Міністерство освіти і науки України

Луцький національний технічний університет
(повне найменування закладу вищої освіти)

Факультет комп'ютерних та інформаційних технологій
(повне найменування факультету)

Кафедра комп'ютерної інженерії та безпеки
(повне найменування кафедри)

КВАЛІФІКАЦІЙНА РОБОТА ЗА СТУПЕНЕМ ВИЩОЇ ОСВІТИ «МАГІСТР»

АНАЛІЗ ТА РОЗРОБКА СИСТЕМИ РОЗУМНОГО ПАРКІНГУ З ЧАСТКОВИМ ВИКОРИСТАННЯМ МОДЕЛЕЙ ШТУЧНОГО ІНТЕЛЕКТУ

ANALYSIS AND DEVELOPMENT OF A SMART PARKING SYSTEM WITH PARTIAL USE OF ARTIFICIAL INTELLIGENCE MODELS

спеціальність 123 Комп'ютерна інженерія
(шифр і назва спеціальності)

освітня програма Комп'ютерна інженерія
(назва освітньої програми)

Виконав: здобувач вищої освіти
групи КІм-21
Марчук Олександр Юрійович

(підпис)

Керівник:
к.т.н., доцент
Христинець Наталія Анатоліївна

(підпис)

Кваліфікаційну роботу
допущено до захисту
«_____» грудня 2025 р.
Гарант освітньої програми:
к.т.н., доцент
Гринюк Сергій Васильович

(підпис)

Луцьк – 2025 року

ЛУЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет *комп'ютерних та інформаційних технологій*
Кафедра *комп'ютерної інженерії та безпеки*
Ступінь вищої освіти: *магістр*
Галузь знань: *12 Інформаційні технології*
Спеціальність: *123 Комп'ютерна інженерія*
Освітня програма: *«Комп'ютерна інженерія»*

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ доц. Т.ТЕРЛЕЦЬКИЙ
«_____« _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ ЗДОБУВАЧУ ВИЩОЇ ОСВІТИ

Марчуку Олександру Юрійовичу
(прізвище, ім'я, по батькові)

1. Тема кваліфікаційної роботи *Аналіз та розробка системи розумного паркінгу з частковим використанням моделей штучного інтелекту*

Керівник роботи *к.т.н., доцент Христинець Наталія Анатоліївна*

затверджені наказом закладу вищої освіти від «17» червня 2025 року № 290/01-02
2. Строк подання здобувачем вищої освіти кваліфікаційної роботи 10.12.2025р.

3. Вихідні дані до роботи *Джерелом розробки є науково-технічна література та публікації в періодичних виданнях з даного питання, опубліковані зарубіжні та вітчизняні роботи в даній області, різні інтернет-ресурси технічного спрямування*

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити):
Вступ
Аналіз проблем та підходів до розробки системи інтелектуального паркінгу
Проєктування архітектури системи
Програмна реалізація та дослідження функціональних можливостей системи
Висновки

5. Перелік графічного (ілюстративного) матеріалу:

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис	
		завдання видав	завдання прийняв
Аналіз проблем та підходів до розробки інтелектуальних систем	*Христинець Н. А., доцент*		
Проєктування архітектури системи інтелектуального паркінгу	*Христинець Н. А., доцент*		
Розробка та тестування системи	*Христинець Н. А., доцент*		
Нормоконтроль	*Багнюк Н. В., доцент*		
Гарант ОП	*Гринюк С. В., доцент*		
Показник запозичень тексту	***___%***		
Академічна доброчесність	*Міскевич О. І., ст.викладач*		

7. Дата видачі завдання: 18.06.2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	*Огляд літератури із досліджуваної проблеми*	До 02.08.2025 р.	
2.	*Аналіз проблем та підходів до розробки систем паркінгу*	До 17.08.2025 р.	
3.	*Проєктування інтерфейсу системи*	До 11.09.2025 р.	
4.	*Розробка веб-сервісів та REST API*	До 04.10.2025 р.	
5.	*Висновки та пропозиції*	До 22.10.2025 р	
6.	*Формування списку використаних джерел*	До 28.10.2025 р.	
7.	*Формування додатків*	До 30.10.2025 р.	
8.	*Оформлення ілюстративного матеріалу*	До 11.11.2025 р.	
9.	*Представлення остаточного варіанту кваліфікаційної роботи керівникові*	До 16.11.2025 р.	
10.	*Нормоконтроль*	До 29.11.2025 р.	
11.	*Інструментальна перевірка на академічний плагіат*	До 02.12.2025 р.	
12.	*Здача кваліфікаційної роботи та всіх супровідних документів на кафедру*	До 09.12.2025 р.	

Здобувач вищої освіти _____ Марчук О. Ю.
(підпис) (прізвище, ініціали)

Керівник кваліфікаційної роботи _____ Христинець Н. А.
(підпис) (прізвище, ініціали)

АНОТАЦІЯ

Марчук О. Ю. Аналіз та розробка системи розумного паркінгу з частковим використанням моделей штучного інтелекту. Рукопис.

Кваліфікаційна робота магістра ОП «Комп'ютерна інженерія» спеціальності 123 Комп'ютерна інженерія. Луцький національний технічний університет. Луцьк, 2025.

Кваліфікаційна робота складається зі вступу, трьох розділів, висновків, списку використаних джерел, додатку.

У першому розділі роботи проведено огляд сучасних інтелектуальних підходів до розробки систем розумного паркінгу. Розглянуто аналітику моделей штучного інтелекту та особливості налаштування GPT-5 для прогнозування та оптимізації процесів паркування. Оцінено підходи до створення веб-сервісів та реалізації REST API для взаємодії з зовнішніми клієнтськими застосунками, а також протестовано основні механізми роботи системи: фіксацію заїзду та виїзду автомобілів, прогнозування середнього часу перебування транспортних засобів на паркінгу та аналіз загального простору паркінгу з реалізацією системи сповіщень.

Другий розділ присвячено методиці та інструментам розробки інтелектуальної системи паркінгу. Висвітлено основи побудови системи розумного паркінгу, вибір програмних інструментів та середовища реалізації.

У третьому розділі описано процес проектування веб-інтерфейсу з використанням засобів Filament. Подано налаштування серверу та деплой проекту на віддалений сервер, а також розробку інтелектуального інтерфейсу користувача для зручного керування паркінгом. Детально описано проектування REST API, що забезпечує безпечну та передбачувану взаємодію клієнтських застосунків із серверною частиною, фіксацію заїздів та виїздів автомобілів, обробку оплат та отримання аналітичних даних.

Ключові слова: розумний паркінг, штучний інтелект, GPT-5, REST API, веб-інтерфейс, сесія паркування, прогнозування завантаженості.

ANNOTATION

Marchuk O. Analysis and Development of a Smart Parking System with Partial Use of Artificial Intelligence Models. Manuscript.

Qualifying work of a Master's of EP «Computer Engineering» specialty 123 Computer Engineering. Lutsk National Technical University. Lutsk, 2025.

The thesis consists of an introduction, three chapters, conclusions, a list of references, and appendices.

The first section of the thesis provides an overview of modern intelligent approaches to the development of smart parking systems. It examines artificial intelligence model analytics and the specifics of configuring GPT-5 for forecasting and optimizing parking processes. Approaches to creating web services and implementing REST API for interaction with external client applications are evaluated, and the main mechanisms of the system are tested: recording the arrival and departure of cars, predicting the average time vehicles spend in the parking lot, and analyzing the total parking space with the implementation of a notification system.

The second section is devoted to the methodology and tools for developing an intelligent parking system. It covers the basics of building a smart parking system, the selection of software tools, and the implementation environment.

The third section describes the process of designing a web interface using Filament tools. It presents server configuration and project deployment to a remote server, as well as the development of an intelligent user interface for convenient parking management. It describes in detail the design of the REST API, which ensures secure and predictable interaction between client applications and the server side, recording of vehicle arrivals and departures, payment processing, and receipt of analytical data.

Keywords: smart parking, artificial intelligence, GPT-5, REST API, web interface, parking session, traffic forecasting.

ЗМІСТ

ВСТУП

В умовах стрімкого зростання урбанізованих територій та кількості транспортних засобів проблеми організації паркувальних місць набувають особливої актуальності. Традиційні методи управління паркінгами часто не забезпечують ефективного використання наявної інфраструктури, що призводить до перевантаження майданчиків, затримок у пошуку вільного місця та зниження рівня комфорту користувачів. У зв’язку з цим, розробка інтелектуальних систем розумного паркінгу стає важливою складовою сучасних смарт-міст, оскільки вони здатні автоматизувати процеси моніторингу, контролю та оптимізації завантаженості паркомайданчиків.

Особливу увагу в таких системах приділяють інтеграції алгоритмів штучного інтелекту для прогнозування заповненості, аналізу статистичних даних і забезпечення адаптивного керування інфраструктурою. Використання сучасних веб-технологій дозволяє реалізувати зручні адміністративні та клієнтські інтерфейси, що сприяють швидкій взаємодії користувачів із сервісом та підвищують ефективність роботи паркінгів.

Актуальність теми. Сучасний розвиток транспортної інфраструктури та зростання кількості автотранспортних засобів у містах зумовлюють потребу у створенні ефективних систем керування паркувальними просторами. Традиційні рішення, засновані на ручному контролі або простих датчиках, поступово втрачають ефективність через обмежену масштабованість, низьку швидкість обробки даних та відсутність можливостей прогнозування. У цьому контексті інтеграція технологій штучного інтелекту, машинного навчання, комп’ютерного зору та аналітики даних відкриває нові перспективи для побудови інтелектуальних паркувальних систем, здатних автоматично виявляти вільні місця, оптимізувати розподіл трафіку та підвищувати рівень комфорту користувачів. Розвиток таких систем сприяє зниженню заторів, економії часу водіїв, ефективнішому використанню міського простору та зменшенню екологічного навантаження, що визначає актуальність даного дослідження.

Метою дослідження є розробка архітектури інтелектуальної системи розумного паркінгу, що поєднує алгоритми штучного інтелекту, аналітику даних і сучасні веб-технології для забезпечення автоматизованого моніторингу, прогнозування завантаженості та зручної взаємодії користувача із системою.

Об’єктом дослідження є інтелектуальні паркувальні системи, що використовують технології штучного інтелекту, відеоаналітики, Інтернету речей та хмарних обчислень для оптимізації процесів пошуку, резервування та контролю паркомісць.

Предметом дослідження є методи, технології та архітектурні принципи побудови інтелектуальних паркінгових систем, зокрема підходи до аналізу відеопотоку, прогнозування завантаженості, організації взаємодії між клієнтськими та серверними компонентами, а також реалізації зручного веб-інтерфейсу управління.

Для досягнення мети роботи потрібно вирішити наступні задачі:

– проаналізувати сучасні підходи до побудови систем розумного паркінгу та виявити їхні ключові технологічні особливості, переваги й обмеження;

– розробити архітектуру системи розумного паркінгу, що включає серверну частину, адміністративну панель, клієнтські інтерфейси та механізми інтеграції зовнішніх застосунків через REST API;

– забезпечити автоматизацію налаштування серверного середовища та деплой веб-застосунку з використанням сучасних інструментів адміністрування та пакетних менеджерів;

– реалізувати прототип системи з можливістю обробки відеопотоку, аналізу статистичних даних і візуалізації результатів роботи;

– оцінити ефективність та точність роботи моделі;

– провести тестування системи, оцінити коректність функціонування всіх модулів.

Апробація результатів. Результати роботи опубліковані у фаховому збірнику «Комп’ютерно-інтегровані технології: освіта, наука, виробництво» [1] та представлені в додатку А кваліфікаційної роботи.

РОЗДІЛ 1
ОГЛЯД ТА АНАЛІЗ ІНТЕЛЕКТУАЛЬНИХ ПІДХОДІВ ДО РОЗРОБКИ СИСТЕМИ РОЗУМНОГО ПАРКІНГУ

1.1 Аналітика моделей штучного інтелекту та налаштування GPT-5

Сучасні системи розумного паркінгу активно використовують моделі штучного інтелекту для оптимізації розподілу місць, прогнозування завантаженості паркінгу та покращення взаємодії з користувачами. Основу таких систем становлять алгоритми машинного навчання, нейронні мережі, а також інтелектуальні моделі, здатні аналізувати потоки даних у реальному часі.

Огляд літератури [2-3] показує, що сучасні публікації виділяють два основних класи підходів до реалізації інтелектуальних підсистем паркінгу. Ряд досліджень базується на розподіленому обробленні відеопотоків і застосуванні спеціалізованих нейронних мереж для детекції та класифікації стану місць, що дозволяє досягати високих значень точності при відносно невеликому відгуку завдяки локальному інференсу. Цей напрям підтримують оглядові статті та прикладні дослідження [4-5], які демонструють практичні реалізації із застосуванням архітектур на основі SSD і YOLO та з використанням датасетів CNRPark і PKLot для валідації якості класифікації. Джерела вказують на існування стійких викликів, пов’язаних з освітленням, частковою оклюзією та погодними умовами, що вимагає адаптивних стратегій попередньої обробки та аугментації даних для отримання узагальнювальної моделі.

Ряд наукових публікацій орієнтовані на прогнозні модулі та системи підтримки прийняття рішень [6-7], які використовують часові ряди подій паркування, дані з платіжної інфраструктури та зовнішні показники трафіку для прогнозування завантаженості і середнього часу перебування транспортних засобів. Для таких задач застосовуються регресійні моделі, рекурентні мережі та сучасні трансформерні архітектури, що дозволяє інтегрувати багатовимірні фактори впливу. Порівняльні дослідження у сфері прогнозування показують

перевагу комбінованих гібридних підходів, які поєднують класичну статистику з глибинним навчанням при наявності достатніх історичних даних.

Особливу увагу у дослідженнях приділено застосуванню моделі GPT-5 як гібридного інструменту для підтримки аналітичних, комунікаційних і тестових процесів у середовищі паркінгу. Відомо, що GPT-5 відрізняється покращеним контекстним аналізом, здатністю підтримувати тривалу пам’ять сеансів і можливістю формувати узагальнені висновки на основі великих обсягів даних [8]. Її використання у сфері транспортної аналітики дає змогу створювати системи, що самонавчаються, реагують на зміни навколишнього середовища та підтримують персоналізовану взаємодію з користувачами. З цих досліджень можна зробити висновки, що GPT-5 виявила високу ефективність у виконанні завдань діалогової взаємодії, прогнозування запитів та аналітики у режимі реального часу. У таблиці 1.1 подано оцінку рівня залученості моделі у різних підсистемах системи розумного паркінгу.

Таблиця 1.1 – Класифікація підсистем за ступенем інтеграції GPT-5 та їх функціональними ролями

Підсистема	Рівень залученості GPT-5	Основна функція
Модуль бронювання місць	Високий	Обробка запитів користувачів, контекстне уточнення
Аналітичний модуль	Середній	Формування звітів, рекомендацій та прогнозів
Сенсорний модуль	Низький	Обробка неструктурованих даних від датчиків
REST API інтерфейс	Високий	Інтерпретація запитів зовнішніх систем

Оцінка ролі GPT-5 у задачах інтелектуальних систем демонструє і різноманітні можливості [9-10], і обмеження цієї генеративної моделі.

GPT-5 ефективна як модуль для інтерпретації природномовних запитів користувачів [11], автоматичного формування тестових сценаріїв [12], генерації пояснювальних звітів і допоміжної аналітики при побудові пайплайнів обробки даних.

Водночас, GPT-5 має обмеження у вигляді потреби у контролі фактичності і ризику генерації неточних тверджень якщо модель застосовується без додаткових механізмів верифікації, що робить необхідним поєднання GPT-5 з жорсткими валідаційними модулями і зовнішніми джерелами правди для критичних операцій.

Порівняльний аналіз підходів до інтеграції GPT-5 у систему розумного паркінгу дозволяє виокремити три характерні архітектурні моделі.

Перша модель [13-14] передбачає розгортання GPT-5 на серверному рівні як інтелектуального сервісу для формування текстових відповідей, звітів і рекомендацій, що використовуються в користувацькому інтерфейсі та адміністративних модулях.

Друга модель базується на поєднанні GPT-5 із символьними контролерами, де генеративна модель формує пропозиції дій, а спеціалізовані алгоритми та правила забезпечують їхню безпечну реалізацію.

Третя модель розглядає GPT-5 як автоматизований інструмент тестування API та сценаріїв інтеграції [15], що дає змогу істотно скоротити час виконання регресійних випробувань.

Результати спостережень підтверджують доцільність вибору конкретної архітектури залежно від рівня критичності задачі та наявності механізмів зовнішньої перевірки коректності роботи системи.

Таблиця 1.2 демонструє узагальнені результати аналізу продуктивності розглянутих моделей, оцінених за середнім часом відповіді, точністю прогнозування та витратами ресурсів.

Таблиця 1.2 – Порівняльний аналіз продуктивності моделей інтелектуальних систем паркінгу

Модель, архітектура	Середній час відповіді, мс	Точність прогнозу, %	Витрати ресурсів CPU, %
GPT-5 і REST API	210	96,3	48
GPT-4 Turbo і FastAPI	315	91,7	55
LLaMA-3	280	89,1	61
CNN-LSTM	190	93,8	42

Огляд способів тестування інтелектуальних систем паркування підтверджує наявність фундаментальних праць та прикладних стандартів, що мають вагу у світовій практиці [16].

Серед них слід виокремити роботу, що пропонує метод побудови тестових сценаріїв для різних систем сприйняття і побудови наборів сценаріїв для комплексного тестування, який опубліковано на платформі SAE і широко цитується в інженерній спільноті за методичну строгость формування сценаріїв і вимог до верифікації систем.

Іншими цікавими і сучасними напрямками робіт з тематики інтелектуальних систем паркінгу є ті, які досліджують розподілені рішення і архітектури Edge Computing для паркувальних систем.

У цих дослідженнях модулі CNRPark і PKLot використовуються для валідації алгоритмів класифікації стану паркомісць і містять приклади складних умов зображення, які часто наводяться у супровідній ілюстративній частині досліджень.

Авторами [17] представлено архітектуру взаємодії (рисунок 1.1) між сенсорними потоками, модулями прогнозування та генеративним аналізом.

Рисунок 1.1 – Архітектура взаємодії компонентів системи розумного паркінгу за автором H. Zhang [17]

На поданому рисунку відображено багаторівневу структуру взаємодії сенсорних потоків, аналітичних модулів і генеративного ядра на базі моделей штучного інтелекту.

Архітектура демонструє, яким чином дані від IoT-сенсорів трансформуються у прогнозні та аналітичні рішення за допомогою інтеграції з генеративною моделлю GPT-типу.

1.2 Розробка веб-сервісів і створення REST API для зовнішнього використання

У сучасних системах розумного паркінгу веб-сервіси виступають основною ланкою інтеграції між апаратними сенсорними модулями, аналітичними блоками та зовнішніми клієнтськими додатками [18]. Аналітичний

огляд підходів до створення REST API демонструє тенденцію переходу від класичних монолітних серверів до мікросервісних архітектур, у яких кожен сервіс відповідає за окремий аспект системи – реєстрацію паркувального місця, обробку платежів, прогнозування часу звільнення, або управління повідомленнями користувачів. Такий підхід суттєво підвищує масштабованість системи, спрощує тестування та оновлення окремих компонентів.

На рисунку 1.2 зображено комплексну інфраструктуру розумного паркування, в основі якої використано локальну хмару Arrowhead для інтеграції сенсорних вузлів, шлюзів та сервісів управління.

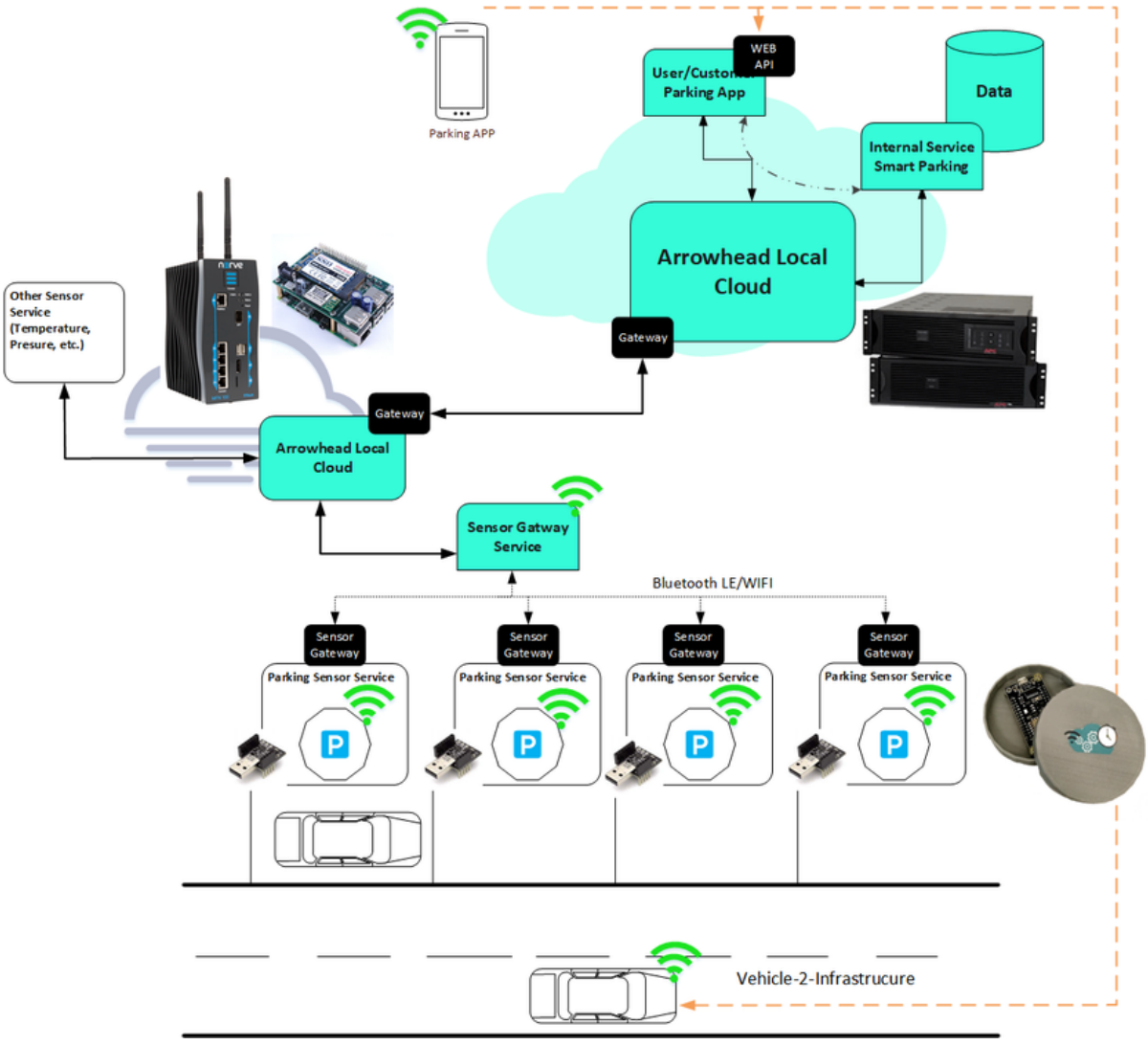

Рисунок 1.2 – Функціональна архітектура IoT-системи розумного паркування на основі локальної хмарної платформи Arrowhead [18]

Паркувальні датчики, розміщені на місцях стоянки, зчитують інформацію про зайнятість та передають її через шлюзи за допомогою Bluetooth LE або Wi-Fi у сервісний шар. Далі дані потрапляють до локальної хмари Arrowhead, де взаємодіють з внутрішніми сервісами, зокрема сервісом Smart Parking, що керує агрегуванням, зберіганням та обробкою подій. Користувачі отримують доступ до інформації про вільні місця через мобільний застосунок або веб-API.

Передбачено можливість включення додаткових сенсорних служб, а також інтеграцію з інфраструктурою Vehicle-to-Infrastructure для взаємодії транспорту з паркувальною системою в реальному часі. Така архітектура забезпечує масштабованість, безпеку передачі даних та сервіс-орієнтовану взаємодію між усіма компонентами системи.

Створення інтерфейсу REST API є базовим підходом до забезпечення взаємодії між компонентами системи в умовах розподіленої архітектури [19].

REST API забезпечує уніфікований спосіб обміну даними через протокол HTTP/HTTPS, що дозволяє клієнтським застосункам, зокрема мобільним, веб-панелям і стороннім аналітичним платформам, ефективно звертатися до функціональності системи паркінгу (рисунок 1.3).

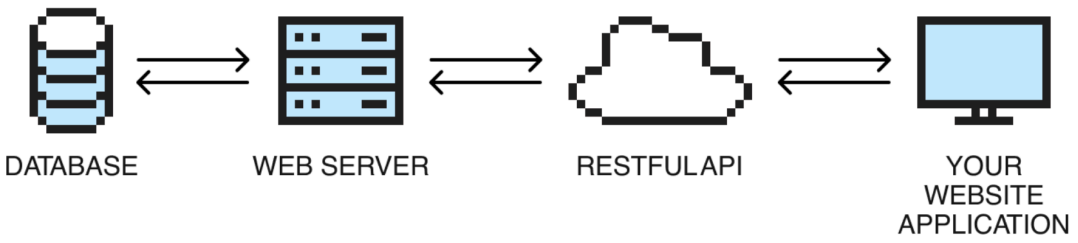

Рисунок 1.3 – Схема взаємодії компонентів через REST API [19]

На рисунку показано логічний ланцюг взаємодії між базою даних, веб-сервером, REST API та клієнтським застосунком. REST API виступає посередником, через який зовнішні застосунки або сервіси отримують доступ до даних і функцій серверної частини. Така модель є доречною для систем розумного паркування, оскільки потік інформації про зайнятість місць, запити

мобільних застосунків та інтеграція зі сторонніми сервісами відбувається саме через стандартизовані HTTP-запити. REST-архітектура дозволяє абстрагувати фізичний рівень сенсорів від клієнтських інтерфейсів, забезпечує платформонезалежність, масштабованість і можливість безпечного надання доступу до даних паркування іншим інформаційним системам, зокрема для аналітики, навігаційних сервісів або міської «розумної» інфраструктури.

У контексті інтеграції моделей штучного інтелекту, зокрема GPT-5, REST API виконує роль проміжного рівня, який дозволяє моделі взаємодіяти з даними з сенсорів, базами даних та інтерфейсами користувачів. Наприклад, запити типу GET або POST можуть використовуватися для надсилання вхідних параметрів до модуля GPT-5, де відбувається обробка природної мови, формування рекомендацій або генерація текстових описів для аналітичних звітів. У свою чергу, відповіді моделі надсилаються у структурованому форматі JSON, що забезпечує легкість інтеграції з іншими сервісами.

Дослідження останніх років [20-21] свідчать, що ефективність REST-архітектури у напрямку «smart parking» значною мірою залежить від швидкодії обробки запитів та надійності передачі даних. Для цього активно застосовуються такі технології, як FastAPI, Node.js та інші, які забезпечують асинхронну обробку запитів, автоматичну валідацію даних і підтримку токенів безпеки.

Важливою особливістю є також впровадження WebSocket-підключень, що дозволяє здійснювати миттєве інформування клієнтів про зміну стану паркомісць у реальному часі. Наприклад, коли автомобіль залишає місце, REST API через подієвий канал передає оновлення всім активним клієнтам без необхідності перезапитування даних.

Крім технічної оптимізації, увагу дослідників [22-23] зацікавили питання безпеки REST-інтерфейсів. Ці наукові праці підкреслюють важливість дотримання принципів Least Privilege та Secure Input Validation, а також регулярного тестування API за допомогою інструментів на кшталт Postman, Swagger.

Авторами доведено, що у системі розумного паркінгу REST API виступає не тільки каналом комунікації, а й інтелектуальним посередником. Він здатен контролювати послідовність подій, синхронізувати потоки даних і підтримувати когнітивну взаємодію між користувачем та системою. Інтеграція GPT-5 через REST-інтерфейс надає додаткові переваги – автоматичне формування описів помилок, контекстну інтерпретацію запитів і гнучку адаптацію до мовних відмінностей користувачів. Тобто, розробка REST API у проєкті системи розумного паркінгу є не лише інженерним завданням, а й концептуальною складовою створення відкритої, розширюваної та когнітивно орієнтованої архітектури, де кожен запит і відповідь формують частину інтелектуального циклу обробки інформації.

1.3 Тестування основних механізмів і методів взаємодії

1.3.1 Паркування та виїзд автомобіля

На першому етапі тестування перевіряються базові сценарії в’їзду і виїзду автомобілів із використанням ідентифікаторів – ID-міток або реєстраційних номерів. Система автоматично фіксує зайнятість місця, оновлювала статус у базі даних і відправляла сповіщення через API. Подібні дослідження, наведені у роботах [24-25], демонструють, що коректність реакції системи на такі елементарні події є критичним передумовним кроком для перевірки складніших сценаріїв бронювання, комбіновані події, інтеграція з навколиш. інфраструктурою, оскільки саме вони визначають синхронність між сенсорним шаром, сервісною логікою та зовнішніми інтерфейсами доступу до даних.

Під час тестування сценарію виїзду автомобіля оцінюється коректність зворотної події: чи фіксується звільнення місця, чи не виникають помилкові тригери, чи відбувається своєчасне оновлення статусу в базі даних і віддзеркалення цього оновлення у фронтенд-інтерфейсі без додаткових запитів від користувача. Аналізуються також механізми узгодженості транзакцій та аварійні стани (наприклад, некоректне завершення сесії, розрив передачі даних,

розсинхронізація часових міток між підсистемами). Результати тестування цього базового сценарію дозволяють оцінити надійність ланцюгів обробки реальних подій, виявити критичні точки у взаємодії модулів та підтвердити працездатність методів, покладених в основу сервісної логіки управління паркувальними ресурсами.

1.3.2 Прогнозування середнього часу розміщення автомобіля

Задача прогнозування середнього часу перебування автомобіля на паркувальному місці є ключовою для планування завантаженості, тарифікації та оптимізації політики бронювання. Базовою емпіричною величиною є середній час стоянки [26], що обчислюється за історичними даними.

Якщо для n завершених сеансів паркування відомі інтервали $\{T_1, T_2, \dots, T_n\}$, то їх середнє значення $\bar{T}$ задається як формула (1.1):

$$\bar{T} = \frac{1}{n}\sum_{i=1}^{n} T_i \tag{1.1}$$

Для оцінки стабільності процесу розглядається дисперсія та стандартне відхилення, що характеризують розкид значень відносно середнього. Дисперсія σ^2 при вибірковому наближенні обчислюється за формулою (1.2).

$$\sigma^2 = \frac{1}{n-1}\sum_{i=1}^{n}(T_i - \bar{T})^2, \tag{1.2}$$

а стандартне відхилення $\sigma = \sqrt{\sigma^2}$. Висока варіативність часу стоянки означає низьку передбачуваність і вимагає застосування імовірнісних або регресійних моделей, замість статичних середніх значень.

Для прогнозу майбутніх значень середнього часу використовується авторегресійна модель з ковзним середнім ARIMA(p,d,q), яка апроксимує процес на основі історичних спостережень з урахуванням тренду та шуму. У спрощеному вигляді ця підкомпонента моделі задається формулою (1.3).

$$T_t = c + \sum_{k=1}^{p} \varphi_k T_{t-k} + \varepsilon_t, \quad (1.3)$$

де φ_k – авторегресійні коефіцієнти;

c – константа;

ε_t – стохастичний шум.

Такий підхід дозволяє оцінювати очікуваний час звільнення місця у реальному часі на основі вже зафіксованої тривалості поточної сесії.

Альтернативно середній час може оцінюватися умовно – з урахуванням додаткових факторів як день тижня, година доби, тип зони, наявність подій, через умовне математичне сподівання $E[T \mid X]$, що зводить задачу до регресії з пояснювальними змінними [27], що обчислюється формулою (1.4).

$$E[T \mid X] = f(X_1, X_2, \dots, X_m), \quad (1.4)$$

де X – це контекстні атрибути середовища.

Отримані моделі дозволяють не лише оцінювати очікуваний час перебування автомобіля, а й приймати управлінські рішення – коригувати політику бронювання, формувати динамічні тарифи, прогнозувати дефіцит місць у пікові інтервали та оцінювати ефективність інфраструктурних змін.

Ці обчислення дозволяють не лише кількісно описати фактичну поведінку користувачів паркувальної інфраструктури, а й переходити до керованого прогнозування завантаженості в майбутні періоди. Наявність оцінених середніх та умовних середніх значень дає змогу визначати очікуваний горизонт звільнення місць, будувати динамічні політики бронювання та формувати тарифні стратегії, адаптовані до реального попиту. Використання регресійних і авторегресійних моделей розширює аналіз у бік сценарного прогнозування, де можливо завчасно оцінити вплив зовнішніх чинників (час доби, календарні події, сезонність) на тривалість перебування автомобілів. Таким чином, аналітичні результати стають операційним інструментом, що підтримує ухвалення рішень

при управлінні ресурсами паркування, підвищуючи їхню передбачуваність, економічну ефективність та сервісну якість.

1.3.3 Аналіз загального простору паркінгу та система сповіщень

За результатами аналізу літератури та публікацій щодо реалізації систем розумного паркування, досліджено абстракції інтерфейсів інформування користувача і внутрішнього обміну подіями [28-30].

На рисунку 1.4 представлено топ-даун карту паркінгу з маркуванням паркувальних місць.

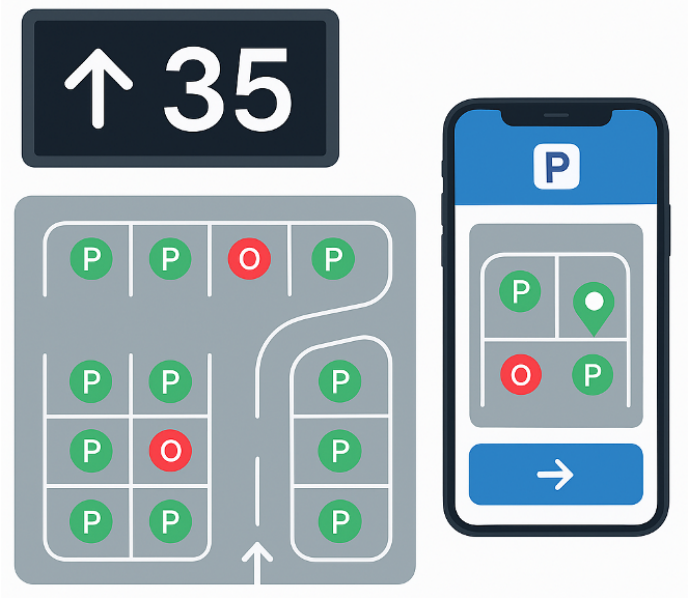

Рисунок 1.4 – Вигляд зони паркомісць з цифровим табло та мобільним інтерфейсом [30]

Перший рисунок відображає просторово-візуальний інтерфейс паркінгу в масштабі «план-зони» із зовнішнім відліком вільних місць і мобільним відображенням, тоді як другий – функціональну модель передачі подій та генерації сповіщень між сенсорною підсистемою, сервісною логікою і користувацькими каналами доставки. Нижче подано розгорнутий науковий опис кожного зображення та пояснення, чому й як ці схеми реалізують цілісну систему сповіщень для загального простору паркінгу.

Це подано у вигляді індикаторів стану: «вільне/заняте», цифрове табло загальної доступності і спрощений мобільний інтерфейс, що відображає актуальний розподіл вільних місць та дозволяє відмічати/брати місце для бронювання. Така візуалізація відповідає двом головним інформаційним задачам: швидка орієнтація водія при під'їзді (агрегований індикатор загальної доступності і зонування) та забезпечення індивідуального вибору місця через додаток. Науково така схема обґрунтована концепціями інформаційної візуалізації та когнітивної економії: агрегований показник (наприклад, кількість вільних місць на табло) знижує навантаження на прийняття рішення, а детальна мапа у мобільному інтерфейсі дає керований рівень точності для планування маршруту до місця. З архітектурної точки зору, цей рівень інтерфейсу повинен отримувати оновлення в режимі, близькому до реального часу; тому обмін даними реалізується або через push-канали для мінімізації затримок, або через короткі інтервали опитування в разі відсутності push-підтримки.

Ключові якісні характеристики, що забезпечують коректну роботу схеми, точність детекції як частка правильних класифікацій «вільне/зайняте», латентність обробки події від сенсора до відображення і ймовірність помилок. Практичні цільові орієнтири, рекомендовані в літературі, включають точність більше 95% і латентність доставки оновлення до кінцевого користувача на рівні одиниць секунд для офлайн-таборів з високим потоком; однак конкретні значення коригуються під локальні вимоги експлуатації.

На рисунку також подано абстрактний характер сххеми, що відображає послідовність перетворень події: фізичний сенсор або вузол детектує зміну стану місця, шлюз агрегує повідомлення в пакет і передає його до локальної хмари або сервісного шару, де відбувається кореляція подій, збереження транзакційного запису у базі даних та генерація повідомлень для таких споживачів як мобільні клієнти, табло, операторські панелі.

У цьому контексті система сповіщень має дві головні ролі: внутрішню з синхронізацією станів між підсистемами і збереження консистентності та зовнішню з інформуванням кінцевих користувачів і інтеграції зі сторонніми

сервісами. На рівні алгоритмів обробки подій доцільно поєднувати правило-орієнтований рушій для простих реакцій типу якщо місце вільне – помітити доступним, із черга-орієнтованою обробкою і механізмами дедуплікації для уникнення шумних повторень. Для зовнішніх сповіщень слід віддавати перевагу архітектурам, що підтримують гарантовану доставку повідомлення та класифікацію пріоритетів.

Наукові праці з розподілених сенсорних систем підкреслюють важливість метричних механізмів: час від детекції до генерації події t_{detect}, час від генерації до доставки повідомлення t_{notify}, ймовірність втрати повідомлення p_{loss} і швидкість зміни станів r_{change}. Для формалізації SLA можна використовувати вирази на кшталт $P(t_{notify} \leq T) \geq \alpha$, де T – допустима латентність наприклад, 2-5 секунд, а α – бажаний рівень виконання.

З погляду перевірки та валідації обох схем, аналітичний підхід передбачає моделювання і емпіричну оцінку ряду показників.

Поєднання статистичного аналізу історичних даних, симуляційних експериментів і польових випробувань дозволяє встановити чисельні пороги для тригерів і виявити вузькі місця. Важливим елементом аналітики є кореляція між локальними подіями та зовнішніми факторами: сезонність, час доби, заходи в місті – усі вони змінюють інтенсивність подій і вимагають адаптивної логіки генерації сповіщень.

Питання безпеки, приватності та масштабованості також є невід’ємною частиною аналізу. Транспортні ідентифікатори та дані про місце перебування мають бути захищені на всіх етапах, тоді як механізми анонімізації або агрегування можуть знижувати ризики витоку персональних даних при публічному відображенні загальної доступності.

Масштабування системи сповіщень вимагає горизонтального масштабування шлюзів і обробних сервісів, а також розумної політики кешування на фронтенді.

Поєднання усіх цих рівнів у єдиній архітектурі, доповнене чіткими метриками якості обслуговування й механізмами валідації, є необхідною умовою

для впровадження ефективної інформаційно-сповіщувальної підсистеми в загальному просторі паркінгу.

1.4 Веб-інтерфейси для систем розумних паркінгів

Веб-інтерфейси для систем розумного паркінгу відіграють роль основного каналу взаємодії між кінцевими користувачами, операторами інфраструктури та аналітичними сервісами; вони об’єднують візуалізацію стану слотів, механізми бронювання й оплати, адміністративні панелі та стріми телеметрії в єдину узгоджену екосистему.

У літературі та практиці останніх років наголошується на необхідності поєднання низької затримки оновлень реал-тайм або псевдо-реал-тайм через websocket або MQTT, надійних REST-інтерфейсів для інтеграції з зовнішніми сервісами і зручних дашбордів для моніторингу ключових показників ефективності, серед яких заповненість, тривалість стоянки та доходи від комерційної складової паркінгу (рисунок 1.5).

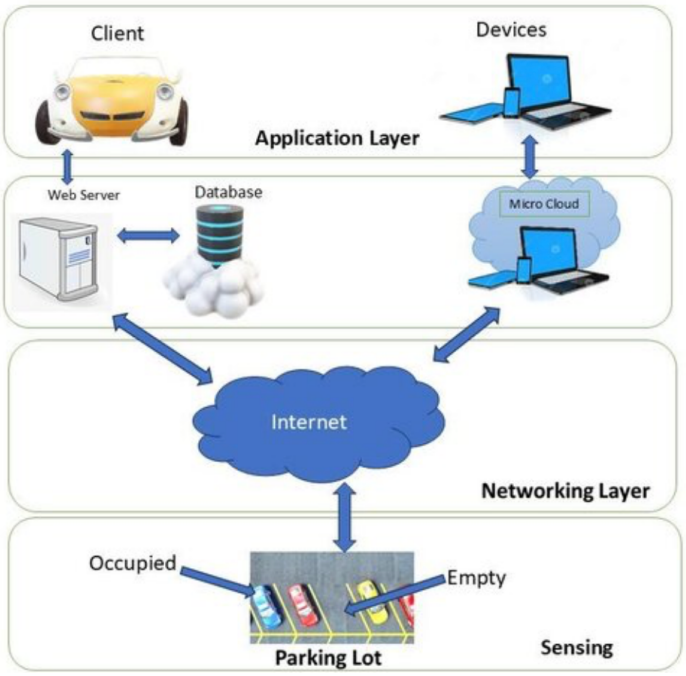

Рисунок 1.5 – Узагальнена багаторівнева архітектура веб-орієнтованої системи розумного паркінгу [31]

Це визначає архітектурні та UX-вимоги до веб-частини системи. Практичні приклади комерційних рішень демонструють [31], що інтерфейс має бути адаптивним для використання на настільних комп'ютерах та у форматі PWA, забезпечувати деталізовані фільтри та шарові карти зайнятості, а також механізми керування тарифами й зонами в режимі реального часу; такі підходи підтверджуються у публікаціях і промислових рішеннях.

Комерційний проєкт EasyPark – це приклад індустріального підходу до веб-інтерфейсів паркінгу, в якому центральною є панель Parking Dashboard (рисунок 1.6) для міст і операторів.

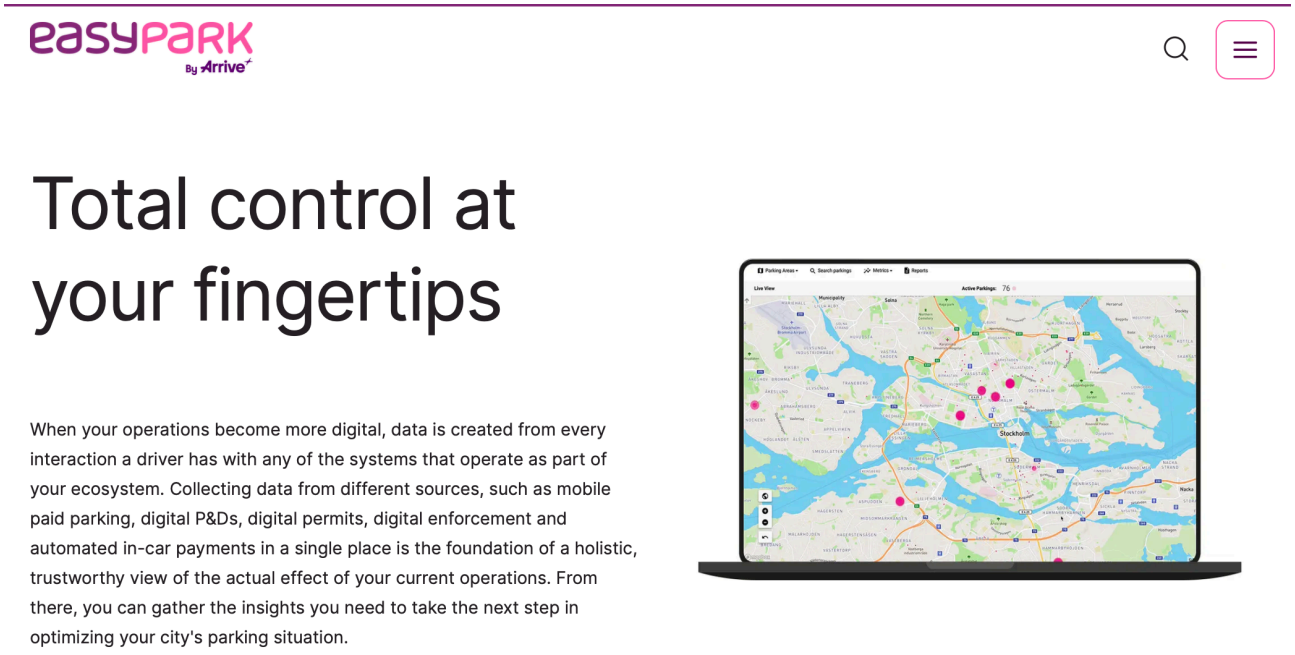

Рисунок 1.6 – Інтерфейс аналітики EasyPark із картою міського паркування [32]

Взаємодія реалізована через веб-дашборд з інструментами візуалізації зон, часовими рядами транзакцій та конфігурацією тарифних зон; архітектурно рішення поєднує збір транзакційних даних з мобільних клієнтів і зовнішніх сенсорів із аналітичними сервісами на бекенді. У цьому проєкті новизна полягає не в окремій технології, а в інженерній інтеграції великих потоків транзакцій та в інструментах для управління політиками парковки; можливе вдосконалення – більш глибока інтеграція з on-edge аналітикою для зменшення часу оновлення

заповненості та впровадження відкритих API для сторонніх аналітичних додатків, що дозволило б нашому проєкту застосувати готові патерни масштабування й безпечної мультитенант-архітектури.

У наукових роботах останніх років помітна тенденція до edge-центричних архітектур веб-інтерфейсів, коли частина попередньої обробки та інференсу перенесена ближче до камери або вбудованого контролера, а веб-інтерфейс концентрується на агрегованій візуалізації та керуванні.

Дослідження [33] з журналу Sensors описує систему з on-site inference на низькопотужному вбудованому пристрої, яка генерує слот-рівневі результати для бекенду; веб-інтерфейс у цій схемі використовується як консоль для візуалізації агрегованих результатів і для конфігурації політик збору даних. Новаторство підходу – економія пропускної здатності та зниження латентності оновлень, а можливе вдосконалення для нашої розробки полягає у визначенні гібридної моделі: локальний інференс на периферії і поступове декорумінування результатів для веб-дашборда через стандартизований WebSocket/REST шлюз, що дозволить зменшити навантаження на сервер і підвищити стабільність UX (рисунок 1.7).

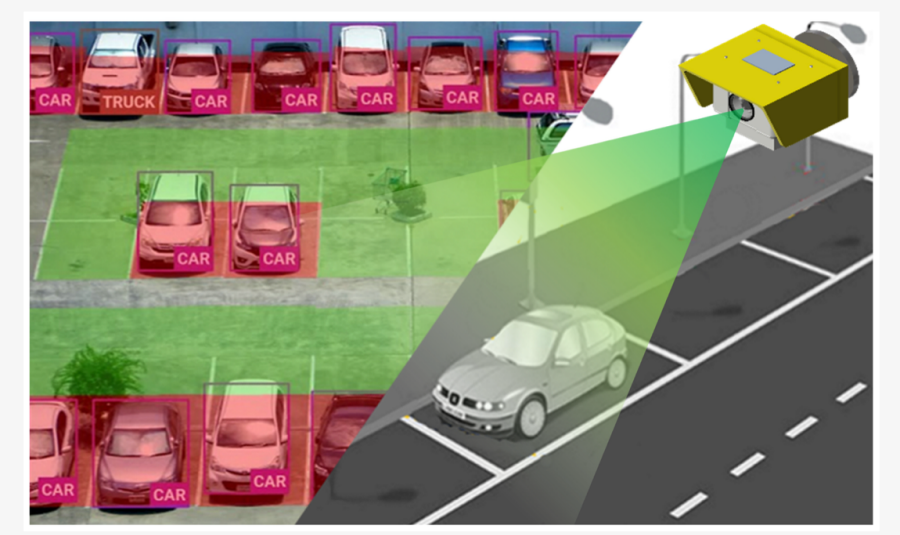

Рисунок 1.7 – Схема роботи системи комп’ютерного зору для автоматичного виявлення зайнятих та вільних паркувальних місць [33]

Підсумовуючи, під час розробки веб-інтерфейсу системи розумного паркінгу на Laravel і Filament було використано ключові підходи сучасних промислових рішень, зокрема багаторівневу панель оператора з агрегованими даними про заповненість та стан слотів у режимі, наближеному до реального часу. На основі концепцій, реалізованих у проєктах на кшталт EasyPark, створено адаптивний модульний дашборд із фільтрами, зонуванням та інтерактивними елементами.

Ураховано тенденції до edge-центричної обробки: хоча повний on-edge інференс не впроваджений, архітектура передбачає можливість подальшої інтеграції WebSocket або MQTT-каналів для передачі попередньо оброблених даних від периферійних пристроїв, що забезпечує потенціал масштабування й зменшення навантаження на сервер.

Використання Filament дало змогу реалізувати структуровану адмін-панель із CRUD-модулями, керуванням зонами, тарифами та телеметрією, спираючись на принципи комерційних дашбордів і картографічної візуалізації.

У результаті досліджених методів, можна зробити висновок, що веб-інтерфейс поєднує індустріальні підходи: оперативне оновлення, багаторівневу аналітику та адаптивність із сучасними дослідницькими напрацюваннями, що формує основу для подальшого розширення функцій і архітектури системи.

РОЗДІЛ 2
МЕТОДИКА ТА ІНСТРУМЕНТИ РОЗРОБКИ ІНТЕЛЕКТУАЛЬНОЇ СИСТЕМИ ПАРКІНГУ

2.1 Основи побудови системи розумного паркінгу

Методика розробки системи базується на компонентно-модульному підході з інтеграцією інтелектуальних алгоритмів у функціональні частини архітектури. Основна ідея полягає у створенні гнучкої багаторівневої системи, де кожен рівень відповідає за певний аспект управління паркінгом:

Рівень збору даних відповідає за первинне виявлення стану паркомісць і надходження інформації в систему. У цьому шарі використовуються сенсори, відеокамери та RFID-мітки, які фіксують факт наявності або відсутності автомобіля. Дані з фізичних пристроїв перетворюються у цифрові сигнали, що формують базу для подальшої обробки. Саме на цьому етапі визначається точність і швидкість реакції всієї системи, оскільки достовірність вхідної інформації безпосередньо впливає на ефективність наступних рівнів.

Рівень обробки та аналітики є інтелектуальним ядром системи, де відбувається аналіз даних, отриманих із сенсорів, та застосування алгоритмів машинного навчання. Серверна частина виконує обчислення, виявляє закономірності у завантаженості паркінгу, прогнозує час зайнятості місця та формує аналітичні звіти. У цьому рівні реалізується можливість адаптації системи до змінних умов, наприклад до різної інтенсивності руху в залежності від часу доби або днів тижня. Завдяки інтелектуальним моделям забезпечується не лише відображення поточного стану, а й прогнозування майбутньої ситуації.

Рівень комунікації забезпечує взаємодію між усіма компонентами архітектури та зовнішніми системами. Обмін інформацією відбувається за допомогою REST API, що дозволяє інтегрувати розроблену систему з мобільними додатками, веб-сервісами або сторонніми інформаційними платформами. Цей рівень виступає проміжною ланкою між аналітикою та користувацькими застосунками, забезпечуючи безперервний обмін даними в

режимі реального часу. Висока швидкість і стабільність передачі інформації є критично важливими для зручності користувачів та ефективності управління паркінгом.

Рівень інтерфейсу користувача формує візуальне уявлення про роботу всієї системи. Через веб- або мобільний додаток користувач отримує доступ до актуальної інформації про наявні вільні місця, може переглядати карти паркінгу, отримувати повідомлення про зміни статусу місць та рекомендації щодо оптимального вибору. Зручний і зрозумілий інтерфейс, побудований на основі сучасних принципів UX/UI, підвищує ефективність взаємодії людини з системою, зменшує час пошуку місця та сприяє загальному комфорту користування паркінгом.

Розробка системи здійснюється в ітераційній формі згідно з підходом Agile/Scrum. Такий підхід передбачає послідовне виконання циклів планування, реалізації, тестування та корекції моделей на основі отриманих результатів.

Загальна схема функціонування системи подана на рисунку 2.1.

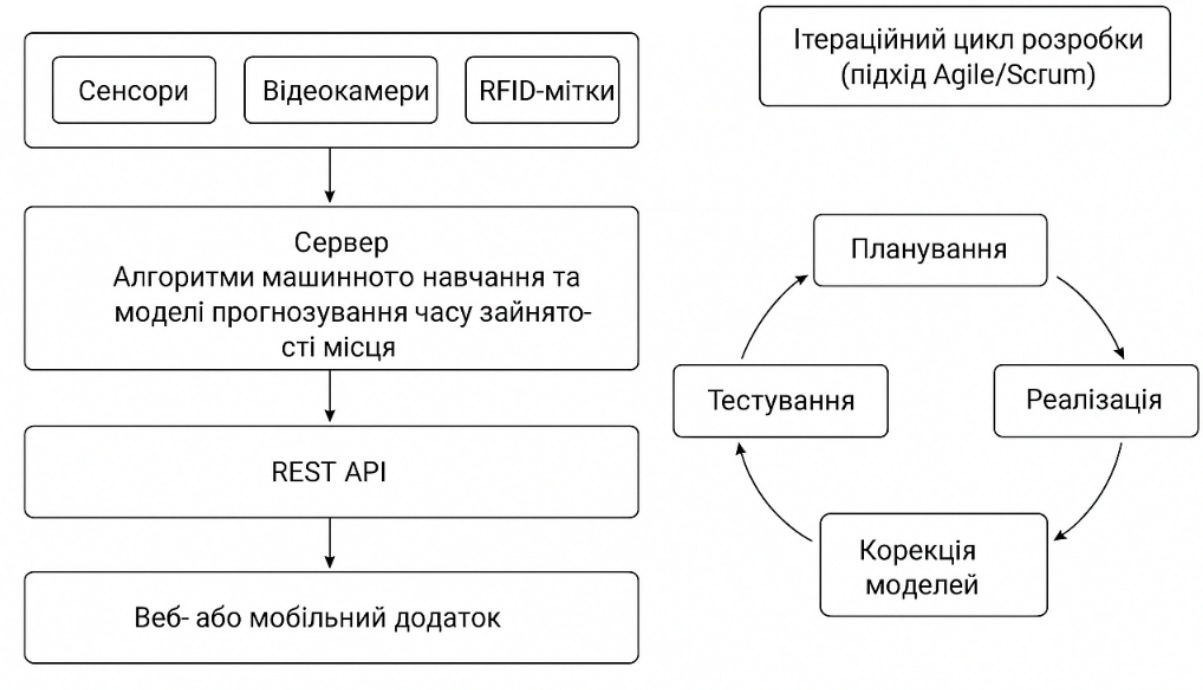

Рисунок 2.1 – Схема логіки функціонування системи управління паркінгом

На схемі відображено логіку взаємодії між основними рівнями архітектури, послідовність обміну даними та місце інтеграції інтелектуальних алгоритмів у процесі управління паркінгом.

2.2 Інструменти програмної розробки та середовище реалізації

У процесі створення системи розумного паркінгу було обрано саме ті технології, які оптимально поєднують продуктивність, гнучкість і сумісність, забезпечуючи стабільну роботу та можливість подальшого масштабування рішення. Вибір кожного інструмента є обґрунтованим як з технічного, так і з функціонального погляду.

Основною мовою для обробки даних і розробки інтелектуальних модулів обрано Python, оскільки вона має широку екосистему бібліотек для машинного навчання, статистики та комп'ютерного зору. Завдяки цьому Python ідеально підходить для реалізації моделей прогнозування зайнятості паркомісць і аналізу відеопотоку. Її простота синтаксису та численні інструменти оптимізації дозволяють швидко створювати, тестувати й оновлювати алгоритми.

Для розробки серверної частини, відповідальної за комунікацію між модулями, застосовано JavaScript, що забезпечує гнучку інтеграцію з REST API та легку взаємодію із фронтендом, тому він став оптимальним вибором замість важчих серверних платформ.

Мови HTML, CSS та React використано для створення веб-інтерфейсу користувача. Такий стек дає змогу розробити сучасний, динамічний і зручний інтерфейс, який швидко реагує на зміни даних у режимі реального часу. React було обрано як основну бібліотеку для побудови інтерфейсу завдяки її компонентній структурі, що дозволяє легко оновлювати або розширювати функціональність без порушення роботи всієї системи.

На рівні серверної логіки реалізацію виконано за допомогою Flask або FastAPI. Обидва фреймворки побудовані на Python і дозволяють швидко створювати надійні веб-сервіси з мінімальними накладними витратами. FastAPI

відзначається високою швидкодією та підтримкою асинхронних викликів, що є перевагою для обробки великої кількості запитів, тоді як Flask забезпечує гнучкість і простоту розгортання. Саме така комбінація дає змогу створити легку, але потужну серверну частину.

Для побудови та тренування моделей штучного інтелекту використано TensorFlow і PyTorch – два провідні фреймворки в галузі машинного навчання. Вибір цих бібліотек обумовлений їхньою стабільністю, підтримкою GPU-обчислень і великою кількістю готових архітектур нейронних мереж. Вони дозволяють швидко адаптувати моделі під конкретні завдання, наприклад, прогнозування заповненості паркінгу або класифікацію об’єктів на зображеннях із камер.

Для роботи з відеопотоком обрано OpenCV, тому що ця бібліотека має великий набір інструментів для аналізу та обробки зображень у реальному часі. Вона дозволяє розпізнавати автомобілі, визначати їхню присутність на паркомісцях і передавати ці дані на сервер для подальшого аналізу. На відміну від інших рішень, OpenCV добре інтегрується з Python і TensorFlow, що спрощує побудову комплексних алгоритмів.

Для командної роботи і контролю версій коду використано GitHub, який забезпечує прозорість і безпечне зберігання історії змін коду.

У сукупності, цей набір інструментарію технологій формує цілісну, гнучку та надійну архітектуру розумного паркінгу, здатну працювати в режимі реального часу, адаптуватися до навантажень і підтримувати інтеграцію з іншими інтелектуальними міськими сервісами.

2.3 Технологічні основи серверної частини системи розумного паркінгу

Для реалізації серверної частини системи розумного паркінгу було обрано технологічний стек, що поєднує мову програмування PHP, фреймворк Laravel та адміністративну панель Filament. Такий вибір було здійснено з огляду на потребу

забезпечити стабільну серверну логіку, структуровану архітектуру та зручні інструменти адміністрування.

Використання PHP було обумовлено його широким застосуванням у веб-розробці, наявністю великих можливостей інтеграції з реляційними базами даних та здатністю ефективно обробляти значні обсяги запитів. У межах даного проекту PHP було залучено як основне середовище виконання серверної логіки, що забезпечує реєстрацію заїздів та виїздів автомобілів, взаємодію з REST API, опрацювання інформаційних потоків та підготовку даних для аналітичних модулів.

Для структуризації коду, підвищення масштабованості та реалізації чіткої логіки було використано фреймворк Laravel. Його застосування дозволило впорядкувати архітектуру системи та забезпечити гнучку роботу з даними завдяки вбудованій ORM Eloquent. Через цей механізм було визначено та реалізовано основні сутності системи: автомобілі, паркомісця, зони паркінгу та історичні записи подій.

Окрім цього, було передбачено створення REST API за допомогою маршрутизації Laravel, що дало змогу забезпечити стандартизовані запити й відповіді, а також розмежувати доступ за допомогою middleware.

Для обробки подій у режимі реального часу було використано систему подій і слухачів, що дозволило автоматизувати оновлення стану паркінгу та передавати відповідні зміни під’єднаним клієнтам. Для виконання довготривалих або ресурсомістких операцій було залучено механізм черг Laravel, що підвищило загальну продуктивність системи та забезпечило плавність її роботи під навантаженням.

Для забезпечення адміністрування системи було обрано панель Filament, яка надає комплекс інструментів для керування даними та моніторингу стану паркінгу. Використання Filament дозволило створити зручний інтерфейс керування конфігурацією паркувальних зон, переглядом історії подій, контролем поточного навантаження та внесенням необхідних змін до параметрів роботи системи. Завдяки генерації CRUD-ресурсів адміністративна панель була швидко

адаптована до структури моделей Laravel. Крім того, було залучено можливості Filament Widgets для формування інформаційних панелей та відображення статистичних даних, що забезпечило адміністраторам доступ до оперативної аналітики. Інтеграція Filament із системою автентифікації Laravel дозволила передбачити розмежування доступів та забезпечити належний рівень безпеки (рисунок 2.2).

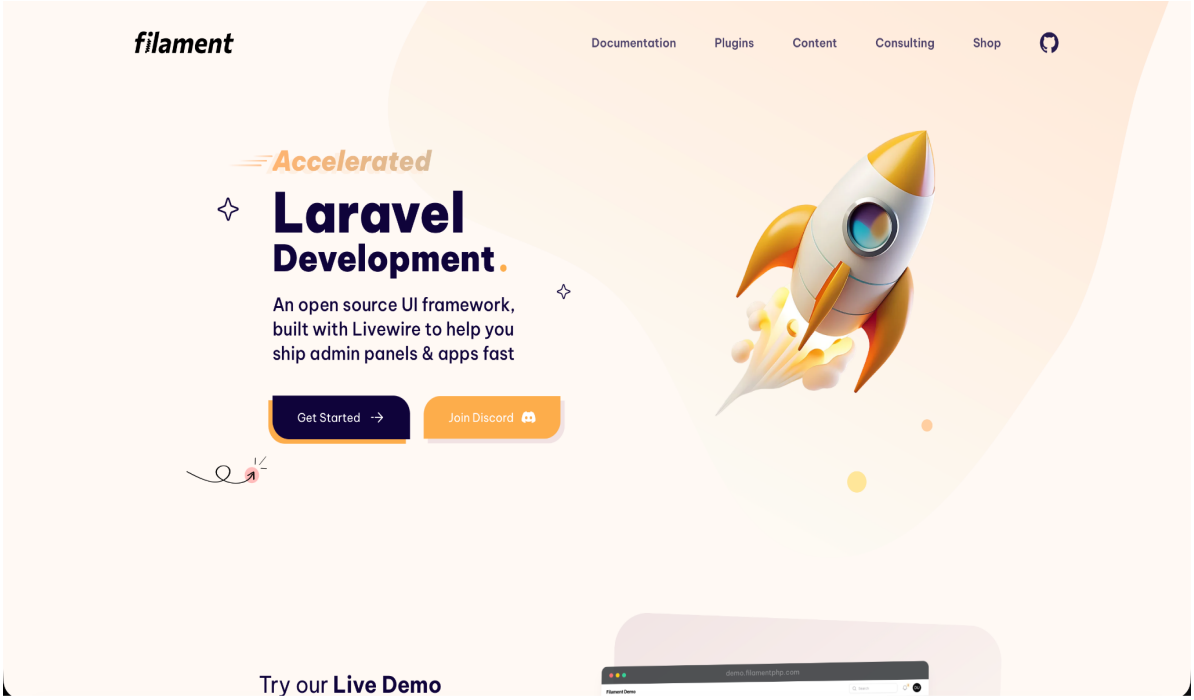

Рисунок 2.2 – Сторінка офіційного сайту Filament

Обрана інтеграція компонентів дозволила реалізувати систему, що відповідає вимогам надійності, масштабованості та готовності до інтеграції з модулями штучного інтелекту.

Загалом, поєднання PHP, Laravel та Filament було обрано як оптимальне рішення для побудови серверної частини системи розумного паркінгу.

У розробці кваліфікаційної роботи PHP забезпечив основу серверної логіки, Laravel – структуровану та безпечну архітектуру, а Filament – інструментарій для ефективного управління даними.

2.4 Підготовка IDE та інструментів розробки

Для забезпечення ефективної та структурованої роботи над системою розумного паркінгу було здійснено підготовку середовища розробки, що включала встановлення та налаштування IDE, інструментів командного рядка, серверних компонентів і допоміжних утиліт. На цьому етапі було визначено перелік програмних засобів, необхідних для повноцінної реалізації проєкту, його тестування та подальшої підтримки.

Як основне інтегроване середовище розробки було обрано Visual Studio Code, оскільки воно забезпечує гнучкість конфігурації, підтримку великої кількості розширень та інтеграцію з Git. У процесі налаштування було встановлено низку спеціалізованих розширень, зокрема для роботи з PHP, Laravel, Blade-шаблонами та базами даних. Додатково було передбачено встановлення інструментів статичного аналізу коду, що дозволило забезпечити контроль якості програмного забезпечення та дотримання стандартів PSR.

Для роботи фреймворку Laravel було встановлено PHP необхідної версії, а також пакетний менеджер Composer, через який здійснюється керування залежностями проекту. Під час підготовки робочого середовища було налаштовано локальний сервер, реалізований за допомогою Laravel Sail або альтернативно – XAMPP/Valet залежно від платформи. Це забезпечило можливість запуску застосунку в ізольованому середовищі, наближеному до умов реального серверного розгортання.

Для роботи з базою даних було використано MySQL. Було створено окрему робочу базу, налаштовано доступи, а також встановлено графічний клієнт phpMyAdmin, що значно спростило візуальний аналіз структур даних та проведення тестових маніпуляцій. Всі параметри підключення було визначено у файлі конфігурацій .env відповідно до вимог безпеки Laravel.

Для управління репозиторієм було залучено систему контролю версій Git, а також створено віддалене сховище на GitHub. Це забезпечило можливість зберігання історії змін, організації командної роботи та контролю за внесенням

оновлень. Попередньо було налаштовано Git-ігнорування службових файлів, що дозволило уникнути зайвих конфліктів та забезпечити чистоту репозиторію.

Окрему увагу було приділено налаштуванню середовища для роботи з адміністративною панеллю Filament. Було встановлено необхідні пакети, додаткові розширення, а також модулі для створення віджетів та інтерактивних панелей управління. Завдяки цьому було підготовлено інструмент, який дозволяє адмініструвати систему паркінгу в реальному часі.

Таким чином, підготовка IDE та інструментів розробки включала комплекс дій, спрямованих на створення повноцінного локального середовища, необхідного для розробки, тестування та подальшого масштабування системи. Виконані налаштування забезпечили ефективну роботу з кодом, стабільність виконання додатка та можливість безперервної інтеграції нових модулів і функцій.

РОЗДІЛ 3
ПРОЕКТУВАННЯ ВЕБ-ІНТЕРФЕЙСУ ЗАСОБАМИ FILAMENT

3.1 Налаштування серверу та деплой проекту на віддалений сервер

Процес деплою програмного забезпечення є критичним етапом життєвого циклу розробки, оскільки саме він забезпечує перенесення системи з середовища розробки до продуктивної інфраструктури.

Для систематизації цього процесу було сформовано послідовність операцій, яка охоплює всі необхідні етапи підготовки, розгортання та перевірки працездатності системи

Процес деплою розпочинається з підготовки серверного середовища та встановлення основних компонентів інфраструктури. На цьому етапі перевіряється конфігурація сервера, виконується оновлення системних пакетів та встановлюється веб-сервер разом із необхідним рантаймом, зокрема PHP та його модулями. Також створюється база даних, користувачі та політики доступу, що забезпечує готовність середовища до приймання програмного коду.

Наступним кроком є перенесення проєкту на сервер і встановлення залежностей. Вихідні файли застосунку завантажуються через SSH, FTP або систему CI/CD, після чого виконується інсталяція бібліотек та оновлення конфігураційних параметрів. На цьому етапі проводяться міграції бази даних, імпорт необхідних структур та запускається збірка статичних ресурсів, якщо застосунок містить фронтенд-компоненти.

Завершальний етап охоплює конфігурацію серверних служб та фінальну перевірку працездатності. Налаштовуються права доступу до файлової системи, створюється конфігурація віртуального хоста, а всі сервіси перезапускаються для застосування змін.

Після цього виконується тестування доступності та коректності роботи застосунку, а також впроваджуються механізми моніторингу й логування для контролю стабільності системи у продуктивному середовищі.

Етапи циклу розробки наведені в таблиці 3.1.

Таблиця 3.1 – Структурована таблиця алгоритму деплою

№ п/п	Етап проекту	Опис
1	Підготовка середовища	Перевірити конфігурацію сервера, наявність ОС та доступів
2	Оновлення системи	Оновити системні пакети та компоненти безпеки
3	Встановлення веб-сервера	Розгорнути веб-сервер і виконати базове налаштування
4	Інсталяція PHP або іншого рантайму	Встановити потрібну версію PHP та модулі
5	Налаштування бази даних	Створити базу даних і користувача, налаштувати доступ
6	Передача файлів проєкту	Завантажити вихідний код на сервер
7	Встановлення залежностей	Встановити необхідні бібліотеки
8	Конфігурація середовища	Оновити файли конфігурації та змінні середовища
9	Міграції БД	Запустити міграції або імпорт структури БД
10	Збірка фронтенду	Виконати збірку статичних файлів
11	Налаштування прав доступу	Призначити коректні права для веб-сервера
12	Налаштування віртуального хоста	Створити конфігурацію домену та кореневої директорії
13	Перезапуск сервісів	Перезапустити веб-сервер і супутні сервіси
14	Тестування роботи	Перевірити доступність та функції застосунку
15	Моніторинг і логування	Налаштувати логування, моніторинг та сповіщення

На сьогодні чимала кількість компонентів вже є готовими і реалізованими, тобто, певні інтерфейсні або програмні компоненти вже створені розробниками раніше та доступні для повторного використання без необхідності писати їх з нуля. Тому було прийнято рішення використати деякі базові компоненти, такі як таблиці, списки та кнопки з перемикачами, для реалізації базової адмін панелі.

Для підготовки адміністративної панелі було виконано встановлення Filament за допомогою пакетного менеджера Composer. Перед початком встановлення необхідно перейти в папку з проектом, де раніше було виконано створення пустого Laravel проекту за допомогою таких команд: composer -V та composer create-project laravel/laravel myproject.

Виконаємо наступний ряд команд для встановлення адмін панелі composer require filament/filament, після чого буде завантажено повний пакет файлів розширення.

Після додавання пакета до проекту було виконано публікацію конфігурацій та початкових ресурсів за допомогою: php artisan filament:install і остання команда, яка створить для нас початкового базового користувача php artisan make:filament-user. На рисунку 3.1 відображено результат ідентифікації акаунту. Для перевірки потрібно перейти за адресою localhost/admin, перед тим попередньо запустивши веб-сервер.

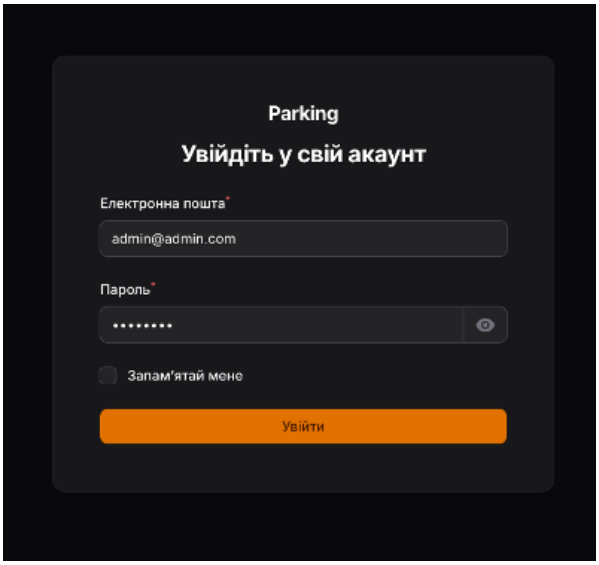

Рисунок 3.1 – Форма входу адміністративної панелі Filament системи Parking

На рисунку 3.2 представлено діалогове вікно створення DNS-запису в адміністративній панелі сервісу хостингу. Форма дозволяє вказати субдомен, обрати тип DNS-запису (зокрема, тут вказано A-запис), а також задати відповідне значення – IP-адресу, на яку має спрямовуватися доменне ім’я.

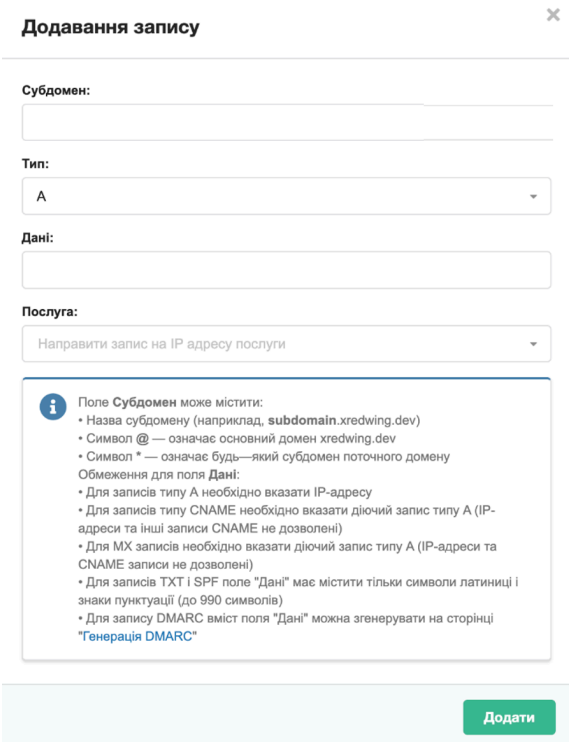

Рисунок 3.2 – Інтерфейс додавання DNS-запису домену в панелі керування хостингом

Інтерфейс містить довідковий блок, що пояснює правила формування субдомену, призначення окремих типів записів та обмеження щодо їх синтаксису. Такий елемент є важливою частиною інфраструктурної підготовки проєкту, оскільки забезпечує коректне налаштування зв’язку між доменним ім’ям і сервером, на якому розгортається майбутня адміністративна панель.

Заповнена форма додавання DNS-запису наведена на рисунку 3.2, де в полі субдомену вказано ім’я tech, а у полі «Дані» – IP-адресу виділеного VPS-сервера. Система автоматично підтягує назву відповідного серверного ресурсу, що спрощує взаємозв’язок між доменом та інфраструктурними елементами.

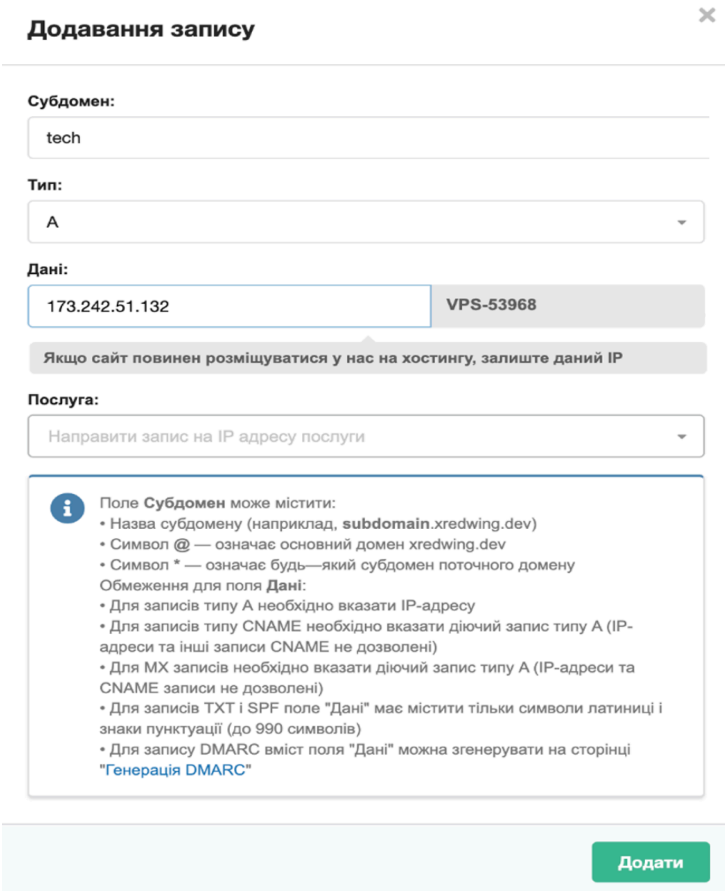

Рисунок 3.3 – Інтерфейс додавання DNS-запису домену в панелі керування

Така процедура є необхідною для забезпечення маршрутизації HTTP-запитів до правильного сервера, що є критично важливим під час розгортання Laravel-проєкту та подальшої інтеграції Filament-панелі в робоче середовище. Коректно налаштований DNS дає змогу використовувати доменну адресу для доступу до адміністративного інтерфейсу, що є невід'ємною частиною професійного розгортання системи.

На рисунку 3.4 представлено модуль автоматизованого встановлення програмного забезпечення на сервер. Інтерфейс дозволяє обрати операційну систему, визначити необхідний набір додаткових компонентів, зокрема Fastpanel, а також додати SSH-ключі для подальшого безпечного доступу. В проекті це Linux-подібна операційна система Ubuntu 22.04. Дана процедура є обов'язковою для підготовки серверного середовища, у якому розгортатиметься Laravel-додаток та Filament-адмін панель.

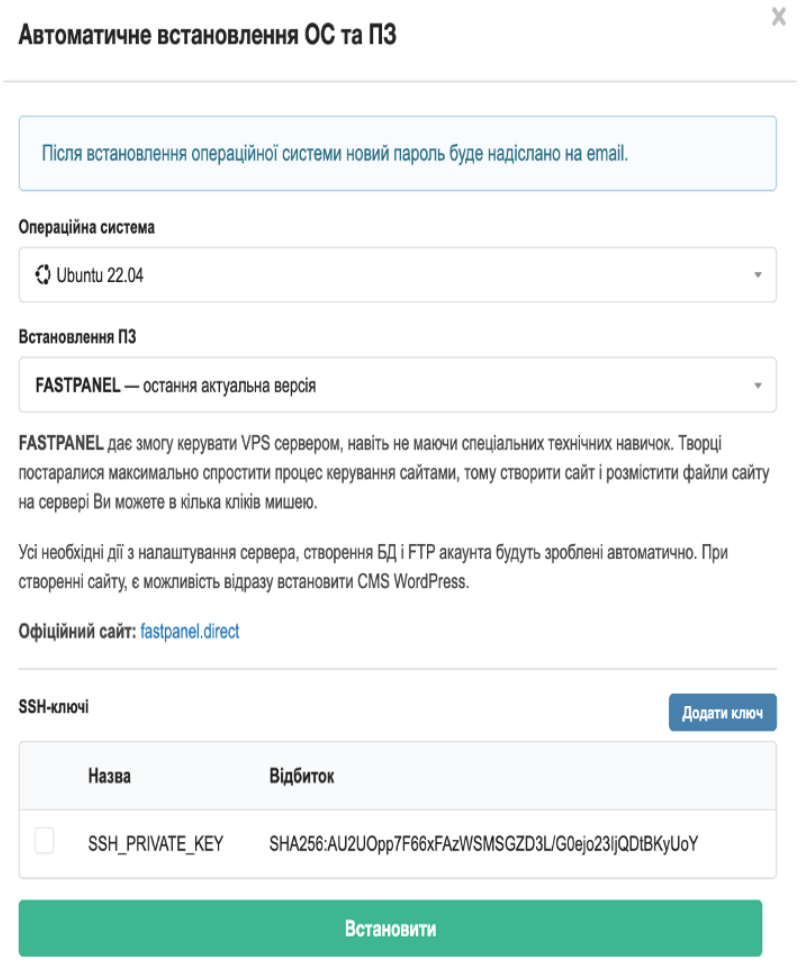

Рисунок 3.4 – Автоматичне встановлення операційної системи та програмного забезпечення на VPS-сервері

Описані на рисунках процеси формують основу для стабільної роботи системи управління. Налаштування DNS гарантує доступність проєкту за доменним ім’ям, а підготовка серверного середовища забезпечує коректне функціонування програмної частини, включно з Filament. Без цих кроків створення, тестування та впровадження адміністративної панелі було б неможливим або суттєво ускладненим.

Автоматизація інсталяції дозволяє мінімізувати ризики помилок, забезпечує відповідність середовища вимогам фреймворку та прискорює процес налаштування інфраструктури.

Використання стандартизованих інструментів дає змогу мінімізувати ймовірність помилок, що можуть виникати під час ручного конфігурування, а

також гарантує відповідність параметрів системи вимогам програмного забезпечення та застосованого фреймворку.

На початковому етапі виконується інсталяція операційної системи та розгортання панелі керування Fastpanel, що суттєво прискорює подальше налаштування серверного середовища, оскільки забезпечує зручний інтерфейс для адміністрування (рисунок 3.5).

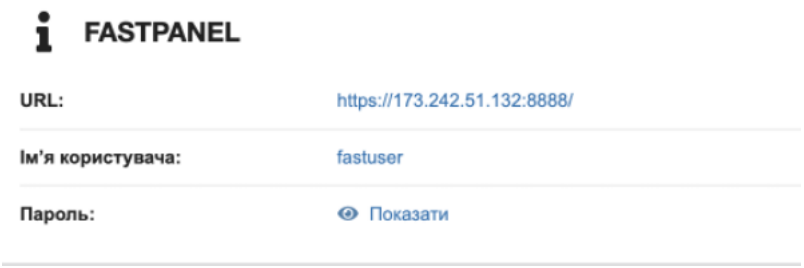

Рисунок 3.5 – Інтерфейс Fastpanel після встановлення операційної системи та серверної панелі керування

Після завершення встановлення адміністратор переходить у веб-інтерфейс Fastpanel (рисунок 3.6).

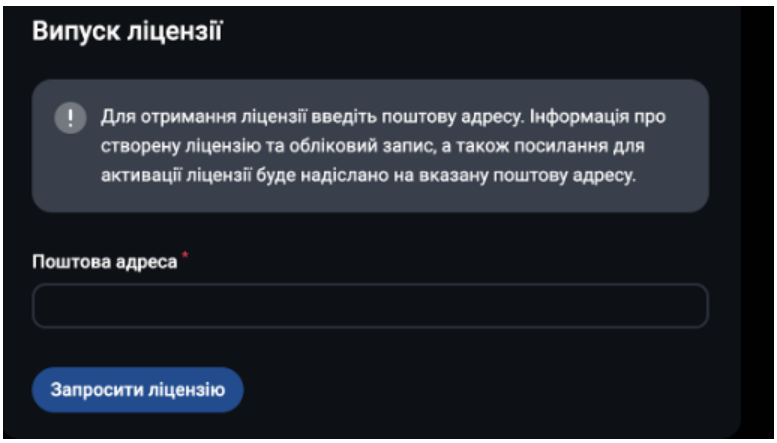

Рисунок 3.6 – Форма запиту безкоштовної ліцензії у панелі керування Форма запиту безкоштовної ліцензії у панелі керування

На цьому етапі система ініціює процедуру отримання ліцензії, яка є безкоштовною. Для її випуску користувач вводить адресу електронної пошти, на яку надходить інформація щодо створеного облікового запису та посилання для

активації ліцензійного ключа. Налаштування дозволяє централізовано керувати інстанціями програмного забезпечення, забезпечуючи їх автентичність та контрольоване використання. Після активації ліцензії здійснюється конфігурація компонентів програмного середовища. Одним із ключових параметрів є вибір версії PHP, на основі якої функціонуватиме веб-застосунок.

У розділі налаштування, у вкладці управління PHP-додатками, адміністратор обирає та встановлює необхідну версію інтерпретатора PHP відповідно до вимог програмного продукту (рисунок 3.7).

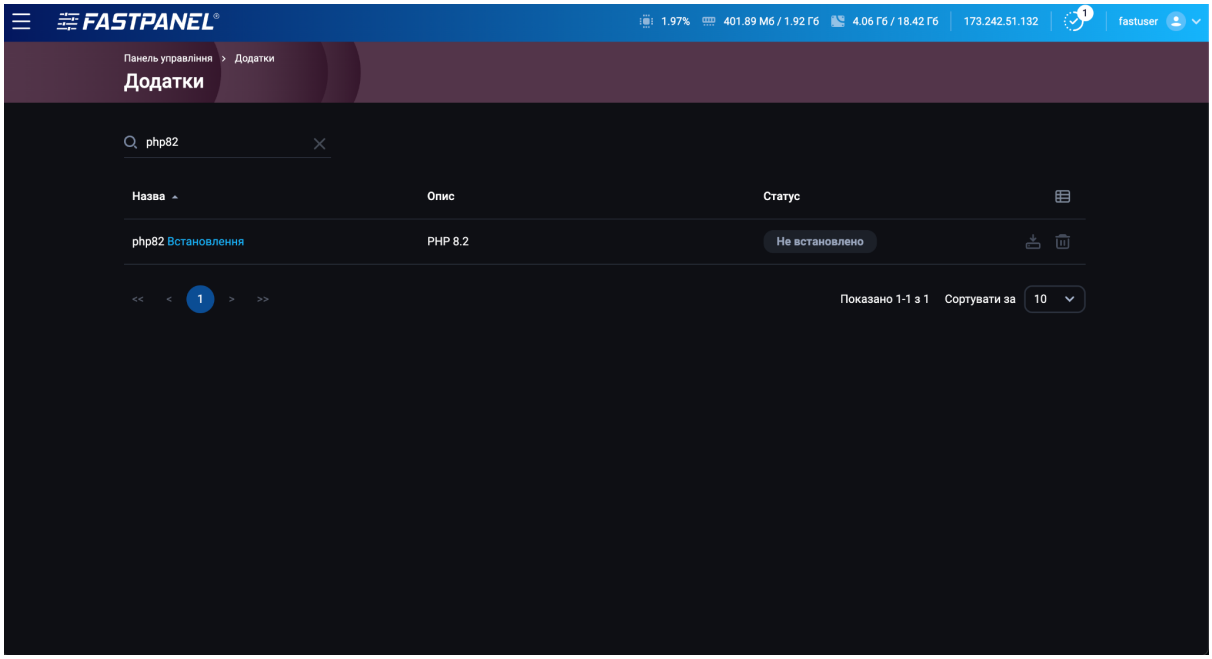

Рисунок 3.7 – Вибір та встановлення версії PHP у модулі керування додатками

Коректність цього вибору має важливе значення для сумісності з фреймворком, бібліотеками та модулями, які використовуються у додатку, та забезпечує стабільність його виконання у продуктивному середовищі. Інтерфейс першого етапу майстра створення вебсайту у Fastpanel, який відповідає за підключення доменного імені наведено на рисунку 3.8.

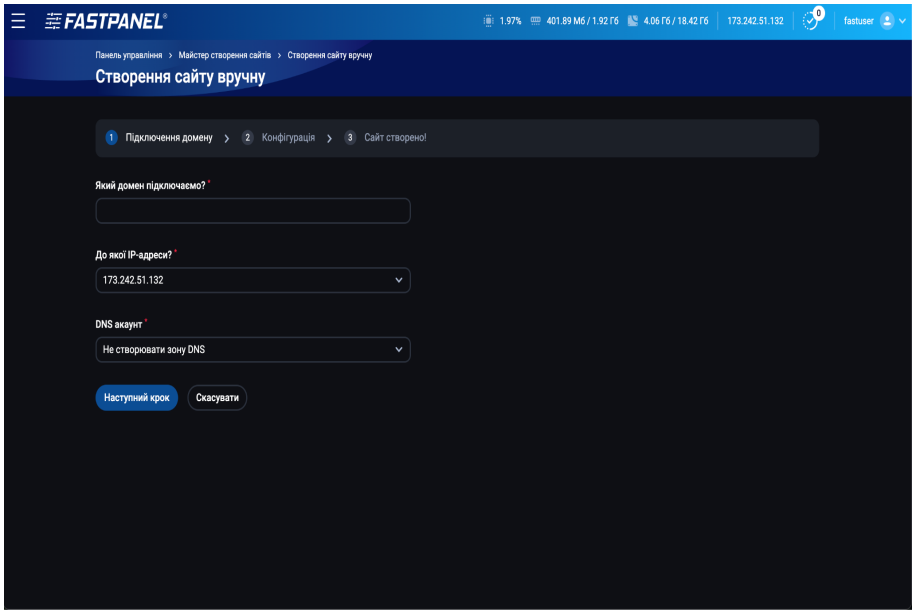

Рисунок 3.8 – Інтерфейс майстра ручного створення вебсайту у панелі керування

На цьому кроці адміністратор визначає домен, що буде прив’язаний до вебресурсу, та обирає IP-адресу сервера, на яку надходитимуть запити від користувачів. Така прив’язка забезпечує коректну маршрутизацію HTTP-трафіку та є передумовою для подальшого налаштування сервера.

Окремо передбачено параметр DNS-аккаунта, який дозволяє автоматично створити або не створювати DNS-зону для домену. Якщо зона не створюється через панель, адміністратор може управляти DNS-записами на стороні реєстратора або зовнішнього DNS-провайдера. Після внесення початкових даних користувач переходить до наступного етапу налаштування вебсайту за допомогою кнопки «Наступний крок», що ініціює конфігурацію параметрів хостингу.

Наведені етапи налаштування серверного середовища, конфігурації доменних служб, встановлення системного програмного забезпечення та інтеграції керуючої панелі Fastpanel є критично важливими для забезпечення коректного розгортання, стабільності та подальшої експлуатації програмного продукту. Вони формують технологічну основу, на якій функціонує

Laravel-застосунок і адміністративна панель Filament, гарантують узгодженість параметрів серверної інфраструктури з вимогами фреймворку та мінімізують ризики помилок, пов’язаних із некоректною маршрутизацією, невідповідністю версій програмних компонентів чи неправильною конфігурацією доступів.

3.2 Розробка і проектування інтелектуального інтерфейсу користувача

Після встановлення Filament сформовано базову адміністративну панель. Доступ до цієї панелі здійснюється через стандартну сторінку автентифікації (рисунок 3.9). Через відповідний інтерфейс уповноважені користувачі отримують доступ до функціональних модулів системи.

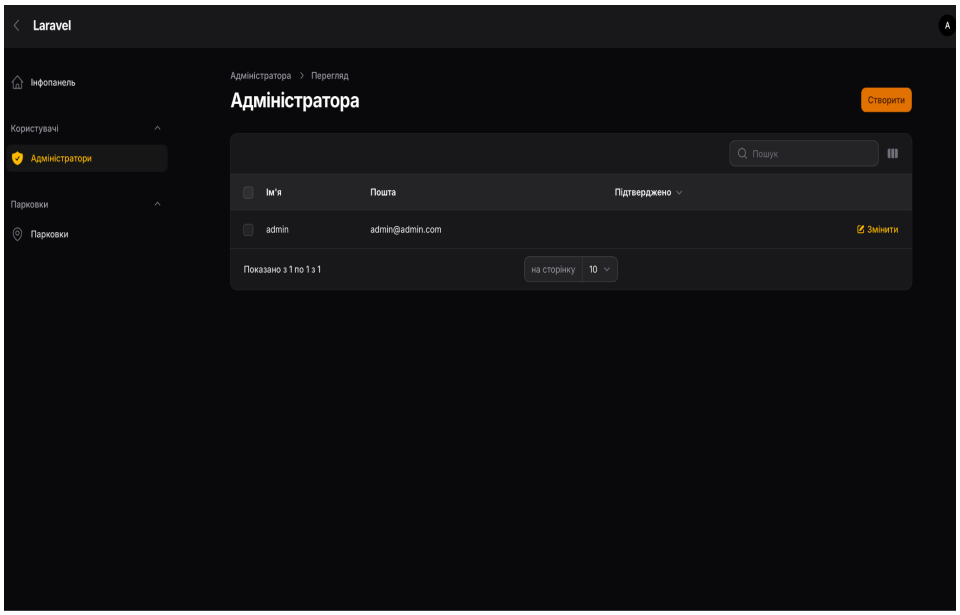

Рисунок 3.9 – Налаштування адміністратора паркінгу

Одним із первинних етапів конфігурації є створення та налаштування облікового запису адміністратора та інтерфейс редагування. Це забезпечує визначення відповідальної особи та надання їй необхідних повноважень для подальшого адміністрування. Після виконання початкових налаштувань адміністративного доступу сформовано модуль управління паркомайданчиками.

Зведена інформація про наявні майданчики відображається у вигляді структурованого списку (рисунок 3.10), який містить основні параметри кожного об’єкта.

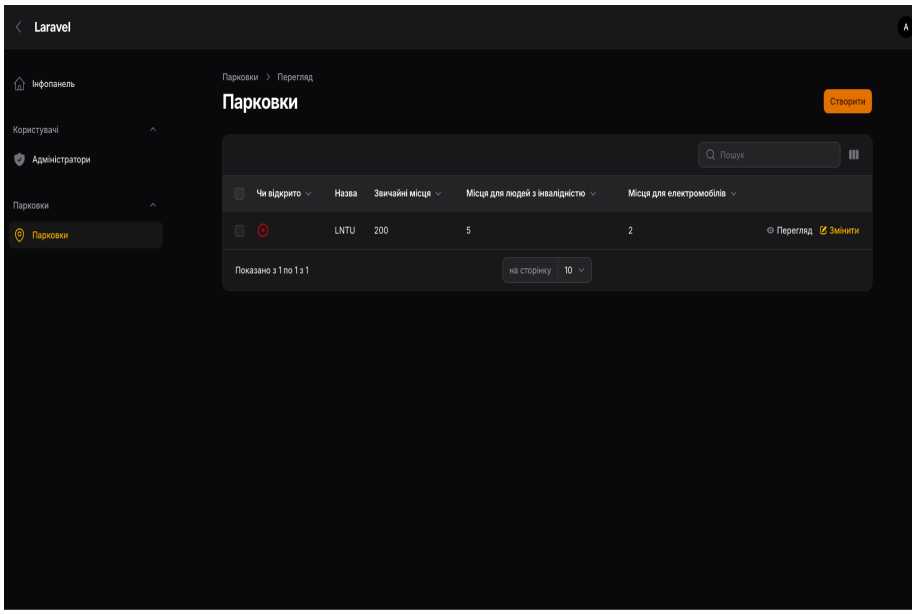

Рисунок 3.10 – Список паркомайданчиків

Для детального керування передбачено окрему сторінку конфігурації, що надає можливість змінювати схему розміщення місць і параметри режимів роботи (рисунок 3.11).

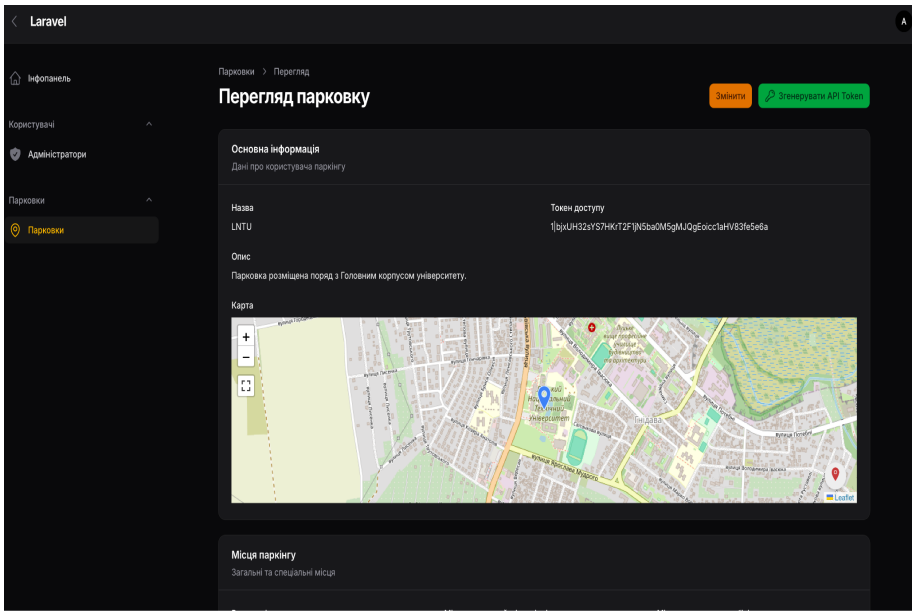

Рисунок 3.11 – Налаштування схеми паркінгу та годин роботи

Додатковий інтерфейс редагування (рисунок 3.12) забезпечує доступ до розширених параметрів, включно зі структурою зон, кількістю паркувальних місць та правилами функціонування майданчика.

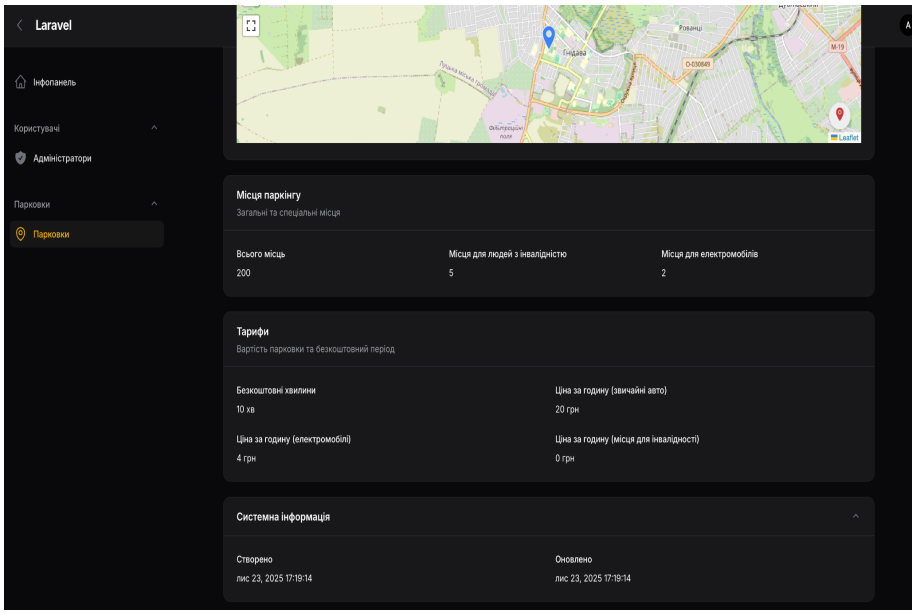

Рисунок 3.12 – Налаштування схеми паркінгу та годин роботи

Створення нових адміністраторів реалізовано через спеціалізовану форму (рисунок 3.13), яка надає можливість додавання користувачів та призначення їм відповідних ролей.

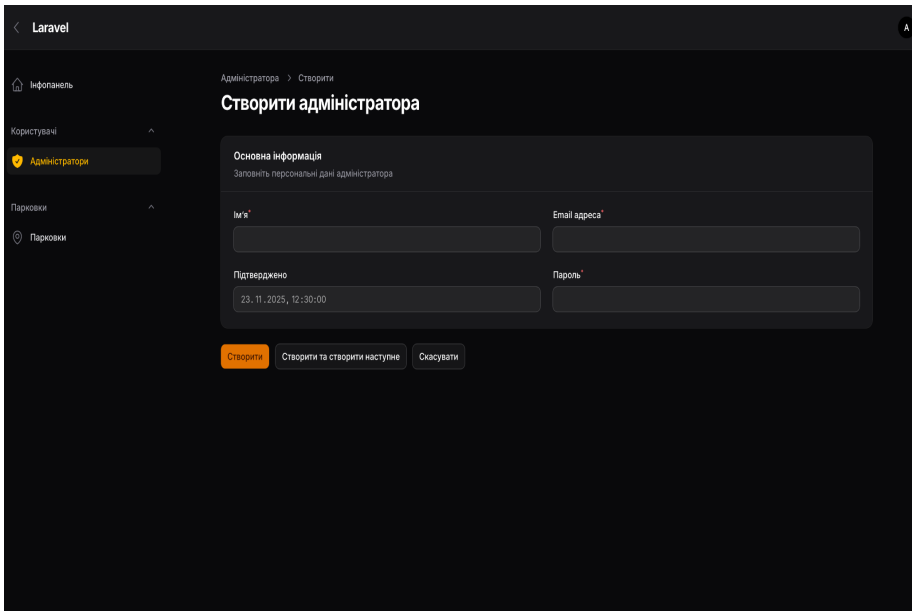

Рисунок 3.13 – Створення ролі адміністратора

У разі потреби здійснюється повторна генерація токена доступу, що гарантує безпечну взаємодію із серверною частиною та API (рисунок 3.14).

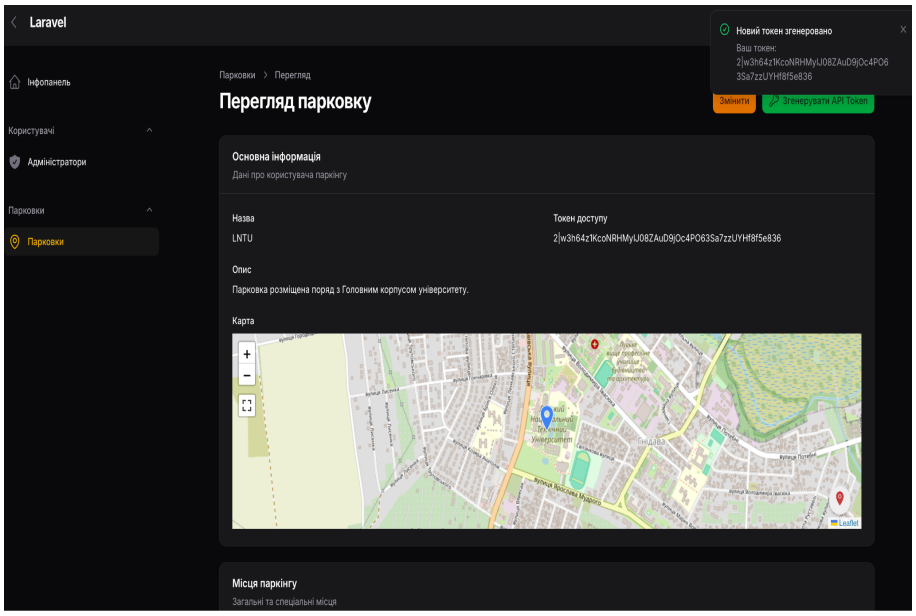

Рисунок 3.14 – Повторна генерація токену доступу

Налаштування окремих параметрів конкретного паркомайданчика, зокрема його статусу й режимів роботи, виконується за допомогою спеціального редактора (рисунок 3.15), який забезпечує точність і повноту конфігурації інфраструктурних елементів системи.

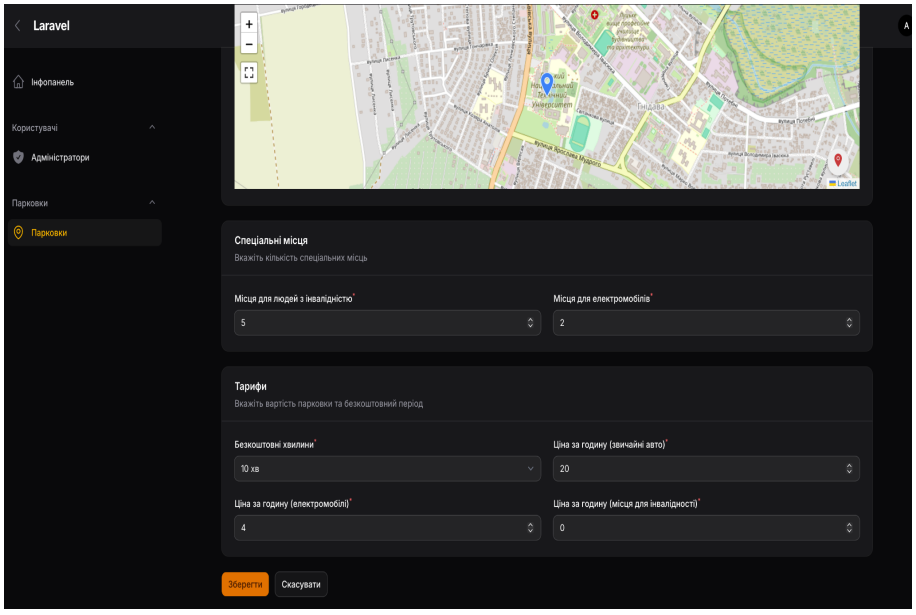

Рисунок 3.15 – Режим зміни паркомайданчика

Структуровані інтерфейси адміністрування спрощують конфігурацію паркомайданчиків, облікових записів та параметрів взаємодії з API. Отриманий функціонал створює цілісне середовище для ефективного управління інфраструктурою та подальшого розширення системи.

3.3 Проектування REST API системи

Для інтеграції зовнішніх клієнтів було реалізовано REST API, перелік доступних запитів якого подано у вигляді окремого каталогу. Одним із базових запитів є отримання інформації про конкретний паркінг, що забезпечує доступ клієнтів до актуальних даних щодо його поточного стану. Додатково передбачено запит оновлення профілю адміністратора або оператора, який використовується для коригування параметрів облікових записів.

Оперативне керування робочим станом паркінгу реалізовано через API-запити відкриття та закриття, що визначають можливість прийому нових транспортних засобів та впливають на логіку обробки сесій паркування. У процесі тестування системи були виявлені помилкові сценарії, зокрема некоректна фіксація заїзду автомобіля, що дало змогу перевірити коректність механізмів валідації та обробки виняткових ситуацій.

Основною функціональною операцією є фіксація заїзду на паркінг, під час якої створюється нова сесія паркування, прив’язана до конкретного автомобіля. Завершення сесії передбачає виконання запиту оплати, після чого система оновлює стан місця та передає користувачу інформацію про завершення надання послуги.

Для забезпечення взаємодії клієнтських застосунків із серверною частиною системи реалізовано REST API, запити якого виконуються через протокол HTTPS. Передавання даних здійснюється у форматі JSON із використанням заголовка Content-Type: application/json, а доступ до захищених ендпоінтів забезпечується токеном, що передається в заголовку Authorization. Базовим ендпоінтом є отримання профілю користувача за допомогою методу

GET, який не потребує тіла запиту та повертає структурований JSON-об’єкт із даними користувача, прив’язаними автомобілями та доступними паркомайданчиками. Аналогічно реалізовано ендпоінт оновлення профілю, що здійснюється методом POST із переданням у тілі запиту JSON-структури з оновленими параметрами (лістинг 3.1).

Лістинг 3.1 – JSON-об’єкт з параметрами паркінгу

```
{
   "name" : "test2",
   "description" : "test2",
   "number_of_spaces" : 25,
   "number_of_spaces_for_disabled" : 10,
   "number_of_spaces_for_electric_cars" : 10,
   "free_minutes" : 40,
   "hour_price" : 40,
   "electric_hour_price" : 20,
   "disabled_hour_price" : 0
}
```

кінець лістингу 3.1

Система керування паркінгом використовує низку REST-запитів, кожен із яких відповідає за певний етап роботи: зміну статусу паркінгу, обробку в’їздів, виїздів та оновлення конфігурації. Нижче наведено розширений опис логіки, яку виконує сервер.

Сервер отримує дані від клієнтів у вигляді JSON-об’єктів, зчитує передані параметри та проходить кілька етапів валідації. Спочатку перевіряється повнота та коректність даних, наприклад тип транспортного засобу, формат державного номерного знаку, наявність вільних місць або валідність ідентифікатора паркомайданчика. Якщо перевірки успішні, сервер оновлює відповідний запис у базі даних – змінює конфігурацію паркінгу або додає нову активну сесію

паркування. Якщо ж валідація завершується помилкою, запит повертає повідомлення про некоректні дані або відхиляється.

Для централізованого керування роботою майданчика передбачено окремі POST-запити відкриття та закриття паркінгу. Такі запити використовуються виключно адміністраторами системи. У тілі запиту передається ідентифікатор паркінгу та додаткові параметри – наприклад час, з якого повинен змінитися його статус. Після надходження запиту на відкриття сервер фіксує зміну стану та переводить паркінг у режим обслуговування клієнтів. При запиті на закриття структура переданих даних залишається аналогічною, однак перед зміною статусу система додатково перевіряє, чи немає активних неоплачених сесій. Якщо такі сесії існують, закриття блокується або переноситься до моменту завершення розрахунків.

Найважливішими елементами функціонування системи є запити заїзду та виїзду транспортних засобів. Запит заїзду створює нову паркувальну сесію: сервер перевіряє тип автомобіля, наявність вільних місць відповідної категорії та поточний статус паркінгу. У разі успіху система резервує місце, фіксує час в’їзду та записує інформацію про транспортний засіб у базу даних. Запит виїзду завершує сесію: на основі часу перебування та тарифів сервер обчислює суму до сплати, перевіряє коректність завершення, оновлює статистику паркінгу та звільняє паркомісце. Таким чином, саме ці два запити забезпечують основний робочий цикл паркінгу та правильну взаємодію між клієнтами та системою.

Для фіксації заїзду було реалізовано метод (лістинг 3.2), у якому передаються дані автомобіля та ідентифікатор паркомайданчика.

Лістинг 3.2 – JSON-об’єкт з параметрами транспортних засобів

```
{
  "type_vehicle" : "standard",  // in standard, electric, disabled
  "vehicle_number" : "AC 0000 BC"
}
```

кінець лістингу 3.2

На основі отриманих даних створюється нова сесія паркування, а у відповіді сервер повертає унікальний ідентифікатор сесії та, за потреби, попередньо розрахований прогнозований час завершення паркування.

Для фіксації виїзду використовується окремий ендпоінт, також реалізований методом POST. У тілі запиту необхідно передати ідентифікатор раніше створеної сесії та фактичний час виїзду (лістинг 3.3).

Лістинг 3.3 – JSON-об’єкт з описом автомобіля з парковки

```
{
  "type_vehicle" : "standard",  // in standard, electric, disabled
  "vehicle_number" : "AC 0000 BC"
}
```

кінець лістингу 3.3

Система на основі цих даних обчислює фактичну тривалість перебування, формує суму до оплати відповідно до тарифів та готує інформацію для наступного запиту оплати. Окремим ендпоінтом було реалізовано запит оплати паркування. Він також використовує метод POST, а в тілі містить ідентифікатор сесії та параметри оплати (лістинг 3.4).

Лістинг 3.4 – JSON-об’єкт операції оплати паркінгу

```
{
  "type_vehicle" : "standard",  // in standard, electric, disabled
  "vehicle_number" : "AC 0000 BC",
  "amount" : 4
}
```

кінець лістингу 3.4

Після успішної обробки запиту система змінює статус сесії на оплачений, фіксує транзакцію та повертає клієнту підтвердження у вигляді JSON-відповіді з реквізитами платежу.

Для інтеграції з модулем штучного інтелекту було передбачено окремий запит аналізу, який реалізовано методом GET. У цьому випадку параметри, пов’язані з паркінгом (наприклад, ідентифікатор, часовий інтервал чи режим аналізу), передаються через рядок запиту, після чого сервер формує запит до моделі GPT і повертає JSON із результатами аналітики та прогнозу завантаженості.

Усі запити REST API мають уніфіковану структуру, що включає код статусу операції, текстове повідомлення та, за наявності, об’єкт data з корисним навантаженням (рисунок 3.16).

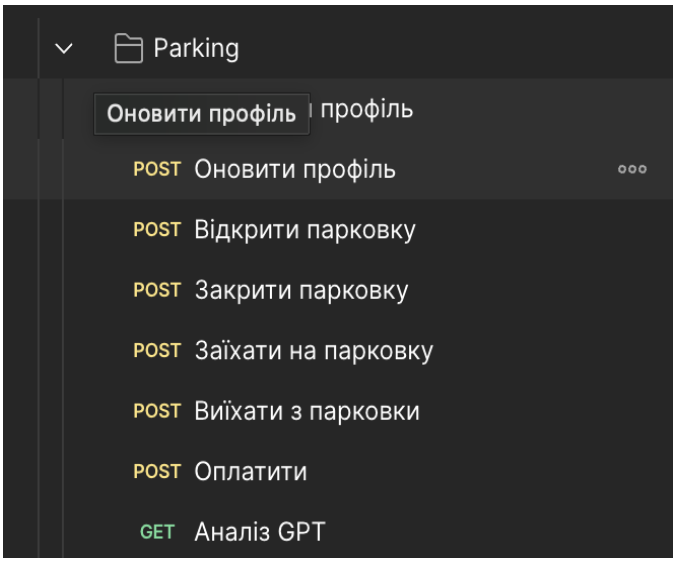

Рисунок 3.16 – Список API запитів до системи

У разі помилок валідації або порушення бізнес-логіки система повертає JSON із описом помилки, що дає змогу клієнтським застосункам коректно обробляти виняткові ситуації. Такий підхід до проєктування REST API забезпечує передбачувану поведінку системи, спрощує інтеграцію з зовнішніми сервісами та дозволяє безпечно й ефективно керувати всіма операціями, пов’язаними з роботою паркінгу.

Реалізована система REST API забезпечує повноцінну та безпечну взаємодію клієнтських застосунків із серверною частиною паркінгу, дозволяючи керувати конфігурацією майданчиків, фіксувати в’їзди та виїзди транспортних засобів, обробляти оплату та отримувати аналітичні дані. Використання єдиного формату передачі даних у JSON, стандартизованих ендпоінтів та механізму аутентифікації через токен гарантує коректність і передбачуваність роботи системи, спрощує інтеграцію з зовнішніми сервісами та забезпечує надійне керування всіма операціями, пов’язаними з функціонуванням паркінгу.

ВИСНОВКИ

За результатами виконаного дослідження сформульовано наступні висновки.

Проаналізовано існуючі підходи до побудови систем розумного паркінгу, визначено їх ключові технологічні особливості, переваги та обмеження, що дозволило обґрунтувати вибір оптимальних методів інтеграції алгоритмів штучного інтелекту, аналітики даних і веб-технологій у проєктовану систему.

Розроблено архітектуру системи розумного паркінгу, яка включає серверну частину, адміністративну панель, клієнтські інтерфейси та механізми інтеграції зовнішніх застосунків через REST API, що забезпечує централізоване управління, масштабованість та гнучкість взаємодії між компонентами системи.

Забезпечено автоматизацію налаштування серверного середовища та деплой веб-застосунку з використанням сучасних інструментів адміністрування та пакетних менеджерів, що підвищує швидкість розгортання та мінімізує ризики помилок у конфігурації.

Реалізовано прототип системи з можливістю обробки відеопотоку, аналізу статистичних даних і візуалізації результатів роботи, що дозволяє проводити моніторинг завантаженості паркінгів і оптимізувати управління інфраструктурою.

Оцінено ефективність та точність роботи моделі, підтверджено відповідність алгоритмів вимогам прогнозування та обробки даних у реальному часі, а також визначено області для подальшого вдосконалення.

Проведено тестування системи, перевірено коректність функціонування всіх модулів, забезпечено стабільність роботи в продуктивному середовищі та підтверджено надійність взаємодії користувачів із системою.

Як підсумок, можна стверджувати, що мету роботи досягнуто, а система інтелектуального паркінгу забезпечує ефективне управління паркомайданчиками, автоматизований моніторинг завантаженості, інтеграцію зовнішніх клієнтських застосунків та надійну взаємодію користувачів із

сервісом. Впроваджені алгоритми обробки даних і прогнозування дозволяють оптимізувати використання інфраструктури, а реалізовані адміністративні та клієнтські інтерфейси забезпечують гнучкість керування та доступність інформації у реальному часі. Отримана система демонструє стабільність роботи, точність функціонування модулів та готовність до подальшого розширення й інтеграції нових технологічних рішень.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Христинець Н. А., Биков С. О., Марчук О. Ю. Прототипи інтелектуальних систем безпеки і управління інфраструктурою на базі IoT-платформ. *Комп'ютерно-інтегровані технології: освіта, наука, виробництво.* 2025. №60. С. 63-69.

2. An Analysis of Artificial Intelligence Techniques in Surveillance Video Anomaly Detection: A Comprehensive Survey / E. Şengönül et al. *Applied Sciences*. 2023. Vol. 13, no. 8. P. 49-56.

3. Improved KNN-based Stock Price Prediction. *Academic Journal of Computing & Information Science*. 2024. Vol. 7, no. 6. P. 128-147.

4. UK turns against Chinese Hikvision surveillance cameras. *Biometric Technology Today*. 2022. Vol. 2022, no. 4. P. 99-113.

5. Rick Mullin. AI developer Nvidia invests in Recursion. *C&EN Global Enterprise*. 2024. Vol. 101, no. 23. P. 8-9

6. NVIDIA Launches Affordable Gen AI Supercomputer. *The Engineer*. 2025. Vol. 302, no. 7963. P. 40-58.

7. Tencent A. L. H. Deep multiple instance learning-enabled gene mutation prediction of lung cancer from histopathology images. 2024. Vol. 7, no. 2. P. 24-39.

8. Goli V. R. Cross-Platform Mobile Development: Comparing React Native and Flutter, and Accessibility in React Native. *International Journal of Innovative Research in Computer and Communication Engineering*. 2023. Vol. 11, no. 03. P. 311-324.

9. Duong H.-T., Le V.-T., Hoang V. T. Deep Learning-Based Anomaly Detection in Video Surveillance: A Survey. *Sensors*. 2023. Vol. 23, no. 11. P. 117-126.

10. Zhang W., Lazaro J. P. A Survey on Network Security Traffic Analysis and Anomaly Detection Techniques. *International Journal of Emerging Technologies and Advanced Applications*. 2024. Vol. 1, no. 4. P. 8-16.

11. Singh R., Gill S. S. Edge AI: A survey. *Internet of Things and Cyber-Physical Systems*. 2023. URL: https://doi.org/10.1016/j.iotcps.2023.02.004 (date of access: 19.07.2025).

12. Kuchuk H., Malokhvii E. Integration of IOT with Cloud, Fog and Edge Computing: a Review. *Advanced Information Systems*. 2024. Vol. 8, no. 2. P. 65-78.

13. Privacy-Preserving Video Anomaly Detection: A Survey / Y. Liu et al. *IEEE Transactions on Neural Networks and Learning Systems*. 2025. P. 1-22. URL: https://doi.org/10.1109/tnnls.2025.3600252 (date of access: 28.07.2025).

14. Alioanei C., Popescu N. AI-Based Solutions for Security and Resource Optimization in IoT Environments: A Systematic Review. *Information*. 2025. Vol. 16, no. 10. P. 841-857.

15. Kwak N., Lee B. Deep Learning Applied Abnormal Human Behavior Detection in Video Surveillance Systems - A Survey. *International JOURNAL OF CONTENTS*. 2024. Vol. 20, no. 4. P. 84-95.

16. Survey: Anomaly Detection in Surveillance Videos / E. A. Mahareek et al. *International Journal of Theoretical and Applied Research*. 2024. Vol. 3, no. 1. P. 328-342.

17. Zhang H.. Hikvision Price Prediction based on ARIMA, OLS Multiple Regression and Random Forest. *Highlights in Science, Engineering and Technology*. 2024. Vol. 88. P. 546-552.

18. A Comparison of Various Class Balancing and Dimensionality Reduction Techniques on Customer Churn Prediction / S. P. Bhushan Mada et al. *2022 IEEE 7th International Conference on Recent Advances and Innovations in Engineering (ICRAIE)*, MANGALORE, India, 1-3 December 2022. 2022. URL: https://doi.org/10.1109/icraie56454.2022.10054321 (date of access: 28.07.2025).

19. Plosz S., Hegedus C., Varga P. Advanced Security Considerations in the Arrowhead Framework. *Lecture Notes in Computer Science*. Cham, 2021. P. 234-245. URL: https://doi.org/10.1007/978-3-319-45480-1_19 (date of access: 28.07.2025).

20. Денисенко А. Вступ до REST API-REST. Ful вебсервіси: robot_dreams: онлайн-курси для фахівців у сфері Big Data, Machine Learning, Data Science.

Робот Дрімс. URL: https://robotdreams.cc/uk/blog/466-vstup-do-rest-api-restful-vebservisi (дата звернення: 28.07.2025).

21. Pradeep J. Efficient Vehicle Detection System using OCR and REST API. *International Journal of Engineering Research*. Vol. V9, no. 07. P. 19-31.

22. Edge Computing and Cloud Computing for Internet of Things: A Review / F. C. Andriulo et al. *Informatics*. 2024. Vol. 11, no. 4. P. 312-342.

23. Security-aware Data-driven Intelligent Transportation Systems / J. Malik et al. *IEEE Sensors Journal*. 2020. P. 1-84.

24. Researcher. Blockchain Technology: Advancing Data Integrity And Security In Modern Systems. *International Journal of Research In Computer Applications and Information Technology (IJRCAIT)*. 2024. Vol. 7, no. 2. P. 1574-1593.

25. Нападій О., Цьопа Н., Батрак Є. Система автоматизації пошуку місць для паркування. *Адаптивні системи автоматичного управління*. 2023. Т. 2, № 43. С. 140-153.

26. Лисак В. В., Деркач М. В., Скарга-Бандурова І. С. Інтеграція алгоритмів транспортної логістики для сервісу паркування. *Вісник східноукраїнського національного університету імені володимира даля*. 2021. № 8. С. 5-9.

27. Goli V. R. React native evolution, native modules, and best practices. *International journal of computer engineering and technology*. 2021. Vol. 12, no. 2. P. 73-85.

28. Орда О., Бондаренко Н., Бондаренко В. Пристрій визначення зайнятості паркомісць в системі смарт-паркінгу. *Herald of Khmelnytskyi National University. Technical sciences*. 2025. Т. 355, № 4. С. 414-418.

29. Пашкевич С., Никончук В., Кристопчук М. Оцінка пропускної спроможності міської дорожньої мережі з урахуванням пропозиції паркування. *Сучасні технології в машинобудуванні та транспорті*. 2023. Т. 2, № 21. С. 163-175.

30. Олійник В., Данилюк Ю. Інтелектуальна модель автономного паркування для різних типів паркувальних місць на основі глибокого навчання з

підкріпленням. *Адаптивні системи автоматичного управління*. 2025. Т. 1, № 46. С. 237-246.

31. Eladl S. G. Effective Smart parking systems. *Nile Journal of Communication and Computer Science.* 2025. Vol. 9, no. 1. P. 1-34.

32. EasyPark Parking Dashboard. *Die Park-App, die Ihnen mehr Zeit für andere Dinge gibt EasyPark.* URL: https://www.easypark.com/en-de/cities-and-operators/our-services/easypark-parking-dashboard?utm_source=chatgpt.com (date of access: 25.07.2025).

33. Real-Time Parking Space Management System Based on a Low-Power Embedded Platform / K. Kim et al. *Sensors*. 2025. Vol. 25, no. 22. P. 70-89.