Вінницький національний технічний університет
Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки

МАГІСТЕРСЬКА КВАЛІФІКАЦІЙНА РОБОТА

на тему:

РОЗПОДІЛЕНА СИСТЕМА З ПІДТРИМКИ ФУНКЦІОНУВАННЯ АВТОПАРКІНГУ

Виконала: студентка 2 курсу, групи 2КІ-22м
спеціальності 123 — «Комп'ютерна інженерія»
_____ Кривенька В. О.

Керівник: к.т.н., доц.каф. ОТ
_____ Тарновський М. Г.

«07» 12 2023 р.

Опонент: к.т.н., доц. каф. ПЗ
_____ Ткаченко О. М.

«08» 12 2023 р.

Допущено до захисту

Завідувач кафедри ОТ
_____ д.т.н., проф. Азаров О. Д.
«11» 12 2023 р

ВНТУ 2023

ВІННИЦЬКИЙ НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ

Факультет інформаційних технологій та комп'ютерної інженерії
Кафедра обчислювальної техніки
Галузь знань — Інформаційні технології
Освітній рівень — магістр
Спеціальність — 123 Комп'ютерна інженерія
Освітньо-професійна програма — Комп'ютерна інженерія

ЗАТВЕРДЖУЮ

Завідувач кафедри обчислювальної техніки

О.Д. Азаров

"26" вересня 2023 р.

З А В Д А Н Н Я

НА МАГІСТЕРСЬКУ КВАЛІФІКАЦІЙНУ РОБОТУ

студентці Кривенькій Вікторії Олегівні

1 Тема роботи «Розподілена система з підтримки функціонування автопаркінгу» керівник роботи Тарновський Микола Геннадійович к.т.н., доцент, затверджено наказом вищого навчального закладу від **18.09.23** року № **247.**

2 Строк подання студентом роботи **11.12.23**.

3 Вихідні дані до роботи: призначення : інформування користувача про наявність вільних місць на парковці, засоби — середовище програмування Visual Studio Code , мови програмування React, JavaScript, Node.js.

4 Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити): вступ, аналіз технологій для підтримки функціонування автопаркінгу, вибір та обгрунтування підходів до організаціїї розподіленої системи, розробка розподіленої системи для підтримки функціонування автопаркінгу, рекомендації з розгортання системи.

5 Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень): архітектура системи, структурна схема системи, ER-діаграма бази даних, блок-схема алгоритму.

6 Консультанти розділів роботи приведені в таблиці 1.

Таблиця 1— Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
1-4	Тарновський Микола Геннадійович к.т.н., доцент		
5	Небава Микола Іванович проф., к.е.н		

7 Дата видачі завдання **19.09.2023** .

8 Календарний план виконання МКР приведений в таблиці 2.

Таблиця 2 — Календарний план

№ з/п	Назва етапів МКР	Строк виконання	Підпис
1	Постановка задачі	**19.09.23**	
2	Огляд існуючих рішень	**20.09.23**	
3	Розробка структурної схеми	**28.09.23**	
5	Розрахунок аналогової частини	**5.10.23**	
6	Вибір ПЗ для розробки	**15.10.23**	
7	Розробка роботи системи	**23.10.23**	
8	Розрахунок економічної частини	**2.11.23**	
9	Оформлення пояснювальної записки та ілюстративного матеріалу	**12.11.23**	
10	Виконання магістерської кваліфікаційної роботи	**25.11.23**	
11	Перевірка якості виконання магістерської кваліфікаційної роботи та усунення недоліків	**4.12.23**	
12	Підписи супроводжувальних документів у керівника, опонента, нормоконтролера	**8.12.23**	
13	Перевірка «антиплагіат»	**8.12.23**	
14	Попередній захист	**12.12.23**	

Студент Кривенька Вікторія Олегівна

Керівник к.т.н., доц. Тарновський Микола Геннадійович

АНОТАЦІЯ

УДК 004

Кривенька В. О. Розподілена система з підтримки функціонування автопаркінгу. Магістерська кваліфікаційна робота зі спеціальності 123 — Комп’ютерна Інженерія, Вінниця: ВНТУ, 2023 — 106 с. На укр. мові. Бібліогр.: 27 назв; рис.: 24; табл. 8.

У роботі розглянуто технології для підтримки функціонування автопаркінгу, вибрано та обгрунтуванно підходи до організаціїї розподіленої системи, розроблено клієнтську та серверну частину розподіленої системи для підтримки функціонування автопаркінгу, написані рекомендації з розгортання системи.

Ключові слова: парковка, автопаркінг, система, веб-застосунок.

ABSTRACT

УДК 004

Kryvenka V.O. Distributed system to support parking functionality. Master's thesis in the field of 123 — Computer Engineering, Vinnitsa: VNTU, 2023 – 106 p. In Ukrainian. Bibliography: 27 titles; figures: 24; tables: 8.

The paper explores technologies to support parking functionality, selects and justifies approaches to organizing a distributed system, develops the client and server components of a distributed system to support parking functionality, and provides recommendations for system deployment.

Keywords: parking, car parking, system, web application.

ЗМІСТ

					08-54.МКР.031.00.000 ПЗ			
Змн.	*Арк.*	*№ докум.*	*Підпис*	*Дата*				
Розробила		Кривенька В. О.			Розподілена система з підтримки функціонування автопаркінгу. Пояснювальна записка	*Літ.*	*Аркуш*	*Аркушів*
Керівник		Тарновський М.					6	106
Опонент		Ткаченко О.М.				ВНТУ, гр. 2КІ-22м		
Н.контр.		Швець С. І.						
Затвердж.		Азаров О.Д						

ВСТУП

Формування спеціалізованих зон для стоянки автотранспорту почалося зі збільшенням кількості автомобілів на вулицях міст. Тому сьогодні парковки є невід'ємною частиною міської інфраструктури. Головною ознакою теперішнього часу є високі темпи розвитку автомобілізації, що збільшує автомобільний трафік на вулицях та породжує проблеми, пов’язані з паркуванням. Відсутність вільних місць для паркування, особливо в центральних районах у робочий час, є серйозною проблемою сучасного великого міста.

Вирішенню проблем з пошуком місця для паркування сприяє розвиток комп’ютерних технологій, які дозволяють відслідковувати наявність вільних місць в дозволених для паркування зонах. Це надає можливість формувати мапу вільних місць для надання цієї інформації водіям через спеціальні табло або мобільні застосунки.

З іншого боку сучасні технології дозволяють автоматизувати управління автопаркінгами, роблячи автомобільні парковки комфортними для користувачів і економічно ефективними для їх власників. Комп’ютеризоване управління заїздом та виїздом з автоматичним розрахунком вартості надає можливість підвищити пропускну здатність та скоротити експлуатаційні витрати. За рахунок формування фінансових, статистичних та технічних звітів вдається покращити прозорість функціонування автопаркінгу, приймати виважені управлінські рішення для покращення ефективності його роботи.

Актуальність теми дослідження полягає в тому, що завдяки сучасним Інтернет технологіям користувачі отримують можливість здійснювати пошук і бронювання місця для паркування свого автомобіля онлайн. Це не лише сприяє зменшенню витратам на паливо і часу пошуку місця для паркування, а й, як наслідок, зменшенню заторів на дорогах та викидів шкідливих речовин у повітря. Поряд із цим, комп’ютерні системи дозволяють повністю автоматизувати процес функціонування парковки,

починаючи від управління заїздом на її територію та завершуючи розрахунком вартості послуги з використанням гнучкої системи тарифікації, запобігти зловживанням з боку користувачів та обслуговуючого персоналу тощо.

Метою роботи є розширення функціональних можливостей системи з підтримки автопаркінгу за рахунок хмарних технологій та більш гнучкого принципу розподілу вільних місць, що дозволить зробити користування послугами паркінгу більш зручним для клієнтів.

Для досягнення поставленої мети у роботі розв’язуються такі **задачі**:

— аналіз сучасних технології для підтримки функціонування автопаркінгу;

— вибір та обгрунтування підходів до побудови розподіленої системи, розробка її структурної схеми, вибір відповідного обладнання;

— розробка веб-застосунку для відслідковування наявності вільних місць для паркування в режимі онлайн.

Об'єктом дослідження є процес моніторингу зайнятості місць для паркування.

Предметом дослідження є електронні та інформаційні засоби для підтримки функціонування автопаркінгу.

Методи дослідження: методи системного та статистичного аналізу для розрахунку статистичних даних, методи схемотехнічного та алгоритмічного проектування для розробки структурної схеми розподіленої системи.

Новизна роботи полягає в тому, що набув подальшого розвитку принцип використання хмарних технологій для надання інформації про наявність вільних місць для паркування, в якому на відміну від існуючих використовуються статистичні данні для оцінювання імовірності наявності вільного місця протягом певного часового інтервалу, що дозволяє зменшити кількість відмов у наданні послуг паркування безпосередньо перед заїздом на парковку.

Практичне значення роботи полягає в тому, що впровадження запропонованої системи дозволяє скоротити час пошуку місяця для парковки, що сприятиме зменшенню паразитного автомобільного трафіку на вулицях міста.

Апробація результатів роботи здійснена у доповідях на LI Науково-технічній конференції підрозділів Вінницького національного технічного університету (2022) та Міжнародній науково-практичній Інтернет-конференції «Електронні інформаційні ресурси: створення, використання, доступ» (2023) [1] , [2] .

1 АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ ПІДТРИМКИ ФУНКЦІОНУВАННЯ АВТОПАРКІНГУ

1.1 Автопаркінг як складова інфраструктури сучасного міста

Автопаркінг — це спеціально облаштована територія або споруда, призначена для паркування автотранспортних засобів. Це місце, де водії можуть тимчасово залишити свої автомобілі, коли вони не використовуються [3].

Автопаркінги можуть бути розташовані в різних місцях, включаючи центральні частини міст, бізнес-центри, торгові центри, аеропорти, залізничні станції та інші об'єкти. Їхнє обладнання може включати в себе майданчики для паркування, багаторівневі гаражі, підземні паркінги, а також додаткові сервіси, наприклад, зарядні станції для електромобілів чи системи автоматизованого пошуку вільних місць.

Вони грають важливу роль у врегулюванні руху транспорту в містах, зменшенні заторів та поліпшенні якості життя мешканців. Вони також сприяють збереженню громадського простору та раціональному використанню міської інфраструктури.

Впроваджена в інфраструктуру сучасного міста парковка, виконує ряд важливих функцій, сприяючи поліпшенню міського середовища та комфорту для мешканців. Цей елемент інфраструктури вирішує проблеми транспортних заторів і допомагає в збереженні якості повітря в місті. Водіям стає легше знаходити місце для паркування, уникати витрат часу на марні пошуки і запобігати заторам на дорогах [3].

Зменшення кількості автомобілів, що кружляють у пошуках парковки, призводить до зменшення викидів шкідливих речовин у повітря, що відразу позитивно впливає на якість середовища і здоров'я мешканців. Перетворення вільних площ під парковки на зелені зони або інші інфраструктурні проєкти також додає естетичної цінності та покращує життя в місті [3].

Стратегічне розташування парковок поруч з громадським транспортом сприяє зручності мешканців і може підвищити популярність громадського

транспорту, що сприяє зменшенню використання особистих авто. Важливим елементом є також безпека, яку надають організовані парковки, зменшуючи ризик крадіжок та пошкоджень автомобілів [3].

Існує ряд різноманітних видів парковок:

— наземні парковки;

— підземні;

— багаторівневі;

— механізовані;

— автоматизовані;

— парковки для електромобілів.

Розглянемо кожну з них. Наземні парковки представляють собою традиційний тип парковок, що розташовані на поверхні землі. Вони можуть бути відкритими площадками, асфальтованими ділянками або іншими зонами для паркування автомобілів (рисунок 1.1).

Рисунок 1.1 — Наземний паркінг у новобудові

Переваги наземних парковок включають легку доступність, економічність, простоту обслуговування та можливість паркуватися на відкритому просторі. Однак існують і недоліки, такі як велике зайняття

міського простору, забруднення водопостачання, потенційні конфлікти та погіршення естетики, що може впливати на зовнішній вигляд міста. Також, пошук вільних місць може спричиняти затори та витрати часу для водіїв [3].

Підземні парковки — це форма паркування, розташована під поверхнею землі, що відзначається спеціально обладнаними просторами для стоянки автомобілів. Ці парковки можуть бути вбудовані в будівлі чи розміщуватися в підземних спорудах, де вони залишаються невидимими з поверхні [3].

Однією з переваг підземних парковок є їхній естетичний вигляд, оскільки вони не займають значний міський простір і не впливають на обласні ландшафти. Крім того, вони забезпечують ефективне використання доступного простору, що є важливим у густонаселених міських областях (рисунок 1.2).

Рисунок 1.2 — Підземна парковка у торговельному центрі

З іншого боку, будівництво та обслуговування підземних парковок можуть бути витратними та трудомісткими завданнями, порівняно з наземними аналогами. Зокрема, такі парковки можуть вимагати великих інвестицій для створення або реконструкції [3].

Ще однією перевагою є те, що підземні парковки часто обладнані додатковими системами безпеки, що зменшує ризик крадіжок чи

пошкоджень автотранспортних засобів. Крім того, вони можуть надавати зручний доступ до певних об'єктів чи будівель [3].

Однак варто враховувати, що виходження з підземної парковки може вимагати більше часу та зусиль порівняно з виходом із звичайної наземної парковки. Також важливо враховувати екологічні аспекти, зокрема водостічні системи та можливість витоків [3].

Багаторівневі парковки — це спеціалізовані паркувальні об'єкти, які мають кілька рівнів для розташування автомобілів. Ці парковки можуть бути вбудовані у будівлі чи мати власну конструкцію, що зазвичай включає в себе рухомі елементи, які дозволяють пересувати автомобілі між рівнями [3].

Багаторівневі парковки забезпечують ефективне використання обмеженого простору, оскільки дозволяють розміщувати більше автомобілів на одному місці, що особливо важливо в умовах обмеженого міського простору. Це може бути ефективним рішенням для міст з високою густотою населення [3].

Однією з переваг багаторівневих парковок є їхня здатність зменшити транспортні затори та поліпшити рух по дорогах, оскільки водіям не потрібно довго шукати місце для паркування. Вони також можуть бути обладнані автоматизованими системами пошуку вільних місць, що робить процес паркування більш ефективним [3].

Механізовані парковки — це форма паркування, де використовуються різноманітні механізми та автоматизовані системи для розміщення та зберігання автомобілів. Ці парковки можуть використовувати автоматизовані шахти, висувні конструкції або інші технології для оптимізації використання місця та полегшення процесу паркування.

Механізовані парковки дозволяють значно зменшити кількість простору, який потрібен для паркування, оскільки автомобілі розміщуються один поруч з іншим та використовуються механізми для руху. Це особливо важливо в умовах обмеженого міського простору [3].

Рисунок 1.3 — Механізовані багатоярусні паркувальні системи

Перевагою механізованих парковок є їхня ефективність та швидкість процесу паркування. Водії можуть залишати свої автомобілі на спеціальних платформах або вводити їх у автоматизовані конструкції, після чого система самостійно розміщує транспортний засіб. З іншого боку, вартість будівництва та обслуговування механізованих парковок може бути значною, що робить їх менш доступними для впровадження в деяких місцевостях. Також важливо враховувати можливі проблеми, пов'язані з аварійністю та обслуговуванням механізмів [3].

Автоматизована парковка — це інтелектуальна система, яка використовує автоматизовані технології для управління паркуванням та оптимізації використання простору. Головною особливістю є використання різноманітних сенсорів, камер, та програмних алгоритмів для керування рухом автомобілів та ефективного використання парковочного простору [3].

Основна перевага автоматизованих парковок - це здатність ефективно розміщати автомобілі без прямого участі водія. Система може автоматично визначати вільні місця, приймати автомобіль, переміщати його від входу до відповідного місця для паркування та повертати його власнику за його запитом [3].

Такі парковки мають значний потенціал у зменшенні просторового використання, оскільки вони дозволяють ефективно використовувати кожен

квадратний метр парковочного майданчика. Вони також можуть зменшити час, який водії витрачають на пошук парковочного місця, та зменшити викиди шкідливих речовин в повітря.

Однак важливо враховувати вартість впровадження та обслуговування таких систем. Їхня комплексність та залежність від технологічних рішень можуть створювати труднощі в управлінні та експлуатації, що потребує відповідних фінансових та технічних ресурсів. У той час як автоматизовані парковки можуть бути інноваційним рішенням для покращення ефективності паркування, їхній впровадження потребує глибокого аналізу та підготовки для успішної інтеграції у міську інфраструктуру [4].

Ще існують автоматизовані парковки, які керуються веб-додатком. Ці парковки зазвичай включають в себе веб-інтерфейс для керування та моніторингу, що дозволяє користувачам легко контролювати та взаємодіяти з парковкою за допомогою Інтернету. Головною перевагою автоматизованих парковок з веб-керуванням є можливість віддаленого управління. Власники автомобілів чи оператори парковки можуть використовувати веб-додаток на смартфоні чи комп'ютері для знаходження вільних місць, резервування парковочного місця, а також керування процесами введення та виведення транспортних засобів [4].

Також, ці системи часто обладнані додатковими функціями, такими як онлайн-платіжі за паркування, моніторинг стану автомобілів, віддалене відкривання бар'єрів чи воріт, інтеграція з системами електронного заряджання для електромобілів та інші опції, які полегшують використання та підвищують комфорт користувачів.

Парковки для електромобілів — це спеціально облаштовані місця, призначені для паркування електричних автомобілів та забезпечення їхнього заряджання. Це інфраструктурний елемент, який відповідає потребам власників електромобілів та сприяє розвитку та популяризації зеленого транспорту (рисунок 1.4).

Головною ідеєю є наявність зарядних станцій, які дозволяють водіям заряджати свої транспортні засоби протягом перебування на парковці. Це робить використання електромобілів більш зручним та практичним, сприяючи подальшій адаптації цих транспортних засобів в міському середовищі. Окрім того, парковки для електромобілів можуть включати в себе інші зручності, такі як спеціально відведені місця, обладнані зарядними станціями, що дозволяє забезпечити більше автомобілів одночасним заряджанням. Також можливе встановлення додаткових сервісів, наприклад, систем автоматизованого пошуку вільних місць, аплікацій для моніторингу та управління заряджанням, що підвищує зручність використання парковок для власників електромобілів [4].

Рисунок 1.4 — Стоянки для електромобілів

1.2 Сучасні технології для підтримки функціонування автопаркінгу

Із зростанням транспортних потреб у сучасному світі стає очевидною необхідність вдосконалення систем управління паркуванням. Розподілена система, що спрямована на підтримку функціонування автопаркінгів, відіграє ключову роль у цьому процесі, впроваджуючи інноваційні підходи для покращення користувацького досвіду та оптимізації управління паркінговим простором.

Технологічний прорив у цій області дозволяє автоматизувати та спростити пошук паркувальних місць, надаючи водіям оперативний доступ

до інформації про вільні місця за допомогою розподілених систем. Важливо відзначити, що розподілена система для підтримки функціонування автопаркінгів є основою для розвитку майбутніх технологій у цьому напрямку. При збільшенні обсягів автотранспорту важливо мати ефективні системи, що забезпечують комфорт та безпеку для користувачів. Цей інноваційний підхід до управління паркуванням відкриває нові можливості для міст і комерційних об'єктів, пропонуючи вдосконалені та передові рішення для ефективного використання простору паркування та покращення мобільності в містах [4, 5].

Сучасні технології для підтримки функціонування автопаркінгу включають в себе різноманітні інновації, які спрощують управління паркінговим простором, забезпечують більшу ефективність та зручність для водіїв. Розглянемо аспекти таких технологій.

Чудовим варіантом може бути мобільний додаток для управління паркуванням, такі додатки дозволяють водіям легко знаходити вільні парковочні місця, здійснювати онлайн-оплату за паркування, отримувати повідомлення про час завершення паркування, а також інші сервіси, які роблять процес паркування більш зручним.

Сучасні технології дозволяють проводити моніторинг зайнятості місць для паркування в автоматичному режимі, надають можливість користувачам віддалено отримувати інформацію про кількість наявних вільних місць. Для цього кожне місце для паркування обладнується датчиком присутності, який визначає відсутність або наявність на ньому транспортного засобу. Інформація з датчиків збирається концентратором, який передає її на виділений або хмарний сервер, де вона стає доступною для онлайн перегляду за допомогою мобільного застосунку. В розглядуваній системі передбачається не лише можливість перегляду наявних вільних місць, а резервування місця для паркування, здійснення оплати за користування парковкою, можливість створення віртуальної картки, яку водії можуть поповнювати та з неї будуть списуватись кошти.

Для швидкого пошуку місця паркування поряд з місцем знаходження водія підтримується геолокація. Крім того, паркування може бути обмежено часом доби, днем тижня чи іншими факторами. Такі стоянки будуть вважатися недіючими і світитися сірим кольором. В зазначений період вони будуть не доступні до вибору.

Контроль за безпекою здійснюватиметься за допомогою камер відеоспостереження, встановлених по всьому периметру автостоянки. Це надаватиме можливість водіям контролювати через додаток безпеку автомобіля, переглядаючи відео з камер в реальному часі.

Серед додаткових систем можуть бути системи розпізнавання номерних знаків, які використовуватимуть камери для розпізнавання номерів автомобілів. Вони можуть бути використані для вхідного та вихідного контролю, автоматичної оплати паркування або виявлення порушень. Також сюди може підійти система Інтернет речей — сенсори та датчики можуть бути розміщені в різних частинах паркінгу для моніторингу використання місць, визначення ступеня зайнятості та забезпечення безпеки. Ці дані можуть бути використані для оптимізації розташування паркомісць і покращення управління простором.

Слід також врахувати аналітику даних, що дозволяє ефективніше управляти паркінговими ресурсами. Моделі машинного навчання можуть аналізувати дані про використання та прогнозувати потреби, а також виявляти аномалії та оптимізувати потоки руху.

Ці технології спрямовані на покращення доступності паркомісць, зменшення транспортного затору, покращення безпеки та забезпечення зручності для водіїв.

1.3 Аналіз сучасних електронних систем для автопаркінгу

Одним з аналогів розглядуваної системи є Parkopedia — це онлайн-платформа та мобільний додаток, які надають інформацію про парковки у різних частинах світу. Ця платформа допомагає водіям знаходити

інформацію про вільні парковочні місця, їх ціни, години роботи, та інші важливі дані, які можуть бути корисними під час пошуку та вибору парковки [5].

Користувачі можуть використовувати додаток або веб-сайт Parkopedia, щоб знайти доступні парковочні місця в містах, аеропортах, торгових центрах та інших місцях. Платформа надає докладну інформацію про кожну парковку, включаючи її місцезнаходження, вартість паркування, час роботи, доступність для інвалідів, відгуки користувачів та інші характеристики (рисунок 1.5).

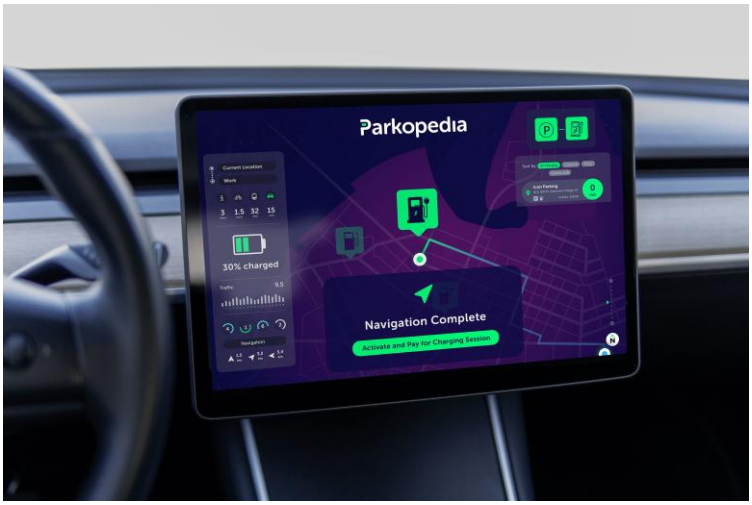

Рисунок 1.5 — приклад системи Parkopedia

У деяких місцях користувачі можуть бронювати парковочні місця через додаток. Вона може інтегруватися з різними навігаційними додатками, допомагаючи водіям легко знайти шлях до обраної парковки. Користувачі можуть залишати відгуки та рейтинги для парковок, ділившись своїм досвідом з іншими. Деякі версії Parkopedia можуть надавати інформацію про велосипедні стоянки та станції міського транспорту [5].

Parkopedia допомагає водіям зекономити час і гроші, шукаючи оптимальні парковочні місця та отримуючи актуальну інформацію про парковки. Платформа використовує дані з різних джерел, включаючи

операторів парковок, місцеві органи влади та користувачів, щоб забезпечити точну та корисну інформацію [5].

Онлайн-платформа охоплює парковки у більш ніж 75 країнах та понад 15 000 містах, що робить її однією з найбільших баз даних про парковки у світі. Сервіс надає інформацію про різні типи парковок, включаючи вуличні парковки, парковки в аеропортах, великі парковочні комплекси у торгових центрах, готелях, а також місця для велосипедів та електрокарів [5].

У деяких випадках, за допомогою Parkопедії, можна бронювати парковочні місця наперед. Це особливо корисно для користувачів, які планують подорожі або відвідують популярні місця.

Другим аналогом є "Smart Parking Barrier System" (система розумних парковочних бар'єрів) — це технологічне рішення, яке використовується на парковках для управління доступом, вказівки на наявність або відсутність вільних парковочних місць і оптимізації використання парковочного простору. Ця система може бути частиною більшого "розумного паркування" і інтернету речей інфраструктури, яка дозволяє віддалено керувати та моніторити паркову [6].

Основні компоненти і функції системи розумних парковочних бар'єрів можуть включати таке:

— система має автоматизовані бар'єри або шлагбауми, які можуть відкриватися або закриватися автоматично на основі інформації про доступність парковочних місць;

— датчики розташовані на парковці і вимірюють, чи є вільні парковочні місця, це може бути зроблено за допомогою камер, ультразвукових сенсорів або інших технологій;

— система може включати камери відеоспостереження, які реєструють події на парковці та надають зображення в режимі реального часу;

— інформація про доступність парковочних місць може відображатися на моніторах або світлових індикаторах, які вказують водіям, куди їм слід їхати для паркування;

— всі компоненти системи керуються спеціальним програмним забезпеченням, яке аналізує дані від сенсорів та приймає рішення щодо відкриття або закриття бар'єрів;

— деякі системи розумних парковочних бар'єрів можуть бути інтегровані з мобільними додатками, що дозволяють водіям заздалегідь бронювати парковочні місця або отримувати інформацію через смартфони;

— системи можуть збирати дані про використання парковки, що може бути корисним для аналітики та планування.

Система розумних парковочних бар'єрів спрямована на оптимізацію управління парковкою, покращення безпеки та зручності для водіїв, а також зменшення заторів на парковці. Вона може бути використана на комерційних, громадських та інших типах парковок, приклад системи наведено на рисунку 1.6.

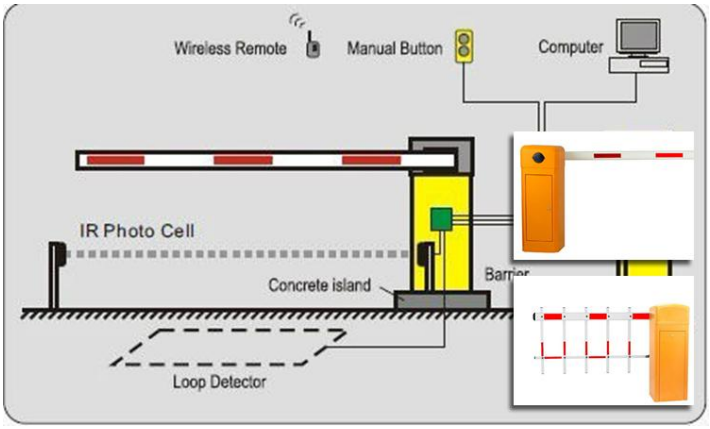

Рисунок 1.6 — Smart Parking Barrier System

Розглянемо третій аналог автоматизованої парковки. Parquery є провідним швейцарським постачальником технологій для розумного паркування та рішень для мобільності, транспорту та логістики на основі ШІ. Понад 200 торгових посередників, партнерів і клієнтів у понад 35 країнах по

всьому світу розширили свій бізнес, підвищили ефективність і здобули час, гроші та простір за допомогою нашого штучного інтелекту (ШІ) [6].

Розробники почали з рішень для розумного паркування на основі камери, використовуючи та розвиваючи наш видатний штучний інтелект. Штучний інтелект виявляє та розпізнає будь-яку форму на зображеннях і відео з будь-якої камери — камери відеоспостереження, веб-камери чи дрони і перетворює це на дані, які можна використовувати для управління, прийняття рішень, контролю, бізнес-аналітики, що завгодно. Інтелектуальні рішення Parquery навчаються на прикладі та досвіді, тому можуть адаптуватися до більшості обставин [6].

Універсальне рішення для паркування працює для всіх видів паркування:

— великі прибудинкові ділянки;

— неасфальтовані, нерозмічені та навіть засніжені паркувальні майданчики;

— паркування на вулиці, як паралельне, так і діагональне;

— бордюрні зони доставки;

— криті гаражі;

— місця відпочинку вантажівок вздовж автомагістралей;

— електронне заряджання.

Він працює для всіх транспортних засобів, включаючи легкові автомобілі, вантажівки, мікроавтобуси, причепи, автобуси, транспортні засоби доставки, мотоцикли тощо.

Розгорніть програмне забезпечення там, де воно вам потрібно: у хмарі, на локальному сервері, вставте його безпосередньо в камеру чи зовнішній комп’ютер або покладіться на хмарні сервери Parquery [6].

Parquery — це технологічна компанія, яка спеціалізується на розробці систем візуального моніторингу парковки на основі штучного інтелекту та комп'ютерного зору. Головна мета Parquery — надавати інформацію про доступність парковочних місць у режимі реального часу, щоб полегшити

пошук парковки та оптимізувати використання парковочних просторів в містах та на інших громадських територіях [7].

Рішення для розумного паркування надає результати та аналітику в реальному часі партнерам і клієнтам із розумного паркування.

Інтерфейс прикладного програмування (API) інтегрує наші результати в реальному часі в будь-яке власне програмне забезпечення, панелі вказівок, програми тощо. У той же час інформаційна панель надає менеджерам інформацію про використання паркувальних місць (рисунок 1.7).

Рисунок 1.7 — Parquery

Ця технологія може бути корисною для місцевих органів влади, власників парковок і водіїв, оскільки вона допомагає оптимізувати використання парковочних місць, зменшує час пошуку парковки та сприяє раціональному використанню парковочної інфраструктури.

Третім аналогом є Klaus Multiparking — німецька компанія, яка спеціалізується на розробці та виробництві систем автоматизованих парковок. Компанія відома своїми інноваційними рішеннями для механізованих багатоповерхових паркінгів та систем управління парковкою. Klaus Multiparking заснована у 1907 році і вже більше століття спеціалізується на розвитку та виробництві парковочних систем [7].

Основні характеристики систем Klaus Multiparking включають наступне: механізовані багатоповерхові парковки, системи управління, інтеграція з IoT, роботизовані системи, індивідуальні рішення (рисунок 1.8).

Klaus Multiparking є одним із провідних гравців у світі на ринку автоматизованих парковочних систем і відома як виробник надійних та ефективних рішень для паркування автомобілів. їхні продукти допомагають покращити використання парковочного простору та забезпечують більше комфорту для водіїв і власників парковок [7].

Рисунок 1.8 — Klaus Multiparking

2 ВИБІР ТА ОБГРУНТУВАННЯ ПІДХОДІВ ДО ОРГАНІЗАЦІЇ РОЗПОДІЛЕНОЇ СИСТЕМИ

2.1 Визначення архітектури розподіленої системи

Як було зазначено у першому розділі система з підтримки функціонування автопаркінгу забезпечує надання послуг з паркування на закритій території та як правило містить такі основні компоненти:

— датчики виявлення наявності автомобілів, вони встановлюються на місцях паркування і вказують системі, чи зайняте конкретне місце чи ні; можуть використовуватися різні типи датчиків, такі як ультразвукові, інфрачервоні, радіочастотні або візуальні датчики;

— камери відео спостереження, що використовуються для відстеження руху автомобілів та визначення доступних місць для паркування;

— автоматизовані шлагбауми, що на основі різних сигналів, таких як сигнали від датчиків виявлення автомобілів автоматично відкриваються та закриваються;

— датчики безпеки для шлагбаумів, які виявляють перешкоди на шляху шлагбаума та автоматично зупиняють його рух, щоб уникнути аварій;

— автоматизовані термінали, де водії можуть оплачувати паркування готівкою, монетами або кредитними/дебетовими картками;

— безконтактні платіжні системи (такі як Apple Pay, Google Pay) для швидкої і безпечної оплати паркування.

Взаємодія з користувачем здійснюється через мобільний додаток, що дозволяє водіям знаходити вільні парковочні місця, резервувати їх та оплачувати послуги з паркування.

Для того щоб зберігати та оброблювати дані про транзакції оплати, а також дані про авторизацію користувачів, необхідний сервер. Він буде отримувати дані від різних датчиків (наприклад, датчики наявності автомобілів, камери відеоспостереження, датчики безпеки) та керувати

роботою обладнання, такого як шлагбауми, світлофори, паркомати та інші системи. Сервер виконує процеси авторизації користувачів, включаючи перевірку карт доступу, ідентифікацію через мобільні додатки, забезпечує інформацією для користувачів, такою як вільні парковочні місця, розташування паркоматів, тощо. Він відповідає за забезпечення безпеки всієї системи, включаючи захист від несанкціонованого доступу, відновлення після збоїв, та захист від злому чи кібератак.

Існує декілька варіантів підходів до організації серверів. У першому усі парковки обслуговуються одним центральним сервером. Він може розташовуватися в центральному офісі чи дата-центрі. Цей підхід може бути зручним для централізованого керування та моніторингу всіх парковок.

У другому кожна парковка може мати свій власний сервер, який відповідає за управління та моніторинг конкретної області паркування. Це може бути корисним в разі, якщо парковки розташовані на великій відстані одна від одної або якщо вони фізично відокремлені.

У третьому використання комбінації централізованих та децентралізованих серверів. Наприклад, декілька парковок, які знаходяться в одному місті, можуть використовувати спільний централізований сервер, але при цьому кожна парковка може мати свій власний сервер для локального керування та моніторингу.

Виділений сервер (он-преміс сервер або dedicated server) — це фізичний сервер, який повністю призначений для використання однією конкретною організацією або користувачем. Це означає, що весь обсяг обчислювальних ресурсів, обладнання та мережеві з'єднання призначені лише для користувача або компанії, яка володіє або орендує цей сервер.

Основні характеристики виділеного сервера включають [8]:

— виділений сервер представляє собою фізичний сервер, розташований в дата-центрі або в спеціально обладнаному приміщенні; користувач має повний контроль над апаратним забезпеченням.

— усі ресурси, такі як процесор, оперативна пам'ять, дисковий

простір та мережеві з'єднання, призначені тільки для використання конкретним користувачем або організацією;

— користувач має адміністративний доступ і може встановлювати будь-яке програмне забезпечення, налаштовувати операційну систему та керувати всією конфігурацією сервера;

— оскільки ресурси не діляться з іншими користувачами, виділені сервери можуть забезпечити вищий рівень безпеки та конфіденційності;

— користувач може прогнозувати та контролювати продуктивність сервера, оскільки ресурси не діляться з іншими користувачами.

Хмарний сервер — це віртуальний сервер, який функціонує в хмарному обчислювальному середовищі. Хмарні сервери не розташовані фізично на місці користувача; вони базуються на інфраструктурі хмарних обчислень, яка може включати в себе деяку кількість серверів, з'єднаних мережею та управляються провайдером хмарних послуг. Хмарні сервери використовуються для різних завдань, включаючи зберігання даних, обчислення, розгортання веб-сайтів та інші варіанти віртуалізованого обчислення. Популярні провайдери хмарних послуг включають Amazon Web Services (AWS), Microsoft Azure, Google Cloud Platform та інші.

Основні характеристики хмарних серверів включають [9]:

— хмарні сервери використовують віртуалізацію для створення віртуальних екземплярів серверів на фізичних машинах; кожен віртуальний сервер має своє власне віртуальне обладнання та операційну систему;

— ресурси хмарних серверів можуть бути спільно використані між кількома користувачами або клієнтами; це робить їх більш гнучкими і ефективними з точки зору використання ресурсів;

— хмарні сервери доступні через Інтернет, що дозволяє користувачам отримати доступ до своїх ресурсів з будь-якого місця та пристрою з підключенням до мережі;

— хмарні сервери легко масштабуються вгору або вниз залежно від потреб користувача; вони можуть адаптуватися до змінного навантаження та

обсягу даних;

— багато хмарних сервісів пропонують модель оплати за використання, коли користувачі платять тільки за ті ресурси, які вони фактично використовують.

Порівняльні характеристики Виділеного та Хмарного серверів наведені у таблиці 2.1.

Таблиця 2.1 — Порівняльні характеристики Виділеного та Хмарного серверів

Характеристика	Виділений Сервер	Хмарний Сервер
Контроль та конфіденційність	Високий рівень контролю та конфіденційності, оскільки сервер призначений тільки для одного користувача або компанії.	Менший рівень фізичного контролю та конфіденційності, оскільки ресурси можуть бути спільно використані.
Масштабованість та гнучкість	Обмежена масштабованість, важко масштабувати ресурси вгору або вниз залежно від потреб.	Висока гнучкість та масштабованість, можливість легко змінювати обсяг ресурсів відповідно до потреб.
Модель оплати	Зазвичай фіксовані витрати, незалежно від фактичного використання ресурсів.	Можливість оплати за фактичне використання ресурсів (платіть те, що використовуєте).
Безпека та доступність	Високий рівень безпеки, оскільки ресурси ізольовані від інших користувачів.	Забезпечується високий рівень безпеки, але менше фізичного контролю. Висока доступність через резервування ресурсів.

Продовження таблиці 2.1

Управління та адміністрування	Користувач має повний адміністративний доступ та може налаштовувати сервер за своїми потребами.	Адміністрування та налаштування ресурсів здійснюється через інтерфейс хмарного провайдера, менший рівень контролю.
Прогнозована продуктивність	Дозволяє прогнозувати та контролювати продуктивність, оскільки ресурси призначені лише для одного користувача.	Продуктивність може бути варіабельною через спільне використання ресурсів з іншими користувачами.
Вартість володіння	Може бути витратним через необхідність утримання фізичного обладнання та інфраструктури.	Зазвичай економічний, оскільки не потрібно власного обладнання, оплата лише за використані ресурси.

Виходячи з розгляду переваг та недоліків серверів доцільніше використовувати технологію хмарних серверів.

Умовно усі види хмарних послуг можна поділити на три типи [9]:

— Infrastructure as a Service (інфраструктура як послуга);

— Platform as a Service (платформа як послуга);

— Software as a Service (програмне забезпечення як послуга).

Приставка as a Service, яка є усюди, означає, що всі види хмар надаються за моделлю підписки, тобто ви використовуєте їх тільки тоді, коли в них є необхідність. Суть хмарних послуг концепції Pizza-as-a-Service пояснює рис. 2.1.

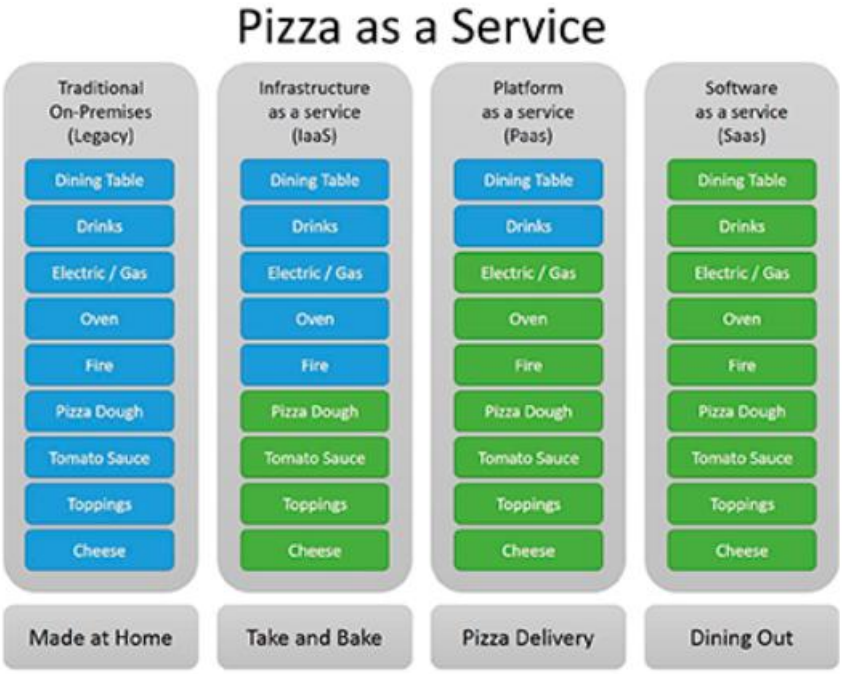

Рисунок 2.1 — Концепція Pizza-as-a-Service

Розглянемо категорії хмарних технологій, починаючи з Software as a Service (SaaS), що є однією з найвідоміших. Цей тип технологій надає готові рішення клієнтам з мінімальною потребою в налаштуванні. Підписуючись на SaaS, будь-який користувач може ефективно управляти сервісом, майже не залучаючи системного адміністратора. Прикладами корпоративних SaaS-сервісів є, наприклад, Office 365, а для малих та середніх підприємств - Dropbox, Evernote, Trello та інші.

Платформа as a Service (PaaS) призначена, насамперед, для розробників. Це набори готових компонентів для створення додатків та фреймворки для управління платформою. Компонентами можуть бути, наприклад, сервіси даних, репозиторії, інструменти автоматизованого деплою, середовища тестування та інші. Приклади PaaS-сервісів включають Google AppEngine, VMWare Pivotal Cloud Foundry, Red Hat's OpenShift, Heroku і т.д.

Інфраструктура as a Service (IaaS) — це область, яка найбільше нагадує володіння власним обладнанням та віртуалізацію. В цьому випадку IaaS надає хмарні ресурси, такі як процесори, пам'ять, диски та мережі. З цих

компонентів користувач створює сервери-маршрутизатори та конфігурує мережеву топологію згідно зі своїми потребами.

На даний момент у світі правлять три гіганти — AWS, Azure, Google Cloud. Ці компанії займають левову частку ринку по всьому світові (крім Китаю, там ще є Alibaba Cloud), є технологічними лідерами та задають тренди в розвитку хмарних IaaS сервісів. Наприклад, на момент написання статті AWS мав у своєму портфоліо більше 100 сервісів (IaaS, SaaS, PaaS). Однак, незважаючи на всю свою міць, повної монополізації ринку вони поки що не досягли і в найближчому майбутньому малоймовірно, що досягнуть.

Розглядувана розподілена система є комплексом апаратно-програмних засобів, що будуть взаємодіяти з хмарним сервером, а отже принципи її функціональної організації відповідають архітектурі Інтернету речей (IoT). Мережа Інтернету речей є системою різноманітних пристроїв, які не мають прямого зв'язку між собою і існують для збору та передачі різноманітних даних. Ці пристрої не втручаються в роботу один одного. Ключовою умовою їхнього успішного функціонування є використання єдиного протоколу передачі даних. Серед найвідоміших світових протоколів IoT можна виділити Bluetooth, Wifi, Zigbee, MQTT, NFC, DDS, LTE та LoRaWAN.

Одним з найпопулярніших стандартів стає стандарт LoRaWAN — стандарт передачі даних в мережах широкого радіусу дії з низьким рівнем енергоспоживання пристроїв (Low Power Wide Area Network, LPWAN). Цей стандарт отримав популярність завдяки міжнародній некомерційній організації LoRa Alliance, головними учасниками якої є компанії Semtech (NASDAQ: SMTC), IBM і Cisco. Вони відіграють ключову роль в розвитку протоколу LoRaWAN з самого початку [10].

Важливо відзначити, що стандарт LoRaWAN є відкритим. Його відкритість дозволяє існування конкурентного ринку постачальників обладнання, а також сумісність пристроїв від різних виробників (можна підключити базові станції декількох брендів до одного сервера і включити пристрої відразу декількох виробників в одну мережу). Отримана відкритість

також гарантує споживачеві, що розробник чи постачальник не може однобічно змінювати умови експлуатації, використовуючи відсутність альтернатив, пов'язаних із пропрієтарністю їхніх рішень (рисунок 2.2).

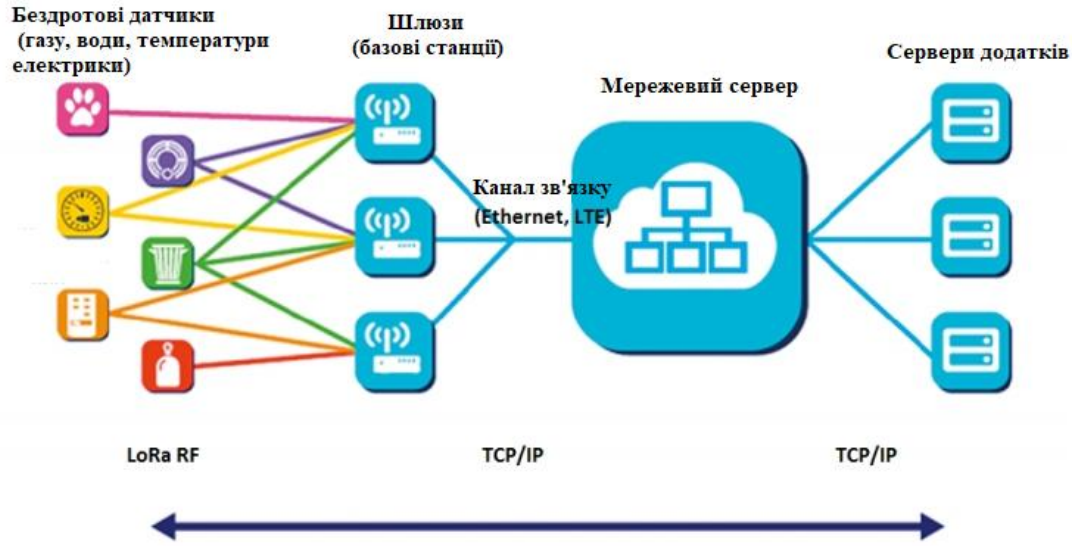

Рисунок 2.2 — Структурна схема страндарту LoRaWAN

Стандарт зв’язку LoRaWAN (анг. Long Range Wide Area Network, тобто «глобальна мережа далекого радіусу дії») забезпечує мережеву передачу даних на великі відстані. Він призначений для одночасного обслуговування великої кількості малопотужних промислових абонентських пристроїв [11].

Широковідомі формати бездротового зв’язку GSM / 3G / LTE / WiFi орієнтовані на «живого користувача» і використовують ліцензований спектр радіочастот для надання безперервної передачі даних на високій швидкості та значній ємності мережі на відносно невеликі відстані. LoRaWAN же орієнтований на «користувачів-машин» — пристрої і датчики, які працюють в неліцензованому спектрі (отже, розгортання мережі не вимагає узгодження з чиновниками) і не потребують великого за ємністю каналу для передачі даних [12].

Сигнал мережі стандарту LoRaWAN, завдяки дальності дії (до 10 кілометрів на відкритій місцевості і від 1 до 2 кілометрів від базової станції в умовах міської забудови) і можливості проникати у важкодоступні місця

(колодязі, підвали і т.д.) ідеально підходить для малопотужних датчиків і пристроїв. Швидкість і дальність передачі даних можна коригувати за рахунок індивідуальної настройки коефіцієнта розширення спектра (Spreading Factor). Це дозволяє досягти оптимального балансу між швидкістю і дальністю передачі даних і необхідного рівня стікойсті до радіошуму [12].

Як згадувалося вище, LoRaWAN використовує неліцензовану частину радіочастотного спектру в діапазоні від 868,0 МГц до 868,6 МГц (в Україні) та енергозберігаючі технології. Стандарт передбачає наявність базових станцій і абонентських пристроїв, які, за умови автономного живлення, більшу частину часу перебувають в режимі збереження енергії. LoRaWAN датчики «прокидаються» лише для обміну даними з сервером. Це дозволяє виробникам гарантувати експлуатацію IoT-пристрою протягом 5 і навіть 10 років без необхідності заміни батарей [12].

Відповідно до розглянутого вище приходимо до архітектури розподіленої системи з підтримки функціонування автопаркінгу, що наведена у додатку Б. Система реалізована на основі мережі LoRaWAN.

Основним завданням системи є моніторинг зайнятості паркомісць та надання підтримка можливості отримувати інформацію про кількість вільних місць для паркування онлайн у режимі реального часу. Моніторинг кількості доступних для паркування місць реалізується за допомогою датчиків присутності автомобіля, яким обладнується кожне місце для паркування. Датчик забезпечує можливість автоматично отримувати інформацію про відсутність чи наявність на місці для паркування транспортного засобу, тобто визначати статус місця: «вільне» або «зайняте».

Усі датчикі розділені на групи, кожна з яких відповідає певній ділянці паркувального майданчика. Інформація з датчиків про зміну статусу місця для паркування передається у шлюзи, кожний з яких зв’язаний зі своєю групою датчиків.

Шлюз є вузлом мережі LoRaWAN, що забезпечує передачу даних до концентратора. Концентратор є центральним комунікаційним вузлом, який

збирає інформацію та передає її до програми хмарної служби паркування, яка може знаходитися на хмарному або виділеному сервері. Зазначена програма використовує інформацію з датчиків для відстеження вільних та зайнятих місць, відображення подій паркування на веб-сторінці, надання за допомогою мобільних додатків даних про доступні паркувальні місця водіям. Крім того, це дозволяє впроваджувати додаткові сервіси, наприклад енергозберігаюче керування освітленням, забезпечення охорони, пожежної безпеки тощо.

Хмарна служба паркування може обмінюватися даними про паркування у режимі реального часу з іншими службами розумного міста, що знаходяться у віданні муніципальної або районної влади. З використанням інформації, зібраної з багатьох частин міської інфраструктури, можуть бути запропоновані різноманітні програми.

2.2 Обгрунтування алгоритму з надання послуг з паркування

Варіанти паркування можуть бути класифіковані на основі різних критеріїв, таких як бронювання, тип території, доступність послуг та інші. Головним призначенням розглядуваної розподіленої системи є надання можливості водіям отримувати інформацію про вільні місця для паркування в режимі онлайн для зменшення автомобільного трафіку на вулицях міста. При цьому необхідно визначити, яким чином буде задовольнятися попит на ці місця. Враховуючи те, що будь який з водіїв, що перебуває поруч з парковкою, при наявності вільного місця може скористатися ним для паркування у режимі «живої черги», розглянемо можливі підходи розподілу вільних місць між тими, кому потрібний певний час на те, щоб дістатися до парковки.

Перший варіантом є підтримка можливості онлайн бронювання місця для паркування. Бронювання дозволяє водіям заздалегідь зарезервувати наявне вільне парковочне місце. Гарантією того, що зарезервоване місце буде точно використане і компанія не втратить прибуток є попереднє стягнення плати при бронюванні. Такий підхід є зручним для користувачів, оскільки

дозволяє заздалегідь спланувати поїздку, гарантуючи відсутність проблем, пов’язаних з пошуком місяця, на якому можна залишити автомобіль. З іншого боку заброньоване місце буде залишатися не використаним протягом усього часу від моменту бронювання до моменту заїзду автомобіля. Збільшення кількості заброньованих місць буде збільшувати кількість тих водіїв, які змушені дарма пересуватися вулицями міста у пошуку місця для паркування. Цю проблему можна частково вирішити наданням можливості скористатися зарезервованим місцем іншому водієві до настання часу, що визначений при бронюванні. Проте при цьому можуть виникнути проблеми, пов’язані з несвоєчасним звільненням забронюваного місця, що тимчасово використовується іншим водієм.

Повністю протилежним підходом є розподіл вільних місць тільки у режимі «живої черги». Водій може лише отримати інформацію про кількість наявних на даний момент вільних місць, а тому має можливість припарковати свій автомобіль лише тоді, коли на момент прибуття вільні місця виявляться не зайнятими. Перевагою такого варіанту є те, що при наявності вільного місця водій може залишити свій автомобіль одразу та на будь який час. При цьому оплата за користування послугою з паркування буде здійснюватися за фактичним часом перебування автомобіля на парковці. Місце, що звільнилося, одразу може бути використане іншим водієм. Проте в умовах щільного трафіку кількість бажаючих скористатися послугою з паркування може перевищувати кількість наявних вільних місць, тому при такому варіанті скористатися ними зможуть лише ті, хто прибуде раніше. Інші проїдуть до парковки дарма, що не сприяє зменшенню трафіку.

Можливий ще й третій варіант, який можна вважати проміжним між двома попередніми. При такому варіанті система не підтримує можливість бронювання, але надає інформацію не лише про кількість наявних на даний момент вільних місць, а й про орієнтовний час, протягом якого ці місця будуть залишатися вільними. Такий прогноз система буде робити на основі статичних даних, що збираються самою системою та характеризують

динаміку заповнення/звільнення місць для паркування у залежності від часу доби. У результаті водій може самостійно оцінити шанси припарковати автомобіль у зазначеному місці з врахуванням часу, який потрібен йому, щоб дістатися до парковки.

За порівнянням з попередніми варіантами останній підхід надає можливість більш ефективно задовольняти попит на послуги з паркування, тому саме його будемо використовувати у розглядувані розподіленій системі. Крім того, реалізація зазначеного підходу дозволяє розробляти різні стратегії оптимального використання парковочних ресурсів та підвищення задоволення від користування парковочними послугами. Врахування статистичних даних дозволяє прогнозувати та реагувати на зміни у попиті, забезпечуючи ефективне управління парковкою в режимі реального часу.

Аналіз даних щодо користування паркуванням враховує не лише кількісні аспекти, але й позволяє розглядати різні сценарії оптимізації, враховуючи індивідуальні потреби користувачів та особливості місцевого транспортного потоку.

Загалом, інтеграція статистичних даних у систему управління паркуванням є перспективною та ефективною стратегією для вдосконалення міського транспортного обслуговування, що сприятиме зручності користувачів та покращенню ефективності управління парковочними ресурсами.

Розглянемо основні статистичні показники, за якими можна оцінити та спрогнозувати попит на користування послугами з паркування.

Імовірність наявності вільного місця визначається формулою:

$$P_в = \frac{M}{N}$$

де M — кількість наявних вільних парковочних місць;

N — загальна кількість місць.

Відповідно імовірність зайнятості усіх місць можна знайти як

$$Р_з = 1 - Р_в$$

Середній час паркування можна визначити як

$$Т_{пар.сер} = \frac{1}{N} \sum_{i=1}^{N} Ti$$

де $Т_i$ — час паркування для і-го автомобіля;

N — загальна кількість автомобілів.

Середній інтервал між прибуттями автомобілів на парковку:

$$Т_{пр.сер} = \frac{1}{N-1} \sum_{i=2}^{N} Tпрі$$

де $Т_{прі}$ — інтервал між прибуттями автомобілів;

N — загальна кількість прибуттів.

Величина зворотна до середнього часу між прибуттям автомобілів характеризує інтенсивність потоку автомобілів на в’їзді на парковку:

$$I = 1/ Т_{пр.сер}$$

Аналіз цих параметрів надає можливість виробити стратегії для оптимізації розподілу парковочних ресурсів та покращення використання інфраструктури. Детальний розгляд даних про вільні та зайняті місця протягом доби може бути представлений середнім попитом на паркувальні місця, коефіцієнтом зайнятості парковки та інтегральною імовірністю наявності вільного місця протягом години:

Середній попит на парковочні місця, що виникає протягом доби може бути визначений як (ADP - Average Demand for Parking):

$$\mu_{\text{пм}} = \frac{1}{24} \sum_{t=1}^{24} Dt$$

де D_t — кількість автомобілів, що прибувають протягом години t.

Коефіцієнт зайнятості парковки (OCP — Occupancy Rate):

$$\text{КЗ} = \frac{\text{N}-\text{M}}{\text{N}}$$

де N — загальна кількість місць;

M — кількість вільних місць.

Фактично коефіцієнт зайнятості парковки характеризує імовірність того, що вільних місць немає.

Оскільки тривалість паркування є випадковою величиною, то ймовірність того, що парковочне місце буде зайняте протягом часу t, визначається законом розподілу, наприклад, експоненціальним або нормальним.

Якщо припустити, що тривалість паркування є випадковою величиною, що розподілена за експоненціальним законом, то ймовірність того, що парковочне місце буде зайняте протягом часу t, може бути виражена як:

$$\text{P(T <= t) = 1- } e^{-\delta t}$$

де T — тривалість паркування;

l — параметр інтенсивності експоненціального розподілу.

Іншим підходом є використання нормального розподілу для моделювання тривалості паркування. У цьому випадку ймовірність того, що тривалість паркування буде в певному інтервалі \([a, b]\), може бути визначена як:

$$\text{P(a <= T <= b)} = \int_a^b \frac{1}{\sqrt{2\pi\sigma^2}}\ e^{-\frac{(t-\mu)^2}{2\sigma^2}} \text{dt}$$

де T— тривалість паркування;

u — середнє значення;

o — стандартне відхилення нормального розподілу.

Використання таких законів розподілу надає можливість моделювати різноманітні сценарії тривалості паркування та передбачити ймовірності виникнення конкретних ситуацій на парковці. Наприклад, можна аналізувати ймовірність того, що автомобіль залишить парковочне місце протягом певного інтервалу часу, що важливо для розробки ефективних стратегій управління паркуванням.

2.3 Вибір технологій розробки

Було вирішено реалізувати розробку програмного забезпечення у формі веб-додатку, який забезпечить можливість користувачам отримувати доступ до нього через мережу Інтернет.

Веб-додаток є програмним забезпеченням, яке розробляється для використання у веб-середовищі. Однією з основних відмінностей веб-додатків є те, що вони доступні для користувачів через веб-браузер, без необхідності встановлення на їхні пристрої. Розробка веб-додатків часто базується на веб-технологіях, таких як HTML, CSS та JavaScript, і вони можуть використовувати серверні технології для взаємодії з базами даних та обробки запитів користувачів.

Важливою перевагою веб-додатків є їхня доступність через Інтернет, що дозволяє користувачам взаємодіяти з програмою з будь-якого місця, де є Інтернет-з'єднання. Це полегшує роботу з додатком і виключає необхідність завантаження та встановлення на кожен пристрій окремо. Крім того, веб-додатки можуть бути оновлені централізовано, що дозволяє швидко впроваджувати нові функції та виправляти помилки без необхідності оновлення кожного пристрою окремо.

Веб-додаток для автоматизованої парковки призначений для спрощення життя користувачів і оптимізації управління парковкою. Важливою перевагою веб-додатку є можливість моніторингу та аналітики використання місць, що надає власникам парковки інформацію для управління ресурсами. Також, система оповіщень надсилає користувачам повідомлення про стан їхніх бронь, що робить процес користування парковкою більш комфортним та підтримує зв'язок з клієнтами.

2.3.1 Аналіз технологій для розробки клієнтської частини

Розглянемо клієнтську частину додатку, яку буде бачити користувач. У ній будемо використовувати HTML, CSS, JS та фреймворк Bootstrap.

Клієнтська частина веб-додатку реалізується за допомогою HTML, що відповідає за структуру та маркування веб-сторінок. HTML визначає елементи, такі як заголовки, абзаци, таблиці, форми і багато інших, які визначають, як контент повинен бути відображений на веб-сторінці. Це базова складова, яка дозволяє визначати структуру та ієрархію елементів на сторінці.

CSS використовується для визначення вигляду та стилю веб-сторінок, які створені з використанням HTML. Він включає в себе правила, які визначають кольори, шрифти, розміри, відступи та інші атрибути, щоб створити привабливий та зручний для взаємодії дизайн. CSS дозволяє розділити логіку структури та дизайну, щоб полегшити роботу розробникам та підтримку коду.

JS відповідає за динамічність та інтерактивність веб-сторінок. Він вбудовується у HTML-документ та використовується для взаємодії з користувачем, маніпулювання DOM (Document Object Model), анімації та виконання асинхронних операцій. JavaScript може бути використаний для валідації введених даних, реагування на події користувача та обробки даних без необхідності перезавантаження сторінки.

HTML включає різноманітні теги, такі як `<head>`, `<body>`, `<div>`, `<form>`, які визначають різні частини веб-сторінки. Теги `<head>` містять мета-інформацію, таку як заголовок сторінки і підключення зовнішніх ресурсів, тоді як теги `<body>` містять основний вміст сторінки, такий як текст, зображення та форми.

CSS визначає стилі, які надають веб-додатку вигляд і відчуття. Відбувається це за допомогою визначення класів, ідентифікаторів та інших селекторів, які стилюють конкретні елементи HTML. CSS може також бути використаний для роботи з різними типами пристроїв, застосовуючи принципи адаптивного дизайну.

JavaScript забезпечує можливість створювати відгук на події, взаємодіяти з сервером за допомогою AJAX-запитів та динамічно оновлювати вміст сторінок. Це дозволяє розробникам створювати багатофункціональні та динамічні веб-додатки, які забезпечують більш якісний користувацький досвід.

Bootstrap — це відомий фреймворк для створення адаптивних та стилізованих веб-сторінок, розроблений Twitter. Він надає великий арсенал готових компонентів, таких як кнопки, таблиці, форми, які полегшують роботу з дизайном та забезпечують єдність вигляду. Bootstrap вражає своєю адаптивністю, забезпечуючи легке пристосування сторінок до різних екранних розмірів, включаючи комп'ютери, планшети та мобільні пристрої. Застосовуючи гнучку систему сітки, Bootstrap дозволяє розміщати елементи на сторінці відповідно до різних екранних розмірів, створюючи рівномірні та зручні для читання макети.

Фреймворк включає різноманітні JavaScript-компоненти, такі як модальні вікна, випадаючі списки, каруселі, які додають інтерактивність до веб-сторінок без необхідності писати складний код з нуля. Bootstrap також надає можливості кастомізації, дозволяючи розробникам налаштовувати вигляд компонентів відповідно до дизайну свого проекту. Широка та активна спільнота розробників сприяє обміну досвідом, а висока якість офіційної

документації робить використання фреймворку зручним для всіх - і новачків, і досвідчених розробників.

Bootstrap не єдиний фреймворк, є ще React.js, Angular, Vue.js і Ember.js. Розглянемо та порівняємо між собою кожен з них.

React.js — це потужний фреймворк для розробки веб-інтерфейсів, створений інженерами у Facebook. Одна з його ключових особливостей - це концепція компонентів, яка дозволяє розбити інтерфейс на невеликі, незалежні елементи, спрощуючи розробку та підтримку коду. React також використовує віртуальний DOM, що дозволяє ефективно взаємодіяти з реальним DOM, забезпечуючи швидке оновлення сторінок та реакцію на зміни.

Однією з переваг React є його велика спільнота та екосистема, що сприяє великій кількості готових бібліотек та рішень. Також, React дозволяє використовувати JSX, що дозволяє вписувати HTML-подібний код безпосередньо в JavaScript, що полегшує роботу з компонентами та шаблонами.

Недоліками React може бути крутий початковий поріг входження, особливо для новачків у веб-розробці. Також, велика свобода вибору інших інструментів та бібліотек може призвести до великої різноманітності підходів у спільноті та важкості вибору оптимального стеку для конкретного проекту.

Ще однією перевагою React є можливість використання його для розробки мобільних додатків за допомогою React Native, що дозволяє розробникам використовувати один і той же код для веб та мобільних платформ, що полегшує підтримку та розвиток проектів.

Angular — це повноцінний фреймворк для розробки веб-додатків, створений командою в Google. Однією з ключових особливостей Angular є його використання TypeScript, що надає сильну типізацію та можливості роботи з об'єктно-орієнтованим кодом. Angular пропонує компонентний підхід до розробки, де весь інтерфейс поділений на невеликі, самостійні

компоненти, що полегшує управління кодом та підтримку проектів великих розмірів.

Однією з основних переваг Angular є його повноцінний набір інструментів, таких як система залежностей, система шаблонів, інструменти для тестування та інші. Це робить його ідеальним вибором для великих команд розробників, які шукають стандартизований та повний фреймворк для розробки.

Недоліками Angular може бути крутий початковий поріг входження через велику кількість понять і концепцій, а також складність вивчення для початківців. Однак, якщо команда розробників вже знайома з об'єктно-орієнтованим програмуванням і TypeScript, цей фреймворк може надати великий рівень контролю та структури для проектів будь-якої складності.

Vue.js представляє собою прогресивний фреймворк для розробки веб-інтерфейсів, який вирізняється своєю легкістю використання та гнучкістю. В основі Vue.js лежить концепція компонентів, аналогічно до React та Angular, що дозволяє розбити інтерфейс на повторно використовувані модулі. Це допомагає розробникам легко управляти кодом та підтримувати проекти будь-якого розміру.

Однією з основних переваг Vue.js є його простота вивчення. Навіть розробники з мінімальним досвідом можуть швидко освоїти Vue.js, завдяки зрозумілому API та документації. Це робить Vue.js привабливим вибором для початківців у веб-розробці та для тих, хто шукає ефективний інструмент для швидкого створення інтерактивних веб-додатків.

Vue.js також славиться своєю гнучкістю. Ви можете використовувати Vue.js для створення цілого односторінкового застосунку або інтегрувати його частково в існуючий проект. Його легкість інтеграції дозволяє розробникам використовувати Vue.js як засіб поступового оновлення або розвитку існуючих додатків.

Ще однією особливістю Vue.js є акцент на прогресивному вдосконаленні. Ви можете використовувати лише ті частини фреймворка, які

вам потрібні, що дозволяє уникнути зайвого навантаження на додаток. При цьому, при необхідності, ви можете додавати нові функції по мірі розвитку проекту.

Однак, через меншу кількість заснованих на компонентах бібліотек та менший розмір спільноти порівняно з React та Angular, може бути складніше знайти готові рішення та бібліотеки для Vue.js. Також, на відміну від React і Angular, Vue.js не має великого підтримуваного компанією, що може вплинути на його довгострокову стабільність та розвиток.

Ember.js — це фреймворк для розробки веб-додатків, який надає високорівневий рівень абстракції, спрощуючи процес розробки амбіційних веб-проектів. Однією з ключових особливостей Ember.js є його конвенції над конфігурацією підходу, що забезпечує однорідність та структурованість проектів.

Основною перевагою Ember.js є швидка розробка завдяки своєму високорівневому підходу та готовим конвенціям. Він надає стандартну структуру проекту, що полегшує спільну роботу розробників у великих командах, а також сприяє швидкому впровадженню нових розробок.

Ще однією особливістю Ember.js є Ember CLI — потужний інструмент командного рядка, який спрощує створення, відладку та тестування додатків. Це дозволяє розробникам ефективно керувати проектом та автоматизувати багато задач.

Недоліками Ember.js може бути його крутий початковий поріг входження для новачків, оскільки високорівневий підхід та конвенції можуть вимагати часу для освоєння. Крім того, у порівнянні з іншими популярними фреймворками, спільнота Ember.js може бути меншою, що може впливати на доступність готових бібліотек і рішень для розробки.

Нижче зображена порівняльна таблиця фреймворків.

Отож, розглянувши порівняльну таблицю фреймворків було прийнято рішення для розробки веб-застосунку використовувати React.

Таблиця 2.2 — Порівняльна таблиця фреймворків

Характеристика	React.js	Angular	Vue.js	Ember.js
Основна концепція	Компонентний підхід, віртуальний DOM	Компонентний підхід, TypeScript, RxJS	Компонентний підхід, прогресивний	Високорівневий підхід, конвенції
Мова програмування	JavaScript (може використовувати JSX)	TypeScript	JavaScript	JavaScript (може використовувати TypeScript)
Вивчення та початковий поріг	Помірно високий	Високий	Низький	Високий
Швидкодія	Швидкий рендерінг через віртуальний DOM	Потужний інструментарій, але може бути важким	Легкий та ефективний	Потужний та ефективний
Спільнота	Велика, активна	Велика, активна	Велика, активна	Середня, менша в порівнянні з іншими
Екосистема	Розгалужена, з великою кількістю бібліотек та інструментів	Обширна, з багатьма інструментами та бібліотеками	Розвинена, але менше порівняно з React та Angular	Широка, з Ember CLI та готовими конвенціями
Можливість використання для мобільних додатків	За допомогою React Native	За допомогою Angular NativeScript	За допомогою Vue NativeScript, Quasar	Невизначено, менш популярний в контексті мобільних додатків

2.3.2 Аналіз технологій для розробки серверної частини

Розглянемо серверну частину, яка відповідає за обробку запитів та взаємодію з базою даних.

Серверна частина веб-додатка — це механізм, що відповідає за обробку та управління даними, які відправляються та отримуються від клієнтської частини. Сюди входять різноманітні компоненти та технології, які забезпечують функціональність та взаємодію із базою даних.

Центральним елементом є сервер, який приймає запити від клієнта та повертає відповіді. Сервер виконує різні завдання, такі як обробка запитів, автентифікація користувачів, управління сесіями та віддача даних, необхідних для побудови сторінок.

Безпека також важлива частина серверної частини, і тут використовуються різні методи для захисту від атак, таких як SQL-ін'єкції чи атаки на переповнення буфера. Також в серверну частину може бути включений механізм кешування, що дозволяє прискорити віддачу даних та поліпшити продуктивність додатка.

Серверна частина зазвичай використовує мови програмування, такі як Python, Java, Ruby або Node.js. Основним завданням є обробка запитів, взаємодія з базою даних для отримання та збереження інформації, а також надання даних клієнтській частині у зручному для обробки форматі. Серверна частина відіграє ключову роль у створенні ефективного та безпечного веб-додатка.

Java — це мова програмування, що визначається своїм об'єктно-орієнтованим підходом та високою переносимістю коду. Завдяки своїй спрощеній структурі та вбудованій підтримці багатозадачності, вона стала популярним вибором для розробки серверних додатків та веб-служб.

Однією з ключових особливостей Java є її здатність використовувати віртуальну машину (JVM), що дозволяє коду працювати на різних платформах без перекомпіляції. Високорівневі бібліотеки для роботи з

потоками та багатопоточністю дозволяють ефективно обробляти багато запитів одночасно.

Java відома своєю системою безпеки, що використовує механізми для уникнення багатьох програмних помилок та гарантує надійність додатків. З великою стандартною бібліотекою, що включає інструменти для мереж, обробки XML, баз даних, вона стає потужним інструментом для розробників серверних застосунків. Вона також заслуговує визнання за свою екосистему, яка включає в себе фреймворки, такі як Spring і Hibernate, що полегшують розробку та управління складними додатками.

Python — це мова програмування, яка вражає своєю читабельністю та простотою синтаксису. Її гнучкість дозволяє вам виражати ідеї меншою кількістю рядків коду, що полегшує розробку та підтримку програм. Однією з визначних особливостей Python є його велика екосистема бібліотек та фреймворків, що сприяє розвитку різноманітних застосунків у сферах від веб-розробки до штучного інтелекту.

Важливою перевагою Python є його загальне використання в галузях науки, освіти та бізнесу. Завдяки великій спільноті розробників та активному внеску в удосконалення мови, Python постійно оновлюється та пристосовується до сучасних вимог. Також варто відзначити підтримку об'єктно-орієнтованого та функціонального програмування, що розширює можливості розробників та забезпечує високий рівень абстракції.

Однак Python не позбавлений недоліків. Його виконавча швидкість може бути меншою порівняно з іншими мовами, такими як C++ чи Java, що може становити проблему для великих обчислювальних завдань. Також, відсутність строгої типізації може призводити до потенційних помилок на етапі виконання програми. Незважаючи на це, враховуючи його широке використання та зручний синтаксис, Python залишається однією з найпопулярніших та універсальних мов програмування у світі.

Ruby — це мова програмування, яка виділяється своєю елегантністю та приємністю для розробників. Її синтаксис допомагає створювати код, що

легко читається та зрозуміліший для новачків. Однією з ключових особливостей Ruby є принцип "зручності для програміста", що сприяє швидкій інтеграції та розробці програм.

Мова Ruby відома своєю принциповою об'єктністю - усе в Ruby є об'єктом, включаючи числа та навіть класи. Це робить код більш модульним та дозволяє використовувати об'єктно-орієнтований підхід для різноманітних завдань. Ruby також володіє активною спільнотою та розширюється багатозадачністю, що робить її популярною серед розробників веб-додатків.

Незважаючи на свої переваги, Ruby може мати проблеми з продуктивністю порівняно з іншими мовами, особливо у великих проектах. Також, в порівнянні з іншими мовами, вона може виявитися менш популярною в окремих областях, таких як наукові дослідження чи розробка вбудованих систем. Тим не менше, зручний синтаксис та активна спільнота роблять Ruby привабливим вибором для великої кількості проектів, особливо в сфері веб-розробки.

Крім того, важливою рисою мови Ruby є її фреймворк для веб-розробки — Ruby on Rails. Rails пропонує конвенції над конфігурацією, що дозволяє розробникам швидше створювати стабільні та сучасні веб-додатки. Інтеграція цього фреймворку з Ruby робить їх популярними в середовищі веб-розробки, адже вони пропонують швидкий старт та високий рівень продуктивності.

Однак, подібно до Python, Ruby може стикатися з обмеженнями у виконавчій швидкості порівняно з деякими мовами програмування. Це може вплинути на вибір Ruby для великих обчислювальних завдань або проектів з високими вимогами до продуктивності. Також, спільнота Ruby менша порівняно з деякими іншими мовами, що може призвести до обмеженої кількості бібліотек та ресурсів порівняно з іншими мовами програмування.

Node.js — це середовище виконання JavaScript за межами веб-браузера, яке використовується для створення серверних додатків та мережевих застосунків. Відмінністю Node.js є можливість використовувати JavaScript

для написання серверного коду, що дозволяє розробникам використовувати одну мову програмування на обох сторонах — клієнтській та серверній.

Однією з ключових переваг Node.js є його асинхронна модель виконання, яка дозволяє обробляти багатозадачні операції без блокування інших операцій. Це робить Node.js особливо ефективним для операцій, які вимагають великої кількості одночасних підключень, таких як реального часу веб-додатки.

Node.js також відомий за своєю великою бібліотекою модулів та фреймворків, таких як Express.js, що спрощують розробку веб-додатків. Це забезпечує розробникам широкий вибір інструментів для створення різноманітних застосунків, включаючи API, чат-боти, та інші мережеві сервіси.

Незважаючи на свої переваги, Node.js має певні обмеження. Використання одного мови для клієнтської та серверної частини може призвести до плутанини коду в більших проектах. Також, через асинхронний характер, деякі завдання можуть бути важкими для розуміння та відлагодження для новачків у світі асинхронного програмування.

Усупереч своїм недолікам, Node.js залишається популярним вибором для створення ефективних та масштабованих серверних додатків, зокрема в сферах реального часу та додатків з великою кількістю одночасних підключень.

Нижче зображена порівняльна таблиця 2.3 технологій для серверної частини.

Отож, розглянувши усі переваги та недоліки, було прийнято рішення обрати для серверної частини Node.js.

Таблиця 2.3 — Порівняльна таблиця фреймворків

	Node.js	Ruby	Python	Java
Використання	Серверні додатки, API	Веб-розробка, скриптінг	Веб-розробка, наука, AI	Веб-розробка, мобільні додатки, корпоративні системи
Спосіб виконання	Асинхронний	Синхронний	Змішаний	В основному синхронний
Об'єктно-орієнтована	Так	Так	Так	Так
Спільнота та Екосистема	Активна, велика бібліотека модулів та фреймворків	Активна, зокрема Ruby on Rails	Велика, багато бібліотек та фреймворків	Велика, розгалужена спільнота та багато інструментів
Швидкість виконання	Висока	Помірна	Помірна, але може бути повільною	Висока
Фреймворки	Express.js, Koa, Nest.js	Ruby on Rails	Django, Flask, FastAPI	Spring, Hibernate
Підтримка паралельності	Так	Ні	Так	Так
Складність вивчення	Помірна	Помірна	Низька	Середня

3 РОЗРОБКА РОЗПОДІЛЕНОЇ СИСТЕМИ ДЛЯ ПІДТРИМКИ ФУНКЦІОНУВАННЯ АВТОПАРКІНГУ

3.1 Розробка структурної схеми розподіленої системи

За результатами проведеного у першому розділі аналізу розглядувана розподілена система автопаркінгу, виходячи із завдань, що вирішуються аналогами, на фізичному рівні повинна забезпечувати контрольований доступ автомобілів на територію парковки та контрольований їх виїзд з оплатою послуг за користуванням паркінгом, вести моніторинг кількості та розташування вільних місць, надавати клієнтам інформацію про кількість вільних місць не лише онлайн, а й безпосередньо на в’їзді, інформаційно допомагати водіям з пошуком вільних місць, запобігати зловживанням обслуговуючого персоналу.

Контрольований заїзд на територію парковки зазвичай забезпечується з використанням в’їзного шлагбаума. Для розрахунку вартості послуги паркінгу необхідно вести відліку часу перебування автомобіля на парковці, а тому потрібно зафіксувати час в'їзду автомобіля. Фіксація часу заїзду здійснюється в момент відкриття в’їзного шлагбауму. Найпростішим варіантом, який набув найбільшого поширення, є відкриття в'їзного шлагбауму за натисканням кнопки, яке здійснюється самим водієм. Відкриття відбувається, якщо на парковці є вільні місця. При цьому водію надається в'їзний талон, за допомогою якого встановлюється зв′язок між автомобілем (водієм) та записом у базі даних інформаційної системи паркінгу. Основною інформацією, що зазначається на талоні, є час та дата заїзду. Для автоматизації процесів керування парковкою уся інформація, що відповідає запису у базі даних, подається на талоні у вигляді штрих або QR-коду. З її використанням розраховується вартість послуги, яку має сплатити водій при виїзді з парковки. Відкривання виїзного шлагбауму відбувається після зчитування даних з талону та проведення оплати.

Моніторинг кількості доступних для паркування вільних місць на парковці можна здійснювати програмним шляхом за моментами в'їзду та

виїзду автомобілів. Проте при такому підході місце для паркування, що звільнилося, буде вважатися зайнятим доти, поки автомобіль не проїде виїзний шлагбаум. Більш ефективне використання місць для паркування, що звільнилися, може буте досягнуте якщо контролювати момент звільнення місця для паркування, а не момент виїзду автомобіля. Такий контроль може бути забезпечений за допомогою датчиків присутності автомобіля, що підвищить ефективність обслуговування клієнтів.

Інформацію про наявну кількість вільних місць доцільно надавати не лише онлайн, а й безпосередньо перед в'їзним шлагбаумом на інформаційному табло. Це буде особливо актуально тоді, коли кілька автомобілістів одночасно хочуть скористатся парковкою, у результаті чого утворюється черга на в'їзді. Використовуючи інформацію про кількість наявних вільних місць, водії у черзі можуть оцінити можливість залишити свій автомобіль на даній парковці.

Зменшенню заторів на території парковки сприятиме інформування водіїв про розташування вільних місць, що можна реалізувати кількома способами. По-перше, номер вільного місця можна зазначати на талоні, що надається водієві в момент заїзду на парковку. На самій парковці будуть розміщуватися стенди зі схемою нумерації місць для паркування.

При другому варіанті стенди можуть бути замінені на електронні табло, на яких на схемі розташування місць для паркування будуть відображатися вільні місця. У цьому випадку водій самостійно обиратиме місце для паркування.

При третьому способі можна використати більш прості табло, які будуть розміщуватися у місцях розгалужень разом з вказівниками напрямів проїзду. На табло буде відображатися кількість вільних місць у зазначеному вказівником напрямі.

Серед цих можливих трьох варіантів найкращим є третій. На відміну від випадку, при якому вільні місця розподіляються на в'їзді і вказуються на талоні, він є більш привабливим з точку зору водія, оскільки залишає за ним

право самостійно вибирати найбільш зручне з його погляду місце, з тих що пропонуються. У порівнянні з випадком, що передбачає використання інформаційних табло, на яких вільні місця позначаються на схемі проїзду, він більш простий та менш затратний для технічної реалізації, дозволяє більш швидше зорієнтувати водія тощо. З врахуванням викладеного у розглядуваній системі будемо використовувати підхід, при якому у місцях розгалужень будуть розташовані табло для відображення кількості вільних місць за вказаним напрямом проїзду.

Розрахунок вартості користування парковкою відбуватиметеся на виїзді за фактичним часом перебування автомобілю на парковці. Право виїзду надаватиметеся за допомогою виїзного шлагбауму після здійснення оплати. Фактичний час користування парковкою визначається за часом під'їзду до вихідного шлагбауму та часом заїзду, що зазначений записом у базі даних, доступ до якого відбувається за шрих або QR-коду на талоні.

З врахуванням викладеного вище приходимо до структурної схеми апаратно-програмної частини розподіленої системи, яка наведена у Додатку В. Основними структурними блоками є: сервер парковки, концентратор, датчики присутності автомобіля, в'їзна стійка, термінал управління виїздом, в'їзний та виїзний шлагбауми, головне та допоміжні інформаційні табло.

Головним елементом розглядуваної частини системи є сервер парковки. Основними його завданнями є: облік часу перебування автомобіля на парковці та розрахунок вартості послуги, моніторинг кількості вільних місць, відображення цієї інформації на інформаційних табло. Вирішення завдань управління та інформаційної підтримки роботи парковки забезпечується взаємодію сервера з в'їзною стійкою, обладнанням терміналу управління виїздом та інформаційними табло.

В'їзна стійка забезпечує управління заїздом автомобілів на парковку. Заїзд автомобіля ініціалізується натисканням кнопки, що розміщена на ній. Сервер отримує цей сигнал і при наявності вільних місць надсилає на в'їзну стійку сигнал про дозвіл на здійснення заїзду. За цим сигналом у в'їзній

стійці відбувається друк талону та формується сигнал на підняття шлагбауму.

Термінал управління виїздом забезпечує стягування оплати та керування виїзним шлагбаумом. Стягування оплати забезпечується оператором за допомогою зчитувача штрих або QR-коду, який сканує код на наданому водієм талоні, на підставі чого здійснюється розрахунок суми до оплати за користування парковкою. Після проведення оплати, яка може здійснюватися як у готівковій, так і або безготівковій формі за допомогою банківського платіжного терміналу, оператор виконує дії, активізують підняття виїзного шлагбауму.

Відкривання шлагбаумів забезпечується контролерами, які входять до системи їх управління. Ці контролери не лише здійснюють вмикання механізму, що піднімає стрілу шлагбаум, а й контролюють проїзд автомобіля, забезпечують автоматичне закриття шлагбауму.

Моніторинг кількості вільних місць забезпечується шляхом отримання сервером даних від концентратора. Концентратор є центральним комунікаційним вузлом, який збирає вихідні дані з окремих датчиків за допомогою бездротового зв'язку. Для розширення радіусу охоплення можуть використовуватися ретранслятори. Інформація про поточний стан датчиків (зайнятості місця для паркування) надсилається концентратору, який передає її у сервер парковки через мережеве підключення Ethernet. З використанням інформації, отримуваної від концентратора, сервер веде облік кількості вільних місць, відображає цю інформацію на інформаційних табло та ретранслює її до мережевого хмарного серверу.

Для реалізації запропонованої схеми скористаємося готовими рішеннями, що дозволить зменшити фінансові витрати та скоротити час на впровадження системи. Аналіз ринку систем управління паркінгом показав, що повністю готового рішення для реалізації усіх зазначених завдань з використанням обладнання одного виробника не існує. Пропонуються або системи автоматичного або напівавтоматичного управління паркингом у

режимі оф лайн, або системи управління паркінгом онлайн. Виходячи з цього для реалізації розглядуваної системи скористаємося обладнанням двох виробників: компанії MOKOLoRa, яка пропонує інтелектуальні рішення у сфері IoT, та компанії SEA, що пропонує обладнання для систем автопаркінгу.

За допомогою обладнання MOKOLoRa реалізуємо функції моніторингу зайнятості місць для паркування, а за допомогою обладнання SEA — функції управління та інформаційної підтримки.

Як датчик присутності автомобіля будемо використовувати датчик LW009-IG/SM, який є бездротовим детектором транспортних засобів, який поєднує в собі мікрохвильовий радар та технологію геомагнітного виявлення. Він придатний для визначення стану внутрішньої та зовнішньої парковки [11]. Для надійного та точного визначення зайнятості парковочного місця та статистики часу паркування використовується вдосконалений алгоритм виявлення сигналу (рисунок 3.1).

Рисунок 3.1 — Датчик присутності автомобіля LW009-IG/SM

Як концентратор використаємо шлюз MKGW2-LW, що є бездротовим модулем, який обмінюється даними між термінальним пристроєм та LNS у LoRa-мережі. В основному використовується для передачі даних від датчиків з електронних пристроїв в хмару. Шлюзи LoRa можна використовувати для створення мережевих реалізацій відповідного електричного обладнання,

особливо в середовищах, де інші типи мереж недоступні через технічні обмеження [11].

Сам шлюз підключається до мережевого сервера LoRaWAN через мережу з високою пропускною здатністю, таку як WiFi, Ethernet або мобільна мережа, для забезпечення прямого з'єднання між вузлами LoRa та серверами застосунків. Завдяки відносно низькій швидкості передачі даних між кінцевими вузлами (вузли або термінальні пристрої) та шлюзами LoRaWAN, дальність передачі даних дуже велика, а споживання енергії дуже низьке (рисунок 3.2).

Рисунок 3.2 — Шлюз MKGW2-LW

Як в’їзну стійку вибираємо стійку серії PT21IGM-010100 (рисунок 3.3), що може керувати шлагбаумами, воротами, боллардами (висувними стовпами) різних виробників, моделей і типів [12].

Дана стійка дозволяє здійснювати керування шлагбаум з датчиками індукційних петель і оптичними датчиками безпеки та має вбудований модуль безперебійного живлення (до 20 хв.).

Як інформаційне табло вибираємо табло "кількість вільних місць". Такий тип інформаційних світлодіодних табло використовується у якості в'їзних табло для організації відкритих автоматичних та напівавтоматичних парковок, у тому числі і перехоплюючих парковок [12]. Для відображення

змінної інформації в табло парковки використовується сучасний повноколірний LED-екран (рисунок 3.4).

Рисунок 3.3 — В’їзна стійка

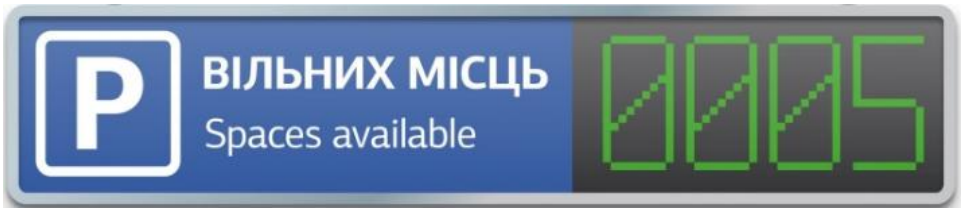

Рисунок 3.4 — Інформаційне табло

Технічні характеристики інформаційного табло:

— Виконання, для установки на вулиці;
— захист від зовнішнього середовища, фронт — ip65, тил — ip54;
— робочий діапазон температур, від мінус 20 до +60 °С;
— напруга живлення, 220 В;
— оптимальна зона видимості написів, від 15 до 85 м;
— кути огляду, горизонталь/вертикаль, 120° / 60°;

— яскравість, 7000 ніт.

Як сервер парковки може бути використаний комп’ютер, основні параметри якого наведені у таблиці 3.1

Таблиця 3.1 — Основні вимоги до сервера парковки

Параметр	Значення
Операційна система	Microsoft Windows 8-16 або один з дистрибутивів Linux
Процесор	Процесор x64: Intel Xeon E26xx 2,4 ГГц або аналогічний AMD
Оперативна пам′ять	не менше 8 Гбайт

3.2 Розробка мобільного веб-застосунку

Для розробки веб-застосунку було обрано середовище Visual Studio Code. VS Code — це потужний та легкий текстовий редактор, розроблений Microsoft для підтримки розробки програмного забезпечення на різних платформах. Його основними перевагами є зручний інтерфейс, висока продуктивність та розширена функціональність, які роблять його популярним серед розробників усього світу (рисунок 3.5).

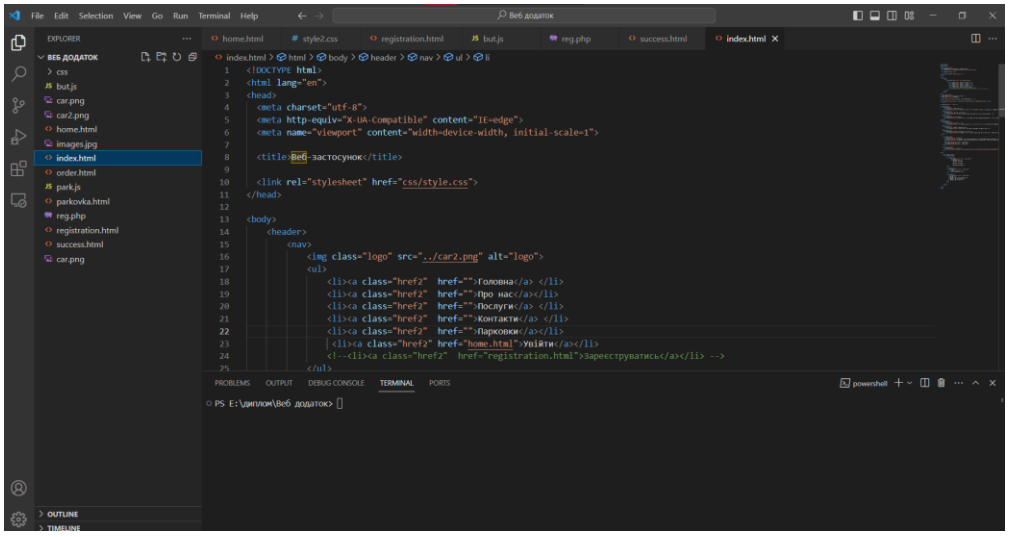

Рисунок 3.5 — Середовище Visual Studio Code

Однією з ключових особливостей VS Code є його висока розширюваність. Редактор підтримує велику кількість розширень, які дозволяють розробникам використовувати різноманітні мови програмування, інструменти та сервіси прямо в середовищі розробки. Розширення можуть додавати підтримку для конкретних мов, надавати додаткові функції відлагодження, інтегрувати системи контролю версій та забезпечувати інші корисні інструменти.

VS Code володіє вбудованими інструментами для відлагодження, включаючи точки зупинки, перегляд змін змінних, інтерактивні консолі та інше. Це робить процес відлагодження більш ефективним та зручним.

Також, редактор підтримує інтеграцію з системами контролю версій, що дозволяє розробникам легко взаємодіяти з репозиторіями, використовувати розгалуження та злиття, а також переглядати історію змін.

У додатку Д було розроблено блок-схему алгоритму роботи даного веб-застосунку.

3.2.1 Розробка веб-застосунку з використанням методу SPA

SPA (односторінковий додаток) — це сучасний метод розробки веб-застосунків, що використовується для створення ефективних та інтерактивних користувацьких інтерфейсів. Основна ідея полягає в тому, щоб завантажувати лише необхідний контент під час першого входу, а подальша навігація відбуватися без повторного завантаження сторінок. Цей підхід забезпечує плавну та швидку роботу додатку, зменшуючи час очікування користувача.

Однією з ключових переваг є висока швидкодія SPA. Завдяки використанню асинхронного завантаження контенту через AJAX-запити, користувачі можуть ефективно взаємодіяти з додатком без зайвого часу на перезавантаження сторінок. Це особливо важливо в епоху, коли швидкість та ефективність є ключовими чинниками задоволення усіх користувачів (рисунок 3.6).

SPA також легше адаптується для різних платформ, забезпечуючи оптимальний досвід використання незалежно від пристрою. Це робить SPA ідеальним вибором для розробки мобільних додатків, забезпечуючи зручний інтерфейс.

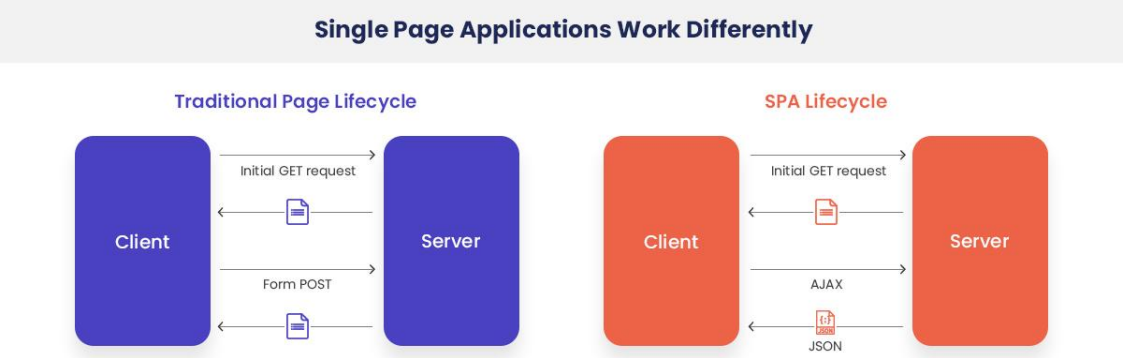

Рисунок 3.6 — Відмінність SPA від багатосторінкових додатків

З необхідністю відповіді на виклики мобільного світу, SPA також може використовувати кешування для забезпечення роботи в офлайн-режимі. Якщо користувач раніше відвідував додаток, він може продовжити взаємодію, навіть якщо втратив з'єднання з Інтернетом.

Незважаючи на ці переваги, важливо враховувати і недоліки SPA. При першому вході може виникнути більше часу на завантаження, оскільки потрібно отримати весь необхідний код і ресурси. Також, у зв'язку з динамічним завантаженням контенту за допомогою JavaScript, індексація додатків пошуковими системами може бути ускладненою, що впливає на SEO-показники.

3.2.2 Програмна реалізація веб-застосунку

Створимо проект для клієнтської частини системи автопаркінгу. Відкриймо термінал у Visual Studio Code та введемо наступну команду: npx create-react-app autoparking. Ця команда використовує create-react-app, щоб автоматично створити новий проект React з усіма необхідними налаштуваннями та структурою каталогів.

Після цього, у каталозі було створено файл package.json. Webpack — це модульний збирач, який дозволяє об'єднувати ресурси та бібліотеки, необхідні для проекту, у єдиний файл. Для його встановлення використовуємо наступну команду: npm install webpack webpack-cli --save-dev. Після цього, webpack та його інтерфейс доступу через командний рядок були додані до проекту як dev-залежності.

Після встановлення всіх необхідних компонентів для розробки, ми заповнимо файли конфігурації та базову структуру React-проекту. Для управління станом веб-додатку ми будемо використовувати бібліотеку Redux. Redux — це відкрита бібліотека для JavaScript, яка призначена для керування станом додатку. Зазвичай вона використовується у поєднанні з React або Angular для розробки клієнтської частини. Бібліотека включає ряд інструментів, які значно спрощують передачу даних між компонентами через контекст.

Щоб встановити Redux та його необхідні пакети у проекті як залежності, використовуємо наступну команду: npm install redux react-redux redux-thunk. Це дозволить нам використовувати Redux для ефективного управління станом нашого React-додатку.

Для ефективної роботи із сховищем в Redux використовуються основні елементи:

— state — це власне сховище, де зберігається поточний стан додатку;

— thunk — асинхронний аналог action. thunk дозволяє виконувати складні асинхронні операції та обробляти їхні результати перед передачею до reducer;

— reducer — це функція, яка обробляє "сигнали" або actions, змінюючи структуру сховища у відповідності до отриманих actions;

— dispatch — це функція, яка надсилає action до глобального обробника. глобальний обробник передає цей action до відповідного reducer для обробки;

— action — це простий об'єкт, який сигналізує, що необхідно змінити стан додатку через виклик певних змін.

Ці елементи працюють у взаємодії, дозволяючи ефективно керувати станом додатку в Redux. Thunk використовується для асинхронних операцій, reducer здійснює зміни в стані на основі actions, і dispatch надсилає ці actions до обробників. Усе це разом утворює потужний інструмент для керування станом додатку в React.

У React існує вбудована система маршрутизації, яка дозволяє визначати компоненти для різних запитів до додатку. Ключовим елементом у цьому процесі є бібліотека react-router, яка надає основний функціонал для роботи з маршрутами. Однак, для забезпечення навігації у браузері, рекомендується використовувати react-router-dom. Для встановлення необхідних залежностей в проекті використовуємо наступну команду: npm install react-router react-router-dom. Це дозволить вам використовувати функціонал маршрутизації в React за допомогою бібліотек react-router і react-router-dom.

Кожен маршрут визначається елементом `Route`, який має кілька атрибутів для конфігурації.

— path — це шаблон адреси, з яким клієнтський URL буде порівнюватися для визначення, яку компоненту відобразити;

— component — це компонента, яка буде відображена, коли виконується відповідний маршрут, визначений шляхом;

— render — цей атрибут приймає функціональну компоненту і дозволяє передати об'єкт `props` для цієї компоненти. Використовується для більш гнучкої конфігурації відображення компоненти;

— exact — якщо цей атрибут встановлено, маршрути будуть порівнюватись строго. Це означає, що клієнтський URL повинен точно відповідати шляху, вказаному у маршруті.

В головному файлі App.js запишемо імпорти інших необхідних компонент та бібліотек:

```
function App() {
  return (
    <Router>
      <div>
        <Switch>
          <Route exact path="/" component={Home} />
          <Route path="/register" component={ParkingRegistration} />
        </Switch>
      </div>
    </Router>
  );
}
export default App;
```

У компоненті Home.js напишемо код логіки отримання списку паркомісць з сервера:

```
const Home = () => {
  const [parkingSpaces, setParkingSpaces] = useState([]);
  useEffect(() => {
    const fetchData = async () => {
      try {
        const response = await fetch('/api/parking-spaces');
        const data = await response.json();
        setParkingSpaces(data);
      } catch (error) {
        console.error('Помилка отримання даних:', error);
      }
    };
    fetchData();
  }, []);
```

Компонент ParkingRegistration.js відповідає за реєстрацію та вибору паркомісця, лістинг якого наявний у Додатку Г.

Тепер перейдемо до розробки бази даних. База даних є централізованим простором, призначеним для збереження різноманітної інформації, такої як дані користувачів, транзакції, текстові документи, аудіо- та відеозаписи. Основна мета бази даних полягає в тому, щоб забезпечити ефективний та структурований доступ до цієї інформації. Для цього вона використовує спеціалізовані мови запитів, такі як SQL (Structured Query Language), які дозволяють взаємодіяти з базою даних, виконувати пошук, фільтрацію та здійснювати зміни у збережених даних.

Однією з ключових особливостей баз даних є їхній вміст у вигляді таблиць, де кожен рядок представляє конкретний запис, а кожна колонка визначає конкретний атрибут цього запису. Це дозволяє легко впорядковувати та аналізувати великі обсяги інформації. Крім того, бази даних використовуються для забезпечення консистентності даних, контролю доступу та забезпечення безпеки інформації.

Розширенням концепції баз даних є реляційні бази даних, де інформація організована у взаємозв'язаних таблицях, що спрощує обробку та аналіз даних. Такі системи грають ключову роль у різних галузях, включаючи бізнес, науку, освіту та багато інших сфер. Бази даних є фундаментальним елементом сучасних інформаційних технологій, дозволяючи ефективно управляти та аналізувати великі обсяги даних для прийняття рішень і вдосконалення бізнес-процесів. Розробку бази даних буде здійснюватися у середовищі MySQL.

MySQL — це відкрите програмне забезпечення для управління реляційними базами даних, розроблене та підтримуване Oracle Corporation. Відзначається відкритістю та безкоштовністю використання в межах умов GNU General Public License. Заснована на стандартній мові запитів SQL, MySQL взаємодіє з базою даних, забезпечуючи сумісність із стандартами реляційних баз даних.

Знаходячи застосування в різних галузях, від веб-розробки та систем управління контентом до електронної комерції та аналітики, MySQL підтримує транзакції для безпечних операцій та забезпечує масштабованість від невеликих проєктів до великих корпоративних систем.

Сильною стороною є активна спільнота користувачів, яка забезпечує підтримку, розробку та спільноту для обміну ідеями та рішеннями. Використовується багатьма великими веб-сайтами та популярними CMS, свідчачи про простоту використання, ефективність та надійність MySQL в різноманітних сценаріях застосування.

Для того аби розпочати розробку бази даних, слід запустити середовище MySQLWorkbench та зобразити ER-діаграму яка буде відображати сутності, їх атрибути та відносини між сутностями. ER-діаграма допомагає визначити основні компоненти бази даних та їх взаємозв'язки перед тим, як створювати схему бази даних та розташована у Додатку Д.

Створюємо саму базу даних, в якій будемо створювати таблиці для роботи (рис 3.7).

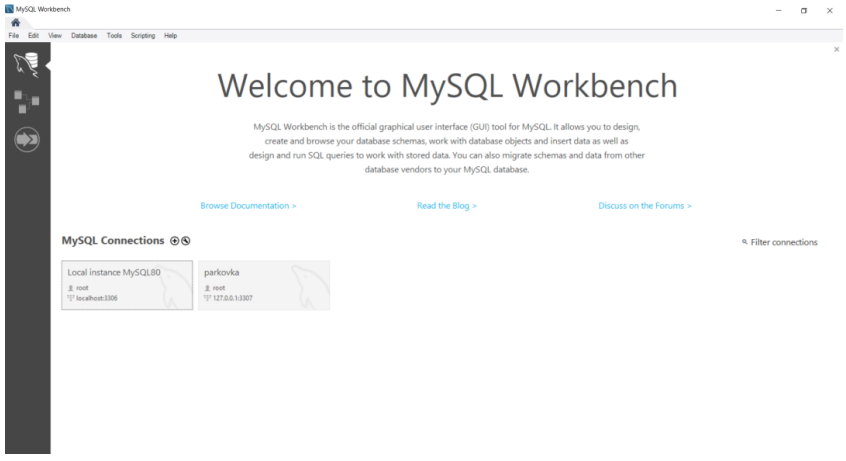

Рисунок 3.7 — MySQLWorkbench

Отож, база даних складається із п’яти таблиць : клієнт, автомобіль, паркінг, оренда паркомісця та платежі. Розпочнемо з таблиці для клієнтів. Вона необхідна для збереження інформації про користувача парковки. Ця

таблиця містить такі поля як, id клієнта, ім’я та прізвище клієнта, його номер телефону та id картка.

Для того аби створити таку таблицю, слід написати такий код:

```
CREATE TABLE prkvk.customers (
  id INT NOT NULL AUTO_INCREMENT,
  first_name VARCHAR(45) NOT NULL,
  last_name VARCHAR(45) NOT NULL,
  phone_number INT NOT NULL,
  customer_card_id INT NULL,
  PRIMARY KEY (id));
```

Наступна таблиця містить інформацію про автомобіль користувача і має такі поля id та реєстраційний номер. Для створення напишемо код.

```
CREATE TABLE prkvk.vehicle (
  id INT NOT NULL AUTO_INCREMENT,
  registration_number INT NOT NULL,
  PRIMARY KEY (id));
```

Наступна таблиця містить інформацію про усі доступні парковки, їхню адресу, кількість загальних, вільних та зайнятих місць. Напишемо код для створення такої таблиці.

```
CREATE TABLE prkvk.parking (
  id INT NOT NULL AUTO_INCREMENT,
  Address VARCHAR(45) NOT NULL,
  number_of_available_spaces INT NOT NULL,
  Occupancy INT NOT NULL,
  PRIMARY KEY (id));
```

Для того аби відслідковувати час коли клієнт прибув на заправку та час коли він виїхав, вартість оренди паркомісця створимо ще одну таблицю, куди будуть записуватись усі ці дані.

```
CREATE TABLE prkvk.rentals (
  id INT NOT NULL AUTO_INCREMENT,
  customer_id INT NOT NULL,
  vehicle_id INT NULL,
  start_date_time DATETIME NOT NULL,
  end_date_time DATETIME NOT NULL,
  status VARCHAR(45) NOT NULL,
  rental_cost INT NOT NULL,
  PRIMARY KEY (id),
  INDEX rental_custom_id_idx (customer_id ASC) VISIBLE,
  INDEX rental_vehicle_id _idx (vehicle_id ASC) VISIBLE,
  CONSTRAINT rental_custom_id
  FOREIGN KEY (customer_id)
  REFERENCES prkvk.customers (id)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION,
  CONSTRAINT rental_vehicle_id
  FOREIGN KEY (vehicle_id)
  REFERENCES prkvk.vehicle (id)
  ON DELETE NO ACTION
  ON UPDATE NO ACTION);
```

І остання таблиця нам необхідна для збереження інформації про оплату паркомісця. Напишемо код для її створення.

```
CREATE TABLE prkvk.payments (
  id INT NOT NULL AUTO_INCREMENT,
```

```
payment_amount INT NOT NULL,
date_time_payment DATETIME NOT NULL,
payment_status VARCHAR(45) NOT NULL,
PRIMARY KEY (id));
```

Отож, ми створили усі необхідні таблиці (рис 3.8).

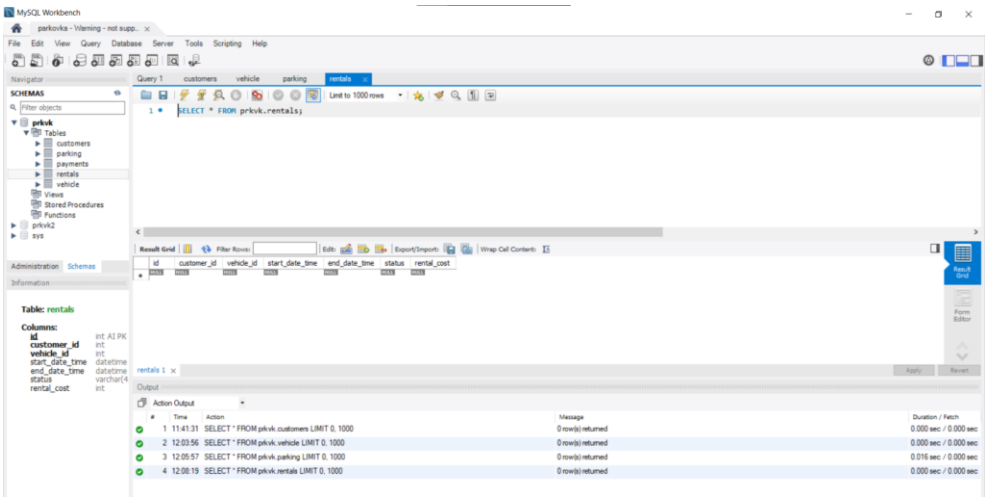

Рисунок 3.8 — Початковий вигляд бази даних у MySQL

Для того аби підключили базу даних до нашого проекту, було прийнято рішення використовувати Node.js. Перш ніж розпочати, переконаймося, що встановлено модуль mysql2. Використаємо команду: npm install mysql2. Тепер є все необхідне для підключення до бази даних. Створимо файл з кодом, наприклад, `app.js`, і у ньому запишемо наступний код.

```
const mysql = require('mysql2');
const connection = mysql.createConnection({
  host: '127.0.0:3307',
  user: 'rool',
  password: '15012001',
  database: 'parkovka'
});
connection.connect((err) => {
```

```
    if (err) {
        console.error('Помилка підключення до бази даних:', err);
    } else {
        console.log('Підключено до бази даних MySQL!');
        connection.end((err) => {
            if (err) {
                console.error('Помилка закриття з\'єднання:', err);
            } else {
                console.log('З\'єднання закрито.');
            }
        });
    }
});
```

Детальний опис коду із SQL запитами додано у Додаток Ж. Отож, базу даних було під’єднано.

4 РЕКОМЕНДАЦІЇ З РОЗГОРТАННЯ СИСТЕМИ

4.1 Опис технологій тестування веб-застосунків

Веб-тестування представляє собою ретельний аналіз веб-сайту для виявлення потенційних помилок та повне перевірка його функціональності перед його впровадженням. Перед тим як веб-система стає доступною для кінцевих користувачів, вона проходить вичерпне тестування від початку до кінця. Процес також включає оцінку зручності та естетичності веб-сайту для користувача, швидкості доступу до необхідної інформації, а також його надійності та безпеки.

Ключовими аспектами веб-тестування є визначення того, наскільки веб-сайт відповідає бізнес-цілям, які були поставлені на етапі початкового проекту. Загальна мета веб-тестування полягає в виявленні проблем, що можуть виникнути під час експлуатації веб-сайту або програми, таких як помилки чи дефекти. Крім того, веб-тестування може використовуватися для ідентифікації конкретних областей або аспектів веб-сайту, які можна оптимізувати для досягнення кращих результатів, таких як збільшення кількості запитів, підвищення обсягу продажів, залучення більше постійних відвідувачів тощо.

Тестування — це комплексний процес, який включає в себе використання різноманітних технік та інструментів для забезпечення якості та надійності веб-додатків перед їх випуском. Однією з основних складових є модульне тестування, яке дозволяє перевірити роботу окремих частин коду на відповідність вимогам.

Функціональне тестування спрямоване на перевірку відповідності застосунку функціональним вимогам. Воно гарантує, що веб-застосунок виконує очікувані завдання та взаємодіє з користувачем належним чином. Одним із важливих аспектів є тестування користувацького інтерфейсу, що включає перевірку навігації та відображення елементів.

Відмовостійке тестування визначає, як застосунок поводиться при великому навантаженні. Це дозволяє оцінити його пропускну здатність та швидкість відгуку під високим трафіком. Паралельно проводиться безпекове тестування, яке виявляє потенційні вразливості та забезпечує захист від атак.

Інтеграційне тестування стежить за взаємодією різних компонентів системи, впевнюючись, що вони працюють разом бездоганно. Автоматизація тестування використовується для ефективного виконання тестових сценаріїв, що дозволяє швидко перевірити новий код та уникнути регресійних помилок.

Тестування користувацького інтерфейсу — це процес перевірки веб-застосунку на відповідність його інтерфейсу вимогам. Воно фокусується на тому, як користувач взаємодіє з додатком, переконуючись в правильності та зручності цієї взаємодії.

Тестування функціональності інтерфейсу включає перевірку відображення всіх елементів та їх взаємодії, таких як кнопки та поля введення. Важливо перевірити відповідність елементів інтерфейсу специфікаціям та дизайну.

Тестування навігації зосереджується на перевірці логічності та ефективності переміщення між сторінками та розділами застосунку. Також важливо перевіряти роботу посилань та кнопок для переходу між частинами додатку.

Відповідність дизайну включає перевірку того, чи відповідають елементи дизайну специфікаціям та стандартам. Тестується відображення інтерфейсу на різних роздільностях екрану та на різних пристроях. Тестування сумісності інтерфейсу проводиться для перевірки його взаємодії з різними браузерами та операційними системами. Також важливо перевірити, як добре інтерфейс адаптується до різних пристроїв.

Адаптивний дизайн передбачає тестування реакції інтерфейсу на зміни розмірів вікна браузера та відображення на різних пристроях. Тестування форм введення даних включає перевірку їхньої валідації та взаємодії з сервером при їхньому відправленні.

Життєвий цикл тестування програмного забезпечення — це процес, що включає в себе планування, розробку, виконання тестів та аналіз результатів з метою забезпечення якості програмного продукту. Цей цикл відображає етапи тестування від початкового планування до завершення проекту (рисунок 4.1).

Починається процес з планування, де визначається мета та обсяг тестування, а також створюється план, включаючи стратегію, ресурси, графік і критерії завершення. Далі йде дизайн тестових випадків, розробка конкретних випадків тестування, визначення даних та очікуваних результатів.

Після цього настає етап реалізації тестових сценаріїв, де готують тестові дані та налаштовують середовище для виконання тестів. Виконують тест-кейси та збирають результати, при цьому виявляють і документують помилки, які обговорюють з розробниками для виправлення.

Після виконання тестів проводиться аналіз результатів, порівняння фактичних та очікуваних результатів, і визначення рівня якості продукту. Після виявлення та виправлення помилок розпочинається регресійне тестування, щоб переконатися, що зміни не вплинули на інші частини системи.

По завершенню тестування готується звіт, в якому фіксуються виявлені проблеми та можливі покращення. Завершується процес передачею результатів команді розробників або менеджменту проекту.

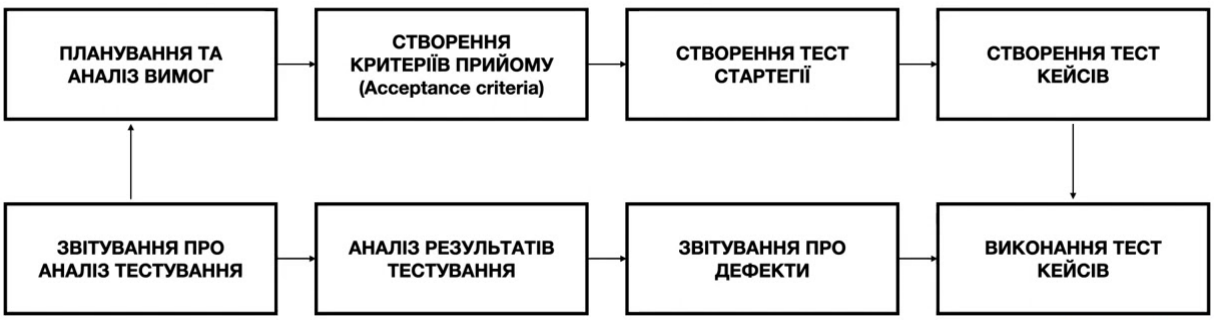

Рисунок 4.1 — Життєвий цикл тестування.

4.2 Тестування веб-застосунку

Тестування починається з підключення директив для відкриття веб-застосунку: npm run watch — для автоматичного оновлення компонентів вебзастосунку, npm start — для запуску клієнту, npm run start — для запуску вебсерверу. Проведемо спочатку тестування користувацького інтерфейсу.

Як було зазначено у третьому розділі, водій може скористатись парковкою безпосередньо дізнавшись про вільні місця вже на місці паркування, що будуть зображенні на таблі. Проте є й інший варіант — скористатись веб-застосунком.

Перше що необхідно зробити водієві це зайти на головну сторінку додатку. Якщо користувач вперше вирішив скористатись даними послугами, йому потрібно буде зареєструватись у системі. Для цього слід натиснути на кнопку “УВІЙТИ” і обрати зареєструватись.

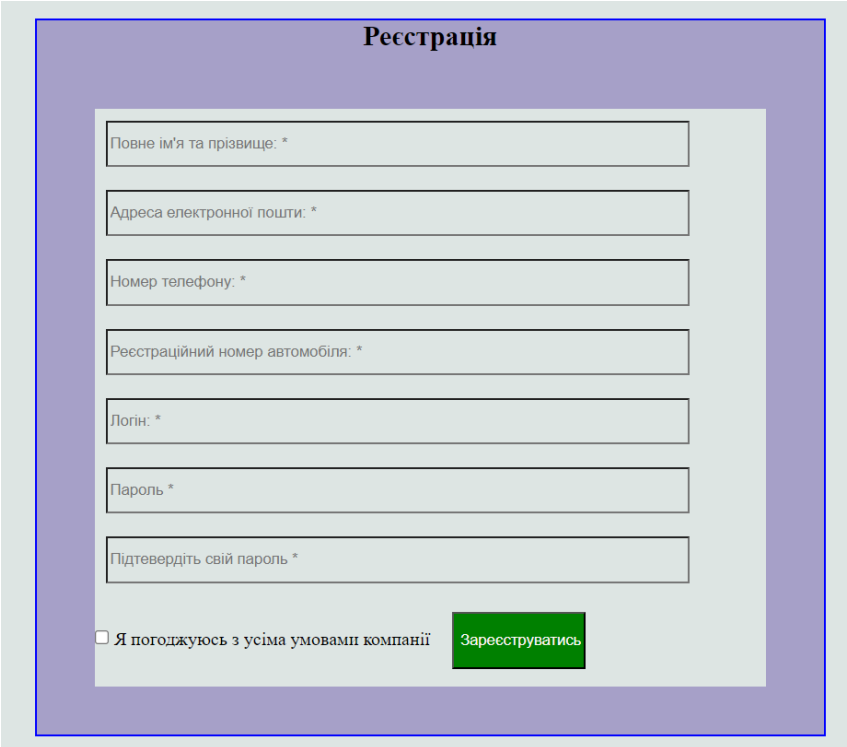

Рисунок 4.2 — Сторінка реєстрації користувача

Користувач реєструється, вводячи свої особисті дані, а після цього використовує цю інформацію для користування можливостями веб-

застосунку. Введені дані дані записуються у базу даних. Ввівши команду SELECT перевіримо успішність запису даних у базу даних (рисунок 4.3).

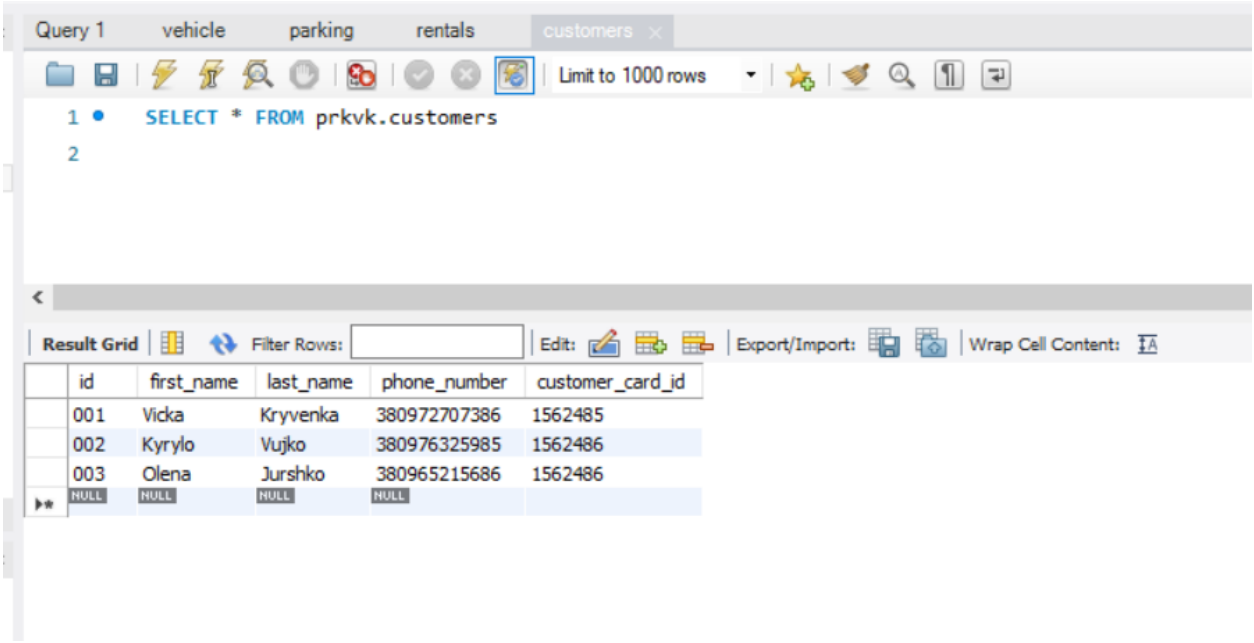

Рисунок 4.3 — Тестування бази даних

Отож, дані було записано успішно. Якщо користувач вже був зареєстрований, йому потрібно лише увійти у систему ввівши свій логін та пароль (рисунок 4.4).

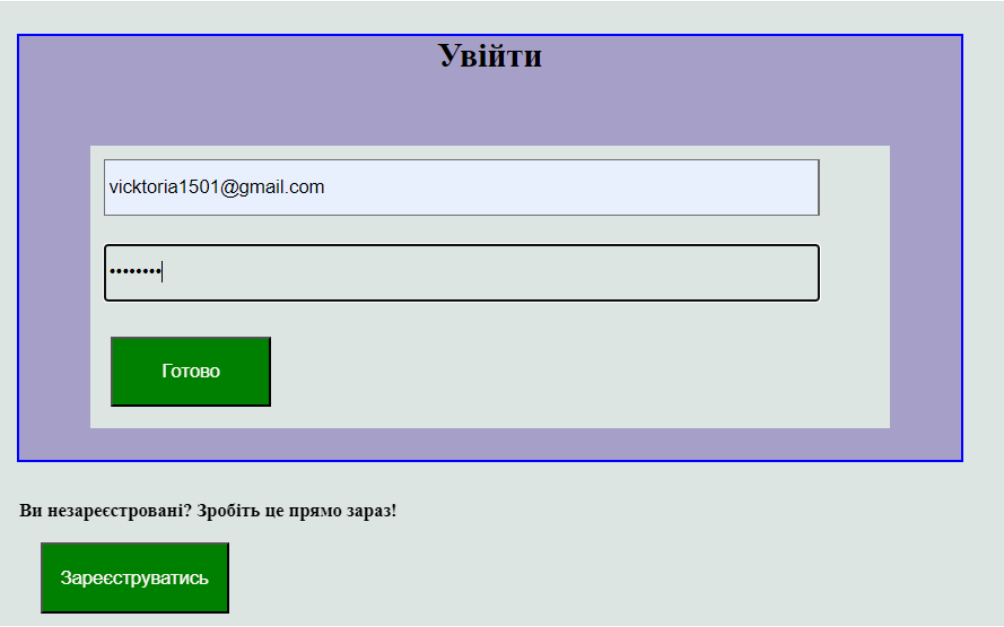

Рисунок 4.4 — Сторінка авторизації користувача

На головній сторінці користувач може ознайомитись із правилами компанії, прочитати про власників та історію створення. Натиснувши на кнопку “ЗНАЙТИ ПАРКОВКУ” водієві буде доступний вибір адрес, де є вільні місця (рисунок 4.5).

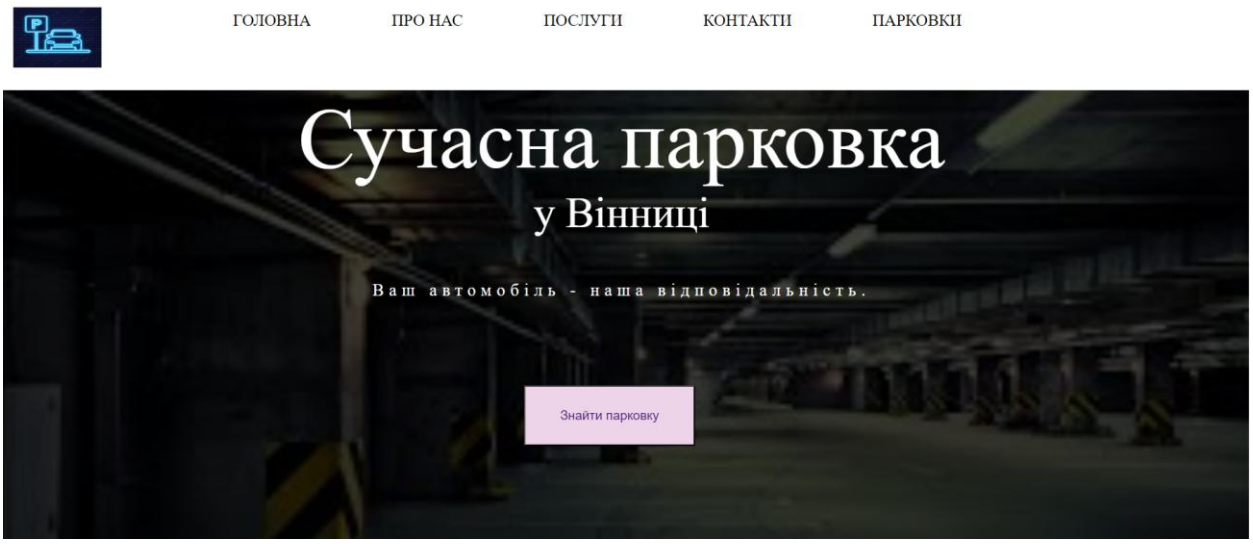

Рисунок 4.5 — Головна сторінка

На сторінці вибору парковки доступна інформація про кількість вільних місць за кожною із доступних адрес (рисунок 4.6).

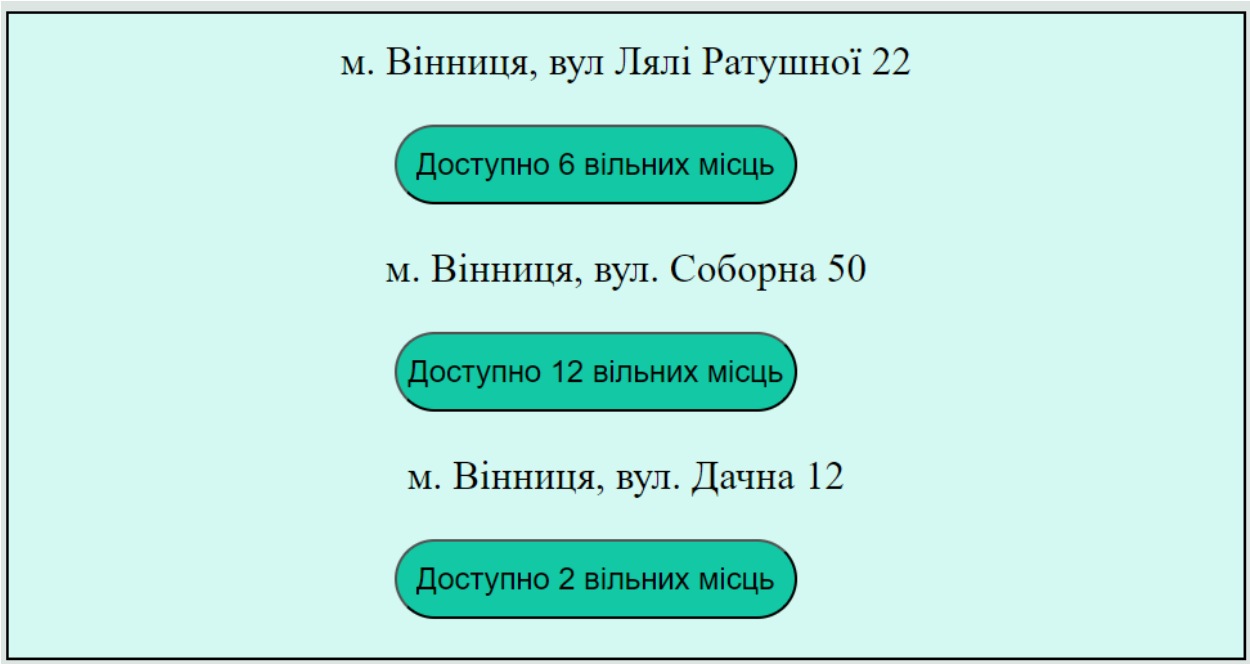

Рисунок 4.6 — Віртуальне табло з кількістю вільних місць на парковці

5 ЕКОНОМІЧНА ЧАСТИНА

5.1 Проведення комерційного та технологічного аудиту науково-технічної розробки

Метою проведення комерційного та технологічного аудиту є оцінювання науково-технічного рівня та рівня комерційного потенціалу розробки, створеної в результаті науково-технічної діяльності.

Магістерська кваліфікаційна робота за темою "Розподілена система з підтримки функціонування автопаркінгу" передбачає вдосконалення системи, яка спрощує та оптимізує управління та ефективність автопаркінгу.

Проведемо оцінювання комерційного та технічного аудиту даної розробки. Для проведення комерційного та технічного аудиту було залучено 3-х незалежних експертів: к.т.н. доц. кафедри ОТ Тарновський Микола Геннадійович, к.т.н. доц. кафедри ОТ Богомолов Сергій Віталійович, к.т.н. доц. кафедри ОТ Савицька Людмила Анатоліївна. Оцінювання комерційного потенціалу буде здійснене за критеріями, що наведені в таблиці 5.1.

Таблиця 5.1 — Критерії оцінювання комерційного потенціалу розробки

Критерії оцінювання та бали (за 5-ти бальною шкалою)					
Кри-терій	0	1	2	3	4
Технічна здійсненність концепції:					
1	Достовірність концепції не підтверджена	Концепція підтверджена експертними висновками	Концепція підтверджена розрахунками	Концепція перевірена на практиці	Перевірено роботоздатність продукту в реальних умовах
Ринкові переваги (недоліки):					
2	Багато аналогів на малому ринку	Мало аналогів на малому ринку	Кілька аналогів на великому ринку	Один аналог на великому ринку	Продукт не має аналогів на великому ринку
3	Ціна продукту значно вища за ціни аналогів	Ціна продукту дещо вища за ціни аналогів	Ціна продукту приблизно дорівнює цінам аналогі	Ціна продукту дещо нижче за ціни аналогів	Ціна продукту значно нижче за ціни аналогів

Продовжння таблиці 5.1

4	Технічні та споживчі властивості продукту значно гірші, ніж в аналогів	Технічні та споживчі властивості продукту трохи гірші, ніж в аналогів	Технічні та споживчі властивості продукту на рівні аналогів	Технічні та споживчі властивості продукту трохи кращі, ніж в аналогів	Технічні та споживчі властивості продукту значно кращі, ніж в аналогів
5	Експлуатаційні витрати значно вищі, ніж в аналогів	Експлуатаційні витрати дещо вищі, ніж в аналогів	Експлуатаційні витрати на рівні експлуатаційних витрат аналогів	Експлуатаційні витрати трохи нижчі, ніж в аналогів	Експлуатаційні витрати значно нижчі, ніж в аналогів
Ринкові перспективи					
6	Ринок малий і не має позитивної динаміки	Ринок малий, але має позитивну динаміку	Середній ринок з позитивною динамікою	Великий стабільний ринок	Великий ринок з позитивною динамікою
7	Активна конкуренція великих компаній на ринку	Активна конкуренція	Помірна конкуренція	Незначна конкуренція	Конкурентів немає
Практична здійсненність					
8	Відсутні фахівці як з технічної, так і з комерційної реалізації ідеї	Необхідно наймати фахівців або витрачати значні кошти та час на навчання наявних фахівців	Необхідне незначне навчання фахівців та збільшення їх штату	Необхідне незначне навчання фахівців	Є фахівці з питань як з технічної, так і з комерційної реалізації ідеї
9	Потрібні значні фінансові ресурси, які відсутні. Джерела фінансування ідеї відсутні	Потрібні незначні фінансові ресурси. Джерела фінансування відсутні	Потрібні значні фінансові ресурси. Джерела фінансування є	Потрібні незначні фінансові ресурси. Джерела фінансування є	Не потребує додаткового фінансування
10	Необхідна розробка нових матеріалів	Потрібні матеріали, що використовуються у військово-промисловому комплексі	Потрібні дорогі матеріали	Потрібні досяжні та дешеві матеріали	Всі матеріали для реалізації ідеї відомі та давно використовуються у виробництві

Продовження таблиці 5.1

11	Термін реалізації ідеї більший за 10 років	Термін реалізації ідеї більший за 5 років. Термін окупності інвестицій більше 10-ти років	Термін реалізації ідеї від 3-х до 5-ти років. Термін окупності інвестицій більше 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій від 3-х до 5-ти років	Термін реалізації ідеї менше 3-х років. Термін окупності інвестицій менше 3-х років
12	Необхідна розробка регламентних документів та отримання великої кількості дозвільних документів на виробництво та реалізацію продукту	Необхідно отримання великої кількості дозвільних документів на виробництво та реалізацію продукту, що вимагає значних коштів та часу	Процедура отримання дозвільних документів для виробництва та реалізації продукту вимагає незначних коштів та часу	Необхідно тільки пові-домлення відповідним органам про виробництво та реалізацію продукту	Відсутні будь-які регламентні обмеження на виробництво та реалізацію продукту

Результати оцінювання комерційного потенціалу експертами розробки зведено в таблицю 5.2.

Таблиця 5.2 — Результати оцінювання комерційного потенціалу розробки

Критерії	Прізвище експерта		
	1 — Тарновський	2 — Богомолов	3 — Савицька
	Бали, виставлені експертами:		
1	3	3	4
Ринкові переваги (недоліки):			
2	2	2	2
3	3	2	2
4	3	3	4
5	4	3	3
Ринкові перспективи			

Продовження таблиці 5.2

6	3	4	3
7	3	4	2
Практична здійсненність			
8	4	4	4
9	1	1	2
10	4	4	4
11	3	4	3
12	2	2	3
Сума балів	$СБ_1$=35	$СБ_2$=36	$СБ_3$=36
Середньоарифметична сума балів $\overline{СБ}$	$\overline{СБ}\frac{\sum_1^3 СБ_i}{3} = \frac{107}{3} = 35.6$		

За даними таблиці 5.2 можна зробити висновок, щодо рівня комерційного потенціалу розробки. Зважимо на результат й порівняємо його з рівнями комерційного потенціалу розробки, що представлено в таблиці 5.3.

Таблиця 5.3 — Науково-технічні рівні та комерційні потенціали розробки

Середньоарифметична сума балів СБ, розрахована на основі висновків експертів	Рівень комерційного потенціалу розробки
0 –10	Низький
11–20	Нижче середнього
21–30	Середній
31–40	Вище середнього
41–48	Високий

Рівень комерційного потенціалу розробки, становить 35,6 балів, що відповідає рівню «вище середнього».

5.2 Розрахунок витрат на здійснення науково-технічної розробки

Витрати на здійснення науково-технічної розробки розраховуються за наступними категоріями: витрати на оплату праці, відрахування на соціальні заходи, матеріали, програмне забезпечення для наукових робіт, накладні (загальновиробничі) витрати та ін. Обрахуємо витрати за кожною категорією.

Основна заробітна плата для розробника-дослідника $З_о$:

$$З_о = \frac{М}{Т_р} \cdot t \text{ [грн]}, \quad (5.1)$$

де М — місячний посадовий оклад , 30000 грн;

$Т_р$ — число робочих днів в місяці, приблизно $Т_р$ = (22) дні;

t — число робочих днів роботи розробника-дослідника , 90.

$$З_о = \frac{30000}{22} \cdot 90 = 123000 \text{ (грн)}.$$

Результати розрахунків зведемо до таблиці 5.6.

Таблиця 5.6 — Основна заробітна плата розробників

Найменування посади	Місячний посадовий оклад, грн.	Оплата за робочий день, грн.	Число днів роботи	Витрати на заробітну плату, грн.
Розробник	30000,00	1365,00	90	123000,00
Керівник роботи	15000,00	681,81	3	2045,40
Всього				125045,40

Додаткова заробітна плата $З_д$ розробників розраховується як 10% від основної заробітної плати:

$$З_д = 0{,}10 \cdot 125045{,}40 = 12504{,}54 \text{ (грн)}.$$

Нарахування на заробітну плату $Н_{зп}$ розробника становить:

$$H_{зп} = (З_о + З_д) \cdot \frac{\beta}{100} \text{ [грн]}, \quad (5.2)$$

де $З_о$ — основна заробітна плата розробника;

$З_д$ — додаткова заробітна плата розробника;

β — ставка єдиного внеску на загальнообов'язкове державне соціальне страхування — 22%.

$$Н_{зп} = (125045,40 + 12504,54) \cdot 0,22 = 30260,98 \text{ (грн)}.$$

Амортизація обладнання, комп'ютерів та приміщень, які використовувались під час виконання даного етапу роботи. Дані відрахування розраховують по кожному виду обладнання, приміщенням тощо.

Амортизація обладнання та приміщення, яке використовувалось для проведення розробки, розраховується за формулою:

$$A = \frac{Ц \cdot Н_а}{100} \cdot \frac{Т}{12} \text{ [грн]}, \quad (5.3)$$

де Ц — балансова вартість обладнання, грн.;

$Н_а$ — річна норма амортизаційних відрахувань;

Т — термін використання під час розробки, місяців.

Розрахуємо амортизаційні витрати для комп'ютера, балансова вартість якого становить 40000 грн, а термін використання — 3 місяці:

Норма амортизації розраховується за формулою:

$$Н_а = \frac{В_п - В_л}{В_п \cdot Т_{кв}} \cdot 100 \text{ [грн]}, \quad (5.4)$$

де $В_п$ і $В_л$ — відповідно первісна та ліквідаційна вартість основних фондів;

$Т_{кв}$ — строк корисного використання.

$$H_a = \frac{40000-2000}{40000 \cdot 5} \cdot 100 = 19\% \text{ (грн).}$$

$$A = \frac{40000 \cdot 19}{100} \cdot \frac{3}{12} = 1900 \text{ (грн).}$$

Таблиця 5.6 — Амортизаційні відрахування

Найменування	Балансова вартість, грн	Термін використання, р	Фактична трив. використання, міс.	Величина амортизаційних відрахувань, грн
Офісне приміщення	100000	25	3	4500,00
Комп'ютер	40000	5	3	1900,00
Всього				6400,00

Під час розробки програмного продукту використовувались лише безкоштовні програмні засоби.

Витрати на енергію визначаються на основі витрат на одиницю продукції та тарифів на енергію за допомогою формули:

$$B_e = B \cdot П \cdot Ф \cdot K_п \ [грн], \quad (5.5)$$

де В — вартість 1кВт електроенергії;

П — установлена потужність обладнання, кВт;

Ф — фактична кількість годин роботи комп'ютера при створенні програмного продукту, годин;

$К_п$ — коефіцієнт використання потужності.

Отже, витрати на енергію становлять:

$$В_е = 6.4 \cdot 0{,}6 \cdot 720 \cdot 0{,}6 = 1658.88 \text{ (грн).}$$

Витрати за доступ до Інтернет можна розрухувати за формулою:

$$B_{\partial i} = Ц_{\partial i} \cdot T\,[грн], \tag{5.6}$$

де $Ц_{ді}$ — це ціна доступу за місяць;

Т — кількість місяців використання доступу до мережі.

$$В_{ді} = 375 \cdot 3 = 1125{,}00\ (грн).$$

Обчислимо витрати на виконання даної роботи, що являтиме собою суму всіх попередніх витрат.

В результаті сума усіх витрат, що вказані вище дає витрати на виконання даного етапу роботи В:

$$В = З_о + З_д + Н_{зп} + А + В_е + В_{ді}\ [грн],$$

В=125045,40+12504,54+30260,98+6400+1658,88+1125,00=177000,8(грн)

Прогнозування загальних витрат ЗВ на виконання та впровадження результатів виконаної роботи здійснюється за формулою:

$$ЗВ = \frac{В_{заг}}{\beta}\ [грн], \tag{5.8}$$

де β — коефіцієнт, який характеризує етап (стадію) виконання даної роботи.

Так, якщо розробка знаходиться:

— на стадії науково-дослідних робіт, то $\beta \approx 0{,}1$;

— на стадії технічного проектування, то $\beta \approx 0{,}2$;

— на стадії розробки конструкторської документації, то $\beta \approx 0{,}3$;

— на стадії розробки технологій, то $\beta \approx 0{,}4$;

— на стадії розробки дослідного зразка, то $\beta \approx 0{,}5$;

— на стадії розробки промислового зразка, $\beta \approx 0{,}7$;

— на стадії впровадження, то $\beta \approx 0{,}9$.

Отже, підставимо дані в формулу й отримаємо результат:

$$ЗВ = \frac{177000{,}8}{0{,}9} = 196\ 668{,}55 \text{ (грн)}.$$

Витрати на виконання наукової роботи та впровадження її результатів становитиме 196 668,55 грн.

5.3 Розрахунок економічної ефективності та обґрунтування економічної доцільності комерціалізації науково-технічної розробки.

Узагальнюючим позитивним результатом, що отримує інвестор від впровадження результатів тієї чи іншої розробки, є збільшення чистого прибутку. Виконання даної наукової роботи та впровадження її результатів складає трохи більше одного року. Позитивні результати від впровадження розробки очікуються вже в перші місяці після впровадження.

Збільшення чистого прибутку у потенційного інвестора протягом декількох років розраховується за формулою:

$$\Delta П_i = (\Delta Ц_o \cdot N + Ц_o \cdot \Delta N)_i \cdot \lambda \cdot \rho \cdot (1 - \frac{\vartheta}{100}) \qquad (5.9)$$

де $\Delta Ц_o$ — зміна основного якісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

N — основний кількісний показник, який визначає величину попиту на аналогічні розробки у році до впровадження результатів нової науково-технічної розробки;

$Ц_o$ — основний якісний показник, який визначає ціну реалізації нової науково-технічної розробки в аналізованому році, $Ц_o = Ц_б \pm \Delta Ц_o$;

$Ц_б$ — основний якісний показник, який визначає ціну реалізації існуючої науково-технічної розробки у році до впровадження результатів;

ΔN — зміна основного кількісного показника від впровадження результатів науково-технічної розробки в аналізованому році;

λ — коефіцієнт, який враховує сплату потенційним інвестором податку на додану вартість. λ =0,8333;

ρ — коефіцієнт, який враховує рентабельність інноваційного продукту. ρ = 0,3;

ϑ — ставка податку на прибуток, який має сплачувати потенційний інвестор, у ϑ = 18%.

В результаті впровадження наукової розробки покращується якість певного продукту, що дозволяє підвищити ціну його реалізації на 1000 грн. Кількість користувачів збільшиться протягом першого року на 500, другого — 1000, третього — 3000. Реалізація продукції до впровадження результатів наукової розробки складала 1000 користувачів, а її ціна — 10000 грн.

Розрахуємо показник прибутку впродовж трьох років відносно базового.

$$\Delta П_1 = 10000 \cdot 1 + (10000 + 1000) \cdot 500) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 1125030,8(\text{грн}).$$

$$\Delta П_1 = (10000 \cdot 1 + (10000 + 1000) \cdot 1000) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 2248020{,}8\text{грн}).$$

$$\Delta П_1 = (10000 \cdot 1 + (10000 + 1000) \cdot 3000) \cdot 0{,}8333 \cdot 0{,}3 \cdot (1 - 18/100) = 6740000{,}8(\text{грн}).$$

Далі розрахуємо приведену вартість збільшення чистих прибутків, що їх може отримати потенційний інвестор від впровадження розробки за формулою:

$$ПП = \sum_{i=1}^{T} \frac{\Delta П_i}{(1+\tau)^t} \qquad (5.10)$$

де Т — період часу, протягом якого очікується отримання позитивних результатів від впровадження науково-технічної розробки, роки;

τ — ставка дисконтування, за яку можна взяти щорічний прогнозований рівень інфляції в країні, τ =0,05…0,15;

t — період часу (в роках) від моменту початку впровадження науково-технічної розробки до моменту отримання потенційним інвестором додаткових чистих прибутків у цьому році.

$$ПП = \frac{1125030{,}8}{(1+0{,}05)^1} + \frac{2248020{,}8}{(1+0{,}05)^2} + \frac{6740000{,}8}{(1+0{,}05)^3} = 6\ 133\ 310\ (грн).$$

Отже, розрахунки показують, що комерційний ефект від впровадження розробки виражається у значному збільшенні чистого прибутку підприємства.

Основними показниками, які визначають доцільність фінансування наукової розробки інвестором, є абсолютна і відносна ефективність вкладених інвестицій та термін їх окупності.

Якщо $E_{абс} > 0$, то результат від проведення наукових досліджень та їх впровадження принесе прибуток, але це також ще не свідчить про те, що інвестор буде зацікавлений у фінансуванні даного проекту (роботи).

Розрахуємо абсолютну ефективність інвестицій, вкладених у реалізацію проекту. Ставка дисконтування τ дорівнює 0,1. Отримаємо:

Розраховуємо величину початкових інвестицій PV, які розробник (замовник) має вкласти для здійснення науково-технічної розробки.

Для цього можна використати формулу:

$$PV = k_{розр} \cdot ЗВ\ [грн]\ , \qquad (5.11)$$

де розр *k* — коефіцієнт, що враховує витрати розробника (замовника) на впровадження науково-технічної розробки, це можуть бути витрати на підготовку приміщень, розробку технологій, навчання персоналу, маркетингові заходи тощо; зазвичай розр k =2...5, але може бути і більшим;

ЗВ — загальні витрати на проведення науково-технічної розробки та оформлення її результатів, грн.

$$PV = 4 \cdot 196\,668{,}55 = 789\,674{,}65 \text{ (грн)}.$$

Розраховуємо абсолютну ефективність вкладених інвестицій $Е_{абс}$ за формулою:

$$Е_{абс} = (ПП - PV) \text{ [грн]}, \qquad (5.12)$$

де ПП — приведена вартість всіх чистих прибутків, що їх отримає підприємство (організація) від реалізації результатів наукової розробки, грн.;

PV — теперішня вартість інвестицій PV = ЗВ, грн.

$$Е_{абс} = 6\,133\,310 - 789\,674{,}65{,}65 = 5\,343\,635{,}35 \text{ (грн)}.$$

Оскільки $Е_{абс} > 0$, то вкладання коштів на виконання та впровадження результатів розробки є доцільним.

Розраховуємо відносну (щорічну) ефективність вкладених в наукову розробку інвестицій $Е_в$. Для цього використовуємо формулу:

$$Е_в = \sqrt[Т_ж]{1 + \frac{Е_{абс}}{PV}} - 1 \qquad (5.13)$$

де $Е_{абс}$ — абсолютна ефективність вкладених інвестицій, грн;

PV — теперішня вартість інвестицій PV = ЗВ, грн;

$Т_ж$ — життєвий цикл наукової розробки, роки.

Тоді відносна ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів складе:

$$E_в = \sqrt[3]{1 + \frac{5\,343\,635\,,35}{789\,674{,}65}} - 1 = 0{,}98$$

Далі, розраховану величину $E_в$ порівнюємо з мінімальною (бар'єрною) ставкою дисконтування τ мін, яка визначає ту мінімальну дохідність, нижче за яку інвестиції вкладатися не будуть.

У загальному вигляді мінімальна (бар'єрна) ставка дисконтування τ мін визначається за формулою:

$$\tau = \text{d} + \text{f}, \qquad (5.14)$$

де d — середньозважена ставка за депозитними операціями в комерційних банках; d = 0,09;

f — показник, що характеризує ризикованість вкладень; зазвичай, величина f = 0,05.

$$\tau \text{ мін} = 0{,}09 + 0{,}05 = 0{,}14.$$

Оскільки $E_в$ = 98% > τ мін = 14%, то у інвестора буде зацікавленість вкладати гроші в дану наукову розробку, оскільки значно більші прибутки він отримає від того, що інвестує кошти розробку, а не розмістить гроші на депозиті у комерційному банку.

Розраховуємо термін окупності вкладених у реалізацію наукового проекту інвестицій за формулою:

$$T_{ок} = \frac{1}{E_в}. \qquad (5.15)$$

Для нашої розробки термін окупності вкладених у реалізацію проекту інвестицій $Т_{ок}$ складе:

$$T_{ок} = \frac{1}{0{,}98} = 1{,}02\ (року),$$

Термін окупності складає 1,02 року, що свідчить про доцільність фінансування даної наукової розробки.

5.4 Результати економічного аналізу

В даному розділі було проведено комерційний та технологічний аудит розробки із залученням трьох незалежних експертів. Оцінка комерційного потенціалу показала, що рівень комерційного потенціалу розробки є вище середнього.

Згідно із розрахунками всіх статей витрат на виконання науково-дослідної, дослідно-конструкторської та конструкторської-технологічної роботи загальні витрати на розробку складають 196 668,55 грн.

Розрахована абсолютна ефективність вкладених інвестицій в сумі 6 133 310 грн свідчить про отримання прибутку інвестором від комерціалізації програмного продукту.

Відносна ефективність вкладних інвестицій в проведення наукових досліджень та впровадження їх результатів становить 98%, що вище за мінімальну бар'єрну ставку дисконтування, яка складає 14%. Це означає потенційну зацікавленість інвесторів у фінансуванні розробки.

Термін окупності вкладених у реалізацію проекту інвестицій становить 1,02 року, що також свідчить про доцільність фінансування нової розробки.

ВИСНОВКИ

У ході виконання магістерської роботи було розроблено систему з підтримки функціонування автопаркінгу, що спрощує водієві користування парковкою.

У першому розділі було проаналізовано технології, що спрямовані на підтримку інформаційного обслуговування автопаркінгу. Було описано сучасні технології та проаналізовано відомі аналоги сучасних електронних систем.

У другому розділі було запропоновано архітектуру розроблювальної системи, обгрунтовано алгоритм з надання послуг з паркування, проаналізовано та обрано технології для розробки клієтської та серверної частини, зокремо React та Node. Js.

У третьому розділі було розроблено структурну схему системи та мобільний веб-застосунок, спроектовано інтерфейс та функціонал системи, розроблено базу даних, у яку записуються дані з датчиків фіксування автомобіля та дані користувачів.

У четвертому розділі було проведено тестування користувацього інтерфейсу веб-застосунку, і зроблено висновок, що все працює коректно.

У п’ятому розділі було виконано економічні розрахунки, в результаті яких зроблено висновок, що потенційна зацікавленість інвесторів у фінансуванні розробки є досить високою.

Отже, у результаті виконання магістерської кваліфікаційної роботи було отримано працездатний веб-застосунок, з готовою робочою базою даних. Було розроблено алгоритм роботи системи, створено структурну схему та обрано апаратну частину завдяки якої можна реалізувати її у житті.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Електронні інформаційні ресурси: створення, використання, доступ. Збірник матеріалів Міжнародної науково-практичної Інтернет конференції 20-21 листопада 2023 р. — Суми/Вінниця: НІКО/ КЗВО «Вінницька академія безперервної освіти», 2023. — 336 с.

2. Кривенька В. О., Тарновський М.Г. «Веб-додаток інформаційного обслуговування автопаркінгу» . Матеріали конференції «LI Науково-технічна конференція підрозділів Вінницького національного технічного університету (2022)», Вінниця, 2022. [Електронний ресурс]. Режим доступу:https://conferences.vntu.edu.ua/index.php/all-fitki/all-fitki-2022/paper/view/14872/12567 Дата звернення: Черв. 2022

3. Архітектура управління паркуванням у розумних містах [Електронний ресурс]. Режим доступу: https://www.researchgate.net/publication/259264362_Architecture_for_parking_management_in_smart_cities

4. Різних типів парковок у містах [Електронний ресурс]. Режим доступу: https://thearchspace.com/9-distinct-types-of-parking/

5. Автоматині багаторівневі парковки [Електронний ресурс]. Режим доступу: https://www.igb-parkings.com/en/

6. Ягузинська И. Ю., Типушова И. О. Сучасні автоматизовані системи паркування автомобілів // Науково-методичний електронний журнал “Концепт” — 2020. — Т. 35. — С. 156–160. — URL: http://e-koncept.ua/2015/95585.htm.

7. Розумна парковка та її переваги [Електронний ресурс]. Режим доступу: https://ula.lantec.ua/statti/rozumna-parkovka-ta-jiji-perevagi

8. Інновації у парковках: які технології можуть допомогти міській інфраструктурі? [Електронний ресурс]. Режим доступу: https://www.forbes.ua/tehnologii/340561-innovacii-v-parkovkah-kakie-tehnologii-mogut-pomoch-gorodskoy-infrastrukture.

9. Сучасні паркувальні системи. [Електронний ресурс]. Режим доступу: https://dentauto.ua/raznoe/sovremennye-parkovochnye-sistemy.html.

10. What are dedicated servers and how are they used on the internet? [Електронний ресурс]. Режим доступу: https://www.namecheap.com /support/knowledgebase/article.aspx/10596/2188/how-does-a-dedicated server-work/

11. Cloud server [Електронний ресурс]. Режим доступу: https://www.techtarget.com/searchcloudcomputing/definition/cloud-server

12. IoT і технологія LoRaWAN [Електронний ресурс]. Режим доступу: https://iotji.io/osoblyvosti-lorawan/

13. Mokolora [Електронний ресурс]. Режим доступу: https://www.mokolora.com/ua/lorawan-parking-sensor-lw009-ig-sm/

14. Паркувальне обладнання [Електронний ресурс]. Режим доступу: https://www.sea.com.ua/ua/parkovochnoe_oborudovanie/informacionnoe-tablo-parkovok/

15. Методичні вказівки до виконання магістерських кваліфікаційних робіт студентами спеціальності 123 «Комп'ютерна інженерія» другого (магістерського) рівня вищої освіти денної та заочної форм навчання. / Укладачі О.Д. Азаров, О.В. Дудник, С.І. Швець − Вінниця : ВНТУ, 2023. − 57 с.

16. Методичні вказівки до виконання економічної частини магістерських кваліфікаційних робіт / Уклад. : В. О. Козловський, О. Й. Лесько, В. В. Кавецький. — Вінниця : ВНТУ, 2021. 42 с.

17. Поняття клієнт-серверних систем [Електронний ресурс]. – Режим доступу: http://bourabai.kz/dbt/client1.html/

18. Barth W. Nagios, 2nd Edition: System and Network Monitoring / Barth W. – London: No Starch Press, 2015. – 719 p. – ISBN - 978-1593271794.

19. What is client-server architecture / Що таке клієнт-серверна архітектура [Електронний ресурс]. – Режим доступу:

http://apachebooster.com/kb/what-is-clientserver-architecture-and-what-are-its-types/

20. Ajit K. Sencha MVC Architecture / Ajit K. – Birmingham: Packt Publishing 2022.-126 p. – ISBN 978-1849518888.

21. Mobile-parking[Електронний ресурс]. – Режим доступу: https://ktps.kyiv.ua/services/pay/mobileparking

22. Розробка веб-застосунків [Електронний ресурс] — режим доступу:https://webstudio2u.net/ua/site-develop/641-razrabotka-veb prilozheniy.html.

23. Кучай А.В., Білан С.О. «Основи HTML» //К: 2018. № 12.С. 20-25.

24. CSS для початківців [Електронний ресурс] – режим доступу: https://www.tutorialspoint.com/css/css_measurement_units.htm

25. What is JavaScript [Електронний ресурс]. Режим доступу: https://www.tutorialsteacher.com/javascript/what-is-javascript

26. Virtual DOM and Internals [електронний ресурс] // Режим доступу: https://reactjs.org/docs/faq-internals.html

27. Webpack Concepts [електронний ресурс] // Режим доступу: https://webpack.js.org/concepts/

ДОДАТОК А

Технічне завдання

Міністерство освіти і науки України

Вінницький національний технічний університет

Факультет інформаційних технологій та комп'ютерної інженерії

Кафедра обчислювальної техніки

ЗАТВЕРДЖУЮ

Завідувач кафедри ОТ

проф., д.т.н.. Азаров О.Д..

"29" вересня 2023 р.

ТЕХНІЧНЕ ЗАВДАННЯ

на виконання магістерської кваліфікаційної роботи

"Розподілена система з підтримки функціонування автопаркінгу"
08-54.МКР.031.00.000 ПЗ

Науковий керівник: доцент
к.т.н. каф.ОТ

_____ Тарновський М. Г.

Студентка групи 2КІ-22м

_____ Кривенька В. О.

ВНТУ 2023

1 Підстава для виконання магістерської кваліфікаційної роботи (МКР)

1.1 Актуальність роботи полягає у інформаційному обслуговуванні водіїв, що бажають припаркувати своє авто та сприяє зменшенні викидів шляхом скорочення часу, проведеного водіями на пошук парковочних місць і внаслідок ефективнішого управління рухом транспорту.

1.2 Наказ про затвердження теми МКР.

2 Мета МКР і призначення розробки

2.1 Мета роботи — розширення функціональних можливостей системи з підтримки автопаркінгу за рахунок впровадження хмарних технологій.

2.2 Призначення розробки — визначається необхідністю створення ефективної системи, що буде підтримувати функціонування автоматизованої парковки та веб-застосунку який буде інформувати про кількість вільних місць на парковці.

3 Вихідні дані для виконання МКР

3.1 Проведення аналізу існуючих методів та принципів.

3.2 Розробка алгоритму системи та веб-застосунку.

3.4 Проведення верифікації та аналізу отриманих результатів.

3.5 Виконання розрахунків для доведення доцільності нової розробки з економічної точки зору.

4 Вимоги до виконання МКР

Головна вимога — що система має надавати точну інформацію про кількість наявних парковочних місць для полегшення процесу пошуку та забезпечення ефективного використання паркінгу.

5 Етапи МКР та очікувані результати

Етапи роботи та очікувані результати приведено в Таблиці А.1.

Таблиця А.1 — Етапи МКР

№ етапу	Назва етапу	Термін виконання		Очікувані результати
		початок	кінець	
1	Аналіз існуючих технологій, огляд аналогів системи.	19.09.2023	28.09.2023	Розділ 1
2	Визначення архітектури розподіленої системи	5.10.2023	15.10.2023	Розділ 2
3	Розробка алгоритму та функціоналу веб-застосунку	16.10.2023	23.10.2023	Розділ 3
4	Тестування системи	23.10.2023	26.10.2023	Розділ 4
4	Підготовка економічної частини	2.11.2023	12.11.2023	Розділ 5
6	Оформлення пояснювальної записки, графічного матеріалу і презентації	4.12.2023	8.12.2023	ПЗ, графічний матеріал і презентація
7	Підготовка і підпис супроводжуючих документів, нормоконтроль та тест на плагіат	8.12.2023	11.12.2023	Оформленні документи

6 Матеріали, що подаються до захисту МКР

До захисту подаються: пояснювальна записка МКР, графічні і ілюстративні матеріали, протокол попереднього захисту МКР на кафедрі, відгук наукового керівника, відгук опонента, протоколи складання державних екзаменів, анотації до МКР українською та іноземною мовами.

7 Порядок контролю виконання та захисту МКР

Виконання етапів графічної та розрахункової документації МКР контролюється науковим керівником згідно зі встановленими термінами. Захист МКР відбувається на засіданні Екзаменаційної комісії, затвердженої наказом ректора.

8 Вимоги до оформлювання та порядок виконання МКР

8.1 При оформлюванні МКР використовуються:

— ДСТУ 3008: 2015 «Звіти в сфері науки і техніки. Структура та правила оформлювання»;

— ДСТУ 8302: 2015 «Бібліографічні посилання. Загальні положення та правила складання»;

— ГОСТ 2.104–2006 «Єдина система конструкторської документації. Основні написи»;

— методичні вказівки до виконання магістерських кваліфікаційних робіт зі спеціальності 123 — «Комп’ютерна інженерія»;

— документи на які посилаються у вище вказаних.

8.2 Порядок виконання МКР викладено в «Положення про кваліфікаційні роботи на другому (магістерському) рівні вищої освіти СУЯ ВНТУ–03.02.02 П.001.01:21

ДОДАТОК Б

Архітектура розподіленої системи з підтримки функціонування автопаркінгу з використанням мережі LoRaWAN

Рисунок Б.1 — Архітектура розподіленої системи з підтримки функціонування автопаркінгу з використанням мережі LoRaWAN

ДОДАТОК В

Структурна схема апаратно-програмної частини

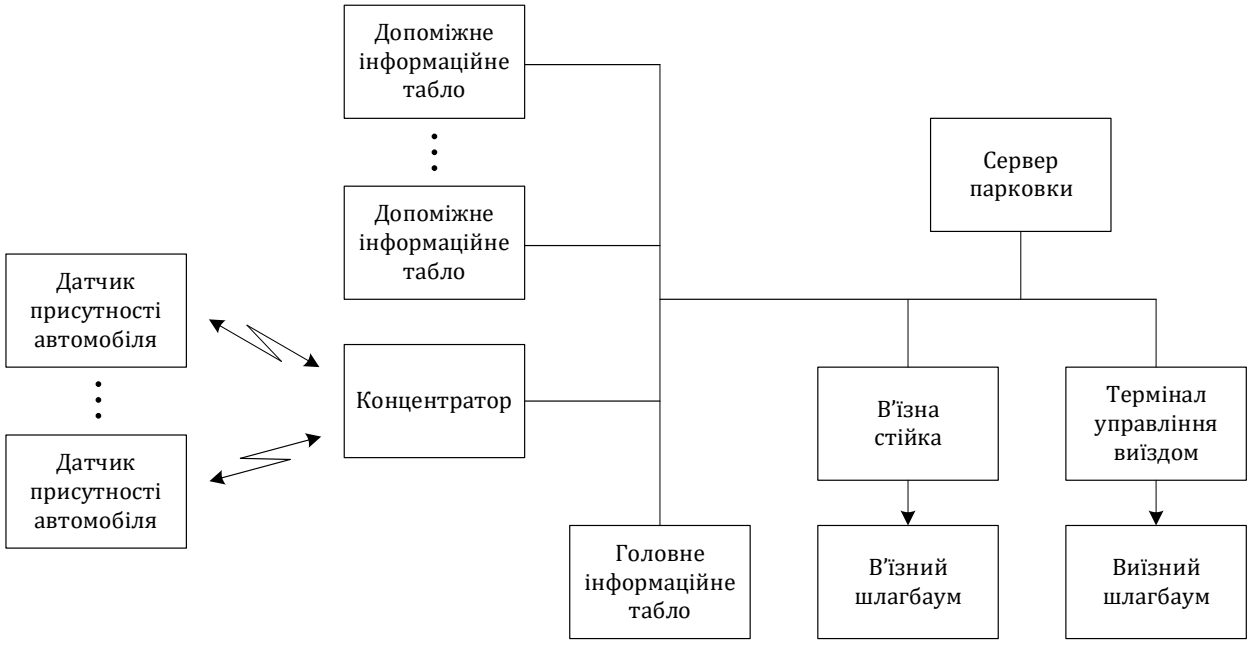

Рисунок В.1 — Схема апаратно-програмної частини

ДОДАТОК Г

ER-діаграма бази даних

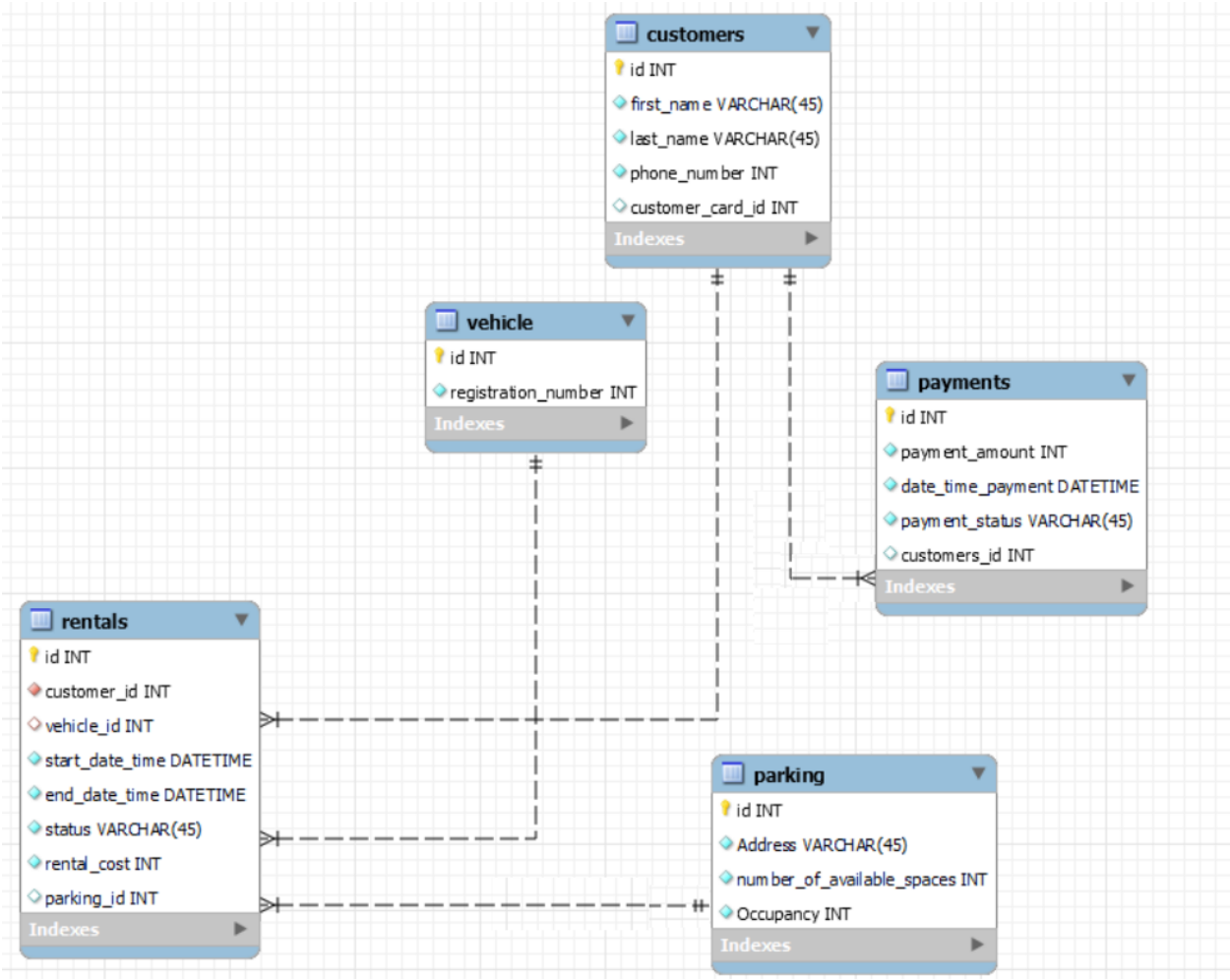

Рисунок Г.1 — ER-діаграма бази даних

ДОДАТОК Д

Блок-схема алгоритму застосунку

Рисунок Д.1 — Блок-схема алгоритму

ДОДАТОК Е

Лістинг для клієнтської частини

```
// ParkingRegistration.js

const ParkingRegistration = () => {
  const [selectedSpace, setSelectedSpace] = useState('');
  const handleSpaceSelection = (spaceId) => {
    setSelectedSpace(spaceId);
  };
  const handleRegistration = () => {
    console.log(`Ви зареєстровані на паркомісце #${selectedSpace}`);
  };
  return (
    <div>
      <h2>Реєстрація на паркомісце</h2>
      <p>Оберіть вільне паркомісце:</p>
      <button onClick={() => handleSpaceSelection(1)}>Паркомісце #1</button>
      <button onClick={() => handleSpaceSelection(2)}>Паркомісце #2</button>
      <button onClick={() => handleSpaceSelection(3)}>Паркомісце #3</button>

      {selectedSpace && (
        <div>
          <p>{`Ви обрали паркомісце #${selectedSpace}`}</p>
          <button onClick={handleRegistration}>Зареєструватися</button>
        </div>
      )}
    </div>
  );
};
export default ParkingRegistration;
```

ДОДАТОК Ж

Лістинг для серверної частини

```
connection.connect((err) => {
   if (err) {
      console.error('Помилка підключення до бази даних:', err);
   } else {
      console.log('Підключено до бази даних MySQL!');

      connection.query('SELECT * FROM ваша_таблиця', (error, results, fields)
=> {
         if (error) {
            console.error('Помилка виконання запиту:', error);
         } else {
            console.log('Результати запиту:', results);
         }
      });
      const новий_запис = { колонка1: 'значення1', колонка2: 'значення2' };
      connection.query('INSERT INTO ваша_таблиця SET ?', новий_запис,
(error, results, fields) => {
         if (error) {
            console.error('Помилка вставки запису:', error);
         } else {
            console.log('Новий запис вставлено, ID:', results.insertId);
         }
      });
      connection.query('UPDATE ваша_таблиця SET колонка1 = ? WHERE
умова', ['нове_значення'], (error, results, fields) => {
         if (error) {
            console.error('Помилка оновлення запису:', error);
```

```
        } else {
            console.log('Запис оновлено');
        }
    });
    connection.query('DELETE FROM ваша_таблиця WHERE умова', (error,
results, fields) => {
        if (error) {
            console.error('Помилка видалення запису:', error);
        } else {
            console.log('Запис видалено');
        }
    });

    connection.end((err) => {
        if (err) {
            console.error('Помилка закриття з\'єднання:', err);
        } else {
            console.log('З\'єднання закрито.');
        }
    });
  }
});
```

ДОДАТОК И

Протокол перевірки кваліфікаційної роботи на наявність текстових запозичень

Назва роботи: Розподілена система з підтримки функціонування автопаркінгу.

Тип роботи: магістерська дипломна робота

Підрозділ кафедра обчислювальної техніки

Показники звіту подібності Unicheck

Оригінальність 96,7% Схожість 3,3%

Аналіз звіту подібності (відмітити потрібне):

- ✓ Запозичення, виявлення 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 виконання автором. Роботу направити на розгляд експертної комісії кафедри.
- ☐ 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.

Особа, відповідальна за перевірку _____ Захарченко С.М.
(підпис) (прізвище, ініціали)

Ознайомлені з повним звітом подібності, який був згенерований системою Unicheck щодо роботи.

Автор роботи _____ Кривенька В. О.

Керівник роботи _____ Тарновський М. Г.