МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМ. В. ДАЛЯ
ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ТА ЕЛЕКТРОНІКИ
КАФЕДРА КОМП'ЮТЕРНИХ НАУК ТА ІНЖЕНЕРІЇ

До захисту допускається
Завідувач кафедри
_____ Скарга-Бандурова І.С.

« ___» _____ 20__р.

ДИПЛОМНИЙ ПРОЕКТ (РОБОТА) БАКАЛАВРА

ПОЯСНЮВАЛЬНА ЗАПИСКА

НА ТЕМУ:

Інформаційна система приватних центрів паркування

Освітньо-кваліфікаційний рівень "бакалавр"
Напрям підготовки 6.050102 "Комп'ютерна інженерія"
(шифр і назва спеціальності)

Керівник проекту: _____ Деркач М.В.
(підпис) (ініціали, прізвище)

Консультант з охорони праці: _____ Критська Я.О.
(підпис) (ініціали, прізвище)

Здобувач вищої освіти: _____ Лисак В.В.
(підпис) (ініціали, прізвище)

Група: КІ-15аД

Сєвєродонецьк 2019

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
СХІДНОУКРАЇНСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ВОЛОДИМИРА ДАЛЯ

Факультет Інформаційних технологій та електроніки
Кафедра Комп'ютерних наук та інженерії
Освітньо-кваліфікаційний рівень бакалавр
Напрям підготовки 6.050102 "Комп'ютерна інженерія"
(шифр і назва)
Спеціальність _____
(шифр і назва)

ЗАТВЕРДЖУЮ:
Завідувач кафедри _____
_____ І.С. Скарга-Бандурова
« ___ »_____20 __ р.

З А В Д А Н Н Я
НА ДИПЛОМНИЙ ПРОЕКТ (РОБОТУ) БАКАЛАВРА

Лисака Владислава Вікторовича
(прізвище, ім'я, по батькові)

1. Тема роботи Інформаційна система приватних центрів паркування
_____
_____
керівник проекту (роботи) ст.викл. Деркач М.В.
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)
затверджені наказом вищого навчального закладу від 13.05.2019 р. № 83/15.15
2. Термін подання студентом роботи 11.06.2019
3. Вихідні дані до роботи матеріали переддипломної практики
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити) ____аналіз стану надання транспортних послуг та засоби для розробки інформаційної системи, розробка інформаційної системи, розробка бази даних для інформаційної системи, охорона праці
_____
_____

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень) електронні плакати_____
_____
_____
_____
_____
_____

6. Консультанти розділів проекту (роботи)

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Охорона праці	ст.викл. Критська Я.О.		

7. Дата видачі завдання 30.04.2019_

Керівник _____
(підпис)

Завдання прийняв до виконання _____
(підпис)

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів дипломного проекту(роботи)	Строк виконання етапів проекту (роботи)	Примітка
1	Аналіз літературних джерел з теми ДП	30.04.19 – 05.05.19	
2	Постановка завдання	06.05.19 – 07.05.19	
3	Розробка БД	08.05.19 – 14.05.19	
4	Розробка інтерфейсу інформаційної системи	15.05.19 – 20.05.19	
5	Тестування розробки	21.05.19 – 24.05.19	
6	Розробка розділу охорони праці	25.05.19 – 30.05.19	
7	Оформлення пояснювальної записки ДП	31.05.19 – 11.06.19	

Здобувач вищої освіти _____ Лисак В.В.
(підпис) (ініціали, прізвище)

Керівник_____ Деркач М.В.
(підпис) (ініціали, прізвище)

РЕФЕРАТ

Пояснювальна записка містить: 73 сторінки, 13 рисунків, 2 таблиці, 2 додатки.

Мета роботи: розробка інформаційної системи центрів паркування, що буде використовуватися для моніторингу та інформування про стан доступності кожного окремого приватного центру.

В ході роботи було виконано аналіз інформаційних технологій у сфері надання транспортних послуг. В роботі розроблена інформаційна система, яка реалізує такі функції: бронювання місця, керування місцем - можливість вказувати, що місце вільне в будь-який проміжок часу, збереження історії паркування, звіт паркування. Інформаційна система розроблена з використанням наступних програмних продуктів - GOLANG, протоколів - NGINX, DOCKER, ORY HYDRA, для графічної частини - REACT JS, для розробки бази даних - GRAPHQL, POSTGRESQL.

Розроблено розділ охорони праці та безпеки в надзвичайних ситуаціях.

Ключові слова: інформаційна система, моніторинг, центр паркування, паркомісце, автовласник.

ЗМІСТ

ВСТУП

Інформаційні послуги транспортної галузі охопили майже весь світ. Їх реалізація дуже корисна в рішенні поточних проблем у великих міських районах, так як вони забезпечують можливість автоматичного планування маршруту, моніторингу безпеки руху, інформації про паркування, збір даних про транспорт, викиди і споживання палива під час подорожі. Різноманітні сервіси використовуються для багатьох транспортних завдань, включаючи навігацію, розташування та паркувальне місце, збір даних про дорожній рух, туристичну інформацію та багато іншого.

Попри велику різноманітність інформаційних систем, мобільних додатків і онлайн-сервісів, пов'язаних з транспортом, оптимальні рішення для паркувальних систем у багатьох країнах ще не були реалізовані, зокрема в нашій країни.

Проблеми, пов'язані з пошуком вільного місця та витраченим часом для паркування, також із заторами, можуть бути вирішені, якщо водії будуть мати змогу заздалегідь володіти інформацією про центри паркування та наявність вільних місць на них, і що набагато зручніше отримати відомості про найближчий центр. Вирішення цих проблем можливе, за рахунок розробці інформаційної системи для розумного паркування, що буде використовуватися для моніторингу та інформування про стан доступності кожного окремого центру паркування, тобто це доцільне завдання з економічної точки зору.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ

IoT	–	Internet of Things
ITS	–	Intelligent transport system
GPS	–	Global position system
GSM	–	Global System for Mobile Communications
CDMA	–	Code Division Multiple Access
API	–	Application programming interface
ATS	–	Average travel speed
RFID	–	Radio Frequency Identification
RPM	–	Red Hat Package manager
ISO/TR	–	International Organization for Standardization / Technical Reports
VPN	–	Virtual Private Network
HTTP	–	Hypertext Transfer Protocol
cURL	–	Client Uniform Resource Locator
ID	–	Identification
SMS	–	Short Message Service

ПЕРЕЛІК ЗМІННИХ

υ – усереднена швидкість руху транспорту

υ_i – поточна швидкість руху транспорту в момент в момент часу i

i – поточний часовий інтервал

d – відстань між двома точками

R – радіус Землі (6378,1 км)

$\Delta\sigma$ – кутова різниця

φ – широта об'єкту

λ – довгота об'єкту

$\Delta\lambda$ – різниця координат за довжиною

t – прогнозний час прибуття транспортного засобу

N – розмір вибірки для прогнозування

1. АНАЛІЗ СТАНУ НАДАННЯ ТРАНСПОРТНИХ ПОСЛУГ ТА ЗАСОБИ ДЛЯ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ

1.1 Транспортна система: проблеми та їх вирішення

Зростання населення та посилення урбанізації породжують різні технічні, соціальні, економічні та організаційні проблеми, які, як правило, ставлять під загрозу економічну і екологічну стійкість міст. Тільки протягом 20-го століття населення у світі зросло з 1,65 млрд. до 6 млрд. осіб. Зміни світового населення протягом 1990-2019 рр. наведено на рисунку1.1 [1].

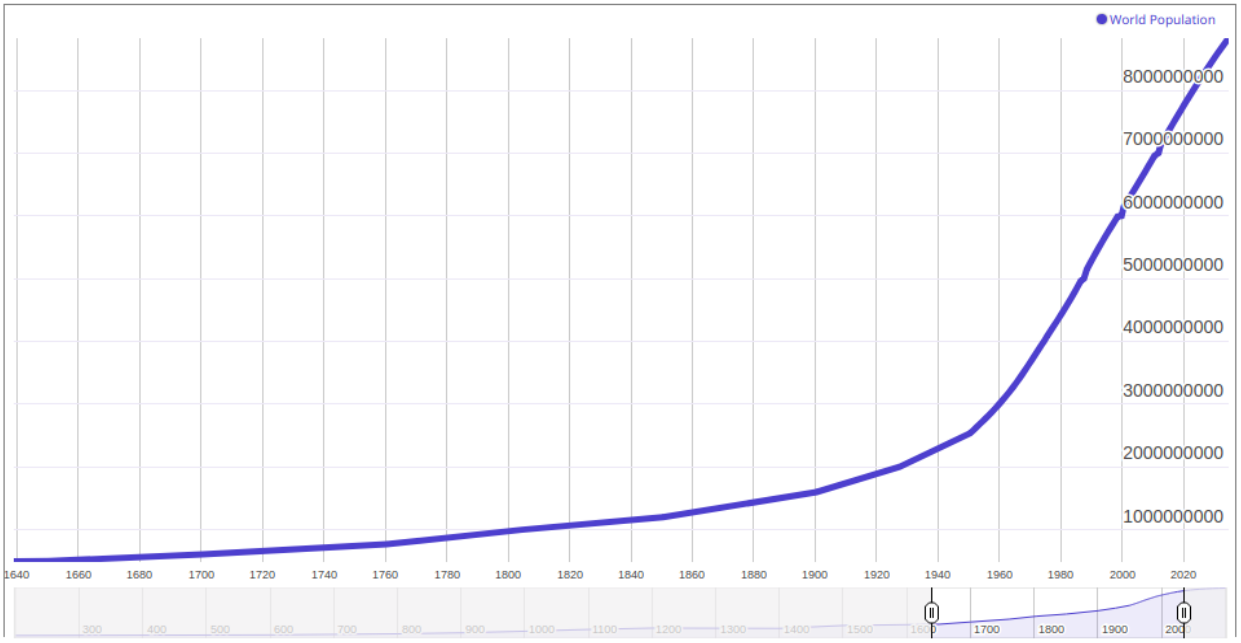

Рисунок 1.1 - Світове населення: минуле, сьогодення і майбутнє

Стосовно транспортної сфери, всесвітня статистика репрезентує число проданих автомобілів з 1990 до 2018 рр., та додатково прогноз для 2019 року (рис. 1.2), згідно якого очікується продати майже 79 мільйонів автомобілів [2].

Огляд зростання чисельності населення, та кількості автомобілів у світі доводить, що така ситуація призводить не лише к посиленню урбанізації, а

також к неминучим проблемам, зокрема пробкам на дорогах, зниженню якості надання транспортних послуг.

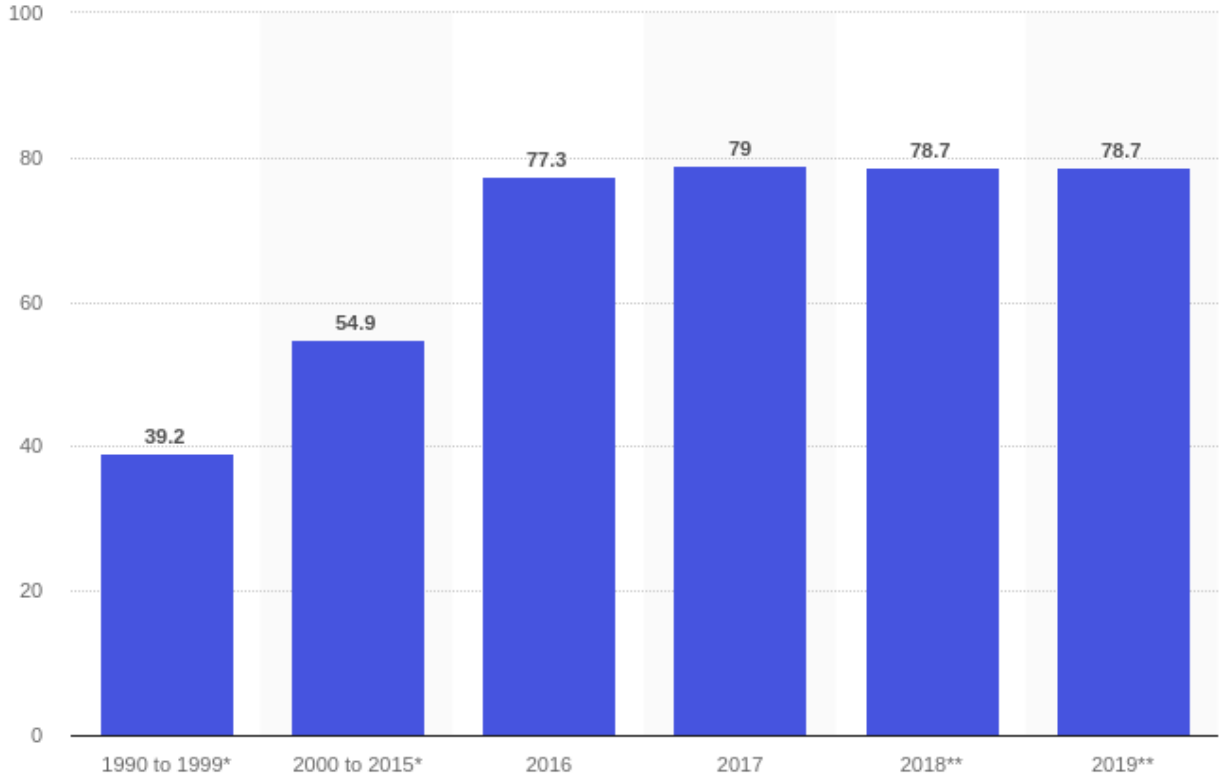

Рисунок 1.2 - Кількість проданих автомобілів по всьому світу з 1990 по 2019 рр. (в млн. одиниць)

Зокрема у великих містах існує проблема, пов’язана з неефективним використанням паркувальних місць та нерівномірним розподілом автомобільного трафіку, що у свою чергу, призводить до заторів транспортного руху.

Вирішення цієї проблеми - можливість вжити заходи для підвищення ефективності ресурсів центрів паркування, що призведе до скорочення часу пошуку, заторів і дорожньо-транспортних пригод, а саме розробка інформаційної системи для інтелектуальної парковки, яка становиться стратегічним питанням не тільки для досліджень, а й з економічної точки зору.

Проблеми, пов'язані з паркуванням та затори можуть бути вирішені, якщо водії будуть мати змогу заздалегідь володіти інформацією про центри паркування та наявність вільних місць на них.

1.2 Транспортна система: аналіз існуючих сервісів

Інформаційні послуги транспортної галузі охопили майже весь світ. Їх реалізація дуже корисна в розв'язання поточних проблем у великих міських районах, тому, що вони забезпечують можливість автоматичного планування маршруту, моніторинг безпеки руху, інформація про паркування, збір даних про транспорт.

Існує безліч додатків цілеспрямованих для безпеки руху, кількісної емісії транспортування [3], відображення інформації про подорожі в режимі реального часу для мандрівки [4]. Відомий проект Національної Адміністрації дорожньої безпеки дорожнього руху [5], було створено мобільний додаток під назвою "безпечний автомобіль". Ця програма спрямована на полегшення доступу та надання інформації про важливі проблеми безпеки автомобіля. Департамент транспорту Теннессі (TDOT) розробив свій мобільний додаток "TDOT SmartWay мобільний додаток", що здійснює моніторинг робочих швидкостей та інцидентів [6].

Стосовно послуг для автовласників, відомі онлайн-сервіси для таких країн, як Великобританія, Німеччина, Швеція, Сінгапур, Індія. Широко використовуються паркувальні системи з різними технологіями, а саме з використанням датчиків для підрахунку кількості [7], або для встановлення точного розташування вільних місць [8], наприклад JustPark, Parkalot, Smart Parking Manager та ін. Але не всі дають можливість паркування приватних місць, саме немає якісного рішення для паркування співробітників великих компаній.

Застосування технологій на основі Інтернету речей (IoT), дає змогу створити сервіси транспортної галузі для ефективного використання наявних

паркувальних систем, що дозволить виконати обробку даних у режимі реального часу та здійснювати моніторинг центрів паркування, тим самим підвищити продуктивність та надійність міської інфраструктури.

У роботі [9], автори пояснюють термін IoT у формі наступного рівняння:

Фізичний об'єкт + Контролер, датчик і виконавчі пристрої + Інтернет = Інтернет речей (IoT)

Тобто, основна функціональність IoT полягає в тому, щоб забезпечити зв'язок між центром паркування та автовласниками, за допомогою Інтернету та інформаційних сервісів.

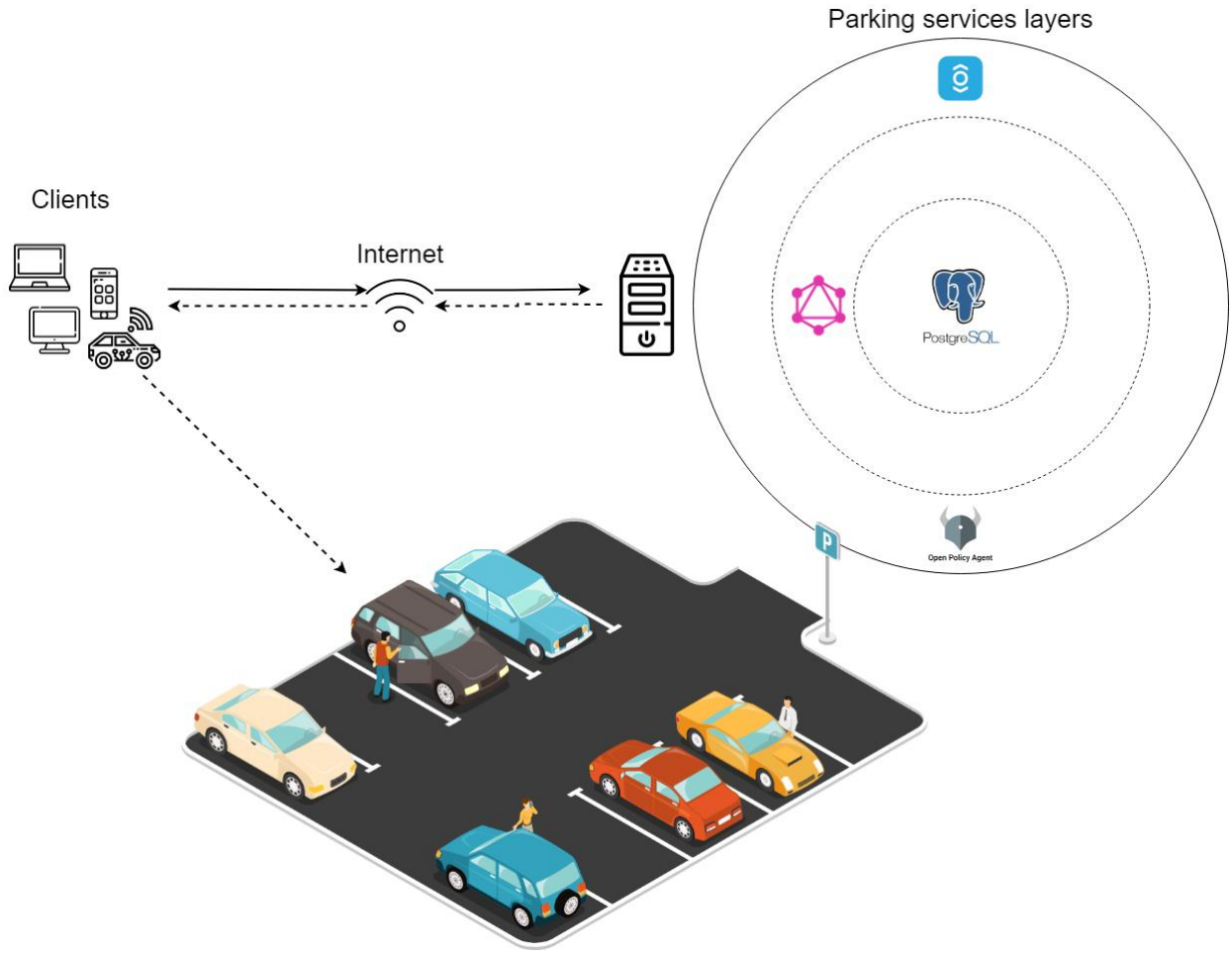

Рисунок 1.3 - Загальний принцип інформаційної системи

На рис. 1.3 зображено загальний принцип інформаційної системи, який полягає в тому, що сервіс володіє інформацією про кілька майданчиків зі своїми паркомісцями, реалізує збереження та передачу шляхом Інтернет-запиту даних про найближчий центр паркування, а саме час, коли місця вільні, на гаджети клієнтів автомобілів.

Попри розмаїття мобільних додатків і сервісів, пов'язаних з транспортом, оптимального рішення для системи паркування у багатьох країнах ще не знайдено. У сучасних містах знайти та зекономити час на пошук вільного місця для паркування все ще є важкою задачею для водіїв, це стає дедалі складнішим завданням. Проблеми, пов'язані з паркуванням та затори можуть бути вирішені, якщо водії будуть мати змогу заздалегідь володіти інформацією про центри паркування та наявність вільних місць на них, що набагато зручніше отримати відомості про найближчий центр [10].

1.3 Засоби для розробки інформаційної системи

1.3.1 Nginx

NGINX - програмне забезпечення, написане для UNIX-систем. Основне призначення - самостійний HTTP-сервер, або, як його використовують частіше, фронтенд для високо навантажених проектів.

Одна з фундаментальних основ будь-якого Unix-додатку - це процес або потік, тобто самодостатній набір інструкцій, який операційна система може запланувати для виконання на ядрі процесора. Більшість великих програм паралельно запускають безліч процесів або потоків з двох причин:

– Щоб одночасно задіяти більше обчислювальних ядер.

– Процеси і потоки дозволяють простіше виконувати паралельні операції (наприклад обробляти безліч з'єднань одночасно).

Процеси та потоки самі по собі витрачають додаткові ресурси. Кожен такий процес або потік витрачається певний обсяг пам'яті, а крім того вони постійно підміняють один одного на процесорі. Сучасні сервери можуть

справлятися з сотнями активних процесів і потоків, але продуктивність сильно страждає, як тільки закінчується пам'ять або величезна кількість операцій введення-виведення призводить до занадто частій зміні контексту.

Найбільш типовий підхід до побудови мережеві додатків - це виділяти для кожного з'єднання окремий процес або потік. Така архітектура проста для розуміння і легка в реалізації, але при цьому погано масштабується коли додатку доводиться працювати з тисячами з'єднань одночасно.

1.3.2 Golang

Go (golang) - компільована, багато поточна, статично типізована мова програмування. Мова була розроблена програмістами в Google. Винайдено для людей хто пише (а також читає, налагоджує і контролює) великі системи ПЗ. Go прагне поліпшити робоче оточення розробника і його колег по роботі для виконання рутинних завдань.

Основні переваги мови Go:

– Простий і зрозумілий синтаксис.

– Статична типізація. Дозволяє уникнути помилок, допущених через неуважність, спрощує читання і розуміння коду, робить код однозначним.

– Швидкість і компіляція. Швидкість у Go в десятки разів більше, ніж у скриптових мов, при меншому споживанні пам'яті. При цьому, компіляція практично миттєва. Весь проект компілюється в один бінарний файл, без залежностей.

– Відхід від ООП. У мові немає класів, але є структури даних з методами. Спадкування замінюється механізмом вбудовування. Існують інтерфейси, які не потрібно явно імплементувати, а лише досить реалізувати методи інтерфейсу.

– Паралелізм. Паралельні обчислення в мові робляться просто, витончено і без головного болю. Goroutine (паралельні потоки) легковагі, споживають мало пам'яті.

– Багато стандартна бібліотека. У мові є все необхідне для веб-розробки і не тільки. Кількість сторонніх бібліотек постійно зростає. Крім того, є можливість використовувати бібліотеки C і C ++.

– Можливість писати в функціональному стилі. У мові є замикання (closures) і анонімні функції. Функції є об'єктами першого порядку, їх можна передавати в якості аргументів і використовувати в якості типів даних.

– Мова має сильне співтовариство і постійно розвивається.

– Open Source.

Додаток на Golang повинно знаходитися в GOPATH вказуються при установці компілятора на комп'ютер. Структура написання програми на Go така: додаток ділитися на пакети (package), в кожному файлі Go на початку документа оголошується ім'я пакету, ім'я має відповідати назві директорії в якому розташований файл з розширенням * .go. В пакетах описуються функції, структури та типи, після чого викликаються в головному пакеті main в функції на виконання main(). Для запуску програми головний пакет компілюється в бінарний файл командою:

go build -o path/to/output/file path/to/main.go

1.3.3 Docker

Docker - програмне забезпечення для автоматизації розгортання та управління додатками в середовищах з підтримкою контейнеризації. Дозволяє «упакувати» додаток з усім його оточенням і залежностями в контейнер, який може бути перенесений на будь-яку Linux-систему, а також надає середовище з управління контейнерами.

1.3.4 PostgreSQL

PostgreSQL - вільна об'єктно-реляційна система управління базами даних. Оскільки це об'єктно-реляційна БД - це дає деякі переваги над іншими SQL базами даних з відкритим вихідним кодом, такими як MySQL, MariaDB і Firebird.

Фундаментальна характеристика об'єктно-реляційної бази даних - це підтримка об'єктів і їх поведінки, включаючи типи даних, функції, операції, домени й індекси. Це робить Postgres неймовірно гнучкою і надійною.

Серед іншого, вона вміє створювати, зберігати та видавати складні структури даних.

1.3.5 GraphQL

GraphQL - це новий і захопливий API для спеціальних запитів і маніпуляцій. Він надзвичайно гнучкий і забезпечує безліч переваг. Він особливо підходить для зображення даних, організованих у вигляді графів і дерев. Facebook розробив GraphQL у 2012 році та відкрив його у 2015 році. GraphQL видає інформацію у вигляді JSON. Найбільшою перевагою сервісу це гнучкість запитів, а саме це великий асортимент фільтрів при запиті та вибір необхідних параметрів для отримання відповіді.

GraphQL - це мова запитів для API і середовище виконання сервера для виконання запитів за допомогою системи типів, яка визначається для необхідних даних. GraphQL не прив'язаний до будь-якої конкретної бази даних або механізму зберігання даних, а натомість підтримується чинним кодом і даними.

Послуга GraphQL створюється шляхом визначення типів і полів на цих типах, а потім надає функції для кожного поля кожного типу. Наприклад, служба GraphQL, яка повідомляє, хто є зареєстрований користувач, а також це ім'я користувача може виглядати приблизно так:

```
type Query {
  me: User
}
type User {
  id: ID
  name: String
}
```

Поряд з функціями для кожного поля кожного типу:

```
function Query_me(request) {
  return request.auth.user;
}
function User_name(user) {
  return user.getName();
}
```

Як тільки служба GraphQL запущена (як правило, на URL-адресу веб-служби), її можна надіслати запити GraphQL для перевірки та виконання. Отриманий запит спочатку перевіряється, щоб переконатися, що він стосується тільки визначених типів і полів, а потім запускає надані функції для отримання результату.

1.3.6 React JS

React - це декларативна, ефективна і гнучка JavaScript бібліотека для побудови призначених для користувача інтерфейсів. React розробляється і підтримується Facebook, Instagram і спільнота окремих розробників і корпорацій. React може використовуватися для розробки односторінкових і

мобільних додатків. Його мета - надати високу швидкість, простоту і масштабованість.

1.3.7 Библиотека JSON

JSON - JavaScript Object Notation текстовий формат обміну даними, заснований на JavaScript. Як і багато інших текстові формати, JSON легко читається людьми. Формат JSON був розроблений Дугласом Кроуфордом.

JSON являє собою набір пар ключ: значення. Як значення в JSON може використовуватися об'єкт, масив, число, літерали true / false, рядок. Приклад:

```
{
  "f_name": "Иван",
  "l_name": "Иванов",
  "address": {
    "street": "Московское ш., 101, кв.101",
    "city": "Ленинград",
    "postalCode": "101101"
  },
  "phones": [
    "812 123-1234",
    "916 123-4567"
  ]
}
```

1.3.8 HTTP protocol

HTTP - протокол прикладного рівня передачі даних у вигляді гіпертекстових документів у форматі «HTML», а також використовується для передачі довільних даних.

Основою HTTP є технологія «клієнт-сервер», тобто передбачається існування:

– Споживачів (клієнтів), які ініціюють з'єднання і надсилають запит.

– Постачальників (серверів), які очікують з'єднання для отримання запиту, виробляють необхідні дії та повертають назад повідомлення з результатом.

1.3.9 OAuth 2.0 protocol

OAuth - відкритий протокол (схема) авторизації, який дозволяє надати третій стороні обмежений доступ до захищених ресурсів користувача без необхідності передавати їй (третій стороні) логін і пароль.

Структура авторизації OAuth 2.0 дозволяє застосуванню сторонніх виробників отримувати обмежений доступ до служби HTTP, або від імені власника ресурсу, організовуючи взаємодію між власником ресурсу та службою HTTP, або дозволяючи програмі третьої сторони отримати доступ від свого імені.

У традиційній моделі аутентифікації клієнт-сервер клієнт запитує ресурс з обмеженим доступом (захищений ресурс) на сервері шляхом аутентифікації на сервері, використовуючи облікові дані власника ресурсу. Щоб надати стороннім додаткам доступ до обмежених ресурсів, власник ресурсу ділиться своїми повноваженнями з третьою стороною. Це створює кілька проблем і обмежень.

OAuth розв'язує ці проблеми, вводячи рівень авторизації та розділяючи роль клієнта від власника ресурсу. У OAuth клієнт запитує доступ до ресурсів, які контролюються власником ресурсів і розміщуються сервером ресурсів, і видає інший набір облікових даних, ніж ті, що належать власнику ресурсу.

Замість використання облікових даних власника ресурсу для доступу до захищених ресурсів, клієнт отримує маркер доступу - рядок, що позначає певну область, час життя та інші атрибути доступу. Токени доступу видаються клієнтам третіх сторін сервером авторизації з дозволу власника ресурсу. Клієнт використовує маркер доступу для доступу до захищених ресурсів, розміщених на сервері ресурсів.

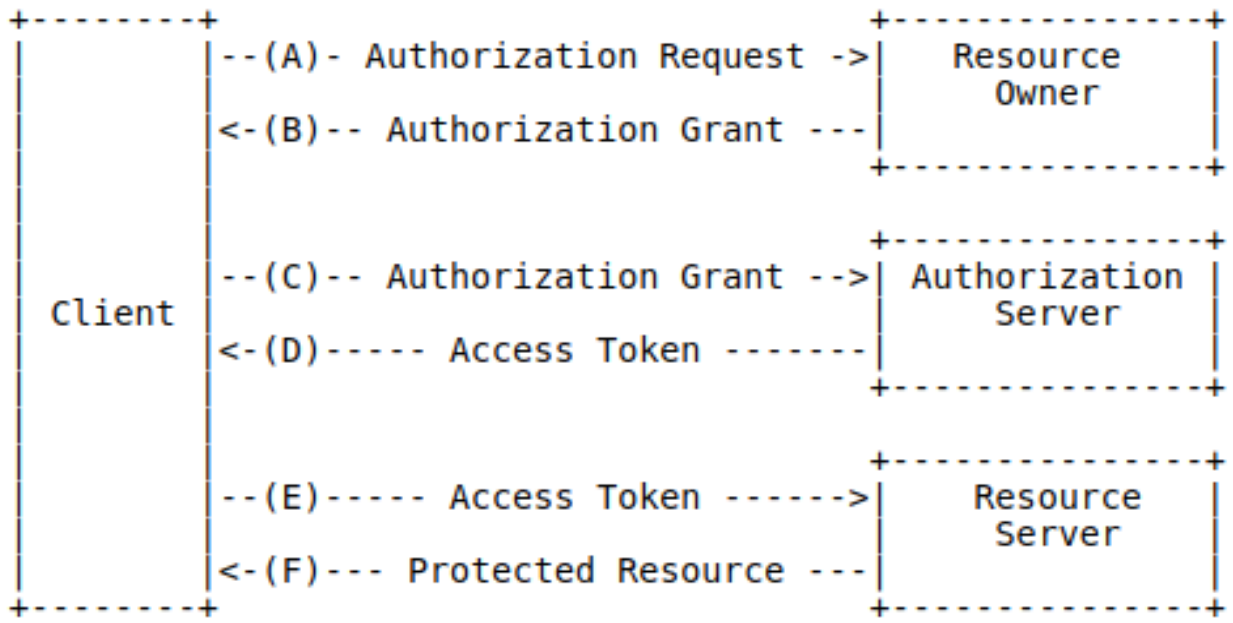

Рисунок 1.4 – Потік протоколу абстрактних зображень

Анотація потоку OAuth 2.0, проілюстрована на рис. 1.4, описує взаємодія між чотирма ролями і включає наступні етапи:

(A) Клієнт запитує авторизацію від власника ресурсу. Запит на авторизацію може бути зроблений безпосередньо власнику ресурсу (як показано на рисунку) або опосередковано через сервер авторизації як посередник.

(B) Клієнт отримує дозвіл на авторизацію, який є повноваженням, що представляє авторизацію власника ресурсу, виражене з використанням одного з чотирьох типів грантів, визначених у цій специфікації, або з використанням типу гранту розширення. Тип надання авторизації залежить

від методу, який використовується клієнтом для запиту авторизації, і типів, що підтримуються сервером авторизації.

(C) Клієнт запитує маркер доступу шляхом аутентифікації з сервером авторизації та подання дозволу на авторизацію.

(D) Сервер авторизації ідентифікує клієнта і перевіряє дозвіл авторизації, і, якщо він дійсно, видає маркер доступу.

(E) Клієнт запитує захищений ресурс від сервера ресурсів і ідентифікує, надаючи маркер доступу.

(F) Сервер ресурсів перевіряє маркер доступу, і, якщо він є, обслуговує запит.

Кращий спосіб для клієнта для отримання дозволу на авторизацію від власника ресурсу (зображений на етапах (A) і (B)) полягає у використанні сервера авторизації як посередника.

1.3.10 OpenID Connect 1.0

OpenID - відкритий стандарт децентралізованої системи аутентифікації, що надає користувачеві можливість створити єдину обліковий запис для аутентифікації на безлічі не пов'язаних один з одним інтернет-ресурсів, використовуючи послуги третіх осіб.

OpenID Connect 1.0 - це простий ідентифікаційний шар поверх протоколу OAuth 2.0. Вона дозволяє Клієнтам перевіряти ідентичність кінцевого користувача на основі аутентифікації, виконуваної авторизаційним сервером, а також отримувати основну інформацію про профіль кінцевого користувача в сумісному і подібному до REST способі.

У якості фону, OAuth 2.0 Authorization Framework і OAuth 2.0 Bearer Token Usage використовують загальні рамки для сторонніх додатків для отримання та використання обмеженого доступу до ресурсів HTTP. Вони визначають механізми отримання та використання маркерів доступу для доступу до ресурсів, але не визначають стандартні методи надання

інформації про ідентифікацію. Зокрема, без профілювання OAuth 2.0, він не може надавати інформацію про аутентифікацію кінцевого користувача.

OpenID Connect реалізує аутентифікацію як розширення процесу авторизації OAuth 2.0.

OpenID Connect дозволяє клієнтам усіх типів, у тому числі веб-клієнтам, мобільним та JavaScript, запитувати та отримувати інформацію про аутентифіковані сеанси та кінцевих користувачів. Пакет специфікації розширюється, що дозволяє учасникам використовувати додаткові функції, такі як шифрування ідентифікаційних даних, відкриття постачальників OpenID і керування сеансами, коли це має сенс для них.

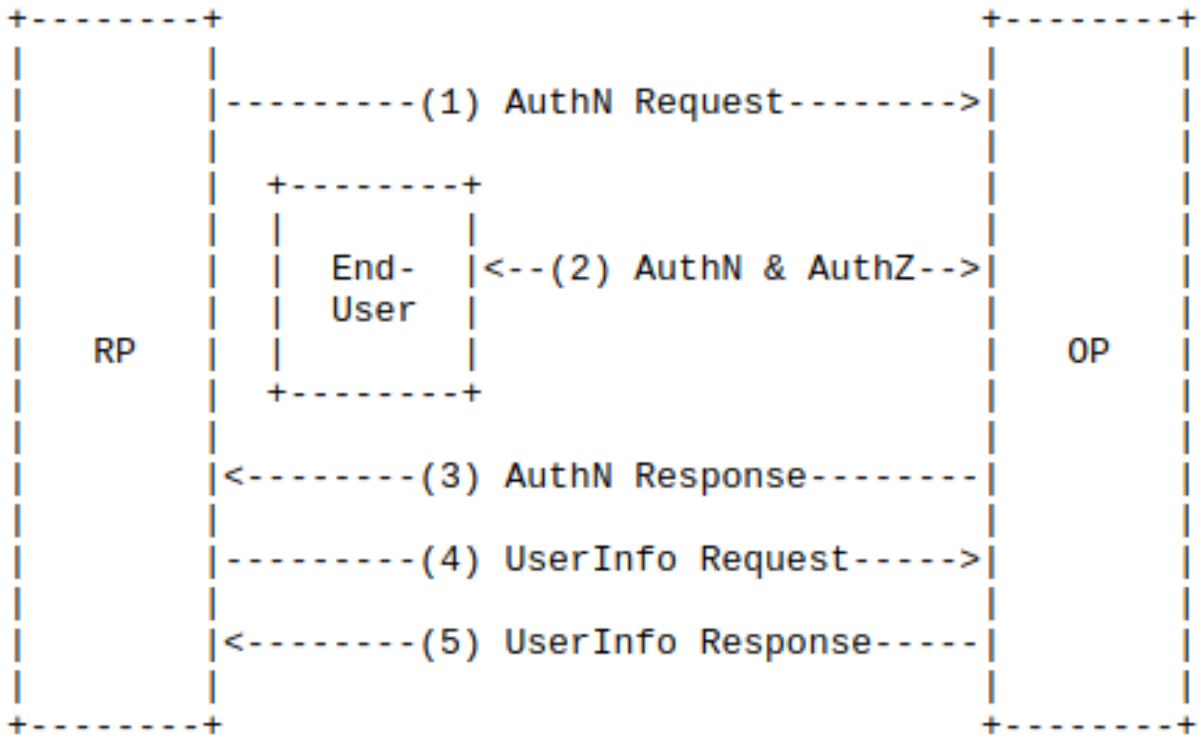

Рисунок 1.5 - Абстрактні кроки протоколу OpenID Connect

Протокол OpenID Connect, у абстрактному вигляді, слідує наступним крокам (рис. 1.5).

1. RP (Клієнт) надсилає запит на постачальника OpenID (OP).

2. OP перевіряє аутентифікацію кінцевого користувача та отримує авторизацію.

3. OP реагує на токен ID і зазвичай на маркер доступу.

4. RP може надіслати запит з маркером доступу кінцевій точці UserInfo.

5. Кінцева точка UserInfo повертає претензії щодо кінцевого користувача.

Relying Party (RP) - Клієнтська програма OAuth 2.0, яка вимагає ідентифікації кінцевого користувача та претензій від постачальника OpenID.

OpenID Provider (OP) - Сервер авторизації OAuth 2.0, здатний аутентифікувати кінцевого користувача та надавати претензії до Сторони, що довіряє, про подію аутентифікації та кінцевого користувача.

1.3.11 ORY Hydra проект з відкритим вихідним кодом для інтеграції OAuth 2.0, OpenID Connect

ORY Hydra - постачальник OAuth 2.0 і OpenID Connect. Таким чином, він здатний видавати маркери доступу, оновлення та ідентифікатори ID. На відміну від інших проектів, ORY Hydra не пропонує управління користувачами (логін, вихід з системи, управління профілями, реєстрація), а замість цього використовує потік на основі перенаправлення і REST API для делегування ідентифікації користувача (логіна) службі, яку ви реалізуєте, і контроль. Це дозволяє вам створювати керування користувачами, яке працює для вас, з технологією інтерфейсу, яка вам подобається, і механізмами аутентифікації, необхідними для вашого випадку використання (наприклад, 2FA на основі маркерів, SMS 2FA).

Таким чином, ORY Hydra є найбільш гнучким постачальником OAuth 2.0 і OpenID Connect і дає велику свободу в реалізації бізнес-логіки, і все ще отримує всі переваги від OAuth 2.0 і OpenID Connect.

Додатково до функціональності OAuth 2.0, ORY Hydra пропонує безпечне сховище для криптографічних ключів (наприклад, для підписування JSON Web-маркерів) і здатне керувати клієнтами OAuth 2.0.

ORY Hydra OpenID Connect і виконує всі вимоги, викладені Фондом OpenID. Таким чином, він правильно реалізує різні потоки OAuth 2.0 і OpenID Connect, як це передбачено IETF і OpenID Foundation.

Hydra є серверною реалізацією рамки авторизації OAuth 2.0 і OpenID Connect Core 1.0. Чинні реалізації OAuth2 зазвичай поставляються як бібліотеки або SDK, такі як node-oauth2-server або fosite, або як повнофункціональні ідентифікаційні рішення з керуванням користувачами та інтерфейсами користувача, наприклад Keycloak або Okta.

Реалізація та використання OAuth2 без розуміння всієї специфікації є складним і схильним до помилок, навіть коли використовуються SDK. Основна мета Hydra полягає в тому, щоб зробити OAuth 2.0 і OpenID Connect 1.0 менш болісними та простішими у використанні.

Hydra реалізує потоки, описані в OAuth2 і OpenID Connect 1.0, не змушуючи використовувати "Hydra User Management" або певний шаблонний движок або заздалегідь визначений інтерфейс. Замість цього він покладається на перенаправлення HTTP та криптографічні методи для підтвердження згоди користувача, що дозволяє використовувати Hydra з будь-якою кінцевою точкою аутентифікації, будь то authboss, auth0.com або власну автентифікацію PHP.

Hydra поєднує найкращі практики в області технології веб-сервісів:

– Hydra поставляється як єдиний двійковий файл для всіх популярних платформ, включаючи Linux, OSX і Windows, без будь-яких додаткових залежностей. Для подальшої простоти, Hydra доступний як Image Docker.

– Hydra спочатку побудована безпека: архітектура і робочі потоки призначені для нейтралізації різних загальних (OWASP TOP TEN) і незвичайних векторів атаки.

– Hydra має низький розмір процесора та пам'яті, короткий час запуску та CLI з розробниками.

– Hydra масштабується без особливих зусиль вгору і вниз на кожній платформі, яка можна собі уявити, включаючи Heroku, Cloud Foundry, Docker, Google Container Engine та багато іншого.

Hydra також має обмеження:

1. Hydra - це не те, що управляє обліковими записами користувачів. Hydra не пропонує користувачам реєстрацію, скидання пароля, вхід користувача, надсилання підтверджень електронною поштою. Саме за це відповідає провайдер ідентичності. Зв'язок між Hydra та постачальником провайдерів ідентичності називається Login і Flow Consent.

2. Якщо розробляється простий сервіс для 50-100 зареєстрованих користувачів, OAuth2 і Hydra, ймовірно, будуть занадто складними.

3. Hydra не підтримує потік даних облікового запису власника ресурсу OAuth2.

4. Якщо є мета дозволити стороннім розробникам, які вже зараз або в майбутньому отримують доступ до API, Hydra ідеально підходить.

5. Якщо є потреба стати провайдером ідентичності, наприклад, Google, Facebook або Microsoft, OpenID Connect і, таким чином, Hydra - ідеально підходить.

6. Запуск провайдера OAuth2 відмінно працює з браузером, мобільними програмами та додатками, які можна носити, оскільки ви можете у будь-який час уникати зберігання облікових даних доступу на пристрої, телефоні або носінні, а також скасовувати маркери доступу та, таким чином, права доступу.

7. Якщо є багато служб і потрібно обмежити автоматизований доступ для цих служб. Приклад: службі коментарів не дозволяється читати паролі користувачів під час завантаження останніх оновлень профілю користувача.

1.4 Постановка завдання

За основу був обраний Nginx сервер. Оскільки серверна мова для проекту є Golang і система контейнеризації Docker, за допомогою Nginx можна легко розподіляти запити і вказувати визначеному маршруту, направляти трафік на той чи інший сервіс написаний на Golang своєчасно розміщений в певному контейнері Docker.

Для реалізації інтерфейсу клієнта використовується JavaScript бібліотека React.

В якості СУБД узятий PostgreSQL. Для взаємодії з СУБД і реалізації API застосовується GraphQL.

Метою даного дипломного проекту є розробка інформаційної системи, що буде використовуватися для моніторингу та інформування про стан доступності кожного окремого центру паркування.

Об’єкт дипломної роботи - геоінформаційні дані та стан доступності центрів паркування.

Предмет дипломної роботи - програмні засоби для реалізації інформаційної системи.

Виходячи з наявних тенденцій на сьогодні виникають наступні основні вимоги до інформаційної системи що розробляється:

1. Забезпечення максимально якісної, надійної та простої системи паркування.
2. Обробка інформації по кожному центру окремо.

Для досягнення поставленої мети слід вирішити такі завдання:

– Виконати огляд існуючих паркувальних систем.

– Обґрунтувати необхідні вимоги щодо розробки.

– Реалізувати всі функції необхідні для роботи інформаційної системи.

– Протестувати роботу функціоналу розробки.

– Виконати розрахунки по розділу “Охорони праці”.

Загалом, інформаційна система матиме наступний функціонал:

1. Реєстраційний календар.
2. Управління паркомісцем.
3. Історія паркування.
4. Візуалізація гаражу.
5. Пошук вільного місця.

2. РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1 Загальна концепція роботи

Дорожній рух вимагає точної та актуальної інформації про поточну ситуацію та доступні послуги. Підвищення якості дорожніх умов є важливим для всіх інфраструктур. Проте, тільки це поліпшення не може задовольнити постійно зростаючі вимоги до безпечних, зручних, економічно ефективних і зручних дорожніх послуг. Можливості вдосконалення існуючих громадських перевезень шляхом впровадження передових технологій у транспортну систему забезпечуються завдяки концепції інтелектуальних транспортних систем (ITC) і технології Інтернет речей (IoT), що у свою чергу дозволить в режимі реального часу здійснювати управління парком, послуги диспетчеризації та планування, аварійні сповіщення, служби безпеки та служби інформування пасажирів.

Технологія Інтернет речей передбачає міжмережеве підключення фізичних пристроїв з електронікою та мережевими зв'язками, які контролюють ці об'єкти для збору та обміну даними. IoT використовується не тільки для відчуття інформації, але й для взаємодії з фізичним світом. IoT може допомогти в інтеграції комунікації, контролю та обробки інформації в різних транспортних системах. IoT використовується для забезпечення взаємодії між пасажиром і ТЗ за допомогою Інтернету.

Однією з найважливіших частин ITC є інфраструктура міського транспорту, спрямована на забезпечення безпечнішої та ефективнішої транспортної системи. Вона забезпечує інформацію про подорож у реальному часі відповідно до потреб людей за допомогою відповідних інструментів. ITC містять системи управління рухом, засоби інформування учасників дорожнього руху, системи контролю, управління та оптимізації маршрутів, засоби і технології управління при надзвичайних ситуаціях, тощо.

Застосування ITC забезпечує декілька переваг, підвищуючи безпеку мандрівників, покращуючи експлуатаційні характеристики транспортної

мережі, головним чином шляхом зменшення пробок, підвищення особистих зручностей, забезпечення кращих умов навколишнього середовища та збільшення економічного зростання та зростання зайнятості.

IoT має справу з різними фізичними об'єктами, об'єднуючи їхні дані в мережі в тій чи іншій формі. Вона в основному займається RFID, інфрачервоними датчиками, глобальними системами позиціонування (GPS) і лазерними сканерами.

Електронні датчики в транспортних засобах можуть відстежувати і реєструвати прискорення, швидкість обертів двигуна (частоту обертання колінчастого вала двигуна) та інші параметри, і ця інформація, що генерується трафіком IoT і збирається на всіх ділянках дороги, може бути представлена менеджерам парків, мандрівників та інших користувачів.

Сучасна інженерна практика для розробки таких систем включає безліч різних методологій і тому потребує ретельно розроблених підходів.

Головна мета роботи сервісу - це спростити систему паркування співробітників і забезпечити надійність завчасного бронювання. Здійснити швидкий і легкий старт роботи з паркувальною системою. Сервіс, структура якого зображена на рисунку 2.1, пропонує максимально мінімізовані дії для початку паркування співробітників фірми.

1. Зареєструвати компанію - вказати ім'я та контактну пошту.
2. Вказати гараж - найменування, його розташування.
3. У візуальному редакторі, конструкторі - розташувати паркувальні місця і їх номери.
4. Зареєструвати працівників (вказати, заповнити список email, після чого співробітники отримають сповіщення-запрошення з паролем доступу в їх особистий кабінет на сервісі).
5. У кожного місця в гаражі, є власний календар за допомогою якого можна вибирати проміжки часу, на які необхідно припаркувати співробітника.

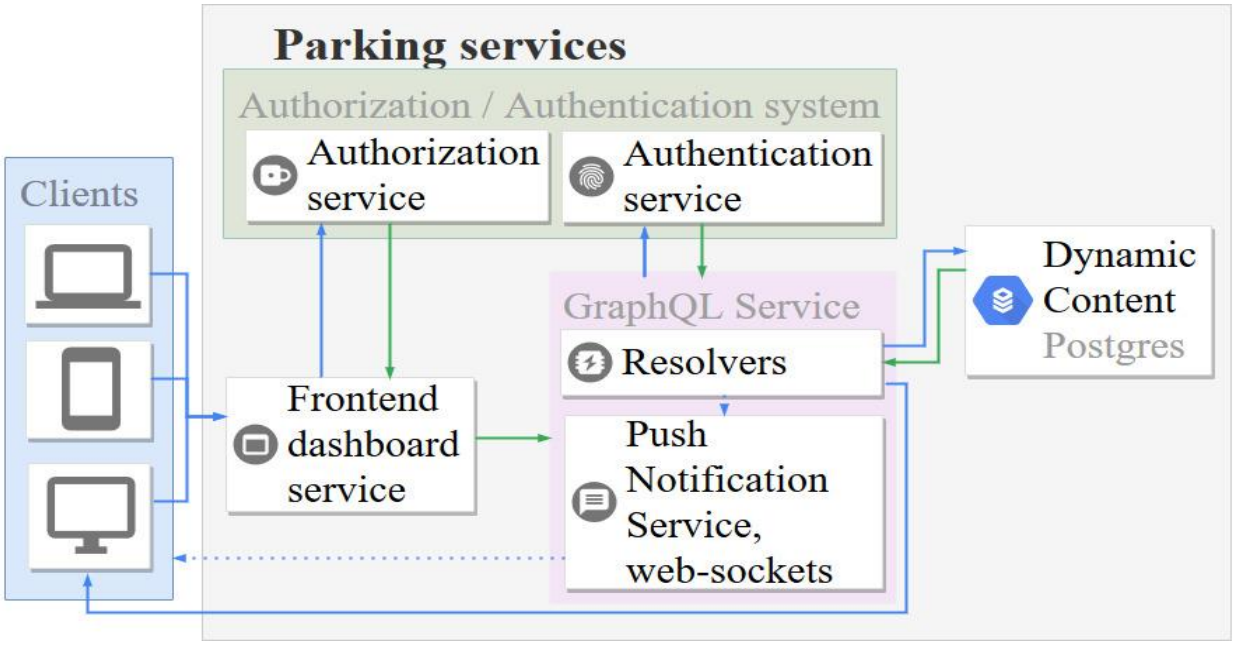

Рисунок 2.1 – Структура інформаційної системи

2.2 Головні функції інформаційної системи

Задля реалізації головної функції розроблено алгоритм пошуку центру паркування [11], що розташований на найкоротшій відстані від поточного місцеперебування автовласника:

1) відомо множина географічних координат центрів паркування $\{K^n\}$:

$$K=\{K^n\},\ K^n=(K^n_{lat},\ K^n_{lng}), \tag{2.1}$$

де K^n_{lat} – широта центру, K^n_{lng} - довгота центру, n – порядковий номер центру серед загальної кількості.

2) відомо розташування автовласника:

$$O=(O_{lat},\ O_{lng}), \tag{2.2}$$

де O_{lat} – широта об'єкту, O_{lng} - об'єкту.

3) дані п.1 та п.2 використовуються для розрахунку відстані між об'єктом та кожним центром паркування за допомогою формул гаверсинусів:

$$d = \Delta\sigma \cdot R, \tag{2.3}$$

де d – відстань між двома точками; R – радіус Землі; $\Delta\sigma$ – кутова різниця.

$$\Delta\sigma = 2\arcsin\left\{\sqrt{\sin^2\left(\frac{\varphi_1 - \varphi_2}{2}\right) + \cos\varphi_1 \cos\varphi_2 \sin^2\left(\frac{\Delta\lambda}{2}\right)}\right\}, \tag{2.4}$$

де φ_1, λ_1; φ_2, λ_2 – широта та довгота двох точок відповідно; $\Delta\lambda$ – різниця координат за довжиною; $\Delta\sigma$ – кутова різниця.

4) в результаті виконання п.3 отримали множину відстаней D:

$$D=\{d_i\}, \tag{2.5}$$

де d_i – відстань між поточним місцеперебуванням об'єкту та кожним центром паркування.

5) серед отриманих відстаней d обирається найменша, що відповідає критеріям пошуку:

$$d=min(D). \tag{2.6}$$

6) визначити мінімальний час прибуття ТЗ до центру паркування.

У традиційній задачі визначення найкоротшого шляху, вартість між вузлом i і вузлом j - це відстань d_{ij}. Беручи до уваги обмеження швидкості, v_{ij}, яке є найвищою швидкістю руху від вузла i до вузла j, час руху може бути визначено, як:

$$T_{ij} = d_{ij} / v_{ij}, \tag{2.7}$$

Замінивши вартість відстані вартістю часу, можна використовувати алгоритм Дейкстри для пошуку мінімального часу шляху наступним чином: мінімізувати

$$t = \sum_{i=S}^{D} \sum_{j=S}^{D} T_{ij} U_{ij} \tag{2.8}$$

за умови

$$\sum_{\substack{j=S \\ j \neq i}}^{D} U_{ij} - \sum_{\substack{j=S \\ j \neq i}}^{D} U_{ji} = \begin{cases} 1, i = S \\ -1, i = D \\ 0 \end{cases} \tag{2.9}$$

$$\sum_{\substack{j=S \\ j \neq i}}^{D} U_{ij} = \begin{cases} \leq 1, i \neq D \\ = 0, i = D \end{cases}$$

для усіх i

$$U_{ij} \in \{1,0\}$$
$$T_{ij} = d_{ij} / v_{ij}$$

де N - безліч всіх вузлів; S - вузол-джерело, $\in N$; D - вузол-призначення, $\in N$; i, j - індекс вузла, i, j, $\in N$; d_{ij} - відстань від вузла i до вузла j; v_{ij} - швидкість на ділянці від вузла i до вузла j; T_{ij} – час руху від вузла i до вузла j; U_{ij} - бінарна змінна, 1, якщо зв'язок від вузла i до вузла j існує, 0 інакше; T - загальний час руху.

Перевірка роботи наведеного алгоритму проводилась шляхом порівняння розрахованих відстаней з результатами отриманими за допомогою засобів Google Maps API.

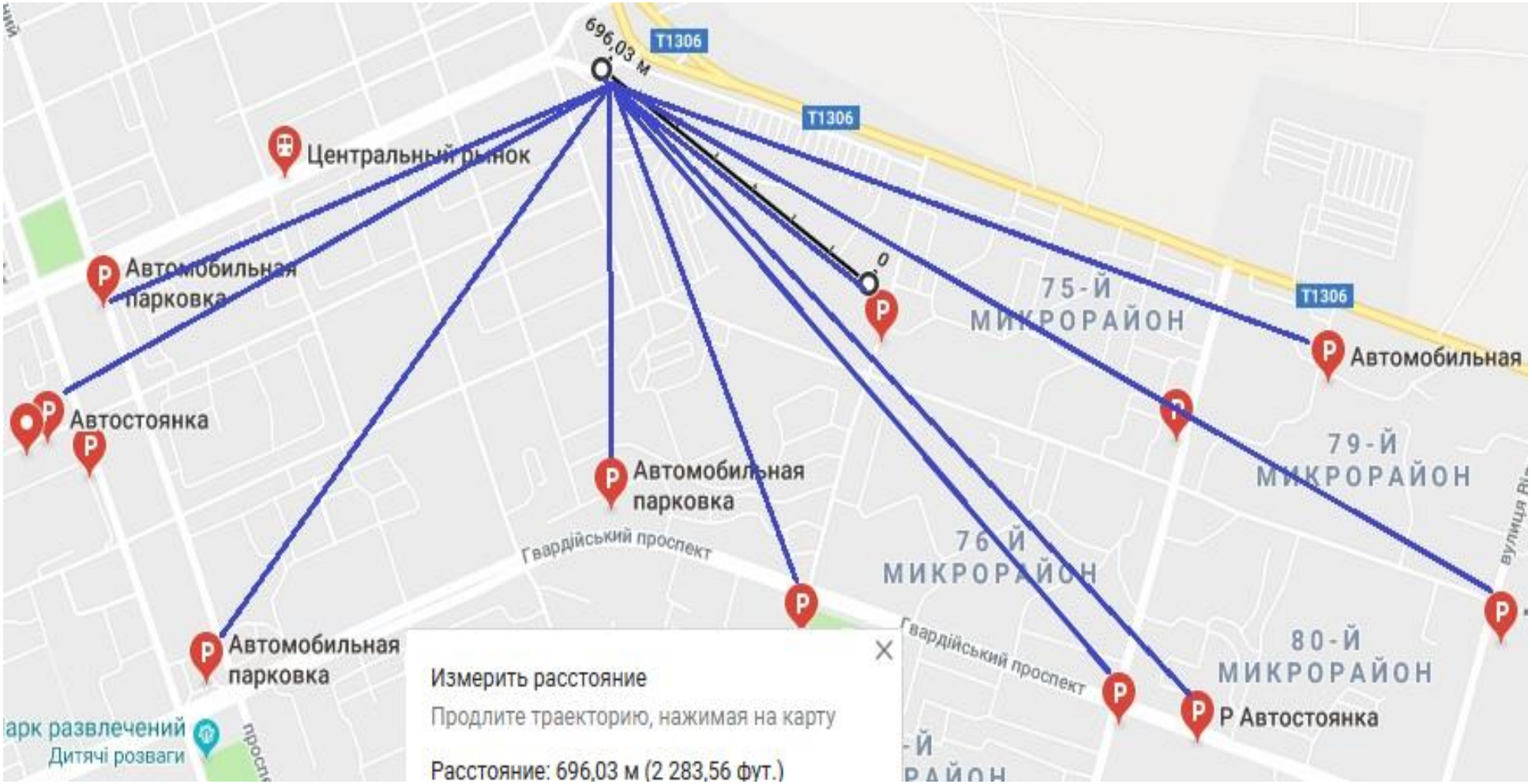

Рисунок 2.2 – Приклад розташування центрів паркування

Попри те що цей сервіс повертає майже точний результат, існує суттєвий недолік, а саме обмеження застосування зводиться до 2500 запитів на добу, а кількість запитів від користувачів мобільного додатку заздалегідь невідома.

Так, для кожної з 10 контрольних точок, розташованих у випадковому порядку, було проведено розрахунок відстаней до всіх можливих паркувальних майданчиків. Приклад розрахунку для точки 1 надано на рис. 2.2.

В результаті, для точки 1 отримано таку множину відстаней: D={1,462; 1,589; 1,514; 1,291; 1,130; 1,328; 0,711; 0,983; 0,696; 1,320}.

Серед яких найкоротша відстань складає 0,696 км, що збігається з результатами, отриманими від Google Maps API. Це дає змогу, зробити висновок, що алгоритм може реалізувати пошук найближчого центру паркування і не буде залежить від кількості запитів.

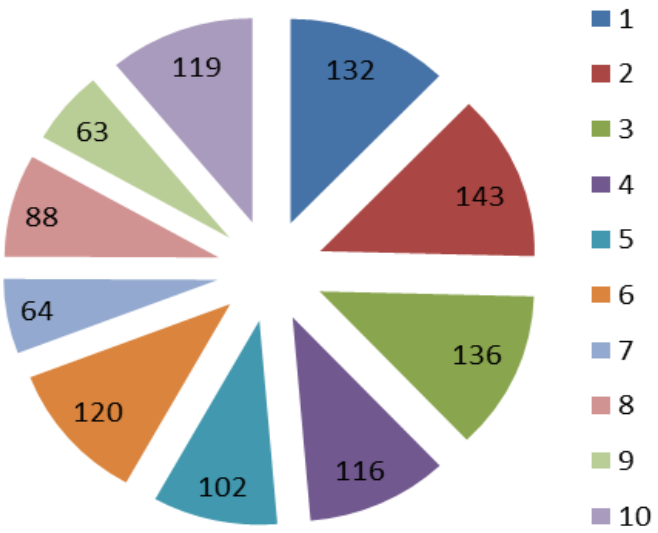

Рисунок 2.3 – Мінімальний час до центрів паркування

Наступний крок – розрахунок часу, що потрібно витратити на подолання шляху до найближчого центру паркування, враховуючи обмеження швидкості руху автомобіля згідно регламентованої швидкості пересування по населеному пункту.

На діаграмі 2.3 зображено час, отриманий в результаті виконання п.6 наведеного алгоритму. Мінімальний час відповідає найближчому центру, тобто для розглянутого прикладу – це центр паркування, що знаходиться на відстані 0,696 м та за 63 секунди від місця перебування автовласника.

Це дає змогу, зробити висновок, що алгоритм може реалізувати пошук найближчого центру паркування і не буде залежить від кількості запитів.

Для реалізації перевірки наявності вільних місць для паркування та надання інформації водієві виконуються наступні дії: треба вибрати центр паркування серед наявних місць у базі даних, а потім обрати необхідний проміжок часу для паркування в режимі “Календар” [12].

Для бронювання в режимі реального часу паркувальних місць необхідна інформація з бази даних сервісу про час і номери наявних вільних місць для кожного паркувального центру. Для цього формується запит по ідентифікатору місця паркування, в результаті виконання якого отримується відповідь у форматі JSON.

Запит виглядає наступним чином:

http://localhost:4000/graphql?=query{placeBooking(placeNum:2205){durat ion{fromto}idisBusyplaceIdplaceNumuser {idfullNameemail}}}

Повертає масив даних про вільний час обраного місця для подальшої інтерпретації у звичайний вигляд для мобільного додатку:

{"data":{"placeBooking":["duration":{"from":"2018-11-16T18:00:00Z",
"to":"2018-11-16T23:00:00Z"},"id":"1542375056291962597","isBusy":false,
"placeId":21, "placeNum":2205, "user":{"email":"","fullName":"","id":0}},
{"duration":{"from":"2018-11-17T03:00:00Z", "to":"2018-11-
17T15:00:00Z"},"id":"1542375056291963558","isBusy":false,
"placeId":21,"placeNum":2205,"user":{"email":"","fullName":"","id":0}}]}
}

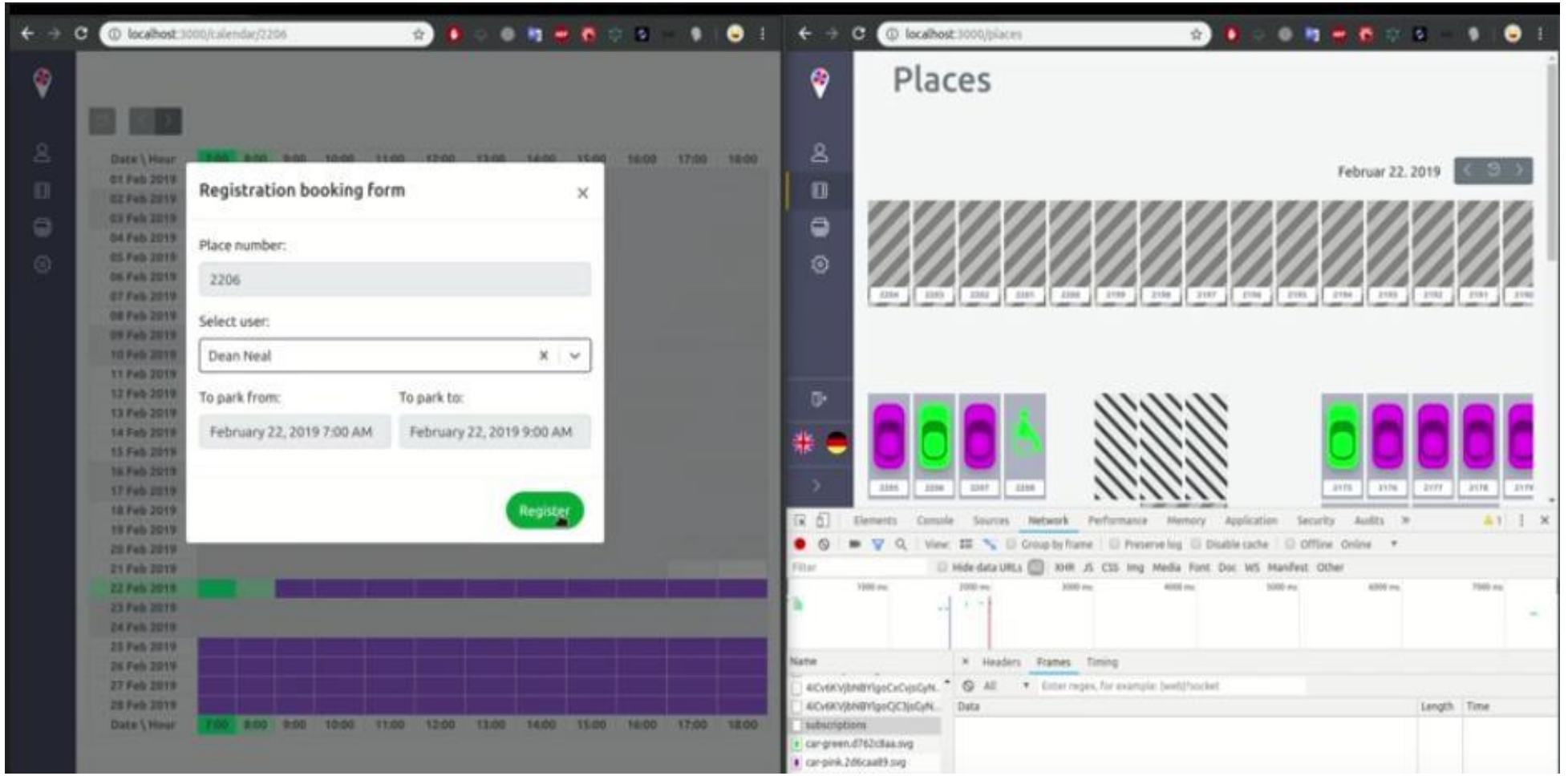

Рисунок 2.4 – Графічне представлення запиту про наявність вільних місць

Також система реалізує такий сервіс, за допомогою якого власники центрів можуть реєструвати свої площі з паркомісцями та здійснювати керування ними, а власники автомобілів можуть переглядати список центрів і місць паркування, орендувати, забронювати, а також вказувати, що орендоване клієнтом місце вільно в певний проміжок часу, це дозволить відстежити зв'язок кожного користувача та зберігати таку інформацію, як час на якій був припаркований автомобіль та тривалість часу для паркування автомобіль (рис. 2.4).

Для перевірки роботи сервісу проводилося тестування протягом лютого 2019 року. Отримані статистичні дані показали, що використання сервісу дає змогу оцінити вільні та зайняті паркомісця протягом доби за певні проміжки часу, відповідність попередньому бронюванню. Такий підхід оптимізував заповненість паркомісць на 19%.

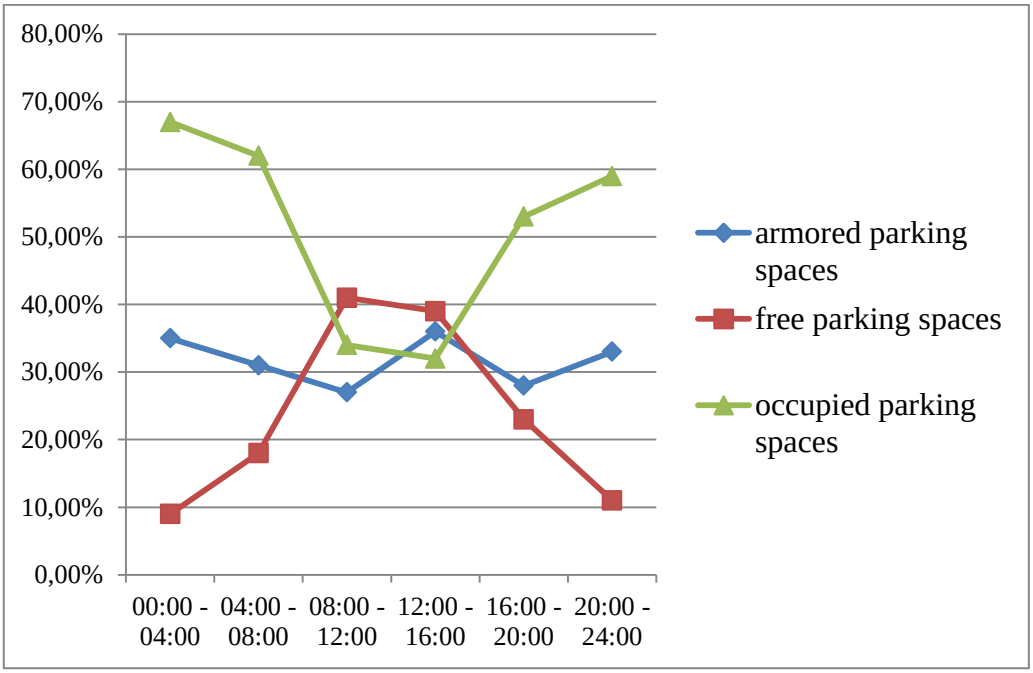

Рисунок 2.5 – Навантаження центру паркування

3. РОЗРОБКА БАЗИ ДАНИХ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ

3.1 Робота з базами даних в Go

Є спеціально написана бібліотека gorm.io [13]. Фантастична бібліотека ORM для Golang прагне бути дружньою до розробника. Бібліотека має досить гнучкий інтерфейс. Підтримує діалекти: sqlite, mysql, postgres, mssql. Також є необхідні функції для роботи з бд:

– Associations (Has One, Has Many, Belongs To, Many To Many, Polymorphism)
– Hooks (Before/After Create/Save/Update/Delete/Find)
– Preloading (eager loading)
– Transactions
– Composite Primary Key
– SQL Builder
– Auto Migrations
– Logger
– Extendable, write Plugins based on GORM callbacks

До того ж, завдяки чудової мови, як Golang у бібліотеки є всі тести, які показують працездатність всіх перерахованих вище функцій.

Початок роботи дуже просте, потрібно оголосити модель (структура на Go) і імпортувати драйвер бази даних:

```
import _ "github.com/jinzhu/gorm/dialects/postgres"

type User struct {
 gorm.Model
 Name        string
 Age         sql.NullInt64
 Birthday    *time.Time
```

```
Email       string  `gorm:"type:varchar(100);unique_index"`
Role        string  `gorm:"size:255"` // set field size to 255
MemberNumber *string `gorm:"unique;not null"` // set member number to unique and not null
Num         int     `gorm:"AUTO_INCREMENT"` // set num to auto incrementable
Address     string  `gorm:"index:addr"` // create index with name `addr` for address
IgnoreMe    int     `gorm:"-"` // ignore this field
}
```

Після чого необхідно під'єднатися до бази даних в main () функції та запустити міграції:

```
func main() {
dsn := fmt.Sprintf("host=%s port=%s user=%s dbname=%s password=%s sslmode=disable", "localhost", "5432", "user", "dbexample", "123")
conn, err := gorm.Open("postgres", dsn)
if err != nil {
 fmt.Println(err)
}
conn.AutoMigrate(&entity.User{})
}
```

Оголосимо моделі (структури) пов'язані між собою. Структури **Company, Garage, Place, User, Booking**.

```
type Booking struct {
ID      int64    `gorm:"primary_key" json:"id"` // record uid
```

CreatedAt time.Time `gorm:"column:created_at;default:current_date;not null" json:"createdAt"` // when record was created

Duration Duration `gorm:"type:tsrange;column:duration;not null" json:"duration"` // time that indicates from which time is free or busy a place

UserID int `json:"userId"` // by who place is busy

User User `gorm:"foreignkey:UserID" json:"user"` // user json struct

PlaceID int `json:"placeId"` // booking related to place

IsBusy bool `gorm:"-" json:"isBusy"`

CompanyID int `json:"companyId"`

}

type Duration struct {

From time.Time `json:"from"`

To time.Time `json:"to"`

}

Завдяки додатковому оголошенню у зворотних лапках `json: fieldName` є можливість перекодувати дані в формат json з ключовим полем fieldName. А також за допомогою `gorm: *` можна додати додаткові параметри атрибуту - первинний ключ, найменування стовпця, тип і т.д. (дод. А)

Перед тим, як подивитися на роботу з базою даних - нам потрібен сервер PostgreSQL. Скористаємося системою контейнеризації Docker для швидкого запуску БД docker-compose.yml:

 # vi: tw=2 ts=2 sw=2 et

version: '3.7'

services:

```
db:
  image: postgres:latest
  container_name: dbexample
  restart: always
  environment:
    POSTGRES_USER: "user"
    POSTGRES_PASSWORD: "123"
    POSTGRES_DB: "dbexample"
  ports:
    - 5432:5432
```

Виконаємо команду docker-compose up -d. Дану команду необхідно виконувати в корені директорії, де розташований docker-compose.yml файл. Прапор -d необхідний для запуску контейнера в демона.

Виконаємо команду docker container ls, щоб перевірити запустився чи контейнер з бд успішно. Результат:

CONTAINER ID	IMAGE	COMMAND	CREATED	STATUS	PORTS	NAMES
e2b3408 6fb7a	postgres :latest	"docker-entrypoint.s…"	21 seconds ago	Up 19 seconds	0.0.0.0: 5432->5432/tcp	dbexample

Тепер запустимо міграції в нашій main() функції (дод. Б):

Оскільки у нашого поля Durations повинен бути специфічний тип tsrange який нам надає за умовчанням Postgres. Нам необхідно виконати row sql скрипт міграції.

Таблиці були успішно створені, схеми таблиць зображені на рис. 3.1.

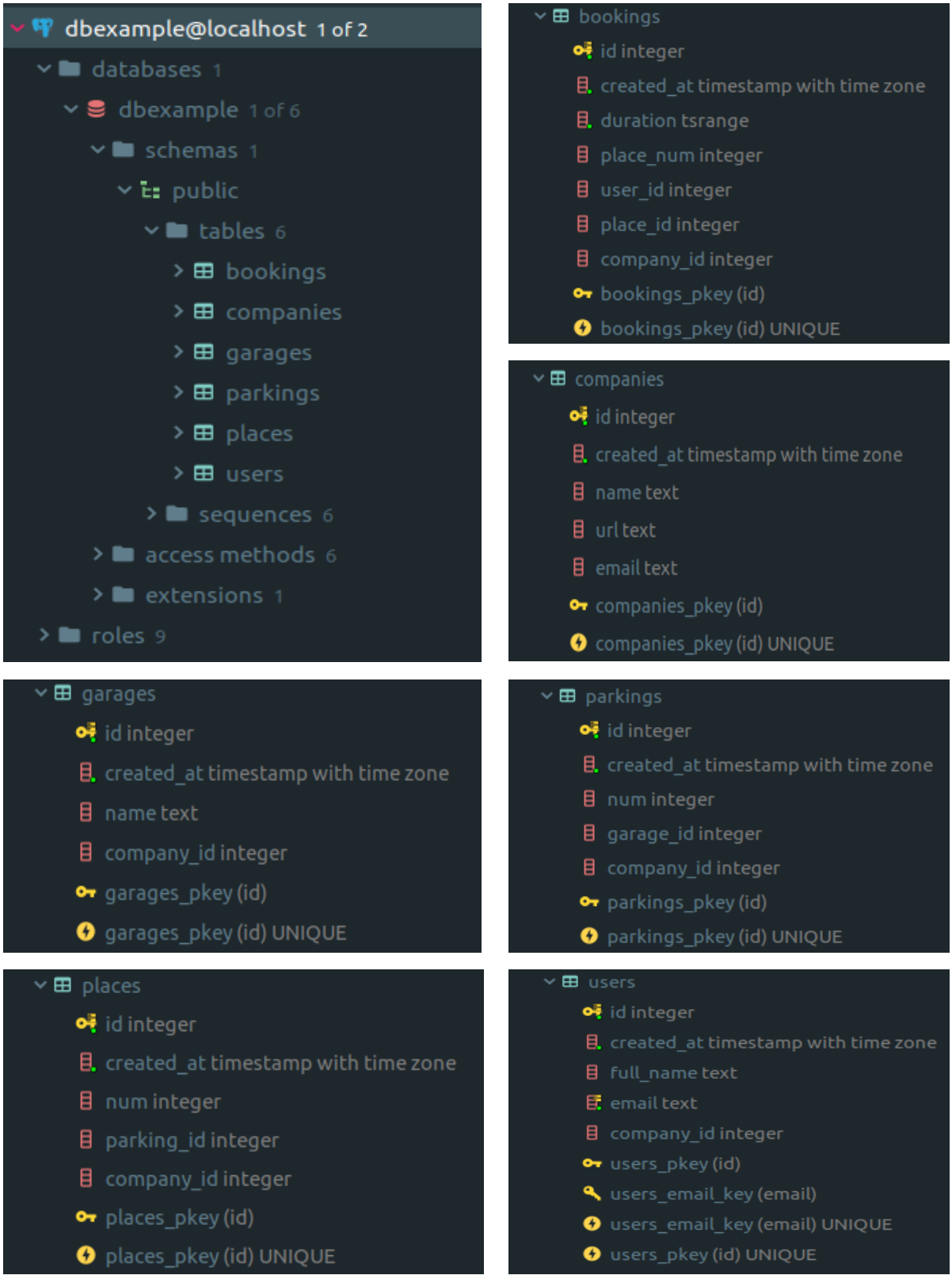

Рисунок 3.1 – Таблиці БД

Тепер оголосимо змінні з даними для БД.

Подивимося на дані в бд (рис. 3.2).

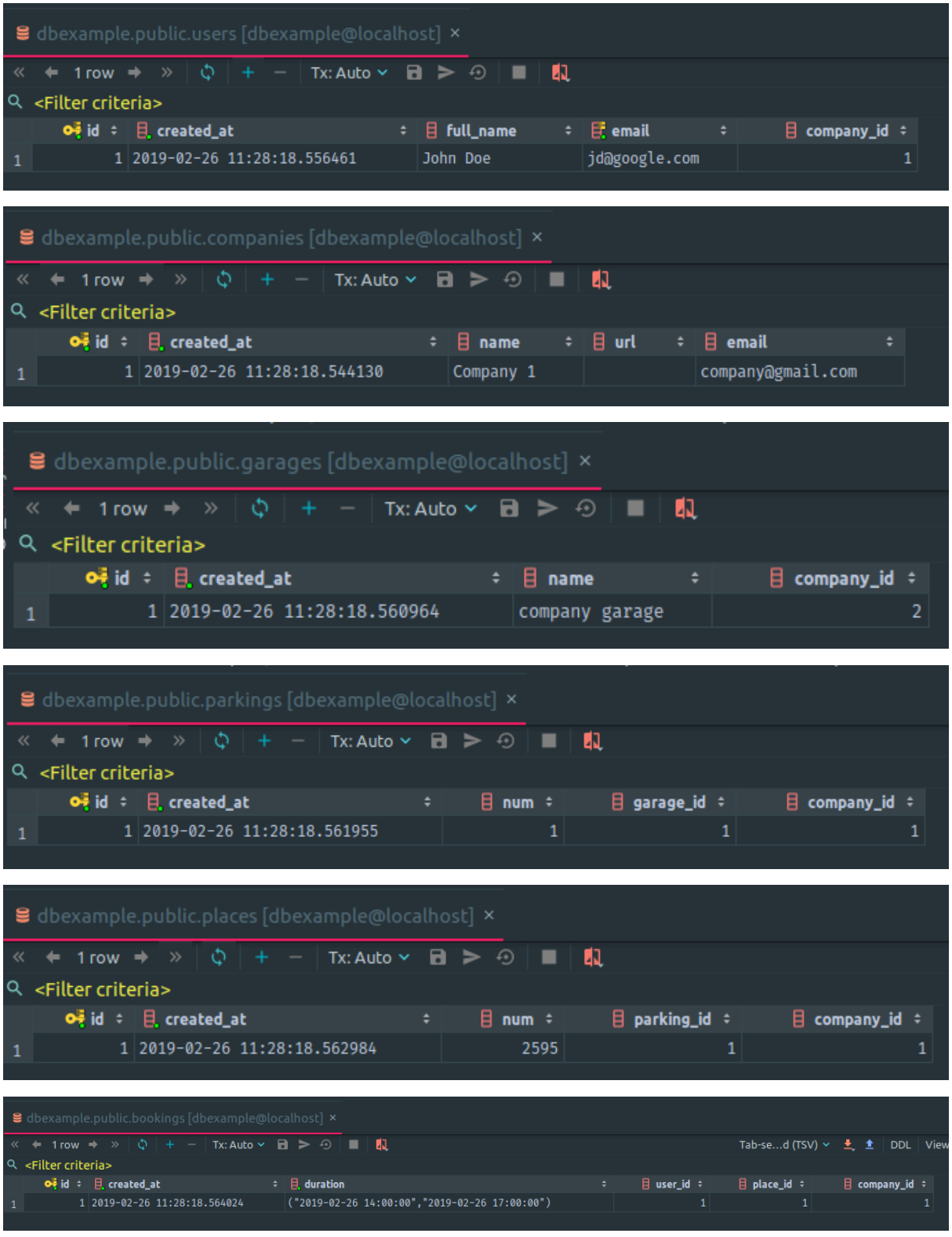

Рисунок 3.2 – Дані таблиць

І запишемо наші дані в БД.

```
if err := conn.Save(&company).Error; err != nil {
fmt.Println(err)
}
if err := conn.Save(&user).Error; err != nil {
fmt.Println(err)
}
if err := conn.Save(&garage).Error; err != nil {
fmt.Println(err)
}
```

3.2 Структура БД

Структура БД паркувальної системи складається з 5 сутностей - **Company, Garage, Place, User, Booking**. (табл. 3.1)

Таблиця 3.1 – Атрибути сутностей

Entities					
attrs	**Company**	**Garage**	**Place**	**User**	**Booking**
1	id	id	id	id	id
2	name	location	coordinates	full_name	duration
3	url	company_id	type	email	place_id
4	email		garage_id	company_id	user_id
5			company_id		company_id

Схема даних, співвідношення між сутностями, зображена на рис. 3.3.

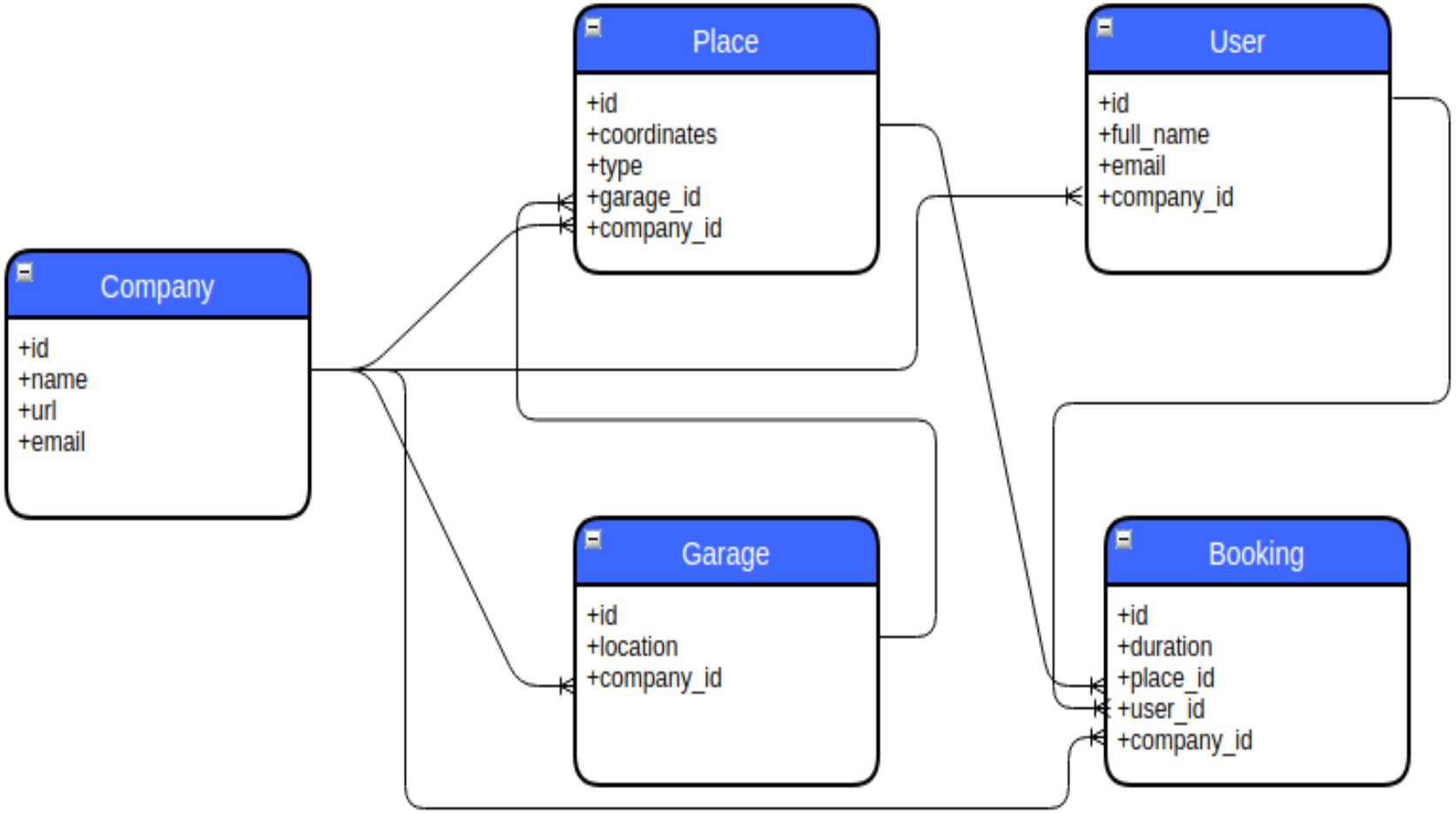

Рисунок 3.3 - Схема даних

4 ОХОРОНА ПРАЦІ

У підрозділі зазначають основні правові та організаційні аспекти з охорони праці, що прийняті як на державному рівні, так і затверджені в організації/виробництві на майбутньому місці праці.

4.1 Аналіз небезпечних та шкідливих факторів при виробництві (експлуатації) виробу

Основними характеристиками персонального комп'ютера є наступні:

1) робоча напруга U=+220В +-5%;
2) робочий струм I=2А;
3) споживана потужність Р=350 Вт.

Роботу користувача розробленої підсистеми слід віднести до категорії Іа (легкі фізичні роботи) відповідно до даної категорії відносяться всі види діяльності, які виконуються сидячи й не вимагають фізичного напруження.

При експлуатації даного програмного продукту відповідно існують наступні небезпечні й шкідливі виробничі фактори:

1) фізичні:

a) підвищений рівень напруги електричної мережі, замикання якої може відбутися через тіло людини;

b) підвищена або знижена вологість повітря;

c) підвищений рівень статичної електрики;

d) підвищена напруженість електричного й магнітного полів;

e) відсутність або нестача природного світла;

f) знижена освітленість робочої зони;

g) підвищений рівень шуму на робочому місці;

h) підвищений рівень електромагнітного випромінювання;

i) знижена контрастність.

2. психофізіологічні:

a) фізичні перевантаження: статичні та динамічні;

b) нервово-психічні перевантаження: розумове перенапруження, монотонність праці, перенапруження аналізаторів та емоційні перевантаження.

4.2 Електробезпека

Основним небезпечним фактором при роботі з ЕОМ є небезпека ураження людини електричним струмом, яка посилюється тим, що органи чуття людини не можуть на відстані виявити наявність електричної напруги на устаткуванні. Проходячи через тіло людини, електричний струм справляє на нього складну дію, що є сукупністю термічної (нагрів тканин і біологічних середовищ), електролітичної (розкладання крові і плазми) і біологічної (подразнення і збудження нервових волокон та інших органів тканин організму) дій.

1) Тяжкість ураження людини електричним струмом залежить від цілого ряду чинників: значення сили струму;

2) електричного опору тіла людини і тривалості протікання через нього струму;

3) роду і частоти струму;

4) індивідуальних властивостей людини і навколишнього середовища.

Відповідно до, приміщення для ЕОМ відноситься до приміщень без підвищеної небезпеки, тобто до приміщень, у яких відсутні умови, що створюють підвищену або особливу небезпеку. Небезпека ураження електричним струмом існує всюди, де використовуються електроустановки, тому приміщення без підвищеної небезпеки не можна назвати безпечними.

Електробезпека забезпечується:

1) відповідною конструкцією електроустановок;

2) застосуванням технічних способів і засобів захисту;

3) організаційними і технічними заходами.

Конструкція електроустановок відповідає умовам їхньої експлуатації і забезпечує захист персоналу від дотику до струмоведучих частин. Основними технічними способами та засобами захисту від ураження електричним струмом, які використовуються окремо або в поєднанні один з одним, є:

1) захисне заземлення;
2) занулення;
3) вирівнювання потенціалів;
4) мала напруга;
5) електричне розділення мереж;
6) захисне відключення;
7) ізоляція струмоведучих частин;
8) компенсація струмів замикання на землю;
9) захисні пристрої;
10) попереджувальна сигналізація, блокування, знаки безпеки;
11) ізолюючі захисні і запобіжні пристосування.

Основними технічними способами і засобами захисту від ураження електричним струмом, що передбачаються в даному дипломному проекті, є: захисне заземлення; занулення; захисне відключення; ізоляція струмоведучих частин.

Занулення в комплексі із захисним відключенням зменшує напругу дотику і обмежує час, в перебігу якого людина, доторкнувшись до корпусу, може потрапити під дію напруги.

4.3 Освітлення

Трудова діяльність людини завжди протікає в певних метеорологічних умовах, які визначаються поєднанням температури повітря, швидкості його руху і відносної вологості, тиском і тепловим випромінюванням від нагрітих

поверхонь. Оскільки експлуатація проектованого програмного засобу відбувається в приміщенні, то ці показники в сукупності (за винятком тиску) називаються мікрокліматом виробничого приміщення. В даний час основним нормативним документом нормалізації мікроклімату є ДСН 3.3.6.042-99 «Державні санітарні норми мікроклімату» [14].

Важкість праці характеризує сукупну дію всіх елементів, що складають умови праці, на працездатність людини, його здоров'я, життєдіяльність і відновлення робочої сили. У такому представлені поняття важкості праці однаково застосовне як до розумової, так і до фізичної праці. Оптимальні норми мікроклімату, згідно санітарних норм ДСН 3.3.6.042-99 [14], в робочій зоні, забезпечувані для робіт легкої категорії 1б приведені в табл. 4.1.

У приміщенні, де знаходяться ЕОМ, повітрообмін реалізується за допомогою природної організованої вентиляції (вентиляційні шахти) і установки кондиціонера БК-2000. Цей метод забезпечує приток потрібної кількості свіжого повітря, що визначається в ДСН 3.3.6.042-99 [14] (30 кубічних метрів на годину на одного працюючого) .

Таблиця 4.1 - Оптимальні норми мікроклімату

Період року	Температура, °С	Відносна вологість, %	Швидкість руху повітря, м/с, не більш
Холодний і перехідної	21 - 23	60 - 40	0,1
Теплий	22 - 24	60 - 40	0,2

Для захисту від електромагнітного випромінювання передбачаються наступні заходи:

1. застосування нових плазмових моніторів;
2. віддалення робочого місця не менше, ніж на 0,4 - 0,5 м, оскільки напруженість електричного поля зменшується при віддаленні від джерела поля;

3. встановлення раціональних режимів роботи персоналу (обмеження часу перебування);

4. раціональне розміщення в робочому приміщенні устаткування, що випромінює електромагнітну енергію.

Оскільки рівень шуму не перевищує гранично допустимих величин, які встановлені санітарними нормами ДСН 3.3.6.037-99 [15], заходи для зниження шуму не проводяться.

Збільшення освітленості сприяє поліпшенню працездатності навіть в тих випадках, коли процес праці практично не залежить від зорового сприйняття. При поганому освітленні людина швидко втомлюється, працює менш продуктивно, виникає потенційна небезпека помилкових дій і нещасних випадків. У проекті, що розробляється, передбачається використовувати суміщене освітлення. У світлий час доби використовуватиметься природне освітлення приміщення через віконні отвори, в решту часу використовуватиметься штучне освітлення. Штучне освітлення створюється газорозрядними лампами. Штучне освітлення в робочому приміщенні передбачається здійснювати з використанням люмінесцентних джерел світла в світильниках загального освітлення, оскільки люмінесцентні лампи мають високу потужність (80 Вт), тривалий термін служби (до 10000 годин), спектральний складом випромінюваного світла, близький до сонячного. При експлуатації ЕОМ виконується зорова робота IV в розряду точності (середня точність). При цьому нормована освітленість на робочому місці (Ен) рівна 200 лк. Джерелом природного освітлення є сонячне світло. У приміщенні, де розташовані ЕОМ передбачається природне бічне освітлення, рівень якого відповідає ДБН В.2.5-28:2018 [16]. Регулярно повинен проводитися контроль освітленості, який підтверджує, що рівень освітленості задовольняє ДБН В.2.5-28:2018 [16] і для даного приміщення в світлий час доби достатньо природного освітлення. Світильники загального освітлення розташовуються над робочими поверхнями в рівномірно-прямокутному порядку. Для організації

освітлення в темний час доби передбачається обладнати приміщення, довжина якого складає 5 м, ширина 4 м, світильниками ЛПО2П, оснащеними лампами типу ЛБ (дві по 80 Вт) з світловим потоком 5400 лм кожна.

Розрахунок штучного освітлення виробляється по коефіцієнтах використання світлового потоку, яким визначається потік, необхідний для створення заданої освітленості при загальному рівномірному освітленні. Розрахунок кількості світильників n виробляється по формулі (4.1):

$$N = E \cdot S \cdot Z \cdot K / F \cdot U \cdot M \qquad (4.1)$$

де E – нормована освітленість робочої поверхні, визначається нормами – 300лк;

S– освітлювана площа, м2; S = 20 м2;

Z – поправочний коефіцієнт світильника (Z = 1,15 для ламп розжарювання та ДРЛ; Z = 1,1 для люмінесцентних ламп) приймаємо рівним 1,1;

K – коефіцієнт запасу, що враховує зниження освітленості в процесі експлуатації – 1,5;

U – коефіцієнт використання, залежний від типу світильника, показника індексу приміщення і т.п. – 0,575;

M – число люмінесцентних ламп в світильнику – 2;

F – світловий потік лампи – 5400лм (для ЛБ-80).

Підставивши числові значення у формулу (4.1), отримуємо:

$$n=(300*20*1.1*1.5)/(5400*0.575*2)=1.59$$

Приймаємо освітлювальну установку, яка складається з 2-х світильників, які складаються з двох люмінесцентних ламп загальною потужністю 160 Вт, напругою – 220 В.

4.4 Заходи з організації виробничого середовища та попередження виникнення надзвичайних ситуацій

Застосовують різні електричні захисні засоби від ураження струмом:

а) *ізолюючі* - ізолюють людини від струмоведучих або заземлених частин, а так-же від землі;

б) *основні* - володіють ізоляцією, здатної довго витримувати робочу напругу електроустановки і тому ними дозволяється стосуватися струмоведучих частин, що знаходяться під напругою;

в) *запобіжні* - володіють ізоляцією нездатною витримати робоча напруга електроустановки, і тому вони не можуть самостійно захищати людину від ураження струмом цим напругою.

Розрахунок захисного заземлення (забезпечення електробезпеки будівлі).

Відповідно до класифікації приміщень за ступенем небезпеки ураження електричним струмом ДСТУ Б В.2.5-82:2016 [17], приміщення в якому проводяться всі роботи належить до першого класу (без підвищеної небезпеки). Під час роботи використовуються електроустановки з напругою живлення 36 В, 220 В, і 360 В. Опір контуру заземлення повинен мати не більше 4 Ом.

Розрахунок проводять за допомогою методу коефіцієнта використання (екранування) електродів. Коефіцієнт використання групового заземлення η - це ставлення чинної провідності цього заземлення до найбільш можливої його провідності при нескінченно великих відстаней між його електродами. Коефіцієнт використання вертикальних заземлювачів $\eta_в$ у залежності від розміщення заземлювачів і їх кількості знаходиться в межах 0,4 ... 0,99. Взаємну екрануючого дії горизонтального заземлювача (сполучної смуги) враховують за допомогою коефіцієнта використання горизонтального заземлювача $\eta_с$.

Послідовність розрахунку:

1) Визначається необхідний опір штучних заземлювачів $R_{шт.з.}$:

$$R_{шт.з} = (R_д \cdot R_{пр.з})/(R_{пр.з} - R_д) \quad (4.2)$$

де $R_{пр.з.}$ – опір природних заземлювачів; $R_д$ – допустимий опір заземлення. Якщо природні заземлювачі відсутні, то $R_{шт.з}.=R_д$.

Підставивши числові значення в формулу (4.2), отримуємо:

$$R_{шт.з.} = \frac{4 \cdot 40}{40 - 4} \approx 4 \text{ Ом}$$

2) Опір заземлення в значній мірі залежить від питомого опору ґрунту ρ, Ом/м. Приблизне значення питомої опору глини приймаємо ρ=40 Ом/м (табличне значення).

3) Розрахункова питомий опір ґрунту, $\rho_{розр}$, Ом•м, визначається відповідно для вертикальних заземлювачів $\rho_{розр.в}$, і горизонтальних $\rho_{розр.г}$, Ом/м по формулі:

$$\rho_{розр..} = \psi \cdot \rho, \quad (4.3)$$

де ψ – коефіцієнт сезонності для вертикальних заземлювачів і кліматичної зони з нормальною вологістю землі, приймається для вертикальних заземлювачів $\rho_{розр.в}$=1,7 і горизонтальних $\rho_{розр.г}$=5,5 Ом/м.

$$\rho_{розр.в} = 1,7 \cdot 40 = 68 \text{ Ом/м}, \quad \rho_{розр.г} = 5,5 \cdot 40 = 220 \text{ Ом/м}$$

1) Розраховується опір розтікання струму вертикального заземлення $R_в$, Ом, по (4.4).

$$R_в = \frac{\rho_{розр.в}}{2 \cdot \pi \cdot l_в} \cdot \left(\ln \frac{2 \cdot l_в}{d_{ст}} + \frac{1}{2} \cdot \ln \frac{4 \cdot t + l_в}{4 \cdot t - l_в} \right), \quad (4.4)$$

де $l_в$ – довжина вертикального заземлювача (для труб - 2−3 м; $l_в$=3 м); $d_{ст}$ – діаметр стрижня (для труб - 0,03−0,05 м; $d_{ст}$=0,05 м); t – відстань від поверхні землі до середини заземлювача, яка визначається за формулою (4.5):

$$t = h_в + \frac{l_в}{2}, \quad (4.5)$$

де $h_в$ – глибина закладення вертикальних заземлювачів (0,8 м); тоді

$$t = 0,8 + \frac{3}{2} = 2,3 \text{ м}$$

$$R_в = \frac{68}{2 \cdot \pi \cdot 3} \cdot \left(\ln \frac{2 \cdot 3}{0,05} + \frac{1}{2} \cdot \ln \frac{4 \cdot 2,3 + 3}{4 \cdot 2,3 - 3} \right) = 18,5 \text{ Ом}$$

2) Визначається теоретична кількість вертикальних заземлювачів n штук, без урахування коефіцієнта використання $\eta_в$:

$$n = \frac{2 \cdot R_в}{R_д} = \frac{2 \cdot 18,5}{4} = 9,25$$

І визначається коефіцієнт використання вертикальних електродів групового заземлення без урахування впливу сполучної стрічки $\eta_в$ =0,57 (табличне значення).

Визначається необхідна кількість вертикальних заземлювачів з урахуванням коефіцієнта використання $n_в$, шт:

$$n_в = \frac{2 \cdot R_в}{R_д \cdot \eta_в} = \frac{2 \cdot 18,5}{4 \cdot 0,57} = 16,2 \approx 16$$

3) Визначається довжина сполучної стрічки горизонтального заземлювача $l_с$, м:

$$l_с = 1,05 \cdot L_в \cdot (n_в - 1), \tag{4.6}$$

де $L_в$ – відстань між вертикальними заземлювачами, (прийняти $L_в$ = 3м); $n_в$ – необхідну кількість вертикальних заземлювачів.

$$l_с = 1,05 \cdot 3 \cdot (16 - 1) \approx 48_{м}$$

8) Визначається опір розтіканню струму горизонтального заземлювача (сполучної стрічки) $R_г$, Ом:

$$R_г = \frac{\rho_{розр.г}}{2 \cdot \pi \cdot l_с} \cdot \ln \frac{2 \cdot l_с^{\,2}}{d_{см} \cdot h_г}, \tag{4.7}$$

де $d_{см}$ – еквівалентний діаметр смуги шириною b, $d_{см}$ = 0,95b, b = 0,15 м; $h_г$ – глибина закладення горизонтальних заземлювачів (0,5 м); $l_с$ - довжина сполучної стрічки горизонтального заземлювача $l_с$, м

$$R_г = \frac{220}{2 \cdot \pi \cdot 48} \cdot \ln \frac{2 \cdot 48^2}{0,95 \cdot 0,15 \cdot 0,5} = 8,1$$

9) Визначається коефіцієнт використання горизонтального заземлювача ηс відповідно до необхідної кількості вертикальних заземлювачів $n_в$.

Коефіцієнт використання сполучної смуги $\eta_с$=0,3 (табличне значення).

10) Розраховується результуючий опір заземлюючого електрода з урахуванням сполучної смуги:

$$R_{заг} = \frac{R_в \cdot R_г}{R_в \cdot \eta_с + R_г \cdot n_в \cdot \eta_в} \le R_д. \quad (4.8)$$

Висновок: дане захисне заземлення буде забезпечувати електробезпека будівлі, так як виконується умова: $R_{заг}$ <4 Ом, а саме:

$$R_{заг} = \frac{18,5 \cdot 8,1}{18,5 \cdot 0,3 + 8,1 \cdot 16 \cdot 0,57} = 1,9 \le R_д$$

При виникненні пожеж при роботі на ПЕОМ від таких можливими джерел запалювання як: іскри і дуги коротких замикань; перегрів провідників, резисторів і інших радіодеталей ПЕОМ [18-19], від тривалого перевантаження і наявність перехідного опору; іскри при розмиканні і розмиканні ланцюгів; розряди статичної електрики; необережне поводження з вогнем, а також вибухи газоповітряних і пароповітряних сумішей. Важливу увагу слід звернути на пожежну безпеку підприємства в цілому і окремих його приміщень. У приміщеннях не повинно накопичуватися сміття, непотрібну папір, мотлох та ін. Речі, які не використовуються у виробничому процесі. Наявний вільний аварійний вихід за межі приміщення в разі пожежі, бути передбачені вогнегасники. Вони повинні бути в робочому стані і перевірятися відповідно до норм. У приміщеннях повинна бути пожежна сигналізація, вогнегасник. У разі виникнення пожежі необхідно повідомити в найближчу пожежну частину, убезпечити інших працівників і по можливості прийняти кроки щодо запобігання можливих наслідків та усунення пожежі.

4.5 Висновок до розділу 4

В результаті проведеної роботи було зроблено аналіз умов праці, шкідливих та небезпечних чинників, з якими стикається робітник. Було визначено параметри і певні характеристики приміщення для роботи над

запропонованим проектом написаному в кваліфікаційній роботі, описано, які заходи потрібно зробити для того, щоб дане приміщення відповідало необхідним нормам і було комфортним і безпечним для робітника.

Приведені рекомендації щодо організації робочого місця, а також важливу інформацію щодо пожежної та електробезпеки. Було наведено значення температури, вологості й рухливості повітря, необхідна кількість і потужність ламп та інші параметри, значення яких впливає на умови праці робітника, а також – наведені інструкції з охорони праці, техніки безпеки при роботі на комп’ютері.

ВИСНОВКИ

В роботі пропонується функціонал сервісу, який використовується для моніторингу паркувальних центрів, інформування користувачів про наявність місць та бронювання / управління паркувальним місцем для клієнтів. Основною відмінністю запропонованого рішення є можливість пошуку безкоштовного паркувального центру на найкоротшій відстані від поточного місця розташування власника автомобіля.

Потенційними користувачами такої інформаційної системи є звичайні люди, що мають автомобіль. Проект корисний одночасно і для користувачів, так і для власників паркувальних майданчиків.

Переваги продукту полягають в універсальності проекту. Сервіс використовується не лише для моніторингу кожної окремої автостоянки та інформування користувача про наявність місць для паркування, а ще й з можливістю бронювання та керування паркомісцем для клієнтів сервісу.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. http://www.worldometers.info/world-population/

2. https://www.statista.com/statistics/200002/international-car-sales-since-1990

3. Dutzik T. A New Way to Go. The Transportation Apps and Vehicle-sharing. Tools that Are Giving More Americans. The Freedom to Drive Less / T. Dutzik, T. Madsen, P. Baxandall // U.S. PIRGEducation Fund. - 2013.

4. Siuhi S. Opportunities and challenges of smart mobile applications in transportation / S. Siuhi, J. Mwakalonge // Journal of traffic and transportation engineering (english edition). - 2016. – No. 3 (6). – pp. 582 - 592.

5. National Highway Transportation Safety Administration (NHTSA), 2013. Safercar. Available at: http://www.safercar.gov/

6. Tennessee Department of Transportation (TDOT), 2014. TDOT SmartWay Mobile App. Available at: http://www.tdot.state.tn.us/tdotsmartway/mobile/

7. Ji Y. Understanding drivers' perspective on parking guidance information / Y. Ji, W. Guo, P. Blythe, D. Tang, W. Wang // IET Intell. Transp. Syst. – 2014. - vol. 8, no. 4. - pp. 398–406.

8. Rajabioun T. On-street and off-street parking availability prediction using multivariate spatiotemporal models / T. Rajabioun, P. Ioannou // IEEE Trans. Intell. Transp. Syst. – 2015. - vol. 16, no. 5. - pp. 2913–2924.

9. Khanna A. IoT based Smart Parking System / A. Khanna, R. Anand // International Conference on Internet of Things and Applications (IOTA) Maharashtra Institute of Technology. - 2016. – pp. 266 - 270.

10. Harshal V. Parking Guidance system using Internet of Things / V. Harshal, G. Mulay, V. Gohokar // International Conference on Inventive Computation Technologies (ICICT 2016) . – 2016. - pp. 518-523.

11. Деркач М.В. Розробка мобільного додатку для приватних центрів паркування / М.В. Деркач, В.В. Лисак, І.С. Скарга-Бандурова // Вісник Східноукраїнського національного університету імені Володимира Даля. –

Сєвєродонецьк: СНУ ім.В.Даля, 2018. – № 6 (247). – С.38-41.

12. Деркач М.В. Створення сервісу для підвищення якості наданих послуг центрів паркування / М.В. Деркач, В.В. Лисак, І.С. Скарга-Бандурова // ІТ-Ідея – 2018: збірник науково-практичних праць. – Сєвєродонецьк: СНУ ім.В.Даля, 2018. – С.67–69.

13. Lysak V. Interaction Golang with PostgreSQL database / V. Lysak, M. Derkach // Advanced Technologies in Research and Education: collection of research materials of the Second International Conference. – Severodonetsk: Volodymyr Dahl East Ukrainian National University, 2019. – P. 96-98.

14. Державні санітарні норми. ДСН 3.3.6.042-99 «Санітарні норми мікроклімату виробничих приміщень» - Режим доступу: https://zakon.rada.gov.ua/rada/show/va042282-99 - 01.02.1999 р.

15. Державні санітарні норми. ДСН 3.3.6.037-99 «Санітарні норми виробничого шуму, ультразвуку та інфразвуку» - Режим доступу: https://zakon.rada.gov.ua/rada/show/va037282-99 - 01.12.1999 р.

16. Державні будівельні норми. ДБН В.2.5-28:2018 «Природне і штучне освітлення» - Режим доступу: http://www.minregion.gov.ua/wp-content/uploads/2018/12/V2528-1.pdf - 03.10.2018 р.

17. Державний стандарт України. ДСТУ Б В.2.5-82:2016 «Електробезпека в будівлях і спорудах. Вимоги до захисних заходів від ураження електричним струмом» - Режим доступу: http://epicentre.co.ua/dstu/doc28522.html - 01.07.2016 р.

18. Державні санітарні правила і норми. ДСанПіН 3.3.2.007-98 «Державні санітарні правила і норми роботи з візуальними дисплейними терміналами електронно-обчислювальних машин» - Режим доступу: https://zakon.rada.gov.ua/rada/show/v0007282-98 - 10.12.1998 р.

19. Нормативно-правовий акт з охорони праці. НПАОП 0.00-7.15-18 «Вимоги щодо безпеки та захисту здоров’я працівників під час роботи з екранними пристроями» - Режим доступу: https://zakon.rada.gov.ua/laws/show/z0508-18 - 14.02.2018 р.

ДОДАТОК А

Електроні плакати

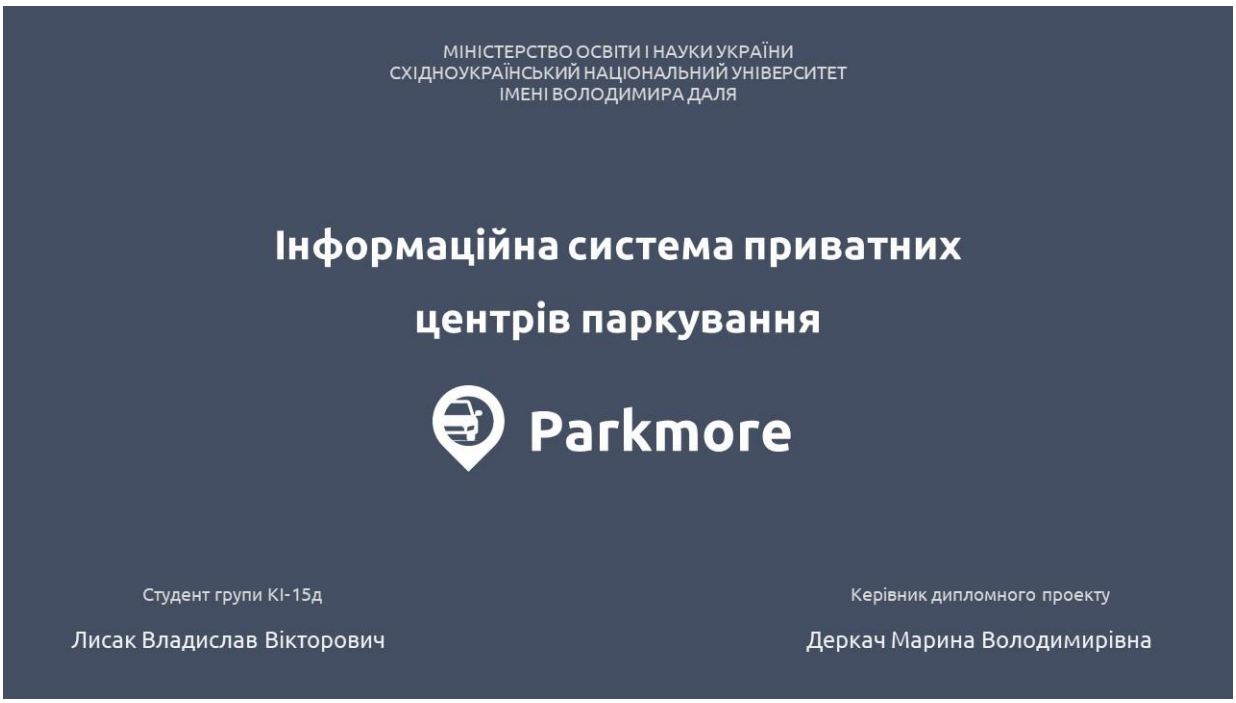

Рисунок А.1 - Слайд №1

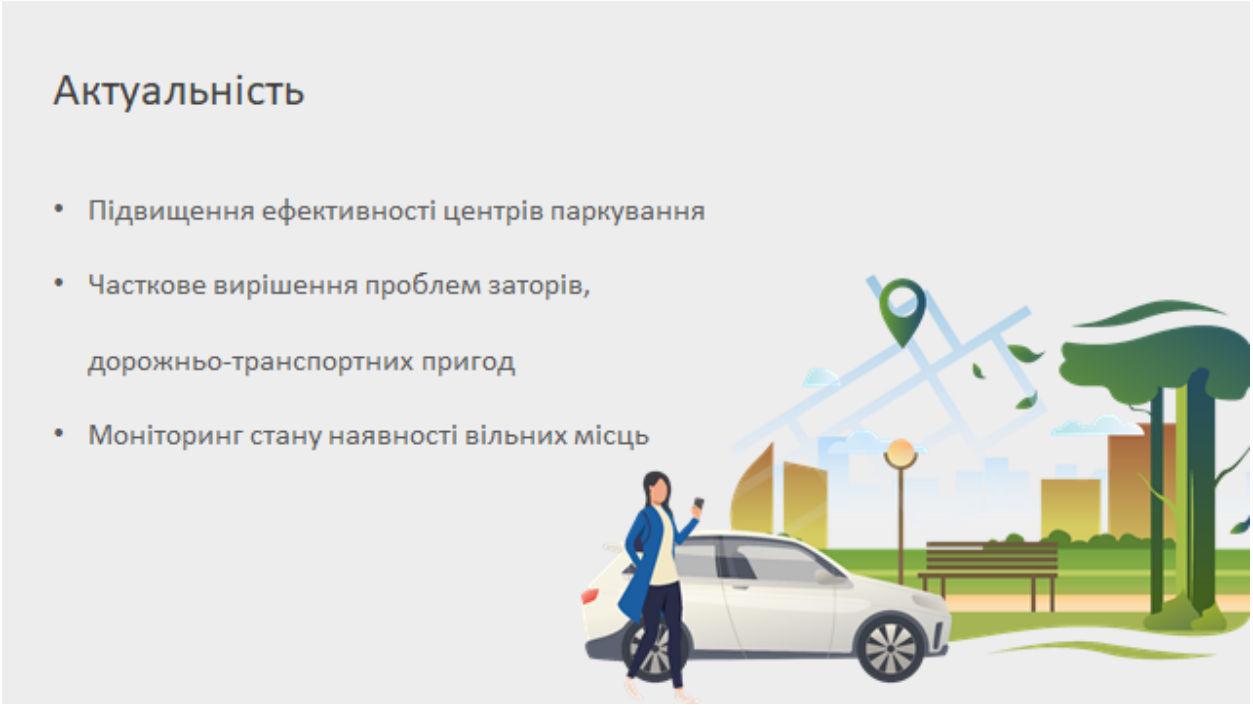

Рисунок А.2 - Слайд №2

Рисунок А.3 - Слайд №3

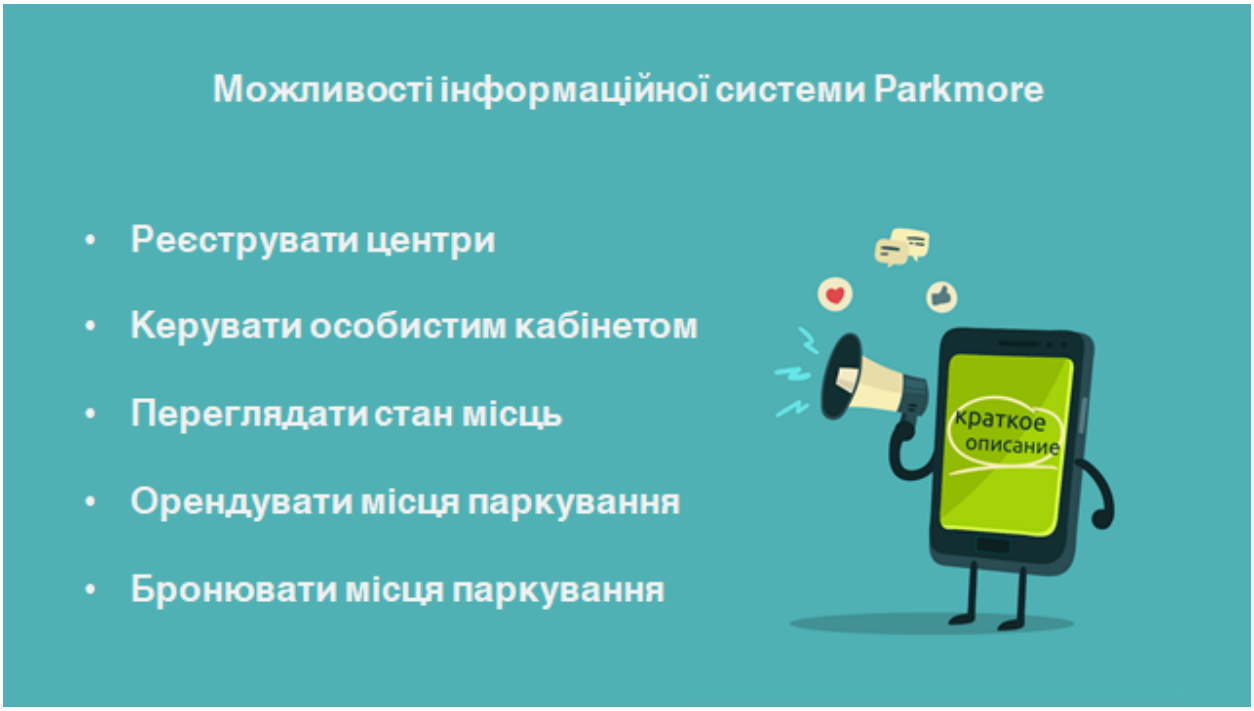

Рисунок А.4 - Слайд №4

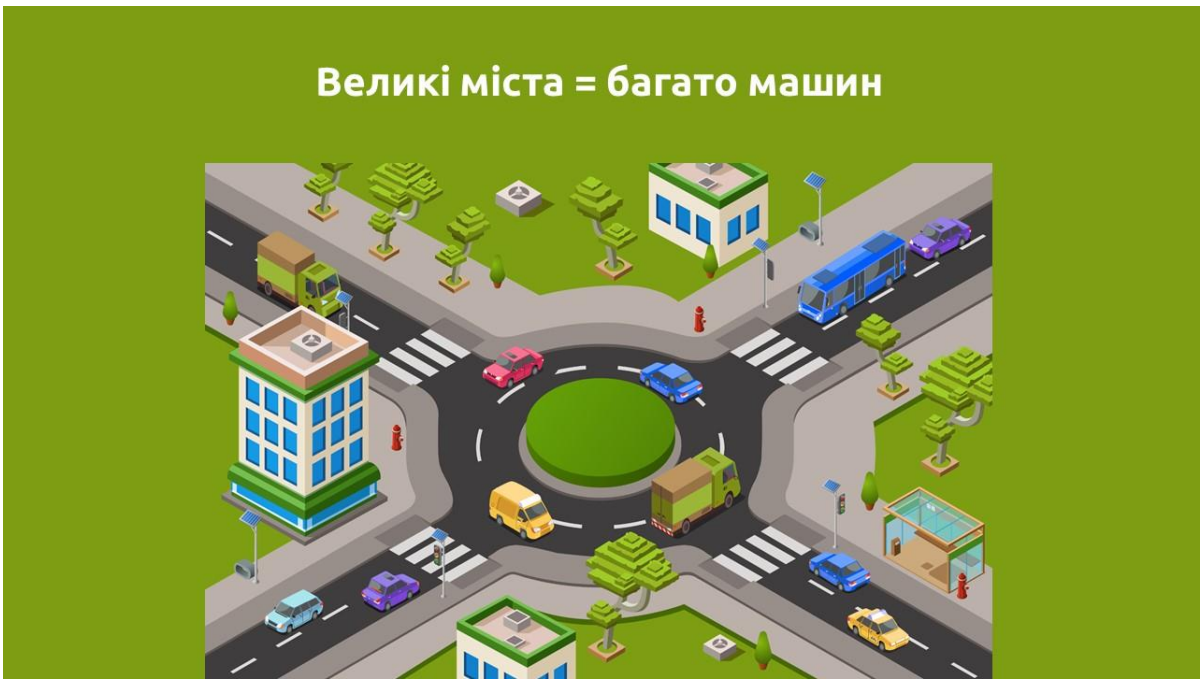

Рисунок А.5 - Слайд №5

Рисунок А.6 - Слайд №6

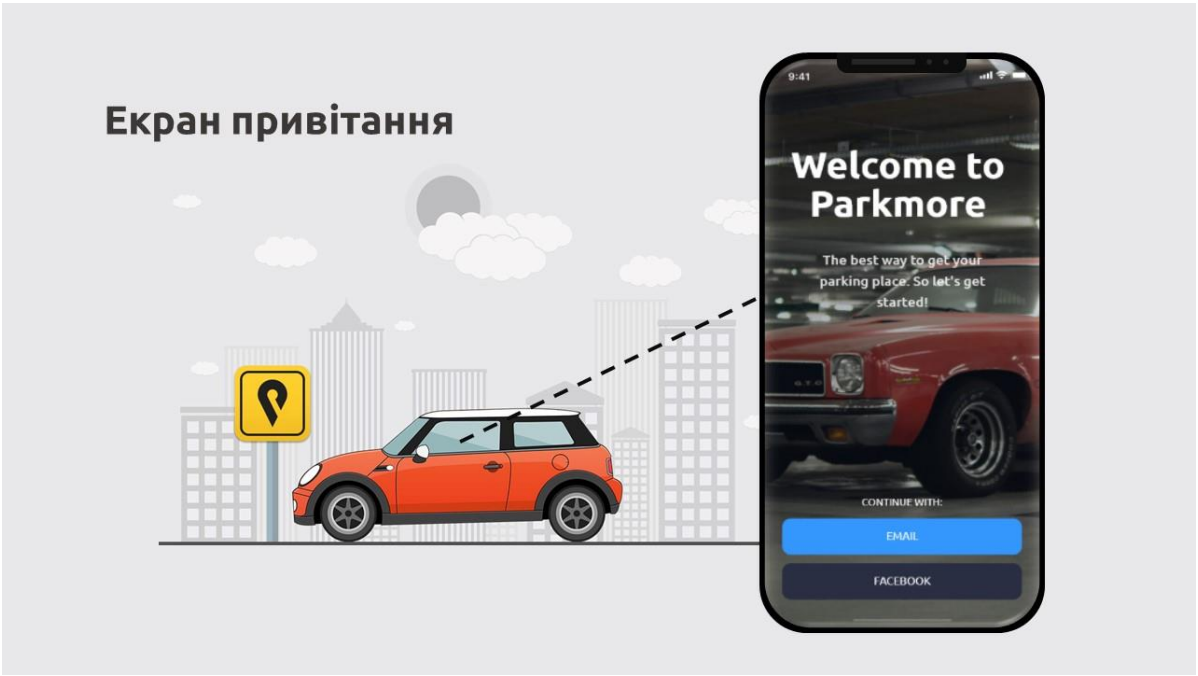

Рисунок А.7 - Слайд №7

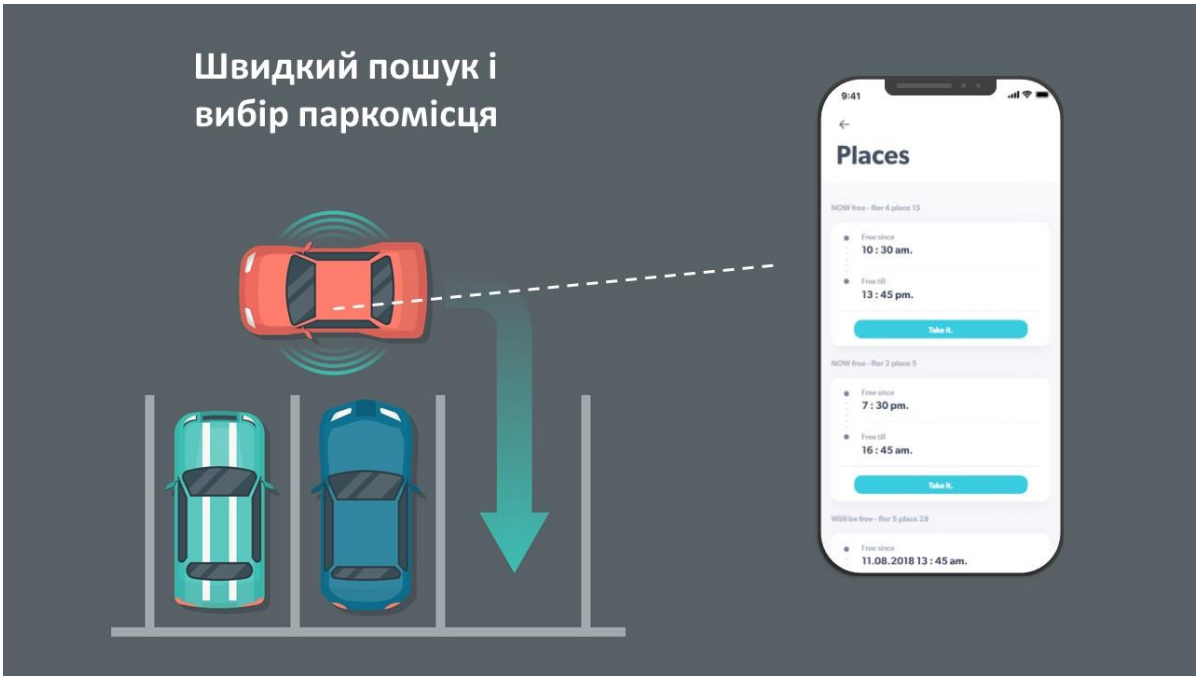

Рисунок А.8 - Слайд №8

Рисунок А.9 - Слайд №9

Рисунок А.10 - Слайд №10

Рисунок А.11 - Слайд №11

Рисунок А.12 - Слайд №12

Висновки:

Розроблено інформаційну систему, яка використовується для моніторингу паркувальних центрів, інформування користувачів про наявність місць та бронювання, управління паркувальним місцем для клієнтів.

Рисунок А.13 - Слайд №13

ДОДАТОК Б

GOLANG СТРУКТУРИ СУТНОСТЕЙ

```
1       type Company struct {
2           ID       int       `gorm:"primary_key" json:"id"`  // record uid
3           CreatedAt time.Time `gorm:"column:created_at;default:current_date;not
4   null" json:"createdAt"` // when record was created
5           Name     string    `json:"name"`
6           Url      string    `json:"url"`
7           Email    string    `json:"email"`  // for mailing
8           Users    []User
9   `gorm:"foreignkey:CompanyID;association_foreignkey:ID" json:"users"`  // list of
10  User
11          Bookings  []Booking
12  `gorm:"foreignkey:CompanyID;association_foreignkey:ID" json:"bookings"`   //
13  list of Booking
14          Garage    Garage
15  `gorm:"foreignkey:CompanyID;association_foreignkey:ID" json:"garage"`     //
16          Parkings  []Parking
17  `gorm:"foreignkey:CompanyID;association_foreignkey:ID" json:"parkings"`   //
18  list of Parking
19          Places    []Place
20  `gorm:"foreignkey:CompanyID;association_foreignkey:ID" json:"places"`     // list
21  of Place}
22      type Garage struct {
23          ID       int       `gorm:"primary_key" json:"id"`  // record uid
24          CreatedAt time.Time `gorm:"column:created_at;default:current_date;not
25  null" json:"createdAt"` // when record was created
26          Name     string    `json:"name"`
27          Parking  []Parking `json:"parking"`
```

```
28              CompanyID  int `json:"companyId"`}
29          type Parking struct {
30              ID        int       `gorm:"primary_key" json:"id"`                                  //
31  record uid
32              CreatedAt time.Time `gorm:"column:created_at;default:current_date;not
33  null" json:"createdAt"` // when record was created
34              Num       int       `json:"num"`                                                    // row
35  number
36              GarageID  int       `json:"garageId"`                                                //
37              Places    []Place   `json:"places"`                                                 //
38              CompanyID int       `json:"companyId"`}
39          type Place struct {
40              ID              int         `gorm:"primary_key" json:"id"`  //  record uid
41              CreatedAt       time.Time
42  `gorm:"column:created_at;default:current_date;not null" json:"createdAt"` //  when
43  record was created
44              Num             int         `json:"num"` //  place serial number
45              HaveAvTime      bool        `gorm:"-" json:"haveAvTime"`  //  have a place
46  available time for parking or not. Field not for a db
47              Booking         []Booking   `json:"booking"` //  related to []Booking{}
48              ParkingID       int         `json:"parkingId"` //  related to Parking{}
49              CompanyID       int         `json:"companyId"`}
50          type User struct {
51              ID        int       `gorm:"primary_key" json:"id"` // record uid
52              CreatedAt time.Time `gorm:"column:created_at;default:current_date;not
53  null" json:"createdAt"` // when record was created
54              FullName  string    `json:"fullName"` // user full name
55              Email     string    `gorm:"unique;not null" json:"email"`  // email
56              Booking   []Booking `json:"booking"` // user's bookings list
57              CompanyID int       `json:"companyId"` // related to Company{}}
```

МІГРАЦІЇ

```
1         package main
2
3         import (
4             "fmt"
5             "gitlab.com/logan83955/gormdb/entity"
6             "github.com/jinzhu/gorm"
7         )
8
9         func main() {
10            dsn := fmt.Sprintf("host=%s port=%s user=%s dbname=%s password=%s
11    sslmode=disable", "localhost", "5432", "user", "dbexample", "123")
12            conn, err := gorm.Open("postgres", dsn)
13            if err != nil {
14             fmt.Println(err)
15            }
16
17            err = conn.AutoMigrate(
18             &entity.User{},
19             &entity.Garage{},
20             &entity.Parking{},
21             &entity.Place{},
22             &entity.Company{},
23            ).Error
24            if err != nil {
25             fmt.Println(err)
26            }
27
28            err = conn.Exec(`
```

```
29              DO
30               $do$
31              BEGIN
32               IF NOT EXISTS(
33                       SELECT 1
34                       FROM information_schema.tables
35                       WHERE table_name = 'bookings'
36                     ) THEN
37               CREATE TABLE bookings
38               (
39                  id SERIAL PRIMARY KEY NOT NULL,
40                  created_at timestamp with time zone DEFAULT CURRENT_DATE
41  NOT NULL,
42                  duration tsrange NOT NULL,
43                  place_num INT,
44                  user_id INT,
45                  place_id INT,
46                 company_id INT,
47               );
48                  END IF; `).Error
49              END
50                $do$;
51
52             if err != nil {
53              fmt.Println(err)
54             }
55          }
```

ДАНІ МІГРАЦІЇ

```
company := entity.Company{
    Name:  "Company 1",
    Email: "company@gmail.com",
}

user := entity.User{
    FullName:  "John Doe",
    Email:     "jd@google.com",
    CompanyID: 1,
}

garage := entity.Garage{
    Name:      "company garage",
    CompanyID: 2,
    Parking: []entity.Parking{
     {
      Num:       1,
      CompanyID: 1,
      Places: []entity.Place{
       {
        Num:       2595,
        CompanyID: 1,
        Booking: []entity.Booking{
         {
          Duration:entity.Duration{
           From: time.Date(2019, 02, 26, 14,0,0,0, time.UTC),
           To: time.Date(2019, 02, 26, 17,0,0,0, time.UTC),
          },
```

```
                UserID: 1,
                PlaceID: 1,
                CompanyID: 1,
              },
            },
          },
        },
      },
    }
```