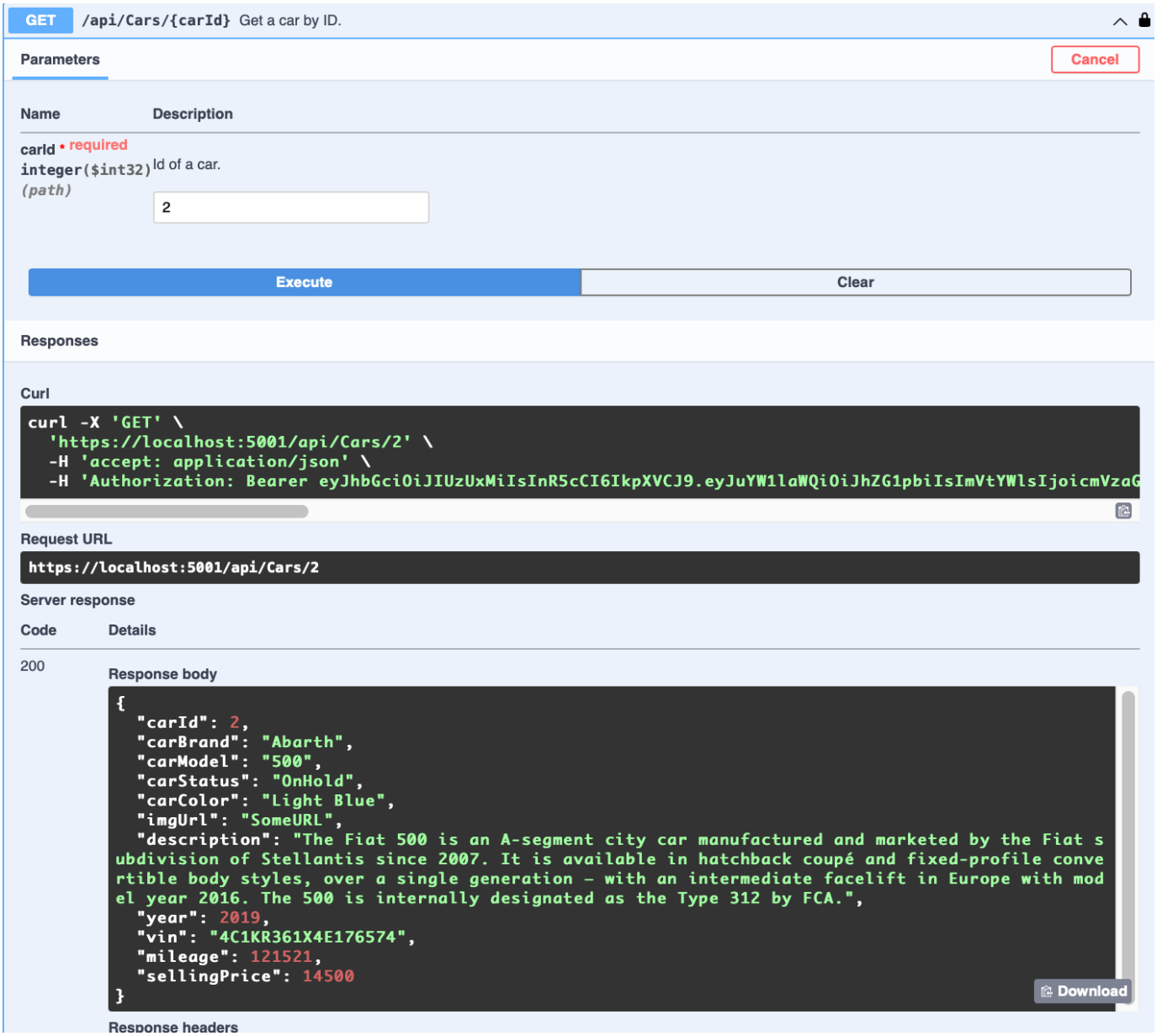


Рисунок 3.4.2.10 – Результат виконання запиту на отримання автомобіль по його унікальному модифікатору

На рисунку 3.4.2.10 можемо що наш додаток відповів на наш запит кодом 200, та відповіддю в форматі JSON з інформацією про автомобіль, але вже з оновленою інформацією.

І останній на черзі ендпоінт який видаляє сутність. З параметрів які він потребує це тільки унікальний ідентифікатор.

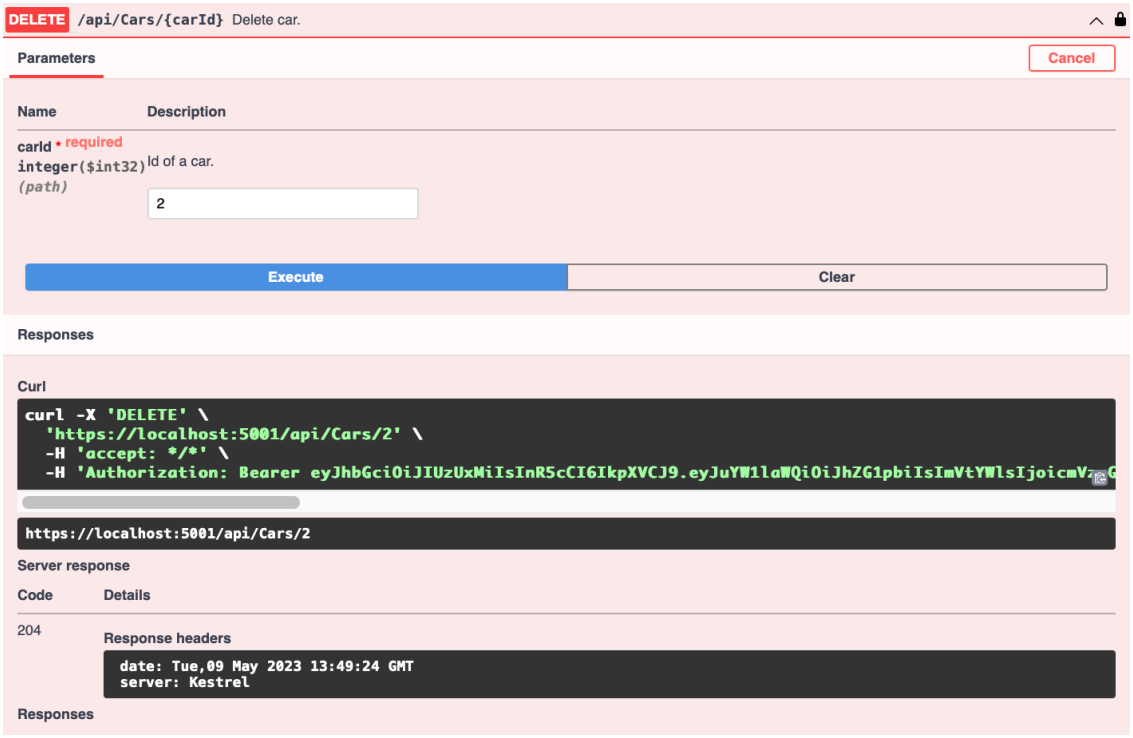


Рисунок 3.4.2.11 – Результат запиту на видалення автомобіля

Як і у відповіді на запит оновлення інформації, тут ми теж отримуємо у відповідь код 204, що означає що автомобіль було видалено. Перевіримо знову запитом на отримання інформації про автомобіль.

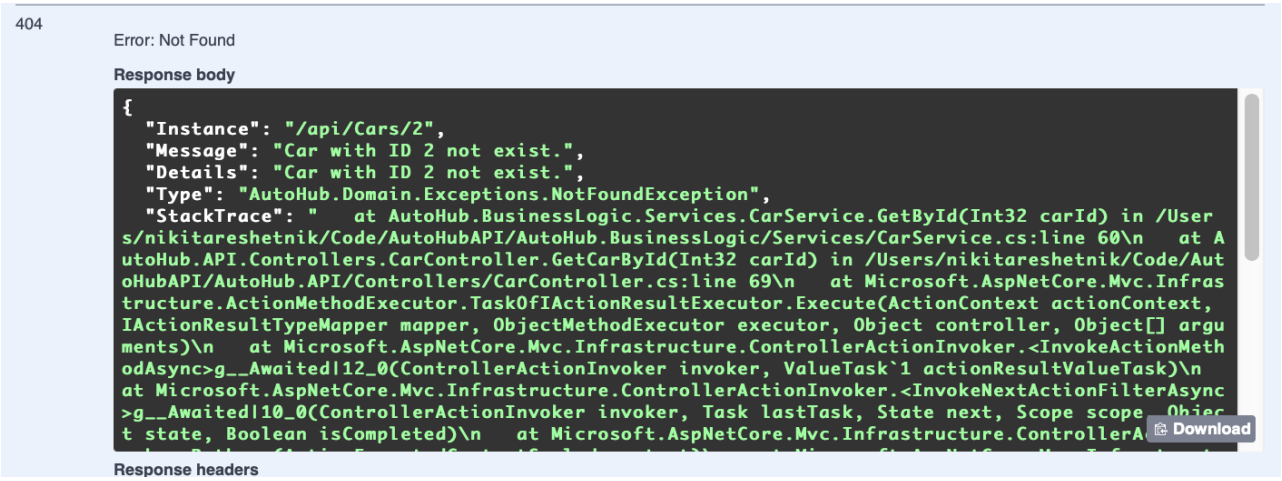


Рисунок 3.4.2.11 – Результат виконання запиту на отримання автомобіль по його унікальному модифікатору після видалення

Тут вже додаток повертає нам у відповідь повідомлення, що автомобіля за унікальним модифікатором 2 не існує, що підтверджує працездатність ендпоінта на видалення.

Теж саме було зроблено для всіх інших ендпоінтів додатку, і результат тестування був позитивним. Окрім того було протестовано правила валідації даних що приходять у запиті, таких як наприклад: VIN номер має бути довжиною 17 символів, рік автомобіля маж бути не нижчим ніж 1900 рік, або на обов’язкову наявність деяких з полів.

Завершити тестування ми можемо тестуванням одної з головної частин додатку, а саме автоматична ідентифікація переможця лоту коли час лоту завершений. Для цього було створено новий лот з датою завершення через 5 хвилин, та створено декілька ставок на цей лот будь-якими користувачами, та з плином 5 хвилин було виявлено що статус лота став «Завершений» та у полі «Переможець» в вказаний користувач з найбільшою ставкою зробленою на лот.

В вигляді доповнення тестування саме коду, на етапі фіналу розробки було проведено статичне тестування (статичний аналіз) додатку що допомогло виявити:

– змінні, що не використовуються;
– код, що не використовуються;
– оптимізація читабельності коду за рахунок використання інших конструкцій;
– змінні з невизначеними значеннями;
– помилки правопису та визначення більш доречного імені змінної

4. ВИСНОВКИ

1. Проведено аналіз предметної галузі.
2. Проведено аналіз існуючих додатків, які існують в сфері онлайн-аукціонів. Визначено їх призначення, переваги та недоліки. Серед ключових недоліків українських додатків визначено широка спеціалізація додатків що призводить до меншої кількості функціоналу яка може використана саме в області автомобільних аукціонів. Серед іноземних додатків основний недолік це часткова або повна відсутність на українському ринку.
3. Виконано проектування додатку, розроблено діаграму використання, діаграму бази даних, діаграму пакетів.
4. Розроблено програмний додаток мовою C# за допомогою платформи .NET та фреймворків ASP.NET, Entity Framework.
5. Перспективи подальших досліджень:
   5.1. Функціонал отримання статистики
   5.2. Розробка власного додатку-клієнту
   5.3. Інтеграція OAuth2 для пришвидшеної аутентифікації за допомогою акаунту Google

5. СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. ASP.NET Microsoft Docs [Електронний ресурс] – Режим доступу до ресурсу: https://docs.microsoft.com/ru-RU/aspnet/core/?view=aspnetcore-6.0.
2. Metanit [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/.
3. Vertical Slice Architecture in ASP.NET Core [Електронний ресурс] // Code Maze. – 2023. – Режим доступу до ресурсу: https://code-maze.com/vertical-slice-architecture-aspnet-core.
4. Schou C. How to build a RESTful Web API using ASP.NET Core and Entity Framework Core (.NET 6) [Електронний ресурс] / Christian Schou. – 2022. – Режим доступу до ресурсу: https://blog.christian-schou.dk/how-to-build-a-restful-web-api-using-net-core6/.
5. Schou C. How to Send Emails with ASP.NET Core using SMTP with MailKit [Електронний ресурс] / Christian Schou. – 2022. – Режим доступу до ресурсу: https://blog.christian-schou.dk/send-emails-with-asp-net-core-with-mailkit/.
6. Документація Fluent Validation [Електронний ресурс] – Режим доступу до ресурсу: https://docs.fluentvalidation.net/en/latest/.
7. Singh S. Implement JWT Authentication in ASP.NET Core 5 Web API [Token Base Auth] [Електронний ресурс] / Satinder Singh // codepedia.info. – 2023. – Режим доступу до ресурсу: https://codepedia.info/jwt-authentication-in-aspnet-core-web-api-token.
8. Murugan P. Installing and Using Swagger in C# 11 with Code Samples [Електронний ресурс] / Pandiyan Murugan // efficientuser.com. – 2023. – Режим доступу до ресурсу: https://efficientuser.com/2023/01/13/installing-and-using-swagger-in-c-11-with-code-samples/.
9. Schedule Jobs with Quartz.NET [Електронний ресурс] // Code Maze. – 2023. – Режим доступу до ресурсу: https://code-maze.com/schedule-jobs-with-quartz-net/.

10. How to design an effective relational database [Електронний ресурс]. – 2022. – Режим доступу до ресурсу: https://blog.airtable.com/how-to-design-an-effective-relational-database/.
11. Ramos C. Schema design 101 for relational databases [Електронний ресурс] / Camila Ramos. – 2022. – Режим доступу до ресурсу: https://planetscale.com/blog/schema-design-101-relational-databases.
12. Chiarelli A. Using xUnit to Test your C# Code [Електронний ресурс] / Andrea Chiarelli // auth0 Blog. – 2021. – Режим доступу до ресурсу: https://auth0.com/blog/xunit-to-test-csharp-code/.
13. Moq Documentation [Електронний ресурс] // Github. – 2022. – Режим доступу до ресурсу: https://github.com/Moq/moq4/wiki/Quickstart.
14. Autofixture Documentation [Електронний ресурс] – Режим доступу до ресурсу: https://autofixture.github.io/docs/quick-start/.
15. Patel A. Tutorial: ASP.NET Core with Identity [Електронний ресурс] / Alpesh Patel // Medium. – 2021. – Режим доступу до ресурсу: https://medium.com/c-sharp-progarmming/tutorial-asp-net-core-with-identity-299d0fe4cba1.
16. Забезпечення надійності та підтримуваності серверного додатку за допомогою багаторівневої архітектури / Решетнік Н.О., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Всеукраїнська науково-технічна конференція. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 29.
17. Використання технології Entity Framework в розробці баз даних на платформі .NET / Решетнік Н.О., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Всеукраїнська науково-технічна конференція. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 150.

6. ДОДАТОК А

ДЕРЖАВНИЙ УНІВЕРСИТЕТ ТЕЛЕКОМУНІКАЦІЙ

НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Розробка серверного додатку для організації роботи автомобільного аукціону з використанням технології ASP.NET Web API, мовою C#

Виконав студент 4 курсу
Групи ПД-41
Решетнік Нікіта Олександрович
Керівник роботи
К.т.н, доц, доцент кафедри ІПЗ Трінтіна Наталія Альбертівна

Київ – 2023

МЕТА, ОБ’ЄКТ ТА ПРЕДМЕТ ДОСЛІДЖЕННЯ

- **Мета роботи** - спрощення процесу цифровізації сфери автомобільних аукціонів переходу до цифрової форми проведення торгів, що дозволить їм ефективніше взаємодіяти зі своїми клієнтами та знизити витрати на проведення аукціонів.

- **Об’єкт дослідження** - розробка серверного додатку для автоматизації проведення автомобільних аукціонів.

- **Предмет дослідження** - додаток для автоматизації проведення автомобільних аукціонів.

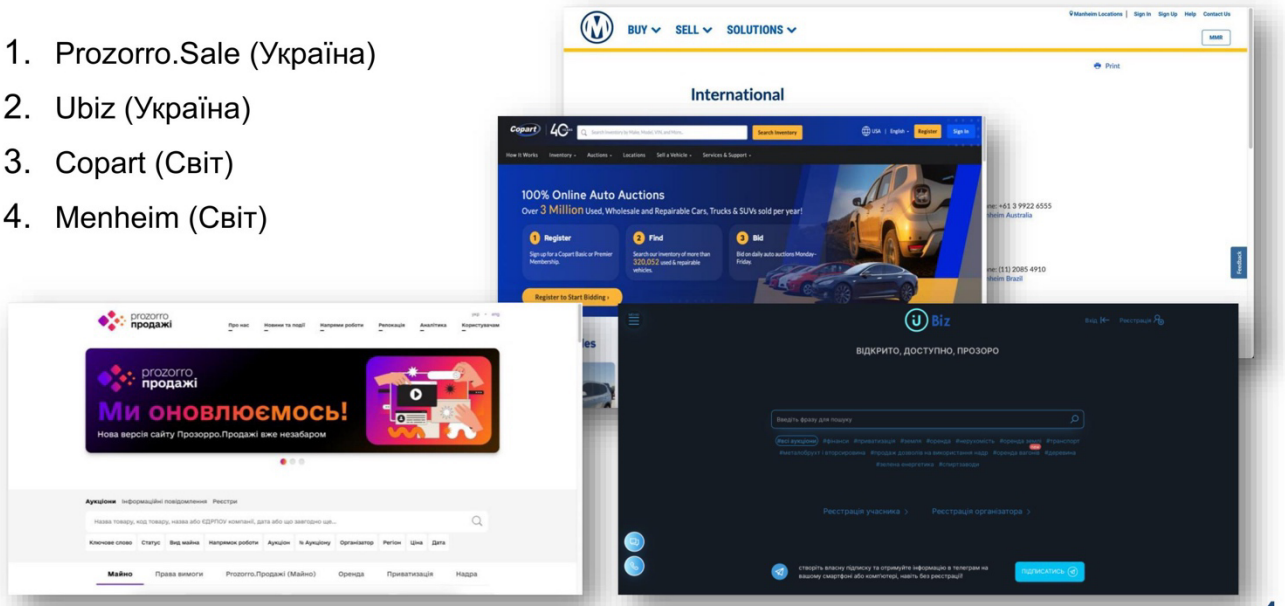
ЗАДАЧІ ДИПЛОМНОЇ РОБОТИ

- Проаналізувати вимоги, потреби, технічні можливості потенційної аудиторії даного програмного продукту, переваги та недоліки існуючих рішень
- Провести аналіз інструментів та програмних засобів реалізації, що покажуть найкращу продуктивність системи, під час застосування продукту в реальних умовах, та що надають найбільшу зручність під час написання продукту
- Спроектувати та розробити серверний додаток для організації проведення онлайн аукціонів
- Провести End-To-End тестування додатку на предмет недоліків, помилок та не валідної поведінки.

3

АНАЛІЗ АНАЛОГІВ

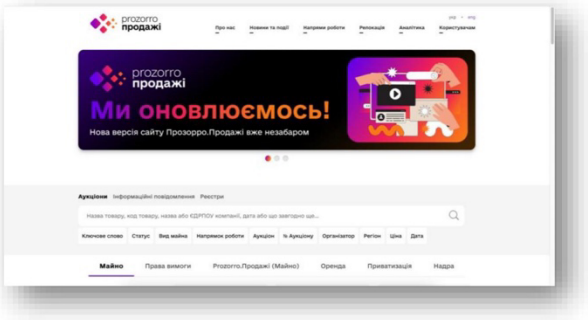
1. Prozorro.Sale (Україна)
2. Ubiz (Україна)
3. Copart (Світ)
4. Menheim (Світ)



4

АНАЛОГ: **Prozorro.sale**

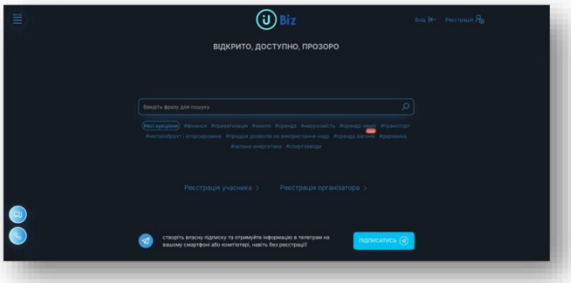
Prozorro.Sale - це державна електронна платформа, що пропонує механізми продажу різних державних активів, зокрема автомобілів. Ця система розроблена для забезпечення прозорості, конкуренції та ефективності у процесі продажу майна державних органів та підприємств. Крім того, Prozorro.Sale дозволяє здійснювати електронну реєстрацію на аукціони та контролювати весь процес торгів від початку до кінця. Це дозволяє максимально оптимізувати і спростити процес продажу майна та забезпечує повну відкритість та чесність у проведенні торгів.



5

АНАЛОГ: **Ubiz.ua**

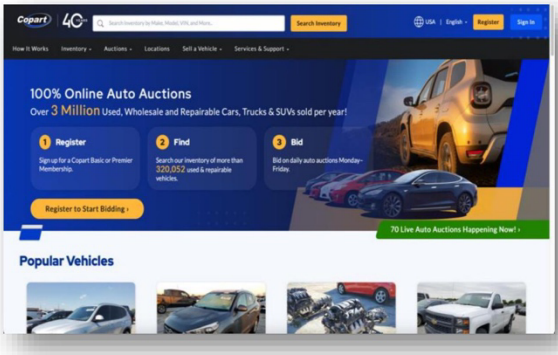
UBiz - це онлайн-платформа та учасник платформи Prozorro для проведення електронних торгів з продажу різноманітного майна, включаючи автомобілі, нерухомість, обладнання та інше. Сервіс дає можливість проводити електронні торги в режимі реального часу, що забезпечує прозорість та ефективність процесу. Продавці можуть додати свої лоти на платформу, а покупці можуть знайти багато пропозицій для покупки майна різного типу та цін. В той же час, UBiz надає розумні інструменти для зручної та швидкої покупки майна, що забезпечує зручність та ефективність процесу.



6

АНАЛОГ: **Copart**

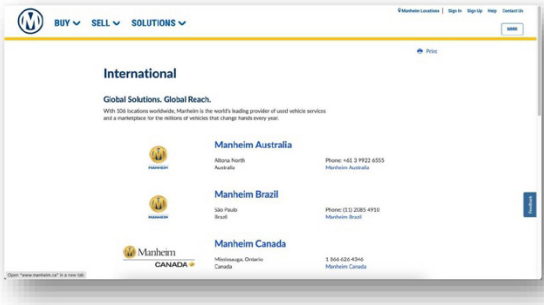
Copart – це один з найбільших в світі інтернет-аукціонів підтриманих автомобілів. Copart забезпечує доступ до міжнародної бази даних з продажу автомобілів, що дає можливість придбати транспорт за конкурентною ціною зі всього світу.
Компанія має розвинуту мережу філій по всьому світу, що дозволяє їм оперативно проводити операції з продажу автомобілів в різних країнах. Крім того, Copart пропонує можливість зареєструватися як дилер та отримувати спеціальні пропозиції та знижки на придбання автомобілів на аукціоні.



7

АНАЛОГ: **Menheim**

Menheim – найбільший в світі аукціон підтриманих автомобілів, він надає послуги з продажу транспортних засобів на аукціонах в США, Канаді, Великобританії та Європі, включаючи інтернет-аукціони та аукціони у форматі "живого" присутності.
Крім того, Manheim дозволяє своїм клієнтам здійснювати продажі на власних майданчиках, а також використовувати платформу для продажу транспортних засобів у віртуальному режимі.



8

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Базовий функціонал серверного додатку

- Функціонал реєстрації та авторизації користувача
- Функціонал підтвердження реєстрації
- Функціонал пагінації відповідей сервісу
- Слідувати протоколам та правилам REST API
- Обмежувати доступ за типом користувача
- Валідація запитів зі сторони додатків-клієнтів

Функціонал аукціону зі сторони користувача

- Створення лотів
- Створення ставок на існуючі лоти
- Додавання автомобілів та даних про них в систему
- Автоматизовану детермінацію переможця лоту

ВИМОГИ ДО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

Функціонал аукціону зі сторони адміністратора

- Додавання/Отримання/Оновлення/Видалення інформації про автомобільні бренди, моделі та колір
- Додавання/Отримання/Оновлення/Видалення інформації про лоти та ставки
- Додавання/Отримання/Оновлення/Видалення інформації про користувачів

Функціонал зберігання даних та доступу до них

- СУБД для зберігання всіх даних
- Надійне зберігання персональних даних користувачів
- Шифрування паролів
- Структура БД має бути достатньо нормалізована там мати розширювану структуру.

ПРОГРАМНІ ЗАСОБИ РЕАЛІЗАЦІЇ

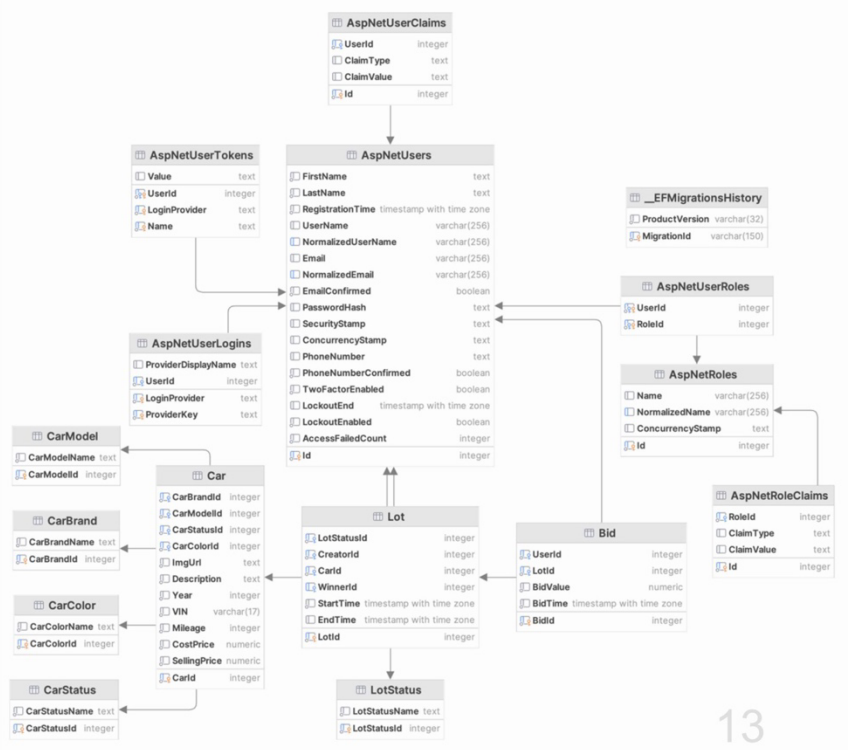
- **Середовище розробки:** JetBrains Rider
- **Мова програмування:** C# 10
- **Платформа розробки:** .NET 7
- **Фреймворк для побудування веб-додатків:** ASP.NET
- **Фреймворк для контролю над авторизацією:** ASP.NET Identity
- **Фреймворк Об’єктно-реляційної проекції (ORM):** Entity Framework
- **Система керування базами даних:** PostgreSQL
- **Фреймворк для тестування:** xUnit

11

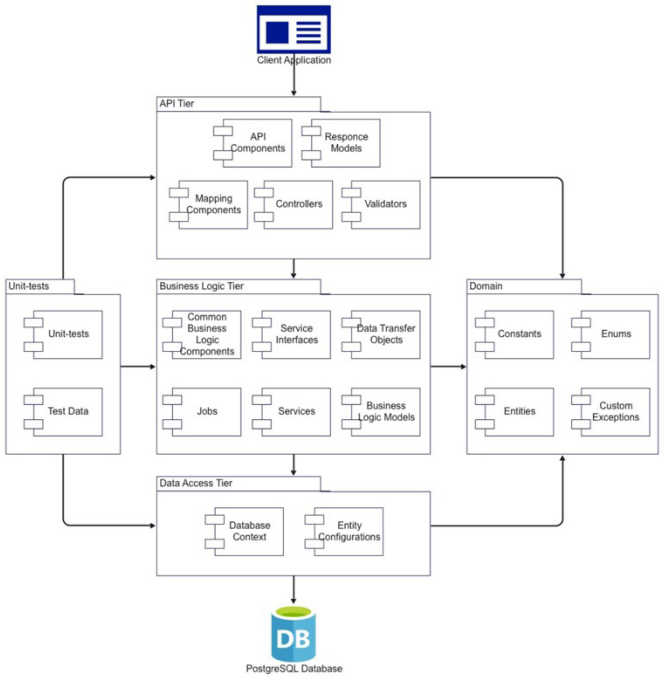
Діаграма варіантів використання

12

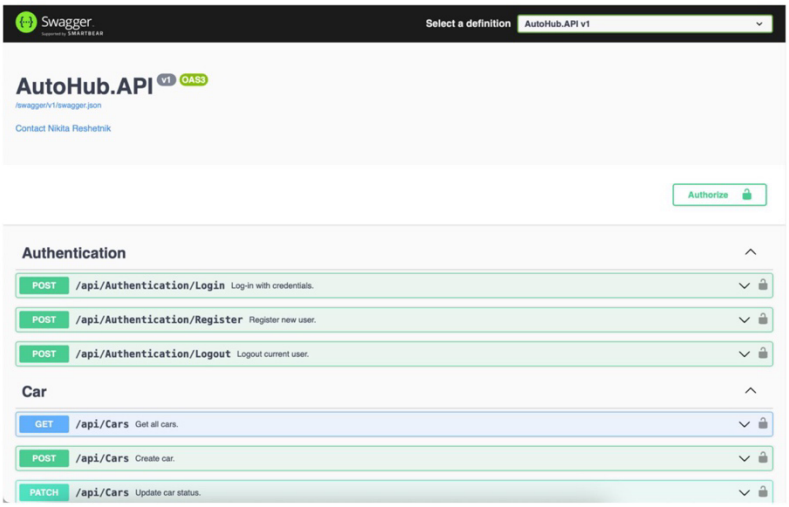
Діаграма бази даних



Діаграма пакетів



ЕКРАННІ ФОРМИ



Головна сторінка Swagger UI

15

АПРОБАЦІЯ РЕЗУЛЬТАТІВ ДОСЛІДЖЕННЯ

1. Забезпечення надійності та підтримуваності серверного додатку за допомогою багаторівневої архітектури / Решетнік Н.О., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Всеукраїнська науково-технічна конференція. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 29.
2. Використання технології Entity Framework в розробці баз даних на платформі .NET / Решетнік Н.О., Трінтіна Н.А. // Застосування програмного забезпечення в інфокомунікаційних технологіях: Всеукраїнська науково-технічна конференція. Збірник тез. 20.04.2023, ДУТ, м. Київ — К.: ДУТ, 2023. — С. 150.

16

ВИСНОВКИ

- Проведено аналіз предметної галузі.
- Проведено аналіз існуючих додатків, які існують в сфері онлайн-аукціонів. Визначено їх призначення, переваги та недоліки.
- Виконано проектування додатку, розроблено діаграму використання, діаграму бази даних, діаграму пакетів.
- Розроблено програмний додаток мовою C# за допомогою платформи .NET та фреймворків ASP.NET, Entity Framework.
- Пройдено апробацію роботи на конференції «Застосування Програмного Забезпечення В Інфокомунікаційних Технологіях».

17

ДЯКУЮ ЗА УВАГУ!