

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування інституту, факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

бакалавра

(ступінь вищої освіти)

на тему ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ БРОНЮВАННЯ
ПАРКУВАЛЬНИХ МІСЦЬ
PARKING LOT BOOKING SOFTWARE

Виконав: студент(ка) 4 курсу, групи КНТ-138
Спеціальності 121 Інженерія програмного
забезпечення

(код і найменування спеціальності)

Освітня програма (спеціалізація)
Інженерія програмного забезпечення

Мороз В.В.

(прізвище та ініціали)

Керівник Льовкін В.М.

(прізвище та ініціали)

Рецензент Скрупський С.Ю.

(прізвище та ініціали)

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»
(повне найменування закладу вищої освіти)

Інститут, факультет ФКНТ
Кафедра програмних засобів
Ступінь вищої освіти бакалавр
Спеціальність 121 Інженерія програмного забезпечення
(код і найменування)
Освітня програма (спеціалізація) Інженерія програмного забезпечення
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
С.О. Субботін
“ ” 2022 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

Мороза Віталія Вікторовича

(прізвище, ім'я, по батькові)

1. Тема проєкту (роботи) Програмне забезпечення бронювання паркувальних місць

Parking Lot Booking Software

керівник проєкту (роботи) Льовкін Валерій Миколайович, к.т.н., доцент,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом закладу вищої освіти від «27» квітня 2022 року № 102

2. Строк подання студентом проєкту (роботи) червень 2022 року

3. Вихідні дані до проєкту (роботи) рекомендована література, технічне завдання

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз предметної області. 2. Проєктування програмного забезпечення. 3. Розробка програми. 4. Експлуатація та тестування програми.

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

6. Консультанти розділів проєкту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-4 Основна частина	Льовкін В.М., доцент		
Нормоконтролер	Камінська Ж.К., асистент		

7. Дата видачі завдання « 4 » квітня 2022 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Постановка завдання роботи	1 тиждень	Завдання, ТЗ
2	Аналіз предметної області	2 тиждень	Розділ 1
3	Проектування програми	3 тиждень	Розділ 2
4	Розробка програми	4-5 тижні	Розділ 3
5	Тестування та експериментальне дослідження програми	6 тиждень	Розділ 4
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	7 тиждень	Додатки
7	Захист роботи	8 тиждень	

Студент(ка)

(підпис)

Мороз В.В.

(прізвище та ініціали)

Керівник проєкту (роботи)

(підпис)

Льовкін В.М.

(прізвище та ініціали)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи бакалавра: 91 с., 26 рис., 3 табл., 3 дод., 13 джерел.

ПАРКУВАННЯ, АВТОМОБІЛЬ, ПАРКУВАЛЬНЕ МІСЦЕ, ВЕБЗАСТОСУНОК, БАГАТОШАРОВА АРХІТЕКТУРА.

Об'єкт роботи – процеси пошуку і бронювання паркувальних місць. Предмет роботи – програмне забезпечення бронювання паркувальних місць. Мета роботи – розробити програмне забезпечення бронювання паркувальних місць для гарантування розміщення власного автомобіля в умовах обмеженості наявних місць.

Матеріали, методи та технічні засоби: мова програмування C#, фреймворки ASP.NET Core і Entity Framework Core, система керування базами даних Microsoft SQL Server.

Результати. Результатом роботи є розроблена програма, побудована на основі використання багатошарової архітектури, що сприяє її подальшій підтримці з можливістю інтеграції пристроїв відслідковування автомобілів на паркувальних місцях, що додатково розширить застосування програми.

Висновки. Розроблена програма призначена для організації пошуку паркувальних майданчиків у місті, для бронювання паркувальних місць на цих майданчиках з приділенням особливої уваги приватним індивідуальним паркувальним місцям і забезпеченням інструментів внесення таких паркувальних місць їх власниками до програми та для надання інструментів для потенційного широкого доступу до засобів бронювання для багатомісних паркувальних майданчиків.

Галузь використання. Пошук паркувальних місць у місті, підтримка бронювання паркувальних місць для власників індивідуальних місць та потенційно для багатомісних паркувальних комплексів.

ABSTRACT

Explanatory note to the diploma qualifying work of the bachelor: 91 pages, 26 figures, 3 tables, 3 appendixes, 13 sources.

PARKING, CAR, PARKING LOT, WEB APPLICATION, MULTILAYER ARCHITECTURE.

The object of work is the process of finding and booking parking spaces. Subject of work is software for booking parking spaces. The purpose of the work is to develop software for booking parking spaces to guarantee the placement of own car in the conditions of limited available space.

Materials, methods and technical means: C# programming language, ASP.NET Core and Entity Framework Core frameworks, Microsoft SQL Server database management system.

Results. The result is a developed software based on the usage of multi-layered architecture, which contributes to its further support with the possibility of integrating car tracking devices in parking lots, which will further expand the application of the software.

Conclusions. The developed software is designed to organize the search for parking spaces in the city, to book parking spaces on these sites with special attention to private individual parking spaces and to provide tools to create such parking spaces in the software by their owners and to provide tools for potentially wide access to multi-seat parking spaces.

Field of use. Search for parking spaces in the city, support the reservation of parking spaces for owners of individual spaces and potentially for multi-seat parking complexes.

ЗМІСТ

	С.
Перелік скорочень та умовних познач	7
Вступ	8
1 Аналіз предметної області	10
1.1 Особливості паркувальних місць у сучасному місті	10
1.2 Концепція програми бронювання паркувальних місць	13
1.3 Аналіз існуючих програм пошуку і бронювання паркувальних місць ..	15
1.4 Висновки за розділом 1	17
2 Проєктування програмного забезпечення	18
2.1 Вибір мови програмування	18
2.2 Визначення функціональних вимог до програми	19
2.3 Проєктування бази даних	20
2.4 Вибір архітектури програми	27
2.5 Висновки за розділом 2	28
3 Розробка програми	29
3.1 Алгоритм функціонування програми	29
3.2 Опис реалізованих класів	31
3.3 Висновки за розділом 3	42
4 Експлуатація та тестування програми	43
4.1 Призначення програми	43
4.2 Умови виконання програми	43
4.3 Експлуатація програми	43
4.4 Тестування програми	49
4.5 Висновки за розділом 4	50
Висновки	51
Перелік джерел посилання	53
Додаток А Технічне завдання	55
Додаток Б Текст програми	59
Додаток В Слайди презентації	84

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

MPV – multi-purpose vehicles;

RV – recreational vehicle;

SUV – sport utility vehicle.

ВСТУП

Проблема паркування в великих містах і містах середнього розміру знайома кожному власнику автомобіля, який жив або перебував у такому місті.

Загалом слід вказати, що сучасні міста у всьому світі мають великі проблеми з паркуванням, що пов'язана з одного боку з надто великою кількістю паркувальних місць з боку використання простору самого міста та з іншого боку, з боку власників автомобілів, з нестачею паркувальних місць для зручного паркування в будь-якій точці міста в будь-який момент. У сучасному місті слід констатувати, що достатньої кількості паркувальних місць не буває і бути не може.

З моменту винаходу автомобіля паркування або, іноді, прогресивні обмеження паркування сильно сформували дизайн, навколишнє середовище та економіку міських районів:

- деякі аналітики кажуть, що Західна Європа містить 300 мільйонів місць, тоді як, за оцінками, Сполучені Штати можуть похвалитися 2 мільярдами, при чому часто ця нерухомість розташована в центрі і дуже цінна, одне стандартне паркувальне місце, розміром трохи більше шести на три метри, займає приблизно стільки ж місця, скільки невелика паризька квартира-студія, квартира для малозабезпечених в Індії або три офісні кабінети;

- паркувальні простори уздовж вулиць у міських кварталах складають поверх на поверх у гаражах і розкинулися навколо офісів і торгових центрів, а найбільша в світі автостоянка в West Edmonton Mall в Альберті, Канада, має 20 000 паркувальних місць, що охоплює достатньо місця для 500 будинків;

- дослідження, проведене Центром технологій сусідства, некомерційною організацією сталого розвитку в Чикаго, показало, що від чверті до третини паркувальних місць навколо багатоквартирних будинків у багатьох містах США пустують;

– у міру зростання світового рівня життя та прискорення урбанізації, особливо в Індії та Китаї, міста по всьому світу спостерігають величезні сплески власників автомобілів, які супроводжуються попитом на паркування: в Індії кількість приватних автомобілів зросла майже на 400 % між 2001 і 2015 роками, з 55 мільйонів до 210 мільйонів, а за даними центрального уряду, у Китаї станом на 2017 рік не вистачало 50 мільйонів паркувальних місць [1].

Але актуальна проблема паркувань у сучасних містах має вираження в результаті в необхідності пошуку потрібних паркувальних місць у місті. Оскільки в різних містах таких місць може не вистачати, то може виникати проблема не тільки в отриманні інформації про доступні паркувальні майданчики, але й бронюванні цих місць. Зважаючи на це, в роботі потрібно розробити програмне забезпечення бронювання паркувальних місць.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Особливості паркувальних місць у сучасному місті

Сучасні паркувальні місця можуть мати багато додаткових можливостей, які є особливостями сучасного світу. До таких особливостей зокрема належить наявність електричної зарядки. Але є і стандартні параметри – розміри такого місця. Багато в чому вони є важливими характеристиками для власника автомобіля. Тому в даному підрозділі розглянуто варіанти розмірів сучасних автомобілів, що дозволить характеризувати в подальшому відповідним чином паркувальні місця.

Європейська класифікація автомобілів за їх розміром:

- клас А (автомобілі міні-класу) – це автомобілі довжиною менше 370 см (міні-автотранспорт), мають невеликий кузов, відстань до голови та колін невелика при сидінні невелика, тому не підходять для тривалих поїздок, вони невеликі за розміром і легкі, тому їх здатність пристосовуватися до важких умов часто дуже низька;

- клас В (автомобілі малого класу): дані автомобілі схожі на автомобілі класу А, але трохи більші за розміром, розміри автомобілів зазвичай від 370 см до 400 см, а технічно вони міцніші за клас А, є більш важкими, також як і автомобілі класу А мають різні варіанти кузовів;

- клас С (автомобілі субсереднього класу) – це один з найбільш продаваних сегментів: хоча він відомий як компактний клас або нижчий середній клас, часто його називають сімейний автомобіль, а завдяки своїм розмірам вони підходять як для сімейного, так і для індивідуального використання, ефективність автомобіля висока, і вони можуть легко проїхати довгий шлях, оскільки вони фіксують високі показники продажів, конкуренція висока, і знайти доступну модель автомобіля в цьому сегменті набагато легше, на даний момент такі автомобілі зазвичай мають довжину від 420 до 460 см;

– клас D (автомобілі вище середнього класу), що також відомі як автомобілі великої родини: відмінності від класу C полягають у тому, що їх розміри більші, а продуктивність набагато вища, автомобілі мають просторі кабінні та є потужними за двигуном і продуктивністю транспортні засоби, довжина в діапазоні від 453 до 520 см;

– клас E (автомобілі вищого класу): автомобілі потужні з точки зору продуктивності та комфорту, а їх витрати палива набагато прийнятніші, ніж у будь-якому іншому класі, при цьому цінові діапазони все ще досить високі в порівнянні з іншими класами, довжина такого автомобіля знаходиться в діапазоні від 470 до 500 см;

– клас F (автомобілі класу люкс) виділяється своїми технологічними характеристиками та зручностями, а також відомостями про продуктивність та фізичні характеристики, а за розміром, оскільки включають зокрема і лімузини, досягають 570 см;

– клас G (супер розкішні автомобілі) – це унікальний сегмент спортивних автомобілів (спортивні купе) в діапазоні від 400 см до 500 см;

– клас M – це багатоцільові транспортні засоби (multi-purpose vehicles (MPV)), які досягли високих показників продажів у світі, сюди зокрема належать мінівени як частина MPV, зазвичай довжина автомобіля в діапазоні від 405 см до 500 см;

– клас J – це спортивні автомобілі бездоріжжя, мають характеристики спортивного автомобіля та силу позашляховика, часто називаються спортивними позадорожниками або sport utility vehicle (SUV): SUV мають високу витрату палива, при цьому мають на меті бути більш економічними в останній період, в діапазоні від 400 см до 510 см [2]-[3].

Окрім виділення цих традиційних класів автомобілів, враховуючи також те, що не тільки легкові автомобілі можуть паркуватися, а також наявність різних інших систем класифікації, було виділено також наступні класи за розміром.

Серед легкових автомобілів було виділено пікапи: є малі та великі пікапи, малі пікапи, такі як Nissan Navara, зазвичай мають довжину близько 4,9 метра або трохи більше 16 футів, між тим, більші пікапи можуть мати довжину до 5,6 метрів або 18,3 фути, у середньому вони досягають 1,9 метра заввишки [3].

Класи мотоциклів за розміром:

- скутери: середні розміри скутера включають приблизно 1,5 м в довжину, 1,1 м у висоту;
- вуличний мотоцикл: середні розміри вуличного мотоцикла включають 1,5 м у довжину та 1,1 м у висоту;
- спортивний мотоцикл: середні розміри спортивного мотоцикла включають 2,1 м у довжину, 0,6 м у ширину і 1,2 м у висоту;
- туристичний мотоцикл: середні розміри туристичного мотоцикла включають 2,1 м у довжину, 0,7 м у ширину та 1,2 м у висоту;
- стандартні мотоцикли: середні розміри стандартного мотоцикла включають 1,8 м у довжину, 0,8 м в ширину та 1,2 м у висоту [4].

Будинки на колесах або recreational vehicle (RV) є також специфічним транспортом, який саме за розміром можна розділити на наступні класи:

- клас А побудований на шасі транспортного засобу, шасі комерційних вантажівок або шасі комерційного автобуса, вміщає простір для житла: довжина в діапазоні від 6,4 м до 13,7 м;
- клас В – найменші транспортні засоби RV: довжина в діапазоні від 5,2 м до 5,8 м;
- клас С є меншою версією автобудинку RV класу А, побудований на шасі вантажівки або фургона і працює на газовому двигуні: довжина в діапазоні від 6,1 м до 9,5 м [5]

Автобуси мають наступні класи:

- мікроавтобуси або маршрутні автобуси – це транспортні засоби з пасажиромісткістю, меншою за більший повнорозмірний автобус і більше,

ніж у меншого мінівена: маршрутні автобуси мають середню довжину 7,01 м, ширину – 2,24 м, висоту – 2,67 м і місткість 14 (+2) місць;

- туристичний автобус – це автобус, розроблений спеціально для далеких відстаней з більшою зручністю для пасажирів під час цих тривалих періодів подорожі: середня довжина – 12 м, ширина – 2,55 м, висота – 3,81 м і місткість 44-49 (+1) місць;

- міські автобуси, або транзитні автобуси, – це типи автобусів, які використовуються для обслуговування поїздок на короткі та середні відстані: мають середню довжину 11,95 м, ширину – 2,55 м, висоту – 2,99 м і мають місткість 29 (+1) місць для сидіння з розширенням за допомогою стоячих місць до 76;

- з’єднані автобуси – це розширені автобуси, які з’єднують дві або більше секцій за допомогою шарнірних з’єднань для забезпечення більшої пасажиромісткості, при цьому дозволяючи автобусу належним чином маневрувати: мають середню довжину 18 м, ширину – 2,55 м, висоту – 3,13 м і місткість 48 (+1) місць з розширенням кількості стоячих місць до 98 [6].

Вантажівки ще набагато складніше розділити на класи, проте було вирішено використати як базовий підхід наступний:

- трьохвісева вантажівка з довжиною до 12,5 м;
- вантажівка і причіп з довжиною до 19 м;
- шестивісева вантажівка з довжиною до 19 м;
- дев’ятивісева подвійна вантажівка з довжиною до 26 м;
- дванадцятивісева потрійна вантажівка з довжиною до 36,5 м;
- подвійний автопоїзд з довжиною до 36,5 м;
- потрійний автопоїзд з довжиною до 53,5 м [7].

1.2 Концепція програми бронювання паркувальних місць

Програмне забезпечення бронювання паркувальних місць має бути побудовано для реалізації спільних можливостей пошуку і бронювання

паркувальних місць.

Саме по собі бронювання місць на паркувальному майданчику потребує такого програмного забезпечення у випадках:

- наявності загальної автоматизованої контролюючої системи, коли автоматично розпізнається реєстраційний номер автомобіля, а далі підіймається або не підіймається шлагбаум для проїзду на паркувальний майданчик загалом або на певну його ділянку: тоді така автоматизована система може бути інтегрована з даним програмним забезпеченням;

- наявності охорони, яка при в'їзді перевіряє наявність бронювання або доступність місць;

- підключення окремих пристроїв зчитування номерних знаків на паркувальних місцях для інтеграції з цим програмним забезпеченням.

Однак ці умови постають в першу чергу при роботі зі стандартними паркувальними майданчиками, які можуть бути окремо виділеними підприємствами, можуть знаходитися на вулицях тощо. Але є ще один варіант паркування, який в сучасному місті дозволяє вирішити проблему нестачі місць на певних ділянках міста. Це пропозиції власників приватних будинків або окремих ділянок для паркування у них. Умовно це можуть бути індивідуальні паркувальні місця. За наявності таких місць або за тимчасової наявності (коли власник, наприклад, знаходиться в іншому місці) власник може запропонувати власні паркувальні місця для паркування, а програмне забезпечення дозволить контролювати цей процес у такому випадку.

Зважаючи на актуальний стан, в результаті програма може бути направлена на пошук безкоштовних місць паркування у місті, на пошук платних місць, на виставлення власниками паркувальних місць на власних територіях і бронювання цих місць. У випадку інтеграції з іншими системами або наявності охоронців програмне забезпечення може використовуватися також для бронювання паркувальних місць на умовно великих паркувальних майданчиках.

1.3 Аналіз існуючих програм пошуку і бронювання паркувальних місць

Parclick [8] (рис. 1.1) надає для забезпечення паркування в різних містах у 8 країнах світу доступ до вебсайту, а також окремі застосунки для Android, iOS.

Даний застосунок дозволяє виконувати пошук паркувальних майданчиків, зазначаючи потрібні дати і час, обираючи тип автомобіля (легкові, мотоцикли, автобуси, будинки на колесах тощо). Також надається можливість виконання бронювання.

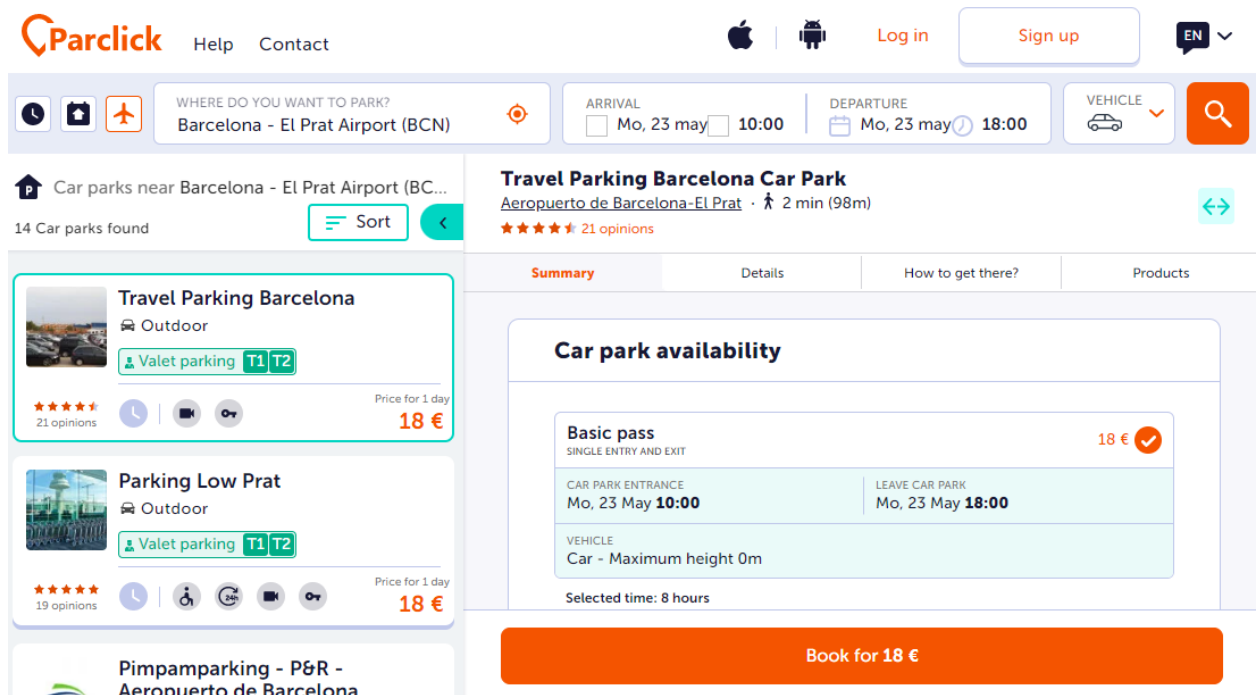


Рисунок 1.1 – Застосунок Parclick

Вебсайт Truck Parking Europe [9] (рис. 1.2) призначений для пошуку паркувальних місць для вантажівок. Оскільки вантажівки вимагають багато додаткової інформації, то цей вебсайт досить цікавий стосовно інформації, яка надається при такому паркуванні, адже паркувальні майданчики мають велику кількість характеристик. Проте логіка організації роботи не відрізняється порівняно з бронюванням для легкових автомобілів.

★★★★★ (4,50)

Lay-by Slamastraße 43

Park your truck at Lay-by Slamastraße 43 close to Wien in Austria. You can always make use of this parking area if there are enough truck parking places available. In this way you can break up your journey through Austria. Some parking areas also offer the possibility to refuel. Truck Parking Europe navigates you to and from petrol stations and truck parking places. Is there still information missing about Lay-by Slamastraße 43 or another roadhouse or highway parking area? Please convey this in the Truck Parking Europe App.

 Share

Security

 (1)

Location

 Slamastraße 43, 1230 Wien Liesing, A

 48°8'15"N 16°20'17"E



Availability

Total spots

 1

Amenities

Comfort

 Shower

 Restaurant

 WC

 Paid parking

 Truckers Life Fitness

 Routers

 Shop

Infrastructure

 Fuel station

 Truck repair

 25m.

 ADR Dangerous goods

 Hotel

 Internet Hotspot

 Medical care

 Opening time

 Washing machine

 ATM

 Food delivery service

 Truck wash

 Power supply

 Autohof

 Ice free scaffolding

Рисунок 1.2 – Вебсайт Truck Parking Europe

Вебсайт дозволяє шукати паркувальні місця, проте не можна встановити потрібну дату і час, а пошук виконується або загалом, або на поточний момент, окрім того надається можливість бронювання місць. У свою чергу саме бронювання можна виконати на потрібну дату і час. За паркувальними майданчиками користувач може створювати власні відгуки.

Результати порівняння представлені в табл. 1.1.

Таблиця 1.1 – Порівняння програмних аналогів

Критерій	Parclick	Truck Parking Europe	Програма
Пошук паркувальних місць	Так	Так	Так
Бронювання	Так	Так	Так
Оцінювання	Так	Так	Так
Типи транспорту	Всі	Вантажівки	Всі
Внесення власних паркувальних місць	Немає	Немає	Так

1.4 Висновки за розділом 1

Проаналізовано особливості проблеми паркування в сучасних містах, зокрема класи автомобілів різного типу за розміром. Визначено концепцію програми, що розробляється. Виконано аналіз програмних аналогів.

Завданнями дипломної роботи є:

- аналіз предметної області;
- визначення вимог до програмного забезпечення бронювання паркувальних місць;
- проєктування програмного забезпечення бронювання паркувальних місць;
- реалізація і налагодження програмного забезпечення бронювання паркувальних місць.

2 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

2.1 Вибір мови програмування

На початковий вибір мов програмування, які можуть бути використані для розробки проєкту, було накладено ідею створення застосунку як вебрішення як універсального засобу організації віддаленої роботи з пристроями різних типів. До результуючої пари включено мови C# [10] і Ruby [11], адже вони дозволяють створювати на поточний момент доволі якісні рішення у цьому розрізі.

Порівняння мов (табл. 2.1) виконувалось на основі критерії часу обробки коду, що визначає швидкість роботи результуючого проєкту, стабільності проєкту, що важливо для якості взаємодії з програмою загалом, швидкості кодування та кількості документації, що визначають зручність роботи розробника над програмою. Мова C# обрана, оскільки за рахунок того, що мова Ruby за рахунок інтерпретованості має більший час обробки коду, відповідно програми працюють повільніше, має менше документації, що зменшує зручність її використання, до того ж якщо брати до уваги великі проєкти, а програмне забезпечення бронювання паркувальних місць навіть у розрізі тільки індивідуальних приватних паркувальних місць, не кажучи вже про підтримку великих паркувальних майданчиків, є саме великим проєктом.

Таблиця 2.1 – Вибір мови програмування

Критерій	C#	Ruby
Фреймворк	ASP.NET	Ruby on Rails
Час обробки коду	Менше	Більше
Стабільність	Висока, в тому числі у великих проєктах	Більша направленість на невеликі проєкти
Швидкість кодування	Краща для більших проєктів	Краща для невеликих проєктів
Кількість документації	Велика	Невелика

2.2 Визначення функціональних вимог до програми

Програма, враховуючи її направленість на роботу з паркувальними майданчиками різного типу і особливо на роботу з індивідуальними приватними паркувальними місцями, повинна дозволяти виконувати:

- бронювання паркувального місця;
- скасування бронювання;
- пошук потрібного паркувального місця;
- відтворення паркувальних майданчиків за заданим містом;
- відтворення списку всіх бронювань користувача за автомобілем;
- внесення власного автомобіля;
- рецензування паркувального майданчика за реалізованим бронюванням;
- внесення паркувальних місць багатомісних і індивідуальних паркувальних майданчиків;
- управління списком менеджерів паркувального майданчика;
- управління доступністю паркувального місця;
- управління бронюваннями за паркувальним майданчиком;
- управління доступом користувачів;
- автоматизоване скасування бронювання;
- автоматизоване блокування акаунту користувача;
- відтворення доступних на заданий момент паркувальних місць у місті або за паркувальним майданчиком;
- уточнення інформації про паркувальний майданчик;
- реєстрація і авторизація користувачів і менеджерів.

Створену множину функціональних вимог було використано під час моделювання роботи користувачів з програмою (рис. 2.1). Серед користувачів було виділено два типи, перший з яких є по суті власником автомобіля, і названий користувачем, а другий – власником або менеджером паркувального майданчика, а названий менеджером.

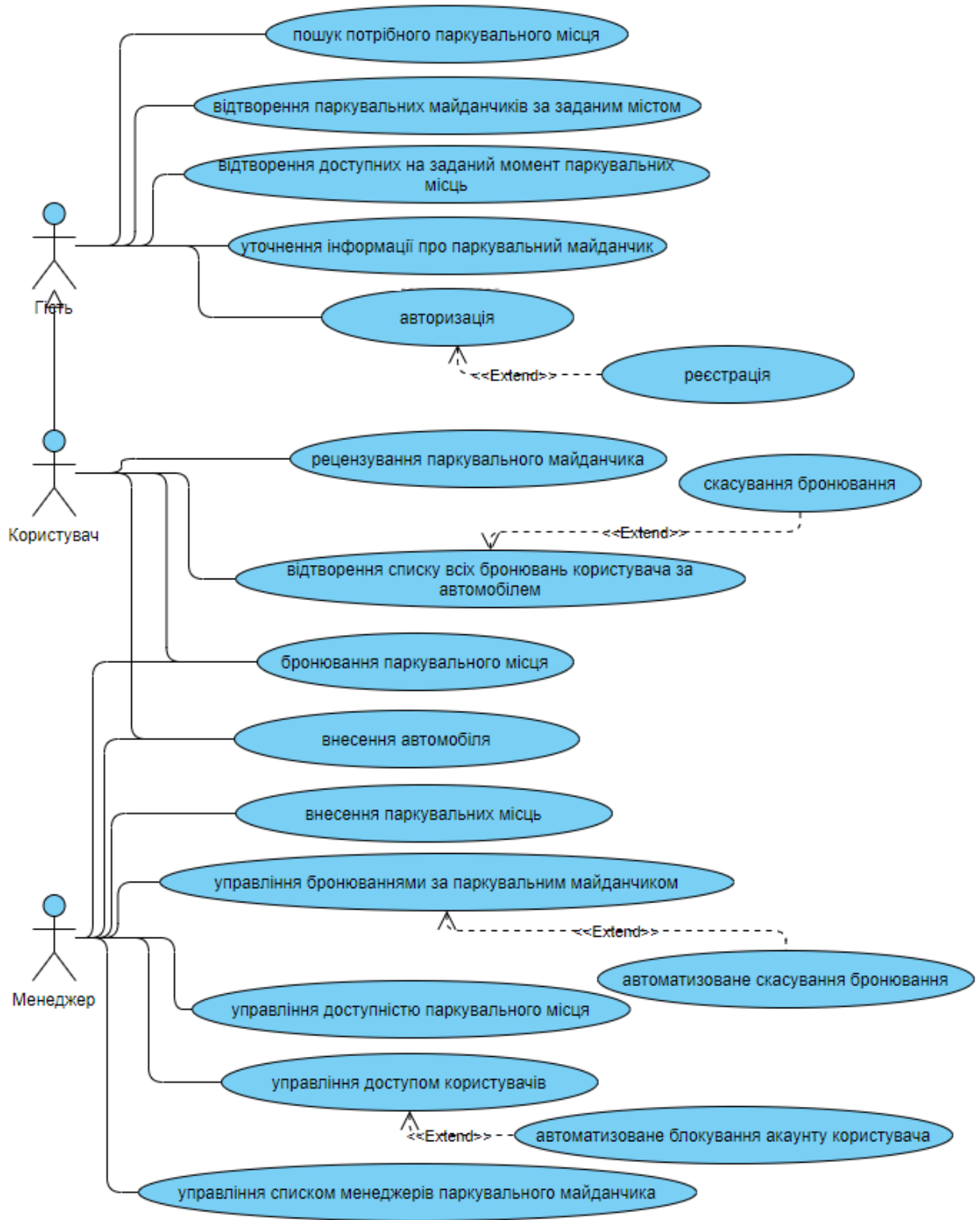


Рисунок 2.1 – Діаграма прецедентів

2.3 Проектування бази даних

Виконуючи проектування бази даних, спочатку було виділено основні сутності предметної області.

Під час роботи з паркомісцями звичайно важливо швидко знайти потрібно паркомісце, а для цього потрібно структурувати адресу. Зазвичай парковки визначаються в межах міста, тому з цієї логіки визначено і структурування, а вже після цього пошук може виконуватися за адресою або за координатами.

З метою структурування доступу до паркомісця спочатку було виділено сутність Країна (Country), а далі вже сутність Місто (City), яка відповідає конкретній країні.

У межах конкретного міста знаходяться різні паркувальні майданчики. Сутність паркувального майданчика (Parking) визначає будівлю, спеціально виділений відкритий майданчик або вуличну парковку (наприклад, розташовану вздовж вулиці). Також до паркувального майданчика відносяться індивідуальні парковки, які можуть бути приватною власністю.

Для керування паркомісцями виділяються відповідні менеджери. Сутність Менеджера (Manager) визначає особу, яка може вносити зміни щодо відповідної парковки. Таким менеджером може бути тільки особа, яка створила запис про паркінг. Для індивідуальної приватної парковки таким менеджером є власник парковки. Хоча він може також додавати менеджерів для тимчасової допомоги керування парковками. Оскільки таких менеджерів може бути декілька, а кожен може працювати з різними парковками, то створено таблицю ParkingManager.

Для кожного паркувального майданчика передбачено, що кількість місць може бути великою, а тоді потрібно їх правильно структурувати. Така структурування може використовуватися як для узагальнення даних про паркувальні місця, щоб у випадку їх однаковості вони не повторювались, так і для більш зручного визначення вартості, адже в такому випадку (коли місця подібні) вартість може бути однаковою. Тому ще однією сутністю є тип паркувального місця Lottype.

Відповідно саме паркувальне місце виділено в сутність Lot. Фактично для приватного паркувального місця, яке власник хоче здати в оренду, це паркувальне місце є єдиною складовою, але для структуризації паркувальних майданчиків виділено структуру сутностей Parking – Lottype – Lot.

Для того щоб користувачі орієнтувались в наявних парковках, виділено сутність Рецензія (Review) для забезпечення збереження рецензій. Рецензії вирішено створювати саме для паркувального майданчика загалом, адже у випадку великої кількості місць на майданчику немає сенсу створювати рецензію про місце під номером 90, адже воно зазвичай є типовим для даного майданчика. У тексті рецензії за необхідності можна відзначити, якої саме позиції стосується рецензія, якщо це все-таки потрібне. Саме таке рішення прийнято.

У відповідність самому бронюванню створено сутність Бронювання (Reservation). Бронювання потрібні для відслідковування бронювань конкретного користувача, а також для відслідковування доступності або зайнятості паркувальних місць у певний час. Бронювання вирішено виконувати для конкретного паркувального місця, хоча можна було б встановити для типу паркувального місця, адже по суті, якщо місця подібні, то вони не будуть фактично відрізнятися для автомобіліста. Але такий варіант потенційно міг би призвести до того, що в певний момент у випадку фізичної реалізації слідкування на основі датчиків і/або камер паркувань користувачу могло б не дістатися вільне місце. Для уникнення такої ситуації бронювання відбувається саме за конкретним місцем, проте може відзначатися, що конкретне місце відповідного типу для користувача непринципове.

Оскільки автомобілі відрізняються між собою в певній мірі, але у ракурсі паркувального місця важливим є розмір автомобіля, то можна під час встановлення відповідності використовувати не безпосередньо повні розміри, а клас автомобіля за розміром. Відповідність цих класів розмірам встановлена в сутності Cartype. Виділення для цього окремої сутності

дозволяє за необхідності використовувати будь-яку систему класифікації автомобілів за розміром, яка потрібна, або створити власну за наявності такої потреби.

Ще одними даними, які потрібно зберігати, є дані про ціну паркування. Ціну паркування вирішено розглядати в двох ракурсах. Перший ракурс – стандартне міське ціноутворення. Зазвичай в такому випадку може вартість паркування може бути або однаковою або різною в різні години різних днів тижня. У такому випадку незалежно від місяця, сезону вартість буде однаковою для одних і тих самих днів тижня. Єдиною відмінністю є випадок, коли відповідний день є святом, тоді може вмикатися інше регулювання цін. Другий ракурс – різна вартість в різні дати, в різні години цих дат. Такий варіант створює більш гнучкий спосіб роботи з цінами, якщо власник паркувального майданчика потребує саме такого варіанту. Він загалом може бути менш розповсюдженим, але з іншого боку його може потребувати приватний власник індивідуальних паркувальних місць. Ще одним варіантом може бути стабільна вартість, тоді для цього варіанту можна використати перший ракурс.

Перший ракурс ціноутворення представлений сутністю Pricesystem. У ній представлене стандартне ціноутворення.

Другий ракурс ціноутворення представлений сутністю Offer. У ній представлене спеціальне ціноутворення.

Сутність Car необхідна для збереження конкретних автомобілів. Така сутність необхідна для збереження реєстраційних номерів автомобілів, щоб потім перевірити наявність для конкретного автомобіля конкретного місця, якщо це необхідне.

Сутність User визначає користувача, який виконує бронювання паркувальних місць. Оскільки кожен користувач може мати різні автомобілі, а кожен автомобіль історично може належати різним власникам або використовуватися різними людьми, які до того ж можуть бути

користувачами програми, інформацію про приналежність автомобіля користувачу представлено таблицею UserCar.

Таблиця Country має зв'язок з таблицею City через її поле CountryId типом зв'язку один до багатьох через те, що країна представлена містами, а окреме місто розташовується виключно в одній країні одночасно.

Таблиця City має зв'язок з таблицею Parking через її поле CityId типом зв'язку один до багатьох через те, що місто в розрізі даного програмного забезпечення представлене паркувальними майданчиками, а окремий паркувальний майданчик розташовується виключно в одному місті одночасно.

Таблиця Parking має зв'язок з таблицею ParkingManager через її поле ParkingId типом зв'язку один до багатьох через те, що кожний паркувальний майданчик може мати багато менеджерів, що виражено відповідним записом в таблиці ParkingManager. Таблиця Manager має зв'язок з таблицею ParkingManager через її поле ManagerId типом зв'язку один до багатьох через те, що кожний менеджер потенційно може працювати з різними паркувальними майданчиками, що виражено відповідним записом в таблиці ParkingManager. По суті таблиця ParkingManager реалізує зв'язок типу багато до багатьох між таблицями Parking і Manager.

Таблиця ApplicationUser має зв'язок з таблицею UserCar через її поле UserId типом зв'язку один до багатьох через те, що кожний користувач може паркуватися з різними автомобілями, що виражено відповідним записом в таблиці UserCar. Таблиця Car має зв'язок з таблицею UserCar через її поле CarId типом зв'язку один до багатьох через те, що кожний автомобіль може керуватися різними користувачами, що виражено відповідним записом в таблиці UserCar. По суті таблиця UserCar реалізує зв'язок типу багато до багатьох між таблицями ApplicationUser і Car.

Таблиця Parking має зв'язок з таблицею Lottype через її поле ParkingId типом зв'язку один до багатьох через те, що кожен паркувальний майданчик

може мати різні типи паркомісць, а окремий такий тип паркомісця визначається тільки для одного відповідного паркувального майданчика.

Таблиця Cartype має зв'язок з таблицею Lotype через її поле CartypeId типом зв'язку один до багатьох через те, що кожен клас автомобіля може мати багато типів паркувальних місць на різних паркувальних майданчиках, а окремий такий тип паркомісця визначається тільки для одного відповідного класу автомобіля.

Таблиця Lotype має зв'язок з таблицею Lot через її поле LotypeId типом зв'язку один до багатьох через те, що кожен тип паркомісця на паркувальному майданчику може мати багато паркомісць, але окреме паркувальне місце відповідає тільки одному типу паркомісця на паркувальному майданчику, що відповідає найбільшому класу автомобіля.

Таблиця User має зв'язок з таблицею Review через її поле UserId типом зв'язку один до багатьох через те, що кожен користувач може мати багато рецензій на паркувальні майданчики, але кожна рецензія може бути створена тільки одним користувачем.

Таблиця Parking має зв'язок з таблицею Review через її поле UserId типом зв'язку один до багатьох через те, що кожен користувач може мати багато рецензій на паркувальні майданчики, але кожна рецензія може бути створена тільки одним користувачем.

Таблиця UserCar має зв'язок з таблицею Reservation через її поле UserCarId типом зв'язку один до багатьох через те, що кожен автомобіль під керуванням одного користувача може мати багато бронювань паркувальних місць в різний час, але кожне бронювання може стосуватися тільки одного конкретного автомобіля під керуванням відповідного користувача.

Таблиця Lot має зв'язок з таблицею Reservation через її поле LotId типом зв'язку один до багатьох через те, що кожне паркувальне місце може мати багато бронювань в різний час, проте кожне бронювання стосується тільки одного конкретного паркувального місця.

Таблиця Lottype має зв'язок з таблицею Pricesystem через її поле Lottypeld типом зв'язку один до багатьох через те, що кожний тип паркувального місця на паркувальному майданчику може мати багато стандартних ціноутворень, проте кожне стандартне ціноутворення стосується тільки одного конкретного типу маркувального місця на паркувальному майданчику.

Таблиця Lottype має зв'язок з таблицею Offer через її поле Lottypeld типом зв'язку один до багатьох через те, що кожний тип паркувального місця на паркувальному майданчику може мати багато спеціальних ціноутворень, проте кожне спеціальне ціноутворення стосується тільки одного конкретного типу маркувального місця на паркувальному майданчику.

Схема бази даних показана на рис. 2.2.

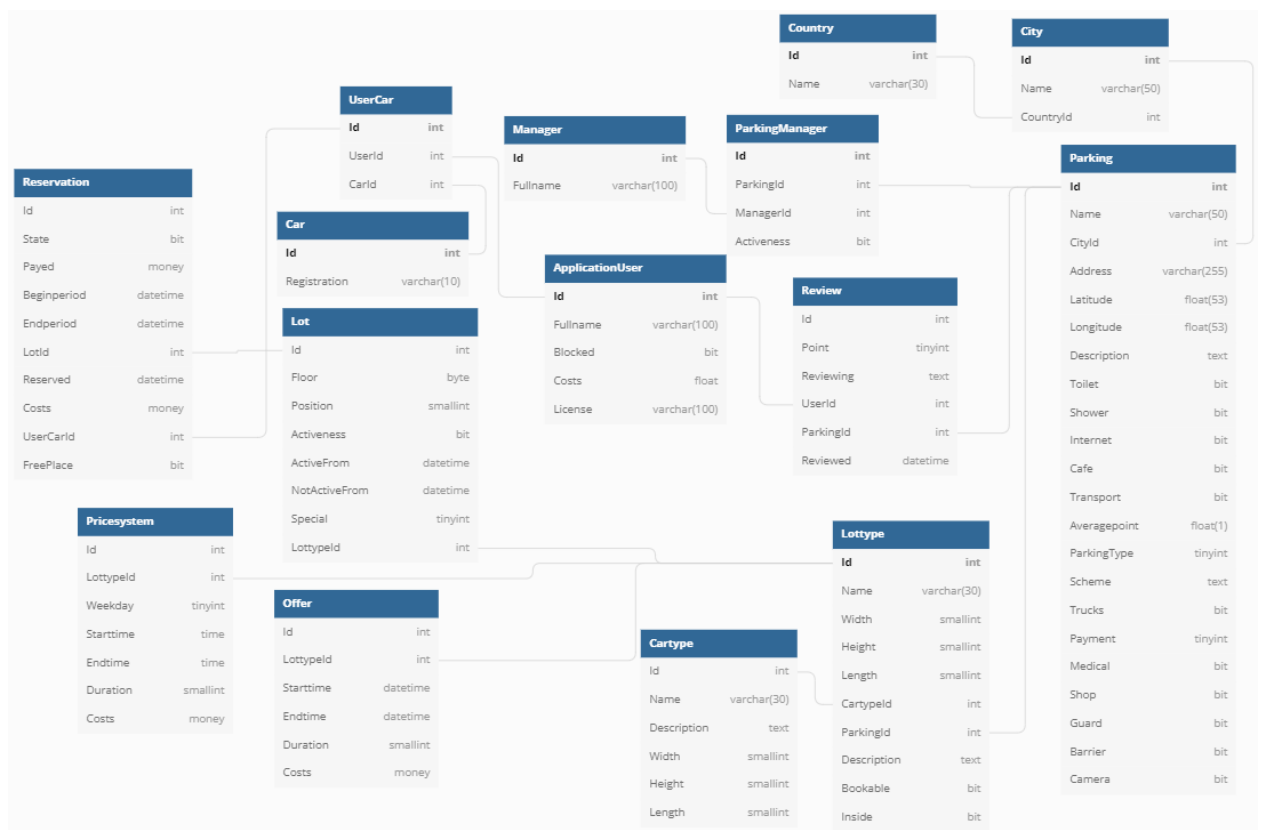


Рисунок 2.2 – Схема бази даних

Для роботи з базою даних використано базову для платформи .NET систему управління базами даних MS SQL Server.

2.4 Вибір архітектури програми

При побудові застосунку використано багатошаровий варіант її побудови (рис. 2.3).



Рисунок 2.3 – Архітектура програми

Використана в результаті архітектура програми є трирівневою: складається з рівня доступу до даних, що має забезпечувати визначення відповідної структури даних, проміжного шару, який надає бізнес-логіку застосунку і сервіси застосунку, необхідні для визначення засобі роботи з даними у відповідності з функціями застосунку, та презентаційного шару, на якому забезпечується формування вебсторінок для доступу користувачів до застосунку [12]-[13].

2.5 Висновки за розділом 2

Порівняно мови програмування C# і Ruby для реалізації програмного забезпечення бронювання паркувальних місць, в результаті вибрано мову C#. Виділення функцій програми дозволило виконати моделювання сценаріїв роботи програми та на основі цього забезпечити проєктування бази даних. Обрано трирівневу архітектуру для побудови програми.

З РОЗРОБКА ПРОГРАМИ

3.1 Алгоритм функціонування програми

Функціонування програми пов'язано з взаємодією з користувачем в різних режимах. Ці режими по суті визначаються роллю, яку при взаємодії виконує користувач.

Робота користувача розпочинається з відтворення паркувальних майданчиків за містом, після чого він може вибрати варіант продовження, що включає пошук паркувальних місць на паркувальних майданчиках, те ж саме відтворення паркувальних майданчиків за містом, завершення роботи з програмою або авторизацію.

Після авторизації визначається роль користувача.

Якщо він працює як користувач без привілеїв (менеджер), то йому стає доступним вибір з меню можливостей відтворення паркувальних майданчиків за містом, що також доступно і без авторизації, пошук паркувальних місць, що також доступно і без авторизації, управління власними бронюваннями, управління автомобілями, рецензування, завершення роботи з програмою або вихід з облікового запису. У разі останнього варіанту робота по суті починається заново, як було описано вище.

Якщо користувач авторизований менеджером, то йому дозволяється виконувати управління бронюваннями за паркувальними майданчиками, які доступні йому під час менеджерської роботи, управління користувачами, управління тарифами на паркування на паркувальних майданчиках, управління доступними для його роботи паркувальними майданчиками (рис. 3.1). Також менеджер може вийти з облікового запису, може завершити роботу з програмою. У першому випадку також робота по суті починається заново, як було описано вище.

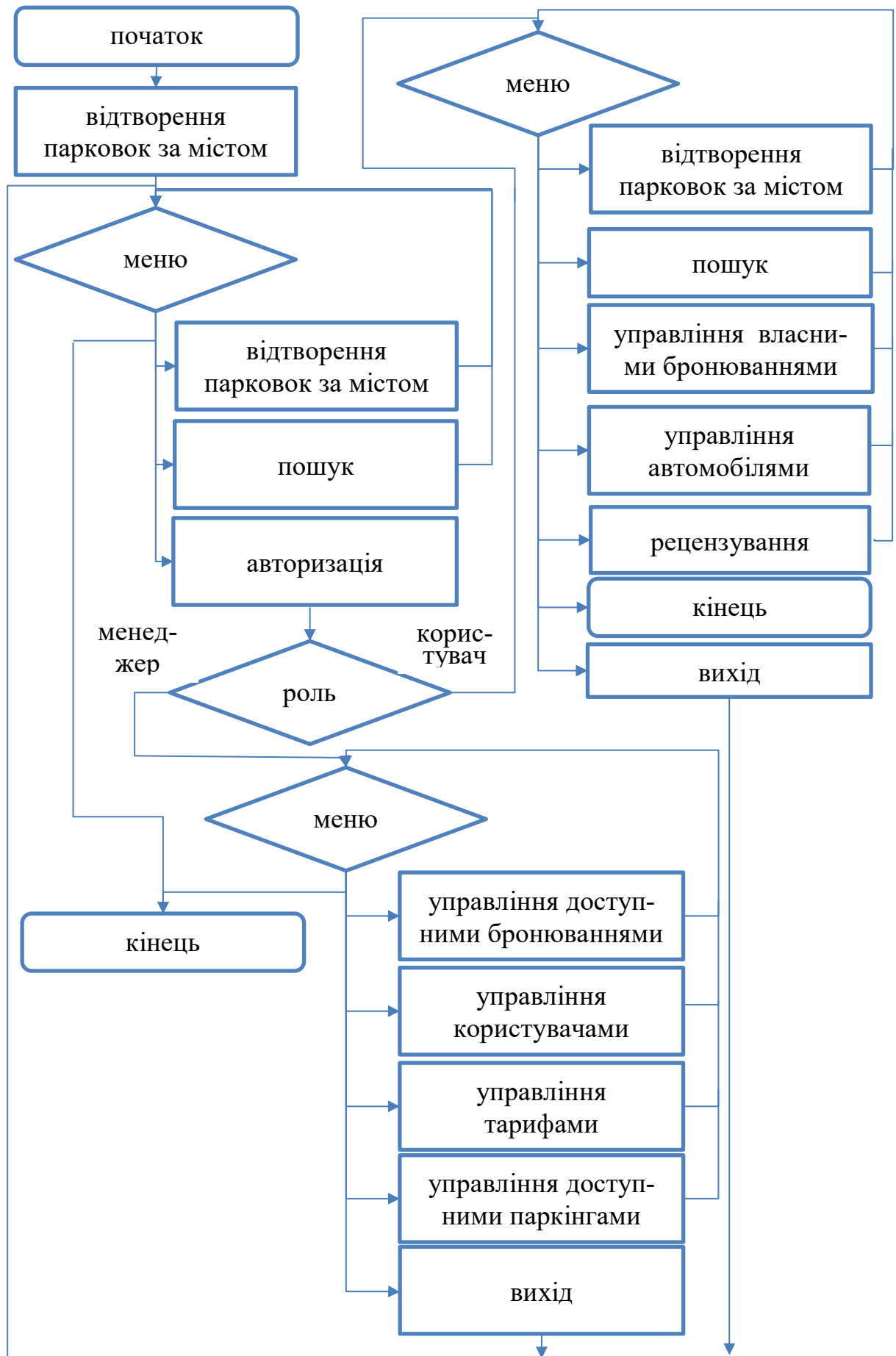


Рисунок 3.1 – Схема функціонування програми

3.2 Опис реалізованих класів

Класи реалізовані у чіткій відповідності зі структурною схемою для забезпечення підтримки функціональності програми.

Клас моделі Lot визначає модель паркувального місця і містить такі поля:

- public int Id – номер паркувального місця;
- public byte Floor – поверх паркувального майданчика, на якому розташовано це паркувальне місце;
- public byte Position – номер місця на визначеному поверсі паркувального майданчика;
- public bool Activeness – поле, яке визначає, чи є дане паркувальне місце активним (доступним для бронювання) у поточний момент;
- public DateTime ActiveFrom – дата і час, з яких паркувальне місце буде активним (доступним для бронювання) у випадку, якщо місце неактивне в поточний момент;
- public DateTime NotActiveFrom – дата і час, з яких паркувальне місце буде неактивним (недоступним для бронювання) у випадку, якщо місце активне в поточний момент;
- public byte Special – поле, яке визначає, чи це звичайне, чи спеціальне місце (місце для водіїв з інвалідністю, для спеціального транспорту, для автобусів тощо);
- public int LottypeId – номер типу паркувального місця;
- public virtual ICollection<Reservation> Reservations – набір усіх бронювань цього паркомісця;
- public virtual Lottype Lottype – тип паркувального місця.

Клас моделі Lottype визначає модель типу паркувального місця і містить такі поля:

- public int Id – номер типу паркувального місця;

- public string Name – назва типу паркувального місця;
- public short Width – ширина паркувальних місць даного типу;
- public short Height – висота паркувальних місць даного типу;
- public short Length – довжина паркувальних місць даного типу;
- public bool Inside – поле, яке визначає, чи знаходяться в приміщенні або ззовні паркувальні місця цього типу;
- public int CartypeId – номер максимального класу машин, що можуть розташовуватися на паркувальних місцях цього типу;
- public int ParkingId – номер паркувального майданчика, на якому розташовані ці паркувальні місця;
- public string Description – опис цього типу маркувальних місць;
- public bool Bookable – поле, яке визначає, чи можливо бронювати паркувальні місця цього типу;
- public bool Electricity – наявність електрозарядки;
- public virtual ICollection<Lot> Lots – набір паркувальних місць цього типу;
- public virtual ICollection<Pricesystem> Pricesystems – набір стандартних цін на бронювання паркувальних місць цього типу;
- public virtual ICollection<Offer> Offers – набір спеціальних цін на бронювання паркувальних місць цього типу;
- public virtual Cartype Cartype – максимальний клас автомобіля, який може розташовуватися на паркувальних місцях цього типу;
- public virtual Parking Parking – паркувальний майданчик, якому відповідає цей тип паркувальних місць.

Клас моделі Parking визначає модель паркувального майданчика і містить такі поля:

- public int Id – номер паркувального майданчика;
- public string Name – назва паркувального майданчика;
- public int CityId – номер міста, де розташований паркувальний майданчик;

- public string Address – повна адреса, за якою розташований паркувальний майданчик;
- public double Latitude – широта, за якою розташований паркувальний майданчик;
- public double Longitude – довгота, за якою розташований паркувальний майданчик;
- public string Description – опис паркувального майданчика;
- public bool Toilet – наявність туалетів на паркувальному майданчику;
- public bool Shower – наявність душі на паркувальному майданчику;
- public bool Internet – наявність інтернету на паркувальному майданчику;
- public bool Cafe – наявність кав'ярні на паркувальному майданчику;
- public bool Transport – наявність громадського транспорту для переміщення містом від самого паркувального майданчика;
- public float Averagepoint – середня оцінка в рецензіях паркувальному майданчику;
- public byte ParkingType – тип паркувального майданчика;
- public string Scheme – шлях до файлу з зображенням схеми паркування;
- public bool Trucks – наявність місць для вантажівок;
- public byte Payment – метод оплати;
- public bool Medical – наявність медичного пункту;
- public bool Shop – наявність магазину;
- public bool Guard – наявність охорони;
- public bool Barrier – наявність бар'єру для в'їзду;
- public bool Camera – наявність камер відеоспостереження;
- public virtual ICollection<Lottype> Lottypes – набір типів паркувальних місць, наявних на паркувальному майданчику;

– public virtual ICollection<Review> Reviews – набір рецензій паркувального майданчика;

– public virtual ICollection<ParkingManager> ParkingManagers – набір менеджерів, які можуть керувати паркувальним майданчиком;

– public virtual City City – місто, в якому розташований паркувальний майданчик.

Клас моделі Review визначає модель рецензії і містить такі поля:

– public int Id – номер рецензії;

– public byte Point – бал, виставлений в рецензії паркувальному майданчику;

– public string Reviewing – текст рецензії;

– public int UserId – номер користувача, який є автором рецензії;

– public int ParkingId – номер паркувального майданчика, за яким створено рецензію;

– public DateTime Reviewed – дата і час створення рецензії;

– public virtual User User – автор рецензії;

– public virtual Parking Parking – паркувальний майданчик, щодо якого створена рецензія.

Клас моделі Reservation визначає модель бронювання і містить такі поля:

– public int Id – номер бронювання;

– public bool State – стан бронювання;

– public float Costs – вартість бронювання;

– public float Payed – сплачена сума за бронювання;

– public int UserCarId – номер автомобілю користувача;

– public int LotId – номер паркувального місця, яке було заброньовано;

– public DateTime Beginperiod – дата і час початку бронювання;

– public DateTime Endperiod – дата і час завершення бронювання;

– public DateTime Reserved – дата і час, коли було виконано бронювання;

- public bool FreePlace – поле, яке визначає можливість вільного вибору місця цього типу користувачем, тобто чи важливо для користувача саме це місце на паркувальному майданчику чи вибір здійснюється тільки за типом паркувальних місць;

- public virtual Lot Lot – паркувальне місце, бронювання якого виконано;

- public virtual UserCar UserCar – автомобіль користувача, для якого бронювання було виконано.

Клас моделі Cartype визначає клас автомобіля і містить такі поля:

- public int Id – номер класу автомобіля;
- public string Name – назва класу автомобіля;
- public string Description – опис;
- public short Width – ширина;
- public short Height – висота;
- public short Length – довжина;
- public virtual ICollection<Lottype> Lottypes – типи паркомісць.

Клас моделі Pricesystem визначає клас стандартної цінової пропозиції і містить такі поля:

- public int Id – номер стандартної цінової пропозиції;
- public byte Weekday – номер дня тижня;
- public TimeSpan Starttime – година, з якої починає діяти цінова пропозиція;
- public TimeSpan Endtime – година, до якої діє цінова пропозиція;
- public byte Duration – максимальна кількість хвилин, за яку можна сплатити за цією ціною пропозицією;
- public float Costs – вартість хвилини паркування;
- public int LottypeId – номер типу паркувального місця на паркувальному майданчику, за яким діє ця стандартна цінова пропозиція;
- public virtual Lottype Lottype – тип паркувального місця на паркувальному майданчику, за яким діє ця стандартна цінова пропозиція.

Клас моделі Offer визначає клас спеціальної цінової пропозиції і містить такі поля:

- public int Id – номер спеціальної цінової пропозиції;
- public DateTime Starttime – дата і час початку дії цієї спеціальної цінової пропозиції;
- public DateTime Endtime – дата і час завершення дії цієї спеціальної цінової пропозиції;
- public byte Duration – максимальна кількість хвилин, за яку можна сплатити за цією ціною пропозицією;
- public float Costs – вартість хвилини паркування;
- public int LottypeId – номер типу паркувального місця на паркувальному майданчику, за яким діє ця спеціальна цінова пропозиція;
- public virtual Lottype Lottype – тип паркувального місця на паркувальному майданчику, за яким діє ця спеціальна цінова пропозиція.

Клас моделі Manager визначає клас менеджера паркувального майданчика через наслідування стандартного класу IdentityUser і містить в якості додаткових полів:

- public string Fullname – повне ім'я менеджера;
- public virtual ICollection<ParkingManager> ParkingManagers – паркувальні майданчики, керуванням яким займається менеджер.

Аналогічно клас моделі ApplicationUser визначає клас користувача також через наслідування стандартного класу IdentityUser і містить в якості додаткових полів:

- public string Fullname – повне ім'я користувача;
- public string License – водійське посвідчення;
- public bool Blocked – поле, яке визначає, чи користувач є заблокованим, чи може продовжувати працювати;
- public float Costs – сумарні витрати користувача на паркування в програмі;
- public virtual ICollection<Review> Reviews – відгуки користувача;

– public virtual ICollection<UserCar> UserCars – автомобілі, якими керував користувач.

Клас моделі ParkingManager визначає клас менеджера паркувального місця і містить такі поля:

– public int Id – номер співвідношення між менеджером і паркувальним майданчиком;

– public int ParkingId – номер паркувального майданчика;

– public int ManagerId – номер менеджера, який керує паркувальним майданчиком;

– public bool Activeness – поле, яке визначає, чи керує цим паркувальним майданчиком менеджер на даний момент, чи це вже минулий факт;

– public virtual Manager Manager – менеджер, який керує паркувальним майданчиком;

– public virtual Parking Parking – паркувальний майданчик.

Клас моделі UserCar визначає клас автомобіля користувача і містить такі поля:

– public int Id – номер співвідношення між користувачем і автомобілем;

– public int UserId – номер користувача;

– public int CarId – номер автомобіля;

– public virtual User User – користувач, який керує автомобілем;

– public virtual Car Car – автомобіль, яким керує користувач;

– public virtual ICollection<Reservation> Reservations – бронювання за дним автомобілем і користувачем.

Клас моделі Car визначає клас автомобіля і містить такі поля:

– public int Id – номер автомобіля;

– public string Registration – реєстраційний номер автомобіля;

– public virtual ICollection<UserCar> UserCars – користувачі, які керують автомобілем.

Клас моделі Country визначає клас країни і містить такі поля:

- public int Id – номер країни;
- public string Name – назва країни;
- public virtual ICollection<City> Cities – міста, які розташовані в цій країні.

Клас моделі City визначає клас міста і містить такі поля:

- public int Id – номер міста;
- public string Name – назва міста;
- public int CountryId – номер країни, в якій розташовано місто;
- public virtual Country Country – країна, в якій розташовано місто;
- public virtual ICollection<Parking> Parkings – паркувальні майданчики, які розташовані в цьому місті.

Клас ParkingService визначає сервіс роботи з паркувальними майданчиками і забезпечує методи:

- public ParkingService (DataContext dc) – визначає dc як контекст даних;
- public Parking GetParking(int parkid) – повертає повний об'єкт паркувального майданчика за його номером parkid;
- public IEnumerable<Parking> GetParkings() – повертає список всіх паркувальних майданчиків;
- public IEnumerable<Parking> GetParkingsByCity(int cityid) – повертає список паркувальних майданчиків у місті з номером cityid;
- public IEnumerable<Parking> GetParkingsByManager(Manager man) – повертає список паркувальних майданчиків, якими може керувати менеджер man.

Клас ReservationService визначає сервіс роботи з самими бронюваннями і забезпечує методи:

- public ReservationService (DataContext dc) – визначає dc як контекст даних;

- `public Reservation GetReservation(int reserid)` – повертає об'єкт бронювання за номером `reserid`;
- `public IEnumerable<Reservation> GetUserReservations(User us)` – повертає бронювання користувача `us`;
- `public IEnumerable<Reservation> GetReservationsByParking(Parking park, byte state)` – повертає бронювання за паркувальним майданчиком `park` у стані `state`;
- `public async Task Add(Reservation res)` – вносить нове бронювання `res` до бази даних;
- `public async Task Update(Reservation res)` – оновлює бронювання `res` у базі даних;
- `public async Task ChangeState(int resid, byte state)` – змінює стан бронювання з номером `resid` на `state`;
- `public async Task AutoCancel()` – автоматично скасовує бронювання, якщо після 15 хвилин від запланованого часу початку за ними не було виконано паркування.

Клас `ParkingController` виконує підготовку даних для наповнення відповідних представлень паркувальних майданчиків на основі методів:

- `public ParkingController (IParking parkingService, IReservation reservationService, IOffer offerService, IPricesystem pricesystemService, ICity cityService, IHostingEnvironment henv, IApplicationUser applicationService, UserManager<ApplicationUser> userManager, IManager managerService)` – конструктор, що виконує визначення відповідних відповідних сервісів, необхідних для роботи контролера;
- `public async Task<IActionResult> Details(int? parkid, int? day, int? month, int? year, int? hour, int? minute)` – виконує заповнення даними представлення сторінки про парковку з номером `parkid` з визначенням кількості вільних місць на дату `day.month.year` на час `hour:minute`;
- `public IActionResult Create()` – виконує підготовку до внесення нового паркувального майданчика за допомогою GET-запиту;

– `public async Task<IActionResult> Create([Bind("Id, Name, Address, Latitude, Longitude, Description, Toilet, Shower, Internet, Cafe, Transport, ParkingType, Trucks, Payment, Medical, Shop, Guard, Barrier, Camera, CityId")] Parking parking, IFormFile schemepath)` – виконує внесення нового паркувального майданчику з використанням схеми майданчику на основі файлу зображення `schemepath` за допомогою POST-запиту;

– `public async Task<IActionResult> Edit(int? parkid)` – виконує підготовку до редагування паркінгу з номером `parkid` за допомогою GET-запиту;

– `public async Task<IActionResult> Edit(int parkid, [Bind("Id, Name, Address, Latitude, Longitude, Description, Toilet, Shower, Internet, Cafe, Transport, ParkingType, Trucks, Payment, Medical, Shop, Guard, Barrier, Camera, CityId")] Parking parking, IFormFile schemepath)` – виконує редагування паркувального майданчика з номером `parkid` з використанням схеми майданчика на основі файлу зображення `schemepath` за допомогою POST-запиту;

– `public IActionResult CityParkings(int cityid)` – виконує заповнення даними представлення сторінки паркувальних майданчиків у місті з номером `cityid`;

– `public IActionResult ManagerParkings()` – виконує заповнення даними представлення сторінки про паркувальні майданчики, з якими може працювати менеджер, який авторизувався в програмі;

– `public IActionResult Index()` – виконує заповнення даними представлення стандартної сторінки, яка містить список всіх паркувальних майданчиків, що повертається у випадку успішного редагування або створення паркувального майданчика.

Клас `ReservationController` виконує підготовку даних для наповнення відповідних представлень бронювання на основі методів:

– `public ReservationController (IReservation reservationService, ICar carService, ILot lotService, IOffer _offerService, IPricesystem`

`_pricesystemService, IApplicationUser applicationService, UserManager<ApplicationUser> userManager, IManager managerService)` – конструктор, що виконує визначення відповідних відповідних сервісів, необхідних для роботи контролера;

- `public async Task<IActionResult> Details(int reserid)` – виконує заповнення даними представлення сторінки про бронювання з номером `reserid`;

- `public IActionResult Create(int parkid)` – виконує створення нового бронювання на паркувальному майданчику з номером `parkid` за допомогою GET-запиту;

- `public async Task<IActionResult> Create([Bind("Id, Payed, UserCarId, LotId, FreePlace")] Reservarion reserv, int Beginperiod_year, int Beginperiod_month, int Beginperiod_day, int Beginperiod_hour, int Beginperiod_minute, int Endperiod_year, int Endperiod_month, int Endperiod_day, int Endperiod_hour, int Endperiod_minute, int parkid)` – виконує створення нового бронювання за допомогою POST-запиту;

- `public async Task<IActionResult> Edit(int? reserid, int parkid)` – виконує редагування бронювання з номером `reserid` за допомогою GET-запиту;

- `public async Task<IActionResult> Edit(int reserid, [Bind("Id, State, Payed, UserCarId, LotId, FreePlace")] Reservarion reserv, int Beginperiod_year, int Beginperiod_month, int Beginperiod_day, int Beginperiod_hour, int Beginperiod_minute, int Endperiod_year, int Endperiod_month, int Endperiod_day, int Endperiod_hour, int Endperiod_minute)` – виконує редагування бронювання з номером `reserid` за допомогою POST-запиту;

- `public IActionResult OwnReservations()` – виконує заповнення даними представлення сторінки зі списком власних бронювань авторизованого користувача;

- `public IActionResult Start(int reserid)` – виконує визначення початку паркування за бронюванням `reserid`;

- `public IActionResult Cancel(int reserid)` – виконує скасування бронювання з номером `reserid`;
- `public IActionResult Return(int reserid, float payed)` – виконує визначення завершення паркування за бронюванням `reserid` зі сплатою суми `payed`;
- `public IActionResult ParkingReservations(int? parkid)` – виконує заповнення даними сторінки зі списком бронювань за паркувальним майданчиком з номером `parkid` або для першого паркінгу зі всіх доступних для авторизованого менеджера.

3.3 Висновки за розділом 3

Представлено результати розробки програми, що охоплюють структуру розробленого рішення за відповідною структурною схемою програми та структуру основних реалізованих класів.

4 ЕКСПЛУАТАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМИ

4.1 Призначення програми

Програма призначена для організації пошуку паркувальних майданчиків у місті, для бронювання паркувальних місць на цих майданчиках з приділенням особливої уваги приватним індивідуальним паркувальним місцям і забезпеченням інструментів внесення таких паркувальних місць їх власниками до програми та для надання інструментів для потенційного широкого доступу до засобів бронювання для багатомісних паркувальних майданчиків.

4.2 Умови виконання програми

Програмне забезпечення в якості апаратних вимог визначає вимоги, що включають мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

4.3 Експлуатація програми

Оскільки з програмою можуть працювати користувачі в різних режимах, то і її вигляд в залежності від цього буде змінюватися.

Якщо робота з програмою реалізується гостем, то йому доступно через меню “Пошук” виконувати пошук паркувальних місць, під час якого він вказує місто парковки, може задати потрібну паркову або задати потрібні її характеристики або характеристики паркувальних місць, включаючи їх кількість, а також потрібний час паркування. У такому випадку за кожним паркувальним майданчиком відображається кількість загалом доступних місць та кількість вільних місць на заданий період.

Окрім того через меню також доступний перехід до перегляду паркувальних майданчиків за обраним містом (рис. 4.1) – пункт “Міста”. У такому випадку потрібно ввести місто і підтвердити введення кнопкою “Відобразити”, в результаті відобразиться мапа зі всіма парковками в місті та окремо список всіх парковок з відображенням середньої оцінки, доступності місць для вантажівок та типом парковки.

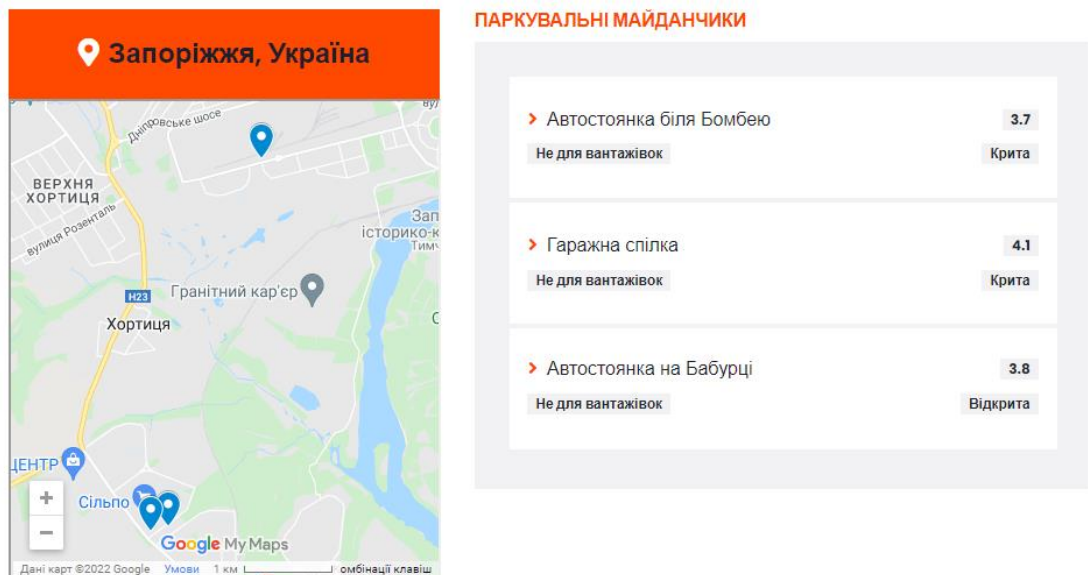
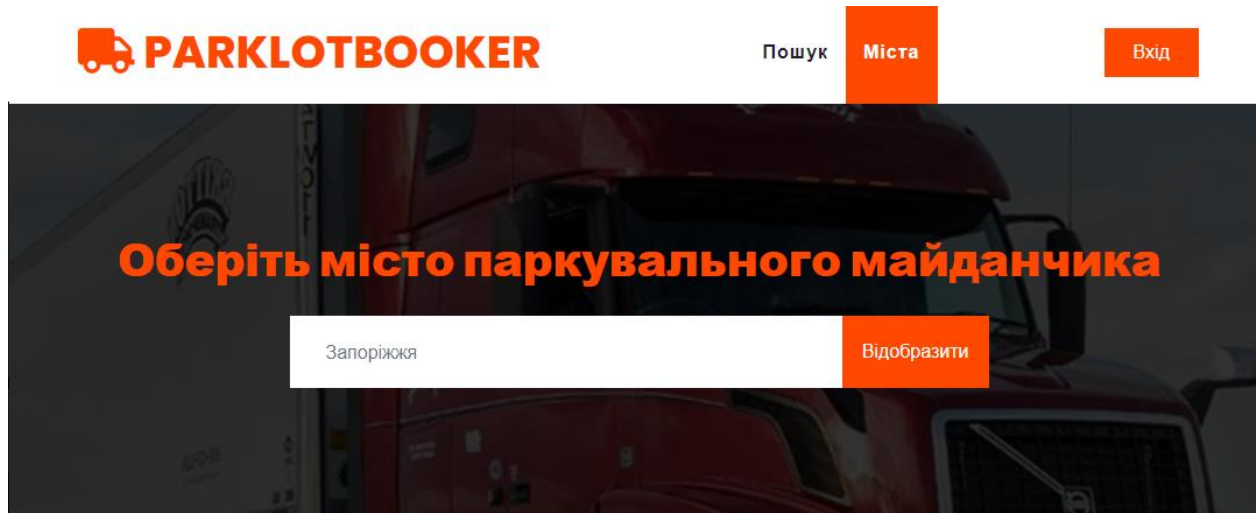


Рисунок 4.1 – Паркувальні майданчики за заданим містом

Гість може також авторизуватися через перехід за кнопкою “Вхід”. Після авторизації, якщо авторизувався користувач, а не менеджер, то йому

буде доступне меню, що включає додаткові пункти. До таких пунктів належить “Бронювання”, “Автомобілі”, звідки можна вносити власні автомобілі, “Рецензії”, звідки можна переглядати власні рецензії, а також створювати рецензії за паркувальними майданчиками, де було виконано бронювання, “Користувач”. Останній пункт потрібний для зміни даних власного запису користувача.

За результатами пошуку або переглядаючи паркувальні майданчики за заданим містом користувач може перейти до конкретного паркувального майданчика (рис. 4.2).

PARKLOTBOOKER Пошук Міста Бронювання Автомобілі Рецензії Користувач Вийти

Деталізація паркувального майданчика

ПРО ПАРКУВАЛЬНИЙ МАЙДАНЧИК

Паркінг на околицях (Львів, Україна)

Паркінг на околицях Львова, в Сокольниках. Величезний простір, на якому кожен зможе залишити автомобіль будь-якого розміру для прогулянок або по справах у місті. Можливі довготривалі, короткотривалі (годинні) паркування. До того ж доступний додатковий сервіс.

4.8 Оцінка

10 Відгуків

90 грн. мінімум

Початок

15.06.2022

17:00

Завершення

15.06.2022

20:00

Шукати

Готівка, картки, безконтактний

Відкрита парковка

Місця для вантажівок

Транспорт доступний

Туалети доступні

Душ немає

Інтернет відсутній

Кав'ярня є

Рисунок 4.2 – Перегляд паркувального майданчика

Тоді користувачу доступна інформація про середню оцінку майданчика, про кількість відгуків, за якими така оцінка була виставлена, про основні параметри майданчиків (рис. 4.3), схема паркувального майданчика та окремо доступна форма для визначення періоду паркування.

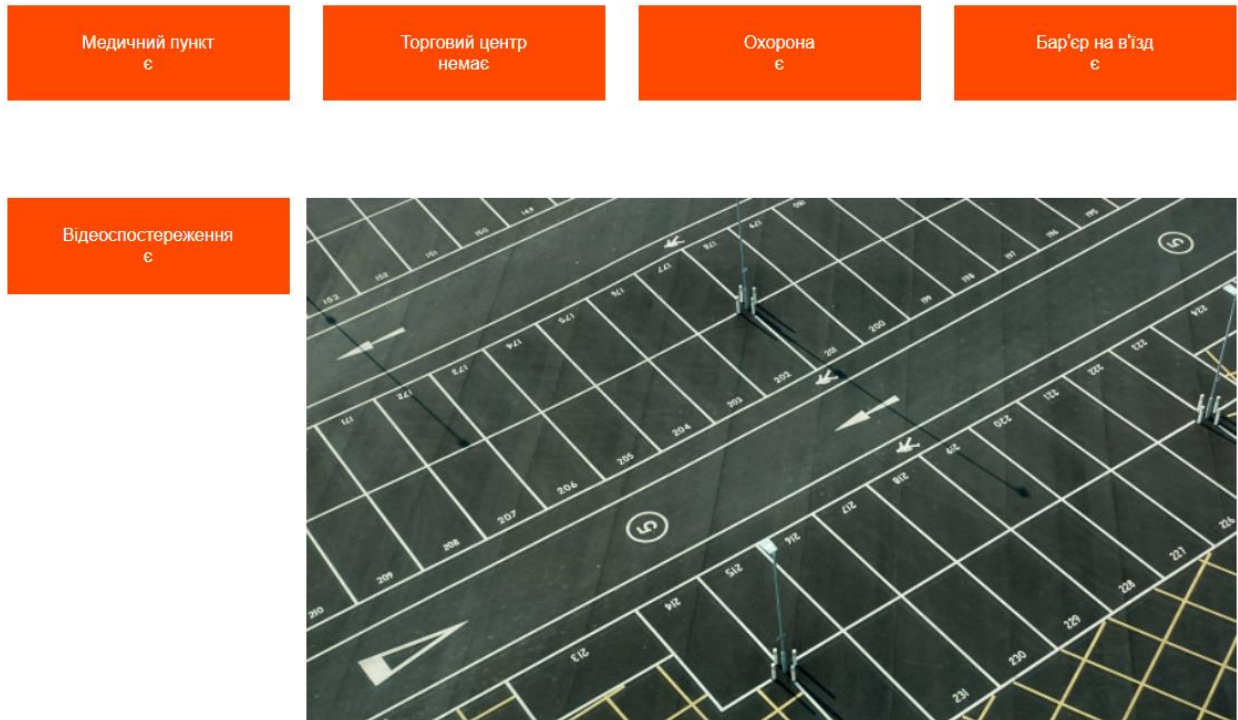


Рисунок 4.3 – Друга частина сторінки паркувального майданчика

Нижче на сторінці також розташовується мапа, на якій можна переглянути адресу паркувального майданчика, типи паркувальних місць та доступна за ними кількість (рис. 4.4). Якщо бронювати місця на такому паркувальному майданчику не можна, то кнопка “Забронювати” не відображається. Якщо натиснути на таку кнопку, то відкриється сторінка бронювання, де можна обрати конкретне місце у відповідності зі схемою паркувального майданчика та обраним типом паркувального місця.

Переглянути список виконаних власних бронювань можна з меню “Бронювання”. Звідси доступний список бронювань (рис. 4.5), а також можна, натискаючи на номер бронювання, перейти до повної інформації

бронювання, а також скасувати бронювання, якщо автомобіль ще не було запарковано.

Доступні класи місць

€90

Легкові клас С

Без електрики

100 місць

Доступно 99 місць

Забронювати

€105

Легкові клас F

Без електрики

10 місць

Доступно 10 місць

Забронювати

€150

Туристичний автобус

Без електрики

5 місць

Доступно 5 місць

Забронювати

Рисунок 4.4 – Перегляд доступних для бронювання місць на паркувальному майданчику

Пошук

Міста

Бронювання

Автомобілі

Рецензії

Користувач

Вийти

Ваші бронювання

Забронювати

Номер і паркінг	Початок і завершення	Стан і місце	Управління
ID 173 Центральний паркінг, Харків	27.08.2022 11:00 – 27.08.2022 12:45	створено П2 місце 5	скасувати
ID 171 Міст паркінг, Дніпро	29.08.2022 18:00 – 29.08.2022 20:00	створено П1 місце 17	скасувати
ID 155 Міст паркінг, Дніпро	26.08.2022 17:00 – 27.08.2022 19:00	запарковано П1 місце 17	

Рисунок 4.5 – Перегляд власних бронювань паркування

Менеджер має меню, що включає “Паркінги” для керування паркінгами, за які відповідає менеджер, “Бронювання”, де можна керувати бронювання за доступними менеджерами паркувальними майданчиками (рис. 4.6).

Паркінги
Бронювання
Тарифи
Користувачі
Вийти

Доступні паркувальні майданчики

Центральний паркінг, Харків
Відобразити

ID бронювання
Відкрити

Бронювання за паркувальним майданчиком
Забронювати

Номер і час	Стан і місце	Сплата	Управління
ID 173 27.08.2022 11:00 – 27.08.2022 12:45	створено поверх 2 місце 5	сплачено: 0 грн. сумарно: 80 грн.	паркувати скасувати редагувати
ID 172 27.08.2022 10:00 – 27.08.2022 20:00	створено поверх 1 місце 7	сплачено: 0 грн. сумарно: 350 грн.	паркувати скасувати редагувати
ID 170 26.08.2022 22:00 – 27.08.2022 12:00	завершено поверх 1 місце 4	сплачено: 420 грн. сумарно: 420 грн.	редагувати
ID 169 26.08.2022 18:00 – 27.08.2022 18:00	запарковано поверх 2 місце 2	сплачено: 600 грн. сумарно: 600 грн.	звільнити редагувати

Рисунок 4.6 – Управління бронюваннями менеджером

За кожним бронюванням менеджеру доступний розширений набір інструментів управління в залежності від поточного стану бронювання.

Меню “Тарифи” дозволяє визначати тарифи на паркування на паркувальних майданчиках, за які відповідає менеджер, а меню “Користувачі” дозволяє управляти доступом користувачів.

4.4 Тестування програми

Результати виконаного тестування за перевіркою функціональних вимог до програми представлено в табл. 4.1.

Таблиця 4.1 – Чеклист тестування програми

Перевірка	Результат
Пошук паркувального місця за параметрами	Пройдено
Відтворення паркувальних майданчиків за заданим містом	Пройдено
Відтворення доступних на заданий момент паркувальних місць у місті	Пройдено
Відтворення доступних на заданий момент паркувальних місць за паркувальним майданчиком	Пройдено
Уточнення інформації про паркувальний майданчик	Пройдено
Бронювання паркувального місця	Пройдено
Відтворення списку всіх бронювань користувача за автомобілем	Пройдено
Скасування бронювання	Пройдено
Рецензування паркувального майданчика за реалізованим бронюванням	Пройдено
Управління бронюваннями за паркувальним майданчиком	Пройдено
Автоматизоване скасування бронювання	Пройдено
Внесення власного автомобіля	Пройдено

Продовження таблиці 4.1

Перевірка	Результат
Внесення паркувальних місць багатомісного паркувального майданчика	Пройдено
Внесення індивідуальних паркувальних місць	Пройдено
Управління доступністю паркувального місця	Пройдено
Управління списком менеджерів паркувального майданчика	Пройдено
Управління доступом користувачів	Пройдено
Автоматизоване блокування акаунту користувача	Пройдено
Реєстрація користувачів і менеджерів	Пройдено
Авторизація користувачів і менеджерів	Пройдено

4.5 Висновки за розділом 4

Програмне забезпечення повністю реалізовано, а доступні основні режими роботи описані в процесі експлуатації з підтвердженням реалізації під час тестування програми.

ВИСНОВКИ

Результатом роботи є розроблена програма, побудована на основі використання багатошарової архітектури, що сприяє її подальшій підтримці з можливістю інтеграції пристроїв відслідковування автомобілів на паркувальних місцях, що потенційно додатково розширить застосування програми. Обрано трирівневу архітектуру для побудови програми, що складається з шару доступу до даних, шару бізнес-логіки та презентаційного шару.

Порівняно мови програмування C# і Ruby для реалізації програмного забезпечення бронювання паркувальних місць, в результаті вибрано мову C#, за допомогою якої, а також фреймворків ASP.NET Core і Entity Framework Core, системи керування базами даних Microsoft SQL Server розроблено програму.

Виділення функцій програми дозволило виконати моделювання сценаріїв роботи програми та на основі цього забезпечити проєктування бази даних. На основі вибраної архітектури програми визначено структуру програмного забезпечення.

Розроблений вебзастосунок дозволяє виконувати пошук потрібного паркувального місця, бронювання паркувального місця, внесення паркувальних місць багатомісних і індивідуальних паркувальних майданчиків, відтворення доступних на заданий момент паркувальних місць у місті або за паркувальним майданчиком, управління бронюваннями за паркувальним майданчиком, рецензування паркувального майданчика за реалізованим бронюванням.

Розроблена програма призначена для організації пошуку паркувальних майданчиків у місті, для бронювання паркувальних місць на цих майданчиках з приділенням особливої уваги приватним індивідуальним паркувальним місцям і забезпеченням інструментів внесення таких паркувальних місць їх власниками до програми та для надання інструментів

для потенційного широкого доступу до засобів бронювання для багатомісних паркувальних майданчиків.

Виконання визначених функцій програми підтверджено проведеним тестуванням програми.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Sisson, P. Cities' parking problem won't be solved with more parking – City Monitor [Electronic resource] / P. Sisson. – Access mode : <https://citymonitor.ai/transport/cities-have-a-parking-problem-more-parking-is-not-the-solution>.
2. Alsoy, H. Types of Cars & Car Segments [Electronic resource] : What Does A–B–C–D–E Car Segments Mean? / H. Alsoy. – Access mode : <https://carwarninglights.net/car-guide/car-segments/>.
3. How Long Is a Car? The Average Car Length Depending On Body Types [Electronic resource]. – Access mode : <https://www.motorverso.com/how-long-is-a-car/>.
4. Belt, J. Will a Motorcycle Fit in a Cargo Van? [Electronic resource] / J. Belt. – Access mode : <https://vandimensions.com/info/will-a-motorcycle-fit-in-a-cargo-van>.
5. The Big Outdoorsy guide to RV classes [Electronic resource]. – Access mode : <https://www.outdoorsy.com/blog/big-guide-to-rv-classes>.
6. Transport Dimensions & Drawings [Electronic resource]. – Access mode : <https://www.dimensions.com/classifications/transport>.
7. Potential of Longer Combination Vehicles in Texas [Electronic resource] / C. M. Walton, J. Prozzi, A. Cruz-Ross, K. Kockelman. – Access mode : https://www.researchgate.net/publication/279513527_Potential_of_Longer_Combination_Vehicles_in_Texas_12.
8. Online Parking Reservations. Parclick [Electronic resource]. – Access mode : <https://parclick.com/>.
9. Home – Truck Parking Europe [Electronic resource]. – Access mode : <https://www.truckparkingeurope.com/>.
10. Albahari, J. C# 10 in a Nutshell [Text] : The Definitive Reference / J. Albahari. – Massachusetts : O'Reilly Media, 2022. – 1058 p.
11. Hartl, M. Ruby on Rails Tutorial [Text] / M. Hartl. – Massachusetts :

Addison-Wesley Professional, 2020. – 848 p.

12. N-tier architecture style [Electronic resource]. – Access mode : <https://docs.microsoft.com/en-us/azure/architecture/guide/architecture-styles/n-tier>.

13. Mittal, A. Understanding Multilayered Architecture in .NET [Electronic resource] / A. Mittal. – Access mode : <https://www.c-sharpcorner.com/UploadFile/1492b1/understanding-multilayered-architecture-in-net/>.

ДОДАТОК А
Технічне завдання

Вступ

Пошук паркувальних місць є важливою задачею для автомобілістів, з якою дуже часто зіштовхуються люди під час подорожей, відряджень у незнайомих містах. Однак у власному великому місті така проблема також виникає постійно, при чому це може посилюватися нестачею паркувальних місць, а тоді важливим стає попереднє бронювання паркувальних місць.

А.1 Підстави для розробки

Дипломна кваліфікаційна робота виконується за темою «Програмне забезпечення бронювання паркувальних місць» у відповідності з наказом № 102 від 27 квітня 2022 р. за Національним університетом «Запорізька політехніка».

А.2 Призначення розробки

Програма призначена для організації пошуку паркувальних майданчиків у місті, для бронювання паркувальних місць на цих майданчиках з приділенням особливої уваги приватним індивідуальним паркувальним місцям і забезпеченням інструментів внесення таких паркувальних місць їх власниками до програми та для надання інструментів для потенційного широкого доступу до засобів бронювання для багатомісних паркувальних майданчиків.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Від набору функціональних характеристик вимагається виконання наступного:

- бронювання паркувального місця;
- скасування бронювання;
- пошук потрібного паркувального місця;
- відтворення паркувальних майданчиків за заданим містом;
- відтворення списку всіх бронювань користувача за автомобілем;
- внесення власного автомобіля;
- рецензування паркувального майданчика за реалізованим бронюванням;
- внесення паркувальних місць багатомісних і індивідуальних паркувальних майданчиків;
- управління списком менеджерів паркувального майданчика;
- управління доступністю паркувального місця;
- управління бронюваннями за паркувальним майданчиком;
- управління доступом користувачів;
- автоматизоване скасування бронювання;
- автоматизоване блокування акаунту користувача;
- відтворення доступних на заданий момент паркувальних місць у місті або за паркувальним майданчиком;
- уточнення інформації про паркувальний майданчик;
- реєстрація і авторизація користувачів і менеджерів.

А.3.2 Умови експлуатації

Для експлуатації програмного забезпечення потрібно забезпечити окремий сервер, на якому буде запущено вебсервер і сервер бази даних для постійного доступу користувачів до сервісу бронювання паркувальних місць.

А.3.3 Вимоги до складу та параметрів технічних засобів

Для забезпечення підтримки роботи вебсервера та сервера бази даних

необхідно надати апаратне забезпечення, що включає мінімально процесор, що має не менше ніж 4 ядра з тактовою частотою не менше 2,1 ГГц, підтриманий оперативною пам'яттю обсягом не менше 8 Гб та жорстким диском з доступними 10 Гб простору на ньому.

А.4 Порядок контролю та приймання

Для контролю виконання дипломної кваліфікаційної роботи повинен використовуватися відповідний календарний план, який затверджено викладачем у завданні на роботу. Приймання роботи відбувається на основі проведення процедури захисту роботи.

ДОДАТОК Б
Текст програми

Б.1 Текст файла Parking.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Parking
    {
        public int Id //номер паркінга
        { get; set; }
        public string Name //назва
        { get; set; }
        public int CityId //номер міста
        { get; set; }
        public string Address //повна адреса
        { get; set; }
        public double Latitude //широта
        { get; set; }
        public double Longitude //довгота
        { get; set; }
        public string Description //опис
        { get; set; }
        public bool Toilet //наявність туалетів
        { get; set; }
        public bool Shower //наявність душу
        { get; set; }
        public bool Internet //наявність інтернету
        { get; set; }
        public bool Cafe //наявність кав'ярні
        { get; set; }
        public bool Transport //наявність громадського
транспорту від самого паркінгу
        { get; set; }
        public float Averagerpoint //середня оцінка
        { get; set; }
        public byte ParkingType //тип паркінга
        { get; set; }
        public string Scheme //схема паркування
        { get; set; }
        public bool Trucks //наявність місць для вантажівок
        { get; set; }
        public byte Payment //метод оплати
        { get; set; }
        public bool Medical //медичний пункт
        { get; set; }
        public bool Shop //магазин
    }
}

```

```

        { get; set; }
        public bool Guard //охорона
        { get; set; }
        public bool Barrier //бар'єр для в'їзду
        { get; set; }
        public bool Camera //наявність камер
відеоспостереження
        { get; set; }
        public virtual ICollection<Lotttype> Lottypes //список
типів місць на паркінгу
        { get; set; }
        public virtual ICollection<Review> Reviews //список
відгуків
        { get; set; }
        public virtual ICollection<ParkingManager>
ParkingManagers //список менеджерів паркінгу
        { get; set; }
        public virtual City City //місто
        { get; set; }
    }
}

```

Б.2 Текст файла Lotttype.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Lotttype
    {
        public int Id //номер типу паркомісця
        { get; set; }
        public string Name //назва
        { get; set; }
        public short Width //ширина
        { get; set; }
        public short Height //висота
        { get; set; }
        public short Length //довжина
        { get; set; }
        public bool Inside //в приміщенні або ззовні
        { get; set; }
        public int CartypeId //номер класу машини
        { get; set; }
        public int ParkingId //паркінг
        { get; set; }
    }
}

```

```

        public string Description //опис
        { get; set; }
        public bool Bookable //чи можливо бронювати
        { get; set; }
        public bool Electricity //електрозарядка
        { get; set; }
        public virtual ICollection<Lot> Lots //список місць
цього типу
        { get; set; }
        public virtual ICollection<Pricesystem> Pricesystems
//список стандартних цін на бронювання
        { get; set; }
        public virtual ICollection<Offer> Offers //список
спеціальних цін на бронювання
        { get; set; }
        public virtual Cartype Cartype //клас машини
        { get; set; }
        public virtual Parking Parking //паркінг
        { get; set; }
    }
}

```

Б.3 Текст файла Lot.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Lot
    {
        public int Id //номер паркомісця
        { get; set; }
        public byte Floor //поверх паркінгу
        { get; set; }
        public byte Position //місце на поверсі
        { get; set; }
        public bool Activeness //активність
        { get; set; }
        public DateTime ActiveFrom //активне з
        { get; set; }
        public DateTime NotActiveFrom //активне по
        { get; set; }
        public byte Special //звичайне або спеціальне місце
        { get; set; }
        public int LottypeId //номер типу паркомісця
        { get; set; }
    }
}

```

```

        public virtual ICollection<Reservation> Reservations
//бронювання цього паркомісця
        { get; set; }
        public virtual Lottype Lottype //тип паркомісця
        { get; set; }
    }
}

```

Б.4 Текст файла Review.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Review
    {
        public int Id //номер рецензії
        { get; set; }
        public byte Point //бал
        { get; set; }
        public string Reviewing //текст рецензії
        { get; set; }
        public int UserId //номер автора
        { get; set; }
        public int ParkingId //номер парковки
        { get; set; }
        public DateTime Reviewed //дата створення
        { get; set; }
        public virtual User User //автор
        { get; set; }
        public virtual Parking Parking //парковка
        { get; set; }
    }
}

```

Б.5 Текст файла Reservation.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Reservation
    {

```

```

        public int Id //номер бронювання
        { get; set; }
        public byte State //стан
        { get; set; }
        public float Costs //вартість
        { get; set; }
        public float Payed //сплачена сума
        { get; set; }
        public int UserCarId //номер автомобілю
        { get; set; }
        public int LotId //номер паркомісця
        { get; set; }
        public DateTime Beginperiod //початок бронювання
        { get; set; }
        public DateTime Endperiod //кінець бронювання
        { get; set; }
        public DateTime Reserved //дата бронювання
        { get; set; }
        public bool FreePlace //вільний вибір місця цього
типу
        { get; set; }
        public virtual Lot Lot //паркомісце
        { get; set; }
        public virtual UserCar UserCar //автомобіль
        { get; set; }
    }
}

```

Б.6 Текст файла Cartype.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Cartype
    {
        public int Id //номер класу автомобіля
        { get; set; }
        public string Name //назва
        { get; set; }
        public string Description //опис
        { get; set; }
        public short Width //ширина
        { get; set; }
        public short Height //висота
        { get; set; }
    }
}

```

```

        public short Length //довжина
        { get; set; }
        public virtual ICollection<Lotttype> Lottypes //типи
паркомісць
        { get; set; }
    }
}

```

Б.7 Текст файла Pricesystem.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Pricesystem
    {
        public int Id //номер стандартної ціни на бронювання
        { get; set; }
        public byte Weekday //номер дня тижня
        { get; set; }
        public TimeSpan Starttime //година початку
        { get; set; }
        public TimeSpan Endtime //година завершення
        { get; set; }
        public byte Duration //максимальна тривалість
        { get; set; }
        public float Costs //вартість години
        { get; set; }
        public int LottypeId //номер типу паркомісця
        { get; set; }
        public virtual Lottype Lottype //тип паркомісця
        { get; set; }
    }
}

```

Б.8 Текст файла Offer.cs

```

using System;
using System.Collections.Generic;

namespace ParkLotBooker.Models
{
    public class Offer
    {

```

```

        public int Id //номер спеціальної ціни
        { get; set; }
        public DateTime Starttime //час початку
        { get; set; }
        public DateTime Endtime //час завершення
        { get; set; }
        public byte Duration //максимальна тривалість
        { get; set; }
        public float Costs //вартість години
        { get; set; }
        public int LottypeId //номер типу паркомісця
        { get; set; }
        public virtual Lottype Lottype //тип паркомісця
        { get; set; }
    }
}

```

Б.9 Текст файла ParkingService.cs

```

using Microsoft.EntityFrameworkCore;
using ParkLotBooker.PDO;
using ParkLotBooker.PDO.Models;
using System;
using System.Collections.Generic;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParkLotBooker.Service
{
    public class ParkingService : IParking
    {
        private readonly DataContext _dc;

        public ParkingService (DataContext dc)
        {
            _dc = dc;
        }

        //повертає повний об'єкт паркінга за номером parkid
        public Parking GetParking(int parkid)
        {
            return _dc.Parkings.Where(p => p.Id == parkid)
                .Include(p => p.Name)
                .Include(p => p.Address)
                .Include(p => p.Latitude)

```

```

        .Include(p => p.City)
        .Include(p => p.Longitude)
        .Include(p => p.Description)
        .Include(p => p.Toilet)
        .Include(p => p.Shower)
        .Include(p => p.Internet)
        .Include(p => p.Cafe)
        .Include(p => p.Transport)
        .Include(p => p.Averagepoint)
        .Include(p => p.ParkingType)
        .Include(p => p.Trucks)
        .Include(p => p.Payment)
        .Include(p => p.Medical)
        .Include(p => p.Shop)
        .Include(p => p.Guard)
        .Include(p => p.Barrier)
        .Include(p => p.Camera)
        .Include(p => p.Scheme)
        .First();
    }

    //повертає список всіх паркінгів
    public IEnumerable<Parking> GetParkings()
    {
        return _dc.Parkings
            .Include(p => p.Name)
            .Include(p => p.City)
            .Include(p => p.Averagepoint)
            .Include(p => p.ParkingType)
            .Include(p => p.Trucks);
        .OrderByDescending(p => p.Averagepoint);
    }

    //повертає список паркінгів у місті з номером cityid
    public IEnumerable<Parking> GetParkingsByCity(int
cityid)
    {
        return _dc.Parkings.Where(p => p.CityId ==
cityid)
            .Include(p => p.Name)
            .Include(p => p.Latitude)
            .Include(p => p.Longitude)
            .Include(p => p.Averagepoint)
            .Include(p => p.ParkingType)
            .Include(p => p.Trucks);
    }

```

```

        //повертає список паркінгів, якими може керувати
менеджер man
        public IEnumerable<Parking>
GetParkingsByManager(Manager man)
        {
            return _dc.Parkings.Where(p =>
p.ParkingManagers.Contains(man))
                .Include(p => p.Name)
                .Include(p => p.City)
                .Include(p => p.Latitude)
                .Include(p => p.Longitude)
                .Include(p => p.Averagepoint)
                .Include(p => p.ParkingType)
                .Include(p => p.Trucks);
        }

        //вносить новий паркінг park
        public async Task Add(Parking park)
        {
            _dc.Add(park);
            await _dc.SaveChangesAsync();
        }

        //оновлює паркінг park
        public async Task Update(Parking park)
        {
            _dc.Update(park);
            await _dc.SaveChangesAsync();
        }
    }
}

```

Б.10 Текст файла ReservationService.cs

```

using Microsoft.EntityFrameworkCore;
using ParkLotBooker.PDO;
using ParkLotBooker.PDO.Models;
using System;
using System.Collections.Generic;
using System.Globalization;
using System.Linq;
using System.Text;
using System.Threading.Tasks;

namespace ParkLotBooker.Service

```

```

{
    public class ReservationService : IReservation
    {
        private readonly DataContext _dc;

        public ReservationService (DataContext dc)
        {
            _dc = dc;
        }

        //повертає об'єкт бронювання за номером reserid
        public Reservation GetReservation(int reserid)
        {
            return _dc.Reservations.Where(r => r.Id ==
reserid)

                .Include(r => r.Activeness)
                .Include(r => r.Costs)
                .Include(r => r.UserCar)
                .Include(r => r.Lot)
                .Include(r => r.Beginperiod)
                .Include(r => r.Endperiod)
                .Include(r => r.Reserved)
                .Include(r => r.FreePlace)
                .First();
        }

        //повертає бронювання користувача us
        public IEnumerable<Reservation>
        GetUserReservations(User us)
        {
            return _dc.Reservations.Where(r => r.UserCar.User
== us)

                .Include(r => r.Activeness)
                .Include(r => r.Costs)
                .Include(r => r.Lot)
                .Include(r => r.Beginperiod)
                .Include(r => r.Endperiod)
                .Include(r => r.FreePlace)
                .Include(r => r.Reserved);
        }

        //повертає бронювання за паркувальним майданчиком
        park у стані state
        public IEnumerable<Reservation>
        GetReservationsByParking(Parking park, byte state)
        {
            if (state != -1)
            {

```

```

        return _dc.Reservations.Where(r =>
r.Lot.Lotttype.Parking == park)
            .Where(r => r.State == state)
            .Include(r => r.Costs)
            .Include(r => r.Lot)
            .Include(r => r.Beginperiod)
            .Include(r => r.Endperiod)
            .Include(r => r.Reserved)
            .Include(r => r.FreePlace)
            .OrderByDescending(r => r.Beginperiod);
    }
    else
    {
        return _dc.Reservations.Where(r =>
r.Lot.Lotttype.Parking == park)
            .Where(r => r.State)
            .Include(r => r.Costs)
            .Include(r => r.Lot)
            .Include(r => r.Beginperiod)
            .Include(r => r.Endperiod)
            .Include(r => r.Reserved)
            .Include(r => r.FreePlace)
            .OrderByDescending(r => r.Beginperiod);
    }
}

//вносить нове бронювання res
public async Task Add(Reservation res)
{
    res.Reserved = DateTime.Now;
    _dc.Add(res);
    await _dc.SaveChangesAsync();
}

//оновлює бронювання res
public async Task Update(Reservation res)
{
    _dc.Update(res);
    await _dc.SaveChangesAsync();
}

//змінює стан бронювання з номером resid на state
public async Task ChangeState(int resid, byte state)
{
    Reservation res = GetReservation(resid);
    res.State = state;
    _dc.Update(res);
    await _dc.SaveChangesAsync();
}

```

```

    }

    //автоматично скасовує бронювання
    public async Task AutoCancel()
    {
        foreach (var res in _dc.Reservations.ToList())
        {
            DateTime start = res.Beginperiod;
            //скасовуються усі бронювання, якщо після 15
            хвилин від запланованого часу початку вони не були використані
            if (res.State == 0 && (start.AddMinutes(15)
            <= DateTime.Now))
            {
                res.State = 4;
                _dc.Update(res);
            }
        }
        await _dc.SaveChangesAsync();
    }
}

```

Б.11 Текст файла ParkingController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ParkLotBooker.PDO.Models.City;
using ParkLotBooker.PDO.Models.Offer;
using ParkLotBooker.PDO.Models.Parking;
using ParkLotBooker.PDO.Models.Pricesystem;
using ParkLotBooker.PDO.Models.Reservation;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace ParkLotBooker.Controllers
{
    public class ParkingController : Controller
    {
        private readonly IParking _parkingService;
        private readonly IReservation _reservationService;
        private readonly IOffer _offerService;
        private readonly IPricesystem _pricesystemService;
        private readonly ICity _cityService;
    }
}

```

```

        private readonly IApplicationUser
        _applicationService;
        private static UserManager<ApplicationUser>
        _userManager;
        private readonly IManager _managerService;
        IHostingEnvironment _henv;

        public LotController (IParking parkingService,
        IReservation reservationService, IOffer offerService, IPricesystem
        pricesystemService, ICity cityService, IHostingEnvironment henv,
        IApplicationUser applicationService, UserManager<ApplicationUser>
        userManager, IManager managerService)
        {
            _parkingService = parkingService;
            _reservationService = reservationService;
            _offerService = offerService;
            _pricesystemService = pricesystemService;
            _cityService = cityService;
            _henv = henv;
            _applicationService = applicationService;
            _userManager = userManager;
            _managerService = managerService;
        }

        //заповнення даними сторінки про парковку з номером
        parkid з визначенням кількості вільних місць на дату day.month.year
        на час hour:minute
        public async Task<IActionResult> Details(int? parkid,
        int? day, int? month, int? year, int? hour, int? minute)
        {
            var park = _parkingService.GetParking (parkid);
            ViewData["parking"] = park;
            //якщо часові параметри не задано, то
            використовується поточний час
            var DateTime dtnow = DateTime.Now;
            if (year != null)
            {
                dtnow = DateTime (year, month, day, hour,
            minute);
            }
            //інформація про типи паркувальних місць на
            парковці
            //0 - номер типу
            //1 - загальна кількість місць даного типу
            //2 - кількість вільних місць даного типу на
            заданий момент часу
            var lottypes = park.Lottypes;
            int[, ] parklots = new int[3, lottypes.Count()];

```

```

        int i = 0;

        foreach(var lt in lottypes)
        {
            parklots[i,0] = lt.Id;
            parklots[i,1] = lt.Lots.Count();
            int resplaces = 0;
            ls = lt.Lots;
            foreach(var l in ls)
            {
                resplaces = resplaces +
                _reservationService.CheckLot(l, dtnow);
            }
            parklots[i,2] = resplaces;
            i = i + 1;
        }

        ViewData["lottypes"] = parklots;
        //доступні спеціальні цінові пропозиції за
        місцями паркінгу на заданий момент часу
        ViewData["offers"] =
        _offerService.GetOffersByLots(lottypes, dtnow);
        //доступні стандартні цінові пропозиції за
        місцями паркінгу на заданий момент часу
        ViewData["pricesystems"] =
        _pricesystemService.GetPriceSystemsByLots(lottypes, dtnow);

        return View();
    }

    //GET-внесення нового паркінгу
    public IActionResult Create()
    {
        ViewData["CityId"] = new
        SelectList(_cityService.GetCities(), "Id", "Name");
        return View();
    }

    //POST-внесення нового паркінгу з використанням схеми
    паркінгу на основі файлу зображення schemepath
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create([Bind("Id,
    Name, Address, Latitude, Longitude, Description, Toilet, Shower,
    Internet, Cafe, Transport, ParkingType, Trucks, Payment, Medical,
    Shop, Guard, Barrier, Camera, CityId")] Parking parking, IFormFile
    schemepath)
    {

```

```

        if (ModelState.IsValid)
        {
            try
            {
                using (var schemetoload = new
FileStream(_henv.WebRootPath + "\\Image\\" + schemepath.FileName,
FileStream.Create))
                {
                    schemepath.CopyTo(schemetoload);
                }
                parking.Scheme = "/Image/" +
schemepath.FileName;

                parking.Averagepoint = 0;
                _parkingService.Add(parking);
                return RedirectToAction(nameof(Index));
            }
            catch (Exception ex)
            {
                return View();
            }
        }
        ViewData["CityId"] = new
SelectList(_cityService.GetCities(), "Id", "Name", parking.CityId);
        return View(parking);
    }

    //GET-редагування паркінгу з номером parkid
    public async Task<IActionResult> Edit(int? parkid)
    {
        if (parkid == null)
        {
            return NotFound();
        }

        var parking = _parkingService.GetParking
(parkid);

        if (parking == null)
        {
            return NotFound();
        }
        ViewData["CityId"] = new
SelectList(_cityService.GetCities(), "Id", "Name", parking.CityId);
        return View(parking);
    }

    //POST-редагування паркінгу з номером parkid з
використанням схеми паркінгу на основі файлу зображення schemepath
[HttpPost]

```

```

[ValidateAntiForgeryToken]
public async Task<IActionResult> Edit(int parkid,
[Bind("Id, Name, Address, Latitude, Longitude, Description, Toilet,
Shower, Internet, Cafe, Transport, ParkingType, Trucks, Payment,
Medical, Shop, Guard, Barrier, Camera, CityId")] Parking parking,
IFormFile schemepath)
{
    if (parkid != parking.Id)
    {
        return NotFound();
    }

    if (ModelState.IsValid)
    {
        try
        {
            if (schemepath != null)
            {
                using (var schemetoload = new
FileStream(_henv.WebRootPath + "\\Image\\" + schemepath.FileName,
FileMode.Create))
                {
                    schemepath.CopyTo(schemetoload);
                }
                parking.Scheme = "/Image/" +
schemepath.FileName;
            }

            _parkingService.Update(parking);
            return RedirectToAction("Index");
        }
        catch (DbUpdateConcurrencyException)
        {
            return NotFound();
        }
    }
    ViewData["CityId"] = new
SelectList(_cityService.GetCities(), "Id", "Name", parking.CityId);
    return View(parking);
}

//заповнення даними сторінки парковки в місті
public IActionResult CityParkings(int cityid)
{
    var parks = _parkingService.GetParkingsByCity
(cityid);

    ViewData["parkings"] = parks;
}

```

```

        return View();
    }

    //заповнення даними сторінки про паркінги, з якими
може працювати менеджер, який авторизувався в програмі
    public IActionResult ManagerParkings()
    {
        var who = _userManager.GetUserId(User);
        var man = _managerService.GetManager(who);

        var parks = _parkingService.GetParkingsByManager
(man);

        ViewData["parkings"] = parks;

        return View();
    }

    //заповнення даними стандартної сторінки, яка містить
список всіх паркінгів, що повертається у випадку успішного
редагування або створення паркінгу
    public IActionResult Index()
    {
        var parks = _parkingService.GetParkings();
        ViewData["parkings"] = parks;

        return View();
    }
}

```

Б.12 Текст файла ReservationController.cs

```

using System;
using System.Collections;
using System.Collections.Generic;
using System.Linq;
using System.Threading.Tasks;
using ParkLotBooker.Models.Reservation;
using Microsoft.AspNetCore.Identity;
using Microsoft.AspNetCore.Mvc;

namespace ParkLotBooker.Controllers
{
    public class ReservationController : Controller
    {
        private readonly IReservation _reservationService;
        private readonly ICar _carService;
    }
}

```

```

        private readonly ILot _lotService;
        private readonly IOffer _offerService;
        private readonly IPricesystem _pricesystemService;
        private readonly IApplicationUser
_applicationService;
        private static UserManager<ApplicationUser>
_userManager;
        private readonly IManager _managerService;

        public ReservationController (IReservation
reservationService, ICar carService, ILot lotService, IOffer
_offerService, IPricesystem _pricesystemService, IApplicationUser
applicationService, UserManager<ApplicationUser> userManager,
IManager managerService)
        {
            _reservationService = reservationService;
            _carService = carService;
            _lotService = lotService;
            _offerService = offerService;
            _pricesystemService = pricesystemService;
            _applicationService = applicationService;
            _userManager = userManager;
            _managerService = managerService;
        }

        //заповнення даними сторінки про бронювання з номером
reserid
        public async Task<IActionResult> Details(int reserid)
        {
            var reserv = _reservationService.GetReservation
(reserid);
            ViewData["reservation"] = reserv;

            return View();
        }

        //GET-створення нового бронювання на паркінгу з
номером parkid
        public IActionResult Create(int parkid)
        {
            var who = _userManager.GetUserId(User);
            var cars = _carService.GetUserCars(who);
            ViewData["UserCarId"] = new SelectList(cars,
"Id", "Registration");
            var lots = _lotService.GetFreeLots(parkid);
            ViewData["LotId"] = new SelectList(lots, "Id",
"Position");

            return View();
        }

```

```

    }

    //POST-створення нового бронювання
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Create([Bind("Id,
    Payed, UserCarId, LotId, FreePlace")] Reservarion reserv, int
    Beginperiod_year, int Beginperiod_month, int Beginperiod_day, int
    Beginperiod_hour, int Beginperiod_minute, int Endperiod_year, int
    Endperiod_month, int Endperiod_day, int Endperiod_hour, int
    Endperiod_minute, int parkid)
    {
        DateTime beginperiod_dt =
        DateTime(Beginperiod_year, Beginperiod_month, Beginperiod_day,
        Beginperiod, Beginperiod_minute);
        DateTime endperiod_dt = DateTime(Endperiod_year,
        Endperiod_month, Endperiod_day, Endperiod_hour, Endperiod_minute);
        if (ModelState.IsValid)
        {
            try
            {
                reserv.State = 0;
                reserv.Reserved = DateTime.Now;
                var cos =
                _offerService.GetOfferCosts(reserv.LotId, beginperiod_dt,
                endperiod_dt);

                if (cos >= 0)
                {
                    reserv.Costs = cos;
                }
                else
                {
                    cos =
                    _pricesystemService.GetPriceSystemCosts(reserv.LotId,
                    beginperiod_dt, endperiod_dt);
                    reserv.Costs = cos;
                }
                _reservationService.Add(reserv);
                return
                RedirectToAction(nameof(OwnReservations));
            }
            catch (Exception ex)
            {
                return View();
            }
        }
        var who = _userManager.GetUserId(User);
        var cars = _carService.GetUserCars(who);
    }

```

```

        ViewData["UserCarId"] = new SelectList(cars,
        "Id", "Registration", reserv.UserCarId);
        var lots = _lotService.GetFreeLots(parkid);
        ViewData["LotId"] = new SelectList(lots, "Id",
        "Position", reserv.LotId);
        return View(reserv);
    }

    //GET-редагування бронювання з номером reserid
    public async Task<IActionResult> Edit(int? reserid,
    int parkid)
    {
        if (reserid == null)
        {
            return NotFound();
        }
        var reserv = _reservationService.GetReservation
        (reserid);

        if (reserv == null)
        {
            return NotFound();
        }
        var who = _userManager.GetUserId(User);
        var cars = _carService.GetUserCars(who);
        ViewData["UserCarId"] = new SelectList(cars,
        "Id", "Registration", reserv.UserCarId);
        var lots = _lotService.GetFreeLots(parkid);
        ViewData["LotId"] = new SelectList(lots, "Id",
        "Position", reserv.LotId);
        return View(reserv);
    }

    //POST-редагування бронювання з номером reserid
    [HttpPost]
    [ValidateAntiForgeryToken]
    public async Task<IActionResult> Edit(int reserid,
    [Bind("Id, State, Payed, UserCarId, LotId, FreePlace")] Reservarion
    reserv, int Beginperiod_year, int Beginperiod_month, int
    Beginperiod_day, int Beginperiod_hour, int Beginperiod_minute, int
    Endperiod_year, int Endperiod_month, int Endperiod_day, int
    Endperiod_hour, int Endperiod_minute)
    {
        DateTime beginperiod_dt =
        DateTime(Beginperiod_year, Beginperiod_month, Beginperiod_day,
        Beginperiod_hour, Beginperiod_minute);
        DateTime endperiod_dt = DateTime(Endperiod_year,
        Endperiod_month, Endperiod_day, Endperiod_hour, Endperiod_minute);
        if (reserid != reserv.Id)

```

```

        {
            return NotFound();
        }

        if (ModelState.IsValid)
        {
            try
            {
                var cos =
                _offerService.GetOfferCosts(reserv.LotId, beginperiod_dt,
                endperiod_dt);

                if (cos >= 0)
                {
                    reserv.Costs = cos;
                }
                else
                {
                    cos =
                _pricesystemService.GetPriceSystemCosts(reserv.LotId,
                beginperiod_dt, endperiod_dt);
                    reserv.Costs = cos;
                }
                _reservationService.Update(reserv);
                return
                RedirectToAction("OwnReservations");
            }
            catch (DbUpdateConcurrencyException)
            {
                return NotFound();
            }
        }
        var who = _userManager.GetUserId(User);
        var cars = _carService.GetUserCars(who);
        ViewData["UserCarId"] = new SelectList(cars,
        "Id", "Registration", reserv.UserCarId);
        var lots =
        _lotService.GetFreeLots(reserv.Lot.Lottype.ParkingId);
        ViewData["LotId"] = new SelectList(lots, "Id",
        "Position", reserv.LotId);
        return View(reserv);
    }

    //заповнення даними сторінки зі списком власних
    бронювань авторизованого користувача
    public IActionResult OwnReservations()
    {
        var who = _userManager.GetUserId(User);
        if (who == null) return new ApplicationUser();
    }

```

```

        var us = _userManager.FindByIdAsync(who).Result;

        var reservations =
_reservationService.GetUserReservations (us);
        ViewData["reservations"] = reservations;

        return View();
    }

    //визначення початку паркування за бронюванням
reserid

public IActionResult Start(int reserid)
{
    var who = _userManager.GetUserId(User);
    if (who == null) return new ApplicationUser();
    var reserv = _reservationService.GetReservation
(reserid);

    if (_userManager.CheckAccess(who, reserv.Id))
    {
        reserv.State = 2;
        _reservationService.Update(reserv);
        ViewData["reservation"] = reserv;
        return View();
    }
    else
    {
        return NotFound();
    }
}

//скасування бронювання з номером reserid
public IActionResult Cancel(int reserid)
{
    var who = _userManager.GetUserId(User);
    if (who == null) return new ApplicationUser();
    var reserv = _reservationService.GetReservation
(reserid);

    if (_userManager.CheckAccess(who, reserv.Id))
    {
        reserv.State = 1;
        _reservationService.Update(reserv);
        ViewData["reservation"] = reserv;
        return View();
    }
    else
    {
        return NotFound();
    }
}

```

```

    }

    //визначення завершення паркування за бронюванням
    reserid зі сплатою суми payed
    public IActionResult Return(int reserid, float payed)
    {
        var who = _userManager.GetUserId(User);
        if (who == null) return new ApplicationUser();
        var reserv = _reservationService.GetReservation
(reserid);

        if (_userManager.CheckAccess(who, reserv.Id))
        {
            var reserv =
_reservationService.GetReservation (reserid);
            reserv.State = 3;
            reserv.Endperiod = DateTime.Now;
            var cos =
_offerService.GetOfferCosts(reserv.LotId, reserv.Beginperiod,
reserv.Endperiod);

            if (cos >= 0)
            {
                reserv.Costs = cos;
            }
            else
            {
                cos =
_pricesystemService.GetPriceSystemCosts(reserv.LotId,
reserv.Beginperiod, reserv.Endperiod);
                reserv.Costs = cos;
            }
            reserv.Payed = payed;
            _reservationService.Update(reserv);
            ViewData["reservation"] = reserv;
            return View();
        }
        else
        {
            return NotFound();
        }
    }

    //заповнення даними сторінки зі списком бронювань за
    паркінгом з номером parkid або для першого паркінгу зі всіх
    доступних для авторизованого менеджера
    public IActionResult ParkingReservations(int? parkid)
    {
        var who = _userManager.GetUserId(User);
        var man = _managerService.GetManager(who);

```

```

var parks = _parkingService.GetParkingsByManager
(man);

if (_managerService.CheckAccess(who, parkid))
{
    if (parkid == null)
    {
        var reservations = _reservationService.
GetReservationsByParking (parks.First(), -1);
        ViewData["reservations"] = reservations;
        ViewData["parks"] = parks;
        return View();
    }
    else
    {
        var reservations = _reservationService.
GetReservationsByParking (parkid, -1);
        ViewData["reservations"] = reservations;
        ViewData["parks"] = parks;
        return View();
    }
}
else
{
    return NotFound();
}
}
}

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
кафедра програмних засобів

**ПРОГРАМНЕ ЗАБЕЗПЕЧЕННЯ БРОНЮВАННЯ
ПАРКУВАЛЬНИХ МІСЦЬ**

Виконав
ст. гр. КНТ-138

В. В. Мороз

Керівник
к.т.н., доц.

В. М. Льовкін

2022

Рисунок В.1 – Слайд № 1

Об'єкт роботи – процеси пошуку і бронювання
паркувальних місць.

Предмет роботи – програмне забезпечення бронювання
паркувальних місць.

Мета роботи – розробити програмне забезпечення
бронювання паркувальних місць для гарантування
розміщення власного автомобіля в умовах обмеженості
наявних місць.

2

Рисунок В.2 – Слайд № 2

Завдання

- аналіз предметної області;
- визначення вимог до програмного забезпечення бронювання паркувальних місць;
- проектування програмного забезпечення бронювання паркувальних місць;
- реалізація і налагодження програмного забезпечення бронювання паркувальних місць.

3

Рисунок В.3 – Слайд № 3

Порівняння програмних аналогів

Критерій	Parclick	Truck Parking Europe	Програма
Пошук паркувальних місць	Так	Так	Так
Бронювання	Так	Так	Так
Оцінювання	Так	Так	Так
Типи транспорту	Всі	Вантажівки	Всі
Внесення власних паркувальних місць	Немає	Немає	Так

4

Рисунок В.4 – Слайд № 4

Вибір мови програмування		
Критерій	C#	Ruby
Фреймворк	ASP.NET	Ruby on Rails
Час обробки коду	Менше	Більше
Стабільність	Висока, в тому числі у великих проєктах	Більша направленість на невеликі проєкти
Швидкість кодування	Краща для більших проєктів	Краща для невеликих проєктів
Кількість документації	Велика	Невелика

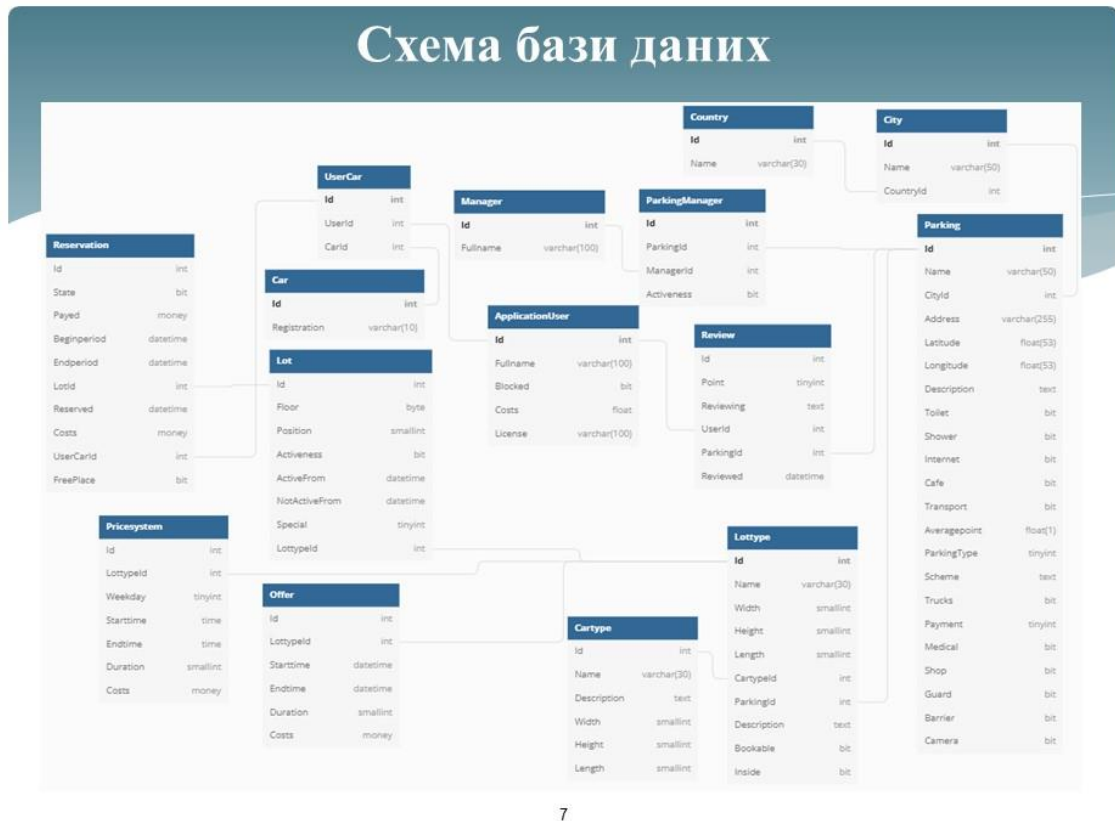
5

Рисунок В.5 – Слайд № 5



6

Рисунок В.6 – Слайд № 6



7

Рисунок В.7 – Слайд № 7



8

Рисунок В.8 – Слайд № 8



Рисунок В.9 – Слайд № 9

Паркувальні майданчики за заданим містом

Пошук

Міста

Вхід

Оберіть місто паркувального майданчика

Запоріжжя

Відобразити

Запоріжжя, Україна

ПАРКУВАЛЬНІ МАЙДАНЧИКИ

Автостоянка біля Бомбею	3.7
Не для вантаків	Крита
Гаражна спілка	4.1
Не для вантаків	Крита
Автостоянка на Бабурці	3.6
Не для вантаків	Відкрита

Рисунок В.10 – Слайд № 10

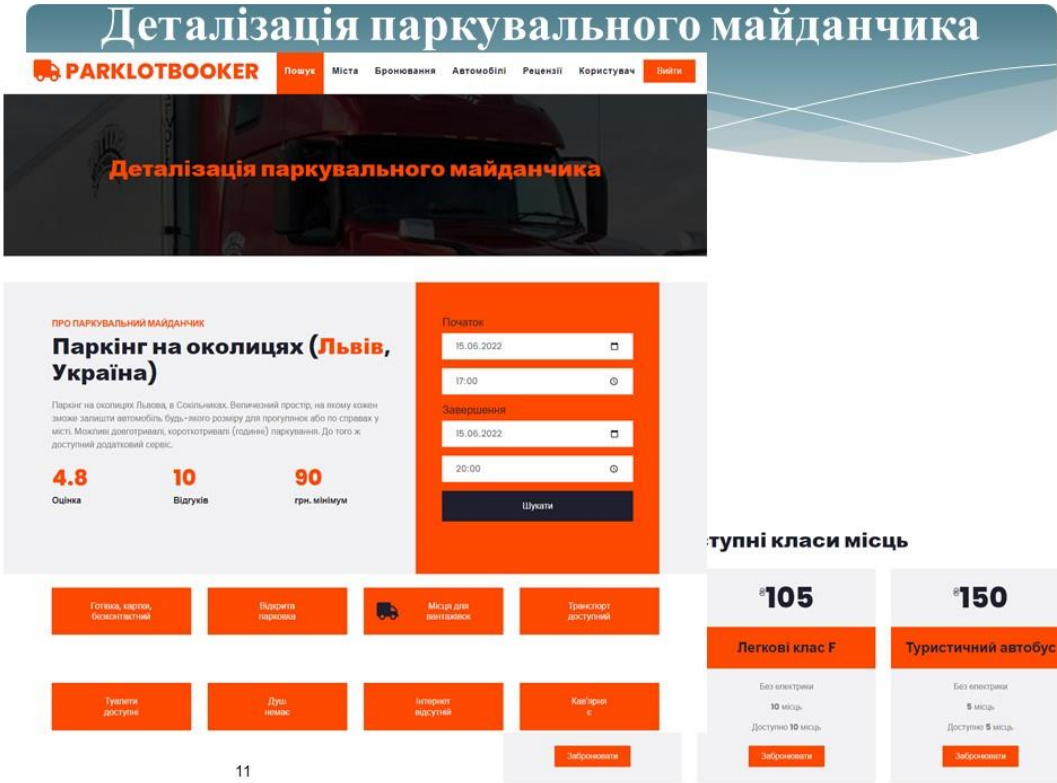


Рисунок В.11 – Слайд № 11

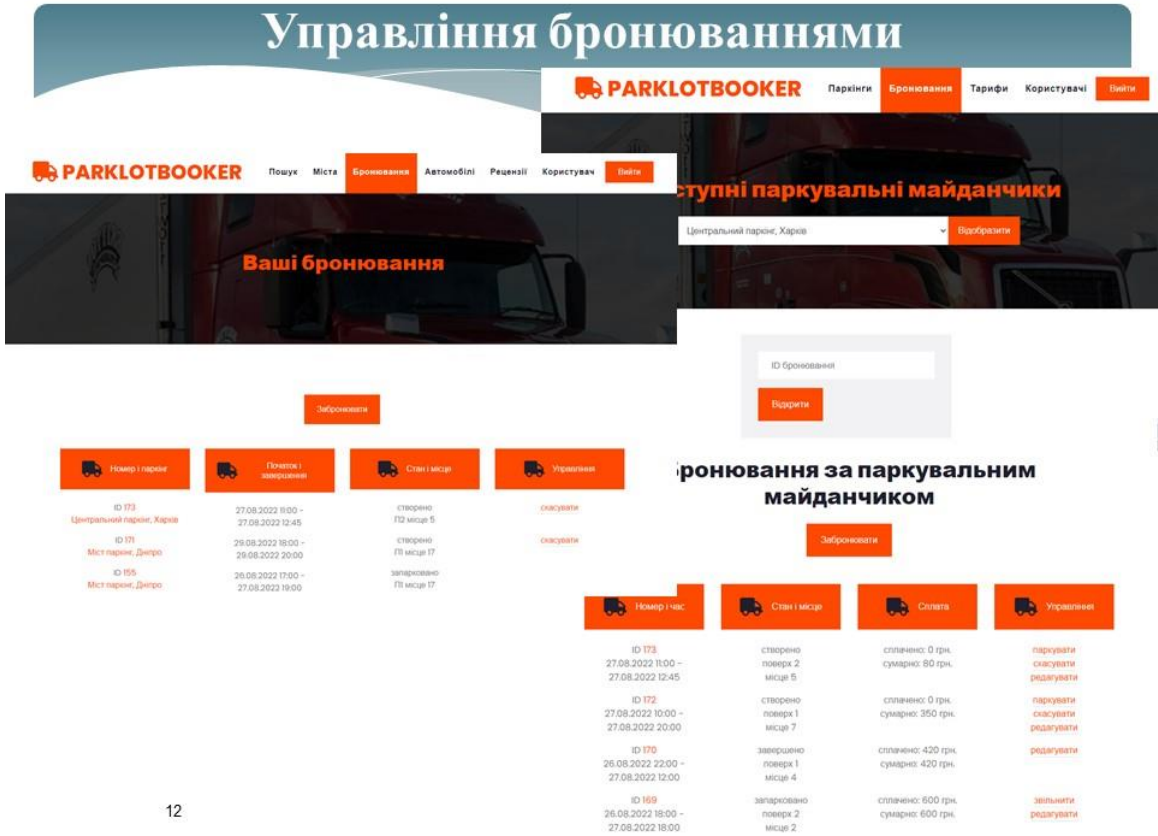


Рисунок В.12 – Слайд № 12

Чеклист тестування програми			
Перевірка	Результат	Перевірка	Результат
Пошук паркувального місця за параметрами	Пройдено	Внесення паркувальних місць багатомісного паркувального майданчика	Пройдено
Відтворення паркувальних майданчиків за заданим містом	Пройдено	Внесення індивідуальних паркувальних місць	Пройдено
Відтворення доступних на заданий момент паркувальних місць у місті	Пройдено	Управління доступністю паркувального місця	Пройдено
Відтворення доступних на заданий момент паркувальних місць за паркувальним майданчиком	Пройдено	Управління списком менеджерів паркувального майданчика	Пройдено
Уточнення інформації про паркувальний майданчик	Пройдено	Управління доступом користувачів	Пройдено
Бронювання паркувального місця	Пройдено	Автоматизоване блокування акаунту користувача	Пройдено
Відтворення списку всіх бронювань користувача за автомобілем	Пройдено	Реєстрація користувачів і менеджерів	Пройдено
Скасування бронювання	Пройдено	Авторизація користувачів і менеджерів	Пройдено
Рецензування паркувального майданчика за реалізованим бронюванням	Пройдено		
Управління бронюваннями за паркувальним майданчиком	Пройдено		
Автоматизоване скасування бронювання	Пройдено		
Внесення власного автомобіля	Пройдено		

13

Рисунок В.13 – Слайд № 13

Висновки

Результатом роботи є розроблена програма, побудована на основі використання багатшарової архітектури, що сприяє її подальшій підтримці з можливістю інтеграції пристроїв відслідковування автомобілів на паркувальних місцях, що потенційно додатково розширить застосування програми. Обрано трирівневу архітектуру для побудови програми, що складається з шару доступу до даних, шару бізнес-логіки та презентаційного шару.

Розроблена програма призначена для організації пошуку паркувальних майданчиків у місті, для бронювання паркувальних місць на цих майданчиках з приділенням особливої уваги приватним індивідуальним паркувальним місцям і забезпеченням інструментів внесення таких паркувальних місць їх власниками до програми та для надання інструментів для потенційного широкого доступу до засобів бронювання для багатомісних паркувальних майданчиків.

Виконання визначених функцій програми підтверджено проведеним тестуванням програми.

14

Рисунок В.14 – Слайд № 14