Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління

Магістерська кваліфікаційна робота
на тему:
«Інтелектуальна система обліку транспортних засобів на паркувальному майданчику»

Виконав: студент 2 курсу, групи 3АКІТР-23м спеціальності 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка

Віктор КОНОТОПЕНКО
Керівник: к.т.н., доцент кафедри КСУ

Олег КОВАЛЮК
«4» грудня 2024 р.

Рецензент: к.т.н., доцент кафедри АІІТ

Костянтин ОВЧИННИКОВ

«4» грудня 2024 р.

Допущено до захисту
Зав. кафедри КСУ

В'ячеслав КОВТУН
«4» грудня 2024 р.

Вінниця ВНТУ – 2024 рік

Вінницький національний технічний університет
Факультет інтелектуальних інформаційних технологій та автоматизації
Кафедра комп'ютерних систем управління
Рівень вищої освіти другий (магістерський)
Галузь знань – 17 – Електроніка, автоматизація та електронні комунікації
Спеціальність – 174 – Автоматизація, комп'ютерно-інтегровані технології та робототехніка
Освітньо - професійна програма – Інформаційні системи і Інтернет речей

ЗАТВЕРДЖУЮ
Зав. кафедри КСУ

В'ячеслав КОВТУН

"14" жовтня 2024 року

ЗАВДАННЯ
НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Конотопенку Віктору Сергійовичу
(прізвище, ім'я, по батькові)

1. Тема роботи Інтелектуальна система обліку транспортних засобів на паркувальному майданчику
керівник роботи Ковалюк Олег Олександрович
затверджені наказом ВНТУ від "17" вересня 2024 року №310
2. Термін подання студентом роботи "1" грудня 2024 року
3. Вихідні дані до роботи:
-організація обліку транспортних засобів на паркувальному майданчику
-загальні принципи оптимізації обліку транспортних засобів на паркувальному майданчику
-потреби та вимоги до інтелектуальної системи обліку транспортних засобів на паркувальному майданчику
- Мобільне паркування [Електронний ресурс]. – Режим доступу: https://ktps.kyiv.ua/services/pay/mobile-parking
4. Зміст текстової частини: вступ, аналіз об'єкта автоматизації, вибір та обґрунтування структури системи, яка проєктується, основні рішення щодо реалізації компонентів системи, розгортання та експлуатація, висновки, перелік посилань, додатки
5. Перелік ілюстративного матеріалу (з точним зазначенням обов'язкових креслень)
Схема реєстрації клієнта, схема реєстрації транспортного засобу, схема бронювання місця для паркування, таблиця основних показники сучасних програмних продуктів для паркування, таблиця порівняння фреймворків веб розробки, Діаграма варіантів використання, Діаграма варіантів «Реєстрація транспортного засобу», Діаграма варіантів «Транспортні засоби», Схема бази даних, Діалогові вікна, Екранні форми

1. Консультанти розділів роботи

- Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	виконання прийняв

2. Дата видачі завдання "14" жовтня 2024 року

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва та зміст етапу	Термін виконання		Примітка
		початок	закінчення	
1	Аналіз предметної області	20.10.24	24.10.24	
2	Огляд сучасних методів розв'язання технічної задачі	25.10.24	27.10.24	
3	Аналіз функцій систем та проєктування архітектури автоматизованої системи	28.10.24	01.11.24	
4	Розробка UML-діаграм системи	02.11.24	03.11.24	
5	Програмна реалізація системи	04.11.24	11.11.24	
6	Тестування програмного забезпечення	12.11.24	14.11.24	
7	Оформлення пояснювальної записки	15.11.24	19.11.24	
8	Захист МКР	11.12.24	11.12.24	

Студент _____ Віктор КОНОТОПЕНКО
(підпис)

Керівник роботи _____ Олег КОВАЛЮК
(підпис)

Анотація

УДК 004.896+625.712.63

Конотопенко В. С. Розробка інтелектуальної системи обліку транспортних засобів на паркувальному майданчику. Магістерська кваліфікаційна робота зі спеціальності 174 - Автоматизація, комп'ютерно-інтегровані технології та робототехніка, освітня програма - Інформаційні системи і Інтернет речей. Вінниця: ВНТУ, 2024. 131с.

На укр. мові. Бібліогр.: 49 назв; рис.: 28; табл.: 10.

Метою даної магістерської роботи є розробка інтелектуальної системи для автоматизації управління процесами паркування транспортних засобів, що дозволяє підвищити ефективність використання паркувальних місць. Основним завданням є створення функціонального програмного продукту для реєстрації, моніторингу та збору статистики про використання паркінгу.

У роботі проведено аналіз сучасних систем управління паркуванням та відстеження транспортних засобів, визначено їхні ключові переваги та недоліки. Розроблено архітектуру системи, що базується на сучасних фреймворках та технологіях, таких як ASP.NET та CMS Umbraco, для забезпечення масштабованості, продуктивності та зручності використання. Система надає користувачам можливість інтерактивного бронювання паркувальних місць та отримання детальної статистики.

Ключові слова: інтелектуальна система, транспортні засоби, паркування, управління паркінгом, ASP.NET, Umbraco.

Abstract

UDC 004.896+625.712.63

Konotopenko V. S. Development of an Intelligent System for Vehicle Accounting at a Parking Lot. Master's qualification thesis in specialty 174 - Automation, Computer-Integrated Technologies, and Robotics, educational program - Information Systems and Internet of Things. Vinnytsia: VNTU, 2024. 131 pages.

In Ukrainian. Bibliography: 49 sources; illustrations: 28; tables: 10.

The aim of this master's thesis is to develop an intelligent system for automating the management of vehicle parking processes, enhancing the efficient use of parking spaces. The primary objective is to create a functional software product for vehicle registration, monitoring, and statistical data collection regarding parking usage.

The thesis includes an analysis of modern parking management and vehicle tracking systems, identifying their key strengths and weaknesses. The system architecture was designed based on contemporary frameworks and technologies such as ASP.NET and the Umbraco CMS to ensure scalability, performance, and user-friendliness. The developed system provides users with features for interactive parking space booking and detailed statistical reporting.

Key words: intelligent system, vehicles, parking, parking management, ASP.NET, Umbraco.

ЗМІСТ

ВСТУП

Актуальність. Автоматизована система моніторингу та збору статистики паркування є сучасним технологічним рішенням, яке дозволяє значно оптимізувати процес управління парковками в торгових центрах, бізнес-комплексах та громадських місцях. З кожним роком кількість автомобілів невпинно зростає, і це підвищує попит на ефективні способи управління паркувальними ресурсами. У торгових центрах, де зосереджено велику кількість магазинів і розважальних закладів, попит на паркувальні місця особливо високий, особливо під час акцій, розпродажів або святкових днів.

Традиційний підхід до управління парковкою часто є ресурсномістким і вимагає значних витрат на оплату праці операторів або охоронців, які займаються обліком і моніторингом автомобілів. Ручний підхід створює значні незручності, особливо у години пік, коли наплив відвідувачів досягає максимуму. Окрім того, відсутність автоматизації унеможливлює збір повноцінної статистики, необхідної для аналізу і планування.

Сучасна автоматизована система має забезпечувати комплексний підхід, включаючи реєстрацію транспортних засобів, моніторинг паркувальних місць у режимі реального часу, інтеграцію з інформаційними табло і зберігання даних для подальшого аналізу. Такі системи дозволяють не лише зекономити час і ресурси, але й покращують досвід користувачів завдяки швидкому пошуку вільних місць.

Об’єктом дослідження є процес моніторингу та збору статистики паркування, яка здатна функціонувати в умовах великого трафіку та забезпечувати облік транспортних засобів у реальному часі.

Предметом дослідження є методика моніторингу, збору та обробки статистичних даних про транспортні засоби та паркувальні місця, а також розробка алгоритмів для оптимізації цих процесів.

Метою дослідження є зменшення часу пошуку вільних місць, підвищення зручності для користувачів та оптимізації управління паркувальними ресурсами.

Для досягнення цієї мети були визначені наступні завдання:

- Провести аналіз існуючих технологій та протоколів, які можуть бути використані для створення системи.
- Розробити адміністративну панель для управління даними операторів, парковок та користувачів.
- Розробити функціонал для реєстрації транспортних засобів та відображення вільних паркувальних місць.
- Забезпечити захист даних і доступ до системи через сучасні методи аутентифікації та шифрування.

Інноваційність.

Розроблено унікальну автоматизовану систему, яка включає реєстрацію клієнтів та транспортних засобів, інтеграцію з базою даних і швидкий доступ до інформації через інтерфейс. Запропоновано оптимізовану структуру бази даних, що дозволяє зберігати і обробляти великі обсяги інформації з високою продуктивністю. Створено інтуїтивні інтерфейси для операторів та користувачів, які значно спрощують процес взаємодії із системою.

Практичною цінністю є

- Використання результатів дослідження для ефективного управління паркувальними майданчиками, що зменшує потребу в людському втручанні.
- Оптимізація процесу паркування, що підвищує зручність для користувачів і скорочує час пошуку паркувального місця.
- Покращення доступності паркувальних ресурсів шляхом використання статистичних даних для управління попитом.
- Впровадження системи, яка забезпечує збір та аналіз даних для подальшого стратегічного планування.

Апробація результатів дослідження: Представлені в роботі результати апробовані в результаті участі в конференції «LIII Всеукраїнська науково-технічна конференція факультету інтелектуальних інформаційних технологій та

автоматизації (2024)». За результатами виконаних досліджень підготовлено та подано тези доповіді. ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ НА ПАРКУВАЛЬНОМУ МАЙДАНЧИКУ [48].

1 АНАЛІЗ ОБ'ЄКТА АВТОМАТИЗАЦІЇ

1.1 Основні поняття

Автостоянка — це спеціально обладнана територія, призначена для тимчасового або довгострокового зберігання транспортних засобів (ТЗ). Управління автостоянкою здійснюється менеджером або директором, які відповідають за бюджет, планування, контроль персоналу та організацію процесу паркування [1].

Вхід, обслуговування та виїзд клієнтів координуються оператором автостоянки. У разі платного паркування оператор відповідає за реєстрацію клієнта (ПІБ, контактний номер) і транспортного засобу (державний номер). Він також перевіряє наявність вільних місць, призначає паркувальну ділянку та узгоджує час використання паркувального місця.

У випадку безкоштовних парковок дані про клієнтів і транспортні засоби не реєструються та не обробляються. Згідно з дослідженнями, тільки 30% парковок мають камери, які фіксують заїзди та виїзди транспортних засобів, однак більшість із них не інтегровані з системами розпізнавання номерних знаків і не надають інформацію про зайнятість паркомісць [3-4].

Процес виїзду з платної парковки передбачає, що клієнт надає документи, які підтверджують право на транспортний засіб, та виконує оплату за використаний час паркування. У вихідні або наприкінці місяця оператор створює звіт про дохід, зайнятість паркомісць, орендний час і статистику за марками та моделями автомобілів. Цей звіт надсилається менеджеру.

Основною метою діяльності автостоянки є забезпечення безпеки автомобілів і майна клієнтів.

Організаційна структура та обов'язки

Менеджер паркування:

- Координує роботу співробітників, приймає рішення щодо найму і звільнення
- Планує бюджет і організовує розвиток паркувальної інфраструктури
- Контролює дебіторську заборгованість та співпрацює з сервісними компаніями
- Надає клієнтам послуги та відповідає за їх якість.
- Оператор/охоронець:
- Реєструє в’їзд і виїзд транспортних засобів
- Веде змінний журнал обліку
- Забезпечує порядок на території парковки
- Створює звіти про зайнятість місць та фінансову діяльність

Користувачами системи є: директор, оператор і клієнт. Оператор, окрім вищезазначених обов’язків, знайомить клієнта з правилами паркування, забезпечує перепусткою та надає інструктаж на випадок евакуації.

Система реєстрації та звітності.

Сучасні автоматизовані системи дозволяють значно спростити процес управління парковкою. Вони забезпечують реєстрацію клієнтів і транспортних засобів, розподіл паркомісць за кодами, ведення бази даних та створення звітів про діяльність парковки.

Щомісячний звіт містить:

- Список зареєстрованих клієнтів і транспортних засобів;
- Час перебування автомобілів на території;
- Загальний дохід за платне паркування;
- Статистичні дані, які допомагають у плануванні.

Рис. 1.1 ілюструє загальну організаційну структуру паркувального комплексу, де менеджер відповідає за стратегічне управління, а оператори — за операційне виконання завдань. Уточнений малюнок може бути створений або знайдений відповідно до вимог.

Рисунок 1.1 – Організаційна структура паркінгу

1.2 Опис предметної області

Аналіз галузі, пов'язаної з управлінням транспортними засобами та паркувальними майданчиками, виявляє ключові бізнес-процеси, спрямовані на оптимізацію та автоматизацію роботи паркінгів. Ці процеси включають:

Формування короткої анкети для клієнтів і транспортних засобів: Анкета є важливим елементом для збору базової інформації про клієнтів і їхні автомобілі. Вона містить ключові дані, такі як ім'я клієнта, контактний номер, державний номер транспортного засобу та інші відомості, що спрощують подальшу обробку інформації[8].

Оформлення та ведення реєстрів: Це передбачає систематизацію інформації про клієнтів, транспортні засоби та операції, що здійснюються на автостоянці. Реєстри забезпечують збереження інформації та доступ до неї для подальшого аналізу[10].

Створення й відображення звітів про реєстрацію: Систематичне створення звітів дозволяє відстежувати динаміку зайнятості паркомісць, кількість клієнтів, фінансові надходження та інші параметри. Це сприяє підвищенню ефективності управління.

Пошук вільних місць для паркування: Цей процес допомагає клієнтам швидко знайти доступні місця на парковці, зменшуючи час очікування і забезпечуючи раціональне використання паркувальних ресурсів[12].

Реєстрація клієнтів

Для реєстрації клієнтів система має забезпечувати збереження їхніх контактних даних на сервері (рис. 1.2). Дані зберігаються у вигляді структурованих записів, які потім використовуються для управління паркувальними місцями, генерації звітів та інформування клієнтів.

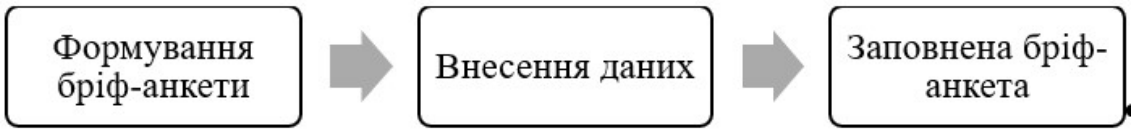


Рисунок 1.2 – Реєстрація клієнта

Рис. 1.2 ілюструє процес реєстрації клієнта. Система автоматично зберігає всі необхідні дані, що мінімізує вплив людського фактора.

Таблиця 1.1 –Формування бріф – анкети

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Реєстрація клієнта
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про клієнта
Вхідні документи	Немає
Вихідна подія	Відповідь про статус реєстрації
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

Таблиця 1.1 демонструє приклад формування анкети клієнта. У ній враховуються основні параметри, що спрощують процес реєстрації та управління.

Таблиця 1.2 –Реєстрація транспортного засобу

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Реєстрація транспортного засобу
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про транспортний засіб
Вхідні документи	Свідоцтво про реєстрацію авто
Вихідна подія	Відповідь про статус реєстрації, присвоєння реєстраційного коду транспортному засобу
Вихідні документи	Паркувальний талон, тимчасова чи постійна перепустка або безконтактні смарт–картки
Клієнт бізнес процесу	Сервер

Реєстри транспортних засобів (рис. 1.3, табл. 1.2) містять таку інформацію:

- Дані про власника транспортного засобу (фізичну або юридичну особу);

- Державний реєстраційний номер транспортного засобу;
- Час перебування на автостоянці.

Таблиця 1.2 описує процес реєстрації транспортних засобів, що забезпечує прозорість і точність в обліку.

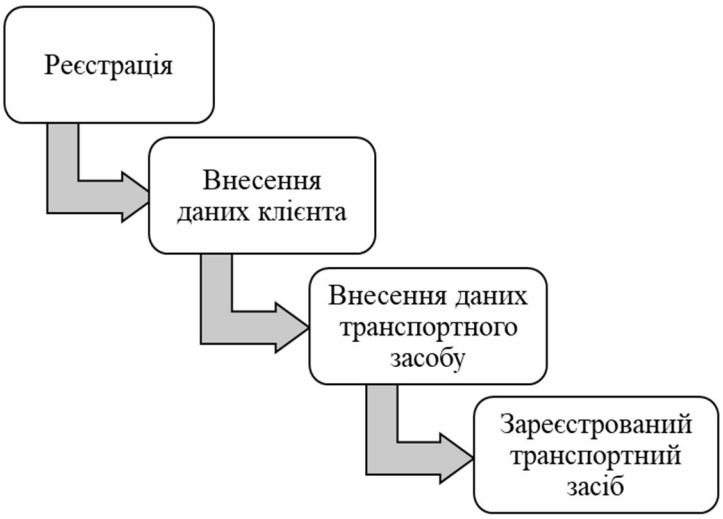


Рисунок 1.3 –Реєстрація транспортного засобу

Процес бронювання (рис. 1.4) включає:

- Визначення доступних паркомісць;
- Закріплення місця за конкретним клієнтом;
- Фіксацію часу бронювання.

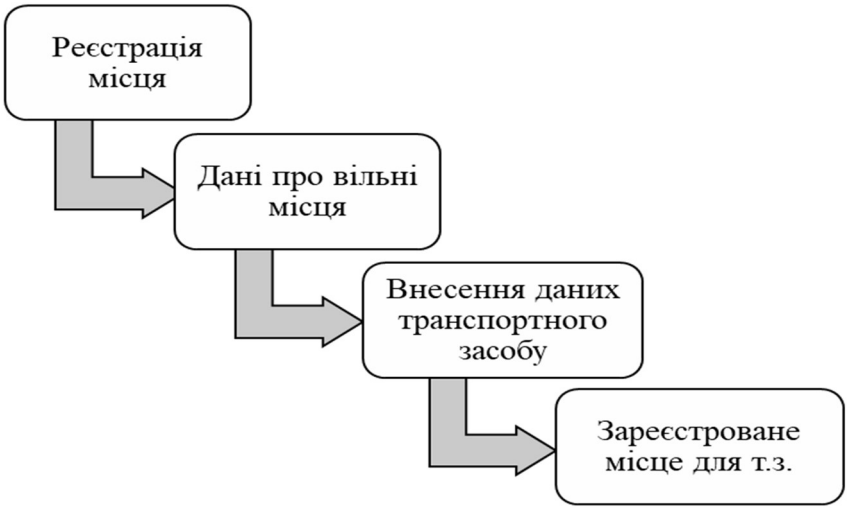


Рисунок 1.4 –Бронювання місця для паркування

Ця функція дозволяє оптимізувати використання паркувальних ресурсів і забезпечує зручність для клієнтів. Таблиця 1.3 описує характеристики процесу бронювання, враховуючи такі параметри, як доступність місць, час оренди та тип транспортного засобу.

Таблиця 1.3 – Опис Бронювання місця для паркування

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Бронювання місця на автостоянці
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про клієнта транспортний засіб
Вхідні документи	Немає
Вихідна подія	Відповідь про статус бронювання, присвоєння коду парко – місця транспортному засобу
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

1.3 Огляд існуючих програмних засобів, що вирішують аналогічні завдання

Зі стрімким розвитком Інтернет-технологій комп'ютери все більше входять практично в усі сфери життя людини. Люди можуть вільно і з користю проводити свій час. Важливого значення набуває планування акцій і розпродажів, статистика відвідуваності торгового центру. .

Кілька років тому середня кількість автомобілів на 1000 осіб становила близько 150. Сьогодні кількість автомобілів перевищує 232. Тому потрібно створювати нові паркомісця та відкривати абсолютно нові паркінги. При такій кількості автомобілів на стоянці дуже доцільно використовувати систему обліку [4].

Сьогодні ринок таких систем дуже обмежений.

Розглянемо існуючу систему паркування та її пошукову систему:

– «Easy Parking» [5];

– «Паркінг Сервіс» [6];

– «Вся Стоянка» [7];

– стоянка в м. Бориспіль [8];

– Mikcom AS101 ProPark [9].

Ці системи призначені для автоматизації процесу реєстрації та реєстрації транспортних засобів на паркінгах.

1.3.1 Програмний продукт «Simple Parking»

Програмний продукт «Simple Parking» являє собою сучасне рішення, розроблене для автоматизації управління паркувальними майданчиками. Ця система орієнтована на покращення зручності обслуговування клієнтів, оптимізацію використання паркомісць і забезпечення ефективного збору та обробки даних.

Система пропонує інтуїтивно зрозумілий інтерфейс, що полегшує роботу оператора. Головне вікно програми дозволяє здійснювати моніторинг зайнятості паркомісць у реальному часі, де вільні та зайняті місця відображаються різними кольорами чи спеціальними позначками. Це сприяє швидкому прийняттю рішень, зокрема щодо реєстрації транспортних засобів або організації паркувальних зон.

Однією з важливих функцій «Simple Parking» є можливість реєстрації заїздів і виїздів автомобілів. Оператор може оперативно вносити дані про клієнта, такі як його ім’я, контактний номер та державний номер транспортного засобу. Система також підтримує ідентифікацію типу авто (наприклад, легковий автомобіль чи вантажівка), що дозволяє оптимально розподіляти місця залежно від потреби.

Програма підтримує роботу в операційних системах Windows XP, Vista та новіших, що робить її доступною для широкого кола користувачів. Серед додаткових можливостей варто відзначити інтеграцію з обладнанням, таким як камери відеоспостереження для автоматичного розпізнавання номерних знаків, та сенсори для визначення зайнятості місць [3].

Ще однією перевагою є функція автоматичного створення звітів. Це дозволяє керівникам отримувати інформацію про загальний дохід, кількість зайнятих місць, а також статистику використання паркінгу в різні періоди. Наприклад, щоденні, тижневі або місячні звіти можуть бути представлені у вигляді таблиць чи графіків.

Крім цього, у «Simple Parking» реалізована підтримка онлайн-доступу, що відкриває можливість для клієнтів попередньо бронювати паркомісця через веб-інтерфейс. Такий підхід значно спрощує процес паркування, особливо під час пікових годин, і підвищує рівень задоволеності користувачів.

У системі також приділено увагу безпеці даних. Всі операції, пов’язані із зберіганням інформації, здійснюються із застосуванням сучасних методів шифрування, а доступ до системи захищено авторизацією користувачів. Це гарантує конфіденційність інформації про клієнтів та їхні транспортні засоби.

Таким чином, програмний продукт «Simple Parking» є багатофункціональним і адаптивним рішенням, яке дозволяє ефективно керувати паркувальними

майданчиками, покращуючи їхню доступність і підвищуючи ефективність роботи [3].

Для ілюстрації роботи системи використовується головне вікно програми, яке показано нижче (рис. 1.5). Візуалізація може бути доповнена іншими графічними матеріалами, що демонструють, наприклад, схему паркомісць, процес реєстрації клієнтів або автоматичне створення звітів. Якщо потрібні додаткові зображення чи уточнення, це можна легко реалізувати.

Рисунок 1.5 – Головне вікно програми «Simple Parking» [5]

1.3.2 Програмний продукт «SERVIO PARKING»

Програмний продукт «SERVIO PARKING» представляє собою потужне рішення для автоматизації управління паркувальними майданчиками, яке поєднує широкий функціонал і зручність використання. Ця система орієнтована на забезпечення ефективного управління паркінгом із можливістю налаштування параметрів відповідно до потреб користувачів та власників об'єктів.

Однією з головних переваг «SERVIO PARKING» є його гнучкість у налаштуванні. Система дозволяє встановлювати різноманітні параметри, такі як час використання паркувальних місць, тип валюти, кількість місць, зайнятість паркомісць та інші показники. Завдяки цьому користувачі можуть адаптувати систему до специфічних умов роботи [4].

Серед ключових функцій програмного забезпечення:

- Гнучке налаштування тарифів. Програма підтримує встановлення погодинних ставок оплати, що дозволяє власникам паркінгів легко керувати ціноутворенням залежно від попиту або часу доби.
- Управління автоматизованими паркувальними системами. Програмне забезпечення інтегрується з обладнанням для автоматизації процесів паркування, забезпечуючи оперативний доступ до системних налаштувань і моніторингу.
- Персоналізовані облікові записи для операторів. Система надає можливість створення унікальних облікових записів для кожного оператора з логіном і паролем, а також налаштуванням рівнів доступу до певних функцій. Це забезпечує додатковий рівень безпеки та прозорості.
- Звітування та архівація даних. Під час проведення будь-яких операцій, таких як оплата чи видача абонементів, система автоматично генерує та архівує відповідні звіти. Це дозволяє користувачам легко відстежувати фінансові та операційні показники.

Програмний продукт «SERVIO PARKING» має інтуїтивно зрозумілий інтерфейс, який забезпечує зручну роботу з системою. Нижче (рис. 1.6, 1.7) наведено приклади зовнішнього вигляду інтерфейсу. Інтерфейс містить основні функціональні блоки, такі як моніторинг паркомісць, реєстрація клієнтів, генерація звітів і управління тарифами.

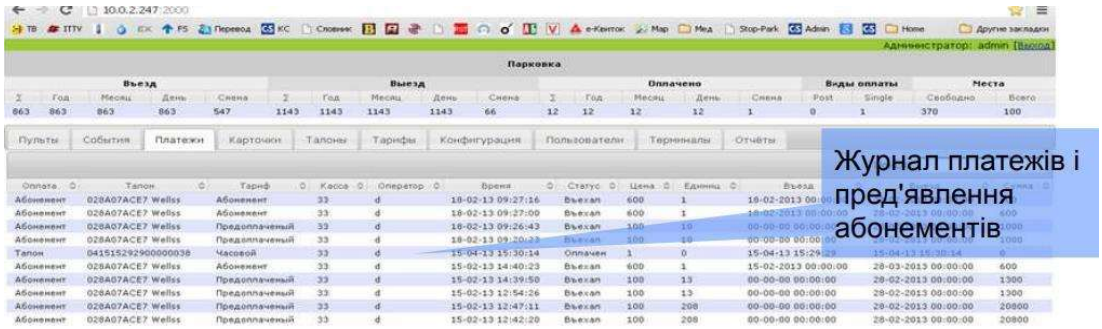


Рисунок 1.6 – Програмний продукт «SERVIO PARKING» [6]

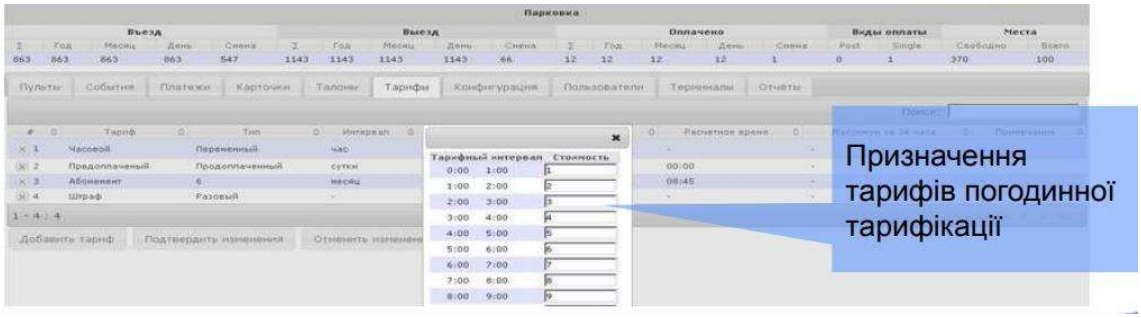


Рисунок 1.7 – Програмний продукт «SERVIO PARKING» [6]

Завдяки чіткій структурі та простоті використання система значно полегшує роботу операторів, зменшуючи кількість помилок і час, необхідний для виконання завдань.

Впровадження системи «SERVIO PARKING» на паркувальних майданчиках має низку переваг:

- Економія часу та ресурсів. Автоматизація процесів скорочує час обслуговування клієнтів і мінімізує потребу в ручній роботі.
- Підвищення точності обліку. Інтеграція з автоматичними системами контролю та створення звітів забезпечують точність даних і легкість їх аналізу.
- Покращення зручності для клієнтів. Завдяки швидкому доступу до послуг і прозорій системі оплати, користувачі отримують кращий досвід взаємодії з паркінгом.

- Можливість масштабування. Система легко адаптується до різних масштабів бізнесу.

1.3.3 Програмний продукт «AllStojanka»

Програмний продукт «AllStojanka» є сучасним інструментом для автоматизації процесів управління автостоянками. Його функціонал орієнтований на зручність використання, оптимізацію рутинних операцій і забезпечення точного обліку даних.

Система забезпечує зберігання записів у електронній формі, що мінімізує ризик втрати інформації та спрощує доступ до неї. Інтуїтивно зрозумілий інтерфейс дозволяє операторам швидко опановувати функціонал програми. Завдяки цьому реєстрація транспортних засобів і ведення обліку стають максимально простими і доступними.

Однією з ключових можливостей «AllStojanka» є можливість отримання своєчасної інформації про взаєморозрахунки між клієнтами та адміністрацією. Програма дозволяє відображати схему автостоянки з чітким поділом на зайняті та вільні місця, що значно спрощує управління і робить процес паркування швидким та зручним для користувачів.

«AllStojanka» підтримує автоматичний розрахунок вартості паркування залежно від встановлених тарифів і додаткових опцій. Це особливо актуально для великих паркувальних комплексів, де необхідно враховувати різноманітні умови обслуговування. Крім того, система дозволяє приймати платежі з видачею фіскального чека, що забезпечує прозорість фінансових операцій [5].

Серед інших важливих можливостей програми варто відзначити облік знижок і автоматичне нарахування комісій за прострочення. Програма контролює дату та час затримки платежів, дозволяючи адміністрації своєчасно реагувати на порушення умов договору. Такий функціонал забезпечує високий рівень точності обліку та дозволяє запобігати фінансовим втратам.

Програмний продукт також підтримує інтеграцію з сучасними платіжними системами, що робить його універсальним рішенням для різних типів автостоянок — від невеликих приватних майданчиків до великих паркінгів у торговельних центрах.

Використання «AllStojanka» дозволяє значно зменшити витрати часу на обробку даних і підвищити якість обслуговування клієнтів. Завдяки автоматизації облікових процесів і зручному інтерфейсу оператори можуть швидко виконувати свої обов’язки, що особливо важливо під час пікових навантажень [5].

1.3.4 Програмний продукт «Парковка, стоянка у Борисполі»

Інтернет-сервіс «Парковка, стоянка у Борисполі» є ефективним рішенням для організації паркувальних послуг поблизу міжнародного аеропорту Бориспіль. Ця система орієнтована на створення максимально зручного інтерфейсу для користувачів, що дозволяє швидко отримувати доступ до інформації про доступні паркувальні місця, тарифи та додаткові сервіси.

Основний інтерфейс сайту (рис. 1.8) представлений простою структурою, яка включає основні функціональні блоки. Головна сторінка містить інформацію про розташування паркінгів, контактні дані та можливості для онлайн-бронювання паркувальних місць. Мінімалістичний дизайн забезпечує інтуїтивно зрозумілу навігацію для користувачів, незалежно від їхнього рівня технічної підготовки.

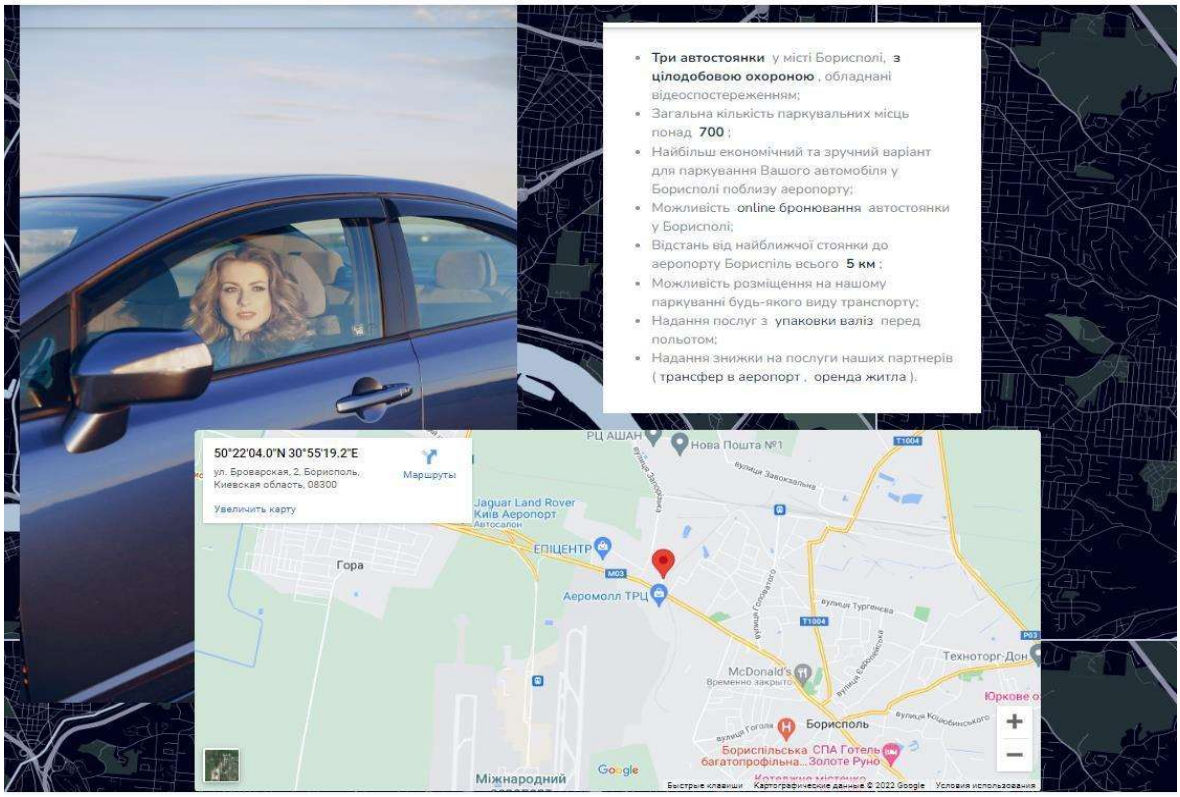


Рисунок 1.8 – Головне вікно програми «Парковка, стоянка у Борисполі» [8]

Однією з найбільш корисних функцій сервісу є «Сценарій подорожі» (рис. 1.9–1.10). Ця функція дозволяє вибирати оптимальні маршрути до трьох різних парковок, використовуючи інтеграцію з Google Maps. Завдяки цьому користувачі можуть отримати точну інформацію про маршрут, тривалість поїздки, GPS-координати, а також адреси та контактні телефони кожної парковки. Подібна інтеграція спрощує процес планування та значно підвищує зручність використання системи.

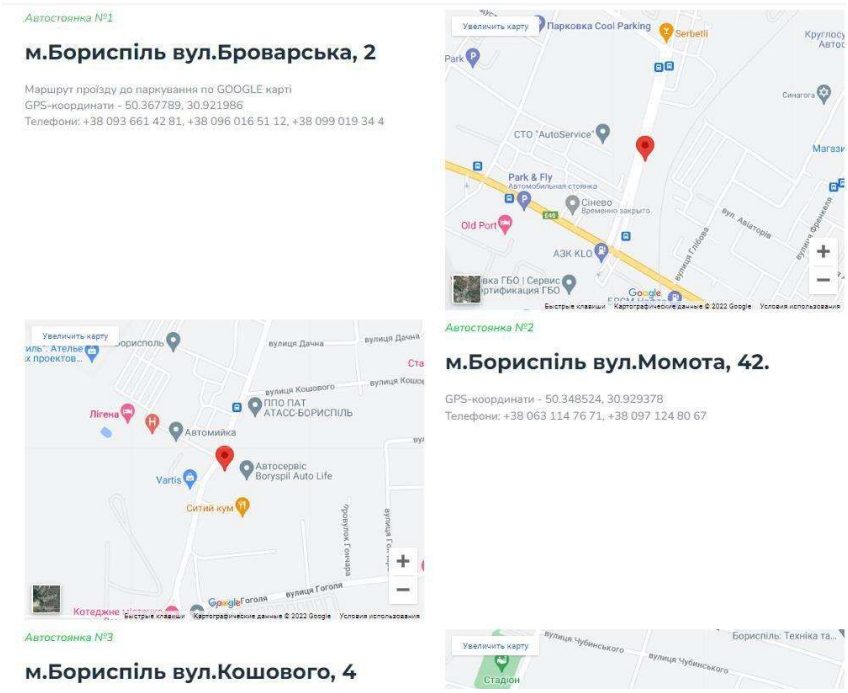


Рисунок 1.9 – Вікно програми «Парковка, стоянка у Борисполі» [8]

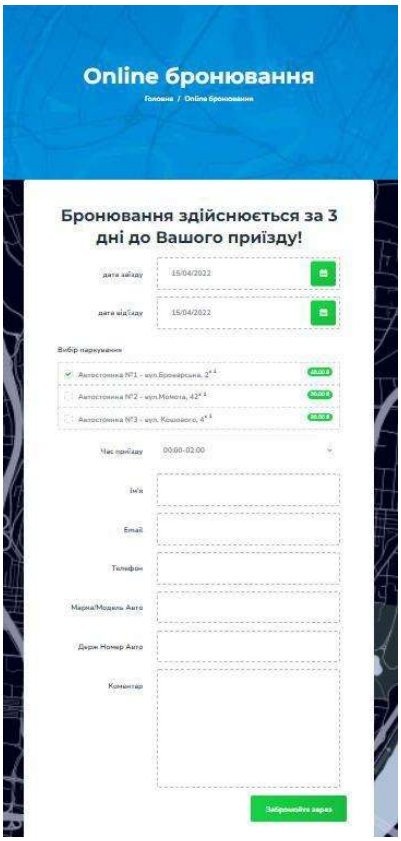


Рисунок 1.10 – Вікно програми «Парковка, стоянка у Борисполі» [8]

Функція «Онлайн-бронювання» (рис. 1.10) є важливим елементом сервісу. Вона дозволяє користувачам заздалегідь резервувати паркувальні місця через спеціальну форму, де вводяться основні дані про власника транспортного засобу та сам автомобіль. Процес бронювання реалізований максимально просто, що дозволяє мінімізувати час на підготовку до подорожі. Крім цього, система автоматично підтверджує бронювання, надсилаючи повідомлення на електронну пошту клієнта.

Додаткові вкладки системи містять корисну інформацію для клієнтів. Зокрема, тут можна знайти контактні дані адміністрації паркінгів, опис додаткових послуг, таких як технічне обслуговування автомобілів, мийка чи допомога в екстрених ситуаціях. Така багатофункціональність робить сервіс не лише засобом для організації паркування, але й центром комплексного обслуговування автомобілістів.

Інтеграція із сучасними технологіями, такими як електронні платежі та автоматичні системи контролю, дозволяє сервісу працювати з високою ефективністю. Автоматизований розрахунок вартості послуг залежно від тривалості перебування автомобіля на парковці забезпечує прозорість фінансових операцій і спрощує адміністрування. Крім того, сервіс надає можливість клієнтам оперативно змінювати деталі бронювання в режимі реального часу, що особливо важливо для мандрівників, чиї плани можуть змінюватися.

Таким чином, інтернет-сервіс «Парковка, стоянка у Борисполі» є багатофункціональною платформою, яка поєднує в собі простоту використання, широкий спектр можливостей і високий рівень обслуговування. Він забезпечує комфорт і зручність для користувачів, водночас допомагаючи адміністрації ефективно управляти ресурсами паркінгів.

1.3.5 Програмний продукт «ParkingPay»

Система «ParkingPay» представляє собою інноваційне рішення для управління паркувальними майданчиками, розроблене з метою забезпечення зручності для користувачів та ефективності для операторів. Цей програмний продукт поєднує сучасні технології, які дозволяють автоматизувати ключові процеси, пов’язані з паркуванням, та підвищити рівень обслуговування клієнтів.

Однією з ключових особливостей «ParkingPay» є її орієнтованість на простоту використання. Головний інтерфейс системи пропонує інтуїтивний доступ до основних функцій, таких як контроль зайнятости паркомісць, управління бронюваннями, а також перегляд фінансових операцій. Оператори паркувальних зон можуть оперативно відстежувати стан кожного паркомісця в режимі реального часу, отримуючи вичерпну інформацію про транспортні засоби, які перебувають на території [47].

Головний інтерфейс системи ми можемо побачити на Рис. 1.11.

Рисунок 1.11 – Вікно програми «ParkingPay»

Мобільний додаток «ParkingPay» забезпечує зручний доступ для користувачів, дозволяючи їм швидко знаходити та резервувати паркомісця. У

додатку реалізовано функцію безконтактної оплати, що робить процес розрахунків швидким і безпечним. Інтеграція з GPS-навігацією допомагає користувачам легко знайти зарезервоване місце, а система сповіщень забезпечує інформування про статус бронювання або термін завершення паркування.

Окрім мобільного додатку, система «ParkingPay» включає функціонал для операторів. Вони мають доступ до веб-панелі керування, яка дозволяє моніторити використання паркінгу, формувати звіти про зайнятість та доходи, а також аналізувати статистичні дані. Цей інструмент дає змогу операторам швидко реагувати на запити клієнтів, знижуючи кількість помилок та затримок у роботі.

Безпека даних є одним із пріоритетів системи «ParkingPay». Використовуючи сучасні методи шифрування та багаторівневу авторизацію, програма забезпечує захист персональних даних користувачів та фінансових транзакцій. Завдяки цьому клієнти можуть бути впевненими в конфіденційності своїх операцій.

Таким чином, «ParkingPay» є не лише засобом автоматизації процесів паркування, а й комплексним рішенням, яке сприяє підвищенню рівня обслуговування та ефективності управління паркувальними майданчиками. Інноваційність і зручність використання роблять цю систему популярною серед міських інфраструктур і приватних операторів паркінгу [47].

1.3.6 Порівняльна таблиця існуючих програм

Різноманіття програмних продуктів для управління паркувальними майданчиками дає можливість операторам обирати оптимальні рішення відповідно до потреб конкретного об’єкта. Сучасні системи автоматизації паркінгу спрямовані на спрощення процесів реєстрації, моніторингу, оплати та управління, а також забезпечують зручність для кінцевих користувачів. Залежно від масштабів і специфіки об'єкта, можуть бути обрані як настільні програми з базовими функціями, так і сучасні мобільні чи веб-орієнтовані рішення, що інтегрують у собі широкий набір можливостей.

Одним із ключових факторів при виборі програмного забезпечення є його функціональні можливості, такі як підтримка інтеграції з електронними системами оплати, управління бронюваннями та відображення реального часу. Для малих і середніх паркінгів важливим є доступ до інструментів моніторингу зайнятості місць, тоді як великі паркінгові комплекси потребують розширеного функціоналу, що включає контроль прострочення, нарахування штрафів, обробку статистичних даних і підтримку мобільних додатків.

Далі наведена порівняльна таблиця, яка відображає основні показники можливостей сучасних програмних продуктів для автоматизації паркувальних процесів. Вона дозволяє краще зрозуміти відмінності між системами та оцінити їх відповідність до конкретних завдань.

Таблиця 1.4 – Основні показники сучасних програмних продуктів для паркування

	SimpleParcking	SERVIO PARCKING	AllStojanka	Парковка, стоянка в Борисполі	ParkingPay
Інтерфейс користувача	ПК	ПК	ПК	веб-сайт	Мобільний додаток
Облік	+	+	+	+	+
Відображення схеми автостоянки	+	+	+	-	+
Приймання та оформлення коштів	+	+	-	-	+
Відображення конкретного ТЗ	+	+	+	-	+
Перевірка клієнтів	+	-	+	-	+
Реєстрація клієнтів	+	+	+	+	+
Контроль прострочення	+	-	+	-	-

Порівняння показало як сильні сторони, так і недоліки кожної системи. Наприклад, Simple Parking має зрозумілий і привабливий інтерфейс, але обмежена

функціоналом перегляду вільних місць онлайн через відсутність веб-доступу. Ця система орієнтована на використання операторами з мінімальними технічними знаннями, що робить її корисною для невеликих паркувальних майданчиків із обмеженим бюджетом.

SERVIO PARKING, на відміну від Simple Parking, доповнює свій функціонал можливістю відображення паркомісць і приймання оплат. Однак її складний інтерфейс може викликати труднощі у нових користувачів, що знижує її привабливість для широкого впровадження без додаткового навчання персоналу.

Система AllStojanka виділяється наявністю додаткових функцій, таких як розрахунок штрафів і знижок. Ці функції дозволяють операторам не лише здійснювати контроль, але й аналізувати ефективність використання паркувальних ресурсів. Проте її простий дизайн без сучасних елементів візуалізації обмежує привабливість для сучасних користувачів.

Система «Парковка, стоянка в Борисполі», хоча й орієнтована на надання інформації та забезпечення бронювання місць, має суттєві недоліки, зокрема відсутність автоматизації фінансових операцій і інтеграції з сучасними системами контролю. Її функціонал в основному обмежений простим резервуванням місць через веб-сайт, що робить її менш ефективною в порівнянні з конкурентами.

ParkingPay, завдяки мобільному додатку, інтегрує сучасні технології, як-от GPS-навігацію та безконтактну оплату, забезпечуючи простий доступ до функцій для користувачів. Ця система орієнтована на масштабні паркувальні комплекси, де потрібне швидке обслуговування клієнтів і мінімізація часу на паркування. Можливість перегляду стану паркінгу в реальному часі робить ParkingPay найбільш прогресивною серед розглянутих систем.

Проведений аналіз дозволяє зробити кілька важливих висновків. Системи, орієнтовані на веб-інтерфейси та мобільні додатки, такі як ParkingPay, пропонують більш сучасний і зручний функціонал, що відповідає потребам великих міських паркінгів і комерційних об’єктів. Натомість настільні системи, такі як Simple

Parking і AllStojanka, залишаються корисними для менших об’єктів, де немає потреби в інтеграції з мобільними чи веб-платформами.

Ці висновки є основою для визначення оптимальних рішень у сфері управління паркувальними майданчиками. Їхній аналіз дозволяє визначити сильні та слабкі сторони кожної системи, що є важливим етапом при створенні або вдосконаленні програмного забезпечення для паркінгів. Подальша робота буде спрямована на інтеграцію сучасних технологій у ці рішення, щоб забезпечити більшу зручність та ефективність управління.

2 ПРОЄКТУВАННЯ СИСТЕМИ ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ НА ПАРКУВАЛЬНОМУ МАЙДАНЧИКУ

2.1 Аналіз функцій системи

У процесі розробки інтелектуальної системи моніторингу та обліку транспортних засобів важливо провести ретельний аналіз функцій, які мають забезпечувати її ефективну роботу. Система повинна відповідати сучасним вимогам до автоматизації процесів, забезпечувати точність, зручність та безпеку даних.

Основними функціями системи є:

1. Реєстрація користувачів та транспортних засобів. Система повинна забезпечувати можливість швидкої реєстрації користувачів та внесення інформації про транспортні засоби. Це включає збір таких даних, як ім'я, контактна інформація користувача, номерний знак, марка та модель автомобіля. Реєстрація є ключовим етапом для ідентифікації клієнтів та управління їх бронюваннями.
2. Бронювання паркувальних місць. Користувачі повинні мати можливість вибору вільного паркувального місця, зазначення дати та часу бронювання. Система має забезпечувати автоматичне оновлення статусу паркувальних місць у реальному часі, щоб уникнути конфліктів між бронюваннями.
3. Управління бронюваннями. Система повинна дозволяти користувачам змінювати або скасовувати свої бронювання за необхідності. Для адміністратора реалізується можливість перегляду та управління всіма бронюваннями на паркінгу, зокрема скасування помилкових або неактуальних записів.
4. Моніторинг завантаженості паркінгу. Інтерфейс адміністратора має відображати актуальний стан зайнятості паркувальних місць, включаючи

заброньовані, вільні та скасовані місця. Це дозволяє оперативно керувати ресурсами паркінгу та покращувати його ефективність.

5. Статистичний аналіз. Система повинна збирати та аналізувати дані про завантаженість паркінгу, частоту бронювань, тривалість перебування автомобілів на паркувальних місцях. Отримані статистичні звіти можуть використовуватися для оптимізації роботи паркінгу та прийняття управлінських рішень.
6. Інтеграція з платіжними системами. Забезпечення можливості прийому оплати за послуги паркування через онлайн-платіжні системи. Інтеграція дозволить користувачам зручно сплачувати за бронювання та уникати черг на місці.
7. Забезпечення безпеки даних. Реалізація надійного захисту інформації через механізми аутентифікації, авторизації та шифрування даних. Це критично важливо для забезпечення конфіденційності інформації користувачів і запобігання несанкціонованому доступу до системи.
8. Гнучкість і масштабованість. Система має бути побудована таким чином, щоб її можна було легко масштабувати залежно від обсягів завантаженості. Це включає можливість роботи з великими базами даних, інтеграцію нових модулів та адаптацію до різних платформ.

На Рис. 2.1 представлено основні функції інтелектуальної системи моніторингу та обліку транспортних засобів.

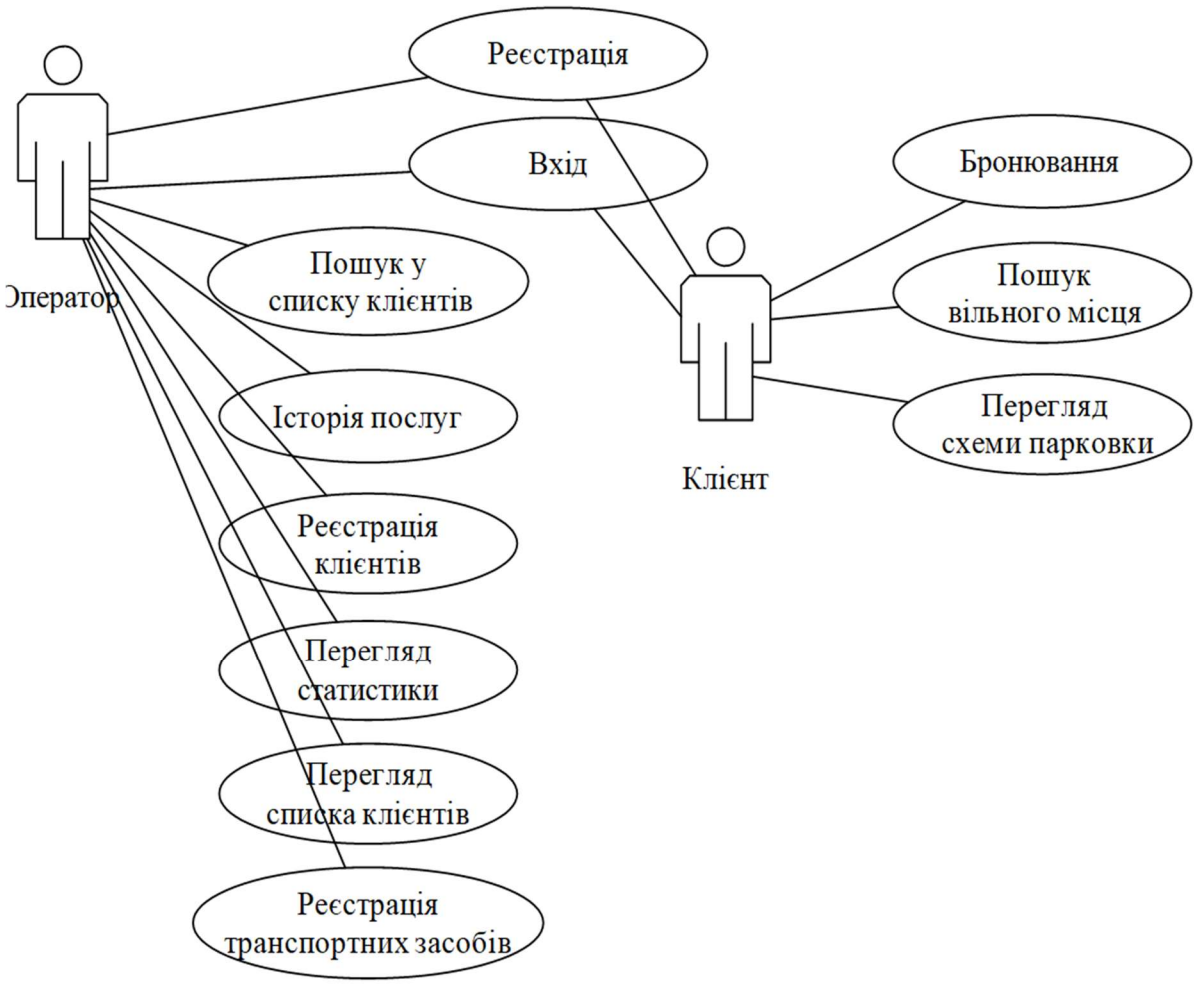


Рисунок 2.1 – Діаграма варіантів використання

Аналіз функцій системи дозволяє сформулювати основні вимоги до її реалізації. Запропонована структура забезпечить ефективну автоматизацію процесів управління паркуванням, підвищить зручність для користувачів і адміністраторів та дозволить оптимізувати роботу паркінгу в умовах різного рівня навантаження.

2.2 Проєктування архітектури системи

Архітектура системи є основою її функціонування, оскільки визначає структуру, зв'язки між компонентами та механізми обробки даних. Для розробки інтелектуальної системи моніторингу та обліку транспортних засобів було обрано сучасний підхід із використанням багаторівневої архітектури, яка забезпечує гнучкість, масштабованість та ефективність.

Основні компоненти архітектури

1. Клієнтський рівень. Відповідає за взаємодію з користувачем. Він забезпечує інтуїтивно зрозумілий інтерфейс, який включає форми реєстрації, бронювання паркувальних місць та перегляд статистики.
2. Серверний рівень. Забезпечує обробку запитів від клієнтської частини, управління бізнес-логікою та взаємодію з базою даних. Він також відповідає за виконання правил безпеки та обробку статистичних даних.
3. Рівень управління контентом. Включає функції, що дозволяють зручно оновлювати дані про паркувальні місця, керувати інформацією про користувачів та переглядати аналітику.
4. База даних. Зберігає всю необхідну інформацію, зокрема:
    - Дані про користувачів.
    - Інформацію про паркувальні місця та статус їх зайнятості.
    - Історію бронювань та статистичні дані.
    - Інтеграційний рівень

Забезпечує зв'язок між компонентами системи, а також із зовнішніми сервісами, такими як платіжні платформи чи аналітичні системи. Він дозволяє інтегрувати додаткові модулі без значних змін у структурі системи.

Принципи проєктування архітектури:

- Модульність. Компоненти системи розділені за функціональними ознаками, що спрощує підтримку та розвиток. Модульний підхід дозволяє легко оновлювати чи замінювати окремі частини системи.
- Масштабованість. Архітектура розроблена для роботи з різним рівнем навантаження. Вона підтримує збільшення кількості користувачів, обсягу даних та функціональних модулів.
- Безпека. Передбачає впровадження механізмів для захисту даних користувачів та збереження конфіденційності інформації. Архітектура включає шари захисту для запобігання несанкціонованому доступу.
- Продуктивність. Забезпечується оптимізацією взаємодії між компонентами, швидким доступом до бази даних і використанням асинхронних процесів для обробки запитів.
- Гнучкість. Система повинна підтримувати легке адаптування до нових вимог, таких як інтеграція додаткових сервісів або впровадження нових функцій.

Проєктування архітектури системи є ключовим етапом розробки, що визначає її ефективність та можливості для подальшого розвитку. Обрана архітектурна модель забезпечує гнучкість, масштабованість і продуктивність, необхідні для реалізації цілей проєкту.

2.3 Вибір технології розробки

Процес вибору технології розробки є одним із ключових етапів у створенні програмного забезпечення, оскільки він визначає архітектуру, продуктивність та функціональні можливості майбутнього продукту. Для даного проєкту розробка має здійснюватися у вигляді веб-додатку, що забезпечує доступ користувачів через Інтернет.

Веб-додаток — це прикладна програма, яка розміщується на віддаленому сервері та доступна через веб-браузер. Основною перевагою веб-додатків є їх доступність без необхідності встановлення на локальні пристрої. Згідно з визначенням Джарела Реміка, редактора Web.AppStorm, будь-який компонент веб-сайту, який виконує певну функцію для користувача, може кваліфікуватися як веб-додаток [17] - [19].

Веб-додатки розробляються для широкого спектру використання, включаючи індивідуальні та корпоративні потреби. Наприклад, такі популярні веб-програми, як поштові клієнти (наприклад, Gmail), онлайн-калькулятори чи платформи електронної комерції, стали невід'ємною частиною сучасного цифрового світу. Незважаючи на те, що деякі веб-додатки можуть бути сумісними лише з певними браузерами, більшість із них підтримуються незалежно від вибору користувача.

Однією з ключових переваг веб-додатків є те, що їх не потрібно завантажувати або встановлювати. Користувачі можуть отримати до них доступ через популярні веб-браузери, такі як Google Chrome, Mozilla Firefox чи Safari, що робить їх універсальними та зручними для використання.

Для ефективної роботи веб-додатків потрібна наявність трьох основних компонентів:

- Веб-сервер — відповідає за обробку запитів від клієнтів і доставку відповідей.
- Сервер додатків — виконує запитувані операції, забезпечуючи функціонування додатку.
- База даних — використовується для зберігання необхідної інформації, такої як дані користувачів чи контент сайту [20].

Веб-додатки зазвичай мають короткий цикл розробки, що дозволяє створювати їх невеликими командами розробників. Вони пишуться з використанням сучасних мов програмування та технологій.

Основними інструментами є:

- Клієнтські технології (JavaScript, HTML5, CSS), що забезпечують створення інтерфейсу користувача.
- Серверні мови програмування (Python, Java, Ruby), які відповідають за реалізацію бізнес-логіки.
- Фреймворки та інструменти для оптимізації процесу розробки, такі як React, Angular чи Django.

Ключові аспекти вибору технології

При виборі технології розробки необхідно враховувати наступні критерії:

- Сумісність з існуючими системами — важливо забезпечити інтеграцію з наявною інфраструктурою.
- Масштабованість — технології повинні дозволяти розширення функціоналу додатку відповідно до зростаючих потреб користувачів.
- Продуктивність — обрані інструменти повинні забезпечувати швидкість роботи навіть при високих навантаженнях.
- Безпека — технології мають включати механізми захисту даних і запобігання кіберзагрозам.

Таким чином, процес вибору технології розробки є багатофакторним і потребує ретельного аналізу цілей проєкту, наявних ресурсів та очікуваних результатів. Обрані рішення повинні сприяти створенню надійного, масштабованого та продуктивного веб-додатку, який задовольнить потреби кінцевих користувачів.

2.4 Мова програмування PHP

PHP (Hypertext Preprocessor) — це популярна сценарна мова програмування, яка використовується для створення динамічних веб-сторінок і додатків. Вона

працює на стороні сервера, генеруючи HTML-код, який відображається у браузері користувача. PHP є однією з найпоширеніших мов у світі веб-розробки завдяки своїй простоті, гнучкості та доступності [27].

PHP вирізняється низкою характеристик, які роблять її популярним вибором для веб-розробників:

- Вбудовані бібліотеки баз даних: PHP підтримує інтеграцію з багатьма базами даних, такими як MySQL, PostgreSQL, SQLite, Oracle та іншими. Це забезпечує легке управління даними для динамічних веб-сайтів.
- Запозичення із C та Perl: Синтаксис PHP базується на принципах цих мов, що робить її знайомою для розробників із досвідом у C або Perl.
- Безкоштовність: PHP є відкритою технологією, доступною для використання безкоштовно, що робить її привабливим вибором для стартапів і невеликих проєктів.
- Ефективність: Мова оптимізована для швидкої обробки запитів, що дозволяє створювати продуктивні додатки з мінімальними ресурсними витратами.
- PHP використовується для розробки широкого спектру додатків, включаючи:
- Системи управління контентом (CMS): Популярні платформи, такі як WordPress, Drupal і Joomla, побудовані на PHP.
- Електронна комерція: Magento та OpenCart — приклади платформ для онлайн-торгівлі, створених на PHP.
- Форуми та соціальні мережі: Багато платформ для взаємодії користувачів також використовують PHP завдяки його гнучкості та продуктивності.

PHP забезпечує швидкий розвиток додатків завдяки великій кількості бібліотек і фреймворків, таких як Laravel і Symfony. Мова легко інтегрується з HTML і JavaScript, що дозволяє створювати інтерактивні веб-додатки. Активна

спільнота розробників надає доступ до численних ресурсів, розширень і технічної підтримки.

PHP продовжує залишатися ключовою технологією для створення динамічних веб-додатків, особливо в малих і середніх проєктах, завдяки своїй простоті, гнучкості та універсальності.

2.5 Мова програмування C ++

C++ — це одна з найвідоміших і найпотужніших мов програмування, яка поєднує можливості об’єктно-орієнтованого підходу з високим рівнем продуктивності. Її створення бере початок у 1980-х роках, і з того часу вона стала базовою мовою для багатьох програмних рішень. C++ дозволяє розробникам створювати програми з високою продуктивністю, контролем над ресурсами та широкими можливостями для оптимізації, що робить її універсальним вибором для розробки програмного забезпечення.

Мова підтримує багаторівневу архітектуру, поєднуючи низькорівневий доступ до апаратного забезпечення з високорівневими можливостями модульності та абстракції. Вона використовується для створення як системного програмного забезпечення, такого як операційні системи та драйвери, так і додатків для настільних комп’ютерів, ігор, високопродуктивних обчислень (HPC), хмарних додатків, а також програм для вбудованих систем. Завдяки цьому C++ часто вважається "мовою вибору" для розробників, які працюють у різних галузях.

C++ також є основою для багатьох інших мов програмування, таких як C#, Java та Python, адже саме на C++ написані бібліотеки, інструменти та компілятори для цих мов. Це робить C++ важливою складовою екосистеми програмування.

Основні переваги C++:

- Продуктивність і ефективність: Завдяки доступу до апаратного рівня програми на C++ працюють швидше, ніж більшість інших мов, що

дозволяє створювати оптимізовані рішення з мінімальними ресурсними витратами.

- Універсальність: Мова дозволяє створювати додатки для різних платформ, включаючи Windows, MacOS, Linux, мобільні пристрої та вбудовані системи.
- Гнучкість: C++ підтримує різні стилі програмування, включаючи об’єктно-орієнтований, процедурний і функціональний підходи, що дозволяє адаптувати її до потреб проєкту.
- Широка екосистема: Стандартна бібліотека STL (Standard Template Library) забезпечує доступ до контейнерів, алгоритмів та інших інструментів, що значно спрощує розробку.
- Контроль над ресурсами: Завдяки ручному управлінню пам’яттю та низькорівневому доступу C++ дозволяє розробникам створювати максимально оптимізовані програми.

C++ активно використовується в різних галузях програмування:

- Системне програмне забезпечення: Мова є ключовим інструментом для створення операційних систем, драйверів і утиліт.
- Ігрова індустрія: Завдяки високій продуктивності C++ є стандартом для розробки ігор та ігрових рушіїв, таких як Unreal Engine.
- Вбудовані системи: C++ дозволяє створювати програми для мікроконтролерів, сенсорів та інших пристроїв із обмеженими ресурсами.
- Наукові обчислення: Використовується для розробки високопродуктивних обчислювальних систем у фізиці, математиці та інших науках.
- Фінансові системи: Завдяки точності та продуктивності C++ використовується у банківських та трейдингових додатках.

C++ є незамінною мовою для розробки проєктів, які потребують продуктивності, контролю над апаратними ресурсами та гнучкості в підходах до вирішення завдань. Вона залишається актуальною й надалі, завдяки активному розвитку стандартів мови та широкій підтримці спільноти розробників [38].

2.6 Мова програмування C# та технологія ASP.NET

C# та ASP.NET утворюють потужний дует для створення сучасних веб-додатків. Розроблені компанією Microsoft, ці технології забезпечують ефективність, продуктивність і гнучкість, що робить їх основним вибором для багатьох розробників.

C# (C-Sharp) — це об'єктно-орієнтована мова програмування високого рівня, яка поєднує простоту використання з потужними можливостями. Вона була створена спеціально для роботи з платформою .NET, що забезпечує її інтеграцію з іншими продуктами Microsoft.

C# характеризується чітким і зрозумілим синтаксисом, який нагадує C++ та Java, але спрощений для розробників. Завдяки цьому мова підходить для створення додатків різного рівня складності: від десктопних і мобільних до веб-додатків та ігор [44].

Особливості C#:

- Об'єктно-орієнтованість: Підтримка наслідування, інкапсуляції та поліморфізму робить мову ідеальною для модульної розробки.
- Універсальність: Мова підходить для створення додатків на різних платформах, включаючи Windows, MacOS, Linux та мобільні пристрої.
- Безпека: Вбудовані механізми управління пам'яттю, такі як автоматичне збирання сміття, зменшують ризик помилок у роботі додатків.

C# активно використовується для розробки ігор (зокрема з використанням Unity), корпоративних додатків, а також веб-рішень, де вона демонструє високу продуктивність у поєднанні з ASP.NET.

ASP.NET — це сучасна платформа для створення динамічних веб-додатків, яка базується на середовищі виконання .NET. Вона дозволяє будувати як прості веб-сайти, так і складні корпоративні системи, пропонуючи розробникам гнучкість і широкий набір інструментів.

ASP.NET підтримує різноманітні архітектурні шаблони, включаючи MVC (Model-View-Controller) і Web API. Завдяки цьому розробники можуть створювати додатки з чітким розділенням логіки, представлення і даних. Платформа також забезпечує інтеграцію з такими хмарними сервісами, як Microsoft Azure, що відкриває нові можливості для створення масштабованих і надійних веб-рішень.

Особливості ASP.NET:

- Кросплатформність: Додатки можуть працювати на Windows, Linux і MacOS, що забезпечує гнучкість у виборі середовища розгортання.
- Безпека: Вбудовані механізми аутентифікації, авторизації та захисту від загроз, таких як SQL-ін'єкції чи CSRF.
- Продуктивність: Використання Just-In-Time (JIT) компіляції забезпечує швидке виконання коду та оптимізацію ресурсів.
- Модульна архітектура: ASP.NET Core дозволяє використовувати лише необхідні компоненти, що зменшує обсяг додатку та покращує його продуктивність.

Поєднання C# та ASP.NET створює ідеальне середовище для розробки сучасних додатків. C# забезпечує стабільну основу з точки зору програмування, тоді як ASP.NET додає необхідні інструменти для веб-розробки. Разом вони дозволяють створювати продуктивні, безпечні та масштабовані додатки, які відповідають сучасним вимогам бізнесу та технологій.[44]

2.7 Фреймворк Laravel

Laravel — це популярний фреймворк для розробки веб-додатків на PHP із відкритим вихідним кодом. Він створений для спрощення процесу створення веб-додатків шляхом використання архітектурного шаблону MVC (Model-View-Controller). Laravel надає потужний набір інструментів і функцій, які полегшують розробку та дозволяють створювати повнофункціональні веб-додатки із сучасною структурою та високою продуктивністю [21].

Laravel оптимізує виконання типових задач, які зустрічаються у більшості веб-проєктів. Серед них:

- Маршрутизація: Фреймворк дозволяє легко налаштовувати маршрути для обробки HTTP-запитів, забезпечуючи інтуїтивну конфігурацію та простоту використання.
- Кешування: Laravel підтримує різні механізми кешування для підвищення продуктивності додатків.
- Перевірка: Вбудовані механізми для перевірки даних допомагають забезпечити цілісність і безпеку введених користувачами даних.

Laravel був розроблений із метою спрощення процесу розробки без втрати функціональності. Використання цього фреймворку дозволяє заощадити час, зменшивши потребу в написанні повторюваного коду, і забезпечує швидку розробку додатків (RAD — Rapid Application Development) [21].

Laravel має численні переваги, які роблять його одним із найпопулярніших фреймворків для розробки веб-додатків:

- Висока безпека: Laravel забезпечує захист додатків від поширених загроз, таких як SQL-ін’єкції, міжсайтовий скриптинг (XSS) та підробка міжсайтових запитів (CSRF).
- Покращена продуктивність: Використання кешування та оптимізації дозволяє значно прискорити роботу додатків.

- Відкритий код: Laravel є безкоштовним і підтримується великою спільнотою розробників, що забезпечує доступ до численних ресурсів та розширень.
- Шаблони Blade: Вбудований шаблонізатор Blade дозволяє легко створювати динамічні веб-сторінки.
- Міграція бази даних: Інструменти міграції полегшують управління базами даних, забезпечуючи контроль за змінами та версіями.
- MVC-архітектура: Чітке розділення логіки додатків, представлення даних і управління базами даних.
- Бібліотеки: Laravel надає об’єктно-орієнтовані бібліотеки, що включають функції аутентифікації, шифрування, обробки сесій та багато іншого.
- Модульне тестування: Вбудовані інструменти для тестування дозволяють перевіряти функціональність додатків на кожному етапі розробки.
- Обробка помилок: Laravel автоматично обробляє помилки та винятки, спрощуючи діагностику та виправлення.
- Система завдань: Фреймворк підтримує відкладене виконання завдань і планування їх виконання за розкладом.
- Інтеграція з поштовими сервісами: Просте підключення до поштових сервісів для надсилання повідомлень із додатка.

Laravel надає розробникам інтуїтивно зрозумілі інструменти, які дозволяють зосередитися на розробці логіки додатку, а не на повторюваних завданнях. Завдяки дружньому коду та підтримці багатомовності, Laravel є ідеальним вибором для створення як локальних, так і міжнародних веб-додатків.

Його широка функціональність і активна спільнота дозволяють швидко вирішувати технічні виклики, а постійне оновлення фреймворку забезпечує відповідність сучасним стандартам розробки.

2.8 Фреймворк Django

Django — це високорівневий веб-фреймворк для мови Python, створений для спрощення процесу розробки веб-додатків. Він забезпечує розробникам набір готових інструментів і компонентів, які дозволяють зосередитися на функціональності програми, а не на вирішенні рутинних завдань. Django є фреймворком із відкритим вихідним кодом, що робить його доступним і популярним серед розробників у всьому світі [23].

Django побудований на принципах швидкості, безпеки та масштабованості. Це означає, що фреймворк дозволяє швидко створювати прототипи додатків, забезпечує захист від поширених загроз, таких як SQL-ін’єкції або міжсайтові скрипти (XSS), та підтримує високі навантаження, що робить його ідеальним для великих проєктів.

Серед ключових характеристик фреймворку Django можна виділити:

- Архітектура Model-View-Template (MVT): Забезпечує чітке розділення даних, логіки та інтерфейсу, що сприяє підтримуваності й повторному використанню коду.
- Вбудована ORM: Django має потужну систему для роботи з базами даних, яка дозволяє виконувати запити без написання SQL-коду, підтримуючи транзакції та зв’язки між моделями.
- Автоматизований адміністративний інтерфейс: Генерується автоматично на основі моделей і дозволяє адмініструвати дані через інтуїтивний веб-інтерфейс із підтримкою багатомовності.
- Гнучкий маршрутизатор URL: Система маршрутизації в Django базується на регулярних виразах, що дозволяє легко налаштовувати маршрути для додатку.
- Шаблонізатор: Django пропонує потужну систему шаблонів із підтримкою тегів і фільтрів для створення динамічних сторінок.

Django має численні переваги, які роблять його одним із провідних фреймворків для розробки веб-додатків:

- ORM (Object-Relational Mapping): Дозволяє працювати з базами даних за допомогою об’єктно-орієнтованих запитів, що прискорює розробку і забезпечує підтримку транзакцій.
- Вбудований адміністративний інтерфейс: Пропонує інструмент для управління даними з перекладами для багатьох мов, що зручно для багатомовних додатків.
- Система кешування: Підтримує різні механізми кешування для оптимізації продуктивності додатків.
- Глобалізація: Django має вбудовану підтримку локалізації, що дозволяє створювати додатки для міжнародної аудиторії.
- Підключувана архітектура: Легко розширюється за рахунок використання сторонніх додатків, які можна інтегрувати до будь-якого проєкту на Django.
- Безпека: Вбудовані механізми захисту забезпечують надійний захист від поширених веб-загроз.
- Підтримка middleware: Надає систему для обробки запитів і відповідей на рівні сервера, що забезпечує масштабованість і модульність.
- Робота з формами: Вбудована бібліотека для обробки форм спрощує створення форм, їх валідацію та обробку даних.

Django надає розробникам потужні інструменти для швидкої і ефективної розробки додатків будь-якої складності — від простих веб-сайтів до великих корпоративних платформ. Завдяки своїй модульній архітектурі, багатій екосистемі додатків та активній спільноті, Django залишається одним із найкращих виборів для розробників Python. Його універсальність і масштабованість роблять цей фреймворк особливо популярним у сфері електронної комерції, медіа-платформ і наукових проєктів.[24]

2.9 Node.js

Node.js — це міжплатформне середовище виконання JavaScript із відкритим кодом, створене для ефективної розробки серверних і мережевих додатків. Його унікальність полягає у використанні JavaScript як універсальної мови для клієнтської та серверної частини, що дозволяє об'єднати весь процес розробки в єдиній екосистемі. Завдяки цьому Node.js став популярним серед розробників, які прагнуть до швидкої, продуктивної та легко масштабованої розробки [25].

Основною особливістю Node.js є його однопотокова архітектура з використанням подій, яка дозволяє обробляти тисячі одночасних з’єднань без перевантаження сервера. Це стало можливим завдяки неблокуючим операціям введення-виведення, які забезпечують ефективне використання ресурсів. Node.js також працює на рушії V8, який значно прискорює виконання JavaScript-коду, що є важливим для додатків, орієнтованих на реальний час, таких як чати або потокове відео.

Середовище Node.js має широкий спектр застосувань завдяки своїй кросплатформності. Воно підтримує операційні системи Windows, MacOS, Linux, Unix і навіть ARM-процесори, такі як Raspberry Pi. Це робить його ідеальним вибором як для невеликих інструментів, так і для великих корпоративних систем. Інтеграція з npm (менеджером пакетів Node.js) надає доступ до десятків тисяч бібліотек, що значно спрощує розробку та впровадження нових функцій.

Node.js також вирізняється своєю здатністю обробляти реальні навантаження в реальному часі. Наприклад, воно дозволяє створювати веб-додатки, які можуть одночасно обслуговувати велику кількість користувачів без втрати продуктивності. Це забезпечується за рахунок високої масштабованості, яку можна реалізувати як горизонтально, так і вертикально, шляхом додавання нових серверів або оптимізації ресурсів на існуючих.

Завдяки своїй універсальності Node.js забезпечує інтеграцію з іншими технологіями та мовами програмування, що дозволяє створювати гібридні рішення

для складних проектів. Крім того, Node.js активно підтримується великою спільнотою розробників, яка регулярно оновлює середовище та додає нові інструменти, що відповідають сучасним стандартам індустрії.

Таким чином, Node.js є потужним інструментом для створення додатків, які потребують високої продуктивності, гнучкості та можливості працювати в умовах реального часу. Завдяки своїй архітектурі та розвиненій екосистемі Node.js залишається одним із провідних виборів для сучасних розробників.[25]

2.10 Фреймворк Qt

Qt — це потужний графічний інструментарій для мови C++, створений компанією Trolltech у 1994 році. Завдяки своїй універсальності, Qt став одним із провідних інструментів для розробки додатків, які працюють на різноманітних платформах. Він підтримує Windows, Unix, MacOS та інші операційні системи, що робить його популярним серед розробників, які прагнуть створювати кросплатформні рішення [26].

Qt є повністю об’єктно-орієнтованим інструментарієм, що базується на потужній бібліотеці класів. Він дозволяє створювати додатки, які можуть функціонувати на різних типах програмного та апаратного забезпечення. Основою фреймворку є модулі Qt Essentials, які забезпечують базову функціональність для побудови інтерфейсів користувача, управління потоками, роботи з мережею та іншими компонентами додатку.

Qt пропонує два типи ліцензування: комерційну ліцензію для підприємств і безкоштовну ліцензію з відкритим кодом. Такий підхід дозволяє розробникам обирати ліцензію відповідно до специфіки свого проєкту.

Фреймворк Qt широко використовується у різноманітних галузях, зокрема:

- Вбудовані системи та Інтернет речей (IoT): Qt є популярним вибором для створення програмного забезпечення, яке працює на вбудованих пристроях, таких як системи розумного дому чи промислові контролери.

- Настільні додатки: Завдяки підтримці складних графічних інтерфейсів Qt залишається одним із провідних рішень для розробки настільних додатків, як для Windows, так і для MacOS та Unix.
- Мобільні додатки: Qt дозволяє створювати кросплатформні мобільні додатки, які працюють як на iOS, так і на Android.

Одним із найвідоміших проектів на основі Qt є KDE Plasma — популярне робоче середовище для Unix-систем, яке поєднує функціональність та естетичність. Окрім того, Qt використовується такими великими компаніями, як LG, Tesla, Microsoft, Samsung, BMW, Siemens, HP та Philips. Цей фреймворк довів свою надійність у розробці програмного забезпечення для автомобільної, медичної та технологічної галузей.

Переваги використання Qt:

- Qt забезпечує високий рівень абстракції, що спрощує розробку кросплатформних додатків. Його потужний набір інструментів дозволяє:
- Розробляти інтерфейси користувача з використанням Qt Widgets або сучасних технологій, таких як Qt Quick.
- Інтегрувати багатофункціональні API для роботи з мережею, базами даних, графікою та мультимедіа.
- Забезпечувати високу продуктивність і масштабованість додатків завдяки ефективній обробці потоків і подій.

2.11 Технологія Umbraco

Umbraco — це потужна система управління контентом (CMS) із відкритим вихідним кодом, створена для забезпечення гнучкості, масштабованості та зручності у використанні. Вона побудована на платформі .NET, що дозволяє інтегрувати її з іншими продуктами Microsoft, такими як Azure, Dynamics і Visual

Studio. Завдяки своєму відкритому коду Umbraco надає можливість адаптувати систему до будь-яких потреб розробника чи кінцевого користувача, забезпечуючи максимальну функціональність та контроль над процесом розробки [45].

Umbraco вирізняється своєю простотою у використанні для контент-менеджерів та розширеними можливостями для розробників. Інтуїтивно зрозумілий інтерфейс дозволяє редагувати контент, керувати сторінками та працювати з мультимедійними матеріалами без потреби у спеціальних технічних знаннях. Для розробників Umbraco пропонує гнучке середовище для створення кастомізованих рішень, використовуючи потужність платформи .NET. Система підтримує складні сценарії інтеграції, дозволяючи поєднувати її з іншими інструментами та сервісами, такими як бази даних, API та хмарні рішення.

Однією з ключових переваг Umbraco є її модульна архітектура, яка забезпечує легкість налаштування та розширення функціональності. Система включає потужний набір інструментів для управління контентом, медіафайлами та SEO-оптимізації. Крім того, Umbraco пропонує підтримку багатомовності, що робить її особливо корисною для міжнародних проєктів.

Основні можливості Umbraco:

- Підтримка багатомовних проєктів: Система дозволяє створювати сайти з локалізацією для різних мов і регіонів.
- Медіа-бібліотека: Вбудований інструмент для управління мультимедійними файлами, такими як зображення, відео чи документи, забезпечує швидкий доступ і організацію контенту.
- Масштабованість: Umbraco ефективно працює з великими обсягами даних і забезпечує стабільність навіть у високонавантажених середовищах.
- Редакторський інтерфейс: Проста у використанні панель адміністратора забезпечує зручність управління сторінками та швидкість доступу до основних функцій.

Umbraco знайшла широке застосування в різних галузях, включаючи корпоративні веб-сайти, платформи електронної комерції, інформаційні портали та блоги. Наприклад, великі компанії використовують цю систему для управління складними веб-ресурсами, інтегруючи її з іншими бізнес-інструментами. Завдяки гнучкості Umbraco також ідеально підходить для створення персоналізованих веб-сайтів із унікальним дизайном і функціональністю.

Переваги використання Umbraco:

- Прозорість та адаптивність: Відкритий вихідний код дозволяє модифікувати систему відповідно до специфічних вимог.
- Інтеграція з .NET: Розробники можуть використовувати потужність екосистеми .NET для створення масштабованих рішень із високою продуктивністю.
- Широка підтримка: Umbraco має активну спільноту користувачів і розробників, що забезпечує доступ до численних плагінів і оновлень.
- Хмарна оптимізація: Можливість розгортання на Microsoft Azure забезпечує швидкість роботи, безпеку та зручність у використанні.

Umbraco постійно вдосконалюється, пропонуючи нові інструменти та рішення, які відповідають сучасним стандартам веб-розробки. Завдяки багатофункціональності та підтримці масштабованості ця система залишається одним із найкращих виборів для створення як простих сайтів, так і складних корпоративних платформ.[45]

2.12 Порівняння мов та фреймворків програмування

Для вибору оптимальної мови програмування та фреймворку для розробки веб-додатків було проведено порівняльний аналіз ключових характеристик популярних мов і платформ. У таблиці 2.1 наведено порівняння мов

програмування, таких як C#, PHP, Python, JavaScript та C++, які широко використовуються у веб-розробці.

Таблиця 2.1 — Порівняння мов програмування

Характеристика	**C#**	**PHP**	**Python**	**JavaScript**	**C++**
Призначення	Веб-додатки, десктопні системи, мобільні додатки	Веб-додатки	Наукові обчислення, веб-додатки	Інтерактивність на стороні клієнта, серверні додатки	Системне програмування, ігри, вбудовані системи
Продуктивність	Висока	Середня	Середня	Висока	Дуже висока
Масштабованість	Висока	Середня	Висока	Висока	Висока
Синтаксис	Об’єктно-орієнтований	Простий, процедурний	Простий, зрозумілий	Гнучкий, нестрогий	Складний, строгий
Інструменти розробки	Visual Studio, Rider	PhpStorm, VS Code	PyCharm, VS Code	VS Code, WebStorm	Visual Studio, CLion
Безпека	Висока	Середня	Висока	Середня	Висока
Кросплатформність	Висока	Висока	Висока	Висока	Висока
Сфера використання	Корпоративні додатки, веб-розробка, ігри	Веб-додатки	Моделювання, машинне навчання	Інтерактивні веб-сайти, сервери	Ігри, операційні системи

Сучасна веб-розробка також потребує вибору відповідного фреймворку. У таблиці 2.2 наведено порівняння популярних фреймворків, таких як Umbraco, Laravel, Django, Node.js та Qt.

Таблиця 2.2 – Порівняння фреймворку веб розробки

Характеристика	**Umbraco**	**Laravel**	**Django**	**Node.js**	**Qt**
Мова програмування	C#	PHP	Python	JavaScript	C++
Архітектура	MVC	MVC	MVT	Event-Driven	Модульна
Ліцензія	Відкрита	Відкрита	Відкрита	Відкрита	Пропрієтарна та відкрита
Кросплатформність	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux
Масштабованість	Висока	Середня	Висока	Висока	Середня
Безпека	Вбудована (авторизація, локалізація)	Обмежена	Висока	Обмежена	Висока
Інтеграція з іншими системами	Висока (Azure, інші CMS)	Середня	Середня	Висока (npm, REST API)	Висока (вбудовані системи)
Простота у використанні	Висока	Середня	Низька	Середня	Низька
Активність спільноти	Середня	Висока	Висока	Висока	Середня

На основі аналізу мов програмування було встановлено, що C# виділяється серед інших своєю продуктивністю, масштабованістю та підтримкою об’єктно-орієнтованого підходу. Завдяки інтеграції з платформою .NET, мова забезпечує широкий набір можливостей для розробників. Python має переваги в наукових

обчисленнях і машинному навчанні, але його продуктивність у веб-розробці поступається C#. JavaScript є незамінним для створення інтерактивних веб-інтерфейсів, але його сфера обмежена. PHP добре підходить для простих веб-додатків, але не забезпечує такої ж масштабованості та безпеки, як C#. C++ залишається важливим для системного програмування, але його складність робить його менш зручним для веб-розробки.

Серед розглянутих фреймворків Umbraco є оптимальним вибором завдяки інтеграції з C# та платформою .NET. Laravel і Django пропонують зручність для розробників, але поступаються Umbraco в масштабованості та підтримці корпоративних рішень. Node.js забезпечує продуктивність у реальному часі, але не має таких розвинених інструментів для роботи з контентом, як Umbraco. Qt, хоч і потужний для вбудованих систем, менш підходить для веб-додатків.

C#, у поєднанні з ASP.NET і Umbraco, забезпечує оптимальні умови для створення сучасних веб-додатків. C# є універсальною мовою, що підтримує масштабованість і безпеку. ASP.NET надає широкий набір інструментів для розробки продуктивних і захищених додатків, а також інтеграцію з хмарними сервісами, такими як Azure. Umbraco додає до цієї комбінації зручність управління контентом, багатомовність та розширюваність завдяки інтеграції з іншими системами .NET.

Таким чином, вибір C#, ASP.NET та Umbraco базується на їхній здатності забезпечити високу продуктивність, гнучкість, безпеку та зручність у розробці, що відповідає сучасним вимогам корпоративних додатків і веб-платформ.

2.13 Вибір бази даних

Для розробки системи було прийнято рішення використовувати Microsoft SQL Server (MSSQL) як основну базу даних. MSSQL є потужною реляційною системою керування базами даних (RDBMS), розробленою компанією Microsoft. Вона забезпечує високу продуктивність, масштабованість і надійність, що робить

її оптимальним вибором для корпоративних додатків та складних систем управління даними.

MSSQL пропонує інтеграцію з платформою .NET, яка дозволяє ефективно використовувати її функції разом із іншими продуктами Microsoft. Вбудовані механізми шифрування та управління правами доступу гарантують високий рівень безпеки даних. Завдяки оптимізованій обробці запитів MSSQL забезпечує швидкий доступ до інформації навіть за умов значного навантаження. Її масштабованість дозволяє працювати з великими обсягами даних, забезпечуючи стабільність і продуктивність.

Система знаходить застосування в різних галузях, включаючи фінансову сферу, медицину та електронну комерцію. Наприклад, MSSQL використовується для зберігання транзакційних даних у банківських системах, де потрібна висока надійність і швидкість доступу. У сфері охорони здоров'я MSSQL допомагає забезпечувати збереження конфіденційної інформації, а в електронній комерції вона підтримує високі обсяги користувачів і швидкий обмін даними.

MSSQL виділяється своїми розширеними можливостями аналітики, що включають інструменти для створення звітів і аналізу даних. Крім того, автоматизоване резервне копіювання мінімізує ризик втрати інформації у разі збоїв, а підтримка складних запитів дозволяє ефективно працювати з великими наборами даних.

Порівняно з іншими системами, такими як MySQL, MSSQL забезпечує тісну інтеграцію з екосистемою Microsoft, що відкриває доступ до потужних хмарних сервісів Azure. Це робить MSSQL універсальним інструментом для розробки веб-додатків будь-якого рівня складності.

Вибір MSSQL як основної бази даних обумовлений її продуктивністю, безпекою та масштабованістю. Це рішення дозволяє реалізувати сучасні вимоги до зберігання та обробки даних, що є важливим для успішного виконання проєкту.[46]

2.14 Вибір доменного імені та його реєстрація

Процес вибору доменного імені є одним із важливих етапів створення веб-ресурсу, адже правильно обране ім’я позитивно впливає на впізнаваність сайту, його маркетингову привабливість і ефективність у досягненні цільової аудиторії. Для забезпечення успіху необхідно дотримуватися кількох ключових рекомендацій.

Перш за все, доменне ім’я повинно бути коротким, легким для запам’ятовування і читання. Імена, які чітко описують суть сайту або сервісу, є більш привабливими для користувачів і допомагають швидше асоціювати його з вмістом. Варто уникати складних символів, дефісів і цифр, які можуть ускладнювати введення адреси або створювати плутанину.

Вибір суфікса доменного імені також має велике значення. Він повинен відповідати напряму діяльності або територіальній належності ресурсу. Наприклад:

- .com — доменне ім’я зареєстроване комерційною організацією;
- .ua — хостинг в Україні, доступний лише для власників зареєстрованих торгових марок;
- .com.ua — домени для бізнес-структур, які працюють на території України;
- .net — домени, що часто використовуються інтернет-магазинами або технологічними компаніями;
- .org — глобальний домен верхнього рівня (gTLD), зазвичай асоційований із некомерційними організаціями.

Для успішного вибору доменного імені важливо перевірити його на унікальність і переконатися, що воно не збігається з відомою торговою маркою. Це допоможе уникнути правових конфліктів і зберегти репутацію вашого бренду.

Для реєстрації доменного імені необхідно звернутися до реєстратора доменів. Серед найпопулярніших сервісів — GoDaddy, Namecheap, Google Domains, чи локальні реєстратори, такі як «HOSTiQ» або «Український хостинг». Процедура реєстрації включає такі кроки:

- Перевірка доступності доменного імені. На сайті реєстратора вкажіть бажане ім’я та перевірте його доступність у вибраній доменній зоні.
- Вибір параметрів реєстрації. Після підтвердження доступності оберіть період реєстрації (зазвичай від 1 року) та додаткові послуги, такі як конфіденційність WHOIS.
- Оформлення заявки та оплата. Заповніть форму реєстрації, вказавши дані власника домену, та здійсніть оплату відповідно до обраного пакету послуг.
- Підтвердження реєстрації. Після успішної оплати та обробки заявки домен стає доступним для використання.

Дотримання цих рекомендацій і послідовна реалізація етапів реєстрації забезпечить вашому веб-ресурсу унікальну і впізнавану адресу, яка відповідатиме всім технічним і правовим вимогам. [10] - [14].

2.15 Хостинг для сайту

Веб-хостинг — це онлайн-сервіс, який забезпечує можливість публікації файлів вашого сайту в Інтернеті. Завдяки веб-хостингу, користувачі з усього світу можуть отримати доступ до сайту, використовуючи браузер. Для нової системи, розробленої на основі .NET 8, вибір хостингу є особливо важливим, оскільки потрібна інфраструктура, що підтримує специфічні вимоги цих технологій.

Вимоги до хостингу.

1. Сумісність із технологіями:
   - Підтримка серверної платформи Windows або Linux із встановленим .NET 8.
   - Наявність MSSQL для управління базами даних, що є критично важливим для роботи Umbraco.
   - Підтримка ASP.NET Core для ефективного виконання веб-додатків.
2. Продуктивність та надійність:
   - Висока пропускна здатність для обробки великих обсягів запитів.
   - Надійні механізми резервного копіювання даних, що забезпечують збереження контенту сайту.
3. Інструменти для розробників:
   - Доступ до середовища розробки, що підтримує CI/CD (безперервна інтеграція та доставка).
   - Підтримка інструментів для віддаленого адміністрування та моніторингу.

Для ефективної роботи системи на базі .NET 8 можна використовувати наступні типи хостингу:

- Спільний хостинг. Підходить для невеликих проектів із низьким рівнем навантаження. Це економічно вигідний варіант, однак спільні ресурси сервера можуть негативно впливати на продуктивність.
- Віртуальний приватний сервер (VPS). VPS надає виділені ресурси, що дозволяє забезпечити стабільну роботу системи. Цей варіант ідеально підходить для середніх за розміром проектів.
- Хмарний хостинг. Найкращий вибір для масштабованих додатків. Хмарний хостинг забезпечує високу гнучкість у розподілі ресурсів та підтримує інтеграцію з інструментами DevOps.

- Виділений сервер. Цей тип хостингу надає максимальну потужність і контроль, проте є найдорожчим варіантом. Він підходить для великих проектів із високими вимогами до безпеки та продуктивності.

Основні критерії вибору хостингу:

- Пропускна здатність і обсяг пам’яті. Для забезпечення стабільної роботи сайтів на Umbraco 13 необхідно звернути увагу на пропускну здатність та обсяг оперативної пам’яті. Це дозволить уникнути затримок у роботі системи під час високого навантаження.
- Інтегровані інструменти. Багато провайдерів хостингу пропонують вбудовані інструменти для моніторингу та управління сайтами, такі як панелі управління Plesk чи cPanel. Вони спрощують процес адміністрування.
- Резервне копіювання та безпека. Хостинг повинен забезпечувати регулярне резервне копіювання даних і використовувати сучасні механізми захисту від DDoS-атак, вірусів та інших загроз.

Серед найкращих варіантів для хостингу проектів на основі .NET 8 можна виділити:

- Microsoft Azure — ідеальний варіант для розгортання .NET-додатків. Пропонує масштабовану хмарну платформу з інтеграцією сервісів DevOps.
- AWS (Amazon Web Services) — забезпечує потужну хмарну інфраструктуру для підтримки великих проектів із глобальною аудиторією.
- SmarterASP.NET — спеціалізований хостинг для ASP.NET-додатків, що забезпечує оптимальну продуктивність.

- Umbraco Cloud — платформа, створена спеціально для роботи з Umbraco CMS. Вона пропонує вбудовану підтримку для CI/CD і зручний інтерфейс для розробників.

Правильний вибір хостингу дозволяє забезпечити високу продуктивність, надійність та масштабованість системи, що базується на .NET 8 та Umbraco 13. У результаті це сприятиме ефективній роботі веб-ресурсу та задоволенню потреб кінцевих користувачів.

3 ОСНОВНІ РІШЕННЯ ЩОДО РЕАЛІЗАЦІЇ КОМПОНЕНТІВ СИСТЕМИ

3.1 Постановка задачі

Автоматизована система моніторингу паркування є клієнт-серверною програмою, яка забезпечує контроль над паркувальними місцями, їхньою доступністю, а також візуалізацію інформації через електронні табло. Система зберігає статистичні дані про кількість транспортних засобів, які відвідують паркінг, і надає користувачам інструменти для зручного управління.

Необхідно розробити систему, яка складається з двох основних компонентів: панелі адміністратора та панелі оператора.

- Панель адміністратора повинна включати функції управління всіма аспектами системи, зокрема управління операторами, парковками, транспортними засобами, а також зареєстрованими користувачами.
- Панель оператора має надавати функціонал для реєстрації користувачів, реєстрації транспортних засобів та пошуку вільних паркувальних місць.

Система також повинна включати базу даних, яка містить схеми паркування, кількість паркувальних місць та інформацію про їхню доступність.

Система повинна відповідати таким вимогам:

- Реєстрація клієнтів та їх транспортних засобів.
- Збереження даних у базі даних на сервері.
- Забезпечення високої швидкості обробки даних.
- Наявність зручних інтерфейсів для операторів та адміністраторів.

На основі системних вимог було визначено основні сценарії використання, що ілюструються діаграмами варіантів використання (рис. 3.1).

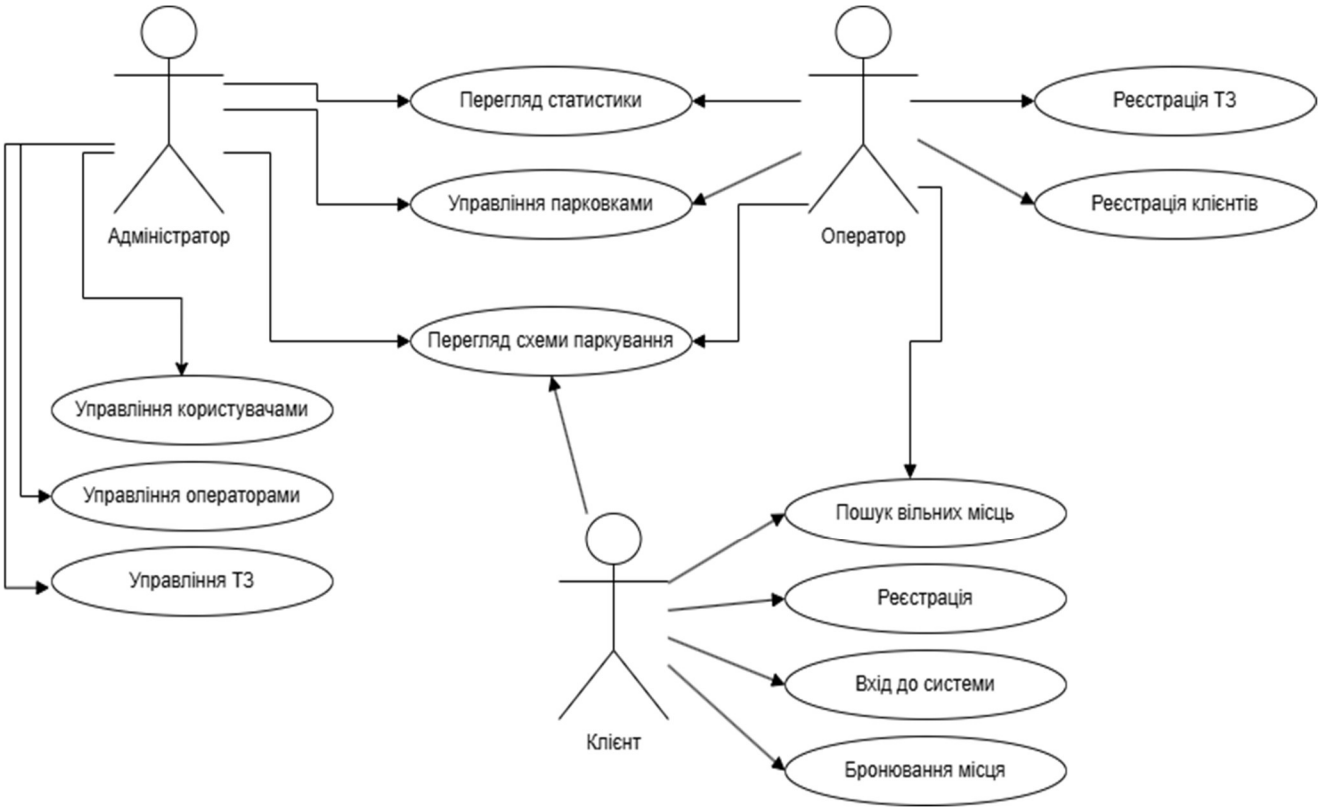


Рисунок 3.1 – Діаграма варіантів використання

Для реєстрації клієнта через офіційний сайт системи необхідно виконати такі кроки:

1. Відкрити офіційний сайт сервісу: http://parckingsystem-001-site1.ltempurl.com/
2. Натиснути кнопку "Реєстрація", яка доступна на кожній сторінці.
3. Ввести дані користувача.

Якщо оператор або охоронець вже зареєстрований у системі, процес реєстрації транспортного засобу здійснюється за спрощеною процедурою (рис. 3.2).

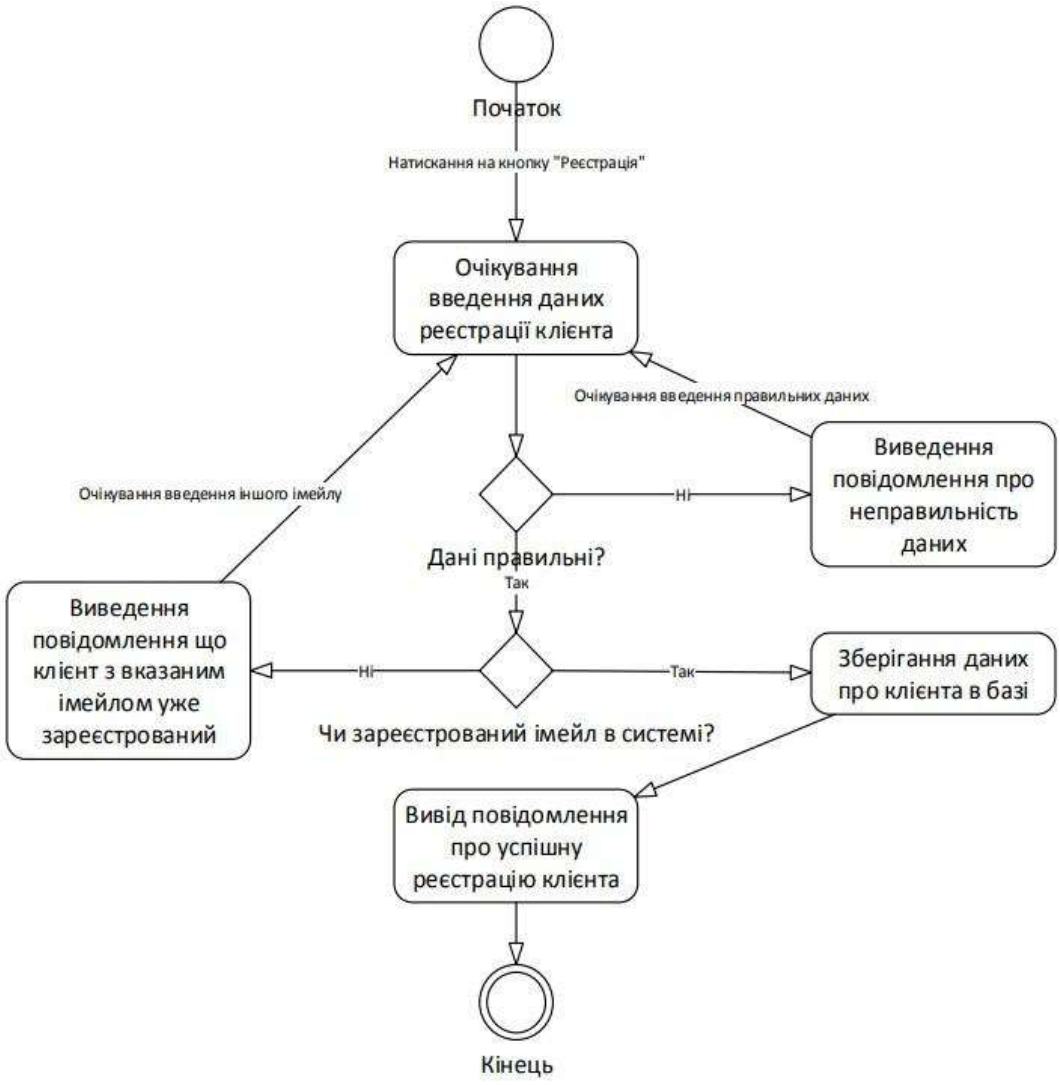


Рисунок 3.2 – Діаграма діяльності «Реєстрація клієнта»

Після входу в систему оператор може перейти до меню "Транспортні засоби" (рис. 3.3), де відображаються всі зареєстровані транспортні засоби. У цьому меню оператор може:

Редагувати інформацію про транспортний засіб, натиснувши кнопку "Редагувати", ввести нові дані та підтвердити зміни.

Видалити інформацію про транспортний засіб за допомогою кнопки "Видалити".

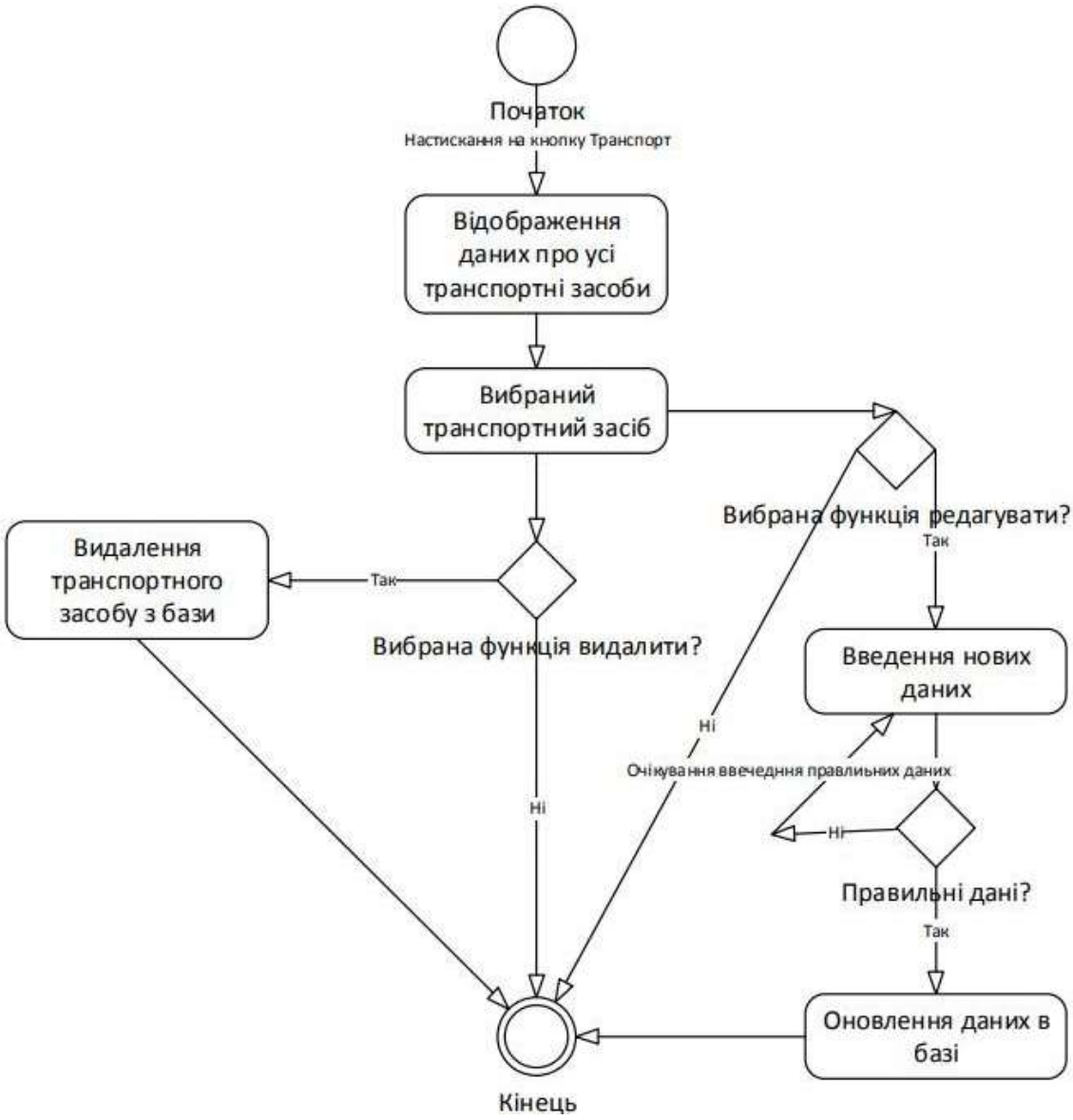


Рисунок 3.3 – Діаграма діяльності «Транспортні засоби»

3.2 Проєктування бази даних

У процесі створення автоматизованої системи управління паркуванням була розроблена схема бази даних, яка дозволяє ефективно організувати збереження і

обробку інформації. Схема відображає взаємозв’язки між ключовими сутностями системи (рис. 3.4).

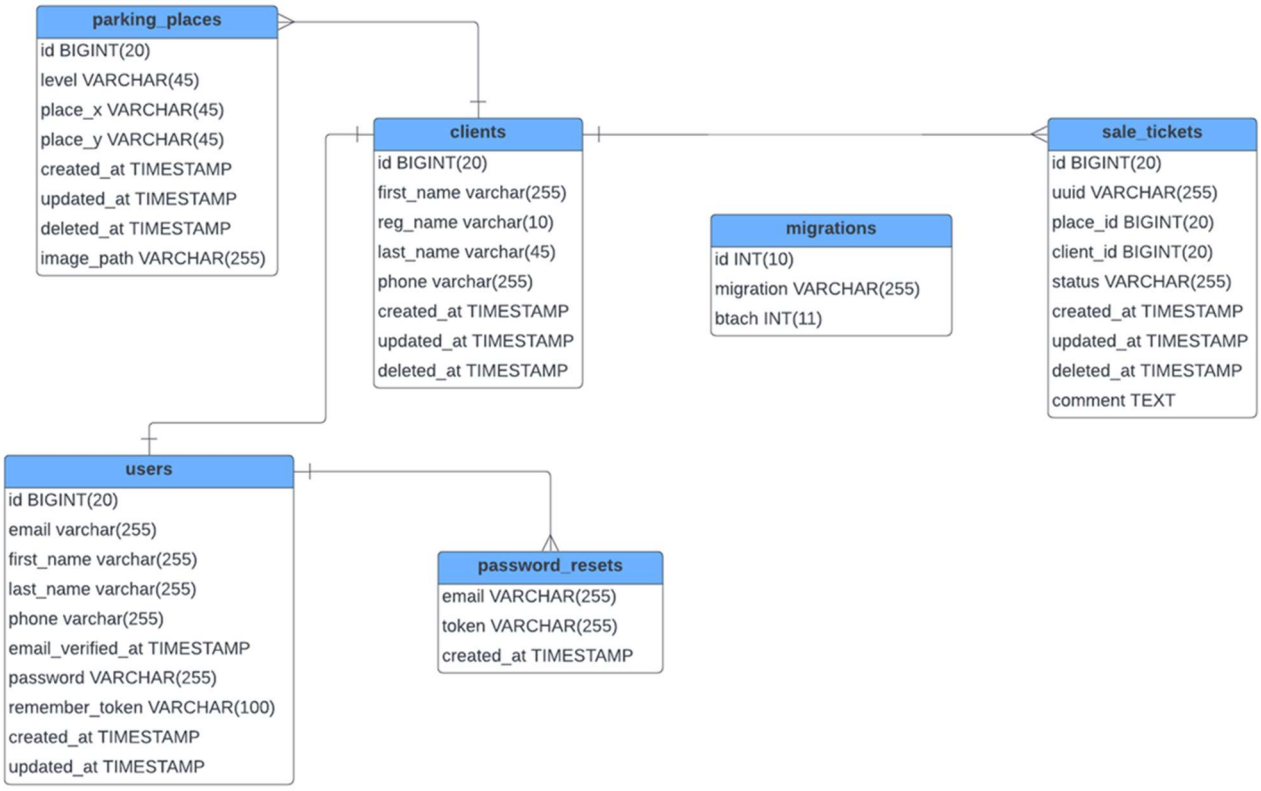


Рисунок 3.4 – Схема бази даних

Опис сутностей бази даних:

1. Parking_places:
   - Таблиця представляє список реальних паркувальних місць.
   - Має поля для визначення рівня паркінгу (стовпець — рівень) і координат місця (— place_x і place_y).
2. Sale_tickets:
   - Містить дані про активні сеанси паркування.
   - Поле place_id — зовнішній ключ, що посилається на таблицю Parking_places.
   - Поле client_id — зовнішній ключ із можливістю null, оскільки клієнт

може бути не зареєстрованим у системі.

3. Clients:
   - Таблиця зберігає список зареєстрованих клієнтів паркування.
4. Users:
   - Містить інформацію про користувачів системи (операторів або системних адміністраторів).
5. Migrations і Password_resets:
   - Службові таблиці системи, які використовуються для технічного забезпечення, зокрема для оновлення схеми бази даних і скидання паролів користувачів.

Розроблена база даних забезпечує:

- Ефективне управління паркувальними місцями: За допомогою таблиці Parking_places можна легко знаходити доступні місця та вести облік їх використання.
- Обробку даних клієнтів: Система дозволяє зберігати інформацію про зареєстрованих клієнтів, їхні контактні дані та історію паркувань.
- Гнучкість для незареєстрованих клієнтів: Поле client_id у таблиці Sale_tickets дозволяє працювати як із зареєстрованими, так і незареєстрованими клієнтами.
- Зручність адміністрування: Таблиця Users спрощує управління доступом операторів і адміністраторів до функцій системи.

Така структура бази даних дозволяє створити масштабовану, зручну у використанні та функціональну систему управління паркувальними місцями. Це забезпечує високу продуктивність і надійність роботи автоматизованої системи управління паркуванням.

3.3 Характеристики програми

Розроблена система автоматизації моніторингу та збору статистичних даних призначена для використання на автостоянках торгових центрів, загальних автостоянках або паркінгах. Програмне забезпечення побудоване на основі сучасних технологій ASP.NET і Umbraco, що забезпечує високу продуктивність, гнучкість і можливість інтеграції з іншими системами.

Завдяки використанню фреймворка ASP.NET, система гарантує швидкість обробки запитів і стабільну роботу навіть за умов великої кількості одночасних користувачів. Umbraco додає до цього інтуїтивно зрозумілий інтерфейс управління контентом, що дозволяє адміністраторам легко налаштовувати систему відповідно до потреб.

Програмний продукт спрощує реєстрацію транспортних засобів операторами або охороною на парковці, автоматизує вибір вільних місць і забезпечує інформативне відображення даних. Крім того, система підтримує доступ з будь-якого пристрою через веб-браузер, що значно підвищує її зручність.

4 РОЗГОРТАННЯ ТА ЕКСПЛУАТАЦІЯ

4.1 Призначення й умови використання

Розроблена система автоматизації моніторингу та збору статистичних даних призначена для використання на автостоянках торгових центрів, загальних автостоянках або паркінгах. Програмне забезпечення побудоване на основі сучасних технологій ASP.NET і Umbraco, що забезпечує високу продуктивність, гнучкість і можливість інтеграції з іншими системами.

Завдяки використанню фреймворка ASP.NET, система гарантує швидкість обробки запитів і стабільну роботу навіть за умов великої кількості одночасних користувачів. Umbraco додає до цього інтуїтивно зрозумілий інтерфейс управління контентом, що дозволяє адміністраторам легко налаштовувати систему відповідно до потреб.

Програмний продукт спрощує реєстрацію транспортних засобів операторами або охороною на парковці, автоматизує вибір вільних місць і забезпечує інформативне відображення даних. Крім того, система підтримує доступ з будь-якого пристрою через веб-браузер, що значно підвищує її зручність.

Таблиця 4.1 — Мінімальні апаратні вимоги

Компонент	**Мінімальна характеристика**
Операційна система	Windows 10
Процесор	Intel Core i3 або аналогічний
Оперативна пам’ять	4 ГБ
Місце на диску	20 ГБ
Мережеве підключення	Широкосмуговий Інтернет

Для обробки середнього навантаження (до 200 клієнтів на годину) необхідні вдосконалені апаратні ресурси, що відповідають характеристикам, наведеним у таблиці 4.2.

Таблиця 4.2 — Апаратні вимоги при середньому навантаженні

Компонент	**Характеристика**
Операційна система	Windows Server 2016/2019
Процесор	Intel Core i5 або аналогічний
Оперативна пам’ять	8 ГБ
Місце на диску	50 ГБ
Мережеве підключення	Широкосмуговий Інтернет

Для обробки великого навантаження (до 500–1000 клієнтів на годину) потрібні покращені апаратні ресурси, характеристики яких наведено в таблиці 4.3.

Таблиця 4.3 — Апаратні вимоги при великому навантаженні

Компонент	**Характеристика**
Операційна система	Windows Server 2019
Процесор	Intel Xeon або аналогічний
Оперативна пам’ять	16 ГБ
Місце на диску	100 ГБ (SSD)
Мережеве підключення	Широкосмуговий Інтернет

Основні характеристики системи:

- Кросплатформенність: Доступ через будь-який сучасний веб-браузер, незалежно від операційної системи пристрою.
- Інтеграція з базою даних: Використання MSSQL для забезпечення надійного зберігання даних.

- Гнучкість налаштувань: Завдяки Umbraco адміністраторам доступний зручний інтерфейс для керування контентом і користувачами.
- Швидкість обробки: ASP.NET забезпечує оптимізоване виконання запитів, що робить систему ідеальною для великих паркувальних комплексів.

Система дозволяє адміністраторам і операторам швидко отримувати необхідну інформацію для ефективного управління паркувальними майданчиками. Завдяки цьому підвищується продуктивність роботи персоналу та зменшуються витрати часу на обробку даних.

4.2 Вхідні та вихідні дані

Розроблена система автоматизації моніторингу та збору статистичних даних призначена для ефективного управління автостоянками в торгових центрах, загальних автостоянках або паркінгах. Система забезпечує обробку вхідних даних та надає корисну вихідну інформацію для покращення якості обслуговування клієнтів і оптимізації роботи паркінгу.

Вхідні дані.

1. Вхідні дані, які обробляє система, включають:
    - Анкета клієнта:
    - Ім’я клієнта.
    - Контактна інформація (телефон, електронна пошта).
2. Дані транспортного засобу (ТЗ):
    - Державний реєстраційний номер ТЗ.
    - Тип та марка автомобіля.
    - Колір транспортного засобу.
3. Заплановані параметри:
    - Час прибуття та запланований час зберігання.

- Спеціальні запити (наприклад, місце ближче до виходу).

4. Інформація про статус паркувальних місць:
    - Доступність місць у реальному часі.
    - Дані про зарезервовані місця.

Ці дані збираються через веб-інтерфейс системи або вводяться оператором вручну, після чого зберігаються у базі даних для подальшої обробки.

Вихідні дані.

Система генерує широкий спектр вихідної інформації, яка допомагає як клієнтам, так і адміністраторам:

1. Дані для клієнтів:
    - Інформація про доступне місце для паркування.
    - Статус бронювання.
    - Фактичний час зберігання ТЗ.
    - Вартість паркування, включаючи можливі знижки.
    - Час затримки платежу (за наявності).
2. Дані для адміністраторів:
    - Статистичні дані про використання паркувальних місць:
    - Інформація про прострочені платежі.
    - Звіти про доходи паркінгу.
3. Інтерактивна схема паркінгу:
    - Відображення статусу кожного місця в реальному часі (вільне, зайняте, зарезервоване).

Переваги роботи з даними:

- Оперативність: Система забезпечує миттєвий доступ до актуальних даних про стан паркування.
- Точність: Уникнення людських помилок завдяки автоматизації обліку.

- Аналіз: Генерація статистичних звітів допомагає оптимізувати роботу паркінгу та покращити планування.

Розроблена система забезпечує комплексну обробку вхідних і вихідних даних, що сприяє ефективному управлінню паркінгами та підвищенню рівня обслуговування клієнтів.

4.3 Повідомлення

Для забезпечення надійної роботи системи автоматизації моніторингу паркувальних місць особливу увагу приділено розробці механізмів обробки помилок і видачі повідомлень. У разі некоректних дій або збоїв у роботі програми користувач отримує інформативні попередження чи повідомлення про помилки. Це дозволяє швидко ідентифікувати проблему та запобігти її повторенню.

Повідомлення системи поділяються на кілька категорій залежно від їх призначення:

1. Інформаційні повідомлення
   - Служать для інформування користувачів про успішне виконання операцій (наприклад, успішна реєстрація або завершення бронювання).
   - Приклад: “Ваше бронювання успішно створено.”
2. Попередження
   - Використовуються для інформування про потенційні проблеми чи неправильне заповнення форм.
   - Приклад: “Заповніть усі обов’язкові поля перед продовженням.”
3. Повідомлення про помилки
   - Генеруються у разі критичних помилок, які перешкоджають виконанню операцій.
   - Приклад: “Помилка збереження даних. Перевірте підключення до бази даних.”
4. Системні повідомлення

1. Відображаються адміністраторам і стосуються внутрішніх проблем системи (наприклад, відсутність підключення до сервера або перевищення тайм-ауту запиту).

Типові діалогові вікна, що використовуються в системі, наведено в таблиці 4.4.

Таблиця 4.4 – Діалогові вікна

Тип діалогового вікна	**Приклад повідомлення**	**Дія користувача**
Інформаційне	“Дані успішно збережено.”	Натиснути “ОК”
Попередження	“Перевірте коректність введених даних.”	Виправити помилку та спробувати ще раз
Помилка	“Сталася помилка. Повторіть спробу пізніше.”	Натиснути “ОК” або звернутися до адміністратора
Системне	“З’єднання з сервером втрачено. Спробуйте підключитися знову.”	Відновити підключення

Обробка винятків є важливим елементом забезпечення стабільної роботи програми. До основних сценаріїв обробки винятків належать:

1. Некоректний ввід даних
    - Валідація на етапі введення забезпечує перевірку форматів полів (наприклад, електронної пошти, номерів телефонів).
2. Відсутність підключення до сервера
    - Система видає повідомлення про втрату з’єднання та повторює спробу автоматично через визначений час.
3. Помилки бази даних
    - У разі збою зберігання чи отримання даних система генерує повідомлення, а також створює запис у журналах для адміністратора.

Таким чином, у системі реалізовано інтуїтивно зрозумілу структуру повідомлень та механізми обробки помилок, що підвищує її надійність і зручність у використанні. Використання діалогових вікон допомагає користувачам швидко орієнтуватися в ситуаціях і ефективно вирішувати можливі проблеми.

4.4 Експлутація програмного продукту

Головний інтерфейс входу (рис. 4.1) використовується для авторизації та реєстрації користувачів, які використовують програмну систему для моніторингу та збору статистичних даних.

Для авторизації необхідно виконати такі дії:

- Ввести своє ім'я користувача (Username) та пароль (Password).
- Натиснути кнопку "Login" для підтвердження даних (рис. 4.1).

На цьому ж екрані доступна опція відновлення пароля для тих користувачів, які втратили доступ до свого облікового запису. Система пропонує зручний механізм відновлення через електронну пошту.

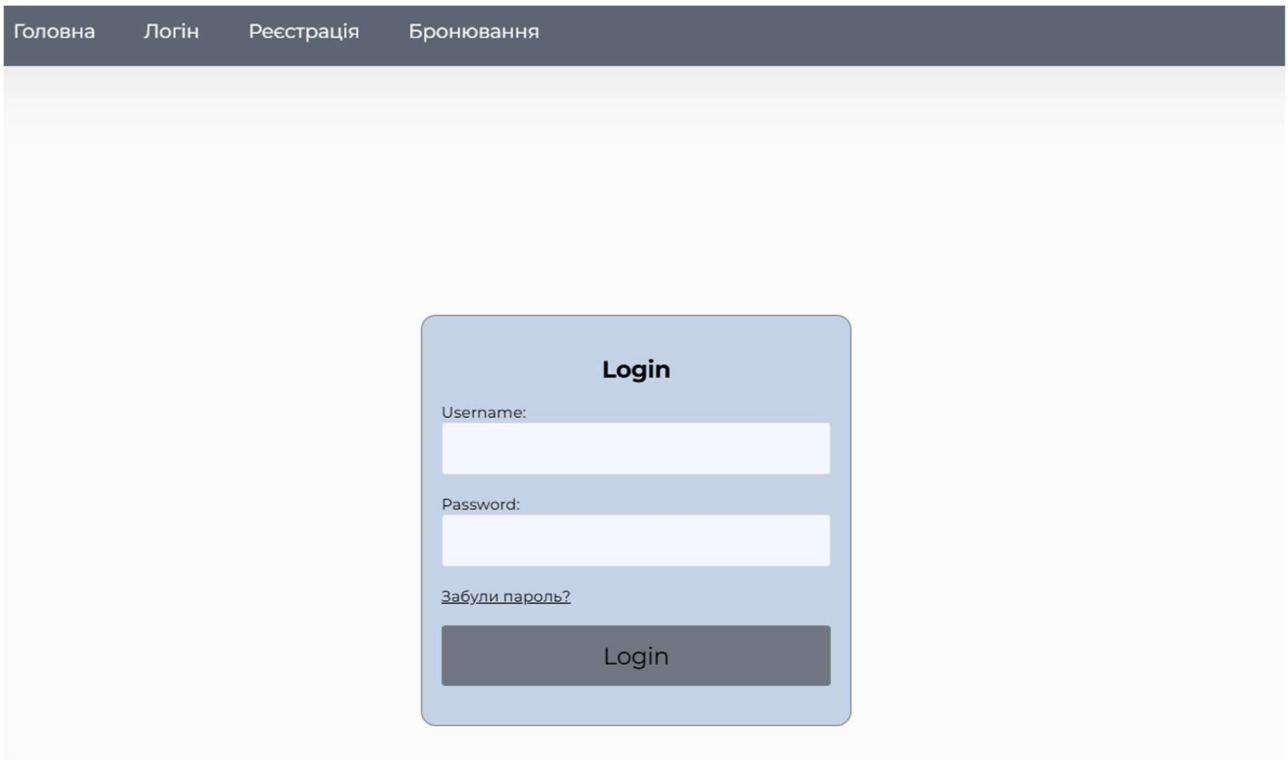


Рисунок 4.1 – Головний екран входу

Для реєстрації в системі передбачено просту й інтуїтивно зрозумілу форму (рис. 4.2). Користувач повинен заповнити такі поля:

- Нікнейм — ім’я користувача, яке буде використовуватися в системі.
- Електронна пошта — для зв'язку та відновлення пароля.
- Пароль — надійний пароль для захисту облікового запису.
- Номер транспортного засобу — для ідентифікації авто.
- Марка автомобіля — для детального обліку.
- Колір транспортного засобу — для візуальної ідентифікації.

Після введення всіх даних необхідно натиснути кнопку "Register" для завершення процесу реєстрації.

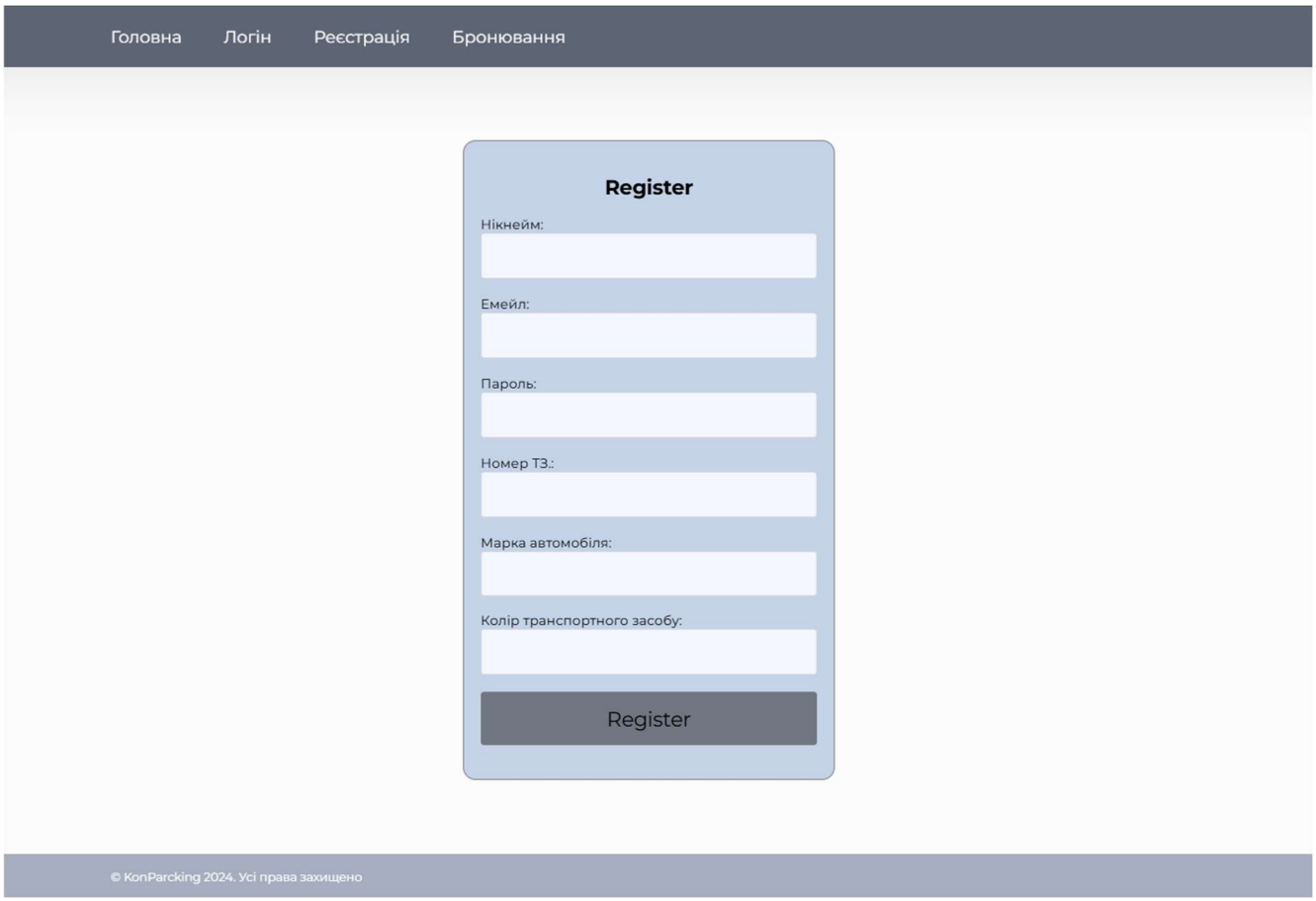


Рисунок 4.2 — Форма реєстрації нового користувача

Після завершення реєстрації користувач отримує на вказану електронну пошту лист із посиланням для активації облікового запису. Для завершення процесу необхідно:

- Відкрити електронний лист від системи.
- Перейти за вказаним у листі посиланням для підтвердження реєстрації.

У разі успішної активації облікового запису користувач отримує доступ до всіх функцій системи. Зокрема користувач може перейти на сторінку бронювання паркувального місця (рис. 4.3)

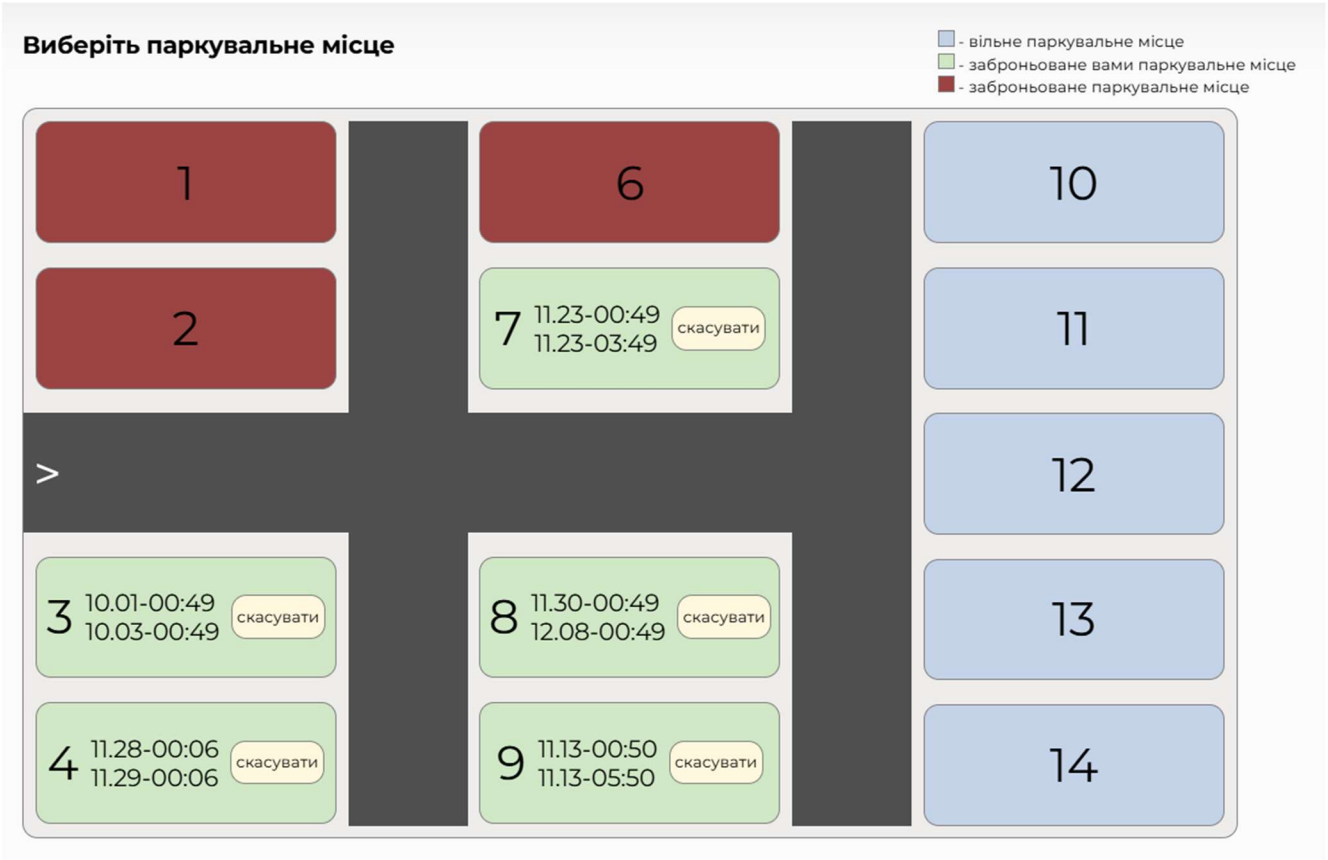


Рисунок 4.3 — Схема парковки

Дана сторінка призначена для користувачів, які використовують систему паркування для бронювання паркувальних місць. Її інтерфейс створено таким чином, щоб забезпечити зручність і зрозумілість процесу вибору та бронювання місця.

Основні функціональні можливості сторінки:

1. Відображення статусу паркувальних місць. Сторінка демонструє актуальний стан кожного паркувального місця у вигляді кольорових блоків:
   - Червоні блоки: Місця, що зайняті іншими користувачами або недоступні.
   - Зелені блоки: Заброньовані користувачем місця із зазначенням часових рамок бронювання.

- Сині блоки: Доступні для бронювання паркувальні місця.

2. Деталі забронюваних місць. Для кожного місця, яке користувач забронював, відображаються:
   - Номер паркувального місця.
   - Час початку та завершення бронювання.
   - Кнопка "Скасувати", яка дозволяє користувачеві зняти бронювання у разі зміни планів.

Сторінка оновлюється в реальному часі, щоб відображати найсвіжішу інформацію про доступність місць. Це дозволяє уникнути конфліктів між користувачами під час вибору місць.

Кожне паркувальне місце відображається як окремий блок у вигляді сітки.

У блоках зайнятих місць відображається лише їхній номер, а у блоках заброньованих користувачем місць — деталі бронювання та кнопка для скасування.

Використання сторінки:

- Вибір вільного місця: Користувач може натиснути на доступне (синє) місце, щоб перейти до процесу його бронювання.
- Перегляд заброньованих місць: Усі місця, заброньовані користувачем, позначені зеленим кольором. У цих блоках відображаються часові рамки бронювання та доступна кнопка для скасування бронювання.
- Скасування бронювання: Якщо користувач натискає кнопку “Скасувати”, система звільняє місце та оновлює статус паркінгу в реальному часі.

Після натискання на вільне паркувальне місце з’являється форма оформлення бронювання (рис 4.4).

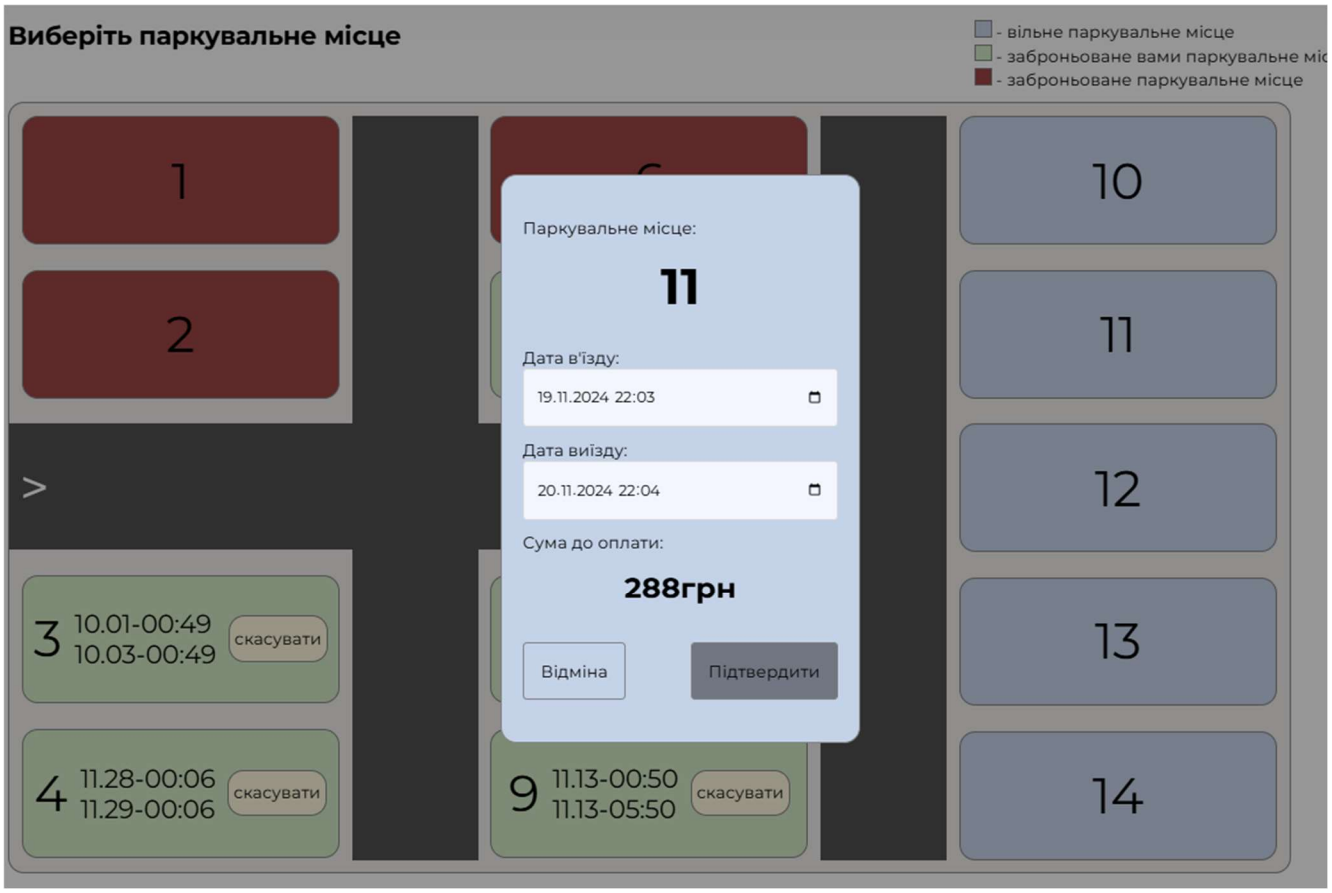


Рисунок 4.4 — Форма бронювання паркувального місця

Форма бронювання з'являється як модальне вікно після кліку на доступне паркувальне місце. Вона надає можливість користувачам легко здійснити бронювання, вказавши ключові параметри, та одразу побачити розрахунок вартості.

Інтерфейс форми містить номер вибраного місця, який розташований у верхній частині для зручності ідентифікації. Далі користувач може вказати дату та час початку і завершення бронювання через спеціальні поля. Ці поля оснащені зручним календарем-вибірником, що дозволяє швидко вибрати потрібний час і уникнути помилок у введенні даних.

Сума до оплати автоматично розраховується на основі введених параметрів і відображається в нижній частині форми. Це значення динамічно оновлюється при

зміні дати або часу. Завдяки цьому користувач одразу бачить підсумкову вартість бронювання без необхідності додаткових розрахунків.

Форма має дві кнопки: "Відміна" і "Підтвердити". Кнопка "Відміна" дозволяє закрити форму без збереження змін і повернутися до вибору паркувального місця. Кнопка "Підтвердити" завершує процес бронювання і зберігає введені дані в системі. Після підтвердження користувач отримує повідомлення про успішне бронювання.

Ця форма створена для забезпечення інтуїтивного і швидкого процесу взаємодії з системою. Простота її використання дозволяє навіть недосвідченим користувачам легко забронювати місце. Автоматичні розрахунки та зручний вибірник дат значно спрощують процес, мінімізуючи час, необхідний для заповнення форми. Усе це робить процес бронювання паркувальних місць максимально комфортним та ефективним.

Якщо увійти в систему під доступами адміністратора ми можемо поглянути на статистику паркування (рис. 4.5).

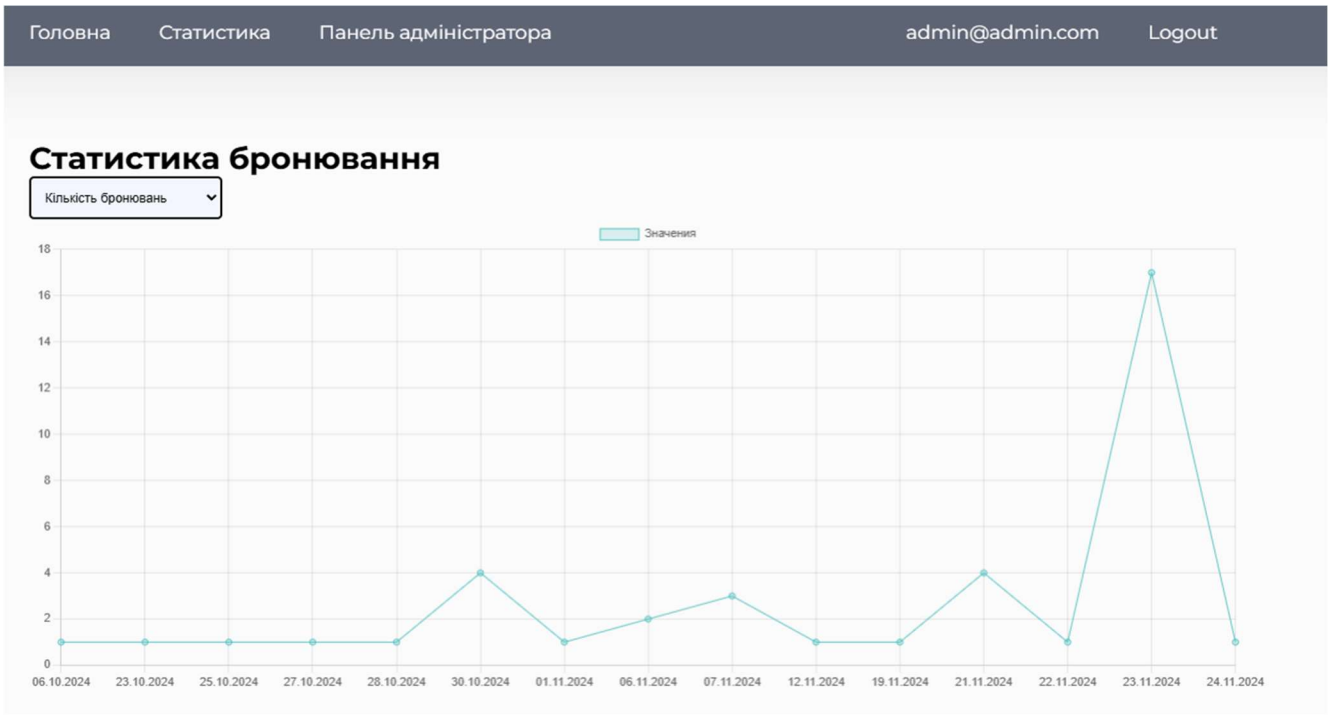


Рисунок 4.5 — Графік кількості бронювань

Цей тип графіка відображає динаміку кількості бронювань за днями. Для кожної дати вказується загальна кількість успішно зареєстрованих бронювань. Графік дозволяє аналізувати зміни у використанні системи протягом певного періоду, наприклад, виявляти дні з найбільшим і найменшим попитом на послуги паркування.

Вісі графіка:

- Горизонтальна вісь представляє дати, протягом яких були зареєстровані бронювання.
- Вертикальна вісь показує кількість бронювань, здійснених у кожний конкретний день.

Такий графік дозволяє виявити тенденції, пов'язані з підвищенням чи зниженням активності користувачів. Наприклад, зростання кількості бронювань у вихідні дні може свідчити про підвищений попит на паркувальні місця в цей період. Подібний аналіз корисний для планування роботи паркінгу, зокрема визначення потреби у додаткових ресурсах.

Далі користувач може вибрати наступний тип графіка для перегляду статистики по параметру тривалості бронювання (рис 4.6).

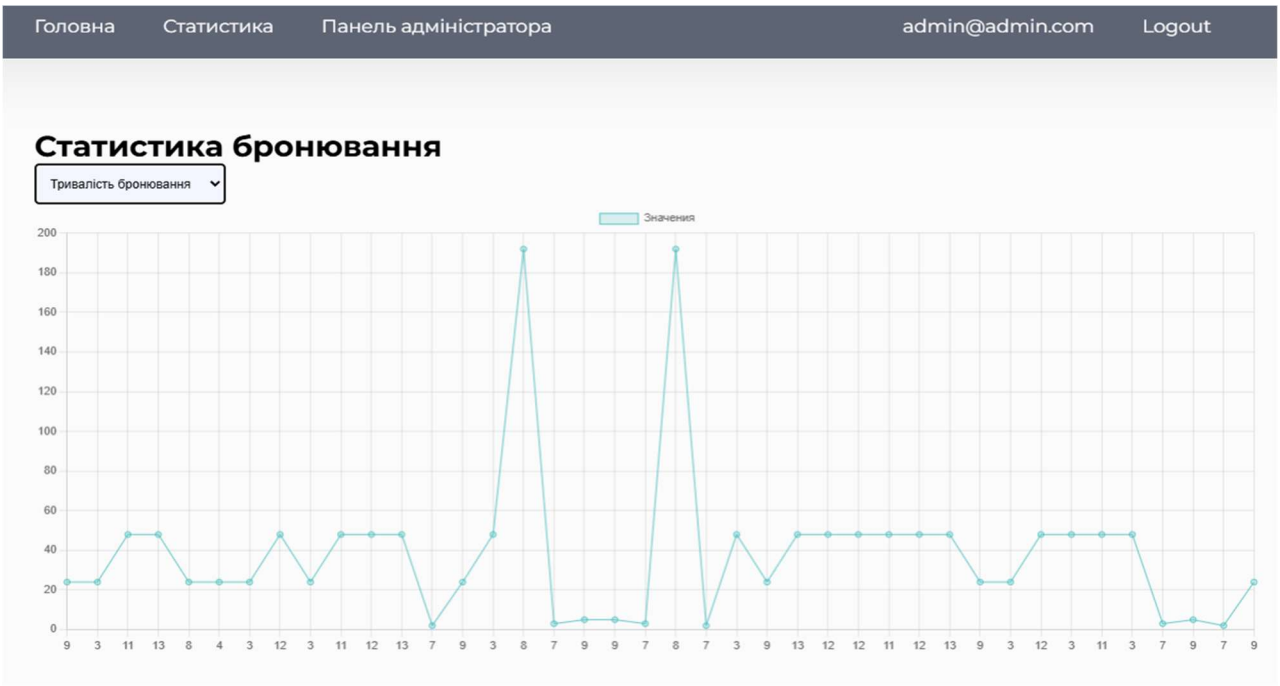


Рисунок 4.6 — Графік тривалості бронювання

На цьому графіку показано, як довго кожне паркувальне місце залишається зайнятим. Дані розраховуються на основі часу початку та завершення бронювання. Графік ілюструє середню тривалість використання паркувальних місць, а також допомагає виявити найбільш популярні місця.

Вісі графіка:

- Горизонтальна вісь відображає номери паркувальних місць.
- Вертикальна вісь демонструє тривалість бронювання у годинах.

Такий графік дозволяє адміністраторам паркінгу виявити місця, які найчастіше використовуються для довготривалого паркування. Це допомагає зрозуміти поведінку клієнтів і оптимізувати розподіл паркувальних місць, а також розробляти цінову політику залежно від тривалості паркування.

Далі поглянемо на наступний тип графіку в системі, графік кількості скасованих бронювань (рис 4.7).

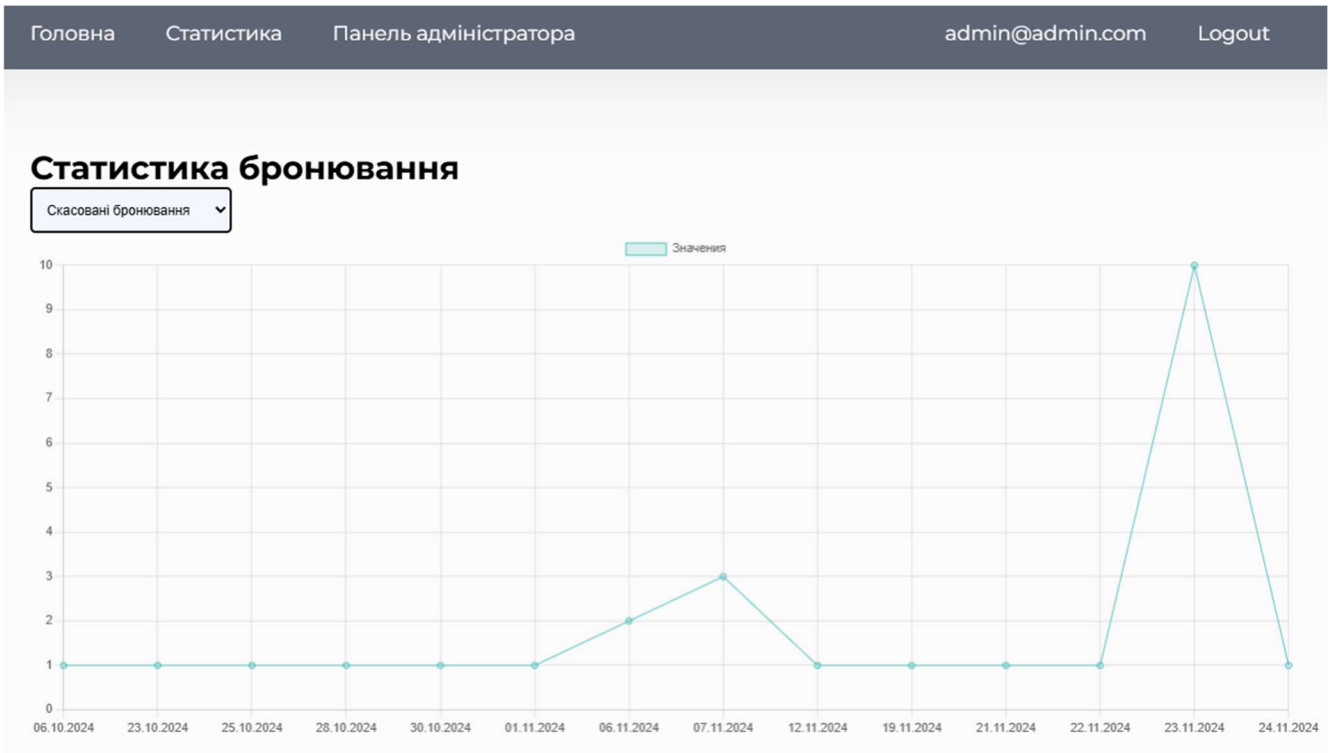


Рисунок 4.7 — Графік скасованих бронювання

Графік демонструє кількість бронювань, які були скасовані користувачами або адміністрацією протягом певного періоду. Він надає інформацію про частоту скасувань у різні дні, дозволяючи оцінити стабільність роботи системи та поведінку користувачів.

Вісі графіка:

- Горизонтальна вісь представляє дати, коли відбувалися скасування бронювань.
- Вертикальна вісь показує кількість скасованих бронювань за кожен день.

Цей графік дозволяє визначити періоди з найбільшою кількістю скасувань. Наприклад, велика кількість скасованих бронювань може свідчити про технічні проблеми в роботі системи, погану організацію паркінгу чи зовнішні фактори, такі як погодні умови. Зменшення рівня скасувань є важливим завданням для підвищення ефективності паркінгу.

Функціонал перегляду заброньованих місць включно із клієнтом та можливістю скасувати бронювання доступно адміністратору на наступній сторінці (рис 4.8).

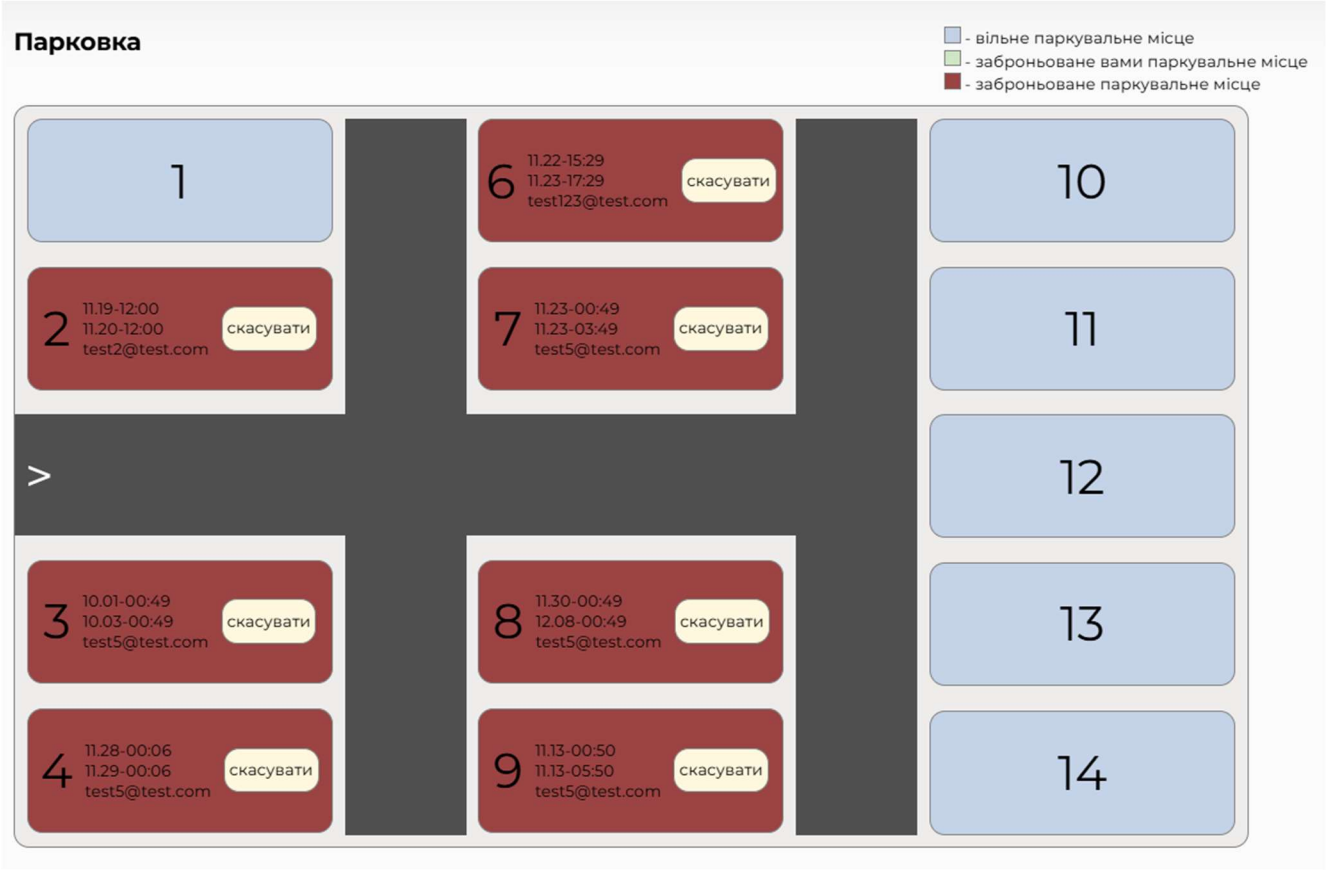


Рисунок 4.8 — Панель адміністратора парковки

Дана сторінка призначена для адміністратора системи паркування і дозволяє переглядати статус заброньованих паркувальних місць, а також керувати ними. Вона має зручний і зрозумілий інтерфейс, що дає змогу швидко отримати доступ до потрібної інформації та виконати необхідні дії.

Основні функціональні можливості сторінки:

1. Відображення статусу паркувальних місць
    - Паркувальні місця, зайняті забронюваними транспортними засобами, відображаються у вигляді червоних прямокутників з відповідною інформацією.
    - Вільні місця відзначені іншим кольором (світло-синім), що дає змогу адміністраторам швидко орієнтуватися в доступності паркувальних місць.
2. Інформація про бронювання. Для кожного забронюваного місця відображається:
    - Номер паркувального місця (у верхньому лівому куті блоку).
    - Дати та час початку і завершення бронювання.
    - Контактна інформація клієнта (email), що дозволяє за потреби зв'язатися з користувачем.
3. Можливість скасування бронювання
    - Кожне зайняте місце має кнопку “Скасувати”. Натиснувши на цю кнопку, адміністратор може зняти бронювання. Це особливо корисно у випадках помилок або за потреби звільнення місця для іншого клієнта.
    - Після скасування місце автоматично переходить у статус вільного, і його колір змінюється відповідно до статусу.

Якщо в url стрічці браузера ми в кінці назви сайту введемо /umbraco то зможемо потрапити на панель керування сайтом.(рис. 4.9, 4.10)

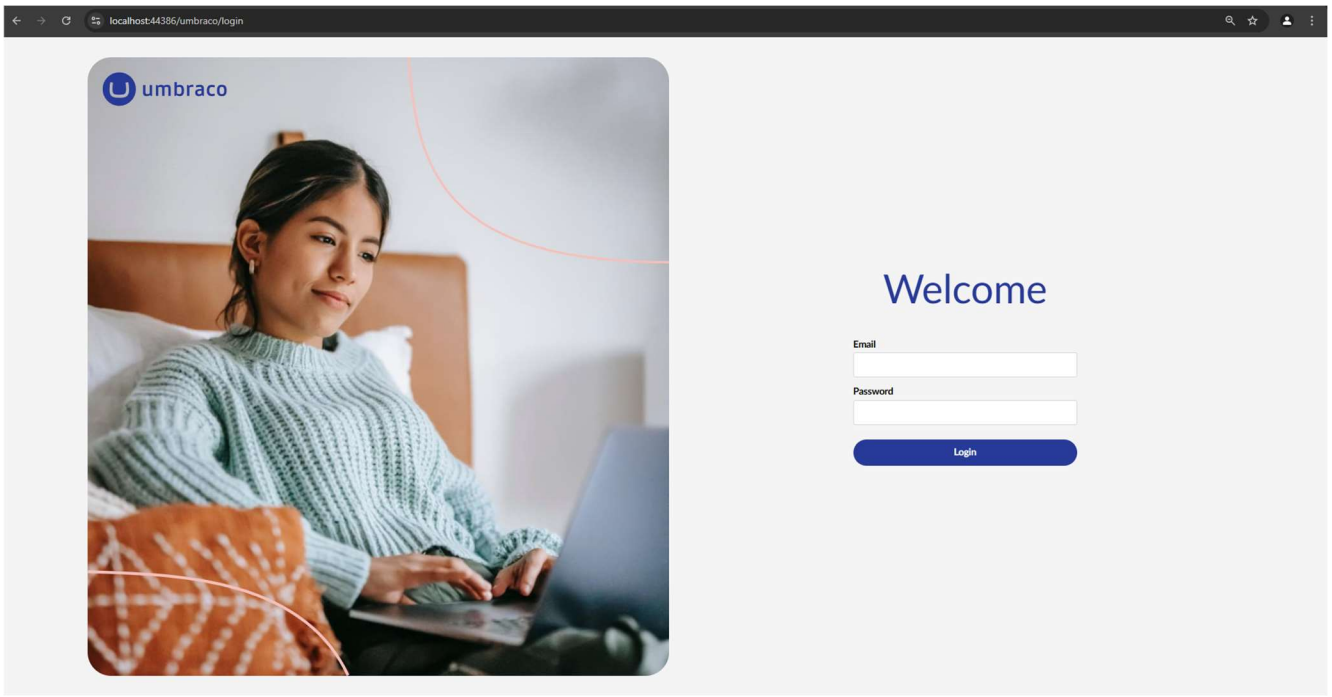


Рисунок 4.10 – Авторизація в бекофіс системи

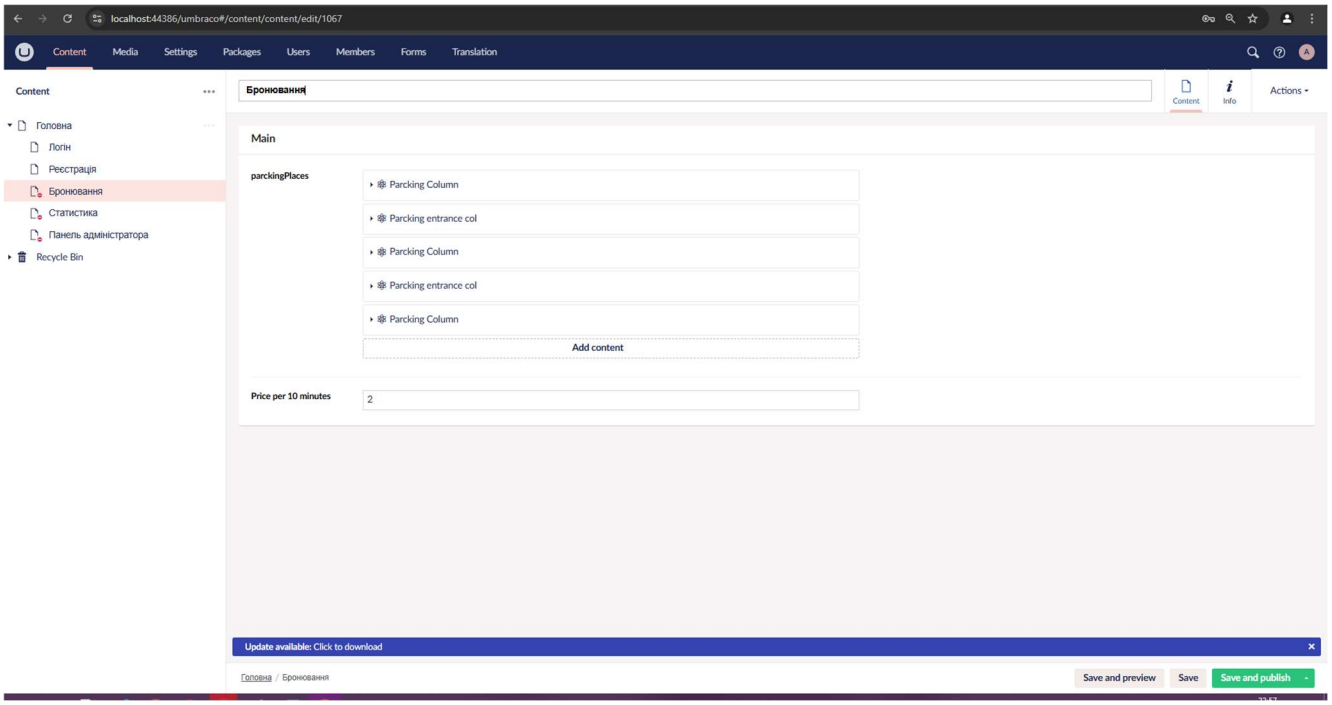


Рисунок 4.11 – бекофіс системи

Бекофіс системи представлений як інтуїтивно зрозумілий інтерфейс адміністратора, що забезпечує повний контроль над функціональністю сайту та його даними. Завдяки використанню CMS-системи Umbraco, адміністратор отримує широкий набір інструментів для управління контентом, структурою та динамічними елементами сайту.

- На бекофісній сторінці, представленій на Рис. 4.11, реалізовано можливості редагування та управління всіма важливими аспектами системи:
- Керування структурою парковки: адміністратор може редагувати конфігурацію парковки, додавати нові місця, змінювати статус існуючих паркувальних зон (наприклад, відображати, які зони доступні або закриті).
- Управління цінами та тарифами: через бекофіс є можливість встановлювати ціни на паркування, наприклад, ціну за 10 хвилин користування, як показано на зображенні.
- Редагування даних: інтеграція із CMS Umbraco дозволяє легко змінювати текстову або графічну інформацію на сайті.
- Моніторинг та аналіз статистики: через бекофіс адміністратор має доступ до даних про заброньовані місця, кількість відвідувачів, тривалість бронювань і скасовані операції.

Інтуїтивний інтерфейс CMS Umbraco забезпечує високий рівень зручності роботи адміністратора. Основне меню навігації розташоване ліворуч і включає категорії, такі як:

- Головна сторінка для перегляду загальних налаштувань.
- Логін та реєстрація для управління формами авторизації.
- Статистика для доступу до інтегрованих звітів і графіків.
- Сторінка адміністратора з усіма основними функціями контролю.

Функціонал бекофісу є гнучким і масштабованим, дозволяючи інтегрувати додаткові модулі або функції у разі потреби.

ВИСНОВКИ

У процесі виконання магістерської кваліфікаційної роботи виконано всі поставлені у вступі завдання, що дозволило досягти основної мети дослідження — розробки системи, яка забезпечує зменшення часу пошуку вільних місць, підвищення зручності для користувачів та оптимізацію управління паркувальними ресурсами.

Аналіз існуючих технологій та рішень, проведений у межах дослідження, дозволив визначити ключові вимоги до системи, серед яких — реєстрація користувачів і транспортних засобів, моніторинг паркувальних місць у реальному часі, захист даних, а також інтеграція із сучасними протоколами аутентифікації.

У результаті виконання роботи створено інтелектуальну систему моніторингу та управління паркувальними майданчиками, яка складається з адміністративної панелі та панелі оператора. Адміністративна панель реалізує функції управління користувачами, транспортними засобами та паркомісцями, а панель оператора дозволяє реєструвати клієнтів, транспортні засоби та здійснювати пошук вільних місць.

Розроблено оптимізовану структуру бази даних, яка забезпечує зберігання та обробку великих обсягів інформації, а також реалізовано модулі збору та аналізу статистики, що дозволяють використовувати дані для стратегічного планування та управління попитом. У системі впроваджено сучасні методи аутентифікації та шифрування для забезпечення безпеки даних.

Отримані результати мають потенціал для подальшого розвитку, зокрема впровадження мобільних додатків, розширення функціоналу статистики, інтеграції з системами розпізнавання номерних знаків та прогнозування зайнятості паркомісць. Таким чином, створена система відповідає сучасним вимогам до управління паркувальними майданчиками і може бути успішно впроваджена в різних організаціях..

ПЕРЕЛІК ПОСИЛАНЬ

1. Мобільне паркування [Електронний ресурс] / Комунальне підприємство «Київтранспарксервіс». – Режим доступу: https://ktps.kyiv.ua/services/pay/mobile-parking.

2. Parkinglogix [Electronic resource] / Parking Logix. – Access mode: https://parkinglogix.com/.

3. Simple Parking [Electronic resource] / Simple Airport Parking. – Access mode: http://www.simpleairportparking.com/.

4. SERVIO PARKING для автоматизації парковок [Електронний ресурс] / Expert Solution. – Режим доступу: https://expertsolution.com.ua/uk/servio-parking.

5. AllStojanka [Електронний ресурс] / Studwood. – Режим доступу: https://studwood.net/2339644/informatika/allstojanka.

6. R commander (Rcmdr) [Electronic resource] / R Commander. – Access mode: https://www.rcommander.com/.Субботін С. О., Олійник А. О. Інтелектуальні системи : навчальний посібник / під заг. ред. проф. С. О. Субботіна. – Запоріжжя: ЗНТУ, 2014. – 216 с.

7. Олійник А. О., Субботін С. О., Олійник О. О. Інтелектуальний аналіз даних. – Запоріжжя : ЗНТУ, 2012. – 278 с.

8. About Keras [Електронний ресурс] / Keras. – Режим доступу: https://keras.io/about/.

9. About Fastai [Електронний ресурс] / Fast.ai. – Режим доступу: https://www.fast.ai/about/.

10. Deafness and hearing loss [Електронний ресурс] / Всесвітня організація охорони здоров'я. – Режим доступу: https://www.who.int/news-room/fact-sheets/detail/deafness-and-hearing-loss.

11. Goodfellow I. J., Pouget-Abadie J., Mirza M., Xu B., Warde-Farley D., Ozair S., Courville A. C., Bengio Y. Generative Adversarial Nets. NIPS 2014: 2672-2680.

12. Chen L., Srivastava S., Duan Z., Xu C. Deep Cross-Modal Audio-Visual Generation. In Proceedings of Thematic Workshops'17, Mountain View, CA, USA, October 23–27, 2017, 9 pages.

13. Duarte et al. Wav2Pix: Speech-conditioned Face Generation Using Generative Adversarial Networks // ICASSP 2019 - 2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), 2019. – С. 8633–8637. – doi: 10.1109/ICASSP.2019.8682970.

14. Ramesh A., Pavlov M., Goh G., Gray S., Voss C., Radford A., Chen M., Sutskever I. "Zero-shot text-to-image generation," arXiv preprint arXiv:2102.12092, 2021.

15. SoundNet: Learning Sound Representations from Unlabeled Video [Електронний ресурс] / MIT Computer Science and Artificial Intelligence Laboratory. – Режим доступу: http://soundnet.csail.mit.edu/.

16. S. Reed, Z. Akata, H. Lee, and et al. Learning deep representations of fine-grained visual descriptions. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 49–58, 2016.

17. Devlin J., Chang M., Lee K., Toutanova K. BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding [Електронний ресурс]. – 2018. – Режим доступу: https://arxiv.org/abs/1810.04805.

18. What is Named Entity Recognition [Електронний ресурс] / Techslang. – Режим доступу: https://www.techslang.com/definition/what-is-named-entity-recognition-ner/.

19. spaCy 101: Everything you need to know [Електронний ресурс] / spaCy.io. – Режим доступу: https://spacy.io/usage/spacy-101.

20. NLTK book [Електронний ресурс] / Natural Language Toolkit. – Режим доступу: https://www.nltk.org/book/.

21. Piontko N. ner-ua [Електронний ресурс]. – Режим доступу: https://github.com/nazarii-piontko/ner-ua.

22. WordPiece [Електронний ресурс] / Papers with Code. – Режим доступу: https://paperswithcode.com/method/wordpiece.

23. What is Python? Executive Summary [Електронний ресурс] / Python Software Foundation. – Режим доступу: https://www.python.org/doc/essays/blurb/.

24. McCormick C. BERT Fine-Tuning Tutorial with PyTorch [Електронний ресурс]. – 2019. – Режим доступу: https://mccormickml.com/2019/07/22/BERT-fine-tuning/.

25. What Is SQLite? [Електронний ресурс] / SQLite Consortium. – Режим доступу: https://www.sqlite.org/index.html.Arora A. What is Focal Loss and when should you use it? [Електронний ресурс]. – Режим доступу: https://amaarora.github.io/2020/06/29/FocalLoss.html.

26. A Gentle Introduction to Cross-Entropy for Machine Learning [Електронний ресурс] / Machine Learning Mastery. – Режим доступу: https://machinelearningmastery.com/cross-entropy-for-machine-learning/.

27. Word2Vec: Як працювати з векторним відображенням слів [Електронний ресурс] / NeuroHive. – Режим доступу: https://neurohive.io/ua/osnovy-data-science/word2vec-vektornye-predstavlenija-slov-dlja-mashinnogo-obuchenija/.

28. Hugging Face Transformers Package – What Is It and How To Use It [Електронний ресурс] / KDnuggets. – Режим доступу: https://www.kdnuggets.com/2021/02/hugging-face-transformer-basics.html.

29. ukr-roberta-base [Електронний ресурс] / YouScan. – Режим доступу: http://book-online.com.ua/read.php?book=3628

30. Optimization [Електронний ресурс] / Hugging Face. – Режим доступу: https://huggingface.co/transformers/main_classes/optimizer_schedules.html.

31. Learning Rate Schedules (Pytorch) [Електронний ресурс]. – Режим доступу: https://huggingface.co/transformers/main_classes/optimizer_schedules.html#learning-rate-schedules-pytorch.

32. Catalyst [Електронний ресурс] / Catalyst Team. – Режим доступу: https://github.com/catalyst-team/catalyst.

33. Хащтамов А. Poetry: новий менеджер залежностей в Python [Електронний ресурс]. – Режим доступу: https://khashtamov.com/ru/python-poetry-dependency-management/.

34. Coding Neural Network — Forward Propagation and Backpropagation [Електронний ресурс] / Towards Data Science. – Режим доступу: https://towardsdatascience.com/coding-neural-network-forward-propagation-and-backpropagtion-ccf8cf369f76.

35. Olah C. Understanding LSTM Networks [Електронний ресурс]. – Режим доступу: https://colah.github.io/posts/2015-08-Understanding-LSTMs/.

36. Stroustrup B. The C++ Programming Language (4th Edition). Addison-Wesley, 2013. – 1366 p.

37. Laravel [Електронний ресурс] / Laravel Documentation. – Режим доступу: https://laravel.com/docs/5.1/releases.

38. Laravel. Blade Part 1 [Електронний ресурс] / Traversy Media. – Режим доступу: https://www.youtube.com/watch?v=jJKM2rc_pO8.

39. Django makes it easier to build better web apps more quickly and with less code [Електронний ресурс] / Django Software Foundation. – Режим доступу: https://www.djangoproject.com/.

40. Django [Електронний ресурс] / Habr. – Режим доступу: https://habr.com/hub/django/.

41. Node.JS [Електронний ресурс] / Habr. – Режим доступу: https://habr.com/hub/nodejs/.

42. ASP.NET Core Guide [Електронний ресурс] / Microsoft. – Режим доступу: https://learn.microsoft.com/aspnet/core.

43. Getting Started with Umbraco CMS [Електронний ресурс] / Umbraco. – Режим доступу: https://umbraco.com/documentation/.

44. MySQL Documentation [Електронний ресурс] / Oracle Corporation. – Режим доступу: https://dev.mysql.com/doc/.

45. Паркування [Електронний ресурс] / ParkingPay. – Режим доступу: https://parkingpay.com.ua.

46. Конотопенко В. Інтелектуальна система обліку транспортних засобів на паркувальному майданчику [Електронний ресурс]. – Режим доступу: https://conferences.vntu.edu.ua/index.php/all-fksa/all-fksa-2024/paper/view/20876.

47. Методичні вказівки до виконання магістерських кваліфікаційних робіт для студентів спеціальності 174 «Автоматизація, комп’ютерно-інтегровані технології та робототехніка» (кафедра комп'ютерних систем управління) [Електронний ресурс] / уклад. В. М. Дубовой. – Вінниця: ВНТУ, 2023. – (PDF, 45 с.).

ДОДАТКИ

Додаток А
(обов'язковий)

ПРОТОКОЛ ПЕРЕВІРКИ НАВЧАЛЬНОЇ (КВАЛІФІКАЦІЙНОЇ) РОБОТИ

Назва роботи: « Інтелектуальна система обліку транспортних засобів на паркувальному майданчику »

Тип роботи: магістерська кваліфікаційна робота
(кваліфікаційна робота, курсовий проект (робота), реферат, аналітичний огляд, інше (зазначити))

Підрозділ кафедра комп'ютерних систем управління
(кафедра, факультет (інститут), навчальна група)

Керівник Ковалюк О.О., доцент кафедри КСУ
(прізвище, ініціали, посада)

Показники звіту подібності

Turnitin	
Оригінальність	99%
Загальна схожість	1%

Аналіз звіту подібності (відмітити потрібне)

√ Запозичення, виявлені у роботі, оформлені коректно і не містять ознак плагіату.

☐ Виявлені у роботі запозичення не мають ознак плагіату, але їх надмірна кількість викликає сумніви щодо цінності роботи і відсутності самостійності її автора. Роботу направити на доопрацювання.

☐ Виявлені у роботі запозичення є недобросовісними і мають ознаки плагіату та/або в ній містяться навмисні спотворення тексту, що вказують на спроби приховування недобросовісних запозичень.

Заявляю, що ознайомлений (-на) з повним звітом подібності, який був згенерований Системою щодо роботи (додається)

Автор _____ Віктор КОНОТОПЕНКО
(підпис) (прізвище, ініціали)

Опис прийнятого рішення

_____
_____
_____
_____

Особа, відповідальна за перевірку _____ Володимир ДУБОВОЙ
(підпис) (прізвище, ініціали)

Експерт _____ _____
(за потреби) (підпис) (прізвище, ініціали, посада)

Додаток Б – Технічне завдання

(обов'язковий)

Вінницький національний технічний університет

ЗАТВЕРДЖЕНО
Зав. кафедри КСУ,
д.т.н., професор
_____В'ячеслав КОВТУН

"14" 10 2024 р

ТЕХНІЧНЕ ЗАВДАННЯ
на виконання магістерської кваліфікаційної роботи
Інтелектуальна система обліку транспортних засобів на паркувальному майданчику.

08-33.МКР.05.00.000 ТЗ

Студент групи ЗАКІТР-23м

_____ Віктор КОНОТОПЕНКО
Підпис *Ім'я ПРІЗВИЩЕ*

Керівник к.т.н., доцент кафедри

_____ Олег КОВАЛЮК
Підпис *Ім'я ПРІЗВИЩЕ*

Вінниця 2024

1. Назва та галузь застосування

1.1. Назва – Інтелектуальна система обліку транспортних засобів на паркувальному майданчику.

1.2. Галузь застосування – Облік, моніторинг та збір статистичних даних про транспорт.

2. Підстава для проведення розробки.

Тема магістерської кваліфікаційної роботи затверджена наказом по ВНТУ від від «17» вересня 2024 року №310.

3. Мета та призначення розробки.

Метою магістерської кваліфікаційної роботи є роботи є дослідження методів моніторингу та збору статистичних даних про паркування та створення автоматизованої системи для моніторингу та збору статистичних даних про паркування.

4. Джерела розробки.

Магістерська кваліфікаційна робота виконується вперше. В ході проведення розробки повинні використовуватись такі документи:

1.Мобільне паркування [Електрон. ресурс]. – Режим доступу: https://ktps.kyiv.ua/services/pay/mobile-parking.

2.Laravel [Electronic resource]. – Access mode: https://laravel.com/docs/5.1/releases.

3.What Is SQLite? [Електронний ресурс] – Режим доступу до ресурсу: https://www.sqlite.org/index.html.

5. Вимоги до розробки.

5.1. Перелік головних функцій:

- з’єднання з віддаленим сервером;
- збір статистики паркування;
- відображення статистики паркування адміністратору;

5.2. Основні технічні вимоги до розробки.

5.2.1. Вимоги до програмної платформи:

- WINDOWS 10, Linux, Windows Server;
- .NET 8;

5.3. Умови експлуатації системи:

- безперебійне цілодобове функціонування системи;
- регулярне оновлення та підтримка актуальності даних
- надійний захист даних

6. Стадії та етапи розробки.

1. Аналіз предметної області «24» жовтня 2024 р.
2. Огляд сучасних методів розв’язання технічної задачі «27» жовтня 2024 р.
3. Аналіз функцій систем та проєктування архітектури автоматизованої системи «01» листопада 2024 р.
4. Розробка UML-діаграм системи «03» листопада 2024 р.
5. Програмна реалізація системи «11» листопада 2024 р.
6. Тестування програмного забезпечення «14» листопада 2024 р
7. Оформлення пояснювальної записки «19» листопада 2024 р
8. Захист МКР «11» грудня 2024 р

7. Порядок контролю і приймання.

7.1. Хід виконання роботи контролюється керівником роботи. Рубіжний контроль провести до «20» листопада 2024 р.

7.2. Атестація МКР здійснюється на попередньому захисті. Попередній захист магістерської кваліфікаційної роботи провести до «1» грудня 2024 р.

7.3. Підсумкове рішення щодо оцінки якості виконання роботи приймається на засіданні ЕК. Захист магістерської кваліфікаційної роботи провести до «7» грудня 2024 р.

Додаток В (вибірковий) – Лістинги

ParckingController.cs (головний бекенд функціонал)

```
[Route("umbraco/api/[controller]")]
 [ApiController]
 public class ParckingController : ControllerBase
 {
    private readonly MemberManager _memberManager;
    private readonly IContentService _contentService;
    private readonly IUmbracoContextAccessor _umbracoContextAccessor;
    private readonly IPublishedContentQuery _publishedContentQuery;
    private readonly IContentTypeService _contentTypeService;
    private readonly IPublishedSnapshotAccessor _publishedSnapshotAccessor;


    public ParckingController(
       IContentService contentService,
       IUmbracoContextAccessor umbracoContextAccessor,
       MemberManager memberManager,
       IPublishedContentQuery publishedContentQuery,
       IContentTypeService contentTypeService,
       IPublishedSnapshotAccessor publishedSnapshotAccessor)
    {
       _memberManager = memberManager;
       _contentService = contentService;
       _umbracoContextAccessor = umbracoContextAccessor;
       _publishedContentQuery = publishedContentQuery;
       _contentTypeService = contentTypeService;
       _publishedSnapshotAccessor = publishedSnapshotAccessor;
    }


    [HttpPost("order")]
    public async Task<IActionResult> OrderPlace(OrderParckingModel model)
    {
       try
       {
          var startDate = DateTime.Parse(model.StartDate);
          var endDate = DateTime.Parse(model.EndDate);
          var content = _contentService.GetById(Guid.Parse(model.ParckingLot));
          var user = await _memberManager.FindByIdAsync(model.MemberKey);
          if (content == null)
             return NotFound();
          var rawValue = content.GetValue("parckingPlaces");
          if (rawValue == null)
             return BadRequest("No parking places found");
```

```
var mainData = JsonConvert.DeserializeObject<dynamic>(rawValue.ToString());
bool isUpdated = false;

foreach (var column in mainData.contentData)
{
   if (column?.parckingPlaces != null)
   {

      var parkingPlacesJson = column.parckingPlaces.ToString();
      var parkingPlacesData = JsonConvert.DeserializeObject<dynamic>(parkingPlacesJson);
      foreach (var place in parkingPlacesData.contentData)
      {
         Guid Udi = Guid.Parse(place.udi.ToString().Split('/')[3].ToString());
         if (Udi == Guid.Parse(model.ParckingId))
         {
            place.bookedBy = user.Id;
            place.bookedStart = startDate.ToString("o", CultureInfo.InvariantCulture);
            place.bookedEnd = endDate.ToString("o", CultureInfo.InvariantCulture);
            place.isBooked = true;
            column.parckingPlaces = JsonConvert.SerializeObject(parkingPlacesData);
            isUpdated = true;
            break;
         }
      }
      if (isUpdated)
         break;
   }

}
content.SetValue("parckingPlaces", JsonConvert.SerializeObject(mainData));
_contentService.SaveAndPublish(content);
var context = _umbracoContextAccessor.GetRequiredUmbracoContext();
var pageId = context.Content.GetAtRoot().DescendantsOrSelf<BookingStatistic>().FirstOrDefault().Key;
var contentStatic = _contentService.GetById(pageId);
if (contentStatic == null)
   return NotFound($"Content with ID {pageId} not found");
var rawValueSt = contentStatic.GetValue("bookingStatisticList3");
dynamic mainDataSt;


if (rawValueSt == null)
{
   mainDataSt = new ExpandoObject();
   mainDataSt.layout = new ExpandoObject();
   mainDataSt.layout.Umbraco = new ExpandoObject();
   mainDataSt.layout.Umbraco.BlockList = new List<dynamic>();
   mainDataSt.contentData = new List<dynamic>();
   mainDataSt.settingsData = new List<dynamic>();
```

```
        }
        else
        {
            mainDataSt = JsonConvert.DeserializeObject<ExpandoObject>(rawValueSt.ToString());
        }
        var layout = (IDictionary<string, object>)mainDataSt.layout;
        var layoutList = (List<dynamic>)layout["Umbraco.BlockList"];
        var newUdi = $"umb://element/{Guid.NewGuid().ToString("N")}";
        layoutList.Add(new
        {
            contentUdi = newUdi
        });


        var contentDataList = (List<dynamic>)mainDataSt.contentData;
        var typeKey = StatisticItem.GetModelContentType(_publishedSnapshotAccessor).Key;
        contentDataList.Add(new
        {
            contentTypeKey = typeKey,
            udi = newUdi,
            parckingNumber = model.PlaceNumber,
            bookedBy = $"umb://member/{user.Key}",
            bookingStart = startDate.ToString("o", CultureInfo.InvariantCulture),
            bookingEnd = endDate.ToString("o", CultureInfo.InvariantCulture),
            bookingTime = DateTime.Now.ToString("o", CultureInfo.InvariantCulture),
            isCancel = false
        });
        var updatedJson = JsonConvert.SerializeObject(mainDataSt);
        contentStatic.SetValue("bookingStatisticList3", updatedJson);
        var result = _contentService.SaveAndPublish(contentStatic);
        return Ok();
    }
    catch(Exception ex)
    {
        var tt = ex;
    }
    return BadRequest();
}


[HttpPost("cancel")]
public async Task<IActionResult> CancelOrder(CancelOrderModel model)
{
    try
    {
        var context = _umbracoContextAccessor.GetRequiredUmbracoContext();
        var contetnCarBooking = (context?.Content?
            .GetById(Guid.Parse(model.ParckingLot)) as CarBooking)
```

```
    .ParckingPlaces?
    .Where(x => x.Content is ParckingColumn)?
    .Select(x => x.Content as ParckingColumn);
var user = await _memberManager.FindByIdAsync(model.MemberKey);
var page = context.Content.GetAtRoot().DescendantsOrSelf<BookingStatistic>().FirstOrDefault();
ParckingPlaceItem bookingItem = null;

foreach(var item in contetnCarBooking)
{
    if(bookingItem == null)
    {
        foreach(var row in item.ParckingPlaces.Where(x => x.Content is ParckingPlaceItem).Select(x => x.Content as ParckingPlaceItem))
        {
            if(row.Key == Guid.Parse(model.ParckingId))
            {
                bookingItem = row;
                break;
            }
        }
    }
}
var contentStatic = _contentService.GetById(page.Key);
if (contentStatic == null)
    return NotFound($"Content with ID {page.Key} not found");
var rawValueSt = contentStatic.GetValue("bookingStatisticList3");
dynamic mainDataSt;
if (rawValueSt == null)
{
    mainDataSt = new ExpandoObject();
    mainDataSt.layout = new ExpandoObject();
    mainDataSt.layout.Umbraco = new ExpandoObject();
    mainDataSt.layout.Umbraco.BlockList = new List<dynamic>();
    mainDataSt.contentData = new List<dynamic>();
    mainDataSt.settingsData = new List<dynamic>();
}
else
{
    mainDataSt = JsonConvert.DeserializeObject<ExpandoObject>(rawValueSt.ToString());
}
var layout = (IDictionary<string, object>)mainDataSt.layout;
var layoutList = (List<dynamic>)layout["Umbraco.BlockList"];
var newUdi = $"umb://element/{Guid.NewGuid().ToString("N")}";
layoutList.Add(new
{
    contentUdi = newUdi
});
var contentDataList = (List<dynamic>)mainDataSt.contentData;
var typeKey = StatisticItem.GetModelContentType(_publishedSnapshotAccessor).Key;
```

```
contentDataList.Add(new
{
   contentTypeKey = typeKey,
   udi = newUdi,
   parckingNumber = bookingItem.Number,
   bookedBy = $"umb://member/{user.Key}",
   bookingStart = bookingItem.BookedStart.ToString("o", CultureInfo.InvariantCulture),
   bookingEnd = bookingItem.BookedEnd.ToString("o", CultureInfo.InvariantCulture),
   bookingTime = DateTime.Now.ToString("o", CultureInfo.InvariantCulture),
   isCancel = true
});
var updatedJson = JsonConvert.SerializeObject(mainDataSt);
contentStatic.SetValue("bookingStatisticList3", updatedJson);
var result = _contentService.SaveAndPublish(contentStatic);
var content = _contentService.GetById(Guid.Parse(model.ParckingLot));
if (content == null)
   return NotFound();
var rawValue = content.GetValue("parckingPlaces");
if (rawValue == null)
   return BadRequest("No parking places found");
var mainData = JsonConvert.DeserializeObject<dynamic>(rawValue.ToString());
bool isUpdated = false;
foreach (var column in mainData.contentData)
{
   if (column?.parckingPlaces != null)
   {
      var parkingPlacesJson = column.parckingPlaces.ToString();
      var parkingPlacesData = JsonConvert.DeserializeObject<dynamic>(parkingPlacesJson);
      foreach (var place in parkingPlacesData.contentData)
      {
         Guid Udi = Guid.Parse(place.udi.ToString().Split('/')[3].ToString());
         if (Udi == Guid.Parse(model.ParckingId))
         {
            place.bookedBy = null;
            place.bookedStart = null;
            place.bookedEnd = null;
            place.isBooked = false;
            place.isCarPlaced = false;
            column.parckingPlaces = JsonConvert.SerializeObject(parkingPlacesData);
            isUpdated = true;
            break;
         }
      }
      if (isUpdated)
         break;
   }
}
content.SetValue("parckingPlaces", JsonConvert.SerializeObject(mainData));
```

```
            _contentService.SaveAndPublish(content);
        }
        catch (Exception ex)
        {
            var tt = ex;
        }
        return Ok();
    }
    [HttpGet("statistic")]
    public async Task<IActionResult> GetStatistic()
    {
        var context = _umbracoContextAccessor.GetRequiredUmbracoContext();
        var page = context.Content.GetAtRoot().DescendantsOrSelf<BookingStatistic>().FirstOrDefault();

        const string dateFormat = "dd.MM.yyyy HH:mm:ss";


        var content = page.BookingStatisticList3.Select(x => {
            var parsed = x.Content as StatisticItem;
            return new StatisticItemDto
            {
                BookedBy = parsed.BookedBy.Name,
                BookingStart = parsed.BookingStart.ToString(dateFormat),
                BookingEnd = parsed.BookingEnd.ToString(dateFormat),
                BookingTime = parsed.BookingTime.ToString(dateFormat),
                IsCanceled = parsed.IsCancel,
                ParckingNumber = parsed.ParckingNumber
            };
        });
        return Ok(content);
    }
    [HttpPost("cancel-admin")]
    public async Task<IActionResult> CancelAdminOrder(CancelOrderModel model)

    {
        try
        {
            var content = _contentService.GetById(Guid.Parse(model.ParckingLot));
            if (content == null)
                return NotFound();
            var rawValue = content.GetValue("parckingPlaces");
            if (rawValue == null)
                return BadRequest("No parking places found");
            var mainData = JsonConvert.DeserializeObject<dynamic>(rawValue.ToString());
            bool isUpdated = false;
            foreach (var column in mainData.contentData)
            {
                if (column?.parckingPlaces != null)
```

```
            {
                var parkingPlacesJson = column.parckingPlaces.ToString();
                var parkingPlacesData = JsonConvert.DeserializeObject<dynamic>(parkingPlacesJson);
                foreach (var place in parkingPlacesData.contentData)
                {
                    Guid Udi = Guid.Parse(place.udi.ToString().Split('/')[3].ToString());
                    if (Udi == Guid.Parse(model.ParckingId))
                    {
                        place.bookedBy = null;
                        place.bookedStart = null;
                        place.bookedEnd = null;
                        place.isBooked = false;
                        place.isCarPlaced = false;
                        column.parckingPlaces = JsonConvert.SerializeObject(parkingPlacesData);
                        isUpdated = true;
                        break;
                    }
                }
                if (isUpdated)
                    break;
            }
        }
        content.SetValue("parckingPlaces", JsonConvert.SerializeObject(mainData));
        _contentService.SaveAndPublish(content);
    }
    catch (Exception ex)
    {
        var tt = ex;
    }
    return Ok();
  }
}
```

Index.js (головний фронтенд функціонал)

```
function headerContol() {
  return {
    logout() {
      axios.post('/umbraco/api/memberauth/logout')
        .then(response => {
          if (response.status === 200) {
            window.location.href = '/';
          } else {
            alert('Logout failed.');
          }
        })
        .catch(error => {
          console.error('Error:', error);
        });
    }
  }
}
function registerForm() {
  return {
    username: '',
    email: '',
    password: '',
    register() {
      const formData = {
        userName: this.username,
        password: this.password,
        email: this.email
      }

      axios.post('/umbraco/api/memberauth/register', formData)
        .then(response => {
          if (response.status === 200) {
            window.location.href = '/login';
          } else {
            alert('Registration failed.');
          }
        })
        .catch(error => {
          console.error('Error:', error);
        });
    }
  }
}


function loginForm() {
```

```
  return {
    username: '',
    password: '',
    login() {
      const formData = {
        userName: this.username,
        password: this.password
      }

      axios.post('/umbraco/api/memberauth/login', formData)
        .then(response => {
          if (response.status === 200) {
            window.location.href = '/';
          } else {
            alert('Login failed.');
          }
        })
        .catch(error => {
          alert('Login failed.');
          console.error('Error:', error);
        });
    }
  }
}

function parkingScheme(price, memberKey, parckingLot) {
  return {
    isVisible: false,
    place_number: "",
    into_date: "",
    out_date: "",
    sum: "",
    price: price,
    parkingId: "",
    memberKey: memberKey,
    parckingLot: parckingLot,

    setParckingPlace(parckingNumber, parkingId) {
      this.place_number = parckingNumber;
      this.parkingId = parkingId;
      this.isVisible = true;
    },
    cancele() {
      this.isVisible = false;
    },
    calculatePrice() {
      if (this.into_date && this.out_date) {
        const startDate = new Date(this.into_date);
```

```
      const endDate = new Date(this.out_date);

      if (startDate >= endDate) {
        alert('Дата в\'їзду повинна бути пізнішою за дату виїзду');
        return;
      }

      const diffInMs = endDate - startDate;
      const intervals = Math.floor(diffInMs / (10 * 60 * 1000));
      this.sum = intervals * price;
    }

  },
  orderPlace() {
    const formData = {
      placeNumber: this.place_number,
      startDate: this.into_date,
      endDate: this.out_date,
      parckingId: this.parkingId,
      parckingLot: this.parckingLot,
      memberKey: this.memberKey

    };
    if (!this.into_date) {
      alert("Заповніть дату в'їзду");
      return;
    }

    if (!this.out_date) {
      alert("Заповніть дату виїзду");
      return;
    }
    axios.post('/umbraco/api/Parcking/order', formData)
      .then(response => {
        if (response.status === 200) {
          window.location.href = window.location.href;
        } else {
          alert('order failed.');
        }
      })
      .catch(error => {
        alert('order failed.');
        console.error('Error:', error);
      });
  },
  cancelOrder(event, parckingId) {
    event.stopPropagation();
    const formData = {
```

```
            parckingId: parckingId,
            memberKey: this.memberKey,
            parckingLot: this.parckingLot
        };
        console.log(formData);
        console.log(this.parkingId);
        axios.post('/umbraco/api/Parcking/cancel', formData)
            .then(response => {
                if (response.status === 200) {
                    window.location.href = window.location.href;
                } else {
                    alert('order failed.');
                }
            })
            .catch(error => {
                alert('order failed.');
                console.error('Error:', error);
            });
    }
  }
}
let url = "/umbraco/api/Parcking/statistic";


function chartComponent() {
  return {
    statistics: [],
    selectedDataset: 'bookingTime',
    chart: null,

    async fetchStatistics() {
      try {
        const response = await fetch(url);
        this.statistics = await response.json();
        console.log('Fetched statistics:', this.statistics);
        this.initChart();
      } catch (error) {
        console.error("Помилка при завантаженні даних:", error);
      }
    },

    prepareData() {
      if (!this.statistics || this.statistics.length === 0) {
        console.error("Данные не загружены");
        return [];
      }

      switch (this.selectedDataset) {
```

```
case 'bookingTime': {
    const groupedData = this.statistics.reduce((acc, item) => {
      const date = item.bookingTime.split(' ')[0];
      acc[date] = (acc[date] || 0) + 1;
      return acc;
    }, {});

    return Object.entries(groupedData)
      .map(([date, count]) => ({
        label: date,
        value: count,
      }))
      .sort((a, b) => new Date(a.label.split('.').reverse().join('-')) - new Date(b.label.split('.').reverse().join('-')));
}

case 'bookingDuration':
    return this.statistics.map(item => ({
      label: `${item.parckingNumber}`,
      value: this.calculateDurationInHours(item.bookingStart, item.bookingEnd),
    }));

case 'canceledBookings': {
    const groupedCanceledData = this.statistics
      .filter(item => item.isCanceled)
      .reduce((acc, item) => {
        const date = item.bookingTime.split(' ')[0];
        acc[date] = (acc[date] || 0) + 1;
        return acc;
      }, {});

    return Object.entries(groupedCanceledData)
      .map(([date, count]) => ({
        label: date,
        value: count,
      }))
      .sort((a, b) => new Date(a.label.split('.').reverse().join('-')) - new Date(b.label.split('.').reverse().join('-')));
}

case 'totalBookings': {
    const groupedBookingData = this.statistics.reduce((acc, item) => {
      const date = item.bookingTime.split(' ')[0];
      acc[date] = (acc[date] || 0) + 1;
      return acc;
    }, {});

    return Object.entries(groupedBookingData)
      .map(([date, count]) => ({
        label: date,
```

```
                value: count,
            }))
            .sort((a, b) => new Date(a.label.split('.').reverse().join('-')) - new Date(b.label.split('.').reverse().join('-')));
        }

        default:
          return [];
      }
    },
    parseDate(dateString) {
      const [day, month, year, hour, minute, second] = dateString
        .split(/[\s.:]/)
        .map(Number);
      if (isNaN(day) || isNaN(month) || isNaN(year) || isNaN(hour) || isNaN(minute) || isNaN(second)) {
        return null;
      }
      return new Date(year, month - 1, day, hour, minute, second);
    },
    calculateBookingTimeInHours(bookingTime, bookingStart) {
      const bookingTimeDate = this.parseDate(bookingTime);
      const bookingStartDate = this.parseDate(bookingStart);
      if (!bookingTimeDate || !bookingStartDate) {
        return NaN;
      }
      const diffInMilliseconds = bookingStartDate - bookingTimeDate;
      return diffInMilliseconds / (1000 * 60 * 60);
    },
    calculateDurationInHours(bookingStart, bookingEnd) {
      const bookingStartDate = this.parseDate(bookingStart);
      const bookingEndDate = this.parseDate(bookingEnd);
      if (!bookingStartDate || !bookingEndDate) {
        return NaN;
      }
      const diffInMilliseconds = bookingEndDate - bookingStartDate;
      return diffInMilliseconds / (1000 * 60 * 60);
    },

    initChart() {
      this.updateChart();
    },

    updateChart() {
      if (this.chart) {
        this.chart.destroy();
        this.chart = null;
      }

      const canvasContainer = document.getElementById('chartContainer');
```

```
        canvasContainer.innerHTML = '';
        const newCanvas = document.createElement('canvas');
        newCanvas.id = 'chartCanvas';
        newCanvas.style.width = '100%';
        newCanvas.style.height = '500px';
        canvasContainer.appendChild(newCanvas);

        const preparedData = this.prepareData();
        const labels = preparedData.map(d => d.label);
        const data = preparedData.map(d => d.value);
        console.log("PPDATA");
        console.log(preparedData);

        const ctx = newCanvas.getContext('2d');
        this.chart = new Chart(ctx, {
          type: 'line',
          data: {
            labels: labels,
            datasets: [{
              label: 'Значения',
              data: data,
              backgroundColor: 'rgba(75, 192, 192, 0.2)',
              borderColor: 'rgba(75, 192, 192, 1)',
              borderWidth: 1,
            }],
          },
          options: {
            responsive: true,
            maintainAspectRatio: false,
            scales: {
              y: {
                beginAtZero: true,
              },
            },
          },
        });
      },

      init() {
        this.fetchStatistics();
      },
    };
}
```

Додаток Г – Ілюстративна частина

(обов'язковий)

ІЛЮСТРАТИВНА ЧАСТИНА

ІНТЕЛЕКТУАЛЬНА СИСТЕМА ОБЛІКУ ТРАНСПОРТНИХ ЗАСОБІВ НА ПАРКУВАЛЬНОМУ МАЙДАНЧИКУ

Студент групи ЗАКІТ-23м

Віктор КОНОТОПЕНКО

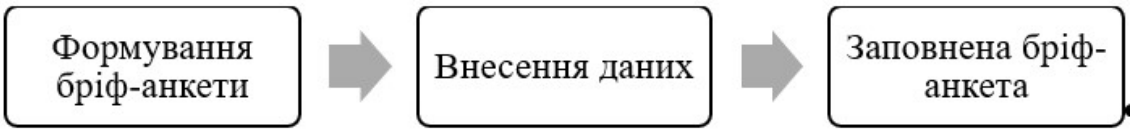
Керівник: к.т.н., доцент кафедри КСУ

Олег КОВАЛЮК

Вінниця 2024

Організаційна структура паркінгу

Реєстрація клієнта



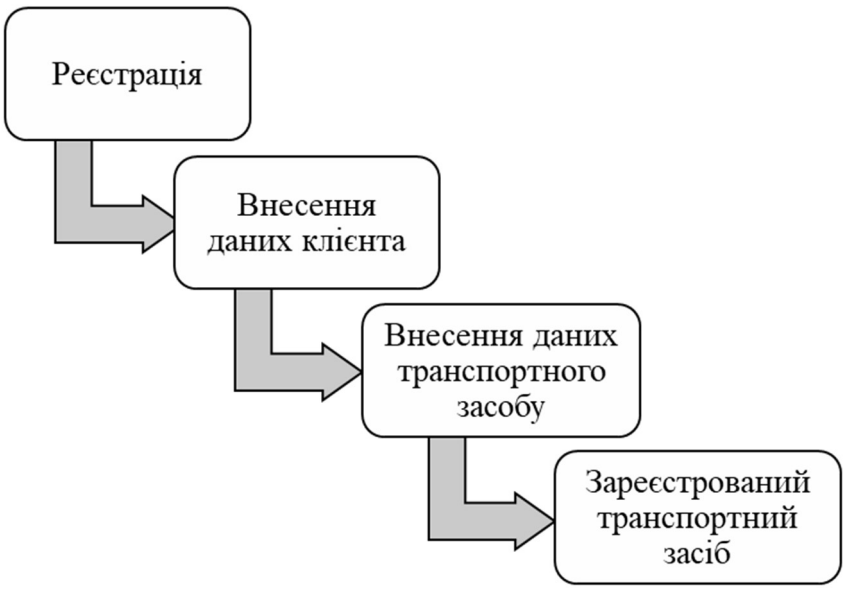
Формування бріф – анкети

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Реєстрація клієнта
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про клієнта
Вхідні документи	Немає
Вихідна подія	Відповідь про статус реєстрації
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

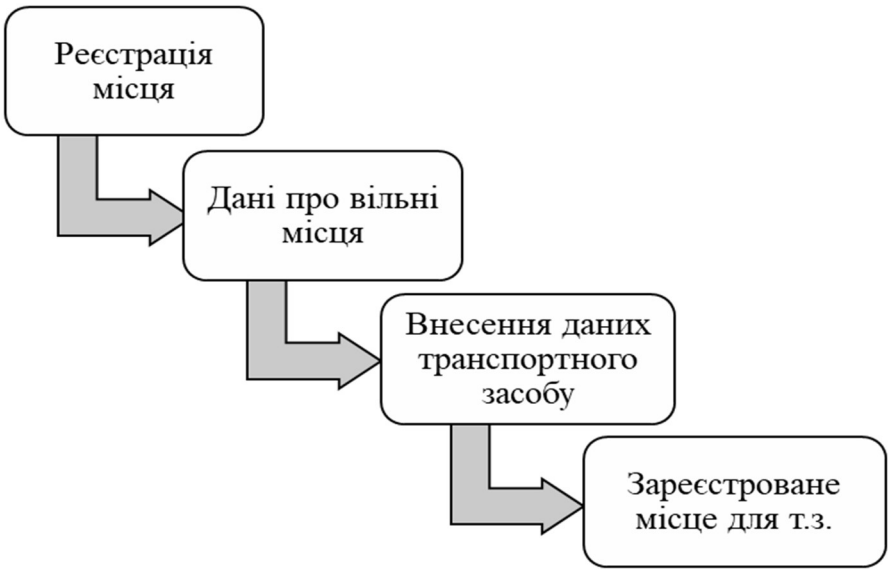
Реєстрація транспортного засобу

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Реєстрація транспортного засобу
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про транспортний засіб
Вхідні документи	Свідоцтво про реєстрацію авто
Вихідна подія	Відповідь про статус реєстрації, присвоєння реєстраційного коду транспортному засобу
Вихідні документи	Паркувальний талон, тимчасова чи постійна перепустка або безконтактні смарт–картки
Клієнт бізнес процесу	Сервер

Реєстрація транспортного засобу



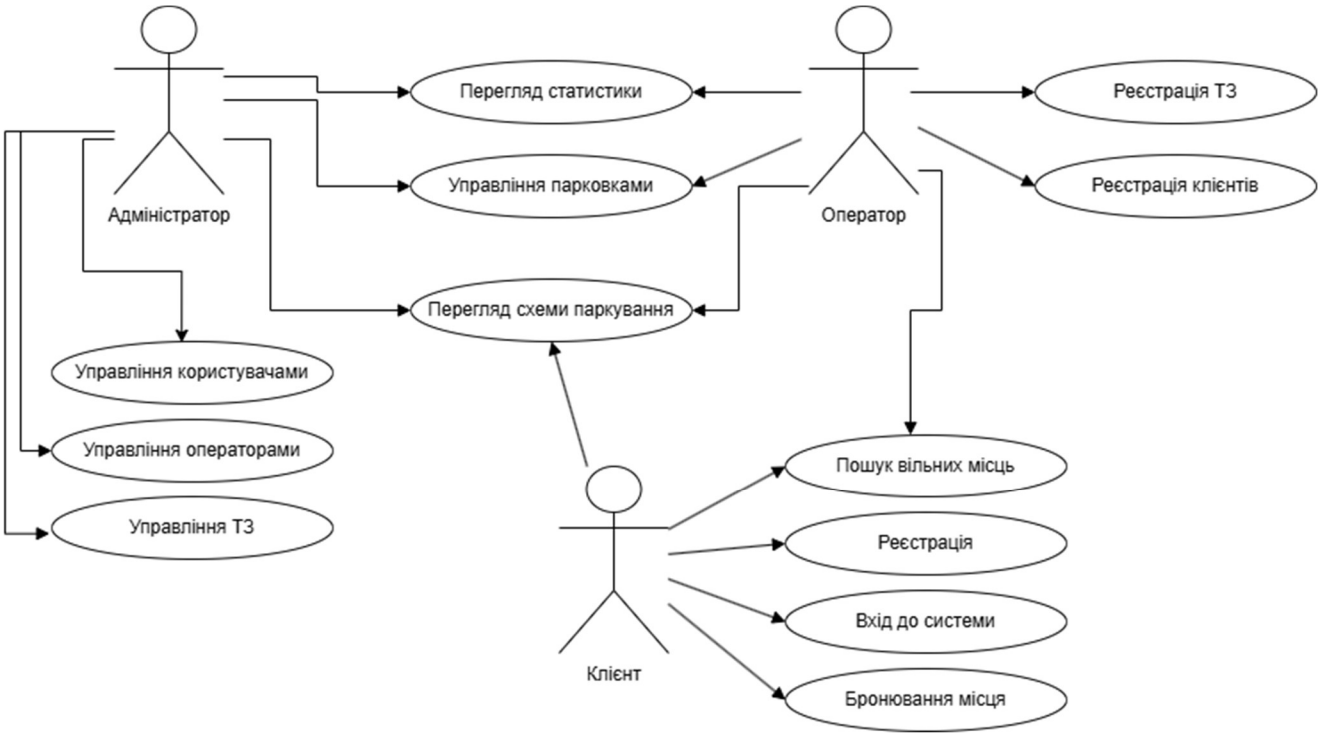
Бронювання місця для паркування



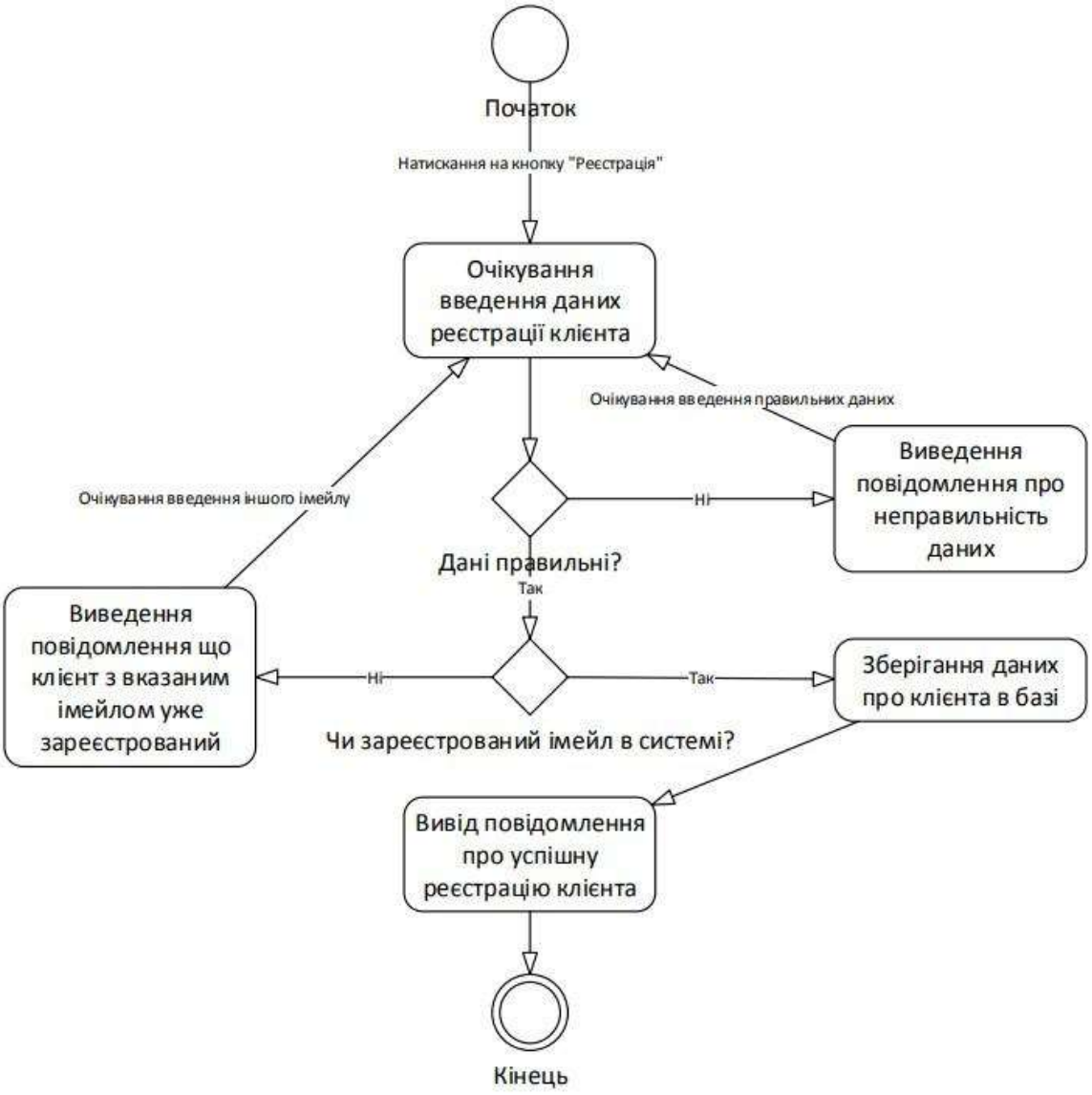
Опис Бронювання місця для паркування

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Бронювання місця на автостоянці
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про клієнта транспортний засіб
Вхідні документи	Немає
Вихідна подія	Відповідь про статус бронювання, присвоєння коду парко – місця транспортному засобу
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

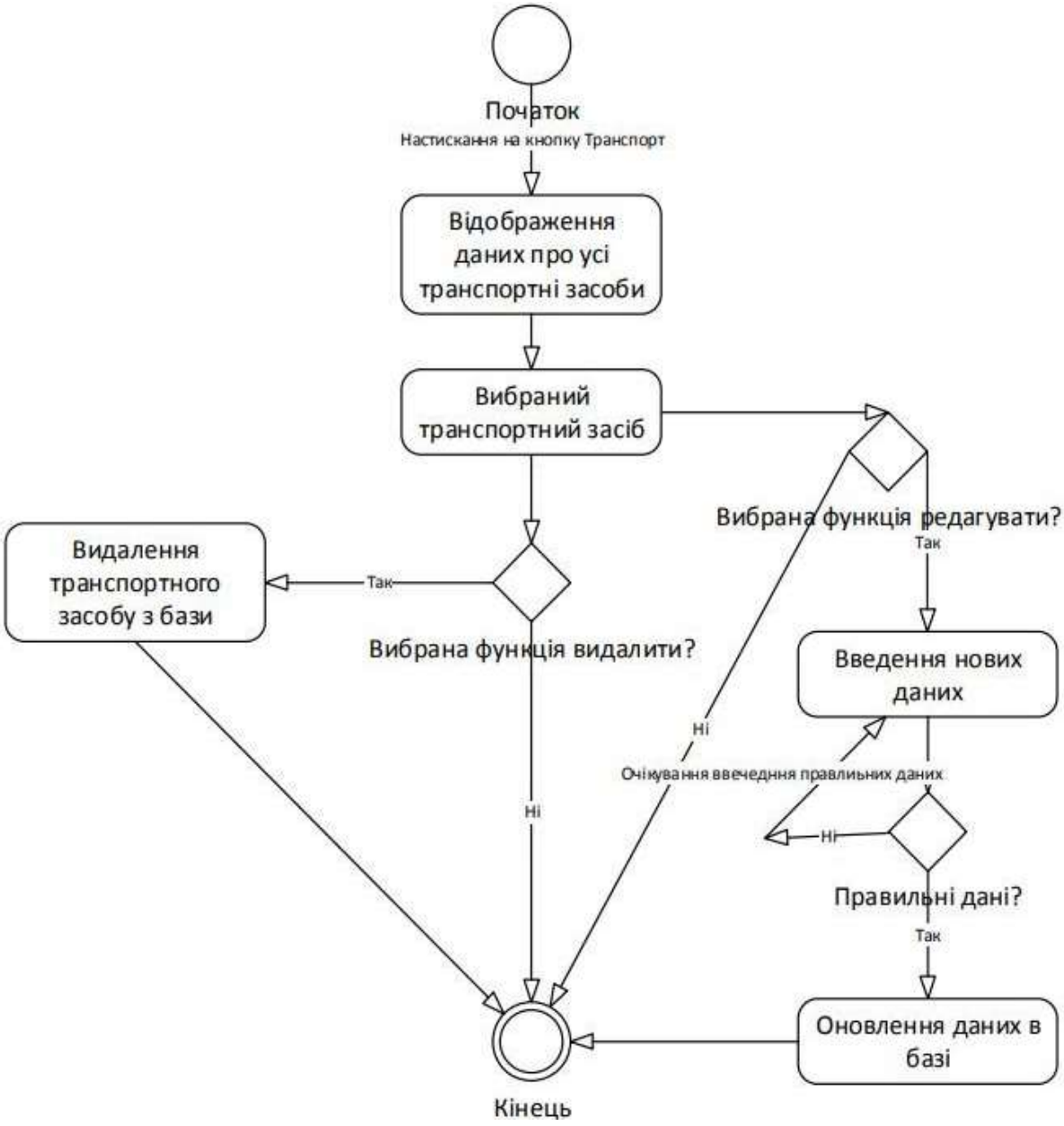
Діаграма варіантів використання



Діаграма діяльності «Реєстрація клієнта»



Діаграма діяльності «Транспортні засоби»



Основні показники сучасних програмних продуктів для паркування

	SimpleParcking	SERVIO PARCKING	AllStojanka	Парковка, стоянка в Борисполі	ParkingPay
Інтерфейс користувача	ПК	ПК	ПК	веб-сайт	Мобільний додаток
Облік	+	+	+	+	+
Відображення схеми автостоянки	+	+	+	-	+
Приймання та оформлення коштів	+	+	-	-	+
Відображення конкретного ТЗ	+	+	+	-	+
Перевірка клієнтів	+	-	+	-	+
Реєстрація клієнтів	+	+	+	+	+
Контроль прострочення	+	-	+	-	-

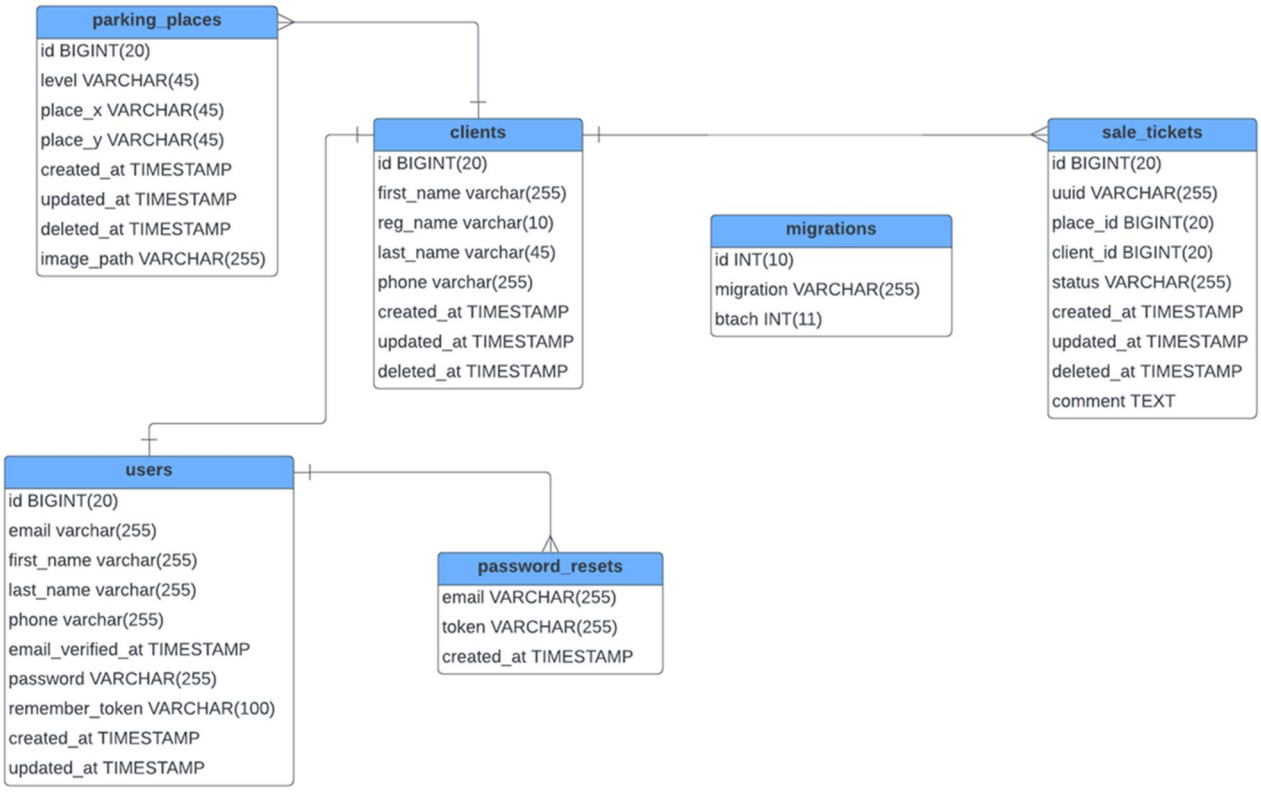
Порівняння мов програмування

Характеристика	**C#**	**PHP**	**Python**	**JavaScript**	**C++**
Призначення	Веб-додатки, десктопні системи, мобільні додатки	Веб-додатки	Наукові обчислення, веб-додатки	Інтерактивність на стороні клієнта, серверні додатки	Системне програмування, ігри, вбудовані системи
Продуктивність	Висока	Середня	Середня	Висока	Дуже висока
Масштабованість	Висока	Середня	Висока	Висока	Висока
Синтаксис	Об’єктно-орієнтований	Простий, процедурний	Простий, зрозумілий	Гнучкий, нестрогий	Складний, строгий
Інструменти розробки	Visual Studio, Rider	PhpStorm, VS Code	PyCharm, VS Code	VS Code, WebStorm	Visual Studio, CLion
Безпека	Висока	Середня	Висока	Середня	Висока
Кросплатформність	Висока	Висока	Висока	Висока	Висока
Сфера використання	Корпоративні додатки, веб-розробка, ігри	Веб-додатки	Моделювання, машинне навчання	Інтерактивні веб-сайти, сервери	Ігри, операційні системи

Порівняння фреймворку веб розробки

Характеристика	**Umbraco**	**Laravel**	**Django**	**Node.js**	**Qt**
Мова програмування	C#	PHP	Python	JavaScript	C++
Архітектура	MVC	MVC	MVT	Event-Driven	Модульна
Ліцензія	Відкрита	Відкрита	Відкрита	Відкрита	Пропрієтарна та відкрита
Кросплатформність	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux	Windows, MacOS, Linux
Масштабованість	Висока	Середня	Висока	Висока	Середня
Безпека	Вбудована (авторизація, локалізація)	Обмежена	Висока	Обмежена	Висока
Інтеграція з іншими системами	Висока (Azure, інші CMS)	Середня	Середня	Висока (npm, REST API)	Висока (вбудовані системи)
Простота у використанні	Висока	Середня	Низька	Середня	Низька
Активність спільноти	Середня	Висока	Висока	Висока	Середня

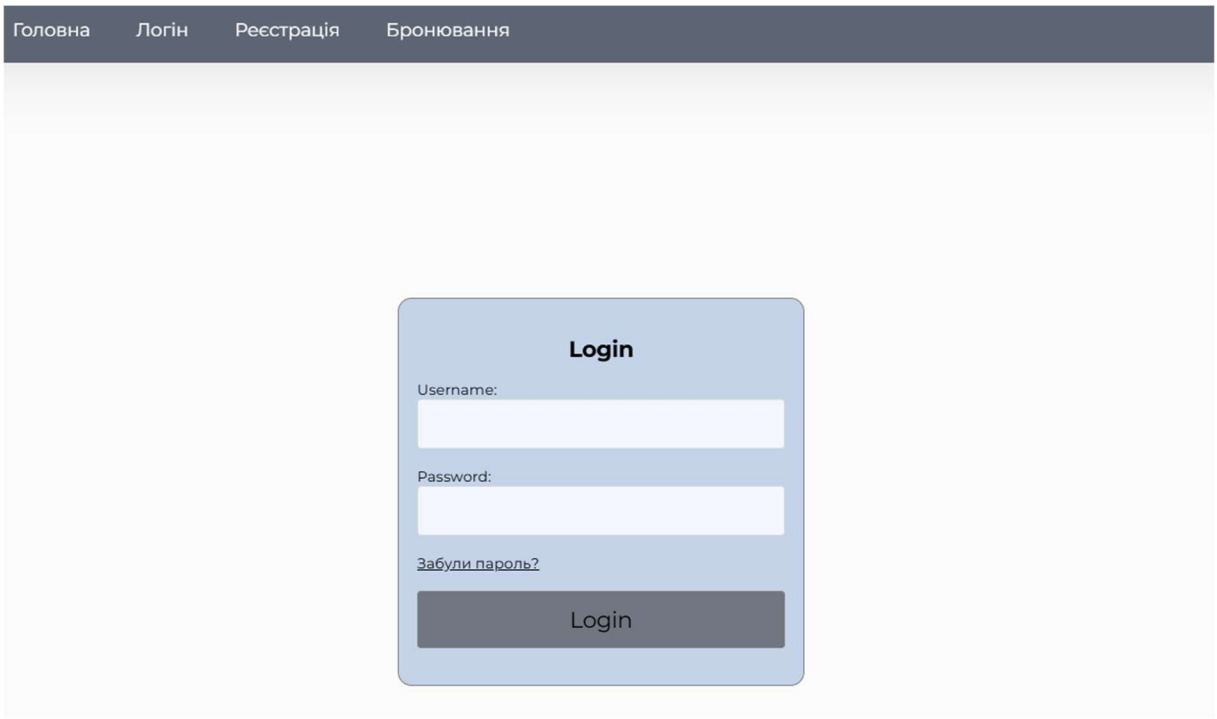
Схема бази даних



Діалогові вікна

Тип діалогового вікна	Приклад повідомлення	Дія користувача
Інформаційне	“Дані успішно збережено.”	Натиснути “ОК”
Попередження	“Перевірте коректність введених даних.”	Виправити помилку та спробувати ще раз
Помилка	“Сталася помилка. Повторіть спробу пізніше.”	Натиснути “ОК” або звернутися до адміністратора
Системне	“З’єднання з сервером втрачено. Спробуйте підключитися знову.”	Відновити підключення

Головний екран входу



Форма реєстрації нового користувача

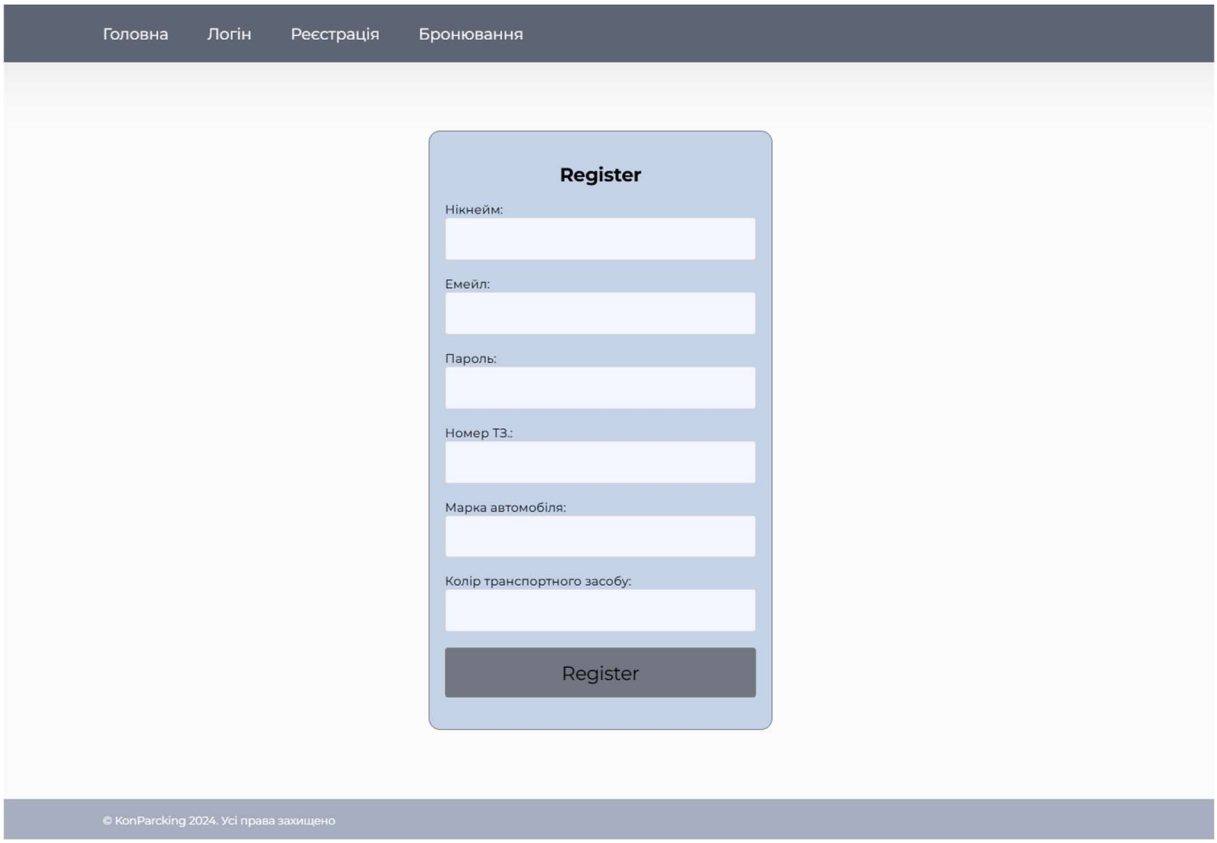
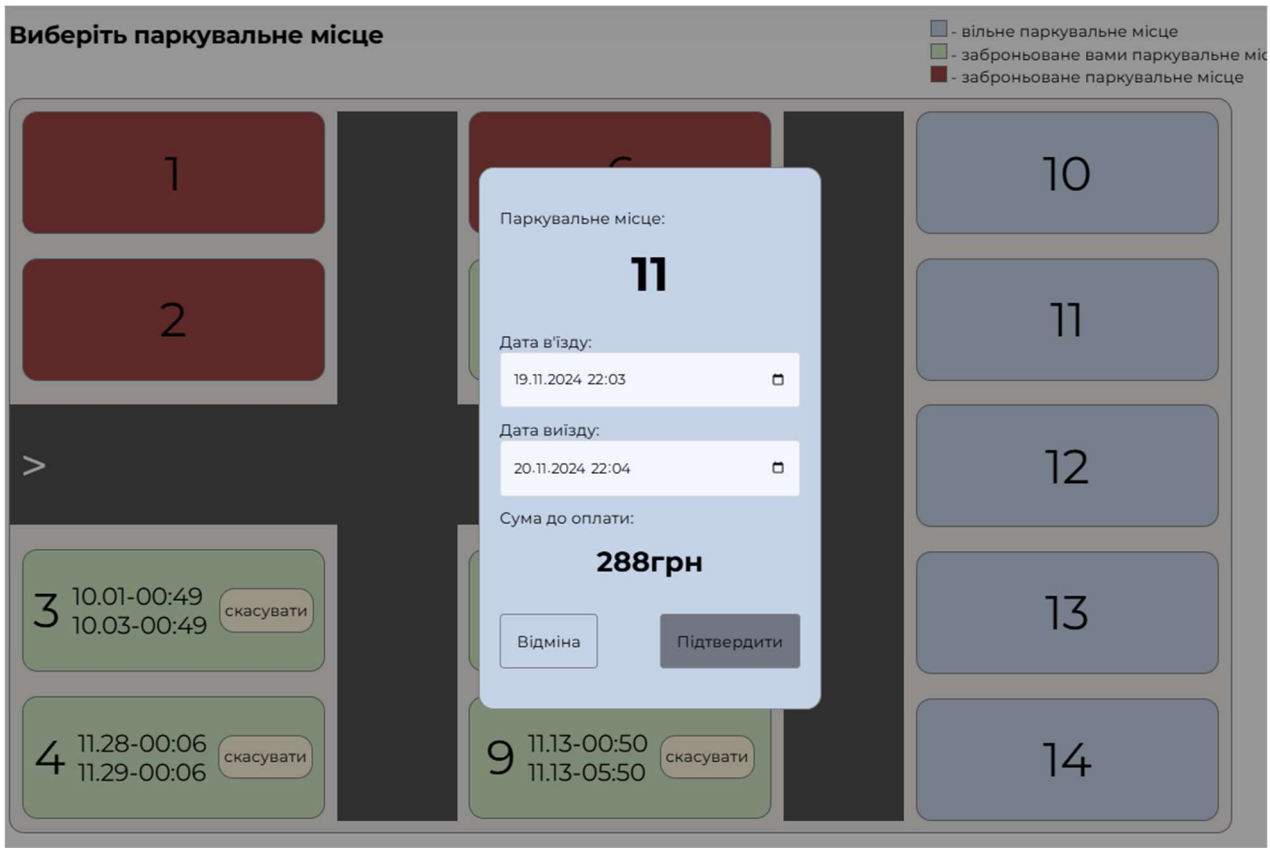
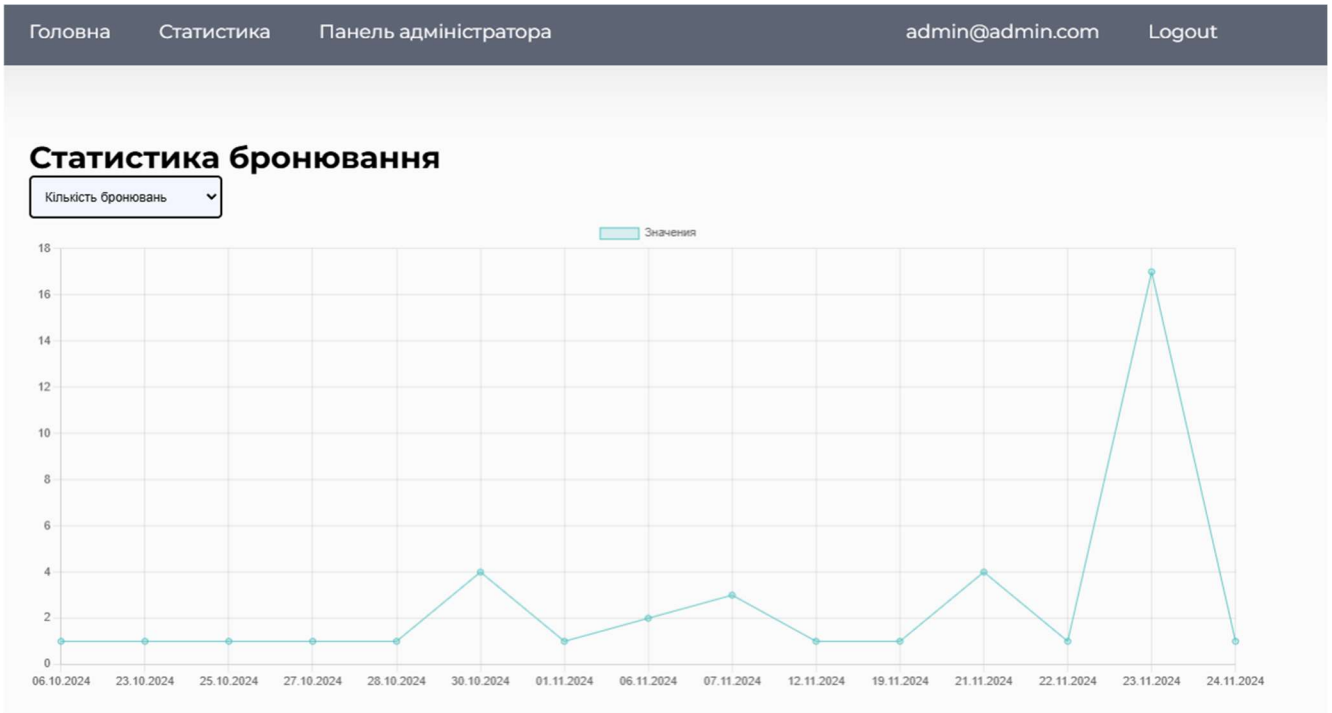


Схема парковки

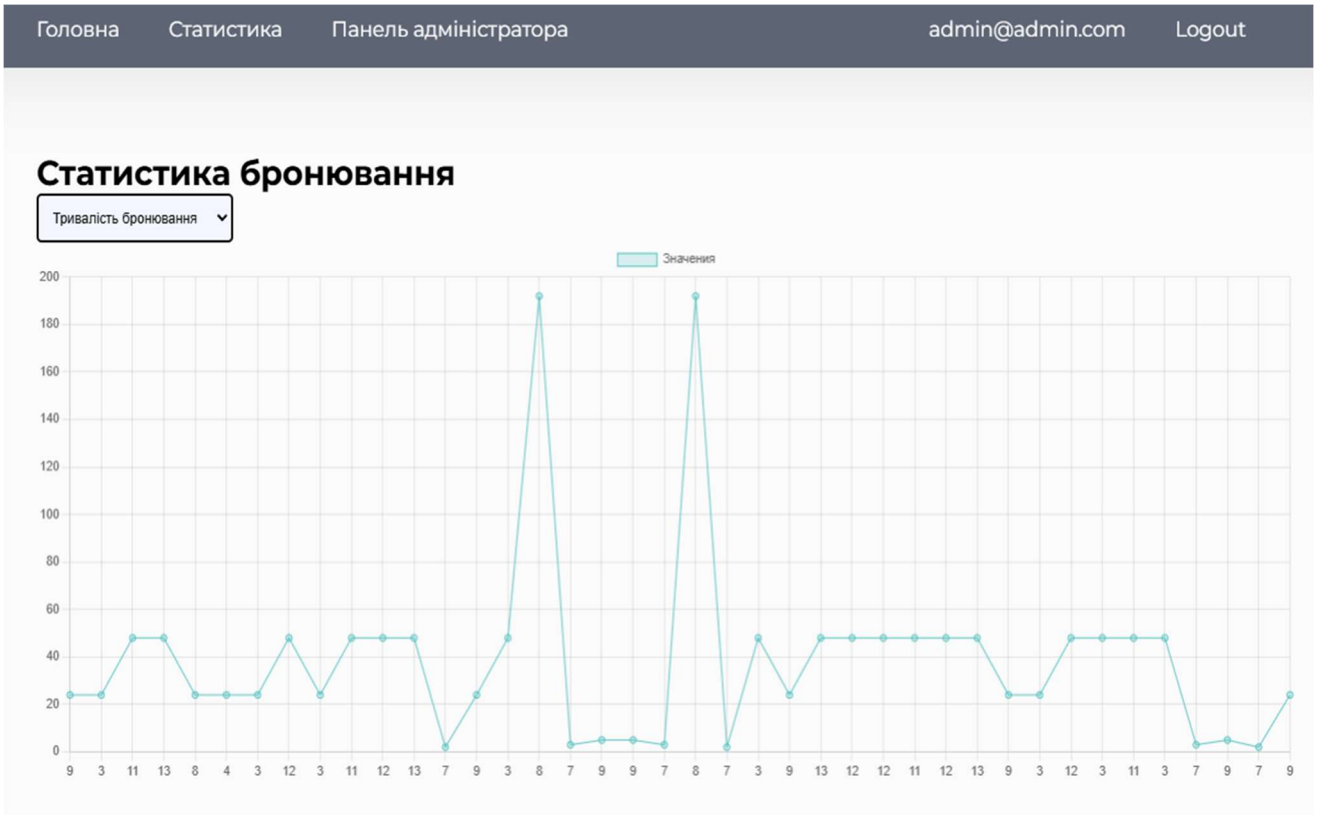
Форма бронювання паркувального місця



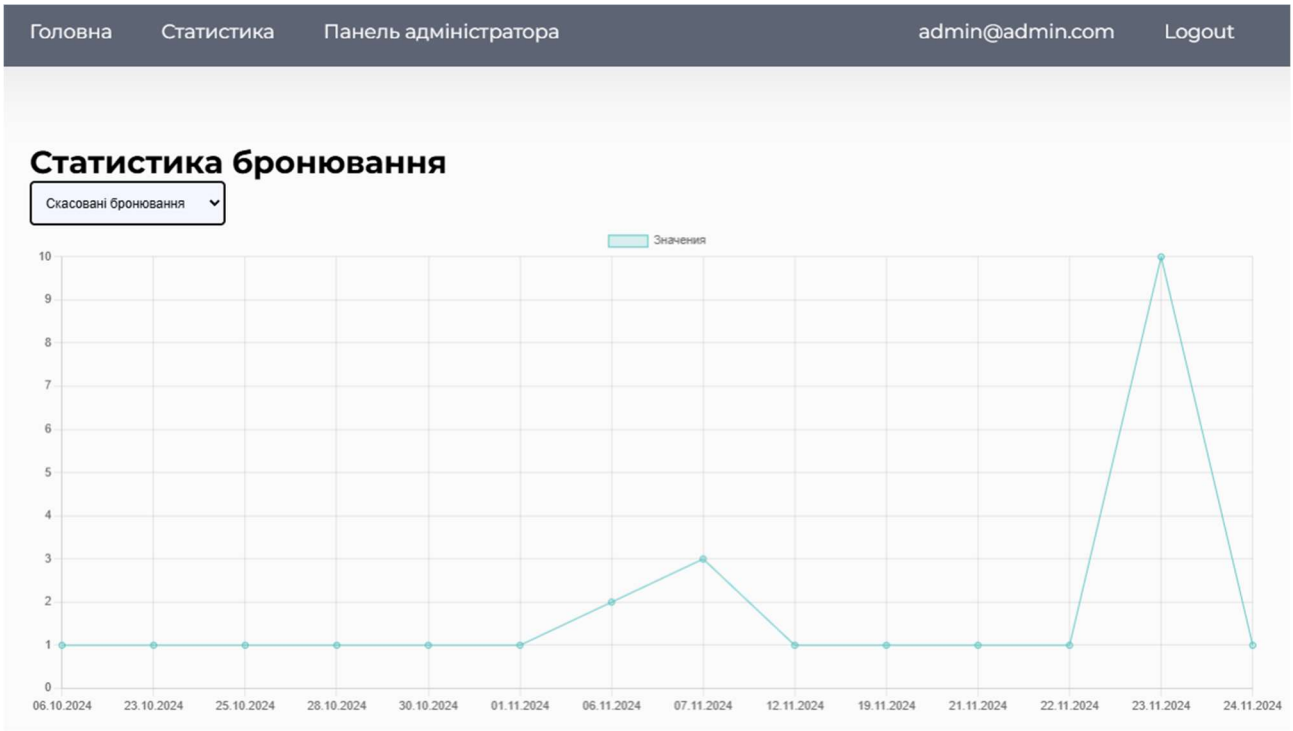
Графік час бронювання



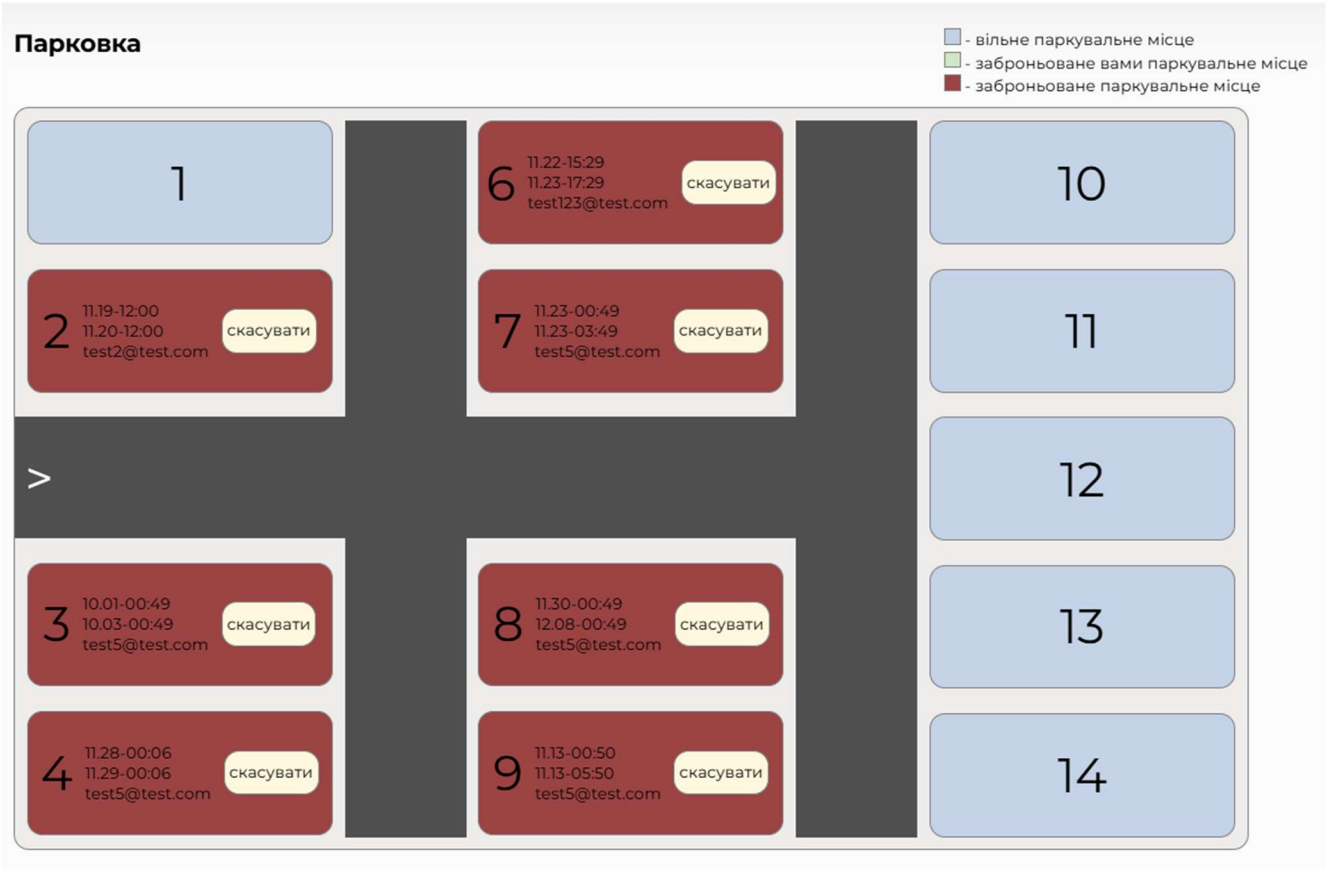
Графік тривалість бронювання



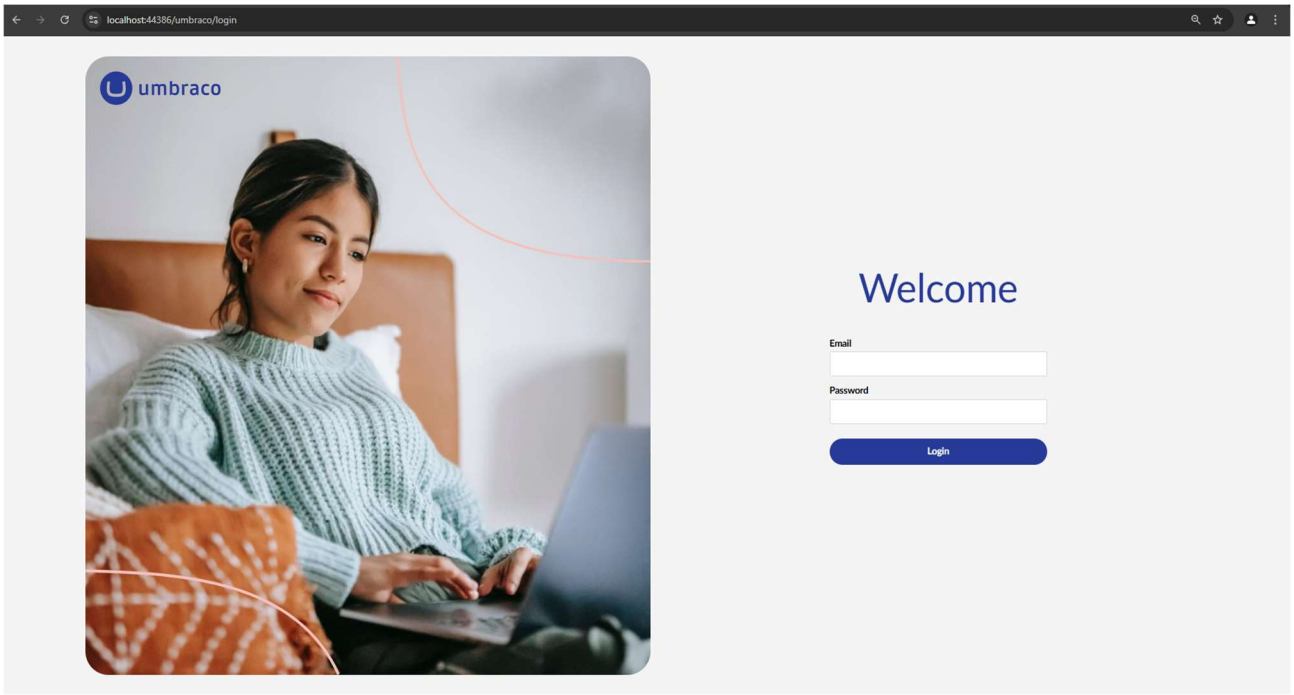
Графік скасовані бронювання



Панель адміністратора парковки



Авторизація в бекофіс системи



Бекофіс системи