

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет «Запорізька політехніка»

Факультет комп'ютерних наук і технологій

(повне найменування факультету)

Кафедра програмних засобів

(повне найменування кафедри)

Пояснювальна записка

до дипломного проєкту (роботи)

магістр

(ступінь вищої освіти)

на тему ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
УПРАВЛІННЯ ПОСЛУГАМИ ПАРКІНГ КОМПЛЕКСІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF METHODS
FOR MANAGING PARKING COMPLEX SERVICES

Виконав(ла): студент(ка) 2 курсу, групи КНТ-222м

Спеціальності 122 Комп'ютерні науки

(код і найменування спеціальності)

Освітня програма (спеціалізація)

Системи штучного інтелекту

КОЦУР М.І.

(ПРИЗВИЩЕ та ініціали)

Керівник СУББОТІН С.О.

(ПРИЗВИЩЕ та ініціали)

Рецензент ПОЛЯКОВ М.О.

(ПРИЗВИЩЕ та ініціали)

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

Факультет комп'ютерних наук і технологій
Кафедра програмних засобів
Ступінь вищої освіти магістр
Спеціальність 122 Комп'ютерні науки
(код і найменування)
Освітня програма (спеціалізація) Системи штучного інтелекту
(назва освітньої програми (спеціалізації))

ЗАТВЕРДЖУЮ

Завідувач кафедри ПЗ, д.т.н, проф.
Сергій СУББОТІН
“ ” 2023 року

З А В Д А Н Н Я

НА ДИПЛОМНИЙ ПРОЄКТ (РОБОТУ) СТУДЕНТА(КИ)

КОЦУРА Михайла Ігоровича

(ПРИЗВИЩЕ, ім'я, по батькові)

1. Тема проєкту (роботи) Дослідження та програмна реалізація методів управління послугами паркінг комплексів. Research and Software Implementation of Methods for Managing Parking Complex Services

керівник проєкту (роботи) д.т.н., проф., СУББОТІН Сергій Олександрович,
(науковий ступінь, вчене звання, ПРИЗВИЩЕ, ім'я, по батькові)

затверджені наказом закладу вищої освіти від “14” листопада 2023 року №437

2. Строк подання студентом проєкту (роботи) 01 грудня 2023 року

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

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) 1. Аналіз проблеми та постановка завдань дослідження. 2. Матеріали і методи. 3. Проєктування програмного забезпечення. 4. Головні рішення щодо впровадження компонентів системи. 5. Експлуатація, тестування та експериментальне дослідження програмного забезпечення.

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

6. Консультанти розділів проєкту (роботи)

Розділ	ПРИЗВИЩЕ, ініціали та посада консультанта	Підпис, дата	
		завдання видав	прийняв виконане завдання
1-5 Основна частина	СУББОТІН С.Ю., зав. каф.		
Нормоконтролер	ЛИПОВЕЦЬ М.В., асистент		

7. Дата видачі завдання “08” вересня 2023 року.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проєкту (роботи)	Строк виконання етапів проєкту (роботи)	Примітка
1	Аналіз проблеми та постановка завдань дослідження	1 тиждень	Завдання, ТЗ
2	Матеріали і методи	2-5 тижні	Розділи 1, 2
3	Проектування програмного забезпечення	6 тиждень	Розділ 3
4	Головні рішення щодо впровадження компонентів системи	7-9 тижні	Розділ 4
5	Експлуатація, тестування та експериментальне дослідження програмного забезпечення	10-11 тижні	Розділ 5
6	Оформлення пояснювальної записки та документів до неї. Нормоконтроль та рецензування	12 тиждень	Додатки
7	Захист роботи	12 тиждень	

Студент(ка)

Підпис
(підпис)

Михайло КОЦУР
(Імя ПРИЗВИЩЕ)

Керівник проєкту (роботи)

Підпис
(підпис)

Сергій СУББОТІН
(Імя ПРИЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до дипломної кваліфікаційної роботи магістра:
110 с., 3 табл., 28 рис., 3 дод., 16 джерел.

ПАРКІНГ КОМПЛЕКС, ПАРКУВАЛЬНІ МІСЦЯ, ДОСТУПНІСТЬ
ВІЛЬНИХ МІСЦЬ, ІНФОРМУВАННЯ ВОДІЇВ, WEBFRAMEWORK
XGBOOST.

Об'єкт роботи – процес управління послугами паркінг комплексів.
Предмет роботи – методи управління та прогнозування доступністю вільних
місць паркінг комплексів.

Мета роботи – розробити програмне забезпечення управління
розподілу паркувальних місць для передчасного інформування водіїв про
вільні місця в паркінг комплексу на певний період часу.

Матеріали, методи та технічні засоби: мова Python, бібліотеки xgboost,
pandas, framework веброзробки Django, система роботи з базами даних
PostgreSQL.

Результати. Наукова новизна цієї роботи полягає в застосуванні
методу управління розподілу паркувальних місць для передчасного
інформування водіїв. Використовуючи алгоритм машинного навчання
XGBoost, система прогнозує мінімальну кількість вільних паркувальних
місць протягом зазначеного часу. Цей підхід дозволяє водіям обґрунтовано
вирішувати, чи потрібно заздалегідь забронювати місця на паркінг
комплексів.

Висновки. Розроблене програмне забезпечення, яке є результатом
даної роботи, включає в себе інструменти для управління даними про вільні
паркувальні місця на майданчиках паркінг комплексів, можливість
заздалегідь бронювати вільні місця та отримувати інформацію з форми
вебдодатку.

Галузь використання – міська та міжміська автомобільна
інфраструктура.

ABSTRACT

Explanatory note to the diploma qualifying work of the master: 110 pages, 3 tables, 28 figures, 3 appendixes, 16 sources.

PARKING COMPLEX, PARKING SPACES, AVAILABILITY OF FREE SPACES, DRIVER INFORMATION, WEB FRAMEWORK, XGBOOST.

The object of the study is the process of managing services in parking complexes.

The subject of the study is the methods of managing and forecasting the availability of free parking spaces in parking complexes.

The goal of the study is to develop software for managing the address of parking spaces to proactively inform drivers about the availability of free spaces in a parking complex for a specific period of time.

Materials, methods and technical means: Python language, xgboost, pandas libraries, Django web development framework, PostgreSQL database management system.

Results: The scientific novelty of this work lies in the application of a method for managing the address of parking spaces to proactively inform drivers. Utilizing the XGBoost machine learning algorithm, the system predicts the minimum number of available parking spaces during the specified period. This approach enables drivers to make informed decisions about whether it is necessary to pre-book parking spaces in parking complexes.

Conclusions: The developed software, which is the result of this work, includes tools for managing data on available parking spaces in parking complex blocks, the ability to pre-book vacant spaces, and obtaining information through a web application interface.

Scope of Application: Urban and interurban automotive infrastructure.

ЗМІСТ

	С.
Перелік скорочень та умовних познач.....	8
Вступ.....	9
1 Аналіз проблеми та постановка завдань дослідження	11
1.1 Аналіз проблем надання послуг паркінг комплексів.....	11
1.2 Огляд сучасного програмного забезпечення для пошуку паркінг комплексів	13
1.3 Висновки за розділом 1	16
2 Матеріали і методи.....	17
2.1 Задача прогнозування вільних місць для паркування автотранспорту	17
2.2 Існуючі методи управління та прогнозування доступністю вільних місць для паркування	17
2.3 Алгоритм XGBoost	19
2.4 Метод прогнозування наявності вільних місць паркінг комплексу	21
2.5 Висновки за розділом 2	23
3 Проектування програмного забезпечення	24
3.1 Створення рішень при розробці програмного забезпечення	24
3.2 Функціональні вимоги до програми	25
3.3 Особливості структури бази даних	30
3.4 Висновки за розділом 3	43
4 Головні рішення щодо впровадження компонентів системи	44
4.1 Алгоритм функціонування програми	44
4.2 Реалізація основних класів	46
4.3 Висновки за розділом 4	53
5 Експлуатація, тестування та експериментальне дослідження програмного забезпечення	54
5.1 Опис роботи з програмним забезпеченням.....	54
5.2 Експериментальне дослідження роботи вебсервісу послуг паркінг комплексів	57

5.3 Висновки за розділом 5	61
Висновки	62
Перелік джерел посилань	63
Додаток А Технічне завдання	65
Додаток Б Текст програми	70
Додаток В Слайди презентації.....	102

ПЕРЕЛІК СКОРОЧЕНЬ ТА УМОВНИХ ПОЗНАК

XGBoost – eXtreme Gradient Boost;

БД – база даних.

ВСТУП

Місця для паркування стали необхідною частиною житлових, офісних, адміністративних комплексів, а також торгових і розважальних центрів. В сучасних умовах практично для кожного будівельного об'єкта обов'язково вирішується питання щодо організації паркувальних місць для автомобілів його відвідувачів.

Процес паркування легкової автомашини або вантажівки потребує спеціальні ресурси: гроші та час. Не планування завідомо пошуку місця паркування авто призводить до збільшення цих ресурсів.

Недостатність паркувальних місць для автомашини веде до появи процесу необхідності прийняття рішення шофером щодо пошуку тимчасової стоянки поблизу місця призначення або кінця маршруту.

При цьому необхідно відзначити що переміщення легкової автомашини або вантажівки, з урахуванням життєвих потреб людини, за різними маршрутами вимагає тимчасової стоянки.

Необхідно відзначити що маршрути руху автомобіля можуть бути короткими або досить великими тому концентрація автомашин змінюється.

У місцевостях із забудовою, яка складається з будинків до п'яти поверхів, розглядається доцільним впровадження постійного зберігання легкових автомобілів. Це може бути реалізовано через створення малоповерхових гаражів, які можуть бути наземними, підземними або комбінованими, включаючи механізовані (автоматизовані) гаражі простих конструкцій, а також на відкритих автостоянках.

Багатьом власникам автомобілів, які проживають у великих містах, актуальною стає проблема знайдення парковочних місць. Гаражів на всіх не вистачає, а ближчі охороняючі стоянки часто знаходяться за кілька кілометрів від будинків, що особливо не зручно, особливо для сімей із дітьми. Тому, обираючи квартиру у новому житловому комплексі, наявність

паркінгу визнається одним із важливих критеріїв для більшості майбутніх мешканців.

Тому автоводію потрібен алгоритм прогнозування доступності машиномісць на парковці.

Для практичного застосування результатів цієї магістерської роботи необхідно розробити алгоритм та програмне забезпечення прогнозування доступності вільних місць для паркування автомобіля.

1 АНАЛІЗ ПРОБЛЕМИ ТА ПОСТАНОВКА ЗАВДАНЬ ДОСЛІДЖЕННЯ

1.1 Аналіз проблем надання послуг паркінг комплексів

На процес паркування авто в сучасному місті впливає цілий набір чинників:

- при запізнюванні на важливий захід у водія авто може виникнути збентеження або стрес. Так як йому необхідно мати місце для паркування авто, причому бажано – на тривалий час;

- однією із труднощів життя в жвавих районах сучасного міста, особливо при перевантаженості вулиць є недостатність місць для паркування у різних мікрорайонах. Це властиво навіть великим країнам таким як наприклад, Сполученим Штатам Америки;

- вирішення цієї задачі можливо за рахунок збільшення кількості місць для паркування та використання, для оптимального управління ними, відповідних сучасних технологій. Сучасні технології допоможуть оптимізувати алгоритм пошуку для паркування місць для паркування процес замість процесу пошуку їх на місці [1].

Кількість місць для паркування автомобіля між пропозицією та попитом відрізняється. Це можна пояснити наступними факторами:

- історичні та старі міста будувалися з вузькими вулицями без врахування транспортних засобів. Кількість населення цих міст у той час була не такою високою в цих містах була не такою великою. Так як, історичні та старі міста практично перебудові не підлягають, то в них виклика щільність руху та потрібність в паркуванні. При цьому, реальне навантаження за цими показниками перевищує максимальну пропускну здатність;

- великою концентрацією діяльності та видів об'єктів, для яких потрібна достатньо велика кількість транспортних засобів в одній зоні. Такою зоною може бути територія з комерційними об'єктами, офісними будівлями та державними установами і так далі;

– у новому сучасному місті, особливо у нових міських районах, дуже часто не коректно розраховують попит на паркування. Це пояснюється тим що населення з рівнем доходу, вищим за середнє, для своїх поїздки не використовують громадський транспорт. Тому ймовірний рівень володіння авто розраховано не вірно;

– при збільшенні кількості транспортних засобів зростає потреба в кількості місць для їх паркування. А тенденція задоволення наявності місць для паркування залежить від бордюрів та зон для паркування. Отже, маємо недостатність земельних ділянок для забезпечення для паркування місцями;

– необхідно враховувати порушення норм будівництва та вимоги для зонування. Недотримання цих вимог впливає на проєктні розрахунки при забезпеченні достатній кількості місць для паркування автотранспорту в цих зонах [2].

У кожному місті існує область, що призначена для стоянки автомобілів. Зазвичай середній автомобіль, виготовлений із двох тонн сталі, залишається на 74% порожнім, коли його використовує лише одна людина. Проте протягом значної частини дня багато автомобілів залишаються абсолютно непотрібними та невикористовуваними, вимагаючи при цьому великий обсяг простору для паркування. Наприклад, у Сполучених Штатах існує мільярд паркінгових місць, що становить чотири місця на кожен існуючий автомобіль. Крім того, значна частина міської території використовується для автомобільних доріг, що призводить до того, що багато центральних частин міст віддають від 45% до 65% своєї обмеженої площі для транспортних потреб [3].

Тому для вирішення проблеми заторів й зміни клімату почато розробку планів використання в сучасному місті безпілотних авто.

Один з таких проєктів розроблено в сучасному місті Сан-Франциско. Основні його переваги полягають у наступному:

– використання спільного транспорту робить послуги перевезення більш доступними та популярними, і це не обмежується лише платформами,

такими як Uber і Lyft. Мешканці сучасного міста використовують ці автомобільні послуги для своїх транспортних потреб, або комбінують їх із масовими транспортними варіантами;

- розробка єдиного нескладного мобільного додатку. Цей мобільний додаток буде використовуватися для поєднання маршрутів, виконання розробки планів послуг по використанню різних видів транспорту (велосипедів, автомобілів та громадського транспорту);

- виконання цих послуг більш доступнішими. Це можливо за рахунок надання людям, які мають низький рівень доходу, доступ до смартфонів і банківських послуг та можливість використовувати безкоштовно громадське Wi-Fi;

- використання безпілотних електромобілів. Це дасть можливість спростити ці транспортні засоби. Також зменшиться кількість смертельних зіткнень та сунеться забруднення. Тобто усі транспортні засоби можна бути об'єднати в єдину централізовану мережу [3].

Резюмуючи вище наведене для вирішення проблеми пошуку місць для паркування в сучасному місті потрібно:

- для завчасного пошуку та по можливості бронювати для клієнта для паркування місця для автотранспорту необхідно створити сучасне програмне забезпечення;

- для прийняття рішення про необхідність бронювання для паркування місця необхідно мати можливість визначати їх доступність в заданому для паркування автомобілів просторі.

1.2 Огляд сучасного програмного забезпечення для пошуку паркінг комплексів

На початку процесу пошуку програмного забезпечення для завчасного знаходження для паркування місць автотранспорту і можливості їх замовлення по перше розглянемо Parkopedia (рис. 1.1) [4].

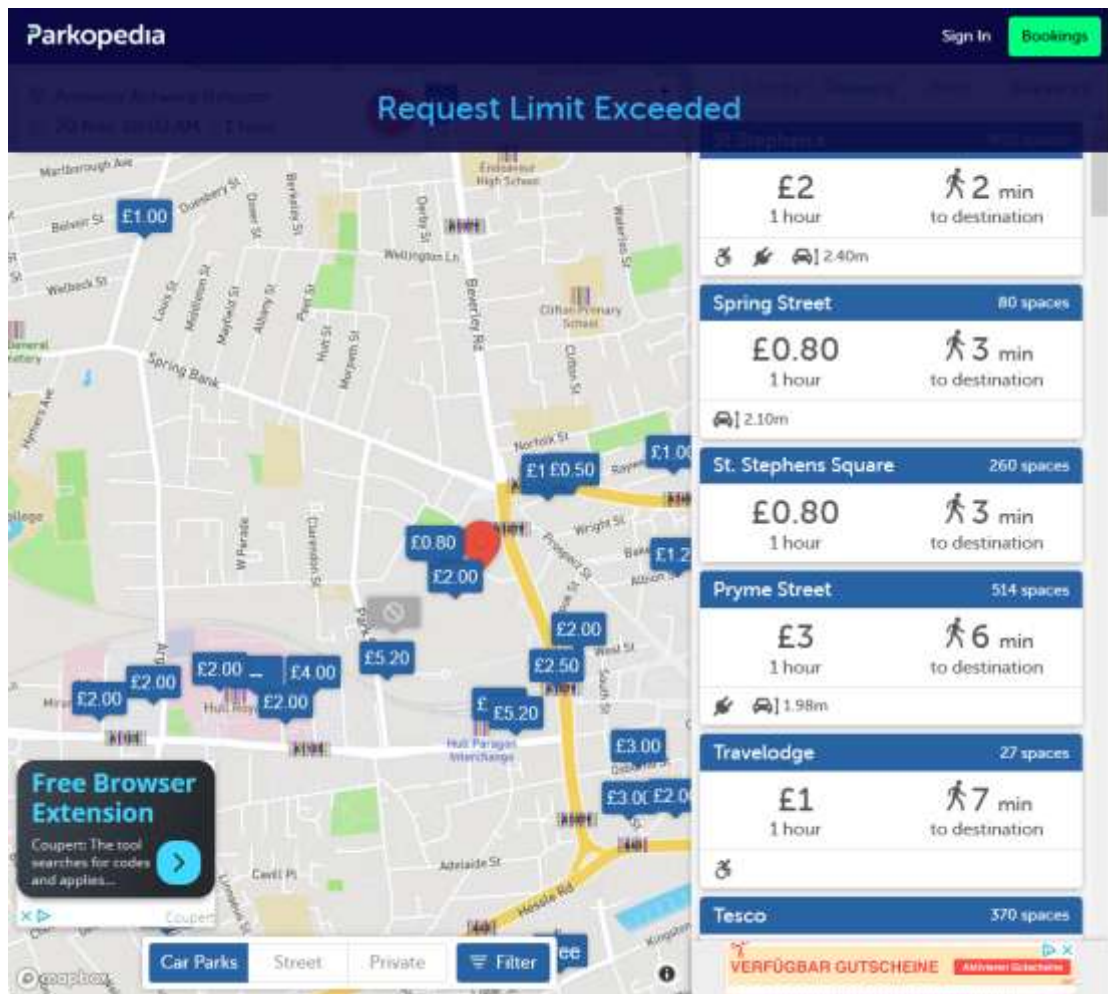


Рисунок 1.1 – Вебзастосунок для замовлення послуг паркінг комплексу Parkopedia [4]

Вебзастосунку Parkopedia притаманна можливість пошуку для паркування місць авто у заданому місті або за заданими адресом та часом. Для кожного знайденого варіанту окресленого проміжку часу встановлюється вартість окресленого проміжку часу паркування, представляється інформація про наявність для паркування авто місць та способи сплати за паркування. Вказуються інформація про способи сплати та можливість бронювання місць для паркування автомобіля.

Існує Вебсервіс Parclick напрямку зі своїм мобільним мобільними додатком (рис. 1.2) [5].

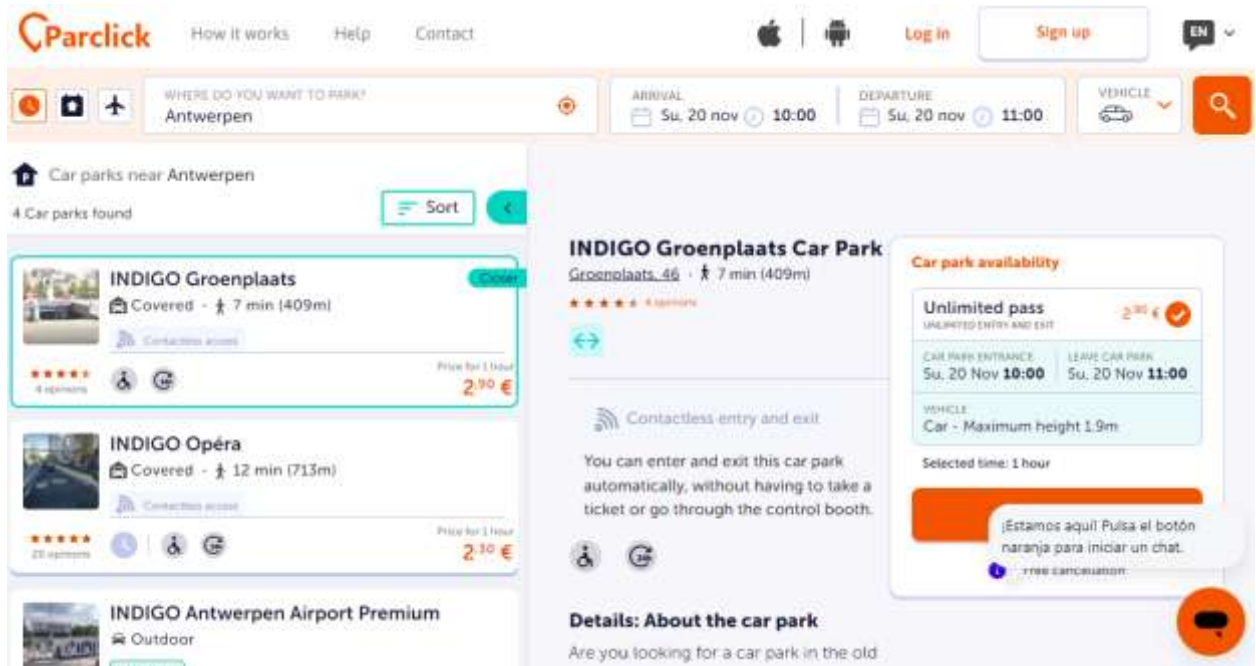


Рисунок 1.2 – Вебзастосунок Parclick [5]

Parclick також дозволяє виконувати пошук вільних місць за вснуючими паркінг комплексами. Крім того, надається можливість забронювати паркінг за додаткову оплату. На відміну від попереднього вебзастосунку, тут клієнти можуть залишати свої відгуки про наданий сервіс. У порівнянні з Parkopedia, де основним обмеженням під час пошуку є висота автомобіля, Parclick також дозволяє задавати тип автомобіля за розміром, такий як автомобіль, автобус, мотоцикл, вантажівка тощо.

Основним напрямком цих додатків є комерційне використання (платне замовлення вільного місця та надання інформації щодо наявності місць для паркування). Проте є істотне обмеження – не має можливості дати клієнту інформацію про наявність вільного місця для необхідного моменту часу.

Пошук для паркування простору здійснюється тільки за фактом. Тому, щоб його розвинути, необхідно додати функцію екстаполяції наявності вільних місць для автомобільного транспорту.

1.3 Висновки за розділом 1

В процесі виконання цієї магістерської роботи необхідно виконати наступні завдання:

- детально виконати аналіз методів, які дозволять прогнозувати наявність та доступність місць для паркування автомобіля;
- розробити алгоритм та програмне забезпечення із замовленням паркувальних місць;
- виконати реалізацію цього програмного забезпечення.

У першому розділі цієї магістерської роботи дана характеристика та проведено аналіз стану проблеми паркування автомобіля в сучасному місті.

Охарактеризовано проблемні ситуації та шляхи для їх вирішення на даний час.

Одним з напрямків вирішення проблеми паркування автомобіля в сучасному місті є розробка програмного забезпечення функцій пошуку, замовлення та прогнозуванню доступності місць паркування автомобіля.

Розглянуто можливості сучасних вебсайтів, які спрямовані на пошук й виконання бронювання вільних місць.

2 МАТЕРІАЛИ І МЕТОДИ

2.1 Задача прогнозування вільних місць для паркування автотранспорту

В даній магістерській роботі проблема доступності місць для паркування аналізується місць в першу чергу з позиції водія автомобіля.

Водій авто повинен зробити вибір між безкоштовним, платним простором для паркування чи необхідно виконати попереднє бронювання.

Для цього випадку йому потрібно знати очікувану кількість вільних місць на планований час приїзду з точністю до хвилини.

При цьому необхідно враховувати вірогідність впливу різних факторів на час коли він дістанеться до мусту призначення.

Тому для водія важливо мати інформацію про мінімальну кількість доступних для паркування автотранспорту місць протягом часу.

Отже, початок процесу прогнозування наявності мінімальної кількості вільних місць для паркування автомобіля відбувається через годину після поточної миті.

Для можливості завчасного прийняття вірного рішення процес прогнозування повинен відбуватися на період через годину відносно моменту, коли водія автомобіля початково це планував.

2.2 Існуючі методи управління та прогнозування доступністю вільних місць для паркування

На даний час декілька досліджень прогнозування доступності місць для паркування автомобіля.

Основними дослідженнями із них можна виділити такі як [6], [7] (табл. 2.1).

Таблиця 2.1 – Порівняльний аналіз методів прогнозування доступністю вільних місць паркінг комплексів

Критерій	Стаття [6]	Стаття [7]	Власний метод
Показник, який прогнозується	Середня кількість місць використаних для паркування за період (год)	Кількість доступних для паркування місць (на даний момент)	Мінімальна доступність для паркування місць (за годин)
Період прогнозування	Година	Година	Година
Вхідні параметри	Часовий ряд використаних для паркування місць, масштаб і тип парковачного місця	Час(година, день тижня), часовий ряд не використаних місць за останні хвилини, середнє значення та стандартне відхилення за минулий період, рівень трафіку	Район, час (година, день тижня) часовий ряд не використаних місць для паркування місць за останні години, рівень трафіку
Модель, яка застосовується в методі	Метод опорних векторів	XGBoost	XGBoost

Пропонований в даній магістерській роботі метод має відміну від існуючих методів. Він прогнозує мінімальну кількість доступних місць протягом години, а не середню або фактичну кількості [6] на даний момент [7]. Це дає можливість автоводію мати корисну інформацію для

прийняття ним рішень в конкретних реальних ситуаціях. Таким чином, в працях [6] та [7] вирішується інша задача по прогнозуванню.

В роботі [7] на базі алгоритмів eXtreme Gradient Boosting (XGBoost) береться як вхідні параметри які належать до вхідного і вихідного трафіків на різних дистанціях.

2.3 Алгоритм XGBoost

Дерева рішень є класом дуже ефективної моделі машинного навчання, що дозволяє отримати високу точність у вирішенні багатьох завдань, зберігаючи у своїй високий рівень інтерпретації [8].

Чіткість подання інформації робить дерева рішень особливо серед інших моделей машинного навчання. Освоєні деревом рішень «знання» безпосередньо формуються в ієрархічну структуру, яка зберігає і представляє знання у зрозумілому навіть для нефахівців вигляді [8].

Алгоритм XGBoost здійснює алгоритм градієнтного бустинга на дереві рішень. Йому притаманні такі властивості:

- не дивлячись що XGBoost асоціюється тільки з послідовним алгоритмом навчання він об'єднує в собі послідовну і паралельну архітектури. Це підвищує ефективність всієї системи;
- XGBoost – підклас машинного навчання з вчителем. Також алгоритм XGBoost – розширений варіант Boosting Machine. А він в свою чергу – підклас алгоритму Random Forest (ансамбля дерев рішень);
- Random Forest паралельно активує незалежні дерева рішень. Усереднює усі результати за рахунок процесу комбінування. При цьому використовується початкова випадкова вибірка. Вона має назву – пакетування. Тому Random Forest притаманні властивості алгоритму паралельного навчання;
- Boosting Machine залучає стратегію з властивістю адитивності. Виконує послідовний запуск простих дерев рішень (базові учні). Тому в

концептуальному сенсі алгоритм Boosting Machine має архітектуру послідовного навчання;

- у алгоритму Random Forest процес навчання паралельний, а у Boosting Machine навчається послідовний;

- алгоритму Gradient Boosting Machine притаманні властивості адитивності тому в ньому для оптимізації реалізовано стратегія градієнтного спуску. При цьому за допомогою розкладання ряду Тейлора апроксимується цільова функція. Використовується тільки члени першого порядку. Для оптимізації процесу навчання ігноруються члени вищого порядку. Таким чином для виявлення наступного предиктора слабкого навчання використовується перша градієнт функції втрати, яка з цільовою функцією. Покращення продуктивності процесу оптимізації для зменшення поточної помилки відбувається за рахунок додавання нового предиктора при зберіганні існуючих слабких. Чим підсилюється градієнт та покращуються продуктивність [9]-[11].

Відомо, що при прогнозуванні зображень та текстів (неструктурованих даних) штучні нейронні мережі переважають інші алгоритми.

Але коли опрацьовуються табличні дані малого й середнього розміру(структуровані дані) то перевага віддається алгоритмам які використовують дерева рішень [12].

Основні характерні відмінності та переваги цього алгоритму:

- властивість розпаралелювання. Воно полягає в послідовній побудові дерева за допомогою процесу розпаралелювання - вживають цикли, за допомогою яких створюють базові учні. Зовнішній цикл виконує функцію розрахунку листових вузлів дерева. Другий цикл є внутрішнім, він виконує процес обчислення функцій та обмежує розпаралелювання. Враховуючи що зовнішній цикл не працює без закінчення внутрішнього циклу. З метою зменшення часу виконання необхідно виконати зміну порядку циклів шляхом глобального сканування сортування та усіх екземплярів. Це дає можливість

поліпшувати продуктивність алгоритму за рахунок компенсації витрат на розпаралелювання в процесі обчислення;

- обрізка дерев, яка використовується для поділу дерева у межах алгоритму Gradient Boosting Machine. Застосовуючи критерій `max_depth` натомість «критерію спочатку» алгоритм XGBoost обрізає дерево в зворотному напрямку, це має назву підхід «підхід спочатку в глибину»;

- оптимізація апаратного забезпечення є алгоритмом спрямованим на ефективне застосування апаратного потенціалу ресурсів за рахунок усвідомлення кешу за рахунок вибору внутрішніх буферів. Подалі виконують оптимізацію дискового простору шляхом обробки великого об'єму даних що не містяться в пам'ять;

- регуляризація виконується для запобігання явищу перенавчання. Штрафуються складні моделі використовуючи регуляризацію LASSO (L1) і Ridge (L2);

- перехресна перевірка є алгоритмом, який застосовує метод перехресної на кожній ітерації перевірки. Це прибирає необхідність в програмуванні цього пошук та визначати кількість ітерацій посилення, які необхідні на один цикл.

2.4 Метод прогнозування наявності вільних місць паркінг комплексу

Для рішення задачі прогнозування наявності вільних місць для паркування в даній магістерській роботі пропонується застосовувати дерева рішень, який використовує алгоритм XGBoost.

На початку використання алгоритму виконується процес формування вибірки даних, виконаних спостережень майданчиків для паркування автомобілів.

Вибірка даних охоплює наступні атрибути ознаки:

- район. Виконують стандартний розподіл міста на райони або місто розбивають на зони де є активність. Наприклад це райони ділового, історичного центру або спальна зона);

- номер дня тижня. Святкові вважаються окремою категорією, якій присвоєно термін «номер»;

- часовий ряд мінімальної доступності місць за проміжок часу останніх годин. Тривалість виконання дослідження - дванадцять останніх годин, тому що менша кількість призводить до ускладнення процесу дослідження у ніч;

- рівень трафіку. Значення середнього рівня трафіку на навколишній території взято 500 метрів;

- вихідна ознака: Критерій мінімальної кількості доступних місць для паркування автомобілів. Він визначається за проміжок часу - одна година, що була чотирнадцятою в послідовності. При цьому, часовий ряд складає дванадцять годин, а тринадцята година пропущена.

Після закінчення процесу формування вибірки вона нормалізується та на основі алгоритму XGBoost виконується процедура навчання моделі.

Далі виконується процедура набору значень вхідних показників, які вже були піддані перетворенню на базі норм для навчальної вибірки. Після цього використовується модель для прогнозування, яка раніше була піддана процедурі навчання. Важливо відзначити що найважливіший набір значень показників береться від простору для паркування автомобіля. Параметр трафік визначається за допомогою сервісу, який перед цим був використано при формуванні вибірки визначення рівня трафіку. Наприклад Google Maps, Here WeGo то що.

Так в алгоритмі Here WeGo використовується слідує рівні трафіку:

- зелений рівень трафіку - вільний рух;
- жовтий рівень трафіку ущільнений рух;
- червоний рівень трафіку- повільний рух;

– чорний рівень трафіку має бути використано для закритих доріг [13].

2.5 Висновки за розділом 2

Сформульовано завдання прогнозування доступності місць для паркування автомобільного транспорту. Виконано огляд та аналіз методів, які спеціалізовані за профілем пошуку рішення подібних завдань. Проаналізовано метод, який ґрунтується на алгоритмі XGBoost.

Представлено метод прогнозування наявності вільних місць для паркування автомобільного транспорту та сформульована наукова новизна цієї магістерської роботи.

Прогнозування доступності вільного для паркування за проміжок часу протягом години. Для цієї години береться відлік від поточного моменту. Процедура машинного навчання ґрунтується на XGBoost алгоритмі. Це дає можливість водію автомобільного транспорту розсудливо вирішувати проблему замовлення місць для паркування автомобільного транспорту.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Створення рішень при розробці програмного забезпечення

Обрання рішення по розробці програмного забезпечення доступності вільних місць паркінг комплексів повинно чинити в комплексному варіанті. Для цього виконується процедура порівняння обох платформ, а не лише перевірка на уподібнення мов програмування при реалізації у всіх їх можливостей. Програмне забезпечення повинне не тільки бути вебзастосунком, а й мати можливість опрацювати дані на основі мови Python. Ця мова, а також мова PHP найкращим чином робить обробку даних. Порівняльний аналіз наведено в табл. 3.1.

Таблиця 3.1 – Порівняльний аналіз рішень для розробки вебсервісу

Критерій	Варіант комплексного рішення на базі Python	Варіант комплексного рішення на базі PHP
Ускладнення формування коду	Простий вираз для складного рішення	Складність реалізації складних рішень. Складність коду читання коду
Підтримка дерев рішень	Scikit-learn, xgboost	Відсутність надійної бібліотеки
Обробка даних	суттєва перевага: pandas, numpy та значна кількість додаткових пакетів	Відсутність гарних рішень
Фреймворк веброзробки	Django	Laravel, Symphony, Yii

Скриптова мова PHP за даним критерієм немає суттєвих переваг та паритету, так як ці платформи містять фреймворки для веброзробки, які вельми відомими, популярні та дуже застосовувані. Крім того вони прості в написанні та сприйнятті коду для прийняття рішення в складних ситуаціях при використанні мови високого рівня програмування Python.

Мова програмування Python має перевагу, так як має значу кількість зручних і популярних бібліотек для опрацювання даних. А на PHP вона відсутня. На Python існує бібліотека для реалізації алгоритму XGBoost.

Тому прийнято остаточне рішення про використання скриптової мови Python на базі фреймворку веброзробки Django й бібліотеки для опрацювання даних і з використанням з дерев рішень.

3.2 Функціональні вимоги до програми

В результаті аналізу аналогів, предметної області та методу, який описується в даній магістерській роботі функціональним умовами до програми є:

- керування даними простору для паркування автомобільного транспорту з урахуванням вільних для паркування місць, режиму роботи та платежів;
- пошук вільних місць для паркування взагалі та на заданий проміжок часу;
- можливість бронювання місця з для паркування простору;
- керування процесом виконаного бронювання;
- виконання керування власними автомобілями;
- надання список виконаного бронювання місць клієнта;
- пошук бронювання враховуючи номер його телефону клієнта;
- надання інформації про наявність вільних місць паркінг комплексів;

- можливість обговорення вражень клієнта про паркувальний простір після процесу паркування авто;
- процес реєстрації та авторизування клієнтів за допомоги програми;
- надання списків обговорення про вільні місця для паркування автомобілів (при цьому інтервал оцінок обмежений);
- надання списку власних думок клієнта про вільні місця для паркування автомобілів;
- можливість авторизування і обмеження в реєстрації операторів для паркування просторів;
- прогнозування наявності місць для паркування на заданий період;
- формування звітів про зайнятість груп місць для паркування за конкретний проміжок часу.

Для наочності користуванням програмою користувачем (базовий актор) на рис. 3.1 наведено діаграма прецедентів користувача.



Рисунок 3.1 – Діаграма прецедентів для користувача

Основні кроки взаємодії користувача з програмою включають такі етапи:

- пошук вільних паркомісць;
- знаходження доступних місць для паркування на вказаний період;
- перегляд списку відгуків про паркомісця;
- виконання процедури авторизації;
- моніторинг зайнятості паркомісць.

Користувач з авторизованим доступом виступає у ролі Клієнта (рис. 3.2). Цей учасник відтворює сценарії взаємодії користувача.

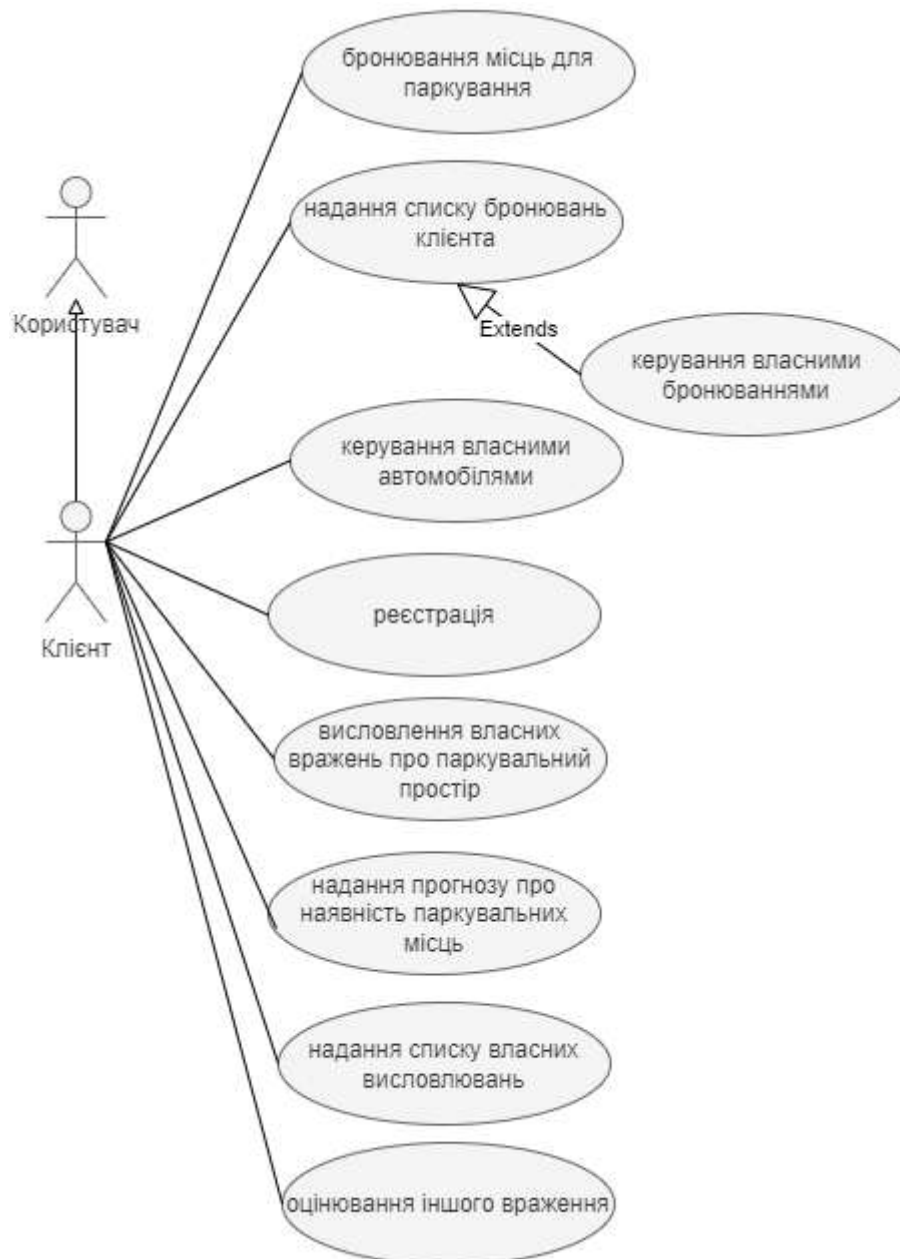


Рисунок 3.2 – Діаграма прецедентів для клієнта

Основними сценаріями для взаємодії клієнта з програмою є:

- бронювання паркомісця;
- перегляд списків автомісць, які вже були заброньовані клієнтом;
- управління власним автомобілем через програму;
- проходження процедури реєстрації;
- можливість висловлювання вражень про паркомісце;
- отримання прогнозу щодо наявності вільних місць;
- перегляд списку особистих висловлювань;
- отримання оцінки враження від інших користувачів.

При цьому передпосилкою цього сценарію виступає потреба в зміні сценарію бронювання, який було клієнтом створеного попереднє, але ще не реалізовано.

Відображення конкретного простору для паркування автомобілів актором, який є оператор, наведено на рис. 3.3 у вигляді діаграми прецедентів оператора.

Сценаріями для головного потоку роботи оператора з програмою визначенні такі:

- сценарій пошуку бронювання паркомісця за простором клієнта;
- сценарій відстеження доступності вільних паркомісць у паркінг-комплексі;
- сценарій керування доступом;
- сценарій управління даними паркінг-комплексу;
- сценарій авторизації;
- сценарій надання звітів про наявність вільних паркомісць у паркінг-просторі для автомобілів;
- сценарій надання прогнозу про наявність вільних паркомісць у паркінг-просторі;
- сценарій управління процедурою бронювання вільних місць за паркінг комплексу для автомобілів.

Для наочності показу широти прав Актора в якості адміністратора наведено на рис. 3.4 діаграма прецедентів адміністратора, який має можливість працювати з усіма наявними для паркування вільними місцями, в тому числі й у власному.

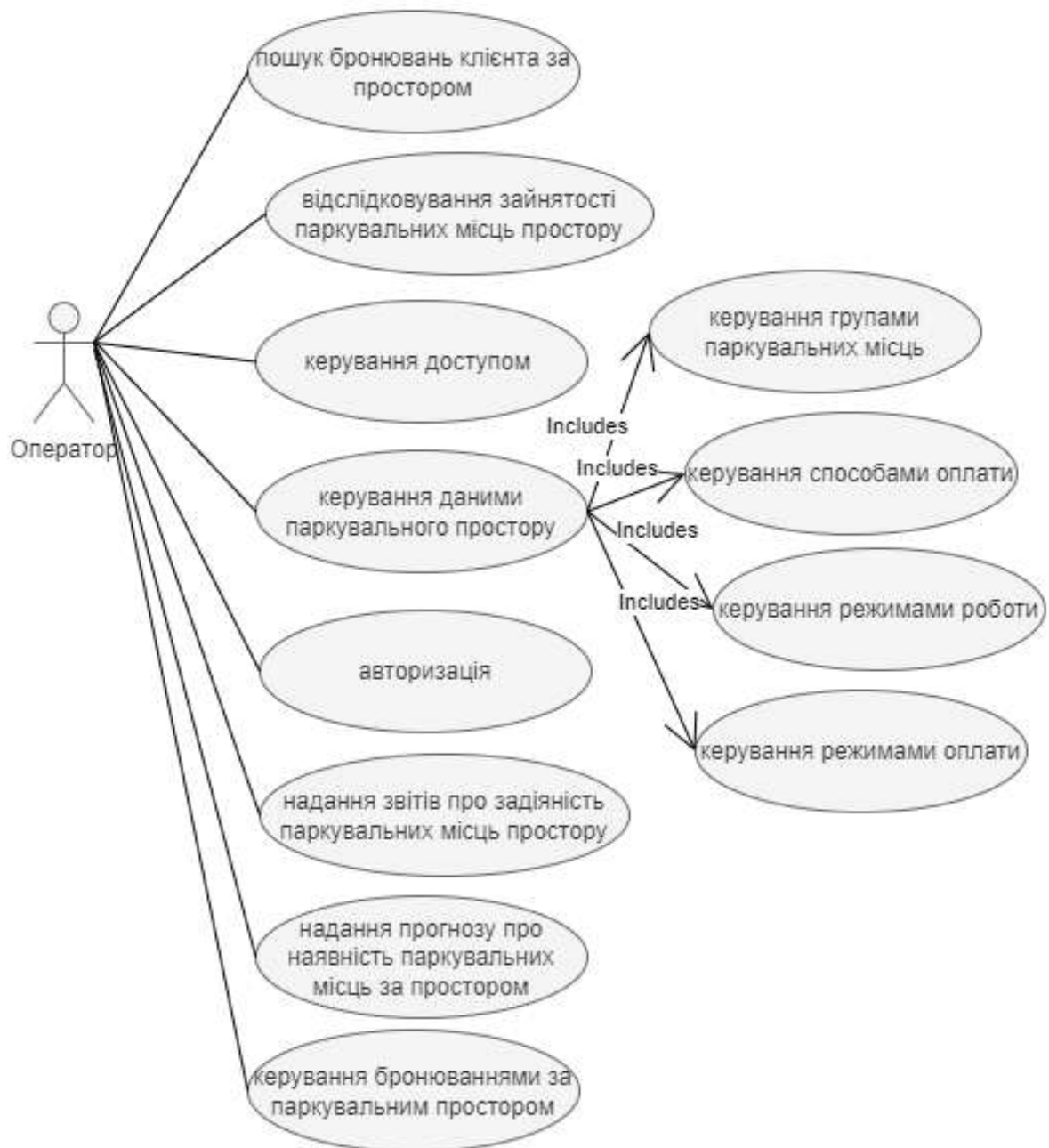


Рисунок 3.3 – Діаграма прецедентів для оператора



Рисунок 3.4 – Діаграма прецедентів для адміністратора

Сценаріями для головного потоку роботи адміністратора з програмою, які не схожі з сценаріями роботи оператора, визначенні такі:

- сценарій з реєстрації операторів;
- сценарій управління даними операторів;
- сценарій з додавання нових просторів для паркування автотранспорту.

3.3 Особливості структури бази даних

База даних (БД) для розробляємої програми застосовується з метою збереження даних головного значення предметної зони, звітів і прогнозів для допомоги виконання процесу прогнозування, що являє собою ознакою цієї програми.

Потужна система керування БД PostgreSQL застосовано для здійснення можливості роботи з БД. Вона досить потужна, надійна, має властивості масштабованості та нема необхідності в витраті коштів [14].

З метою зберігання даних про вільні місця паркінг комплексів розроблена таблиця parkingapp place, базис якої складається з наступних полів:

- id – ідентифікатор вільними місцями паркінг комплексів;
- ididcapture_place _place – ім'я ідентифікатора простору;

- adress – локація (адреса), за якою розташовано простір, у текстовому представленні;
- Y-position – ім'я ідентифікатора значення довготи (першої координати) простору для того щоб відобразити на мапі;
- X-position – ім'я ідентифікатора другої координати (широти) простору;
- info – ім'я ідентифікатора абсолютної текстової інформації для паркування автотранспорту;
- image – ім'я ідентифікатора фотографії, яка має бути використана для сповіщення паркувального місця;
- number_id_place – ідентифікатор загальної кількості всіх типів автомісць для паркування авто паркінг комплексів;
- id_id_type – ідентифікатор виду паркувальних місць (це може бути простір, який охороняється або вуличний. Може бути в вигляді окремого спорудження, яке має роботизовано систему);
- dimention – ім'я ідентифікатора максимальних габаритів автомобіля, що має можливість паркування. Це властиво для критих просторів, просторів у будівлях тощо);
- rang – ім'я ідентифікатора оцінювання (середня оцінка) простору, яке було проведено клієнтами, що паркувалися на даному просторі раніше;
- reserv – ідентифікатор, який виконує визначення можливості процедури бронювання місця на даному просторі для паркування автомобільного транспорту.

З метою зберігання даних про способи оплати для паркування автомобілів розроблена таблиця `rkarr_payment`, базис якої складається з наступних полів:

- id – ім'я ідентифікатора визначення можливості способу сплати в паркінг комплексі для паркування автомобільного транспорту;
- type_payment – варіанти оплати включають у себе можливість оплати безготівковим способом, готівкою в купюрах, готівкою в монетах,

користуванням платіжних карток, а також оплату через сервіси Google Pay та Apple Pay;

- place_payment – паркінг комплекс до якого застосовується відповідний спосіб оплати.

В якості зовнішнього ключа в даній таблиці виступає поле space_payment. Тому для кожного паркомісця є можливість володіти декількома способами сплати. При цьому, має бути варіант без оплати при умові безкоштовного паркування. Реалізується варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до єдиного простору на паркінгу.

З метою зберігання даних про режими роботи або сплати за паркування автомобілів розроблена таблиця rcarp_paymode, базис якої складається з наступних полів:

- id – ім'я ідентифікатора визначення режиму роботи або оплати за паркування автомобільного транспорту на даному просторі;
- place_mode – ім'я ідентифікатора місць паркування при умові визначення режиму роботи та форми сплати;
- cuttent_day – ідентифікатор дня тижня, в якому притаманне такий режим роботи або сплати;
- start_time – ідентифікатор часового інтервалу дії зазначеного режиму;
- stop_time – ім'я ідентифікатора часу, якому притаманно зазначений режим.

В якості зовнішнього ключа в даній таблиці з parkingapp place виступає поле place_mode. Тому для кожного простору є можливість володіти декількома варіантами режиму роботи та способами оплати. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до єдиного простору.

З метою зберігання даних про групи місць паркування автомобілів розроблена таблиця `rкарр_group`, базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора визначення групи місць паркування автомобілів;
- `idcapture_place` – ім'я ідентифікатора назви групи місць паркування авто;
- `place` – ім'я ідентифікатора простір, до якого належить ця група місць;
- `number_id_place` – ідентифікатор кількості місць паркування даної групи;
- `info` – ідентифікатор повного текстового опису ознак описаній групи автомісць;
- `sizeid_type` – тип паркувального місця паркінг комплексу;
- `image` – ім'я ідентифікатора фотографії для застосування інформації групи місць паркування автомобілів.

В якості зовнішнього ключа в даній таблиці з `rкарр_place` виступає поле `place`. Тому для кожного простору є можливість володіти кількома групами місць. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до єдиного простору місць паркінг комплексів.

З метою зберігання даних про недоступність через якість обставини групи місць паркування автомобілів розроблена таблиця `rкарр_blockplace`, базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора про недоступність групи місць паркування авто;
- `block_place` – ім'я ідентифікатора групи автомісць;
- `starttime` – ім'я ідентифікатора дати та часу з моменту яких буде не буде доступне ця група місць для паркування автотранспорту;
- `stoptime` – ім'я ідентифікатора дати та часу до моменту яких буде доступною ця група місць для паркування

– number_id_place – ім'я ідентифікатора кількості автомісць з групи, що є не доступні для процесу паркування.

В якості зовнішнього ключа в даній таблиці з rкарр_blockplace виступає поле block_place. Тому для кожного простору є можливість мати декілька варіантів ситуацій, при яких місця таких груп не досяжні для процесу паркування авто через різні фактори (наприклад ремонт). При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до однієї групи місць для одного простору паркінг комплексу

З метою зберігання даних про бронювання місць паркування автомобілів розроблена таблиця reserv, базис якої складається з наступних полів:

- id – ім'я ідентифікатора процесу бронювання;
- client – ім'я ідентифікатора визначення автомобіля клієнта, для якого реалізовано бронювання;
- block_place – ім'я ідентифікатора визначення групи місць майданчика паркування, на якому здійснено бронювання;
- reserv – ім'я ідентифікатора визначення дати та часу, коли було здійснення з бронювання місця;
- upd_time – ім'я ідентифікатора визначення дати та часу, коли було здійснення оновлення бронювання місця;
- reserv_info – ідентифікатор визначення стану бронювання;
- starttime – ім'я ідентифікатора визначення дати та часу, коли автомобіль може розташоване на місці парки;
- stoptime – ідентифікатор визначення межі дати та часу можливості розташування автомобільного транспорту на майданчику для паркування;
- cost – ідентифікатор визначення вартості бронювання;
- cost_upray – ім'я ідентифікатора визначення сплатення суми за бронювання авто.

В якості зовнішнього ключа в даній таблиці з `parkapp_block_place` виступає поле `block_place`. Тому для кожного простору є можливість мати велику кількість варіантів бронювання для різного проміжку часу. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до однієї групи місць для одного паркування простору.

В якості зовнішнього ключа в даній таблиці з `rkapp_auto` виступає поле `auto`. Тому для кожного простору є можливість мати велику кількість варіантів бронювання для різного проміжку часу. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб кожен з них належав до одного конкретного бронювання паркувальних місць паркінг комплексів.

З метою зберігання даних про клієнтів розроблена таблиця `rkapp_client`, базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора визначення клієнта програми;
- `1_name` – ідентифікатор імені клієнта даної програми;
- `2_name` – ідентифікатор прізвищу клієнта даної програми;
- `password` – ідентифікатор пароля клієнта при роботі з програмою;
- `login` – логін визначений клієнтом для можливості роботи з даною програмою;
- `phone` – ідентифікатор контактного номеру телефону клієнта для даної програми;
- `email` – ідентифікатор визначення електронної пошти клієнта для даної програми;
- `blockreserv` – ідентифікатор заборони можливості доступу до бронювання.

З метою зберігання даних про автомобілі клієнта розроблено таблиця `parkapp_client` на базі таких полів:

- `id` – ім'я ідентифікатора визначення даної програми;

- client – ім'я ідентифікатора визначення клієнта програми до якого відноситься конкретний автомобіль;
- auto – ідентифікатор автомобіля.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_client` виступає поле `client`. Тому для кожного простору є можливість мати велику кількість варіантів автомобілів та бронювання паркування для яких він виконує. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб клієнт мав можливість не мати записів про власні автомобілі в поточний проміжок часу.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_auto` виступає поле `auto`. Тому для кожного авто у програмі є можливість мати велику кількість клієнтів керування цим авто. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю отаким способом, щоб запис стосувався одного конкретного автомобіля.

Сенс таблиці `pkapp_client` полягає в реалізації зв'язку "багато до більшості" і між таблицями `pkapp_auto` і `pkapp_client` таким чином, щоб було можливість проводити процес бронювання паркомісць для одного й того самого автомобіля різними клієнтами, які можуть бути членами однієї родини чи групою. Це дозволяє ефективно управляти можливістю резервування місць для паркування для різних осіб, які використовують один і той же автотранспорт.

З метою зберігання даних про автомобілі клієнта розроблено таблиця `pkapp_auto` базис якої складається з наступних полів:

- `id` – ідентифікатор визначення клієнта даної програми;
- `auto_type` – ідентифікатор визначення типу автомобіля;
- `govnumber` – ідентифікатор визначення реєстраційного номеру автомобільного транспорту;
- `block_reserv` – ідентифікатор визначення заборони бронювання місць для паркування даного автомобільного транспорту.

З метою зберігання даних про враження клієнта від паркомісць для автомобілів розроблено таблицю `parkapp_feedback`, базис якої включає наступні поля:

- `id` – ім'я ідентифікатора визначення висловлювання;
- `client` – ім'я ідентифікатора визначення клієнта, який сформував конкретне висловлювання про конкретне паркувальне місце;
- `place` – ім'я ідентифікатора визначення паркувального простору, про який сформоване дане висловлювання;
- `time_feedback` – ім'я ідентифікатора визначення дати і часу формування даного висловлювання;
- `text_feedback` – ідентифікатор тексту конкретного висловлювання;
- `like_textfeedback` – ім'я ідентифікатора визначення кількості клієнтів, для яких це висловлювання є корисним та не має відповідності до стану простору для паркування;
- `dislike_textfeedback` – ім'я ідентифікатора визначення кількості клієнтів, для яких дане висловлювання некорисне та не має відповідності до стану простору для паркування;
- `rang` – ідентифікатор оцінювання даного простору для паркування автомобіля.

В якості зовнішнього ключа в даній таблиці з таблиці `parkapp_client` виступає поле `client`. Тому для кожного для кожного клієнта у програмі є можливість отримати різну кількість висловлювань. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю от клієнта.

В якості зовнішнього ключа в даній таблиці з таблиці `parkapp_place` виступає поле `place`. Тому для кожного для кожний клієнта у програмі має можливість висловлювання. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю стосовне одного конкретного простору.

З метою зберігання даних сформованих звітів розроблено таблиця `pkarr_per` базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора визначення звіту;
- `block_place` – ім'я ідентифікатора визначення групи місць майданчику паркування в даному звіті;
- `period` – ім'я ідентифікатора визначення періоду формування звіту (наприклад день, тиждень тощо);
- `time_feedbackfor` – ім'я ідентифікатора визначення дати та часу (початок періоду), може бути тільки дата відповідності звіту;
- `max_free_number_id_place` – ім'я ідентифікатора визначення максимальної кількості вільних місць в любий момент часу звітного періоду паркування авто;
- `min_free_number_id_place` – ім'я ідентифікатора визначення мінімальної кількості вільних місць в любий момент часу періоду на просторі паркування авто;
- `avg_free_number_id_place` – ім'я ідентифікатора визначення середньої кількості вільних місць на паркувальному майданчику;
- `reserv_number_id_place` – ім'я ідентифікатора визначення кількості місць, які були бронюванні впродовж даного періоду;
- `reserv_time` – ідентифікатор сумарного часу (хвилинали), впродовж якого було заброньовано місця для паркування автомобіля.

В якості зовнішнього ключа в даній таблиці з таблиці `pkarr_block_place` виступає поле `block_place`. Тому кожна група місць конкретного простору для паркування автомобіля у програмі має можливість мати звіти за весь час доступності для процесу бронювання. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю стосовне однієї конкретної групи місць конкретного простору для паркування авто.

З метою зберігання даних прогнозування наявності вільних місць розроблено таблиця `pkarr_extrapol` базис якої складається з наступних полів:

- id – ім'я ідентифікатора прогнозу;
- time_feedbackfor – ім'я ідентифікатора визначення часу, на який приходить прогноз;
- block_place – ім'я ідентифікатора визначення групи місць для паркування, до якої відноситься прогноз;
- number_id_place – ім'я ідентифікатора визначення мінімальної кількості місць простору паркування. Доступність визначається впродовж конкретного періоду в будь який період часу.

З метою зберігання даних про вартість паркування розроблено таблиця `pkapp_cost` базис якої складається з наступних полів:

- id – ім'я ідентифікатора визначення розцінки за паркування;
- mode_cost – ім'я ідентифікатора визначення способу оплати для даного запису;
- block_place – ім'я ідентифікатора, що визначає групу парковочних місць у контексті відповідно до вартість послуги;
- price_time – ім'я ідентифікатора щодо визначення часу отримання послуги з паркування;
- price_unit – ім'я ідентифікатора щодо визначення ціни за одиницю виміру за ненадання послуги з паркування авто.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_mode_cost` виступає поле `mode_cost`. Тому кожний спосіб сплати має різну кількість записів про вартість паркування автомобіля. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю щоб кожен з них стосувався одного конкретного режиму простору для паркування автотранспорту.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_block_place` виступає поле `block_place`. Тому кожна група місць має різну кількість записів про вартість паркування автомобіля. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення

записів у таблицю щоб кожен з них стосувався однієї конкретної групи місць для паркування авто.

З метою зберігання даних про операторів паркінг комплексів розроблено таблиця `pkapp_operator` базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора визначення оператора простору для паркування автотранспорту;
- `first_name` – ім'я ідентифікатора визначення оператора паркувального простору;
- `surname_name` – ідентифікатор прізвища оператора простору для паркування авто;
- `place` – ім'я ідентифікатора визначення простору для паркування авто;
- `phonenumner` – ідентифікатор визначення номера телефону оператора;
- `password` – ім'я ідентифікатора визначення пароля для можливості роботи з програмним додатком для надання відповідних послуг;
- `login` – ідентифікатор логіна для можливості роботи з програмою оператора простору для паркування автомобілів.

У таблиці "`pkapp_operator`" поле "`place`" використовується як зовнішній ключ, що посилається на таблицю "`pkapp_place`". Кожне паркувальне місце може мати декілька операторів. У таблиці налаштовано записи так, щоб кожен з них вказував на конкретний простір. Тобто можливість працювати з кількома просторами належить лише адміністратору або адміністраторам всієї системи.

На рис. 3.5 наведено схема структури БД яка розроблена за вище наведеним алгоритмом.

З метою зберігання даних про уподобання висловлювань розроблено таблиця `pkapp_answer` базис якої складається з наступних полів:

- `id` – ім'я ідентифікатора визначення запису відповідного відгуку;

- clients – ім'я ідентифікатора визначення клієнта, який залишив відповідне повідомлення;
- text– ім'я ідентифікатора визначення відповідного відгука клієнта.

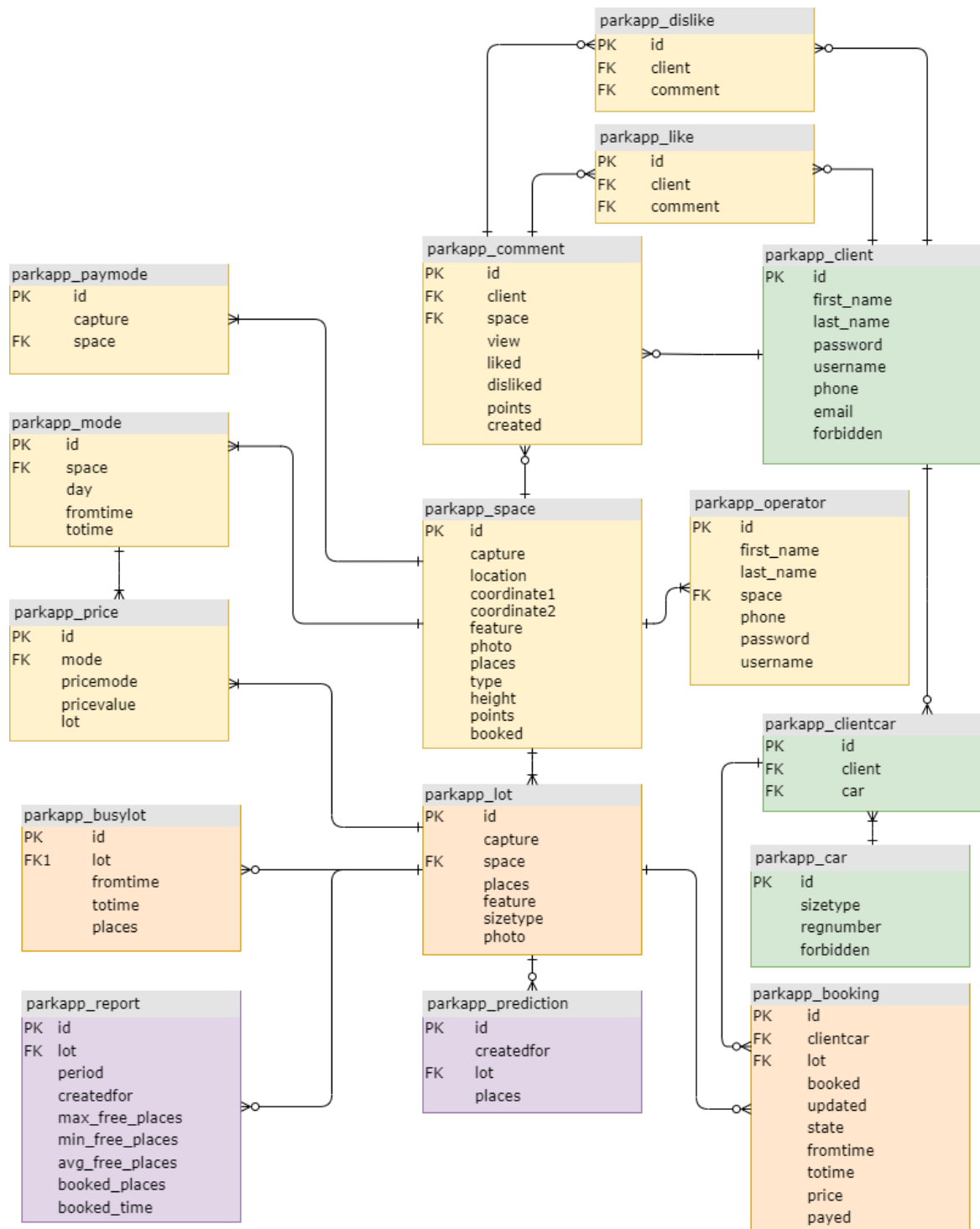


Рисунок 3.5 – Схема структури бази даних відповідних таблиць

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_modes` виступає поле `clients`. Тому клієнт має можливість з ознайомленням різною кількістю висловів інших клієнтів. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю щоб кожен з них стосувався одного конкретного клієнта.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_answer` виступає поле `comment` таблиці `pkapp_answer`. Тому кожний вислів має різну кількість уподобань. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю щоб кожен з них стосувався одного конкретного вислову.

З метою зберігання даних про уподобання висловлювань розроблено таблиця `parkapp_dislike` базис якої складається з наступних полів:

- `id` – ідентифікатор запису про несприятливий ефект;
- `clients` – ім'я ідентифікатора визначення клієнта, який має несприятливий ефект про вислів;
- `comments` – заключна оцінка.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_clients` виступає поле `pkapp_clients`. Тому кожний клієнт має можливість мати різну кількість негативно оцінених висловів решти клієнтів. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю щоб кожен з них стосувався одного конкретного клієнта.

В якості зовнішнього ключа в даній таблиці з таблиці `pkapp_comments` виступає поле `comments`. Тому кожний вислів має різну кількість уподобань. При цьому, має бути реалізовано варіант зв'язку один-до-багатьох, шляхом визначення записів у таблицю щоб кожен з них стосувався одного конкретного вислову.

3.4 Висновки за розділом 3

У третьому розділі проведено розробку програмного забезпечення, включаючи зіставлення і вибір мови програмування Python та фреймворку Django для подальшого впровадження. Проведено моделювання основних функціональних вимог до програми, використовуючи діаграми прецедентів для користувача, клієнта, оператора і адміністратора. Також виконано проєктування структури бази даних.

4 ГОЛОВНІ РІШЕННЯ ЩОДО ВПРОВАДЖЕННЯ КОМПОНЕНТІВ СИСТЕМИ

4.1 Алгоритм функціонування програми

Система починає працювати з моменту запуску вебсервера та самого додатку Django (рис. 4.1). Після чого, одночасно на сервері будуть задіяні на виконання два потоки процесів.

Процедура виконання першого потоку стартує з самого вебдодатку під керівництвом самого додатка Django. При виконанні цього алгоритму усі клієнтські пристрої реалізують звернення до вебсервера. При цьому, вебсервер утворює головну вебсторінку. Формується цикл утворення користувачем (клієнтом, оператором або адміністратором) даних на вебсторінці та визначаються критерії запиту до сервера виконанням процедури внесення їм змін в базу даних (випадку передбачення таких змін запитом). При умові коли такі зміни не були прогнозовані запитом утворюється вебсторінка у відповідь. Процес співдії користувача з вебсервером триває до моменту його закінчення користувачем. Процедура виконання другого потоку базована на скриптах, які реалізовані за розкладом. Послідовна реалізація наступних перевірок обумовлених поточним моментом:

- при умові відповідності поточної миті моменту початка нової години. (на початку для усіх просторів для паркування автотранспорту передбачаються доступність автомісць та готується звіт за годину, яка минула);
- при умові того що поточна мить є началом нової години і началом нового дня, то виконується процедура підготовки звіту за минулу добу;
- при умові того що поточна мить є началом нової години, началом дня і началом нового місяця, то виконується процедура підготовки звіту за минулий місяць;

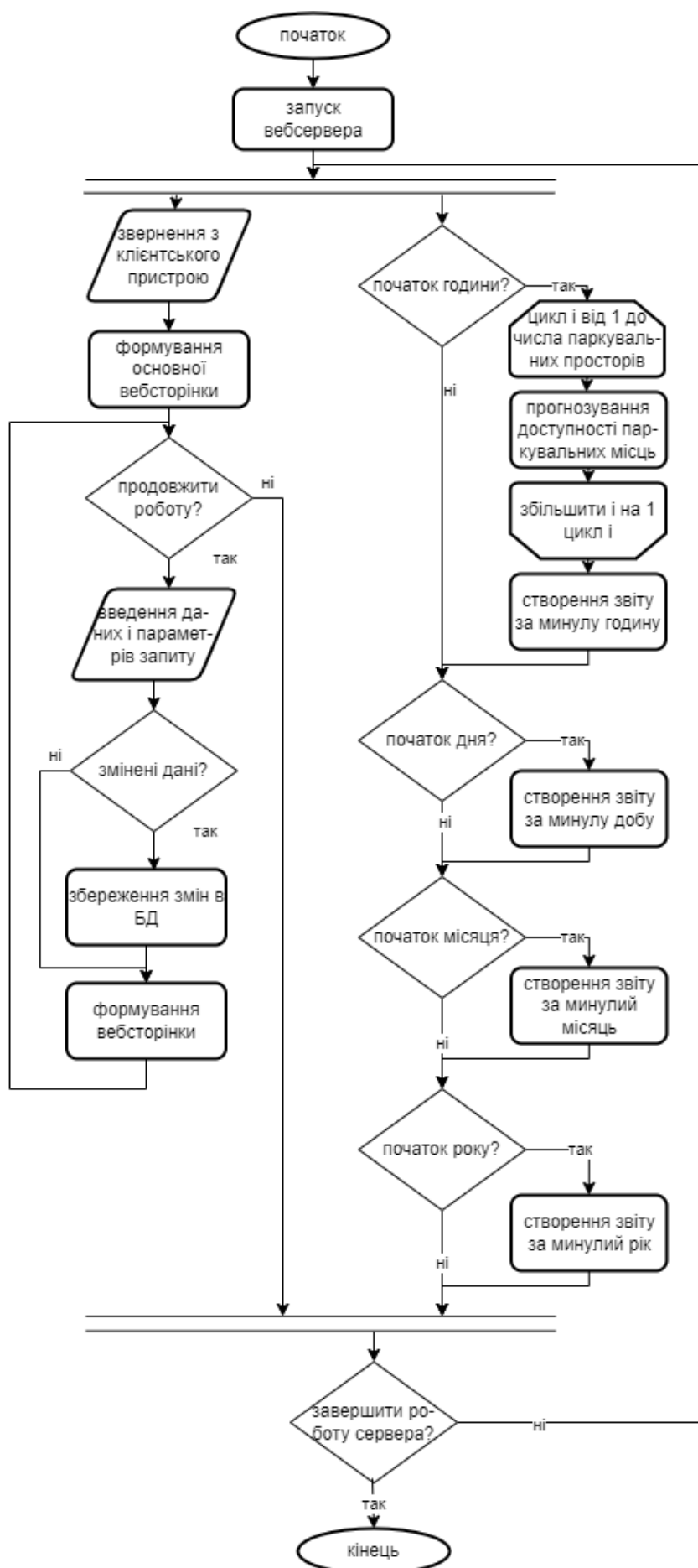


Рисунок 4.1 – Блок-схема роботи програми

– при умові того що поточна мить є не тільки началом нової години, началом дня, началом нового місяця но і началом нового року за минулий рік.

4.2 Реалізація основних класів

Вебдодаток має основні класи, які призначені для створення форм та моделей вебдодатку. Враховуючи, що моделі вебдодатку мають відповідність структурі БД, яка відповідно була досконально визначена в другому розділі, то розглядати повторно ці класи не має потреби. Вони були сформульовані в повному обсязі на базі логіки структур таблиць з використанням вимог упорядкування даних в цих моделях.

Частка рішень вебдодатка сформована не у вигляді класів, що об'єднує в собі менеджер URL, шаблони вебсторінок, деякі і скрипти і призначені для створення етапів, які для наочності наведено на рис. 4.1 (схема функціонування ПЗ). Велике значення надано класам представлень з метою вирішення проблеми наповнення сторінок для відображення різних ролей користувача. Показані положення щодо прийняття рішень з подальшою можливістю їх реалізації.

Шляхом задіяння адміністративної панелі Django реалізується можливість робити адміністратора виконуючи алгоритм реєстрації підхожих моделей у скрипті `admin_edit.py`:

– бібліотека геокодування `geocoder`, яка призначена для одержання інформації про місце розташування користувача. При умові необхідності отримання інформації о його поточного місця розташування необхідно визначити IP-адресу користувача та знайти широту і довготу;

– бібліотека `haversine`. Її функціональне призначення – розрахунок величин відстані між двома точками, обчислення їх широту і довготу. В якості цих точок виступає поточне місце розташування або задаються позиції

клієнта а і розташування самого майданчика для паркування автомобільного транспорту, визначені конкретні координати;

– бібліотека geourl. Її функціональне призначення – користувач розробляє вебсервісі геокодування з метою встановити координати (довгота, широта місця) за конкретним адресом (місто, вулиця, номер будинку).

Приведені бібліотеки та алгоритм їх застосування приманюються з метою формування класів уяви з пошуку.

Клас PlaceSearchView має призначення реалізації можливості пошуку місць для паркування автомобільного транспорту.

Клас Coplex_SearchView має призначення реалізації можливості пошуку просторів для паркування автомобілів.

І той і другий класи пошуку функціонують за рахунок використання можливостей класу List. Він виступає в якості базового класу Django. Це дає можливість виконати упорядковування листів об'єкту, які підходять класу моделей. Так модель Place призначена для класу PlaceSearchView. Модель Place встановлює простір для паркування автомобільного транспорту. Модель Coplex призначена для класу Coplex_SearchView і використовується для встановлення груп місць авто. Процес шукання, керування та бронювання автомобілів здійснюється в межах груп місць, в яких загальні параметри однакові, тому їхня приналежність до конкретної групи легко відстежується. При цьому, не має значення місце розташування на автомайданчику, тому що процес контролювання вчиняється при в'їзді. Отже важною умовою є те щоб автотранспорт мав габарити відповідає вимогам (тільки кемпер (дім на колесах, причеп), автобус або легковий автомобіль) та пересік пункт пропуску.

Виконується процес контролю загальної кількості автомобілів за кожним розміром, а саме:

- чотириколовий автотранспорт;
- двоколовий автотранспорт;
- габаритний автотранспорт (дів на колесах);

- – автобус, електробус.

Процес контролю функціонування Coplex_SearchView виконується за допомогою параметрів GET-запиту:

- address – ідентифікатор адреси, яка має в текстове оформлення (місто, вулиця, номер будинку);
- coordinate_X – ідентифікатор координати довготи, стосовно якої відбувається процес пошуку;
- coordinate_Y – ідентифікатор координати широти, стосовно якої відбувається процес пошуку;
- type_place – ідентифікатор типу місця для паркування авто. Він має мати тільки ціле значення. Під ним розуміється об'єкт в вигляді відкритого або критого простору, вуличного простору, простору під охороною, окремої будівлі, та будівлі яка має роботизовану систему);
- dimention_type – ідентифікатор типу габаритів автомобіля (за умовчанням параметр має дорівнювати 0);
- height_dim – ім'я ідентифікатора висоти автомобіля (за умовчанням параметр має дорівнювати в значенню 213);
- dist_dim – ім'я ідентифікатора відстані
- points – ім'я ідентифікатора мінімальної оцінка простору для паркування авто (за умовчанням параметр має дорівнювати в значенню 1)
- day – ім'я ідентифікатора частини дати виконання пошуку місця та характеризує порядковий номер дня в місяці;
- month – ім'я ідентифікатора частини дати виконання пошуку місця та характеризує місяць;
- year – ім'я ідентифікатора частини дати виконання пошуку місця та характеризує рік;
- hour – ім'я ідентифікатора частини дати виконання пошуку місця та характеризує час;
- minute – ім'я ідентифікатора частини дати виконання пошуку місця та характеризує хвилину.

При функціюванні класу `Coplex_SearchView` розв'язувалось завдання обчислення вартості паркування для відповідного для паркування простору, яка залежить від часу і дня тижня.

Система розпізнавання режиму вартості має вигляд:

- цифра 10 означає значенню вартості в понеділок;
- цифра 20 відповідає вартості у вівторок;
- цифра 30 означає значенню вартості в середу;
- цифра 40 означає значенню вартості в четвер;
- цифра 50 означає значенню вартості в п'ятницю;
- цифра 60 означає значенню вартості в святковий день;
- цифра 70 означає значенню вартості в робочий день;
- цифра 80 означає значенню вартості в вихідний день;
- цифра 90 означає значенню вартості в будь-який день (вартість однакова).

Знайдене значення зберігається в полі моделі `Mode` з ім'ям ідентифікатора `day`.

Виконання умови відповідності варіанту для конкретної дати виконання процесу пошуку можливо за допомогою наступних бібліотек:

- бібліотека `holidays`. Її функціональним призначенням є визначення країни та наявності відповідності поточної дати святковій даті;
- бібліотека `calendar`. Її функціональним призначенням є формулювання кількості днів у місяці;
- бібліотека `pandas`. Її функціональним призначенням є визначення наявності відповідності поточної дати даті робочого дня.

Контроль режиму роботи класу `PlaceSearchView` проводиться за допомогою параметрів GET-запиту:

- `name` – ім'я ідентифікатора назви на належність фрази цієї назви;
- `address` – ім'я ідентифікатора адреси в форматі тексту (наприклад: місто, вулиця, номер будинку);

- `coordinate_X` – ім'я ідентифікатора координати довготи, відносно якої виконується процес пошуку;
- `coordinate_Y` – ім'я ідентифікатора координати широти, відносно якої виконується процес пошуку;
- `type` – ім'я ідентифікатора типу простору для паркування (повинне мати ціле). Позначає відкритий вуличний або критий, під охороною простори, будівлю окрему або будівлю яка має роботизовану систему;
- `height` – ім'я ідентифікатора габариту автомобіля (м) в метрах (за умовчанням параметр має дорівнювати в значенню 213);
- `dist` – ім'я ідентифікатора відстані (м) від місця виконання пошуку (за умовчанням параметр має дорівнювати в значенню 50);
- `points` – ім'я ідентифікатора мінімальної оцінки простору для паркування автомобіля (за умовчанням параметр дорівнює 1).

При функціюванні класу `CommentLikeView` формується, коли виявлено що ці висловлювання є корисними. Клас `CommentLikeView` сформовано шляхом використання функціональних можливостей базового класу `View` (представлення без особливостей) та класу `LoginRequiredMixin` з умовою виконання прелімінарної авторизації. В цьому випадку користувач не має можливості висловлювання, доки не виконає в програмі процедуру авторизації (для появи можливості виконання ролі клієнту).

Для формування вподобання необхідно виконання умови того щоб це уподобання не належало до невподобаних (відсутні попередні висловлювання). Також необхідно щоб дане висловлювання цим користувачем раніше було вподобане, це потрібно для реалізації умови неможливості виконання штучній зміни кількості вподобань висловлювань.

Процес реалізації алгоритму роботи цього класу виконується шляхом натискання на відповідної кнопки в інтерфейсі програми та заснована на методі `post (ourselves, req, pk)`. При умові існування вже такої ситуації формується помилка в вигляді об'єкту класу `IntegrityError` та задіється об'єкт моделі `Dislike`.

При умові відсутності об'єкту спроба створити і зберегти об'єкт уподобання реалізується механізм спроби створення й збереження об'єкту вподобання. Його буде збережено без помилки при умові що він не був раніше збережений в базі даних. При умові коли він був збережений раніше то відбувається збільшення кількості вподобання для даного висловлювання.

Класу `CommentDislikeView` притаманні такі ж властивості для підтримки процедури формування механізму внесення висловлювання до несподобаних висловлювань.

При функціюванні класу `SpaceView` формується представлення роботи з даними простору для паркування автомобілів. З метою реалізації даних задіється клас деталізації `DetailView`. Головне його завдання полягає в наповненні даними сторінки конкретного об'єкту простору паркування авто з номером `pk`. Реалізація виконується з використанням методу `get (ourselves, req, pk)`.

Метод `get` реалізує алгоритм формування об'єкта з паралельним заповненням. Це пов'язане з умовою набору параметрів режиму оплати за допомогою відповідного зворотного поля моделі `raumodes` з урахуванням інформації про групи місць паркування автомобільного транспорту.

Враховуючи що необхідно встановлювати стан кожної групи для конкретного моменту часу формування даних виконується на поточну мить, або на дату передачі з використанням параметрів GET-запиту. Параметрів GET-запиту мають наступне призначення:

- `day` – параметр встановлення дня місяця;
- `month` для встановлення місяця;
- `year` для встановлення року;
- `hour` для встановлення години;
- `minute` для встановлення хвилини.

Визначення дати та часу мов об'єкту `datetime` виконується з переданих значень з реалізацією перевірки цієї дати на приналежність свята або робочого дня. Потім робиться пошук варіанту режимів роботи для цих

випадків. В випадку не виявлення результату реалізується процес пошука за номером робочого дня тижня. Якщо не виявлено результат і в цьому випадку то виконується алгоритм пошуку режиму для всього тижня(постійний графік).

Далі виконується для конкретної групи місць простору для паркування автотранспорту процедура визначання:

- кількості недоступних автомісць, які недоступні на конкретну дату (ремонт або інші причини не можливості використання) за допомогою ключа `busy_places` параметра `lots` змісту конкретної сторінки;
- кількості автомісць яка не належить даній групі місць на цьому просторі для паркування автотранспорту, але мають вірогідність повернення (ремонт або інші причини не можливості використання);
- кількості автомісць, що є заброньованими на заданий момент часу;
- кількості вільних автомісць заданий момент часу (за допомогою ключа `busy_places`);
- інформація про всю групу автомісць з простору для паркування, що можуть бути повернені за допомогою ключа `busy_places`);
- вартості для конкретного режиму паркування що повертаються за допомогою ключа `prices` (параметр `lots`).

Список груп місць груп простору з паркування з необхідними сучинниками, які були наведені вище, як складова параметра `lots` змісту даних, переказуються у шаблон. Спільно з ним, як складова параметра `space`, переказується контекст даних режиму роботи простору для паркування та, як складова параметра `comments` переказується три останніх висловлювання про розглядаємий простір для паркування.

Деяка кількість класів представлень виконують функцію перегляду висловлювань. Так, клас `AuthClientCommentsView` формує список, раніше розроблених авторизованим користувачем, на дану мить. Публікації впорядковуються з моменту кінцевого висловлювання. Ці класи реалізуються

шляхом для долучення даних головних значень, процесу редагування та конфіскації коли це потрібно.

Клас `PredictionView` формує підтримку можливості перегляду варіантів прогнозів простором для паркування та групами автомобілів, які йому належать. Класи `OperatorHourReportView`, `OperatorDayReportView`, `OperatorMonthReportView`, `OperatorYearReportView` формують допомогу роботі зі звітами за різний проміжок час (година, доба, місяць, рік).

Для того щоб реалізувати алгоритм розробленого методу було задіяно слідує бібліотеки:

- бібліотека `pandas`. Її функціональним призначенням є робота з вибіркою даних. Дані цієї вибірки беруться з відповідного CSV-файлу;
- бібліотека `xgboost`. Її функціональним призначенням є організація роботи логіки взагалі та алгоритму XGBoost.

4.3 Висновки за розділом 4

В результаті роботи по тематиці магістерської роботи четвертому розділі описано основний алгоритм роботи вебдодатку і скриптів з функціональним призначенням - обслуговування його роботи.

Сформульовано та представлено описано алгоритм створення основних класів вебдодатку та наведені приклади використання бібліотек.

5 ЕКСПЛУАТАЦІЯ, ТЕСТУВАННЯ ТА ЕКСПЕРИМЕНТАЛЬНЕ ДОСЛІДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Опис роботи з програмним забезпеченням

Для забезпечення взаємодії клієнтів несхожих ролей з вебсервісом було виконано розробку інтерфейсу користувача на базі шаблону, та створено інтерфейс для реалізації в єдиному стилі усіх ролей користувача [15].

Після закінчення процесу авторизації основним призначенням роботи клієнта є пошук місць для паркування автотранспорту, що для наочності проілюстровано рис. 5.1. У випадку, коли клієнта неавторизовано, то меню зменшують користуючись діаграмою прецедентів. Бажання виконати процес бронювання веде на сторінку авторизації.

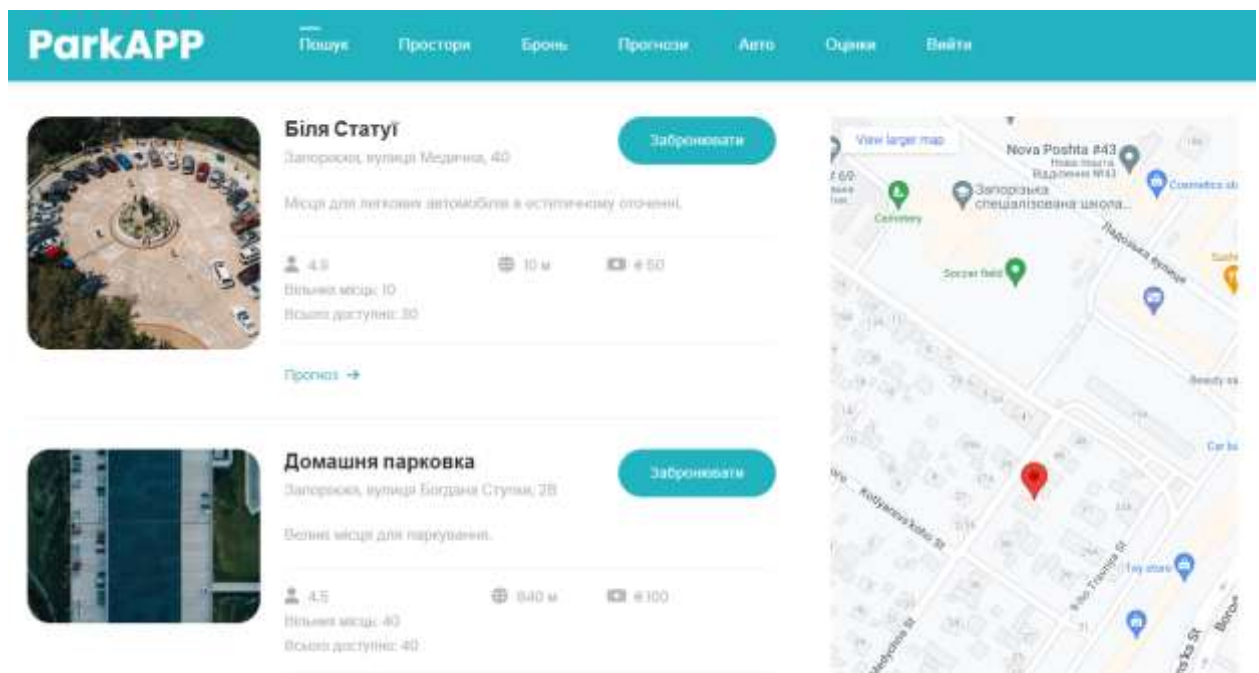


Рисунок 5.1 – Вебсторінка щодо вибірки паркін комплексів на паркувальних для бронювання місця для паркування

Форма для визначення критеріїв пошуку містить:

- адресу в текстовому форматі (Наприклад це може бути місто, вулиця будинок. Інформація вводиться вручну або обирається за координатами на мапі, шляхом натискання на галочки на “ мапі” (зображення ділянки). Також є можливість задавати величину радіусу пошуку (м). За умовчанням параметр має дорівнювати п’ятистам метрам.

- категорія паркувальних комплексів для паркування автомобіля. Вибір проводиться з випадającego списку імовірних варіантів (вуличний, під охороною, критий, будівля окрема або з роботизованою системою);

- тип габариту автомобіля. Вибір проводиться з випадającego списку імовірних варіантів автотранспорту (мотоцикл, легковий або вантажний автомобіль, автобус чи електробус, фургон,);

- габарити автомобіля, вимірюються в м;

- мінімальна оцінка паркінг комплексу. Вибір проводиться з випадającego списку імовірних варіантів;

- дата і час пошуку вільних місць для паркування автомобіля.

Результатом процесу пошуку є відтворення групи автомісць, яка є відповідає конкретному типу авто за критеріями пошуку (назва, адреса, оцінка та фотографія простору, віддаленість від заданого місця, вартість паркування автомобіля, кількість вільних автомісць та які є можливість забронювати на поточний момент).

Можливість бронюванням на просторі для паркування автомобілів з’являється шляхом натиску кнопки з відповідною назвою. В результаті що з’являється сторінка, яка має форму бронювання (рис. 5.2). Потім задається час паркування та обрається необхідний тип автомобіля з пред’явленого користувачу списку за допомогою пункту меню “Авто”. Далі відбувається обчислення вартості бронювання. Потім за умови появи підтвердження відбувається процедура переходу на сторінку оплати.

Користувач має можливість перегляду і при необхідності відмінити не виконану процедуру бронювання (якщо вона ще не було виконано) за допомогою меню “Бронь”. Можливість перегляду прогнозу наявності

кількості вільних місць через годину на конкретному просторі для паркування можливо здійснити за допомогою меню “Прогнози”.

При умові бажання простого знайомства з паркінг комплексами (пошук тільки просторів, а не груп місць), можливо задіяти меню “Простори” (скорочений варіант пошуку).

Меню “Оцінки” дає можливість клієнту перегляду своїх висловлювань щодо просторів.

ParkAPP | Пошук | Простори | Бронь | Прогнози | Авто | Оцінки | Вийти

Бронювання місця на паркувальному просторі Біля Статуї

Початок бронювання, дата: 07.12.2022 | Кінець бронювання, дата: 07.12.2022

Початок бронювання, час: 12:00 | Кінець бронювання, час: 15:55

Оберіть свій автомобіль: AA0330AA

Вартість: € 200

Підтвердити і Перейти До Сплати

Рисунок 5.2 – Вебсторінка з деталізацією опцій бронювання місця

По закінченні процедури авторизації процес роботи оператора виконується з належним простором для паркування.

Меню “Прогноз” дає можливість клієнту перегляду варіантів прогноз про наявність кількості місць на даному просторі для паркування за усіма групами автомісць (рис. 5.3). Корегування інформації про цю групу автомісць можливо за допомогою кнопки “Редагувати”.

Функція керування усіма бронюванням за відповідним простором та пошуку за номером телефону користувача можливо при задіяні меню “Бронь”.

Функція заборони доступу клієнта або автомобіля до процедури бронювання реалізується через меню “Доступ”.

Можливість перегляду звітів з встановлення тривалості зайнятості місця на просторі для паркування авто реалізується через меню “Звіти”.

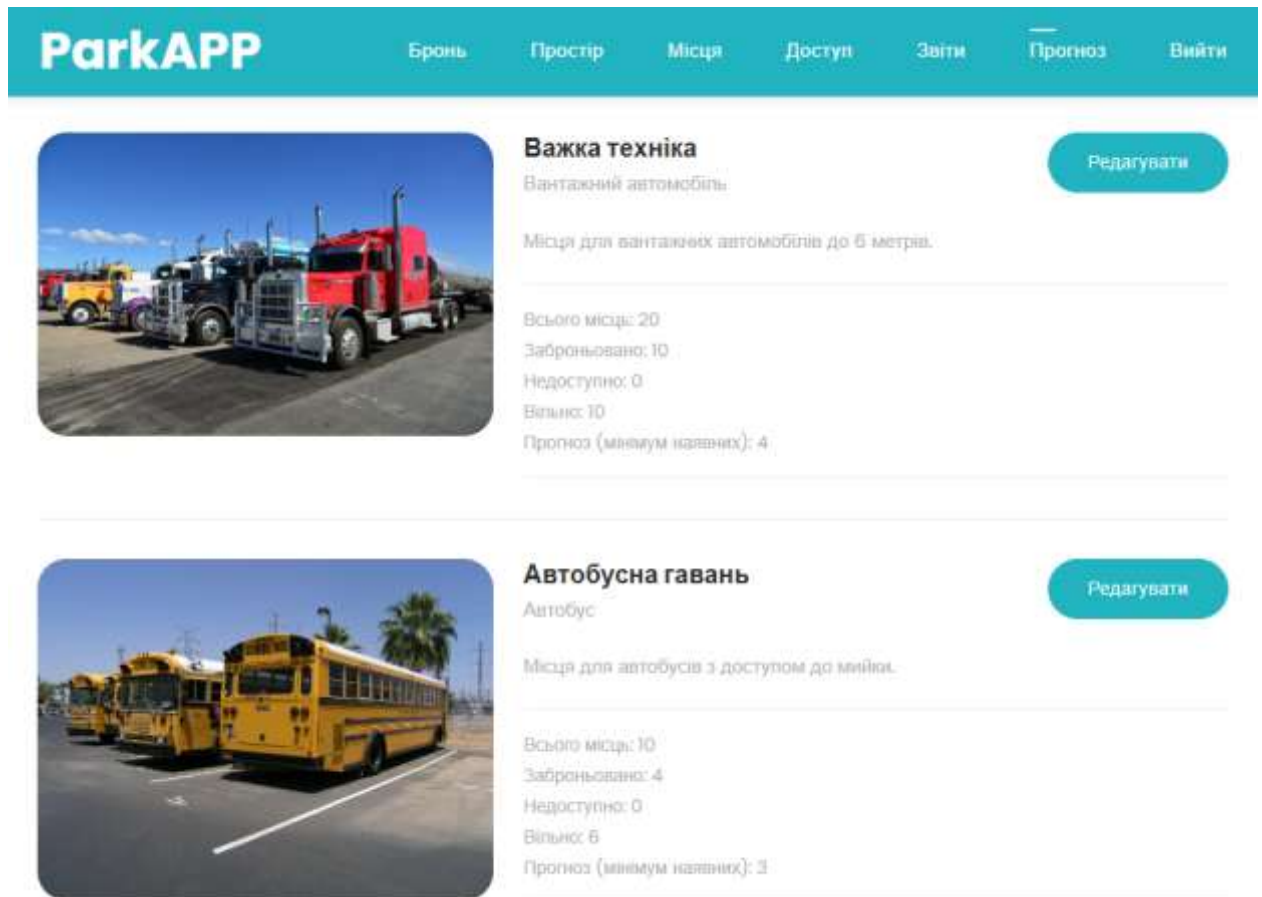


Рисунок 5.3 – Вебсторінка з вибіркою паркінг комплексів з реалізацією опції прогнозування вільних місць

5.2 Експериментальне дослідження роботи вебсервісу послуг паркінг комплексів

Експериментальне дослідження виконано шляхом обробки даних які належать до спеціальної вибірки [16]. Ця вибірка експериментальних даних була отримана на базі даних про вільні паркувальні місця для автомобільного транспорту на протязі 2022 року в Португалії (місто Лісабон).

Параметри даної спеціальної вибірки експериментальних даних:

- ідентифікатор вільних місць для паркування автомобільного транспорту;
- назва паркінг комплексу для автомобільного транспорту;
- значення максимальної кількості місць на відповідному паркінг комплексу для паркування автомобільного транспорту;
- значення кількості зайнятих місць паркінг комплексу для паркування автомобільного транспорту на даний час;
- параметри географічних координат паркінг комплексу для паркування автомобільного транспорту;
- значення часу з моменту отримання цієї інформації;
- значення часу з моменту отримання актуальної інформації такій кількості автомісць.

У процесі і в ході дослідження експериментальних даних вибірки було проаналізовано фактори впливу на кількість вільних місць. На базі здобутих результатів дослідження було створено метод прогнозування доступності місць паркінг комплексу для паркування автомобільного транспорту.

В процесі роботи над магістерською роботою з'ясовано факт наявності незначного взаємозв'язку мінімальної кількості вільних місць паркінг комплексу для паркування автомобіля від конкретної години доби, хоч вона не явно визначена.

Було виявлено що загальна форма зміни такої залежності для не типових паркінг комплексів несхожа. Цей факт пояснюється різницею в кількості доступних місць вдень і вночі та місце розташуванням майданчика для паркування авто та напряду не залежить від сезону року. Приклад конкретного паркінг комплексу для паркування автомобілів наочно представлено на рис. 5.4.

Для підтвердження вище наведеного, було обрано 3 дати на протязі одного року: 15, 28, 29 червня 2023 року.

Для дати двадцять дев'яте червня дві тисячі двадцять третього року характерно наявність великою кількості доступних місць, але вона менше порівнюючи датою.

Хоча загальне зростання кількості доступних місць для паркування в денний час доби притаманно як для двадцять восьмого червня, так і для двадцять дев'ятого червня дві тисячі двадцять другого року, то форма кривої залежності між цими двома сусідніми датами має відмінність.

Для дати п'ятнадцяте червня дві тисячі двадцять третього року характерно, що мінімальна кількість вільних місць у нічну час доби та зранку приблизно така ж що і для двох обраних літніх дат. Однак у середині дня ця зміна стало значною (більшала до 16 год.), але потім різко зменшувалася.

Приведені календарні дати вибрано випадковим чином з метою ілюстрації характеристик взаємозв'язків сезону або сусідніх дат по знаходженні кількості вільних місць у майбутні періоди. Хоч існування самої залежності безперечний факт. Але їй притаманно властивість до зміни для різних дат.

При умові відсутності місць на автомайданчику взагалі ситуація може бути критичною , і як результат - невірне рішення водія.

Експериментальні дослідження робилися за допомогою моделей прогнозування, які було наведено у другому розділі що використовують часові ряди. Це такі моделі прогнозування: SARIMAX, XGBoost [7] та метод опорних векторів [6], розроблений метод. Результати експериментального дослідження приведено в табл. 5.1.

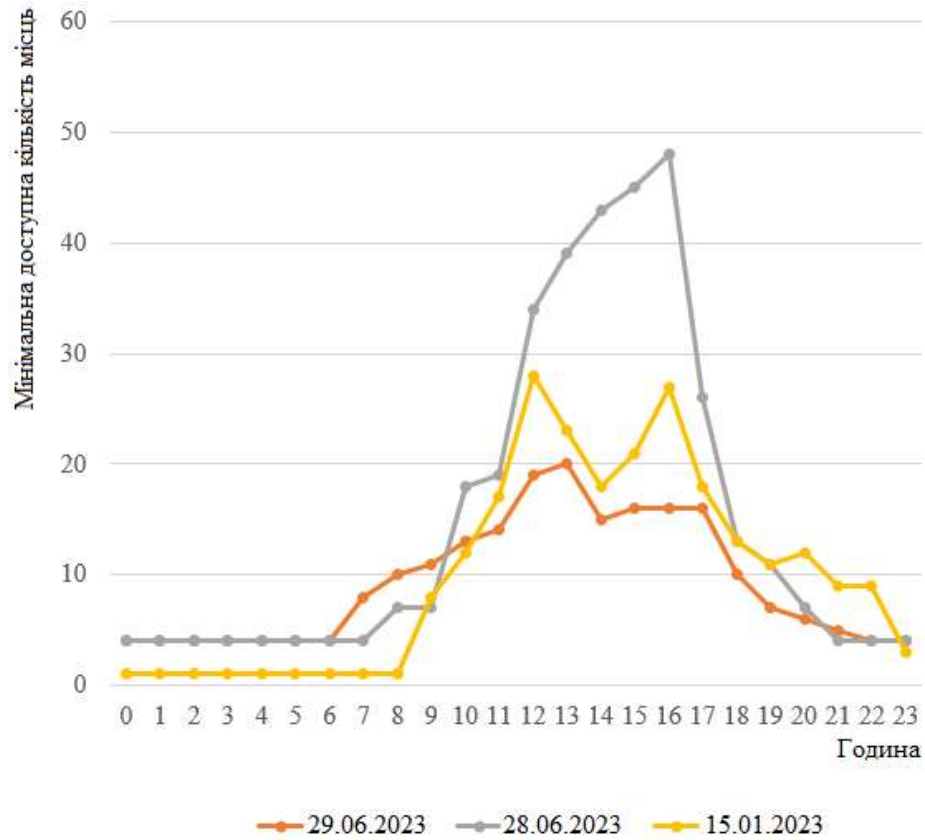


Рисунок 5.4 – Залежність кількості вільних місць паркінг комплексу від годин протягом доби

Таблиця 5.1 – Порівняльний аналіз результатів експериментального дослідження

Назва методу / моделі	Відносна похибка
Модель SARIMAX	0,434
Метод опорних векторів, згідно [6]	0,381
Метод XGBoost, згідно [7]	0,268
Розроблений метод	0,205

За допомогою запропонованого методу отримано найліпший результат: коренева середньоквадратична похибка склала 0,205. Розроблений метод на 23,5 % краще в порівнянні XGBoost за [7] за однаковими для прогнозування показникам.

Запропонований метод в даній дипломній кваліфікаційній роботі магістра при значно меншій кількості звернення до сервера має значно меншу похибку, в порівнянні з SARIMAX, XGBoost [7] та методом опорних векторів [6].

Ця обставина дуже суттєва для сервісу, який забезпечує трафік в даному місті та дає можливість збільшити кількість запитів для проведення наукових досліджень в різних містах світу.

5.3 Висновки за розділом 5

У п'ятому розділі цієї магістерської дипломної роботи надано опис основних можливостей, які є необхідними для роботи з програмою різними клієнтами. Експериментальне дослідження, яке було проведено, підтвердило перевагу запропонованого методу з точністю прогнозування та кількістю запитів.

ВИСНОВКИ

В результаті виконання магістерської дипломної роботи розроблено програмне забезпечення для бронювання місць паркування автотранспорту. У роботі представлено метод прогнозування доступності місць для паркування автомобілів. Використовуючи цей метод, було створено програмне забезпечення, яке надає реальні та прогнозовані дані, дозволяючи автоводіям заздалегідь бронювати місця для паркування своїх автомобілів. Проведено експериментальне дослідження з метою знаходження рішення проблеми прогнозування доступності простору для паркування автотранспорту. Для досягнення поставленої мети в даній магістерській роботі було виконано:

- проведений аналіз наявних аналогів для формування процесу прогнозування вільних місць для паркування автомобілів, виділення унікальних функцій цієї програми;
- прийнято рішення щодо використання Python і Django у процесі розробки, враховуючи використання бібліотек для обробки даних та інтелектуального аналізу, що підвищило ефективність розробки;
- сформована та запропонована структура бази даних, враховуючи функціональність збереження даних про парковочні місця та інформацію, отриману в результаті прогнозування доступності місць для паркування;
- описані заходи по розробці алгоритму роботи та реалізації запропонованого програмного забезпечення;

Обґрунтованість представленого метода та достовірність підвищення його ефективності, порівняно з існуючими рішеннями, були підтверджені експериментальними дослідженнями, які використовували вибірку реальних даних.

Дипломна магістерська робота була виконана на основі раніше сформульованого переліку завдань, і всі вихідні пункти були успішно виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ

1. Lack of Parking in Big Cities [Electronic resource] : Reasons, Causes, and Potential Solutions. – Access mode: <https://studycorgi.com/lack-of-parking-in-big-cities-reasons-causes-and-potential-solutions/>.
2. Ahmed, I. S. Car Parking Problem in Urban Areas, Causes and Solutions [Electronic resource] / I. S. Ahmed // Towards A Better Quality of Life : The 1st International Conference, 24-26 November 2017. – El Gouna, Egypt : Technische Universitat Berlin Campus El Gouna, 2017 – Access mode: https://papers.ssrn.com/sol3/papers.cfm?abstract_id=3163473.
3. Plumer, B. Cars take up way too much space in cities. New technology could change that [Electronic resource] / B. Plumer. – Access mode: <https://www.vox.com/a/new-economy-future/cars-cities-technologies>.
4. Find Parking Block_places, Parking Garages, Parking Meters, Private Garages – Reserve Parking [Electronic resource]. – Access mode: <https://www.parkopedia.com/>.
5. Online Parking Reservations. Parclick [Electronic resource]. – Access mode: <https://parclick.com/>.
6. Zhao, Z. A Comparative Study of Parking Occupancy Prediction Methods considering Parking Id_type and Parking Scale [Electronic resource] / Z. Zhao, Y. Zhang, Y. Zhang // Journal of Advanced Transportation. – 2020. – Volume 2020. – Access mode: <https://www.hindawi.com/journals/jat/2020/5624586/>.
7. Parking Availability Prediction Using Traffic Data Services [Electronic resource] / I. Klandev, M. Tolevska, K. Mishev, D. Trajanov // ICT Innovations 2020 : Conference. – 2020. – Access mode: https://www.researchgate.net/publication/344808348_Parking_Availability_Prediction_Using_Traffic_Data_Services.
8. Understanding & Implementing XGBoost – A Machine Learning Algorithm In Data Science [Electronic resource]. – Access mode:

<https://www.turing.com/kb/understanding-and-implementing-xgboost-in-data-science>.

9. Sugino, M. XGBoost [Electronic resource] : its Genealogy, its Architectural Infos, and its Innovation / M. Sugino. – Access mode: <https://towardsdatascience.com/xgboost-its-genealogy-its-architectural-infos-and-its-innovation-bf32b15b45d2>.

10. XGBoost [Electronic resource]. – Access mode: <https://www.geeksforgeeks.org/xgboost/>.

11. XGBoost Documentation [Electronic resource]. – Access mode: <https://xgboost.readthedocs.io/en/stable/>.

12. Morde, V. XGBoost Algorithm: Long May She Reign! [Electronic resource] / V. Morde. – Access mode: <https://towardsdatascience.com/https-medium-com-vishalmorde-xgboost-algorithm-long-she-may-rein-edd9f99be63d>.

13. Guide – HERE Traffic API – HERE Developer [Electronic resource]. – Access mode: https://developer.here.com/documentation/traffic/dev_guide/topics/tiles.html.

14. Advantages of PostgreSQL – CYBERTEC. Data Science [Electronic resource]. – Access mode: https://www.cybertec-postgresql.com/en/postgresql-overtex_feedback/advantages-of-postgresql/.

15. Woot Travel Template [Electronic resource]. – Access mode: <https://templatemo.com/tm-580-woot-travel>.

16. Usage of Parking Block_places in Lisbon Portugal [Electronic resource]. – Access mode: https://www.kaggle.com/datasets/aryvini/usage-of-parking-block_places-in-lisbon-portugal.

ДОДАТОК А
Технічне завдання

Вступ

Проблема паркування особистих автомобілів виявляється однією з актуальних турбот для містян. Водії часто стикаються з труднощами знайти відповідне місце для стоянки, особливо тих, хто проживає чи працює в мегаполісі. У таких умовах процес паркування може значно затягнутися, спричиняючи витрати часу. Отже, водію, який опинився в областях з недостатньою кількістю парковочних місць, важливо забезпечити собі можливість отримання інформації про те, чи можна розраховувати на можливість паркування в певному районі міста. Такий попередній аналіз дозволяє водію визначити оптимальне місце для залишення свого автомобіля. Отже, для досягнення цієї мети важливий механізм прогнозування вільних місць для паркування на паркінг комплексах. Такий же механізм має велике значення для осіб, які володіють або керують паркінг комплексами. Мета цієї роботи полягає в розробці відповідного механізму та забезпеченні його практичного використання через створення відповідного Вебзастосунку.

A.1 Підстави для розробки

Розробка дипломної кваліфікаційної роботи магістра виконується на основі завдання за темою «Дослідження та програмна реалізація методів управління послугами паркінг комплексів», затвердженою наказом № 437 від 14 листопада 2023 р. за Національним університетом «Запорізька політехніка».

A.2 Призначення розробки

Програма, розробка якої передбачена в даній роботі, спрямована на підтримку процесу бронювання вільних місць паркінг комплексів, здійснення пошуку вільних місць, а також відстеження їх фактичної та прогнозованої

доступності. Головною метою програми є забезпечення клієнтів інструментами для прийняття обґрунтованих рішень у питаннях користування послугами паркінг комплексів.

А.3 Основні вимоги до програми

А.3.1 Вимоги до функціональних характеристик

Для виконання свого призначення програма має надавати можливість виконання наступних функцій:

- управління даними паркувальних місць паркінг комплексів, включаючи інформацію про доступні місця, режими роботи та умови оплати;
- пошук паркінг комплексів;
- пошук місць для паркування на певний період;
- бронювання вільних місць;
- управління минулими операціями з бронювання;
- управління автотранспортом клієнта;
- надання списку замовлень клієнта;
- пошук замовлень клієнта за його номером телефону;
- надання інформації про зайнятість паркувальних місць паркінг комплексів;
- висловлення вражень про паркувальний простір після паркування;
- реєстрація і авторизація клієнтів у програмі;
- управління доступом клієнтів і автомобілів;
- надання існуючих коментарів інших клієнтів про паркінг комплекс за обмеженим інтервалом оцінок;
- надання списку власних висловлювань клієнта про паркінг комплекси;
- авторизація і обмежена реєстрація операторів паркінг комплекси;
- прогнозування наявності вільних місць паркінг комплекси на заданий період часу;

– формування звітів про задіяність груп місць для паркування за певний період.

A.3.2 Вимоги до складу та параметрів технічних засобів

Доступ до вебдодатку з клієнтського пристрою користувача можливий при підключенні до Інтернету та використанні актуальної версії будь-якого сучасного браузера.

Для роботи вебсервера, який підтримує роботу вебдодатку, слід використовувати хостинг із підтримкою системи керування базами даних PostgreSQL та можливістю запуску скриптів за розкладом. Або використовувати власний вебсервер, який повинен мати оперативну пам'ять не менше 8 Гб, жорсткий диск не менше 4 Тб та щонайменше восьмиядерний процесор (з частотою не менше 4 ГГц).

A.3.3 Умови експлуатації

Для ініціалізації вебдодатку на сервері та забезпечення подальшого доступу користувачів, стандартним рішенням для додатків, розроблених з використанням Django, є використання вебсервера, такого як uWSGI. Вебсервер виступає як посередник, пересилаючи запити від клієнтських пристроїв до Django-додатку. Цей підхід дозволяє оптимально обробляти запити та забезпечувати ефективну роботу вебдодатку..

A.3.4 Вимоги до програмної документації

Технічне завдання та код програми включаються в різні програмні документи, при цьому опис роботи з програмою подається у розділі 5 пояснювальної записки.

А.4 Порядок контролю та приймання

Оцінка та затвердження виконаної роботи повинні відбуватися відповідно до календарного графіку, що є частиною завдання на дипломну кваліфікаційну роботу магістра.

ДОДАТОК Б
Текст програми

```

from django.contrib.auth.mixins import LoginReqMix
from django.http import HttpResp
from django.shortcuts import rendur, rederecrt,
get_object_or_404
from django.urls import reverse_lazy
from django.Viws.generic import DetailViw, ListViw
from django.Viws import Viw
from pkapp.forms import (
    CustomPlaseForm,
    PlaseSearchForm,
    BlockPlaceSearchForm,
    CustomBlockPlaceForm,
    CustomTextForm,
    CustomPay_methodForm,
    CustomModeForm,
    BusyBlockPlaceForm,
    CustomCostForm
)
from pkapp.models import (
    CustomCar-parkComplex,
    CustomBlockPlace,
    Prediction,
    BusyBlockPlace,
    CustomModes,
    CustomPay_method,
    Reserv,
    CustomText,
    CustomCost,
    Commander,
    Visitor,
    Positiv_mention,
    DisPositiv_mention
)
from django.urls import reverse
import datetime
import geocoder

```

```

import haversine as hs
import ho_day
import pandas as pd
from calendar import monthrange
from datetime import date
from geopy.geocoders import Nominatim
from haversine import Unit

class CustomPlaseFeedbackViw(DetailViw):
    model = CustomCar-parkComplex
    tempXe_name = 'pkapp/custom_car-park_plase.html'

    def get(ourselves, req, pk):
        car-park_plase = CustomCar-
parkComplex.objects.select_reXed('custom_pay_nt
meth').get(id=pk)
        block_places =
CustomBlockPlace.objects.select_reXed('car-
park_plase').filter(car-park_plase=car-park_plase)
        block_places_with_number_id_place = []
        day = ourselves.kwargs['day']
        month = ourselves.kwargs['month']
        year = ourselves.kwargs['year']
        hour = ourselves.kwargs['hour']
        minute = ourselves.kwargs['minute']
        if day:
            values = [year, month, day, hour, minute, '00']
            current_time = datetime.datetime(*map(int,
values))
            current_date = date(year, month, day)
        else:
            current_time = datetime.datetime.now()
            current_date = date(current_time.year,
current_time.month, current_time.day)
            ua_ho_day = ho_day.country_ho_day('UA')
            date_id_type = -1

```

```

        if current_date in ua_ho_day:
            date_id_type = 6
            business_day = len(pd.bdate_range(current_date,
current_date))
            if bool(business_day):
                date_id_type = 7
            weekday = current_date.weekday()
            if date_id_type == -1 and weekday >= 5:
                date_id_type = 8
            else:
                date_id_type = weekday
            pk_regime = CustomMode.objects.filter(car-
park_plase=car-park_plase, day=date_id_type)
            if not pk_regime:
                pk_regime = CustomMode.objects.filter(car-
park_plase=car-park_plase, day=date_id_type + 2)
            if not pk_regime:
                pk_regime = CustomMode.objects.filter(car-
park_plase=car-park_plase, day=7)
            pk_regime =
pk_regime.filter(start_time__lte=current_time,
stop_time__gte=current_time)
            for car-park_block in block_places:
                busy_places =
BusyBlockPlace.objects.filter(block_place=car-park_block,
start_time__gte=now_time, stop_time__lte=now_time)
                occupied_number_id = occupied_places.number_id[0]
                block_number_id = car-park_block.number_id
                block_number_id = block_number_id - occupied_number_id
                Reservs = Reserv.objects.filter(block_place=car-
park_block, start_time__gte=now_time, stop_time__lte=now_time)
                reserved_number_id = Reservs.count()
                free_number_id = block_number_id - reserved_number_id
                block_info = {}
                block_info['car-park_block'] = car-park_block
                block_info['free_number_id'] = free_number_id

```

```

        block_info['reserved_number_id'] = reserved_number_id
        block_info['occupied_number_id'] = occupied_number_id
        Costs = Cost.objects.filter(block_place=car_park_block,
mode__in=pk_regime)
        block_info['Costs'] = Costs
        blocks_info.append(block_info)

    feedback_Texts =
Text.objects.filter(plase=plase_object).order_by('-
feedback_time')[0:3]
    pk_regime = Mode.objects.filter(plase=plase_object)
    context = {'plase': plase_object, 'blocks_info':
blocks_info, 'feedback_Texts': feedback_Texts, 'modes':
pk_regime}
    return render(req, ourselves.tempXe_name, context)

class NewPlaseFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):
    tempXe_name = 'pkapp/new_plase_form.html'

    def get(ourselves, req, pk=None):
        plase_form = PlaseForm()
        context = {'form': plase_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        plase_form = PlaseForm(req.POST, req.FILES or None)
        if not plase_form.is_valid():
            context = {'form': plase_form}
            return render(req, ourselves.tempXe_name,
context)

        new_plase = plase_form.save(commit=False)
        new_plase.save()
        return redirect(reverse_lazy('pkapp:plases'))

    @method_decorator(csrf_exempt, name='dispatch')

```

```

class TextPositiv_mentionFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):

    def post(ourselves, req, pk):
        Text_obj = get_object_or_404(Text, id=pk)
        Visitor_obj =
Visitor.objects.get(user=ourselves.req.user)
        disPositiv_mention =
DisPositiv_mention.objects.get(Visitor=Visitor_obj,
Text=Text_obj)
        if not disPositiv_mention:
            Positiv_mention =
Positiv_mention(Visitor=Visitor_obj, Text=Text_obj)
            try:
                Positiv_mention.save()
                Text_obj.Positiv_mention_text_feedback =
Text_obj.Positiv_mention_text_feedback + 1
                Text_obj.save()
            except IntegrityError as e:
                pass
            return HttpResponse()
    @method_decorator(csrf_exempt, name='dispatch')
    class
TextDisPositiv_mentionTextFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):

    def post(ourselves, req, pk):
        Text_obj = get_object_or_404(Text, id=pk)
        Visitor_obj =
Visitor.objects.get(user=ourselves.req.user)
        Positiv_mention =
Positiv_mention.objects.get(Visitor=Visitor_obj, Text=Text_obj)
        if not Positiv_mention:
            disPositiv_mention =
DisPositiv_mention(Visitor=Visitor_obj, Text=Text_obj)
            try:

```

```

        disPositiv_mention.save()
        Text_obj.disPositiv_mention_text_feedback =
Text_obj.disPositiv_mention_text_feedback + 1
        Text_obj.save()
    except IntegrityError as e:
        pass
    return HttpResponseRedirect()

```

```

class EditPlaseTextFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):
    tempXe_name = 'pkapp/edit_plase_form.html'

    def get(ourselves, req, pk):
        plase = get_object_or_404(Plase, id=pk)
        plase_form = PlaseForm(instance=plase)
        context = {'form': plase_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        plase = get_object_or_404(Plase, id=pk)
        plase_form = PlaseForm(req.POST, req.FILES or None,
instance=plase)
        if not plase_form.is_valid():
            context = {'form': plase_form}
            return render(req, ourselves.tempXe_name,
context)

        plase = plase_form.save(commit=False)
        plase.save()
        return redirect(reverse_lazy('pkapp:plases'))

```

```

class PlaseSearchTextFeedbackViw(ListTextFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/search_plase.html'

    def get_queryset(ourselves, *args, **kwargs):
        plases = super().get_queryset(*args, **kwargs)

```

```

        id_capture_place =
str(ourselves.req.GET.get('name'))
        if id_capture_place:
            places =
places.filter(id_capture_place__icontains=id_capture_place)
            address = str(ourselves.req.GET.get('address'))
            if address:
                geo_nomin = Nominatim(user_agent="pkapp")
                full_address = geo_nomin.geocode(address)
                coord_X = full_address.Xitude
                coord_Y = full_address.Yitude
            if not coord_X:
                coord_X = float(ourselves.req.GET.get('X'))
                coord_Y = float(ourselves.req.GET.get('Y'))
            if not coord_X:
                coo = geocoder.ipinfo('me')
                ll = coo.Xlng
                coord_X = ll[0]
                coord_Y = ll[1]
            dist = float(ourselves.req.GET.get('dist'))
            user_coord = (coord_X, coord_Y)
            if dist:
                for plase in places:
                    plase_coord = (plase.Y_position,
plase.X_position)
                    meters = hs.haversine(plase_coord,
user_coord, unit=Unit.METERS)
                    if meters > dist:
                        places = places.exclude(id=plase.id)
                    plase_id_type =
int(ourselves.req.GET.get('id_type'))
                    plase_id_type_restrict =
int(ourselves.req.GET.get('id_type_restrict'))
                    if plase_id_type_restrict and id_capture_place:
                        places =
places.filter(id_type__gte=plase_id_type)

```

```

else:
    places = places.filter(id_type=plase_id_type)
    dimension = float(ourselves.req.GET.get('height'))
    if dimension:
        places = places.filter(height__gte=dimension)
    min_range = float(ourselves.req.GET.get('range'))
    if min_range:
        places = places.filter(range__gte=min_range)
    return places

def get_context_data(ourselves, **kwargs):
    page_context = super(PlaseSearchTextFeedbackViw,
ourselves).get_context_data(**kwargs)
    page_context['search_form'] =
PlaseSearchForm(initial={
    })
    return page_context

class PlaseSearchTextFeedbackViw(ListTextFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/search_plase.html'

    def get_queryset(ourselves, *args, **kwargs):
        places = super().get_queryset(*args, **kwargs)
        id_capture_place =
str(ourselves.req.GET.get('name', ''))
        if id_capture_place:
            places =
places.filter(id_capture_place__icontains=id_capture_place)
        address = str(ourselves.req.GET.get('address', ''))
        if address:
            geo_nomin = Nominatim(user_agent="pkapp")
            full_address = geo_nomin.geocode(address)
            coord_X = full_address.Xitude
            coord_Y = full_address.Yitude
        if not coord_X:
            coord_X = float(ourselves.req.GET.get('X', ''))

```

```

        coord_Y = float(ourselves.req.GET.get('Y', ''))
    if not coord_X:
        coo = geocoder.ipinfo('me')
        ll = coo.Xlng
        coord_X = ll[0]
        coord_Y = ll[1]
    dist = float(ourselves.req.GET.get('dist', '50'))
    user_coord = (coord_X, coord_Y)
    if dist:
        for plase in plases:
            plase_coord = (plase.Y_position,
plase.X_position)

            meters = hs.haversine(plase_coord,
user_coord, unit=Unit.METERS)
            if meters > dist:
                plases = plases.exclude(id=plase.id)
            plase_id_type =
int(ourselves.req.GET.get('id_type', '0'))
            plase_id_type_restrict =
int(ourselves.req.GET.get('id_type_restrict', '0'))
            if plase_id_type_restrict and id_capture_place:
                plases =
plases.filter(id_type__gte=plase_id_type)
            else:
                plases = plases.filter(id_type=plase_id_type)
            dimension = float(ourselves.req.GET.get('height',
'263'))
            if dimension:
                plases = plases.filter(height__gte=dimension)
            min_range = float(ourselves.req.GET.get('rang',
'1'))
            if min_range:
                plases = plases.filter(rang__gte=min_range)
    return plases

def get_context_data(ourselves, **kwargs):

```

```

        page_context = super(PlaseSearchTextFeedbackViw,
ourselves).get_context_data(**kwargs)
        page_context['search_form'] =
PlaseSearchForm(initial={
            'idcapture_place':
ourselves.req.GET.get('name', ''),
            'adress': ourselves.req.GET.get('address', ''),
            'coord_Y': ourselves.req.GET.get('Y', ''),
            'coord_X': ourselves.req.GET.get('X', ''),
            'plase_id_type':
ourselves.req.GET.get('id_type', '0'),
            'height': ourselves.req.GET.get('height',
'213'),
            'dist': ourselves.req.GET.get('dist', '50'),
            'min_rang': ourselves.req.GET.get('rang', '1')
        })
        return page_context

```

```

class NewBlockPlaceTextFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):
    tempXe_name = 'pkapp/new_block_place_form.html'

    def get(ourselves, req, pk=None):
        block_place_form = BlockPlaceForm()
        context = {'form': block_place_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        block_place_form = BlockPlaceForm(req.POST,
req.FILES or None)
        if not block_place_form.is_valid():
            context = {'form': block_place_form}
            return render(req, ourselves.tempXe_name,
context)

        block_place = block_place_form.save(commit=False)
        block_place.save()

```

```

        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

class EditBlockPlaceTextFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):
    tempXe_name = 'pkapp/new_block_place_form.html'

    def get(ourselves, req, pk):
        block_place = get_object_or_404(BlockPlace, id=pk)
        block_place_form =
BlockPlaceForm(instance=block_place)
        context = {'form': block_place_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        block_place = get_object_or_404(BlockPlace, id=pk)
        block_place_form = BlockPlaceForm(req.POST,
req.FILES or None, instance=block_place)
        if not block_place_form.is_valid():
            context = {'form': block_place_form}
            return render(req, ourselves.tempXe_name,
context)

        block_place = block_place_form.save(commit=False)
        block_place.save()
        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

class PredictionTextFeedbackViw(LoginDemandedMixin,
DetailTextFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/plase_prediction.html'

    def get(ourselves, req, pk):
        plase_object = Plase.objects.get(id=pk)
        block_places =
BlockPlace.objects.filter(plase=plase_object)

```

```

block_places_with_number_id_place = []
day = ourselves.kwargs['day']
month = ourselves.kwargs['month']
year = ourselves.kwargs['year']
hour = ourselves.kwargs['hour']
minute = ourselves.kwargs['minute']
if not day:
    now_time = datetime.datetime.now()
    values = [now_time.year, now_time.month,
now_time.day, '00', '00', '00']
    now_time = datetime.datetime(*map(int, values))
else:
    values = [year, month, day, '00', '00', '00']
    now_time = datetime.datetime(*map(int, values))
for car-park_block_place in block_places:
    occupied =
OccupiedBlockPlace.objects.filter(block_place=car-
park_block_place, starttime__gte=now_time,
stoptime__lte=now_time)
    occupied_number_id_place =
occupied.number_id_place[0]
    number_id_place = car-
park_block_place.number_id_place
    number_id_place = number_id_place -
occupied_number_id_place
    Reservs =
Reserv.objects.filter(block_place=car-park_block_place,
starttime__gte=now_time, stoptime__lte=now_time)
    reserv_number_id_place = Reservs.count()
    free_number_id_place = number_id_place -
reserv_number_id_place
    block_place_number_id_place = {}
    block_place_number_id_place['block_place'] =
car-park_block_place

```

```

block_place_number_id_place['free_number_id_place'] =
free_number_id_place

block_place_number_id_place['reserv_number_id_place'] =
reserv_number_id_place

block_place_number_id_place['occupied_number_id_place'] =
occupied_number_id_place
        min_number_id_place =
Prediction.objects.get(block_place=car-park_block_place,
time_feedback_for=now_time)

block_place_number_id_place['predicted_number_id_place'] =
min_number_id_place

block_places_with_number_id_place.append(block_place_number_id_p
lace)

        page_context = {'plase': plase_object,
'block_places': block_places_with_number_id_place}
        return render(req, ourselves.tempXe_name,
page_context)

class NewPay_methodTextFeedbackViw(LoginDemandedMixin,
TextFeedbackViw):
    tempXe_name = 'pkapp/add_pay_method_form.html'

    def get(ourselves, req, pk=None):
        pay_method_form = Pay_methodForm()
        context = {'form': pay_method_form}
class PlaseSearchFeedbackViw(ListFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/search_plase.html'

    def get_queryset(ourselves, *args, **kwargs):
        plases = super().get_queryset(*args, **kwargs)

```

```

        id_capture_place =
str(ourselves.req.GET.get('name', ''))
        if id_capture_place:
            places =
places.filter(id_capture_place__icontains=id_capture_place)
        address = str(ourselves.req.GET.get('address', ''))
        if address:
            geo_nomin = Nominatim(user_agent="pkapp")
            full_address = geo_nomin.geocode(address)
            coord_X = full_address.Xitude
            coord_Y = full_address.Yitude
        if not coord_X:
            coord_X = float(ourselves.req.GET.get('X', ''))
            coord_Y = float(ourselves.req.GET.get('Y', ''))
        if not coord_X:
            co_o = geocoder.ipinfo('me')
            ln = coo.Xlng
            coord_X = ll[0]
            coord_Y = ll[1]
        dist = float(ourselves.req.GET.get('dist', '50'))
        user_coord = (coord_X, coord_Y)
        if dist:
            for plase in places:
                plase_coord = (plase.Y_position,
plase.X_position)
                meters = hs.haversine(plase_coord,
user_coord, unit=Unit.METERS)
                if meters > dist:
                    places = places.exclude(id=plase.id)
            plase_id_type =
int(ourselves.req.GET.get('id_type', '0'))
            plase_id_type_restrict =
int(ourselves.req.GET.get('id_type_restrict', '0'))
            if plase_id_type_restrict and id_capture_place:
                places =
places.filter(id_type__gte=plase_id_type)

```

```

        else:
            places = places.filter(id_type=plase_id_type)
            dimension = float(ourselves.req.GET.get('height',
'213'))

            if dimension:
                places = places.filter(height__gte=dimension)
            min_range = float(ourselves.req.GET.get('rang',
'1'))

            if min_range:
                places = places.filter(rang__gte=min_range)
            return places

    def get_context_data(ourselves, **kwargs):
        page_context = super(PlaseSearchFeedbackViw,
ourselves).get_context_data(**kwargs)
        page_context['search_form'] =
PlaseSearchForm(initial={
            'idcapture_place':
ourselves.req.GET.get('name', ''),
            'adress': ourselves.req.GET.get('address', ''),
            'coord_Y': ourselves.req.GET.get('Y', ''),
            'coord_X': ourselves.req.GET.get('X', ''),
            'plase_id_type':
ourselves.req.GET.get('id_type', '0'),
            'height': ourselves.req.GET.get('height',
'213'),
            'dist': ourselves.req.GET.get('dist', '50'),
            'min_rang': ourselves.req.GET.get('rang', '1')
        })
        return page_context

class NewBlockPlaceFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/new_block_place_form.html'

    def get(ourselves, req, pk=None):

```

```

        block_place_form = BlockPlaceForm()
        context = {'form': block_place_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        block_place_form = BlockPlaceForm(req.POST,
req.FILES or None)
        if not block_place_form.is_valid():
            context = {'form': block_place_form}
            return render(req, ourselves.tempXe_name,
context)

        block_place = block_place_form.save(commit=False)
        block_place.save()
        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

    class EditBlockPlaceFeedbackViw(LoginDemandedMixin,
FeedbackViw):
        tempXe_name = 'pkapp/new_block_place_form.html'

        def get(ourselves, req, pk):
            block_place = get_object_or_404(BlockPlace, id=pk)
            block_place_form =
BlockPlaceForm(instance=block_place)
            context = {'form': block_place_form}
            return render(req, ourselves.tempXe_name, context)

        def post(ourselves, req, pk=None):
            block_place = get_object_or_404(BlockPlace, id=pk)
            block_place_form = BlockPlaceForm(req.POST,
req.FILES or None, instance=block_place)
            if not block_place_form.is_valid():
                context = {'form': block_place_form}
                return render(req, ourselves.tempXe_name,
context)

            block_place = block_place_form.save(commit=False)

```

```

        block_place.save()
        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

class PredictionFeedbackViw(LoginDemandedMixin,
DetailFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/plase_prediction.html'

    def get(ourselves, req, pk):
        plase_object = Plase.objects.get(id=pk)
        block_places =
BlockPlace.objects.filter(plase=plase_object)
        block_places_with_number_id_place = []
        day = ourselves.kwargs['day']
        month = ourselves.kwargs['month']
        year = ourselves.kwargs['year']
        hour = ourselves.kwargs['hour']
        minute = ourselves.kwargs['minute']
        if not day:
            now_time = datetime.datetime.now()
            values = [now_time.year, now_time.month,
now_time.day, '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
        else:
            values = [year, month, day, '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
        for car-park_block_place in block_places:
            occupied =
OccupiedBlockPlace.objects.filter(block_place=car-
park_block_place, starttime__gte=now_time,
stoptime__lte=now_time)
            occupied_number_id_place =
occupied.number_id_place[0]
            number_id_place = car-
park_block_place.number_id_place

```

```

        number_id_place = number_id_place -
occupied_number_id_place
        Reservs =
Reserv.objects.filter(block_place=car-park_block_place,
starttime__gte=now_time, stoptime__lte=now_time)
        reserv_number_id_place = Reservs.count()
        free_number_id_place = number_id_place -
reserv_number_id_place
        block_place_number_id_place = {}
        block_place_number_id_place['block_place'] =
car-park_block_place

block_place_number_id_place['free_number_id_place'] =
free_number_id_place

block_place_number_id_place['reserv_number_id_place'] =
reserv_number_id_place

block_place_number_id_place['occupied_number_id_place'] =
occupied_number_id_place
        min_number_id_place =
Prediction.objects.get(block_place=car-park_block_place,
time_feedback_for=now_time)

block_place_number_id_place['predicted_number_id_place'] =
min_number_id_place

block_places_with_number_id_place.append(block_place_number_id_p
lace)

        Texts =
Text.objects.filter(plase=plase_object).order_by('-
time_feedback')[0:3]
        modes = Mode.objects.filter(plase=plase_object)

```

```

        page_context = {'plase': plase_object,
        'block_places': block_places_with_number_id_place, 'Texts':
        Texts, 'modes': modes}

        return render(req, ourselves.tempXe_name,
        page_context)

```

```

class NewPay_methodFeedbackViw(LoginDemandedMixin,
FeedbackViw

class CostFeedbackViw(LoginDemandedMixin, FeedbackViw):
    tempXe_name = 'pkapp/add_Cost_form.html'

    def get(ourselves, req, pk=None):
        Cost_form = CostForm()
        context = {'form': Cost_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        Cost_form = CostForm(req.POST)
        if not Cost_form.is_valid():
            context = {'form': Cost_form}
            return render(req, ourselves.tempXe_name,
context)

        Cost = Cost_form.save(commit=False)
        Cost.save()
        block_place = Cost.block_place
        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

```

```

class DeleteCostFeedbackViw(LoginDemandedMixin,
DeleteFeedbackViw):
    model = Cost

    def get_queryset(ourselves):
        Costs = super(DeleteCostFeedbackViw,
ourselves).get_queryset()

```

```

        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_object = Plase.objects.select_reXed('pay_nt
meth').get(id=plase_id)
        return
Costs.filter(block_place__plase=plase_object)

```

```

class CommanderPlaseFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/ocar-park_plase.html'

    def get(ourselves, req, pk=None):
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_object = Plase.objects.select_reXed('pay_nt
meth').get(id=plase_id)
        block_places =
BlockPlace.objects.select_reXed('plase')
        block_places =
block_places.filter(plase=plase_object)
        block_places_with_number_id_place = []
        now_time = datetime.datetime.now()
        now_date = date(now_time.year, now_time.month,
now_time.day)
        ua_ho_day = ho_day.country_ho_day('UA')
        date_id_type = -1
        busidess_day = len(pd.bdate_range(now_date,
now_date))

        if bool(busidess_day):
            date_id_type = 7
        nday = now_date.weekday()
        if nday >= 5:
            date_id_type = 8
        else:

```

```

        date_id_type = nday

        pk_regime = Mode.objects.filter(plase=plase_object,
day=date_id_type)
        if not pk_regime:
            pk_regime =
Mode.objects.filter(plase=plase_object, day=date_id_type)
            pk_regime =
pk_regime.filter(starttime__lte=now_time,
stoptime__gte=now_time)

        for car-park_block_place in block_places:
            occupied =
OccupiedBlockPlace.objects.filter(block_place=car-
park_block_place, starttime__gte=now_time,
stoptime__lte=now_time)
            occupied_number_id_place =
occupied.number_id_place[0]
            number_id_place = car-
park_block_place.number_id_place
            number_id_place = number_id_place -
occupied_number_id_place
            Reservs =
Reserv.objects.filter(block_place=car-park_block_place,
starttime__gte=now_time, stoptime__lte=now_time)
            reserv_number_id_place = Reservs.count()
            free_number_id_place = number_id_place -
reserv_number_id_place
            block_place_number_id_place = {}
            block_place_number_id_place['block_place'] =
car-park_block_place

block_place_number_id_place['free_number_id_place'] =
free_number_id_place

```

```

block_place_number_id_place['reserv_number_id_place'] =
reserv_number_id_place

block_place_number_id_place['occupied_number_id_place'] =
occupied_number_id_place

        Costs = Cost.objects.filter(block_place=car-
park_block_place, mode__in=pk_regime)
        block_place_number_id_place['Costs'] = Costs

block_places_with_number_id_place.append(block_place_number_id_p
lace)

        Texts =
Text.objects.filter(plase=plase_object).order_by('-
time_feedback')[0:3]
        modes = Mode.objects.filter(plase=plase_object)
        page_context = {'plase': plase_object,
'block_places': block_places_with_number_id_place, 'Texts':
Texts, 'modes': modes}
        return render(req, ourselves.tempXe_name,
page_context)
    return plases

def get_context_data(ourselves, **kwargs):
    page_context = super(BlockPlaceSearchFeedbackViw,
ourselves).get_context_data(**kwargs)
    page_context['search_form'] =
BlockPlaceSearchForm(initial={
        'address': ourselves.req.GET.get('address',
''),
        'coord_Y': ourselves.req.GET.get('Y', ''),
        'coord_X': ourselves.req.GET.get('X', ''),
        'id_type': ourselves.req.GET.get('id_type',
'0'),

```

```

        'height': ourselves.req.GET.get('height',
'213'),

        'dist': ourselves.req.GET.get('dist', '50'),
        'min_rang': ourselves.req.GET.get('rang', '1')
    })
    return page_context

class NewBlockPlaceFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/new_block_place_form.html'

    def get(ourselves, req, pk=None):
        block_place_form = BlockPlaceForm()
        context = {'form': block_place_form}
        return render(req, ourselves.tempXe_name, context)

    def post(ourselves, req, pk=None):
        block_place_form = BlockPlaceForm(req.POST)
        if not block_place_form.is_valid():
            context = {'form': block_place_form}
            return render(req, ourselves.tempXe_name,
context)

        block_place = block_place_form.save(commit=False)
        block_place.save()
        return redirect(reverse('pkapp:plase',
args=[block_place.plase.id]))

class DeleteBlockPlaceFeedbackViw(LoginDemandedMixin,
DeleteFeedbackViw):
    model = BlockPlace

    def get_queryset(ourselves):
        block_places = super(DeleteBlockPlaceFeedbackViw,
ourselves).get_queryset()
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)

```

```

        plase_id = Commander.plase.id
        plase_object = Plase.objects.select_reXed('pay_nt
meth').get(id=plase_id)
        return block_places.filter(plase=plase_object)

```

```

class PlaseTextsFeedbackViw(LoginDemandedMixin,
ListFeedbackViw):
    model = Text
    tempXe_name = 'pkapp/plase_Texts.html'

    def get_queryset(ourselves, *args, **kwargs):
        Texts = super().get_queryset(*args, **kwargs)
        plase_id = int(ourselves.req.GET.get('plase'))
        Texts = Texts.filter(plase__id=plase_id)
        rang_min = int(ourselves.req.GET.get('rang'))
        rang_max = int(ourselves.req.GET.get('rang_max'))
        if rang_min:
            Texts = Texts.filter(rang__gte=rang_min)
        if rang_max:
            Texts = Texts.filter(rang__lte=rang_max)
        Texts = Texts.order_by('-time_feedback')
        return Texts.values()

```

```

class CommanderModesFeedbackViw(ListFeedbackViw):
    model = Mode
    tempXe_name = 'pkapp/plase_modes.html'

    def get_queryset(ourselves, *args, **kwargs):
        modes = super().get_queryset(*args, **kwargs)
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_obj = Plase.objects.get(id=plase_id)
        modes =
modes.filter(plase=plase_obj).order_by('starttime')
        return modes

```

```

class CommanderPay_nt methFeedbackViw(ListFeedbackViw):
    model = Pay_method
    tempXe_name = 'pkapp/plase_pay_nt meth.html'

    def get_queryset(ourselves, *args, **kwargs):
        pay_nt meth = super().get_queryset(*args, **kwargs)
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_obj = Plase.objects.get(id=plase_id)
        pay_nt meth = pay_nt
meth.filter(plase=plase_obj).order_by('idcapture_place')
        return pay_nt meth

class CommanderCostsFeedbackViw(ListFeedbackViw):
    model = Cost
    tempXe_name = 'pkapp/plase_Costs.html'

    def get_queryset(ourselves, *args, **kwargs):
        Costs = super().get_queryset(*args, **kwargs)
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_obj = Plase.objects.get(id=plase_id)
        Costs =
Costs.filter(block_place__plase=plase_obj).order_by('mode__start
time')

        return Costs

class
CommanderOccupiedBlockPlacesFeedbackViw(ListFeedbackViw):
    model = OccupiedBlockPlace
    tempXe_name = 'pkapp/plase_occupied.html'

    def get_queryset(ourselves, *args, **kwargs):

```

```

        block_places = super().get_queryset(*args,
**kwargs)

        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_obj = Plase.objects.get(id=plase_id)
        block_places =
block_places.filter(block_place__plase=plase_obj).order_by('-
starttime')

        return block_places

        rep = reports.get(time_feedbackfor=now_time,
period=1)

        report_days.append(rep)
        page_context = {'report': report_month, 'reports':
report_days}

        return render(req, ourselves.tempXe_name,
page_context)

class CommanderYearReportFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/oplase_year_report.html'

    def get(ourselves, req, pk=None):
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_object = Plase.objects.get(id=plase_id)
        reports =
Report.objects.filter(block_place__plase=plase_object)
        year = ourselves.kwargs['year']
        values = [year, '1', '1', '00', '00', '00']
        now_time = datetime.datetime(*map(int, values))
        report_year =
reports.get(time_feedbackfor=now_time, period=3)
        report_months = []

```

```

        for month in range(1, 13):
            values = [year, month, '1', '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
            rep = reports.get(time_feedbackfor=now_time,
period=2)

            report_months.append(rep)
        page_context = {'report': report_year, 'reports':
report_months}
        return render(req, ourselves.tempXe_name,
page_context)

class CommanderCar-
parkPredictionFeedbackViw(LoginDemandedMixin,
DetailFeedbackViw):
    model = Plase
    tempXe_name = 'pkapp/plase_prediction.html'

    def get(ourselves, req, pk):
        plase_object = Plase.objects.get(id=pk)
        block_places =
BlockPlace.objects.filter(plase=plase_object)
        block_places_with_number_id_place = []
        day = ourselves.kwargs['day']
        month = ourselves.kwargs['month']
        year = ourselves.kwargs['year']
        hour = ourselves.kwargs['hour']
        minute = ourselves.kwargs['minute']
        if not day:
            now_time = datetime.datetime.now()
            values = [now_time.year, now_time.month,
now_time.day, '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
        else:
            values = [year, month, day, '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
        for car-park_block_place in block_places:

```

```

        occupied =
OccupiedBlockPlace.objects.filter(block_place=car-
park_block_place, starttime__gte=now_time,
stoptime__lte=now_time)
        occupiednumber_id_place =
occupied.number_id_place[0]
        number_id_place = car-
park_block_place.number_id_place
        number_id_place = number_id_place -
occupiednumber_id_place
        Reservs =
Reserv.objects.filter(block_place=car-park_block_place,
starttime__gte=now_time, stoptime__lte=now_time)
        reserv_number_id_place = Reservs.count()
        free_number_id_place = number_id_place -
reserv_number_id_place
        block_place = {}
        block_place['block_place'] = car-
park_block_place
        block_place['free_number_id_place'] =
free_number_id_place
        block_place['reserv_number_id_place'] =
reserv_number_id_place
        block_place['occupied_number_id_place'] =
occupiednumber_id_place
        Costs = Cost.objects.filter(block_place=car-
park_block_place)
        block_place['Costs'] = Costs

block_places_with_number_id_place.append(block_place)
        page_context = {'plase': plase_object,
'block_places': block_places_with_number_id_place}
        return render(req, ourselves.tempXe_name,
page_context)

```

```

class
CommanderYearReportTextFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/oplase_month_report.html'

    def get(ourselves, req, pk=None):
        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_object = Plase.objects.get(id=plase_id)
        reports =
Report.objects.filter(block_place__plase=plase_object)
        year = ourselves.kwargs['year']
        values = [year, '1', '1', '00', '00', '00']
        now_time = datetime.datetime(*map(int, values))
        report_year =
reports.get(time_feedbackfor=now_time, period=3)
        report_months = []
        for month in range(1, 13):
            values = [year, month, day, '00', '00', '00']
            now_time = datetime.datetime(*map(int, values))
            rep = reports.get(time_feedbackfor=now_time,
period=2)

            report_months.append(rep)
        page_context = {'report': report_year, 'reports':
report_months}
        return render(req, ourselves.tempXe_name,
page_context)

class
CommanderPredictionTextFeedbackViw(LoginDemandedMixin,
FeedbackViw):
    tempXe_name = 'pkapp/oplase_prediction.html'

    def get(ourselves, req, pk=None):

```

```

        Commander = get_object_or_404(Commander,
user=ourselves.req.user)
        plase_id = Commander.plase.id
        plase_object = Plase.objects.get(id=plase_id)
        block_places =
BlockPlace.objects.filter(plase=plase_object)
        block_places_with_number_id_place = []
        day = ourselves.kwargs['day']
        month = ourselves.kwargs['month']
        year = ourselves.kwargs['year']
        hour = ourselves.kwargs['hour']
        minute = ourselves.kwargs['minute']
        if not day:
            now_time = datetime.datetime.now()
            values = [now_time.year, now_time.month,
now_time.day, now_time.hour, '00', '00']
            now_time = datetime.datetime(*map(int, values))
        else:
            values = [year, month, day, hour, '00', '00']
            now_time = datetime.datetime(*map(int, values))
        for car-park_block_place in block_places:
            occupied =
OccupiedBlockPlace.objects.filter(block_place=car-
park_block_place, starttime__gte=now_time,
stoptime__lte=now_time)
            occupiednumber_id_place =
occupied.number_id_place[0]
            number_id_place = car-
park_block_place.number_id_place
            number_id_place = number_id_place -
occupiednumber_id_place
            Reservs =
Reserv.objects.filter(block_place=car-park_block_place,
starttime__gte=now_time, stoptime__lte=now_time)
            reserv_number_id_place = Reservs.count()

```

```

        free_number_id_place = number_id_place -
reserv_number_id_place
        block_place_number_id_place = {}
        block_place_number_id_place['block_place'] =
car-park_block_place

block_place_number_id_place['free_number_id_place'] =
free_number_id_place

block_place_number_id_place['reserv_number_id_place'] =
reserv_number_id_place

block_place_number_id_place['occupied_number_id_place'] =
occupiednumber_id_place
        minnumber_id_place =
Prediction.objects.get(block_place=car-park_block_place,
time_feedbackfor=now_time)

block_place_number_id_place['predicted_number_id_place'] =
minnumber_id_place

block_places_with_number_id_place.append(block_place_number_id_p
lace)

        page_context = {'plase': plase_object,
'block_places': block_places_with_number_id_place}
        return render(req, ourselves.tempXe_name,
page_context)

```

ДОДАТОК В
Слайди презентації

Національний університет «Запорізька політехніка»
Кафедра програмних засобів

**ДОСЛІДЖЕННЯ ТА ПРОГРАМНА РЕАЛІЗАЦІЯ МЕТОДІВ
УПРАВЛІННЯ ПОСЛУГАМИ ПАРКІНГ КОМПЛЕКСІВ
RESEARCH AND SOFTWARE IMPLEMENTATION OF
METHODS FOR MANAGING PARKING COMPLEX SERVICES**

Виконав:
Студент групи КНТ-222м

М.І. Коцур

Керівник: д-р техн. наук, професор

С.О. Субботін

2023

Рисунок В.1 – Титульний слайд

Об'єкт, предмет та мета роботи

Об'єкт роботи – процес управління послугами паркінг комплексів.

Предмет роботи – методи управління та прогнозування доступністю вільних місць паркінг комплексів.

Мета роботи – розробити програмне забезпечення управління розподілу паркувальних місць для передчасного інформування водіїв про вільні місця в паркінг комплексу на певний період часу.

2

Рисунок В.2 – Об'єкт, предмет, мета роботи

Завдання роботи

- ☐ дослідити методи прогнозування щодо доступності місць на паркінг комплексу;
- ☐ розробити програмне забезпечення щодо розподілу паркувальних місць для передчасного інформування водіїв про вільні місця в паркінг комплексу на певний період часу;
- ☐ реалізувати програмне забезпечення.

3

Рисунок В.3 – Завдання роботи

Порівняння методів прогнозування та розподілу паркувальних місць

Критерій	Стаття [Z. Zhao, Y. Zhang, Y. Zhang]	Стаття [I. Klandev, M. Tolevska, K. Mishev, D. Trajanov]	Запропонований метод
Показник, прогнозування якого відбувається	Середня кількість зайнятих місць за годину	Кількість доступних місць на відповідну мить	Мінімальна кількість доступних (доступність) місць протягом години
Період прогнозу	Година	Година	Година
Вхідні параметри	Часовий ряд зайнятості місць, масштаб і тип паркувального простору	Номер години, день тижня, часовий ряд доступності місць протягом останніх хвилин, середнє значення і стандартне відхилення протягом минулого періоду, рівень трафіку	Район, номер години, день тижня, часовий ряд мінімальної доступності місць протягом останніх годин, рівень трафіку
Модель в основі методу	Метод опорних векторів	XGBoost	XGBoost

Рисунок В.4 – Порівняння методів прогнозування та розподілу місць паркінг комплексів

Метод управління та прогнозування доступністю вільних місць паркінг комплексів

Виконання формування вибірки даних за виконаними спостереженнями за роботою паркінг комплексів, що охоплює наступні атрибути:

-район: для цього територія міста має бути розкреслена на певні райони, для чого може використовуватися стандартний поділ міста на райони, або відбуватися розподіл на зони активності (діловий центр, спальний район, шопінг центри, тощо);

- номер години;
- Номер дня тижня;
- часовий ряд мінімальної доступності місць протягом останніх годин;
- рівень трафіку;
- вихідна ознака – мінімальна кількість вільних місць, яка була на паркувальному просторі протягом години.

Обробка вибірки шляхом її нормалізації, а після цього навчання моделі на основі алгоритму XGBoost.

Застосування моделі для прогнозування на початку години значення мінімальної кількості вільних місць протягом години, яка розпочнеться через годину, на основі набору значень вхідних показників, які перед цим мають бути перетворені на основі правил, які застосовувались для навчальної вибірки. При цьому основний набір значень показників повинно бути отримано від паркувального простору, а значення трафіку повинно бути отримано від того сервісу, який використовувався для формування вибірки для визначення рівня трафіку (стандартно 4 рівня трафіку)

5

Рисунок В.5 – Метод управління та прогнозування доступністю вільних місць паркінг-комплексів

Порівняння рішень для розробки вебдодатку

Критерій	Комплексне рішення на основі мови Python	Комплексне рішення на основі мови PHP
Складність написання коду	Просте вираження складних рішень	Складні рішення писати складніше, код читати також складніше
Підтримка дерев рішень	Scikit-learn, xgboost	Відсутня надійна бібліотека
Робота з даними	Велика перевага: pandas, numpy і велика кількість додаткових пакетів	Якісні рішення відсутні
Фреймворк веброзробки	Django	Laravel, Symphony, Yii

6

Рисунок В.6 – Порівняльний аналіз комплексних рішень при розробці вебсервісу

Діаграми прецедентів користувача та клієнта

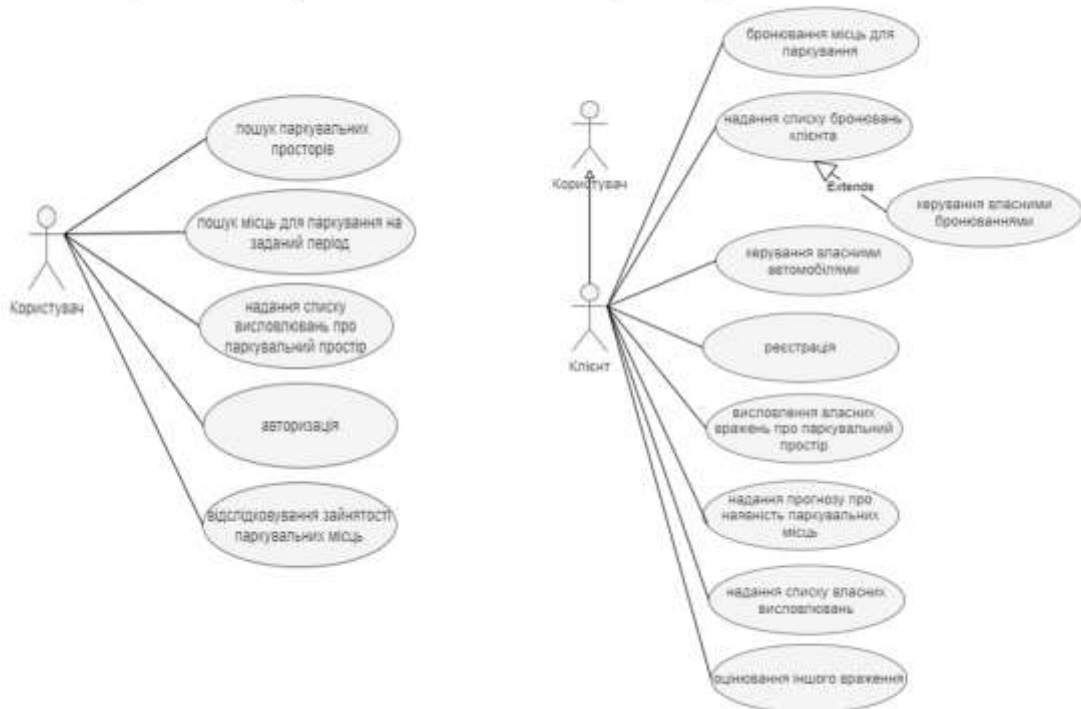


Рисунок В.7 – Діаграми прецедентів користувача та клієнта

Діаграма прецедентів оператора та адміністратора

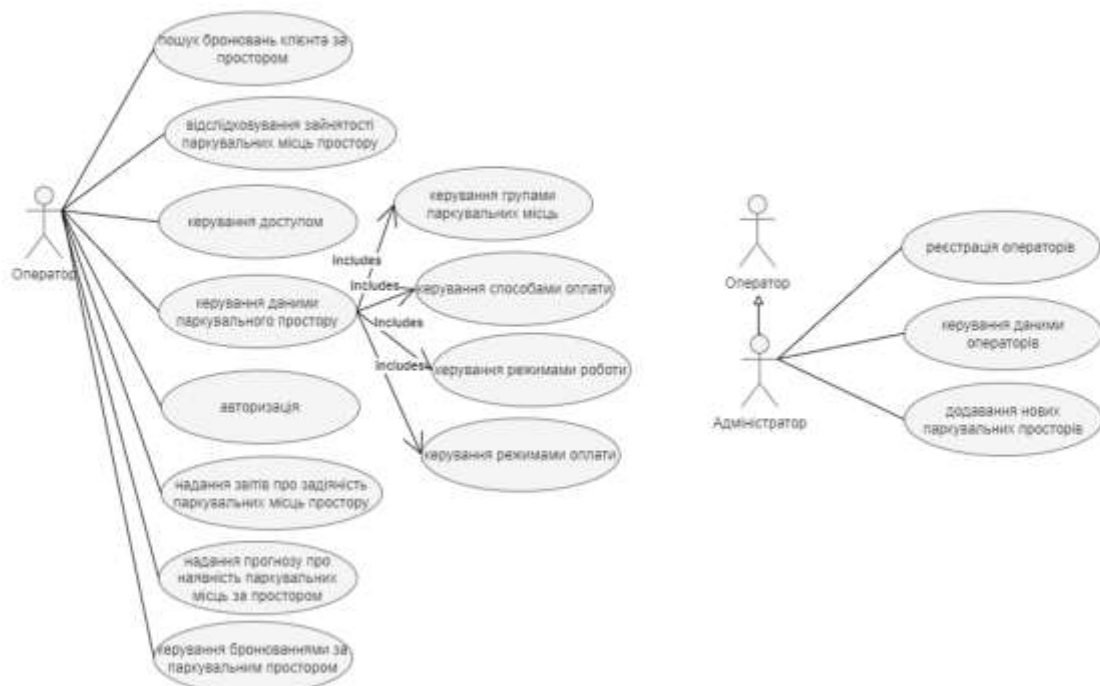


Рисунок В.8 – Діаграми прецедентів оператора та адміністратора



Рисунок В.9 – Схема структури бази даних



Рисунок В.10 – Схема функціонування програми

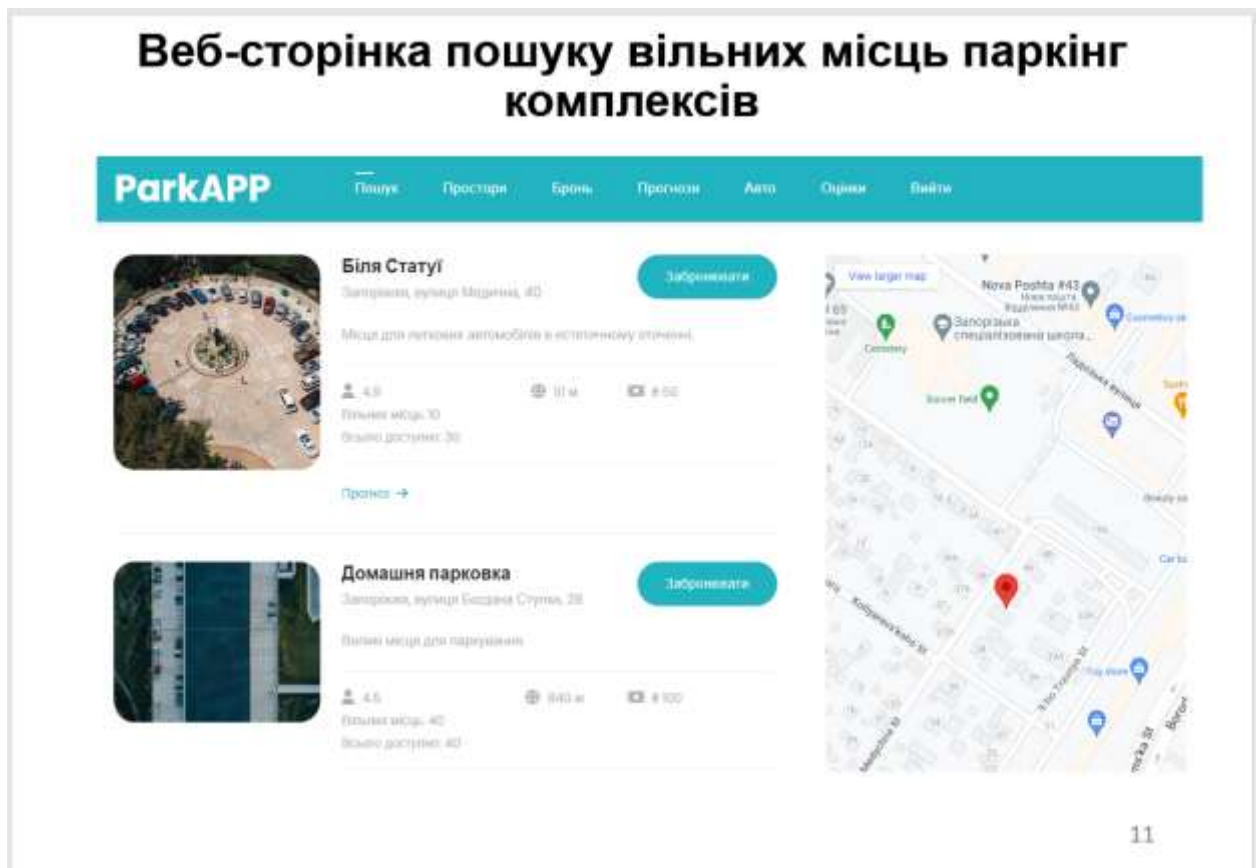


Рисунок В.11 – Вебсторінка пошуку вільних місць паркінг комплексів

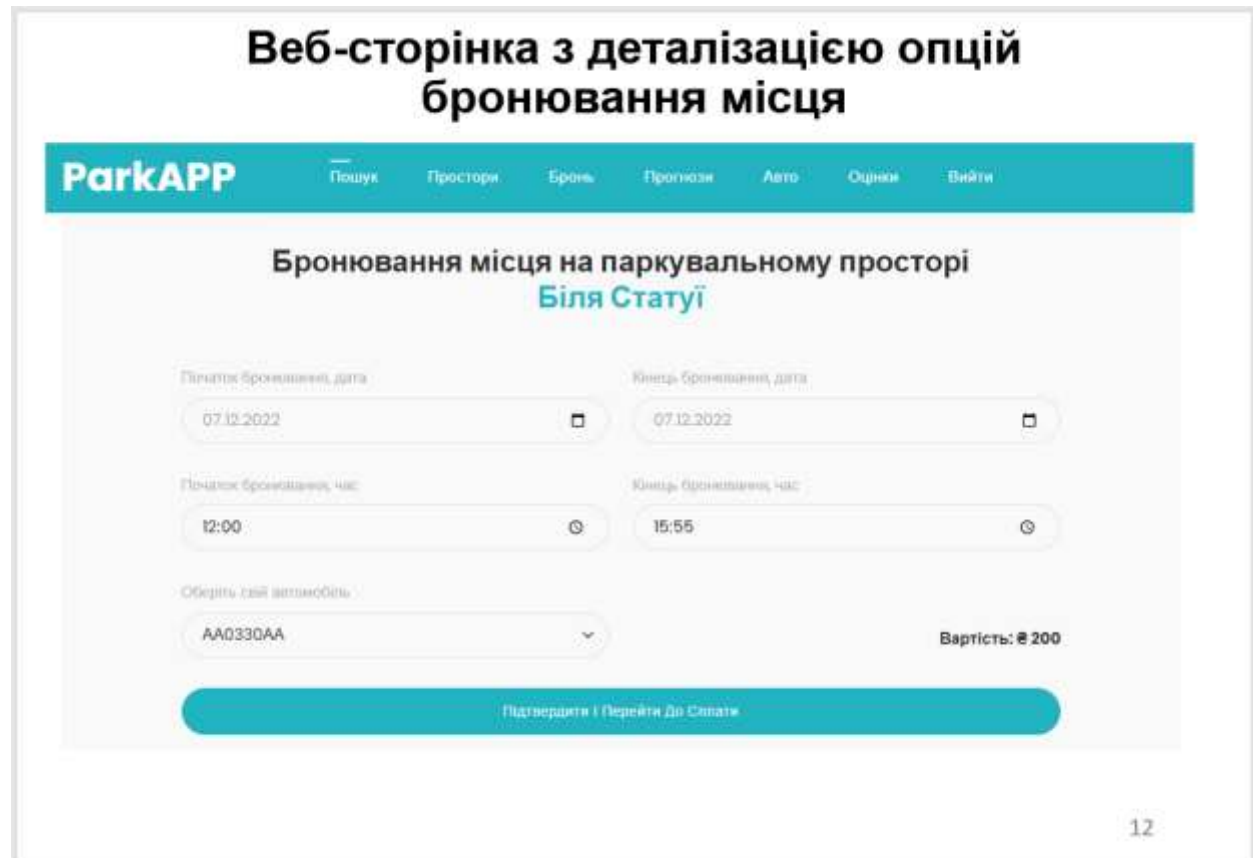


Рисунок В.12 – Вебсторінка з деталізацією опцій бронювання місця пошуку вільних місць паркінг комплексів

Результати експериментального дослідження

Метод / модель	Коренева середньоквадратична похибка
Модель SARIMAX	0,434
Метод опорних векторів	0,381
Метод XGBoost	0,268
Розроблений метод	0,205

15

Рисунок В.15 – Порівняльний аналіз результатів експериментального дослідження

ВИСНОВКИ

В результаті виконання магістерської дипломної роботи розроблено програмне забезпечення для бронювання місць паркування автотранспорту. У роботі представлено метод прогнозування доступності місць для паркування автомобілів. Використовуючи цей метод, було створено програмне забезпечення, яке надає реальні та прогнозовані дані, дозволяючи автоводіям заздалегідь бронювати місця для паркування своїх автомобілів. Проведено експериментальне дослідження з метою знаходження рішення проблеми прогнозування доступності простору для паркування автотранспорту. Для досягнення поставленої мети в даній магістерській роботі було виконано:

- проведений аналіз наявних аналогів для формування процесу прогнозування вільних місць для паркування автомобілів, виділення унікальних функцій цієї програми;
- прийнято рішення щодо використання Python і Django у процесі розробки, враховуючи використання бібліотек для обробки даних та інтелектуального аналізу, що підвищило ефективність розробки;
- сформована та запропонована структура бази даних, враховуючи функціональність збереження даних про парковочні місця та інформацію, отриману в результаті прогнозування доступності місць для паркування;
- описані заходи по розробці алгоритму роботи та реалізації запропонованого програмного забезпечення;

Обґрунтованість представленого метода та достовірність підвищення його ефективності, порівняно з існуючими рішеннями, були підтверджені експериментальними дослідженнями, які використовували вибірку реальних даних.

Дипломна магістерська робота успішно виконана на основі раніше сформульованого переліку завдань.

16

Рисунок В.16 – Висновки