МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Тернопільський національний економічний університет
Факультет комп'ютерних інформаційних технологій
Кафедра комп'ютерних наук

ВІВЧАР Андрій Віталійович

Web-орієнтована система моніторингу автостоянок/ Web-oriented system for parking lots monitoring

спеціальність: 8.05010302 - Інженерія програмного запезпечення
магістерська програма - Інженерія програмного забезпечення

Магістерська робота

Виконав студент групи ІПЗм-21
А. В. Вівчар

_____

Науковий керівник:
к.т.н., доцент ПУКАС А.В.

_____

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

Завідувач кафедри
_____ **А. В. Пукас**

ТЕРНОПІЛЬ - 2016

Зміст

ВСТУП

Актуальність теми. Web-орієнтована система моніторингу автостоянок – це сучасне рішення, яке дозволить жителям та гостям міста почуватись комфортніше, спокійніше та простіше, забронювавши та доїхавши до місця призначення без лишніх витрат часу. З кожним місяцем чи роком кількість людей, які купують транспортний засіб зростає, кожен повинен десь її зберігати в безпеці. Для цього автоводії користуються автостоянками з цілодобовою охороною, яка слідкує за порядком на автостоянці. Та чим більше автомобілів, тим більше записів потрібно робити, обраховувати кількість транспортних засобів для місячного звіту. Все це є дуже ресурсо-затратно та вимагає багато часу. Для гостей міста є дуже проблематично найти відповідний авто майданчик із вільним місцем, чи хоча б просто найти адресу автостоянки та добратись до неї.

Облік транспортних засобів на автостоянці для одного чи декількох операторів чи охоронців – це доволі складний процес, який характеризується великою трудозатратністю, як по кількості необхідних часових ресурсів, так і вимагає особливої уваги при підрахунках. Найбільша інтенсивність таких процесів в періоди, коли люди повертаються з роботи, у святкові дні, і ставлять свій транспортний засіб на автостоянку в час-пік. Також для гостей міста, котрі в свою чергу хочуть залишити свій транспортний засіб на одній із стоянок міста, є дуже тривалий процес, іноді може бути навіть стресовий.

Тому необхідно розробити ресурс, який міг би вмістити у собі усі ці можливості: облік транспортних засобів, підбір і бронювання місця на стоянці та прокладання маршруту до неї. Програмний продукт повинен бути легким в надавані послуг та повинен бути доступним із будь-яких пристроїв із можливістю виходу в мережу інтернет.

На момент написання магістерської роботи в місті є близько 15 автостоянок, з яких близько 70% веде документацію в паперовому вигляді, і немає жодної автостоянки на якій можна було б забронювати місце наперед. Зважаючи на ці дані, розробка єдиної системи із базою усіх стоянок міста є **актуальною**.

Розроблювана система моніторингу автостоянок повинна споживати незначну частину ресурсів мережі та приладу на якому використовується. Тому для

реалізації системи обрано мову програмування C#, що відзначається значною швидкодією і незначними затратами оперативної пам’яті.

Зв’язок роботи з науковими програмами, планами, темами. Магістерська робота виконувалась згідно з напрямками наукових досліджень кафедри комп’ютерних наук Тернопільського національного економічного університету.

Мета і завдання досліджень. Метою роботи є покращення та полегшення процедури бронювання, пошуку автостоянок, реєстрації користувачів та їх транспортних засобів у системі моніторингу автостоянок, що дасть змогу без проблем дістатись до автостоянки, забронювати відповідне місце під транспортний засіб, швидко опрацьовувати та зберігати дані користувачеві та в подальшому буде мати змогу редагувати потрібні дані.

Для досягнення поставленої мети потрібно розв’язати наступні задачі:

– дослідити та проаналізувати існуюче програмне забезпечення із подібним функціоналом;
– створити специфікацію вимог для програмної системи;
– спроектувати та розробити архітектуру програмної системи;
– реалізувати програмну систему;
– провести оцінку продуктивності і надійності розробленого продукту.

Предмет дослідження: методи використання онлайн карт для системи моніторингу автостоянками та алгоритми пошуку маршрутів.

Методи досліджень. Теоретичною основою магістерського дослідження виступають алгоритми пошуку найкоротшого шляху.

Наукова новизна отриманих результатів полягає у тому, що вперше запропоновано розробку web-орієнтованої системи моніторингу автостоянок міста. При написанні web-орієнтованої системи використовувались модулі популярної системи онлайн карт Google Maps за допомогою їхніх API модулів, немає ні однієї системи аналогічній цій.

Практичне значення отриманих у роботі результатів полягає у можливості безперешкодно та з легкістю вести облік транспортних засобів на автомайданчиках, зберігання, редагування та видалення даних про транспортні

засоби, їх власників, без проблем найти, забронювати та отримати координати на карті потрібної автостоянки.

Особистий внесок здобувача. Основні результати магістерської роботи отримані автором самостійно, на основі власних ідей та розробок.

Апробація результатів дослідження. Основні положення та результати дипломної роботи обговорювалися на науковому семінарі «Advanced Computer Information Technology» (ACIT`2016), Тернопіль, 2016.

Публікації. Результати роботи опубліковані в 1 науковій праці

.

Структура та об'єм роботи. Магістерська робота складається з трьох розділів, загальним обсягом 91 сторінок, де розміщено 2 таблиці, 25 рисунків, 9 додатків та 37 джерел в переліку посилань.

РОЗДІЛ 1 АНАЛІЗ СИСТЕМ МОНІТОРИНГУ ТРАНСПОРТУ НА АВТОСТОЯНКАХ

1.1 Коротка характеристика об’єкту управління

Автостоянка – це спеціальний відкритий майданчик, призначений для зберігання транспортних засобів, в основному легкових автомобілів. Також на автостоянці є відведені місця для мототранспорту та великогабаритного транспорту(автобуси, вантажівок різних типів).

Управління персоналом та автостоянкою здійснює директор. Він безпосередньо розпоряджається коштами в межах кошторису витрат, приймає всі інші рішення по питаннях організації роботи автостоянки.

В’їзд(розміщення), обслуговування та виїзд клієнтів контролюється операторами автостоянки.

Розміщення транспортного засобу відбувається наступним чином – проводиться реєстрація клієнта (П.І.П, адреса, номер тел., №. водійського посвідчення) та реєстрація транспортного засобу у журналі (визначаються габаритні розміри даного транспортного засобу, тип, стан, записується державний реєстраційний номер, марка та модель), перевіряється наявність вільних місць під даний тип та розмір транспортного засобу, присвоюється код місця на автостоянці, та визначається термін зберігання транспортного засобу.

Виїзд транспортного засобу (ТЗ) відбувається наступним чином – власник транспортного засобу до закінчення терміну зберігання повинен явитись та надати код місця, на якому стоїть ТЗ та власні дані.

Пошук автостоянки та бронювання місця на ній здійснюється за допомогою мережі інтернет. Здійснюється реєстрація на сервісі, вводяться дані транспортного засобу, зокрема марка та модель, здійснюється пошук найближчих автостоянок та вільних місць на них за певними критеріями.

У кінці тижня операторами автостоянки формується звіт про загальний дохід та статистика (кількість занятих місць, терміни, марки та моделі ТЗ), після чого звіт відправляється директору автостоянки.

Основним напрямом діяльності автостоянки є зберігання транспортних засобів.

Директор автостоянки виконує такі обов’язки:

– укладання договорів щодо прийому на роботу;
– контроль підлеглих щодо ефективності виконання роботи;
– консультація клієнтів на вимогу;
– звільнення підлеглих;
– реєстрація клієнтів та їх транспортних засобів;
– перевірка стану транспортних засобів;
– забезпечення прав клієнтів відповідно до чинного законодавства;
– несення відповідальності у повній мірі за халатність своїх підлеглих;
– розпоряджання всіма коштами.

Оператор виконує такі обов’язки:

– консультування клієнтів;
– реєстрація клієнтів та їх транспортних засобів;
– перевірка стану транспортних засобів;
– формування тижневого звіту.

Загальну схему організаційної структури підприємства зображено на рисунку 1.1.

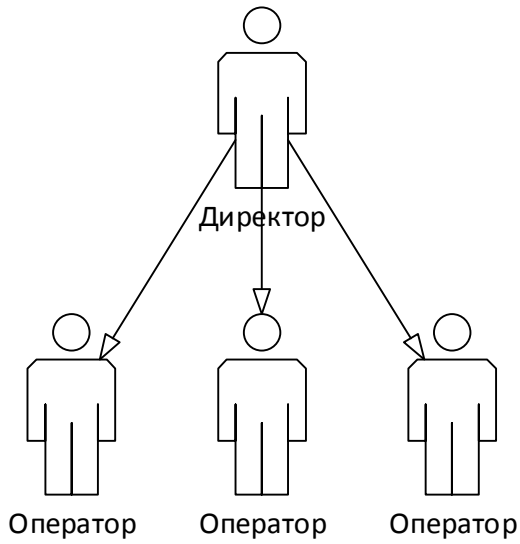


Рисунок 1.1 – Організаційна структура автостоянки

Користувачами системи є: директор, оператор, клієнт.

На рисунку 1.1 видно, що директор автостоянки є найвищою ланкою в його організаційні структурі. Основним обов'язком директора є забезпечення максимальної ефективності роботи автостоянки. Процес реєстрації транспортних засобів та реєстрації клієнтів у системі здійснюють оператори. Оператор присвоює код місця транспортному засобу, на якому останній буде стояти(зберігатись). Оператор ознайомлює клієнта із схемою автостоянки в разі екстреної ситуації для евакуації з території автостоянки, здійснює реєстрацію клієнта та транспортного засобу в системі та видає перепустку клієнту з вказаним місцем на стоянці. Клієнт в свою чергу оплачує послугу зберігання та ставить свій транспортний засіб на місце, яке йому присвоєно. У кінці кожного місяця формується звіт, який включає інформацію про зареєстрованих клієнтів та про реєстрацію транспортних засобів і відправляється директору.

1.2 Опис предметної області

Проаналізувавши предметну область, виділено чотири основні бізнес процеси, що забезпечують роботу системи:

- формування анкети клієнта (реєстрація клієнта);
- реєстрація транспортного засобу;
- формування та відображення звіту по виконаній реєстрації;
- пошук та бронювання місця на автостоянці.

Реєстрація клієнта включає в себе контактні дані клієнта і збереження їх на сервері (рис. 1.2).

Рисунок 1.2 – Функції бізнес процесу «Реєстрація клієнта»

Характеристика цього бізнес процесу наведена в табл. 1.1.

Таблиця 1.1

Характеристика бізнес процесу «Формування анкети»

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

Реєстрація транспортного засобу включає в себе дані транспортного засобу(власник, державний реєстраційний номер, VID, марка та модель, габаритні розміри, термін зберігання) (рис. 1.3).

Рисунок 1.3 – Функції бізнес процесу «Реєстрація транспортного засобу»

Характеристика цього бізнес процесу представлена в табл. 1.2.

Таблиця 1.2

Характеристика бізнес процесу «Реєстрація транспортного засобу»

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Реєстрація транспортного засобу
Основні учасники	Сервер, браузер, клієнт

Продовження таблиці 1.2

Вхідна подія	Відправлення даних про транспортний засіб
Вхідні документи	Немає
Вихідна подія	Відповідь про статус реєстрації, присвоєння реєстраційного коду транспортному засобу
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

Бронювання місця на автостоянці включає в себе дані про наявні місця на автостоянці, дані транспортного засобу (власник, державний реєстраційний номер, VID, марка та модель, габаритні розміри, термін зберігання) (рис. 1.4).

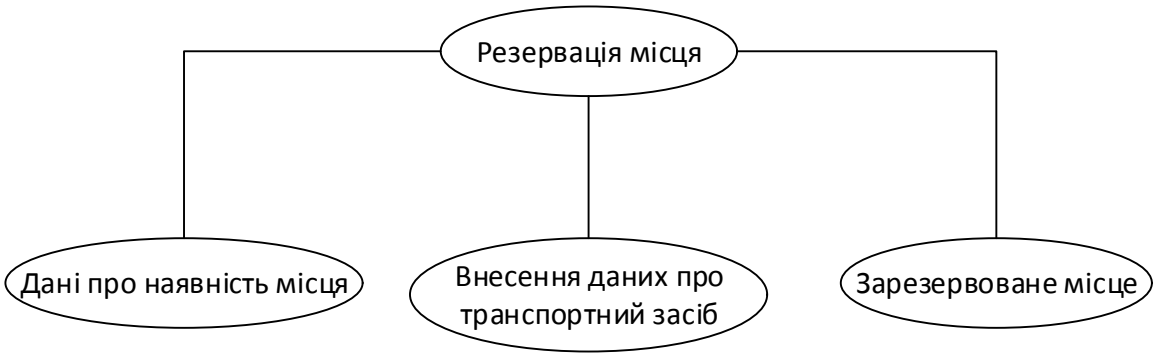


Рисунок 1.4 – Функції бізнес процесу «Бронювання місця на автостоянці»

Характеристика цього бізнес процесу представлена в табл. 1.3.

Таблиця 1.3

Характеристика бізнес процесу «Бронювання місця на автостоянці»

Назва характеристики	Значення характеристики
Ім’я бізнес процесу	Бронювання місця на автостоянці
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про клієнта та транспортний засіб

Продовження таблиці 1.3

Вхідні документи	Немає
Вихідна подія	Відповідь про статус бронювання, присвоєння коду парко - місця транспортному засобу
Вихідні документи	Немає
Клієнт бізнес процесу	Сервер

1.3 Аналіз відомих програмних засобів моніторингу автостоянок\

Із стрімким розвитком інтернет технологій, участі комп'ютерів у практично будь-якому аспекті життя людини та дедалі більшому поширенню можливості вільно та з користю витрачати свій час, із збільшенням кількості транспортних засобів - проблема моніторингу автостоянок стає дедалі актуальнішою.

Наприклад у 2001р. середня кількість автомобілів на 1000 населення, складала близько 50 авто. На сьогоднішній день, кількість авто на 1000 населення перевищує 200 автомобілів. Таким чином є необхідність створення нових парко-місць і відкриття цілих нових автостоянок. При такій кількості автомобілів і автостоянок, є надзвичайно доцільно використовувати єдину систему для обліку, чи пошуку чи бронювання місць заздалегідь.

На сьогоднішній день, ринок подібних систем є дуже обмежений. Немає єдиної цілісної системи, де можна було б отримувати дані про всі автостоянки міста, чи хоча б про певну автостоянку і заздалегідь забронювати місце на ній.

Розглянемо існуючі системи автостоянок, та систем пошуку їх:

- "Simple Parking";
- "Vector AP 2000";
- "AllStojanka".
- Автостоянка, парковка в Борисполе (http://bac.in.ua)
- Яндекс.Парковки (https://play.google.com/store/apps/details?id=ru.yandex.parking)
- Автостоянка 2.5.5 (http://soft.oszone.net/program/6383/Avtostojanka/)

Ці системи розроблені для автоматизації процесу оформлення та реєстрації транспортних засобів на автостоянці.

Розглянемо детальніше систему «Simple Parking». Загальний вигляд системи зображено на рисунку 1.5.

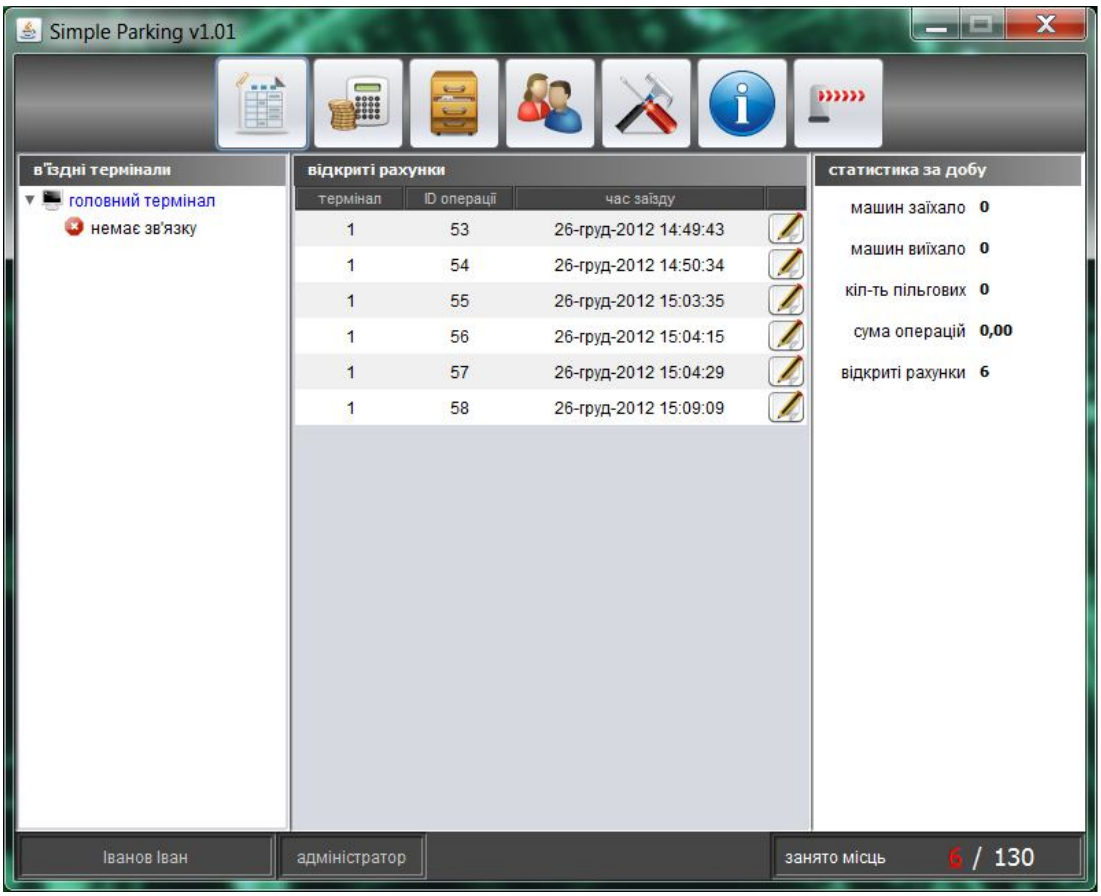


Рисунок 1.5 – Загальний вигляд системи «Simple Parking»

Ця система має зручний користувацький інтерфейс, що дозволяє інтуїтивно керувати нею. Можливий перегляд схеми автостоянки, наочне відображення схеми автостоянки з поділом на цікаві і вільні місця, швидке оформлення операції постановки і зняття транспортного засобу з стоянкового місця, показати на схемі розташування конкретного транспортного засобу. Вона призначена для встановлення на операційних системах типу Windows XP, Vista і новіших.

Система «Vector AP 2000»:

Загальний вигляд системи зображений на рисунку 1.6.

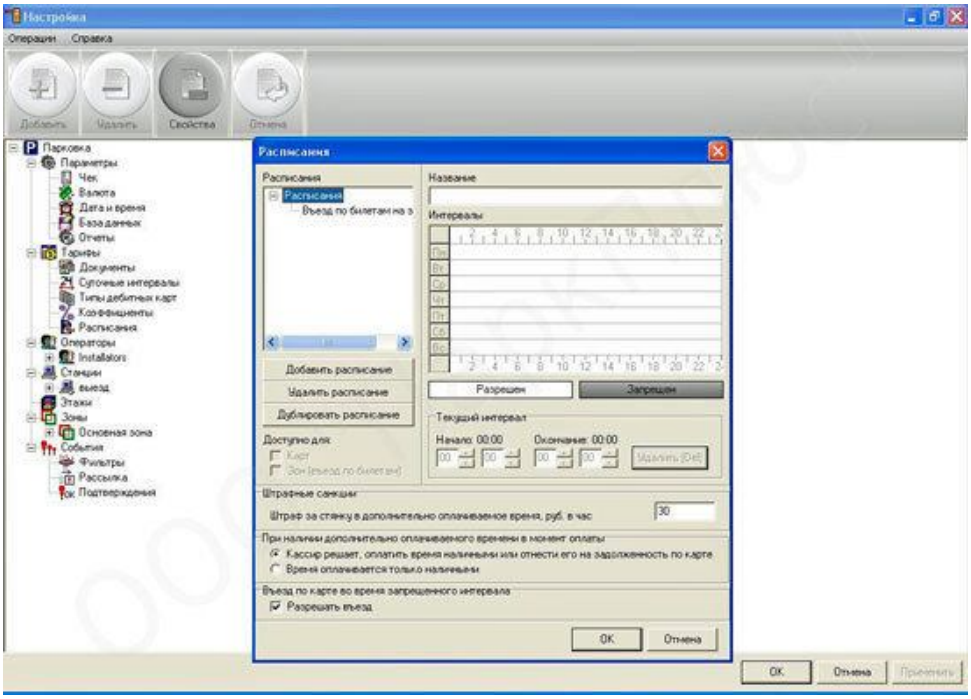


Рисунок 1.6 – Загальний вигляд системи «Vector AP 2000»

Основні можливості програмного забезпечення системи Vector AP 2000:

– задання різних параметрів за часом, виду валюти, кількості місць, що використовуються на паркування, фактично використовуваних і зайнятих місць на даний момент і т.д;

– управління системи автоматичного паркування та задання різних налаштувань з сервера системи;

– задання кожному оператору логін і пароль для роботи з системою, а також різні дозволи для тих чи інших дій;

– можливість отримати і архівувати звітність за будь-який період діяльності.

Система «AllStojanka»:

На рисунку 1.7 зображено загальний вигляд системи «AllStojanka».

Програма AllStojanka (Рис.1.7) дозволяє:

– вести облік в електронному вигляді в базі даних з простим і зрозумілим інтерфейсом;

– отримувати достовірну та оперативну інформацію по взаєморозрахунках;

– наочно відобразити схему автостоянки з поділом на цікаві і вільні місця;

– показати на схемі розташування конкретного транспортного засобу;

– швидко оформляти операцію постановки і зняття транспортного засобу з стоянкового місця;

– приймати і оформляти отримання грошових коштів з печаткою товарного чека;

– автоматично розраховувати вартість послуг по стоянці використовуючи різні варіанти і тарифи;

– вести облік знижок;

– автоматично розраховувати і нараховувати пеню за прострочення;

– контролювати дні і час прострочення.

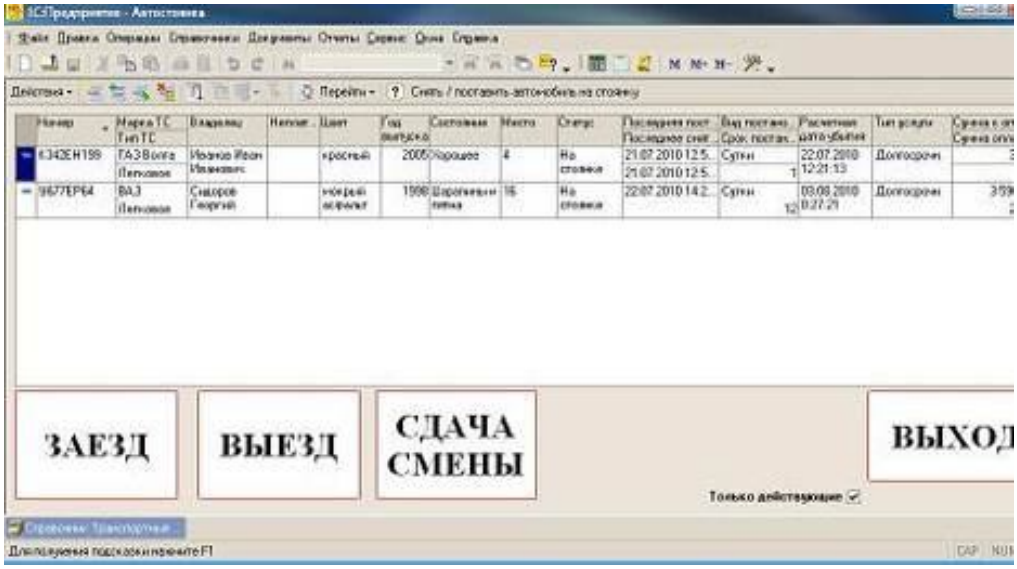


Рисунок 1.7 – Загальний вигляд системи «AllStojanka»

Онлайн система «Автостоянка, парковка в Борисполе».

На рисунках 1.8-1.10 зображений інтернет сервіс «Автостоянка, парковка в Борисполе».

Онлайн система «Автостоянка, парковка в Борисполе» має доволі простий і зрозумілий інтерфейс. Сайт простий, не містить ніяких особливих функцій, в загальному містить інформаційний характер про послуги на автостоянці поблизу аеропорту «Бориспіль». На вкладці «Схемы проезда» (рис. 1.8-1.9) зображено три

варіанти шляху до трьох різних автостоянок. Використовуються позначення автостоянки по Google картах, також є вказані GPS-координати, адреса та контактні телефони.

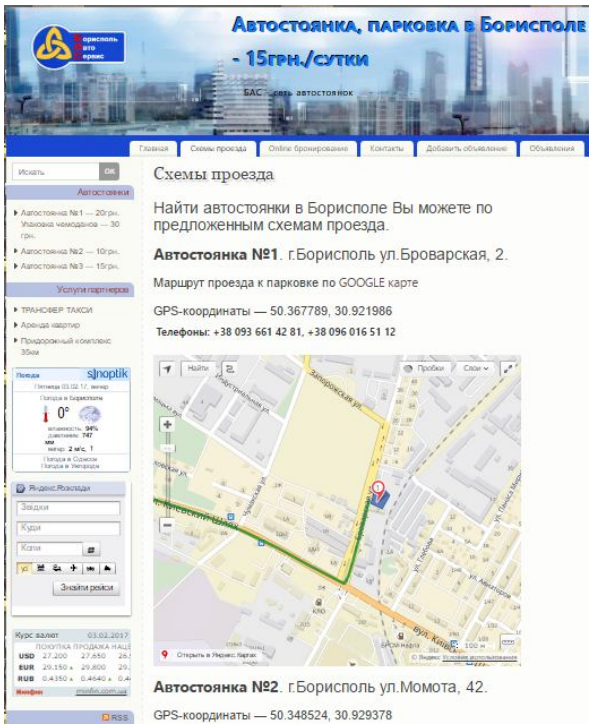


Рисунок 1.8 – Загальний вигляд системи «Автостоянка, парковка в Борисполе», сторінка схеми проїзду

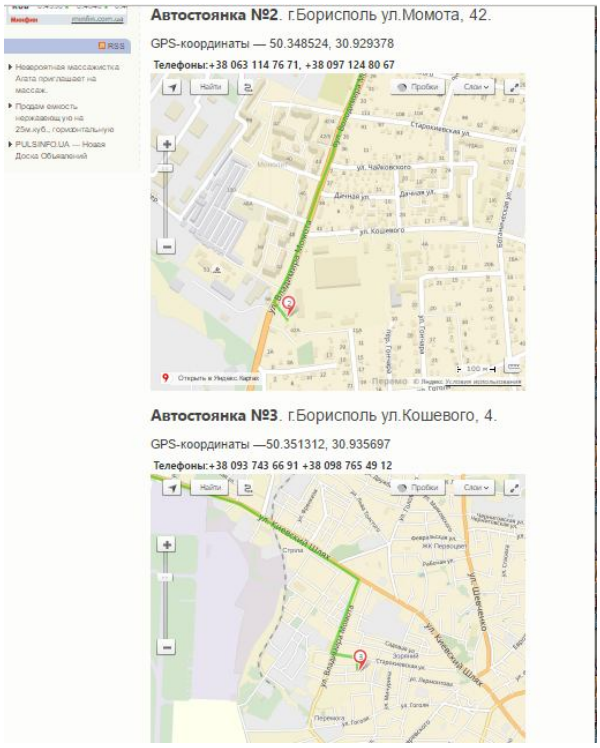


Рисунок 1.9 – Загальний вигляд системи «Автостоянка, парковка в Борисполе», сторінка схеми проїзду

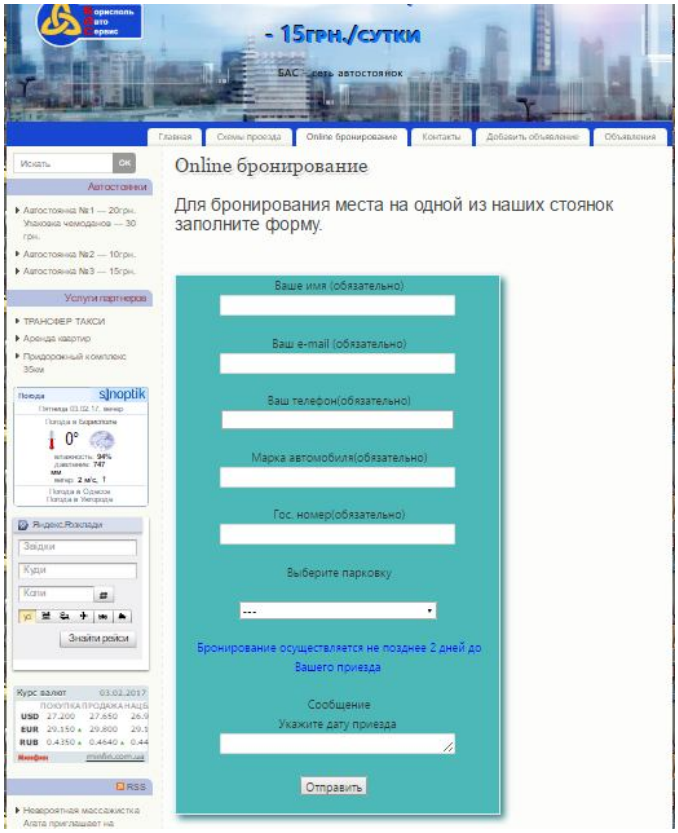


Рисунок 1.10 – Загальний вигляд системи «Автостоянка, парковка в Борисполе», сторінка онлайн бронювання

На наступній вкладці «Online бронирование» (рис. 1.10) зображено форму вводу даних власника авто, та дані автомобіля для бронювання місця на автостоянці. Наступні вкладки це контактні даті усіх автостоянок та адміністрації, та інші послуги які надає вже власне сама автостоянка.

Програма «AvtoStojanka»

На рисунку 1.11 зображено загальний вигляд системи «AvtoStojanka».

На рис. 1.11 зображено загальний вигляд системи автостоянки «AvtoStojanka». Вона має простий та зручний інтерфейс, все міститься на одній сторінці. Для бронювання потрібно ввести усі дані транспортного засобу, такі як державний реєстраційний номер, марка, колір та примітка. Далі вводяться дані автовласника. Після того дані місця на автостоянці та термін зберігання авто. Всі зареєстровані авто які містяться на автостоянці відображаються у таблиці

нижче. Дана система не містить можливості бронювання, вона призначена виключно для користування в адміністрації автостоянки.

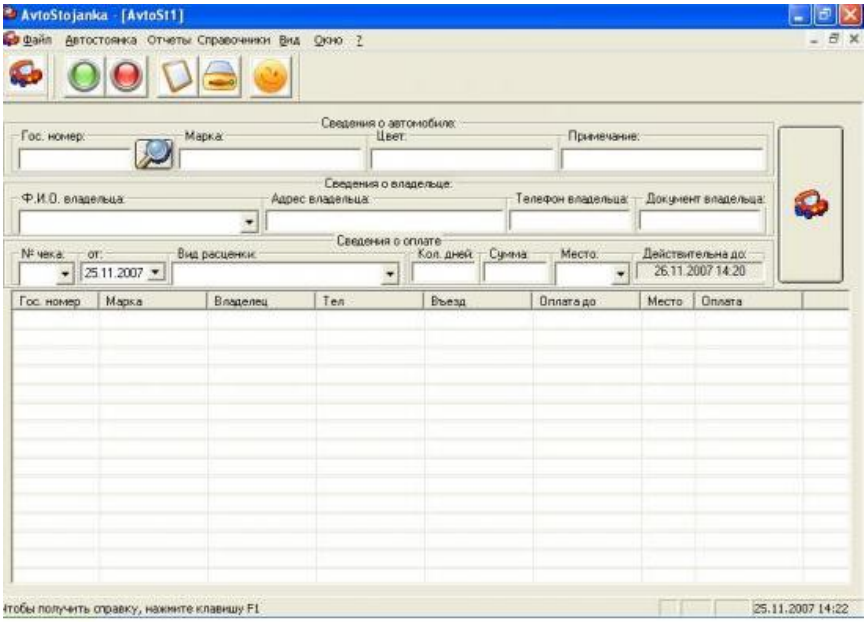


Рисунок 1.11 – Загальний вигляд системи «AvtoStojanka»

Мобільна система «Яндекс.Парковки».

На рисунках 1.12-16. зображено мобільну систему «Яндекс.Парковки».

Додаток на мобільний телефон дуже зручний у використанні. Має не складний інтерфейс та достатньо широкий спектр функцій. Він має можливість знаходити найближчу автостоянку, шукати в своїй базі даних вільне місце на ній.

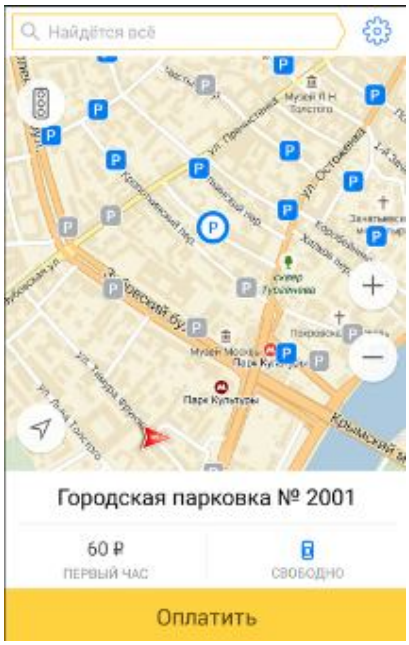


Рисунок 1.12 – Пошук вільних місць системи «Яндекс.Парковки

Також додаток містить інформацію про кожну стоянку в місці.

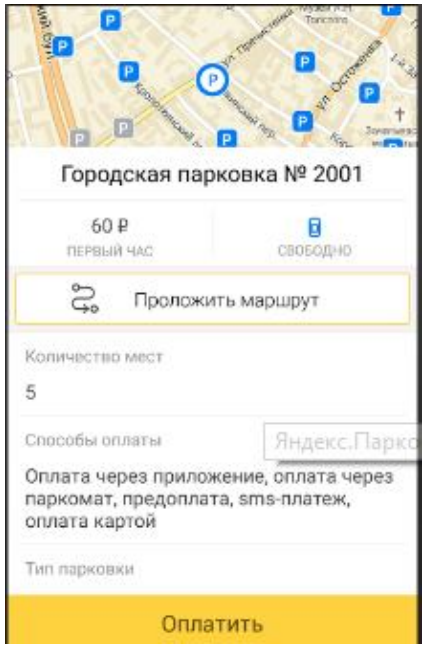


Рисунок 1.13 – Інформація про парковку системи «Яндекс.Парковки»

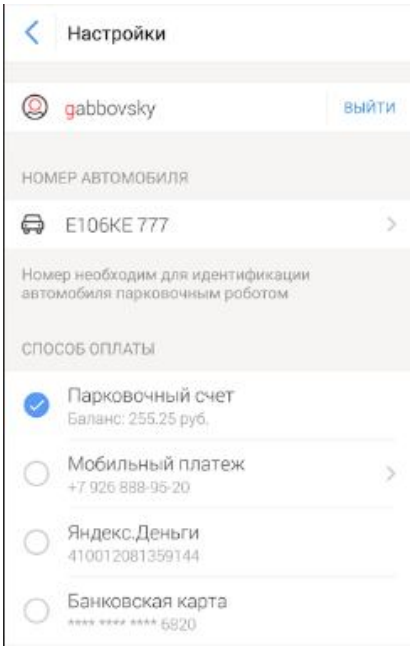


Рисунок 1.14 – Різні способи оплати парковки в системі «Яндекс.Парковки»

Також для більшої зручності є функція оплати автостоянки за умови що автостоянка приймає платежі онлайн. Коли власник, поставив свій транспортний засіб на автостоянку, додаток може засікти час, скільки транспортний засіб ще може залишатись на стоянці. Також якщо він відійшов від автостоянки десь по справах чи просто прогулюючись містом, додаток завжди може показати в якому саме напрямку та де зберігається його авто.

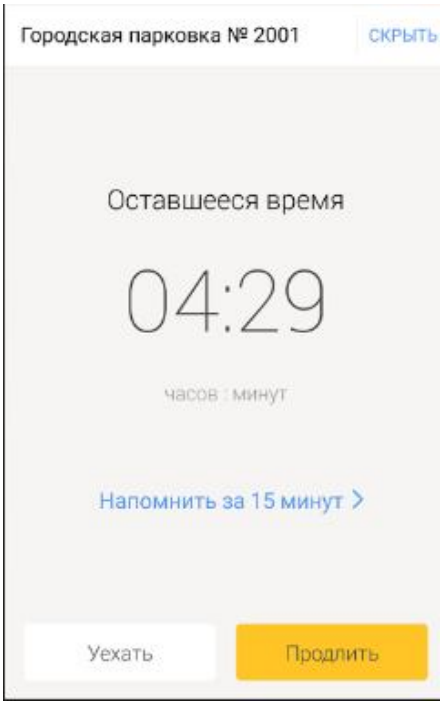


Рисунок 1.15 – Відлік часу в системі «Яндекс.Парковки»

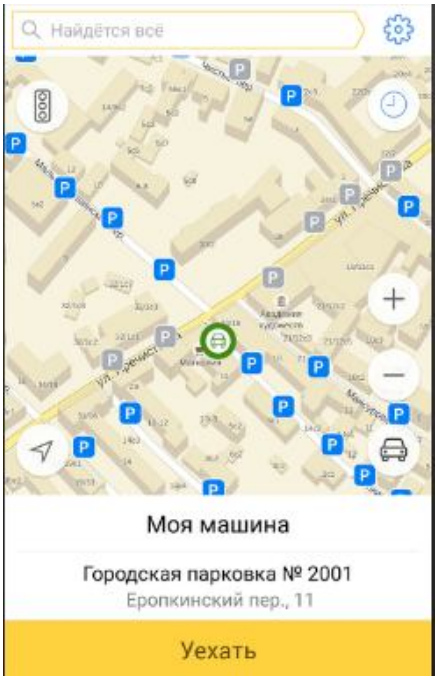


Рисунок 1.16 – Пошук автомобіля на карті в системі «Яндекс.Парковки»

У таблиці 1.4 приведено порівняльну характеристику цих програмних продуктів.

Таблиця 1.4

Порівняльна характеристика програмних продуктів

Назва програмного продукту	Версії продукту	Функціональність	Інтерфейс користувача
Simple Parking	1.01	- Облік - Відображення схеми автостоянки - Відображення конкретного транспортного засобу - Перевірка клієнтів - Реєстрація клієнтів - Контроль прострочення	Desktop-система
Vector AP 2000	Не вказано	- Відображення конкретного транспортного засобу - Приймання та оформлення коштів - Облік знижок - Реєстрація клієнта - Реєстрація транспорту - Відображення вільних місць	Desktop-система
AllStojanka	Не вказано	- Облік - Відображення схеми автостоянки - Відображення конкретного транспортного засобу - Приймання та оформлення коштів - Розраховувати вартість послуг - Облік знижок - Контроль прострочення - Нарахування пені - Реєстрація клієнта	Desktop-система

Продовження таблиці 1.4

Автостоянка, парковка в Борисполе	Не вказано	- Відображення схеми доїзду - Відображення координат - Онлайн бронювання. - Інформація про інші послуги	Web-система
AvtoStojanka	2.5.5	- Облік - Відображення конкретного транспортного засобу - Контроль прострочення	Desktop-система
Яндекс. Парковки	1.5	- Відображення схеми доїзду - Онлайн бронювання. - Відображення конкретного транспортного засобу - Приймання та оформлення коштів - Розраховувати вартість послуг - Контроль прострочення - Відлік часу	Mobile-система

Розглянувши системи-аналоги, які зазначено вище, можна виділити їх основні переваги та недоліки. Система Simple Parking має зручний, приємний на перший погляд та зрозумілий користувацький інтерфейс. Система є desktop, що не дозволить клієнтам подивитись на вільні місця на автостоянці онлайн. У системі Vector AP 2000 – це desktop система, яка на відміну від попередньої має функцію відображення вільних місць на стоянці, також суттєвою перевагою є функція оплати послуг. Недоліком такої системи є складний користувацький інтерфейс, який може бути не зразу зрозумілим новим користувачам системи. Система AllStojanka та AvtoStojanka також як і попередні є desktop системами, на відміну від попередніх систем в цих системах є набагато більше додаткових функцій (нарахування пені, облік знижок, розрахувати вартість послуг). Крім того тут простий на перший погляд користувацький інтерфейс без красивого та яскравого

дизайну. Система Яндекс.Парковки є мобільним додатком, який зручно використовувати в подорожах та повсякденному житті. Має приємний зручний дизайн, через нею можна здійснювати оплату та бронювання місця на автостоянці та вона лише мобільний додаток на Android системи, який неможливо використовувати на ПК. Web-система Автостоянка, парковка в Борисполе, містить лише більш інформаційних характер із інформацією як доїхати до автостоянки та контактними даними. Ця система має лише одну функцію бронювання місця на автостоянці.

Під час виконання магістерської роботи буде враховано усі переваги і недоліки розглянутих систем і розроблено web-систему моніторингу автостоянок із можливістю пошуку найближчої автостоянки, складання маршруту до неї та бронювання будь-якого місця на ній.

1.4 Теоретична основа алгоритму пошуку найкоротшої відстані на карті міста.

Дуже важливим моментом в функціональності пошуку найближчої автостоянки є її правильність. Не кожен подій захоче їздити по колу слідуючи неправильному напрямку в пошуках автостоянки. Напрям руху, маршрут та відстань повинні бути точні та правильні, щоб водій міг з легкістю цим скористатись та дістатись до місця призначення.

Для розробки алгоритму пошуку найкоротшої відстані брались за основу алгоритми пошуку шляху за допомогою графів Алгоритм Дейкстры, Алгоритм A*, Двонаправлений A*.

Для прикладу Алгоритм **A*** — алгоритм пошуку, який знаходить у зваженому графі маршрут найменшої вартості від початкової вершини до обраної кінцевої. У процесі роботи алгоритму для вершин розраховується функція

$$f(v) = g(v) + h(v) \tag{1.1}$$

де $g(v)$ - найменша вартість шляху в v з стартовою вершини,

$h(v)$ - Евристичне наближення вартості шляху від v до кінцевої мети.

Фактично, функція $f(v)$ довжина шляху до мети, яка складається з пройденої відстані $g(v)$ і тої яка залишилась $h(v)$. Виходячи з цього, чим менше значення $f(v)$, тим раніше ми відкриємо вершину v, так як через неї ми імовірно досягнемо відстань до цілі швидше за все. Відкриті алгоритмом вершини можна зберігати в черзі з пріоритетом за значенням $f(v)$. A * діє подібно до алгоритму Дейкстри і переглядає серед всіх маршрутів які ведуть до мети спочатку ті, які завдяки наявній інформації (евристична функція) в даний момент є найкращими.

Щоб A * був оптимальний, обрана функція $h(v)$ повинна бути допустимої евристичної функцією.

Допустима оцінка є оптимістичною, тому що вона передбачає, що вартість рішення менше, ніж воно є насправді.

Друге, більш сильне умова - функція $h(v)$ повинна бути монотонної.

Евристична функція $h(v)$ називається монотонної (або спадкового), якщо для будь-якої вершини v_1 і її нащадка v_2 різниця $h(v_1)$ і $h(v_2)$ не перевищує фактичної ваги ребра с $c(v_1, v_2)$ від v_1 до v_2, а евристична оцінка цільового стану дорівнює нулю.

Розглянемо також двонаправлений A*. Для двобічної версії алгоритму нам потрібні дві потенційні функції:

$p_f(v)$, оцінює $d(v, t)$ (1.2)

$p_r(v)$, оцінює $d(s, v)$

У цьому випадку з'являється додаткова проблема: різні потенційні вартості у ребер для різних обходів:

$$\ell_{p_f}(v, w) = \ell(v, w) - p_f(v) + p_f(w) \quad (1.3)$$

- якщо ребро обробляється в обході, розпочатому в s,

$$\ell_{p_r}(v, w) = \ell(v, w) - p_r(w) + p_r(v) \quad (1.4)$$

- якщо ребро обробляється в обході, розпочатому в t

Щоб уникнути цієї проблеми, необхідно, щоб

$$\ell_{p_f}(v, w) = \ell_{p_r}(v, w) \Leftrightarrow p_f(v) + p_r(v) = p_f(w) + p_r(w) = const. \quad (1.5)$$

Крім того, функції повинні бути однотонними.

Рішення - використовувати усереднені потенційні функції:

$$h_f(v) = \frac{p_f(v) - p_r(v)}{2} \quad (1.6)$$

$$h_r(v) = \frac{p_r(v) - p_f(v)}{2} = -h_f(v) \quad (1.7)$$

При такому виборі потенційних функцій, виконується

$$\forall u : h_f(u) + h_r(u) = 0 \quad (1.8)$$

Одним із рішень для цієї задачі може бути покращення вже існуючих алгоритмів. Візьмемо для основи алгоритм пошуку шляху A* для вдосконалення. Суть алгоритму буде полягати в перестрибуванні багатьох місць, які повинні бути переглянуті. На відміну від подібних алгоритмів він не вимагає попередньої обробки і додаткових витрат пам'яті.

Алгоритм працює на неорієнтованому графі єдиної вартості. Кожне поле карти має <= 8 сусідів, які можуть бути прохідні або ж ні. Кожен крок у напрямку (по вертикалі або по горизонталі) має вартість 1; крок по діагоналі має вартість √2. Рухи через перешкоди заборонені. Позначення відноситься до одного з восьми напрямків руху (вгору, вниз, вліво і т.д.).

Запис y = x + kd означає, що точка y може бути досягнута через k кроків з x в напрямку d. Коли d - рух по діагоналі, переміщення ділиться на два переміщення по прямій d1 і d2.

Шлях p = (n0, n1, ..., nk) - впорядковане переміщення по точках без циклів з точки n0 до точки nk.

Позначення p \ x означає, що точка x не зустрічається на шляху p.

Позначення len (p) означає довжину або вартість шляху p.

Позначення dist (x, y) означає довжину або вартість шляху між точками x і y.

"Стрибкові точки" дозволяють прискорити алгоритм пошуку шляху, розглядаючи тільки "необхідні" точки. Такі точки можуть бути описані двома простими правилами вибору сусідів при рекурсивному пошуку: одне правило для прямолінійного руху і інше - для діагонального. В обох випадках необхідно довести, що виключаючи з набору найближчих сусідів навколо точки, знайдеться оптимальний шлях з батьківської поточної точки до кожного з сусідів, і цей шлях не буде містити в собі відвідану точку. Розглянемо випадок, який відображає основну ідею:

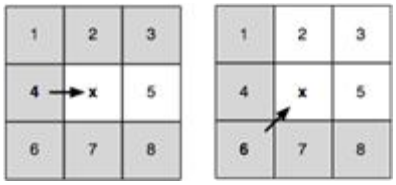


Рисунок 1.17 – Зображення алгоритму пошуку методом перестрибування

X - поточна розглянута точка. Стрілка вказує напрямок руху. І там і там можна відразу відсікти сусідів, виділених сірим, тому що туди можна потрапити по оптимальному шляху з p(x), ніколи не проходячи через x.

Будемо посилатися на безліч точок, які залишаються після відсікання справжніх сусідів поточної точки. Вони відзначені білими на малюнку. В ідеалі, ми хочемо враховувати тільки справжніх сусідів під час перегляду. Проте, в деяких випадках, наявність перешкод може означати, що ми повинні також розглянути невеликий набір до K додаткових точок ($0 \leq K \leq 2$). Ми говоримо, що це точки вимушених сусідів поточної позиції.

ВИСНОВКИ ДО І РОЗДІЛУ

Існуючі програмні продукти для моніторингу автостоянок не проводять повний моніторинг автостоянок, а покривають лише деякий конкретний функціонал, що свідчить про те що немає єдиної цілісної системи моніторингу автостоянок міста, тому в магістерській роботі поставлено за мету створення продукту який би міг включати усі необхідні функції і автоматизувати процес реєстрації та пошуку автостоянок.

Проаналізувавши алгоритми для пошуку найкоротшого шляху, визначено що для розроблюваної системи, яка має інтеграцію з картами, доцільніше використати вже готову системи карт Google Maps API та її алгоритми.

РОЗДІЛ 2
РОЗРОБКА АРХІТЕКТУРИ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ МОНІТОРИНГУ АВТОСТОЯНОК

2.1 Розробка загальної структури системи

«Web-орієнтованої системи моніторингу автостоянок» - це є клієнт-серверна програма для моніторингу автостоянок, вільних місць на них, відображення автостоянок на карті.

Потрібно розробити систему моніторингу автостоянок, яка повинна складатись із адміністративної панелі та панелі користувачів. Адміністративна панель повинна містити усі функції за допомогою яких можна здійснювати керування системою, тобто керування користувачами, стоянками, транспортними засобами, здійснювати реєстрацію користувачів та транспортних засобів власноруч адміністратором. Панель користувача повинна складатись із реєстрації, реєстрації транспортного засобу, пошуку автостоянок, перегляду інформації про них, бронювання місця на автостоянці. Сама ж система повинна містити в собі дані про всі автостоянки міста, їх схеми, розміщення, вільні місця, також повинна давати можливість водіям реєструватись, шукати автостоянки, зберігати дані про транспортний засіб, його власника та інформацію про місце, на якому зберігається цей засіб. Система повинна зберігати дані про транспортний засіб: марку, модель, рік виготовлення, колір, тип, VID, державний реєстраційний номер; дані про власника: ім’я, прізвище, номер телефону, стать, місто проживання. Також система повинна зберігати дані про саму автостоянку (назву, адресу), про місця на автостоянці (код місця, тип, розміри, якій автостоянці належить), зберігати дані про час початку та закінчення терміну зберігання транспортного засобу.

Етап розроблення архітектури програмної системи є дуже важливим, з огляду на те, що на цьому етапі визначаються принципи роботи системи, інтерфейси користувача, всі компоненти системи та зв’язки між ними.

Як було визначено в специфікації вимог, у даній проблемі виділено чотири основні бізнес процеси: це формування анкети клієнта (реєстрація клієнта), реєстрація транспортного засобу, формування та відображення звіту по виконаній

реєстрації клієнтів та транспортних засобів, пошук та бронювання місця на автостоянці.

Проаналізувавши ці бізнес процеси, очевидно, що для того щоб вирішити дану проблему, а саме: здійснювати моніторинг транспортних засобів та користувачів достатньо розробити бізнес логіку, яка б відображала інформацію про транспортні засоби та користувачів у браузері мобільного телефону чи комп’ютера.

Архітектура клієнт-сервер – обчислювальна або мережева архітектура, в якій завдання або мережеве навантаження розподілені між постачальниками послуг, яких називають серверами, і замовниками послуг, яких називають клієнтами. Фізично клієнт і сервер – це програмне забезпечення. Зазвичай вони взаємодіють через комп'ютерну мережу за допомогою мережевих протоколів і знаходяться на різних обчислювальних машинах, але можуть виконуватися також і на одній машині.

Також це архітектура яка визначає загальні принципи організації взаємодії в мережі, де є сервери, вузли-постачальники деяких специфічних функцій (сервісів) і клієнти, споживачі цих функцій. Вона передбачає такі основні компоненти:

– набір серверів, які надають інформацію або інші послуги програмам, які звертаються до них;

– набір клієнтів, які використовують сервіси, що надаються серверами;

– мережа, яка забезпечує взаємодію між клієнтами та серверами.

Архітектура системи зображена на рисунку 2.1.

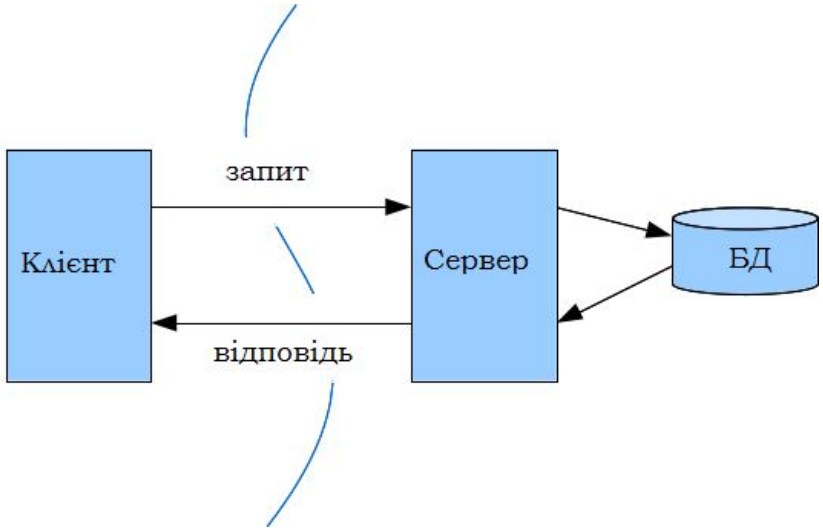


Рисунок 2.1 – Архітектура клієнт-серверної системи

Кожен сервер є незалежним від інших. Усі клієнти є незалежними один від одного і функціонують паралельно. Клієнти і сервери не мають жорсткої прив'язки один до одного. З одного боку є ситуація, коли лише один сервер обробляє запити одразу від кількох різних клієнтів; з іншого боку, в одного клієнта відбувається звернення то до одного сервера, то до іншого. Усі Клієнти повинні мати інформацію про доступні сервери, але можуть не мати ніякого уявлення про інших існуючих клієнтів.

Для проектування прикладної та системної частин застосунку для моніторингу автостоянок скористаємося об'єктно-орієнтованим підходом на базі універсальної мови моделювання UML.

UML – це графічна мова для візуалізації, опису параметрів, конструювання і документування різноманітних систем. Діаграми створюються за допомогою спеціальних CASE засобів. За допомогою технології UML будується єдина інформаційна модель. Серед основних типів діаграм для візуалізації моделі присутні наступні: діаграма варіантів використання (use case diagram), діаграма класів (class diagram), діаграма станів (statechart diagram), діаграма послідовностей (sequence diagram), діаграма компонентів (component diagram).

Ця методика складається з наступних етапів:

1. Написання специфікації системи на природній мові.

2. Опис системи однією із найзагальніших діаграм мови UML, а саме діаграмою способів використання системи (Use-case).

3. Формулювання письмового опису кожного способу використання з супровідними сценаріями.

4. Пошук об'єктів, атрибутів та методів.

На першому етапі проектування необхідно написати специфікацію розроблюваної системи. Згідно з специфікація повинна містити:

- загальний опис проблеми,
- згадку про інші системи, з якими взаємодіятиме система, що проектується,

- згадку про пристрої, якими користується система в процесі свого функціонування,
- події, які потрібно протоколювати,
- ролі користувачів, що взаємодіють з системою,
- інформацію про геометричне або географічне розташування об’єктів, що відносяться до системи.

Web-орієнтована система, яку ми називатимемо TeParking, призначена для моніторингу автостоянок міста, відображення статусу автостоянок, вільних місць на них, контактних даних автостоянок, зберігання інформації про реєстровані транспортні засоби та інформацію про користувачів.

Web-орієнтована система потрібно розробити для перегляду її у будь якому браузері.

Система повинна мати графічний інтерфейс користувача (GUI), що є простим та зручним у користуванні. Усі настройки користувача, дані автостоянок, необхідні для роботи застосунку, будуть зберігатись базі даних MS SQL.

Користувач повинен мати можливість:

- реєструватись у системі,
- реєструвати транспортні засоби у системі,
- керувати профілем користувача,
- шукати найближчу автостоянку та показувати її на карті
- показувати відстань та шлях до неї
- бронювати місце на автостоянці із заданням дати заїзду та дати виїзду.

Адміністратор повинен мати можливість:

- реєструвати у системі нових користувачів,
- реєструвати транспортні засоби у системі,
- керувати профілями користувачів,
- переглядати історію активності на автостоянках
- бронювати місце на автостоянці із заданням дати заїзду та дати виїзду для користувачів.

Кожен варіант використання являє собою окрему функцію програмного продукту, яка корисна кінцевому користувачеві.

Проаналізувавши всі вимоги до системи, виділено наступні варіанти використання. Вони зображені в вигляді діаграми варіантів використання (рис. 2.2). Діаграма варіантів використання дозволяє швидко побачити основні функції, які буде виконувати система.

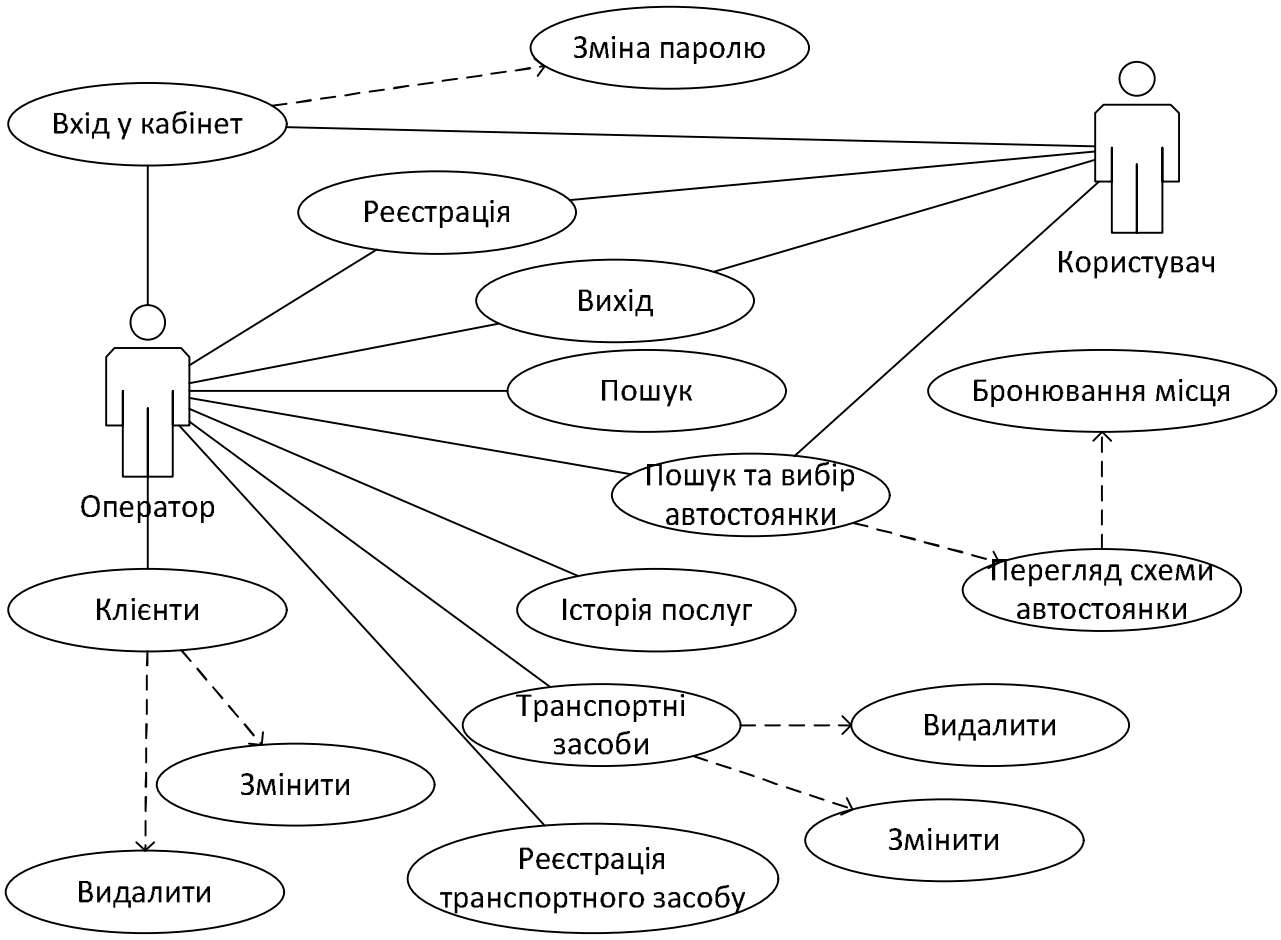


Рисунок 2.2 – Діаграма варіантів використання

Проведемо детальний опис варіантів використання, що зображені на діаграмі варіантів використання (рис. 2.2). Кожен з них поданий у вигляді таблиці, що описує найважливіші характеристики. Варіанти використання системи зображені в таблицях 2.1 – 2.15.

Таблиця 2.1

Варіант використання «Реєстрація»

Контекст	Реєстрація

використання	
Дійові особи	Користувач, оператор

Продовження таблиці 2.1

Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт "Реєстрація" на головній сторінці 3. Ввести дані 4. Підтвердити реєстрацію
Пост умова	Зареєстрований користувач

Таблиця 2.2

Варіант використання «Вхід у кабінет»

Контекст використання	Вхід у кабінет
Дійові особи	Користувач, оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, мати раніше створений профіль
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт "Ввійти у кабінет" на головній сторінці 3. Ввести логін та пароль 4. Підтвердити
Пост умова	Авторизація користувача

Таблиця 2.3

Варіант використання «Перегляд схеми автостоянки»

Контекст	Перегляд схеми автостоянки

використання	
Дійові особи	Користувач, оператор

Продовження таблиці 2.3

Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Перегляд схеми автостоянки»
Пост умова	Відображення Перегляд схеми автостоянки

Таблиця 2.4

Варіант використання «Бронювання місця»

Контекст використання	Бронювання місця
Дійові особи	Користувач, оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Автостоянки» 3. Вибрати необхідну автостоянку 4. Вибрати пункт «Бронювання» 5. Внести дані та зберегти
Пост умова	Забронюване місце на автостоянці.

Таблиця 2.5

Варіант використання «Транспортні засоби»

Контекст використання	Транспортні засоби
Дійові особи	Оператор

Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач

Продовження таблиці 2.5

Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Транспортні засоби»
Пост умова	Відображення усіх транспортних засобів, які містяться на автостоянці, та функції їх редагування та видалення.

Таблиця 2.6

Варіант використання «Клієнти»

Контекст використання	Видалення користувача
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Клієнти»
Пост умова	Відображення усіх клієнтів, які зареєстровані у системі, та функції їх редагування та видалення.

Таблиця 2.7

Варіант використання «Реєстрація транспортного засобу»

Контекст використання	Реєстрація транспортного засобу
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач
Тригер	Немає

Сценарій	1. Перейти на сайт 2. Вибрати пункт «Реєстрація транспортного засобу»

Продовження таблиці 2.7

Пост умова	Зареєстрований транспортний засіб

Таблиця 2.8

Варіант використання «Видалити»

Контекст використання	Видалення транспортного засобу
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач та виконаний вхід у «Транспортні засоби»
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Транспортні засоби» 3. Вибрати ТЗ 4. Вибрати пункт «Видалення»
Пост умова	Видалений ТЗ

Таблиця 2.9

Варіант використання «Видалити»

Контекст використання	Видалення клієнта
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач та виконаний вхід у «Клієнти»
Тригер	Немає

Продовження таблиці 2.9

Сценарій	1. Перейти на сайт 2. Вибрати пункт «Клієнти» 3. Вибрати клієнта 4. Вибрати пункт «Видалення»
Пост умова	Видалений клієнт

Таблиця 2.10

Варіант використання «Змінити»

Контекст використання	Зміна інформації транспортного засобу
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з'єднання з сервером, авторизований користувач та виконаний вхід у «Транспортні засоби»
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Транспортні засоби» 3. Вибрати ТЗ 4. Вибрати пункт «Змінити»
Пост умова	Змінена інформація про транспортний засіб

Таблиця 2.11

Варіант використання «Змінити»

Контекст використання	Зміна інформації клієнта
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне

	встановлення з’єднання з сервером, авторизований користувач та виконаний вхід у «Клієнти»

Продовження таблиці 2.11

Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Клієнти» 3. Вибрати клієнта 4. Вибрати пункт «Змінити»
Пост умова	Змінена інформація про клієнта

Таблиця 2.12

Варіант використання «Історія послуг»

Контекст використання	Історія послуг
Дійові особи	Оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач.
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт "Власний кабінет" на головній сторінці 3. Перейти в меню «Історія послуг».
Пост умова	Відображення усіх попередніх послуг.

Таблиця 2.13

Варіант використання «Зміна паролю»

Контекст використання	Зміна паролю
Дійові особи	Користувач, оператор

Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач.
Тригер	Немає

Продовження таблиці 2.13

Сценарій	1. Перейти на сайт 2. Перейти у «Власний кабінет» 3. Вибрати пункт «Зміна паролю» 4. Ввести новий пароль 5. Підтвердити
Пост умова	Форма зміни паролю, змінений пароль

Таблиця 2.14

Варіант використання «Вихід»

Контекст використання	Вихід
Дійові особи	Користувач, оператор
Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач.
Тригер	Немає
Сценарій	1. Перейти на сайт 2. Вибрати пункт «Вихід» на головній сторінці
Пост умова	Неавторизований користувач.

Таблиця 2.15

Варіант використання «Пошук»

Контекст використання	Пошук
Дійові особи	Оператор

Передумова	Активне підключення до мережі Інтернет та успішне встановлення з’єднання з сервером, авторизований користувач.
Тригер	Немає

Продовження таблиці 2.15

Сценарій	1. Перейти на сайт 2. Ввести у пошукову стрічку дані 3. Натиснути кнопку «Пошук»
Пост умова	Виведені результати пошуку

Проаналізувавши варіанти використання, було сформовано специфікацію функціональних вимог, яка наведена у додатку А.

Для кращого розуміння процесу функціонування веб-орієнтованої програмної системи моніторингу автотранспорту на автостоянці було побудовано діаграми діяльності для основних варіантів використання. Діаграми діяльності проілюстровані на рисунках 2.2 – 2.5.

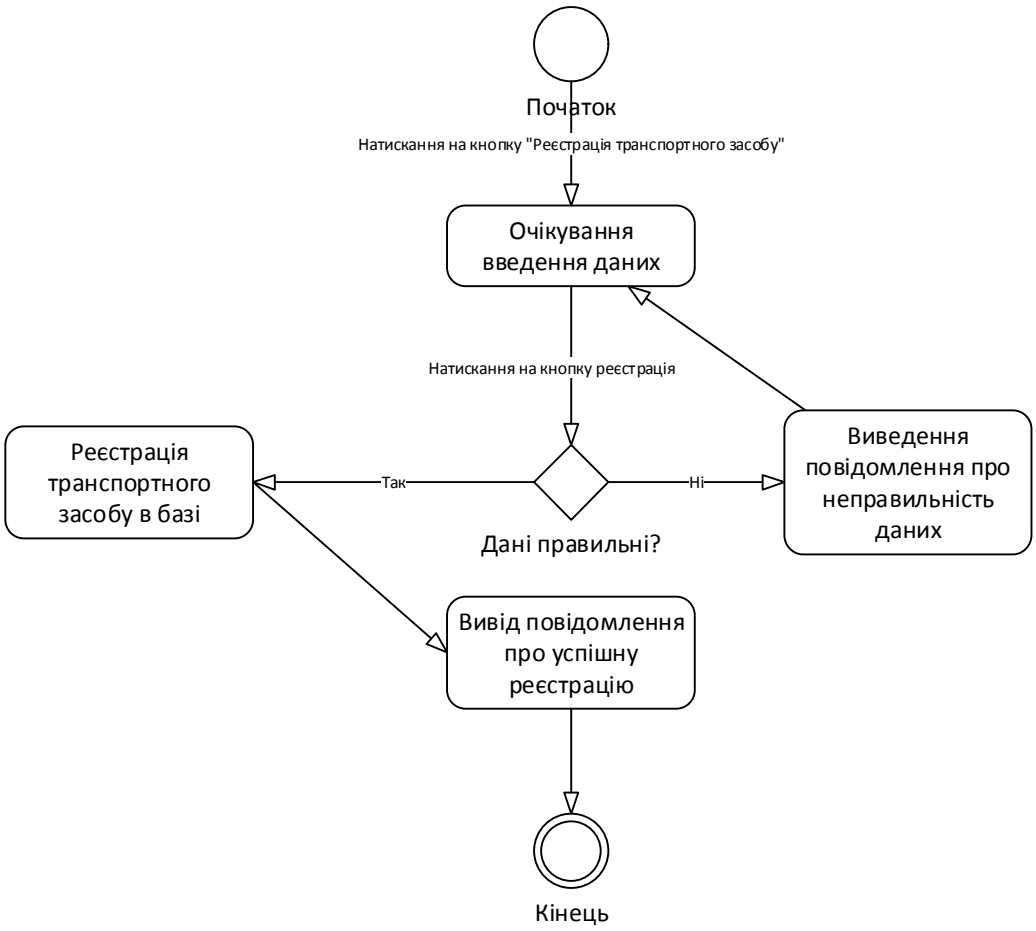


Рисунок 2.2 – Діаграма діяльності варіанту використання «Реєстрація транспортного засобу»

У випадку коли оператор, який залогінений в системі може вибрати функцію «Реєстрація транспортного засобу» та зареєструвати транспортний засіб. Для того щоб реєстрація пройшла успішно, йому потрібно правильно та коректно заповнити усі поля для даних про транспортний засіб, вказати марку, модель, реєстраційний номер, номер шасі, вибрати власника із зареєстрованих клієнтів та вибрати місце із тих, які вільні на той момент.

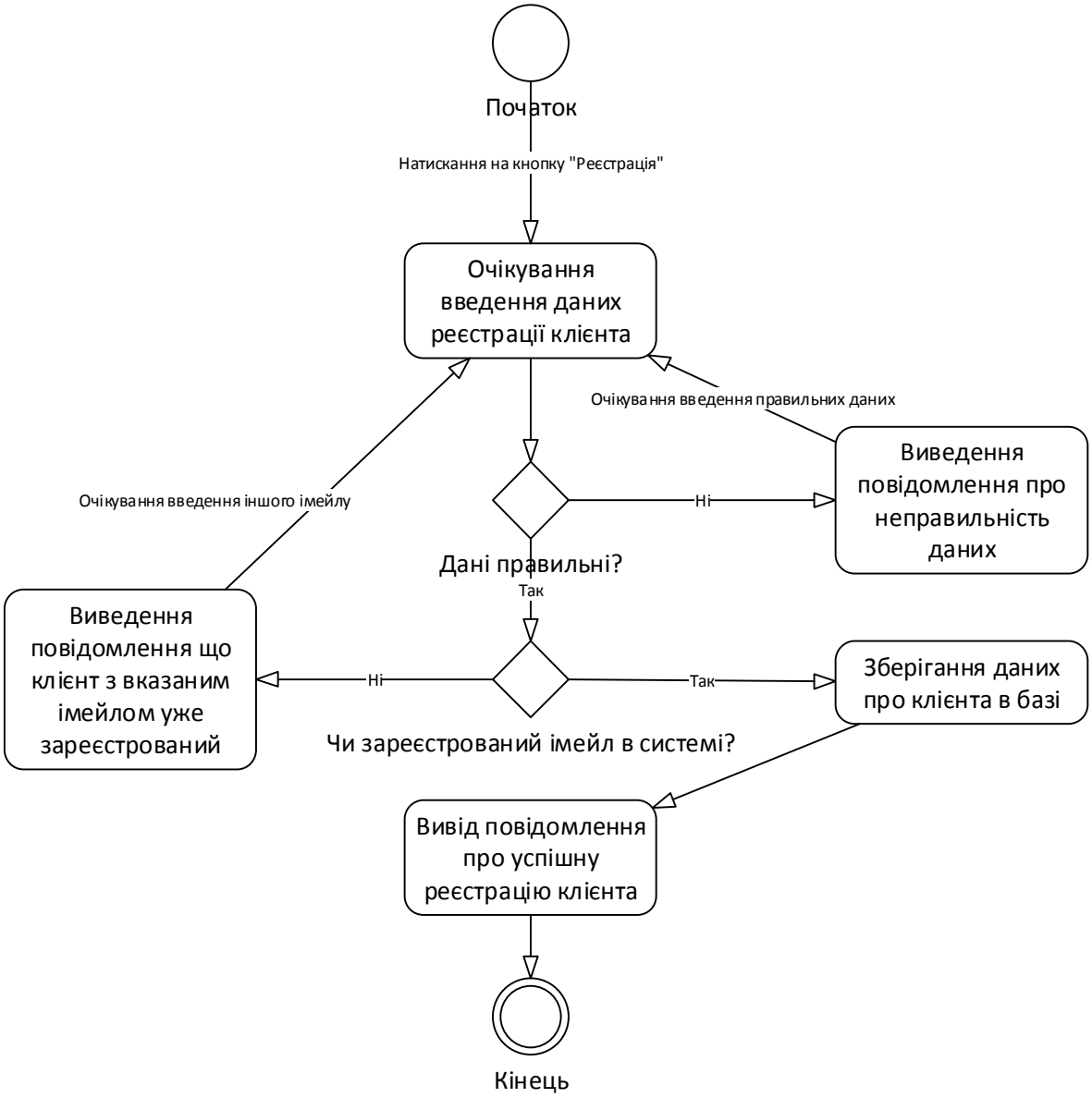


Рисунок 2.3 – Діаграма діяльності варіанту використання «Реєстрація клієнта»

У випадку, коли клієнт, який ще не зареєстрований в системі, може вибрати функцію «Реєстрація клієнта» та зареєструвати себе у системі, якщо це новий клієнт або бути зареєстрованим оператором. Для того, щоб реєстрація пройшла успішно, йому потрібно правильно та коректно заповнити усі поля для персональних даних, вказати електронну пошту, номер телефону, по якому можна зв’язатись із клієнтом в разі потреби.

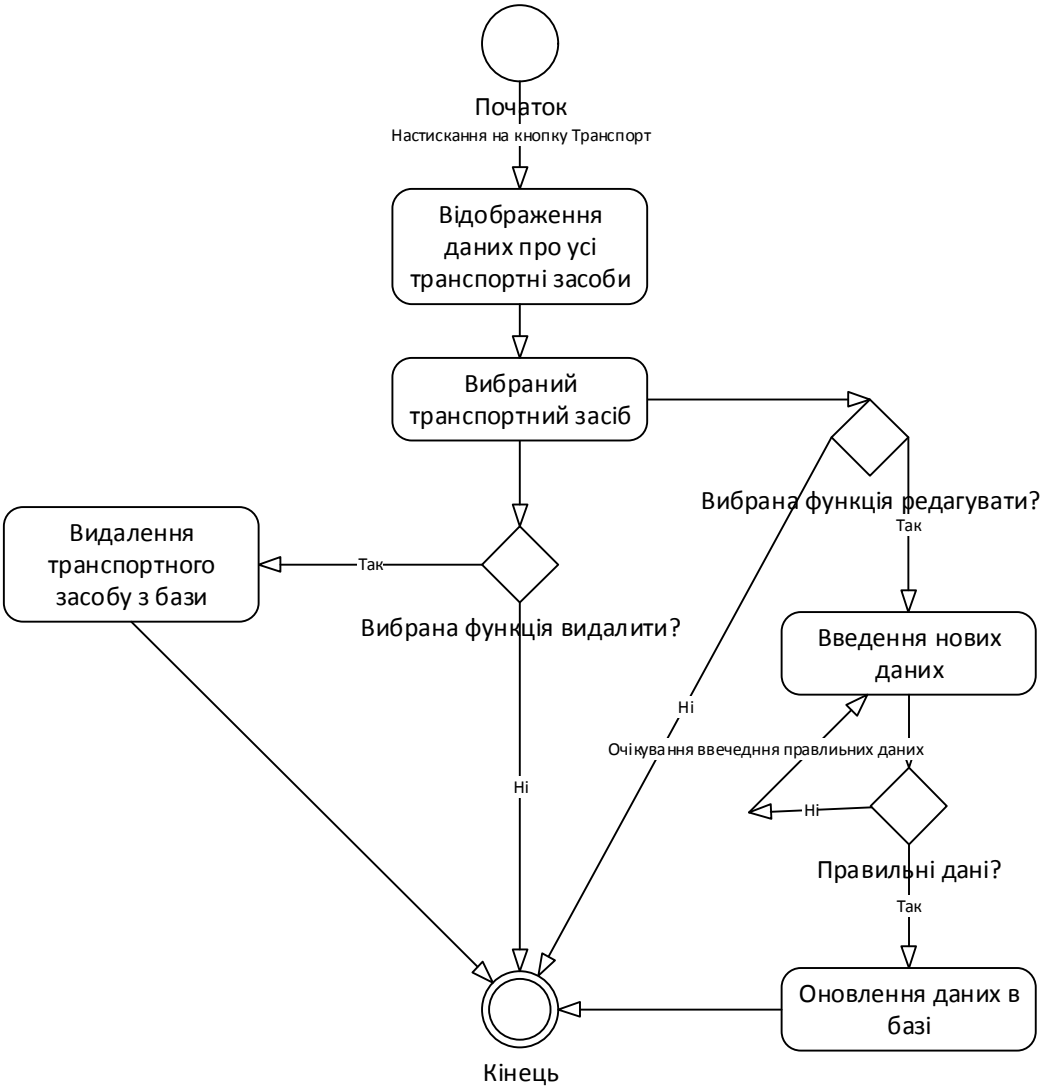


Рисунок 2.4 – Діаграма діяльності варіанту використання «Транспортні засоби»

Оператор, який залогінений в системі, може вибрати функцію «Транспортні засоби» та переглянути усі транспортні засоби, які зареєстровані у системі. Оператору доступні функції видалення та редагування даних транспортного засобу. Для того, щоб видалити транспортний засіб достатньо напроти потрібного транспорту натиснути кнопку «Видалити». Для того, щоб редагувати дані про транспортний засіб достатньо напроти потрібного транспорту натиснути кнопку «Редагувати» і ввести нові дані та підтвердити редагування.

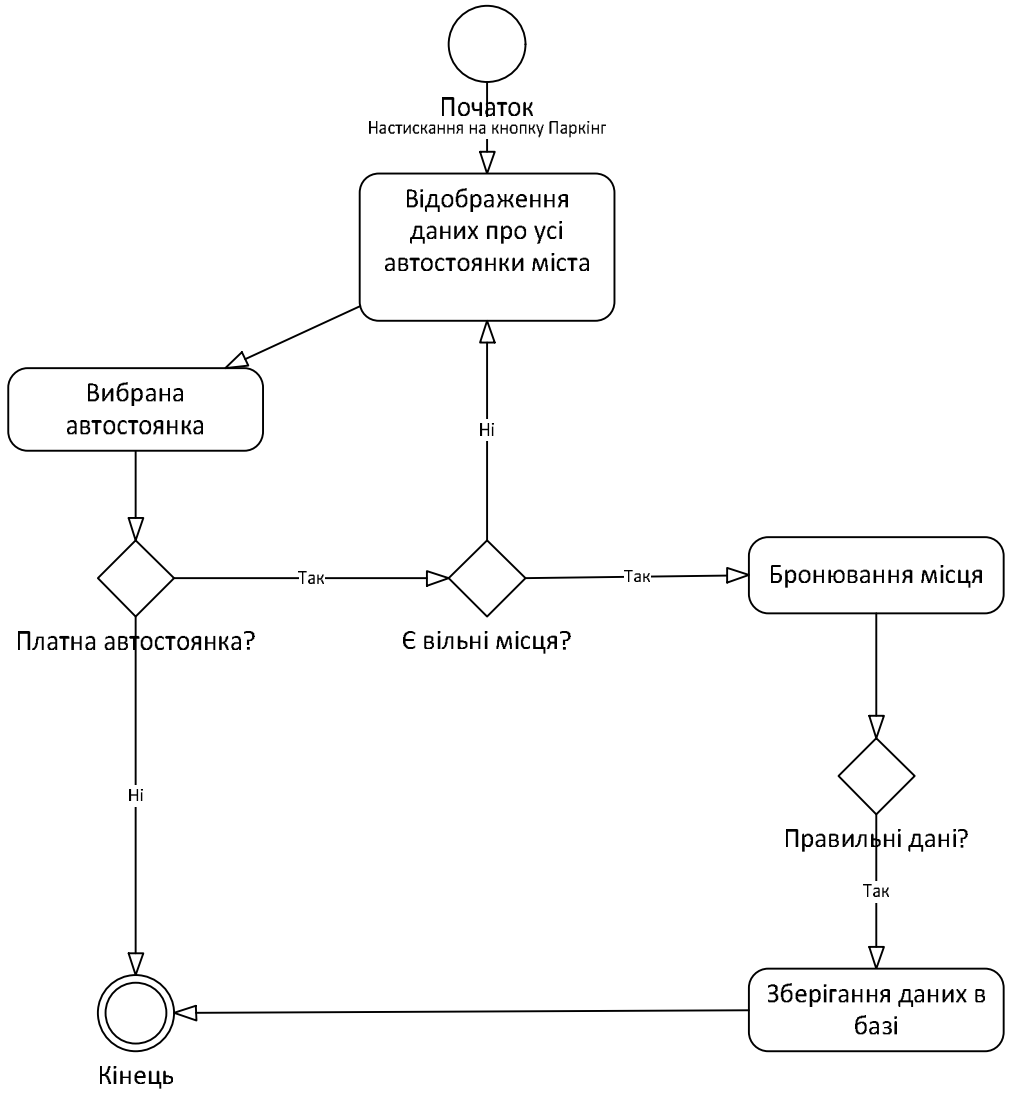


Рисунок 2.5 – Діаграма діяльності варіанту використання «Бронювання місця»

Користувач чи адміністратор, який залогінений в системі, може вибрати функцію «Бронювання місця» та забронювати місце на певний термін на конкретній автостоянці. При виборі меню «Паркінг», будуть доступні усі автостоянки міста (безплатні та платні) на карті та в таблиці. При виборі автостоянки, якщо на ній є вільні місця буде змога забронювати місце. При виборі функції бронювання необхідно буде заповнити форму із коректними даними для бронювання та зберегти зміни. Всі дані будуть внесені в базу даних.

2.2 Проектування графічного інтерфейсу

Для написання графічної частини, інтерфейсу користувача, використовувся сучасний ASP.NET MVC Framework. Це фреймворк для створення веб-застосунків, який реалізує шаблон Model-view-controller. Цей фреймворк доданий Microsoft в ASP.NET. Платформа не має жорстких рекомендацій, як повинні виглядати і працювати додатки. Існує набір рекомендацій, але в основному вони стосуються різного роду елементів типу меню, віджетів та інших речей. На даній платформі, можна писати будь які інтернет сервіси, які будуть доступні з будь якого браузера. Для розробки сервісу на даному фреймворку буде використовуватись середовище Visual Studio. Звичайно ж кожен браузер має свої особливості відображення тих чи інших елементів, тому деякі елементи написані за допомогою ASP.NET будуть відображатися інколи по різному в залежності від браузера.

Оскільки сервісом для моніторингу автостоянок будуть користуватись 2 типи користувачів, а саме адміністратори та користувачі, тому варто передбачити різні вигляди меню програми. Ця функціональність буде розділена на рівні коду, і буде розпізнаватись за логіном користувача.

Ще однією особливістю є те, що додатком можуть користуватися на різної роздільної здатності дисплеїв. Щоб уникнути проблем з розробкою інтерфейсу для великої кількості різної роздільної здатності дисплеїв, було вирішено більшість вікон зробити лише у альбомній орієнтації дисплею. Дане рішення спростить як розробку так і підтримку інтерфейсу. Недоліків не буде оскільки вікна, що вирішено розробити у альбомній орієнтації не містять багато інформації і для даних вікон інший тип орієнтації непотрібен.

Для авторизації користувача буде використовуватись один вигляд для двох типів користувачів, різниця буде лиш в адресі та написі на формі. Він буде складатись з двох полів для вводу та кнопки входу (рис. 2.6).

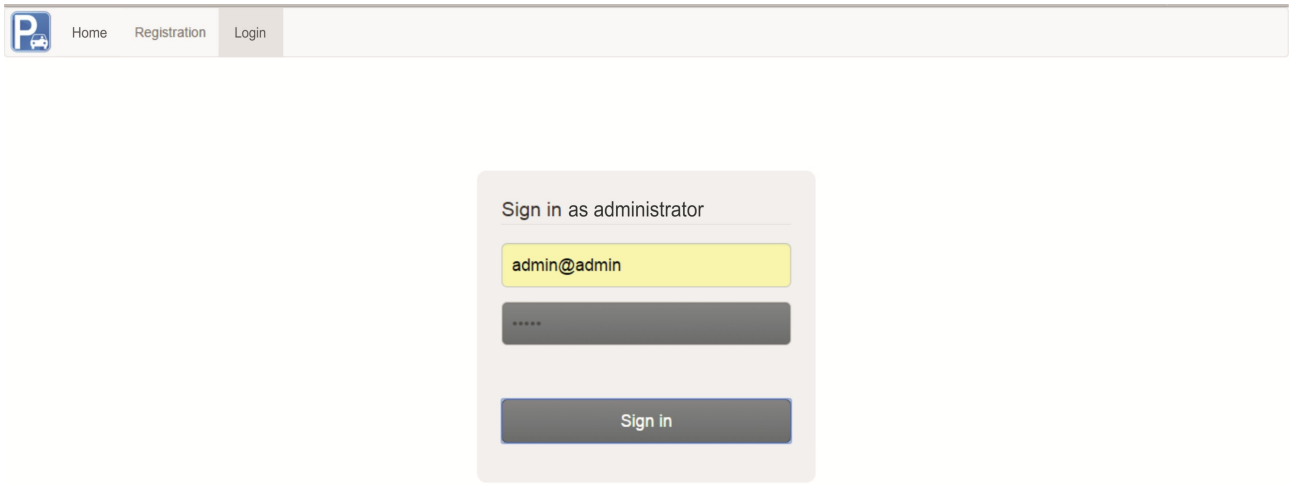


Рисунок 2.6 – Авторизація користувача

Реєстрація користувача буде можлива для адміністратора також і для користувача ззовні. Кожен користувач який захоче користуватись сервісом повинен буде зареєструватись для того, щоб керувати своїм обліковим записом, реєструвати свій транспортний засіб та бронювати місце на своє ім'я. Для адміністратора реєстрація користувача доступна для того, щоб одразу реєструвати клієнтів автостоянки, коли той вперше заїхав на автостоянку.

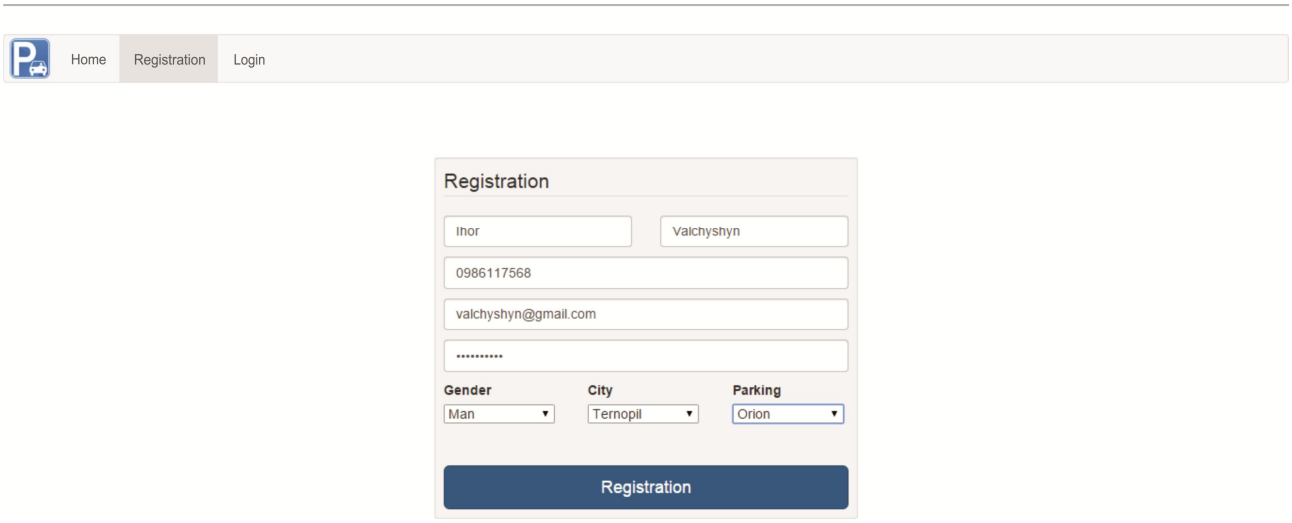
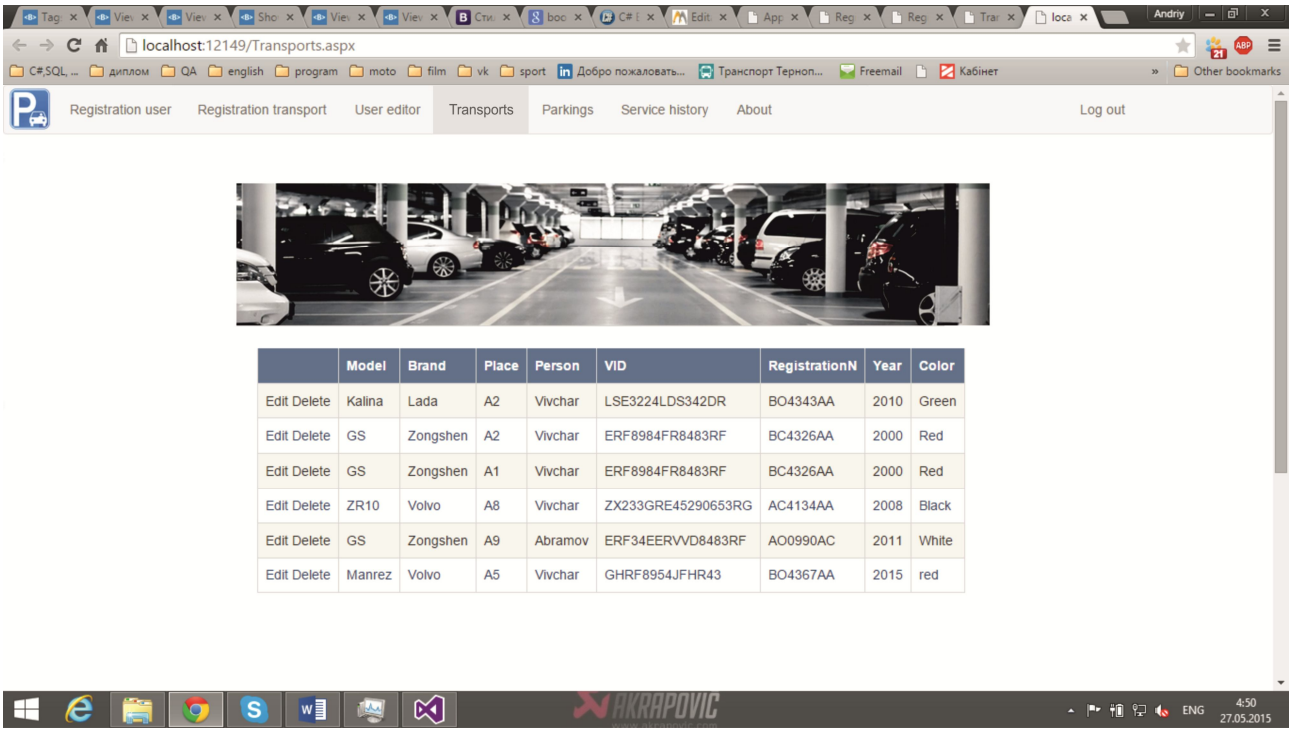


Рисунок 2.7 – Реєстрація користувача

Реєстрація транспортного засобу доступна також для двох типів користувачів. На формі вводу даних транспортного засобу, необхідно ввести VID

номер транспортного засобу, реєстраційний номер, власника, марку, модель, рік, колір та дату з якого числа по яке буде зберігатись транспортний засіб на автостоянці. Для користувача ця функція доступна для бронювання місця на автостоянці, і не потрібно буде повторно вводити дані авто при наступному вході в систему. Адміністратором ця функція використовуватиметься в звичному режимі для обліку усіх транспортних засобі на автостоянці.



	Model	Brand	Place	Person	VID	RegistrationN	Year	Color
Edit Delete	Kalina	Lada	A2	Vivchar	LSE3224LDS342DR	BO4343AA	2010	Green
Edit Delete	GS	Zongshen	A2	Vivchar	ERF8984FR8483RF	BC4326AA	2000	Red
Edit Delete	GS	Zongshen	A1	Vivchar	ERF8984FR8483RF	BC4326AA	2000	Red
Edit Delete	ZR10	Volvo	A8	Vivchar	ZX233GRE45290653RG	AC4134AA	2008	Black
Edit Delete	GS	Zongshen	A9	Abramov	ERF34EERVVD8483RF	AO0990AC	2011	White
Edit Delete	Manrez	Volvo	A5	Vivchar	GHRF8954JFHR43	BO4367AA	2015	red

Рисунок 2.8 – Інформація про весь транспорт на автостоянці

Вкладка Транспорт містить в собі усю інформацію про транспортні засоби, які знаходяться на автостоянці. Вибір автостоянки відбувається вибором з випадаючого списку автостоянки, після чого відкривається таблиця із даними транспортних засобів. Тут відображаються дані про всі наявні на автостоянці авто, які вносив оператор, чи записи які зроблені бронюванням і поточна дата збігається із датою перебування на автостоянці. Дані записи можна редагувати чи видаляти.

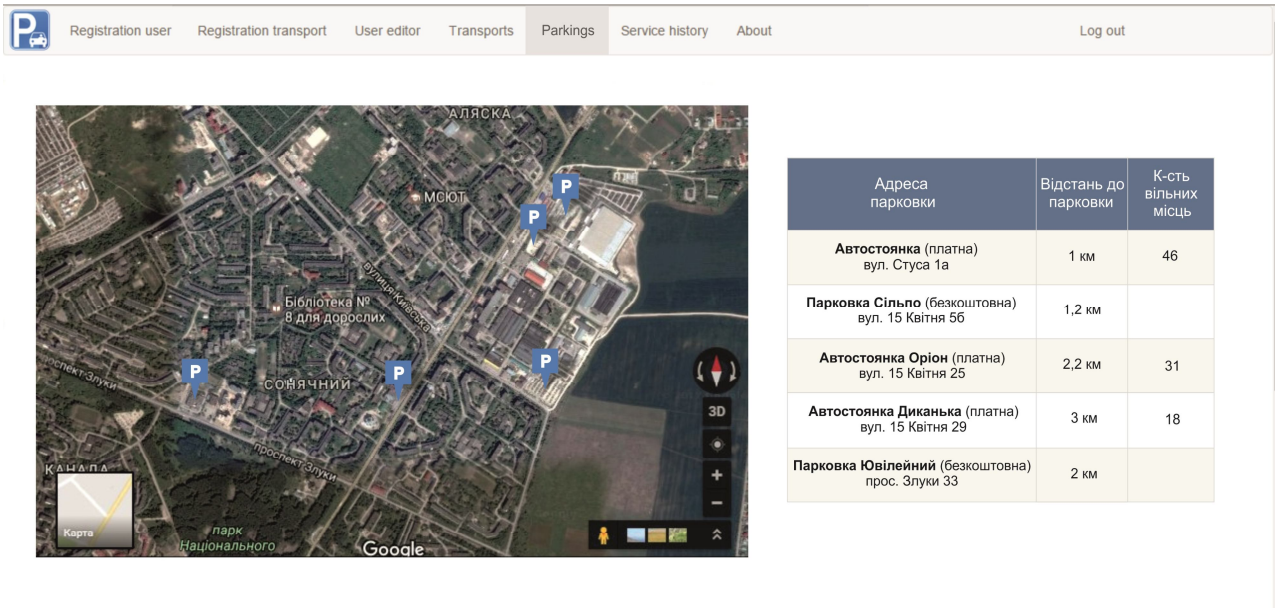


Рисунок 2.9 – Варіанти найближчих автостоянок

Пошук найближчих автостоянок здійснюється за допомогою Google Maps API. Передається значення координат розміщення, та за допомогою алгоритмів пошуку найкоротшого шляху знаходиться автостоянка яка є найближче. Також відображається назва автостоянки, точна адреса та кількість місць на автостоянці які є вільними, якщо це платна автостоянка. Якщо ж безкоштовна, це означає що це звичайна парковка поблизу торгових центрів чи тому подібного.

Рисунок 2.10 – Детальна інформація про автостоянку

Схема автостоянки, із зазначеними вільними місцями на ній зображена на рисунку 2.10. Зображено схему цілої автостоянки, із всіма її місцями, зеленим позначені вільні місця для бронювання. Також зазначено адресу, назву автостоянки та контактний телефон для зв'язку із оператором. Є можливість бронювання місця. Після натиску на кнопку для бронювання, появиться форма, майже ідентична як на реєстрації транспортного засобу, де також потрібно вводити дані авто, вибирати місце на автостоянці.

2.3 Проектування бази данних

Згідно вербального опису можна скласти таку діаграму елементів і зв'язків (рис. 2.11)

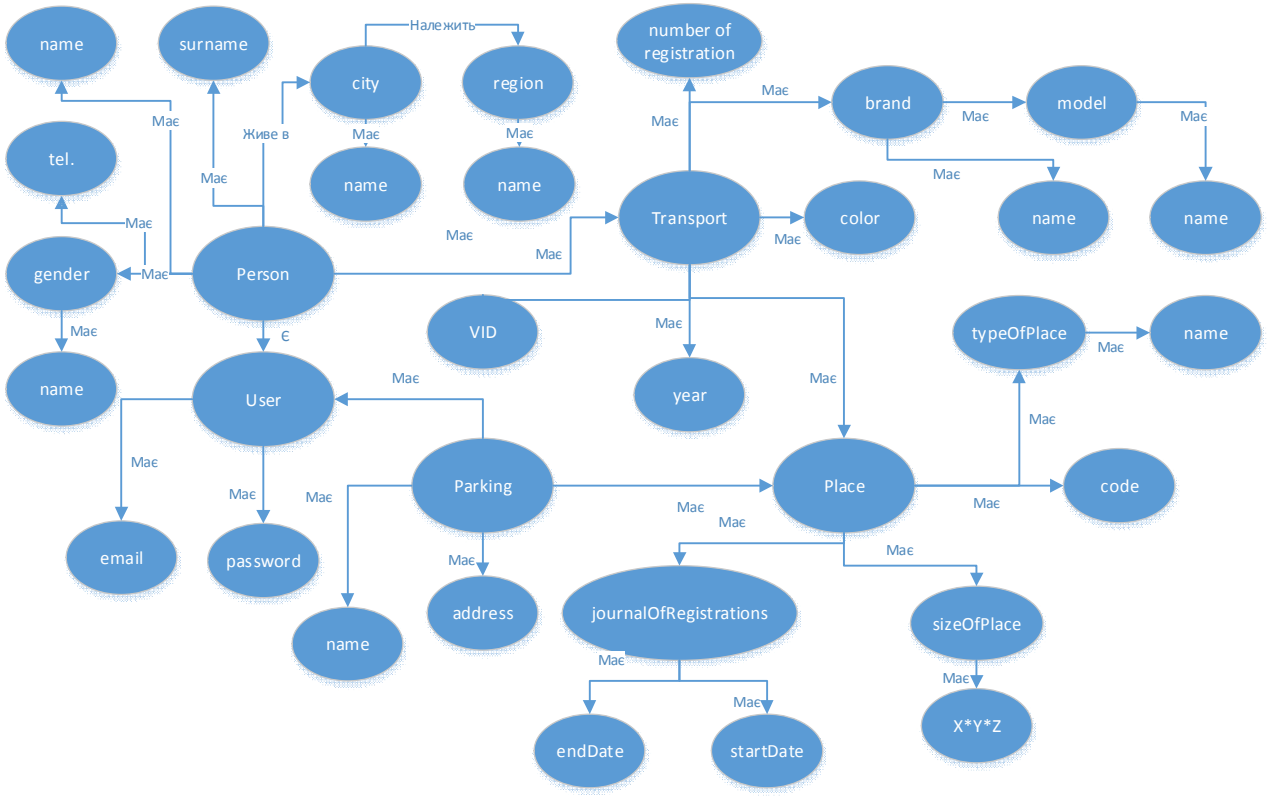


Рисунок 2.11 – Діаграма елементів і зв'язків

На основі діаграми елементів і зв'язків зображеної вище, розроблено реляційну модель бази даних програмної системи моніторингу автостоянки. Модель представлена у вигляді ER діаграми та зображена на рисунку 2.12. Діаграма взаємозв'язків сутностей використовується на концептуальному рівні

проектування бази даних. На основі усіх сутностей та зв'язків між ними була створена база даних.

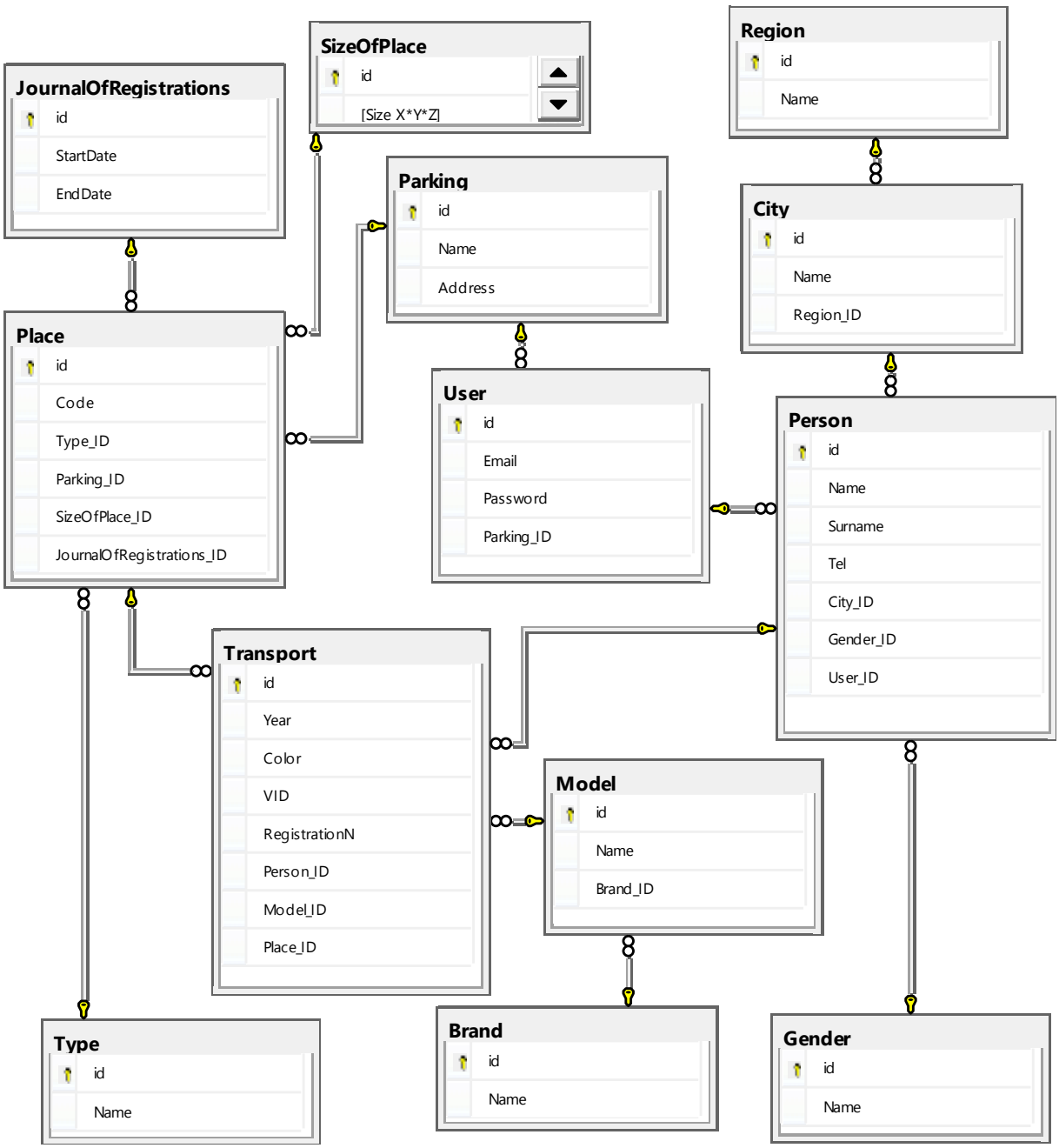


Рисунок 2.12 - ER-діаграма

Відношення SizeOfPlace буде містити унікальний ідентифікатор, та розміри місця.

Відношення JournalOfRegistrations буде містити унікальний ідентифікатор, дату поставлення транспорту на стоянку та дату коли власник має її забрати.

Відношення Parking буде містити унікальний ідентифікатор, назву та адресу.

Відношення Type буде містити унікальний ідентифікатор та назву типу.

Відношення Brand буде містити унікальний ідентифікатор та назву марки.

Відношення Gender буде містити унікальний ідентифікатор та назву статі клієнта.

Відношення Region буде містити унікальний ідентифікатор та назву області.

Відношення City буде містити унікальний ідентифікатор, назву міста та містити посилання на область, щоб знати яке місто належить певній області.

Відношення Model буде містити унікальний ідентифікатор, назву моделі та містити посилання на марку, щоб знати яка модель належить певній марці.

Відношення User буде містити унікальний ідентифікатор, електронну пошту, пароль, та містити посилання на автостоянку, щоб знати який користувач зареєстрований на певній автостоянці.

Відношення Person буде містити унікальний ідентифікатор, ім’я, прізвище, номер телефону та містити посилання на стать, щоб знати якої статі персона, посилання на місто, щоб знати де проживає персона, посилання на користувача, щоб знати дані входу конкретного користувача.

Відношення Transport буде містити унікальний ідентифікатор, рік виготовлення, колір, реєстраційний номер, номер шасі та містити посилання на персону, щоб знати кому належить транспортний засіб, посилання на модель, щоб знати до якої моделі належить транспортний засіб, посилання на місце, щоб знати на якому місці зберігається транспортний засіб.

Відношення Place буде містити унікальний ідентифікатор, код та містити посилання на тип, щоб знати яке місце відноситься до певного типу, посилання на розмір, щоб знати якого розміру місце, посилання на автостоянку, щоб знати які місця належать певній автостоянці, посилання на журнал, щоб знати час в який буде заняте місце.

Таблиця 2.16

Таблиця ідентифікаторів

Об’єкт	Властивість	Тип	Розмірність	Ідентифікатор
Gender	name	varchar	45	Name
Person	name	varchar	45	Name

Продовження таблиці 2.16

	surname	varchar	45	Surname
	tel	varchar	45	Tel
User	password	varchar	45	Password
	email	varchar	45	Email
Transport	VID	varchar	45	VID
	year	int		Year
	registration №	varchar	45	RegistrationN
	color	varchar	45	Color
SizeOfPlace	X*Y*Z	varchar	45	X*Y*Z
Model	name	varchar	45	Name
JournalOfRegistrations	startDate	datatime		StartDate
	endDate	datatime		EndDate
Region	name	varchar	45	Name
Brand	name	varchar	45	Name
City	name	varchar	45	Name
Type	name	varchar	45	Name
Parking	name	varchar	45	Name
	address	varchar	45	Address
Place	code	varchar	45	Code

Таблиця 2.17

Таблиця планування індексів

Назва таблиці	Атрибут	Опис
Person	User_ID	Foreign key
User	Parking_ID	Foreign key
Person	Gender_ID	Foreign key
Person	City_ID	Foreign key
City	Region_ID	Foreign key
Transport	Person_ID	Foreign key

Продовження таблиці 2.17

Transport	Place_ID	Foreign key
Transport	Model_ID	Foreign key
Model	Brand_ID	Foreign key
Place	Type_ID	Foreign key
Place	SizeOfPlace_ID	Foreign key
Place	JournalOfRegistrations_ID	Foreign key
Place	Parking_ID	Foreign key

2.4 Обґрунтування вибору інструментарію розробки

Для розробки WEB-системи моніторингу автостоянок, яка б реалізовувала поставлені функції, необхідно було вибрати оптимальну платформу, яка б задовільняла та підходила під усі вимоги системи. Тому потрібно вибрати мову програмування та середовище розробки, які б максимально сприяли ефективній, якісній та швидкій розробці системи. Вибираючи із існуючих технологій розробки WEB-систем, було виділено три основні та найбільш популярні мови програмування та середовища:

– PHP за допомогою PHPStorm або PHPDesigner;
– Ruby за допомогою RubyMine;
– C# технологію ASP.NET за допомогою Visual Studio 2012.

Для розробки системи моніторингу транспорту на автостоянці було вибрано мову програмування C#, технологію ASP.NET Web Forms. Середовище розробки – Microsoft Visual Studio 2012.

C# відноситься до сім'ї мов із C-подібним синтаксисом, з яких її синтаксис найбільш близький до C++ і Java. Мова має статичну типізацію, підтримує поліморфізм, перевантаження операторів (у тому числі операторів явного і неявного приведення типу), делегати, атрибути, події, властивості, узагальнені типи і методи, ітератори, анонімні функції з підтримкою замикань, винятки, коментарі у форматі XML.

ASP.NET (Active Server Pages) - технологія створення веб-додатків і веб-сервісів від компанії Microsoft. Вона є складовою частиною платформи Microsoft .NET і розвитком старішої технології Microsoft ASP. На даний момент останньою версією цієї технології є ASP.NET 5.

ASP.NET MVC є у певному значенні конкурентом для традиційних Web Form і має в порівнянні з ними деякі переваги, проте не варто однозначно відкидати ASP.NET WebForms. Оскільки вона також має свої переваги, наприклад, модель подій, яка буде ближче тим розробникам, які раніше займалися створенням клієнтських додатків.

У традиційних веб-формах можливий контроль над розміткою і можливо у реальному часі у візуальному редакторі Visual Studio побачити, як виглядатиме та чи інша сторінка. При роботі з MVC Visual Studio подібного не дозволяє робити. Також ASP.NET WebForms має ряд інших переваг:

– надає відмінні можливості для RAD (Rapid Application Development, швидка розробка додатків);

– простота розробки бізнес-додатків, що працюють з великими обсягами даних і зав'язаних на даних (data-heavy);

– величезна кількість третіх компаній, що надають готові бібліотеки контролів, а також здійснюють підтримку проектів на WebForms;

– звична Windows Forms-розробникам концепція подій (серверні події для UI), що дозволяє швидко почати працювати з таким підходом;

– відмінно підходить для швидкого створення прототипів бізнес-додатків. Це добре працює, коли необхідно узгодити з потенційним клієнтом попередню концепцію.

Microsoft Visual Studio - засіб для розробників ПЗ, яке дозволяє вирішувати основні завдання розробки: система спрощує створення, налагодження та розгортання додатків на різних платформах, включаючи SharePoint і хмарну середу.

Основними перевагами Visual Studio є:

- Використання обчислювальних потужностей локального комп'ютера і хмари;

- Проста реалізація спільних завдань та індивідуальний підхід;
- Функція підтримки декількох моніторів.

ВИСНОВКИ ДО ІІ РОЗДІЛУ

У другому розділі проведено проектування основних елементів системи «Web-орієнтована система моніторингу автостоянок», а саме:

- проаналізувавши основні взаємодіючи елементи, такі як: актори, можливості системи було спроектовано загальну структуру системи. Виявлено, що користуватися системою будуть не лише адміністратори автостоянок а і звичайні жителі чи гості міста. Представлено можливі варіанти використання системи виявленим акторам. Визначено основні функції доступні як системі в цілому так і користувачам;
- згідно акторів та пункту призначення системи спроектовано сутності бази даних та надано зв`язки між ними.

РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ WEB-ОРІЄНТОВАНОЇ СИСТЕМИ МОНІТОРИНГУ АВТОСТОЯНОК

3.1 Розробка архітектури системи

Діаграмою класів в сфері UML називається діаграма, на якій зображено набір класів, а також зв’язків між цими класами. Крім того, діаграма класів може містити коментарі і обмеження. Обмеження можуть неформально задаватися на звичайній мові або ж можуть формулюватись на мові об’єктних обмежень OCL. На діаграмах класів відображаються також властивості класів та методи.

Розроблювана програмна система для моніторингу автостоянок складається з одного проекту, який умовно можна поділити на 3 складових. Перша складова це клас Repository, де реалізовано взаємодію та операції з базою даних. Другою складовою є реалізація усіх функцій системи, а третьою складовою є реалізація дизайну програмної системи.

Цей продукт є web додатком, так як на сьогоднішній день такі системи стають все популярніші, то в майбутньому можливий ріст та розвиток даного продукту. Для цього потрібно враховувати можливість внесення змін у продукт.

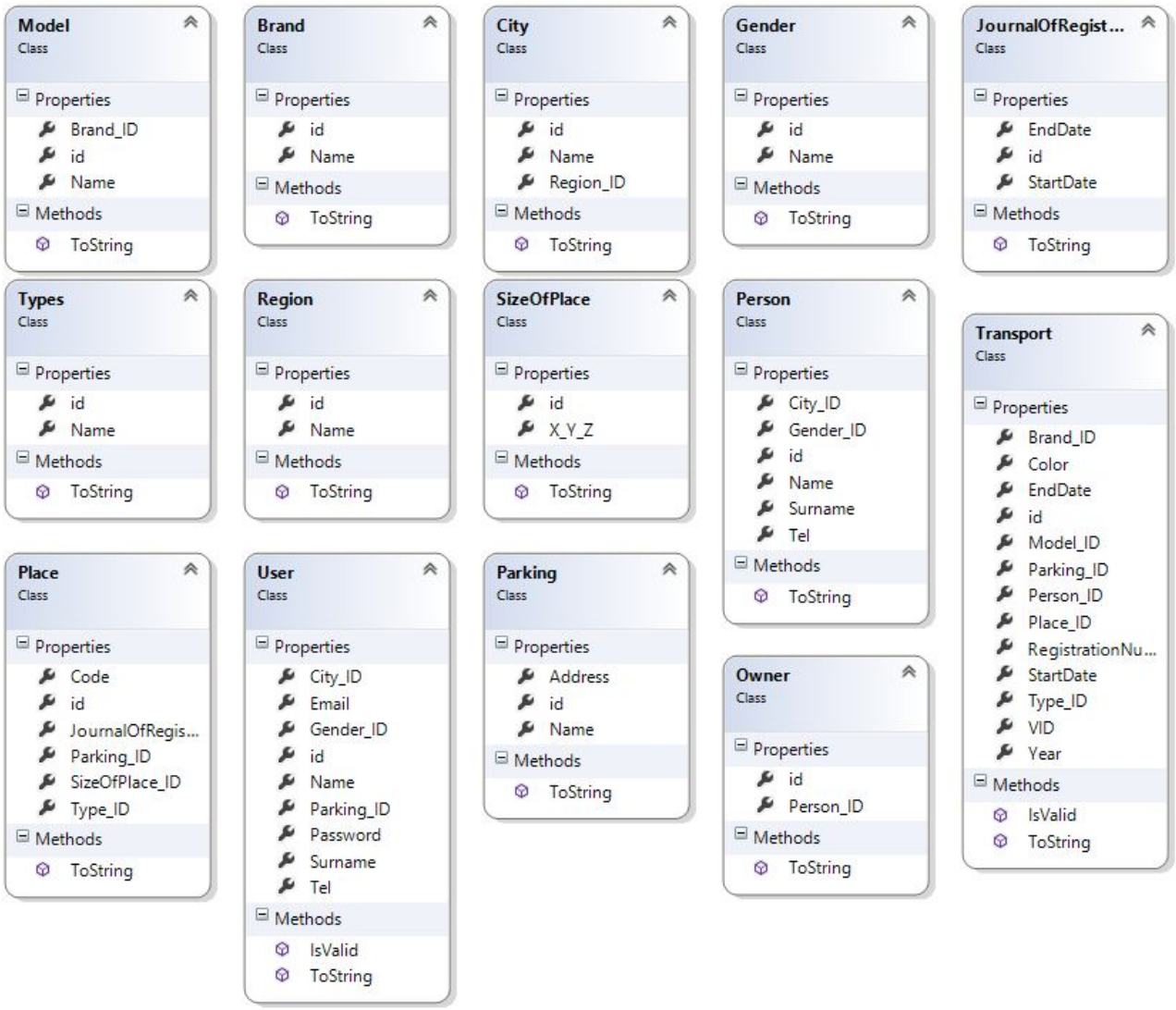


Рисунок 3.1 – Діаграма класів моделей

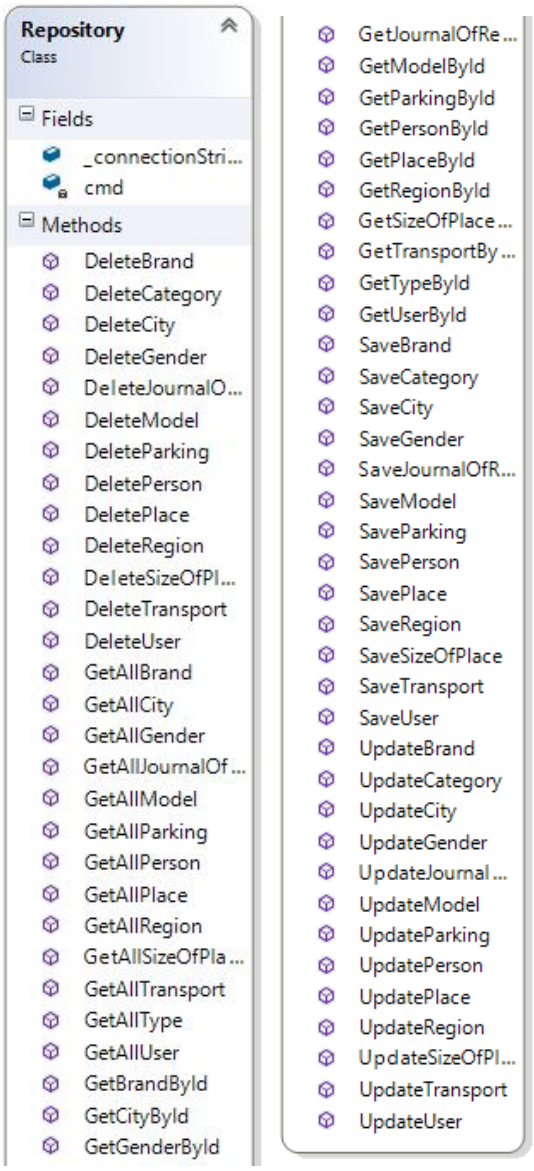


Рисунок 3.2 – Клас Repository

Клас Repository – реалізовує операції CRUD з базою даних для всіх елементів системи.

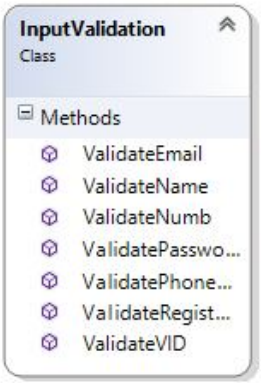


Рисунок 3.3– Клас InputValidation

Клас InputValidation – відповідає за перевірку вхідних даних системи: ім’я, прізвище, номер телефону, реєстраційний номер транспортного засобу, номер шасі, пароль, електронну пошту.

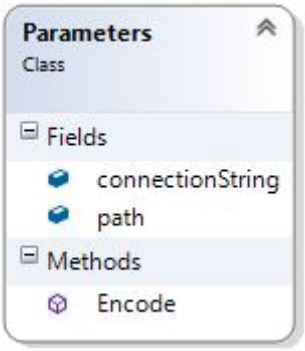


Рисунок 3.4 – Клас Parameters

Клас Parameters – вміщує стрічку підключення до бази даних, шлях зберігання файлу зі звітом та метод, за допомогою якого здійснюється шифрування паролю нового користувача при реєстрації.

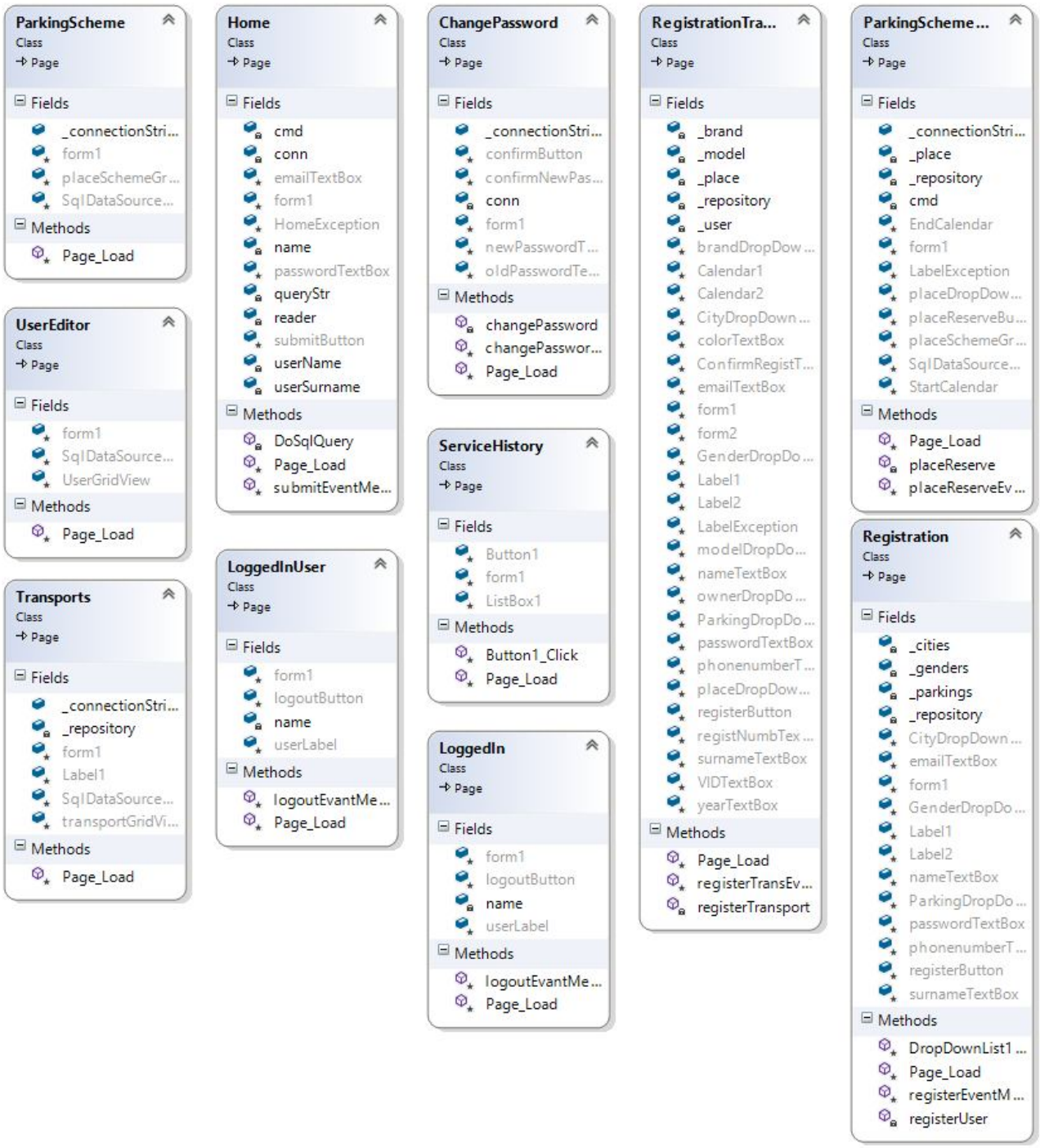


Рисунок 3.5 – Діаграма класів реалізації усіх функцій системи

Кожен з класів, зображених на рисунку 3.5, відповідає за окрему функціональну сторінку системи.

Для прикладу розглянемо клас ChangePassword. На рівні бізнес логіки цей клас перевіряє введення старого пароля, введення нового пароля, підтвердження зміни старого пароля на новий та зміну пароля в базі даних. Детальніше в класі відбувається перевірка нового пароля який вводить користувач. Якщо він відповідає нормам, то користувач може продовжити процес зміни пароля. Далі

відбувається порівняння введеного нового пароля із стрічкою де вводиться повторно новий пароль. Якщо паролі збігаються то для цього паролю викликається метод Encode із класу Parameters. Метод Encode приймає стрічку та шифрує її по алгоритму SHA1 і повертає вже хешовану стрічку. Далі відкривається з'єднання з базою даних і за допомогою SQL запиту вибирається старий пароль користувача. Він порівнюється із введеним старим паролем у захешованому вигляді, якщо вони співпадають, то відбувається оновлення пароля в базі даних.

3.2 Програмна реалізація проекту

Після вибору технології, мови програмування та середовища програмування наступним є безпосередньо етап кодування. Для прикладу, наведемо код оголошення класу Registration.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;
using System.Web.UI;
using System.Web.UI.WebControls;
using System.Data.SqlClient;
using System.Data;
using System.Configuration;
using AutoParkingSystem.Models;
using System.IO;

namespace AutoParkingSystem
{
   public partial class Registration : System.Web.UI.Page
   {
      Repository _repository = new Repository();

      List<City> _cities;
      List<Gender> _genders;
      List<Parking> _parkings;

      protected void Page_Load(object sender, EventArgs e)
      {
         _cities = _repository.GetAllCity();
         if (CityDropDownList.SelectedIndex == -1)
         {
            CityDropDownList.DataSource = _cities;
            CityDropDownList.DataBind();
            CityDropDownList.SelectedIndex = -1;
```

```
            }

            _genders = _repository.GetAllGender();
            if (GenderDropDownList.SelectedIndex == -1)
            {
                GenderDropDownList.DataSource = _genders;
                GenderDropDownList.DataBind();
                GenderDropDownList.SelectedIndex = -1;
            }

            _parkings = _repository.GetAllParking();
            if (ParkingDropDownList.SelectedIndex == -1)
            {
                ParkingDropDownList.DataSource = _parkings;
                ParkingDropDownList.DataBind();
                ParkingDropDownList.SelectedIndex = -1;
            }
        }
        protected void registerEventMethod(object sender, EventArgs e)
        {
            registerUser();
        }
        private void registerUser()
        {

            bool methodStatus = true;

            if (InputValidation.ValidatePhoneNumber(phonenumberTextBox.Text) == false)
            {
                methodStatus = false;
            }
            if (InputValidation.ValidateName(nameTextBox.Text) == false)
            {
                methodStatus = false;
            }
            if (InputValidation.ValidateName(surnameTextBox.Text) == false)
            {
                methodStatus = false;
            }
            if (InputValidation.ValidatePassword(passwordTextBox.Text) == false)
            {
                methodStatus = false;
            }
            if (InputValidation.ValidateEmail(emailTextBox.Text) == false)
            {
                methodStatus = false;
            }
            using (var sw = new StreamWriter(Parameters.path, true))
            {
                sw.WriteLine("Admin has register " + nameTextBox.Text+" "+surnameTextBox.Text + "
on " + DateTime.Now);
```

```
            }
                try
                {

                    User entity = new User()
                    {
                        Name = nameTextBox.Text,
                        Surname = surnameTextBox.Text,
                        Tel = phonenumberTextBox.Text,
                        Email = emailTextBox.Text,
                        Password = passwordTextBox.Text,
                        City_ID = _cities[CityDropDownList.SelectedIndex].id,
                        Gender_ID = _genders[GenderDropDownList.SelectedIndex].id,
                        Parking_ID = _parkings[ParkingDropDownList.SelectedIndex].id

                    };

                    if (!entity.IsValid())
                    {
                      Label1.Text="Not valid data";
                        return;
                    }

                    _repository.SaveUser(entity);
                    Label2.Text = nameTextBox.Text + " " + surnameTextBox.Text + " registered
successfully.";
                    nameTextBox.Text = "";
                    surnameTextBox.Text = "";
                    phonenumberTextBox.Text = "";
                    emailTextBox.Text = "";
                    passwordTextBox.Text = "";
                }
                catch (Exception ex)
                {
                    Label1.Text = "Check your data and try again.";
                    return;
                }

            }
        }
    }
```

Розглянемо детальніше реалізацію методу private void registerUser() класу Registration

```
        private void registerUser()
        {

            bool methodStatus = true;
```

```
                if (InputValidation.ValidatePhoneNumber(phonenumberTextBox.Text) == false)
                {
                   methodStatus = false;
                }
                if (InputValidation.ValidateName(nameTextBox.Text) == false)
                {
                   methodStatus = false;
                }
                if (InputValidation.ValidateName(surnameTextBox.Text) == false)
                {
                   methodStatus = false;
                }
                if (InputValidation.ValidatePassword(passwordTextBox.Text) == false)
                {
                   methodStatus = false;
                }
                if (InputValidation.ValidateEmail(emailTextBox.Text) == false)
                {
                   methodStatus = false;
                }
                using (var sw = new StreamWriter(Parameters.path, true))
                {
                   sw.WriteLine("Admin has register " + nameTextBox.Text+" "+surnameTextBox.Text + "
on " + DateTime.Now);
                }
                   try
                   {

                      User entity = new User()
                      {
                         Name = nameTextBox.Text,
                         Surname = surnameTextBox.Text,
                         Tel = phonenumberTextBox.Text,
                         Email = emailTextBox.Text,
                         Password = passwordTextBox.Text,
                         City_ID = _cities[CityDropDownList.SelectedIndex].id,
                         Gender_ID = _genders[GenderDropDownList.SelectedIndex].id,
                         Parking_ID = _parkings[ParkingDropDownList.SelectedIndex].id

                      };

                      if (!entity.IsValid())
                      {
                        Label1.Text="Fill in all fields";
                         return;
                      }

                      _repository.SaveUser(entity);
                      Label2.Text = nameTextBox.Text + " " + surnameTextBox.Text + " registered
successfully.";
                      nameTextBox.Text = "";
```

```
            surnameTextBox.Text = "";
            phonenumberTextBox.Text = "";
            emailTextBox.Text = "";
            passwordTextBox.Text = "";
        }
        catch (Exception ex)
        {
            Label1.Text = "Check your data and try again.";
            return;
        }

    }
```

Метод private void registerUser() відповідає за перевірку правильності введених користувачем системи даних при реєстрації та організовує збір даних в один об’єкт entity класу User, тобто усім властивостям даного об’єкту присвоюються конкретні дані. Тоді за допомогою методу зберігання даних користувача в базі даних SaveUser відбувається реєстрація. Виводиться повідомлення про успішну реєстрацію користувача.

3.3 Програмна реалізація модуля роботи із картами

Для реалізації методів пошуку автостоянок та роботи із картами використовуємо Google Maps API.

Google Maps API дозволяє вбудовувати карти прямо в сторінки вашого сайту. Для чого вам буде потрібно трохи JavaScript, а для створення красивого оформлення - трохи CSS. Для подальшої роботи з API картами необхідний ключ, при запиті всіх їх скриптів і сервісів в параметри потрібно додавати & key. Отримати його можна на офіційному сайті, створивши там свій проект, система згенерує унікальний ключ. Втім для http://localhost він не потрібен.

Роботу з картою можна описати кількома моментами:

- Додаток декларується в форматі HTML5 з використанням декларації

<! DOCTYPE html>.

- Для зберігання карти створюється елемент div з ім'ям "map".
- Визначається функція JavaScript, яка створює карту в елементі div.
- За допомогою тега script завантажується код Maps API JavaScript.

Для роботи нам знадобиться 3 основних об'єкта. Перше - це карта.

Карта створюється дуже просто. У нас є якийсь певний контейнер:

```
<div id="map_canvas"></div>
```

Загружається Google Maps API за допомогою наступного коду:

```
<script async defer
src="https://maps.googleapis.com/maps/api/js?key= API_KEY&callback=initMap">
</script>
```

URL-адресу в тезі script вказує місце розташування файлу JavaScript, який завантажує всі символи і визначення, необхідні для використання Google Maps API. Цей тег script є обов'язковим.

Атрибут async дозволяє браузеру виводити на екран іншу частину сайту під час завантаження Maps API. Коли API готовий до роботи, він викликає функцію, зазначену в параметрі callback.

Параметр key містить ключ API вашого застосування.

Карта ініціалізується за допомогою наступного скрипта:

```
function initialize() {
var myLatlng = new google.maps.LatLng(-34.397, 150.644);
var myOptions = {
zoom: 8,
 center: myLatlng,
 mapTypeId: google.maps.MapTypeId.ROADMAP
 }
var map = new google.maps.Map(document.getElementById("map_canvas"), myOptions);
}
```

center: myLatlng - це координати центру карти

zoom - це збільшення при ініціалізації

mapTypeId - тип (політична, фізична, гібрид).

Друге - це мітки:

```
map.addMarker({
        lat: 49.3329,
         lng: 25.3857,
         title: 'Parking',
        map: map,
         click: function(e) {
                 alert('You clicked in this marker');
         },
```

```
        infoWindow: {
        content: '<p>Автостоянка, Стуса 1А</p>'
        }
}
});
```

lat, lng - власне координати мітки

map - на яку карту мітку помістити

title - при наведенні миші буде писати "Parking"

content - вміст в мітці.

3.4 Програмна реалізація бази даних

Після процесу проектування бази даних відбувається реалізація зпроектованої схеми у вигляді реальної бази даних, яку отримали як результат виконання скрипта (див. додаток) створення бази даних у RDBMS. Базу даних web-орієнтованої програмної системи для моніторингу транспорту на автостоянці реалізовано із використанням RDBMS MSSQLServer 2012.

Microsoft SQL Server 2012 – система керування реляційними базами даних, розроблена корпорацією Microsoft. Основна використовувана мова запитів - Transact-SQL (скорочено T-SQL), створена спільно Microsoft та Sybase. Transact-SQL є реалізацією стандарту ANSI / ISO по структурованій мові запитів (SQL) з розширеннями. Використовується для роботи з базами даних розміром від персональних до великих баз даних масштабу підприємства.

SQL Server 2012 має дуже добре поставлену менеджмент систему, яка довзоляє розробнику швидко і гнучко оперувати та опрацьовувати дані. У ній можна без зайвих зусиль через зрозумілий інтерфейс без втручання в процес кодування створити нову базу даних, таблички в ній, зв’язки між цими табличками, виконувати запити до бази даних, зберігати процедури збереження даних.

При програмуванні бази даних системи моніторингу транспорту на автостоянці було створено базу даних ParkingSystem із табличками для збереження даних. Для полегшення реалізації основних функцій видалення, оновлення,

читання та створення використовуються моделі, які відображають структуру таблиць у базі даних. Нижче наведений приклад декількох моделей.

```
using System;
using System.Collections.Generic;
using System.Linq;
using System.Web;

namespace AutoParkingSystem.Models
{
  public class User
  {
    public Int32 id { get; set; }
    public String Email { get; set; }
    public String Password { get; set; }
    public Int32 Parking_ID { get; set; }

    public String Name { get; set; }
    public String Surname { get; set; }
    public String Tel { get; set; }
    public Int32 City_ID { get; set; }
    public Int32 Gender_ID { get; set; }
    public Boolean IsValid()
    {
      if (String.IsNullOrWhiteSpace(Email) || String.IsNullOrWhiteSpace(Name) ||
String.IsNullOrWhiteSpace(Surname)
        || String.IsNullOrWhiteSpace(Tel) || String.IsNullOrWhiteSpace(Password))
        return false;
      return true;
    }
    public override string ToString()
    {
      return Name + " " + Surname;
    }
  }
```

}

Можемо бачити у прикладі, наведена модель User. Кожна модель повинна містити властивість id, для того, щоб не виникало помилок, так як система не зможе знайти властивість id таблиці. Далі, за допомогою атрибутів встановлюються усі властивості колонок, та проводиться перевірка на відсутність пустих властивостей. якщо хоча б одна пуста, то програма покаже повідомлення про помилку із зазначенням, що не всі поля є заповнені.

Усі методи роботи з базою даних реалізовані у класі Repository. Для прикладу розглянемо метод SaveTransport(). На початку методу відкривається з'єднання з базою даних, далі створюється об'єкт cmd класу SqlCommand, який зберігає в собі stored-процедуру збереження даних про транспортний засіб. Через властивість Parameter ми присвоюємо колонкам значення з об'єкту entity і виконуємо процедуру командою cmd.ExecuteNonQuery(). Нище наведений код stored-процедури зберігання даних про транспортний засіб.

```
public void SaveTransport(Transport entity)
    {
        using (SqlConnection conn = new SqlConnection(_connectionString))
        {
            conn.Open();

            using (cmd = new SqlCommand("InsertTransport", conn))
            {

                cmd.CommandType = CommandType.StoredProcedure;
                cmd.Parameters.Add("@StartDate", SqlDbType.DateTime).Value = entity.StartDate;
                cmd.Parameters.Add("@EndDate", SqlDbType.DateTime).Value = entity.EndDate;
                cmd.Parameters.Add("@Place_ID", SqlDbType.Int).Value = entity.Place_ID;
                cmd.Parameters.Add("@Year", SqlDbType.Int).Value = entity.Year;
                cmd.Parameters.Add("@Color", SqlDbType.VarChar).Value = entity.Color;
                cmd.Parameters.Add("@VID", SqlDbType.VarChar).Value = entity.VID;
                cmd.Parameters.Add("@RegistrationN", SqlDbType.VarChar).Value =
entity.RegistrationNumber;
                cmd.Parameters.Add("@Person_ID", SqlDbType.Int).Value = entity.Person_ID;
```

```
            cmd.Parameters.Add("@Model_ID", SqlDbType.Int).Value = entity.Model_ID;

            cmd.ExecuteNonQuery();
          }
        }
}
```

Код stored процедури InsertTransport

```
USE [ParkingSystem]
GO
SET ANSI_NULLS ON
GO
SET QUOTED_IDENTIFIER ON
GO

CREATE PROCEDURE [dbo].[InsertTransport]
            @StartDate          DateTime,
            @EndDate            DateTime,

            @Place_ID                       INT=1,
            @JournalOfRegistrations_ID INT=1,
            @Year                           INT,
            @Color                              VARCHAR(50),
            @VID                            VARCHAR(50),
            @RegistrationN                  VARCHAR(50),
            @Person_ID                      INT,
            @Model_ID                       INT

AS
BEGIN
   SET NOCOUNT ON

Begin tran T1
  INSERT INTO dbo.[JournalOfRegistrations]
```

```
(
 StartDate,
                EndDate
)
VALUES
(
 @StartDate,
                @EndDate
)

                Select @JournalOfRegistrations_ID=JournalOfRegistrations.id from
JournalOfRegistrations
                Where JournalOfRegistrations.id=(select max(id) from JournalOfRegistrations)

                 UPDATE  dbo.Place SET
                    --Parking_ID=@Parking_ID,
                    JournalOfRegistrations_ID=@JournalOfRegistrations_ID
                    WHERE Place.id=@Place_ID

                INSERT INTO dbo.Transport
(
 Year,
                    Color,
                    VID,
                    RegistrationN,
                    Person_ID,
                    Model_ID,
                    Place_ID
)
VALUES
(
@Year,
                @Color,
                @VID,
```

```
            @RegistrationN,
            @Person_ID,
            @Model_ID,
            @Place_ID
    )

commit tran t1

END
```

В цій процедурі ми спочатку відкриваємо транзакцію. Транзакція – це група послідовних операцій з базою даних, яка є логічною одиницею роботи з даними. Транзакція може бути виконана або цілком і успішно, дотримуючись цілісності даних і незалежно від інших транзакцій, що йдуть паралельно, або не виконана зовсім, і тоді вона не дає ніякого ефекту. Далі записуємо дані в журнал, для того щоб взяти id цього запису та присвоїти його місцю на яке буде ставитись транспортний засіб. Проводиться оновлення даних місця на яке ставиться транспортний засіб, якщо дані журналу були то оновлюються, якщо ні то записуються нові. І тоді аж проводиться запис усіх даних транспортного засобу та закривається транзакція.

Results | Messages

	id	Year	Color	VID	RegistrationN	Person_ID	Model_ID	Place_ID
1	3014	2015	red	GHRF8954JFHR43	BO4367AA	2	6	6

Рисунок 3.6 – результат виконання stored процедури InsertTransport.

ВИСНОВКИ ДО ІІІ РОЗДІЛУ

У третьому розділі здійснено реалізацію основних частин системи «Web-орієнтованої системи моніторингу автостоянок».

Проаналізувавши основні взаємодіючі елементи, побудовано систему класів, які наявні в системі, описано їх функціонал та вміст. Представлено, який клас за що відповідає та з ким взаємодіє. На основі цього можна чітко зрозуміти, який клас до якої частини програми належить, коли буде виконуватися та де, яку відповідальність на нього покладено;

На основі розробленої архітектури, аналізу моделей, що будуть в системі реалізовано ядро серверу. Для доступу до бази через інтернет використано концепцію ASP.NET MVC, що представляє, базу як модель, джерело даних, та власне контролери, як точки входу у веб-сервіс. Кожен контролер керує власною таблицею, та містить різного роду атрибути для точної ідентифікації запитів з відповідними методами.;

- Згідно проаналізованих вимоги у другому розділі та спроектованого графічного інтерфейсу додатку здійснено реалізацію web-орієнтованої системи. Для цього використано швидко зростаючий у потужності та перспективах фреймворк ASP.NET MVS, який дозволяє писати на мові C#.