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

ЗДИРКО Василь Васильович

Математичне та програмне забезпечення для організації роботи call-центру таксі/ Mathematic tools and software for tasks organizing in taxi call-center

спеціальність: 8.05010301 - Програмне забезпечення систем
магістерська програма - Програмне забезпечення систем

Магістерська робота

Виконав студент групи ПЗСм-21
В. В. Здирко
_____

Науковий керівник:
к.т.н., доцент СТРУБИЦЬКА І.П.
_____

Магістерську роботу допущено
до захисту:

"___" _____ 20___ р.

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

ТЕРНОПІЛЬ - 2016

Зміст

ВСТУП

Актуальність теми

З розвитком комп'ютерних-інформаційних технологій з'являється багато нових можливостей для розширення бізнесу, задля збільшення прибутків фірми та розширення клієнтської бази. Одним із способів впровадження інформаційних технологій у процес є створення власних сайтів.

Тому актуальним є розробка web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі. По-перше, це допоможе обслуговувати більше клієнтів і дозволить уникнути помилок через «людський фактор». По-друге, це заощадить час як клієнта так і диспетчера.

Мета і задачі дослідження

Метою є розробка оптимізваної web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі на основі використання математичної моделі пошуку найкоротшого шляху.

Дана мета реалізується вирішенням наступних завдань:

1. аналіз роботи диспетчера таксі;
2. порівняльний аналіз існуючих web-орієнтованих систем;
3. дослідження алгоритму пошуку найкоротшого шляху від машини до замовника;
4. дослідження Google Maps API для реалізації поставленої задачі;
5. дослідження виявлення геоданих пристрою за допомогою технології GPS;
6. аналіз процесу розподілення замовлення;
7. проектування та розробка методу передачі даних;

Об’єкт дослідження – процес пошуку найкоротшого шляху від водія таксі до замовника.

Предмет дослідження – методи та засоби пошуку найкоротшого шляху.

Методи дослідження

У роботі використано методи теорії скінченних автоматів (для опису схеми зміни станів абонента та сервера SIP), базові методи теорії трансляторів (клієнт та сервер виступають інтерпретаторами мови протоколу SIP).

Наукова новизна одержаних результатів

- Запропоновано розробку інформаційної системи автоматизації роботи диспетчера таксі на основі алгоритму Дейкстри та за допомогою мови програмування PHP;
- *Удосконалено* алгоритм формування замовлення на основі результатів роботи алгоритму Дейкстри.
- *отримали подальший розвиток* розробка програмного та математичного забезпечення для автоматизації роботи диспетчера таксі.

Практичне значення одержаних результатів

Практичне значення одержаних результатів полягає в тому, що на основі запропонованих та удосконалених алгоритмів створено програмне та математичне забезпечення для автоматизації роботи диспетчера таксі, що дозволить зменшити час оформлення замовлення, та зменшить кількість використовуваного палива водіями таксі.

Особистий внесок

Усі результати викладені в дипломній роботі є самостійним досягненням автора. У друкованій праці, автору належать такі результати: - проведено аналіз існуючих інформаційних систем; - розглянуто основні особливості розробки програмних додатків; - проаналізовано можливі мови програмування для написання додатку.

Апробація результатів дослідження

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

Публікації

Здирко В. В. Метод формування замовлень на основі черг з пріоритетом / Василь Васильович Здирко // Сучасні комп'ютерні інформаційні технології / Василь Васильович Здирко. – Тернопіль, 2016. – С. 187.

АНОТАЦІЇ

Здирко В.В. Програмне та математичне забезпечення для автоматизації роботи диспетчера таксі.

Дипломний проект присвячений вирішенню актуального завдання по автоматичному оформленню замовлення, з урахуванням відстані від водія таксі до замовника на основі алгоритму Дейкстри, що підвищить ефективність роботи диспетчера таксі і зменшить витрати пального водіями таксі та час очікування замовлення клієнтом.

Метою дипломного проекту є розробка математичного та програмного забезпечення для автоматизації роботи диспетчера таксі.

З розвитком комп'ютерних інформаційних технологій з'являється багато нових можливостей для розширення бізнесу, для збільшення прибутків фірми та розширення клієнтської бази. Одним із способів впровадження інформаційних технологій у процес є створення власних web-сайтів. Це допоможе обслуговувати більше клієнтів і дозволить уникнути помилок через «людський фактор» (наприклад, диспетчера при прийомі замовлення).

Тому актуальною є оптимізація web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі.

Ключові слова: Class-based queueing, мова програмування PHP, оформлення замовлення, диспетчер таксі, алгоритм Дейкстри.

Здирко В.В. Программное и математическое обеспечение для автоматизации работы диспетчера такси.

Дипломный проект посвящен решению актуальной задачи по автоматическому оформлению заказа, с учетом расстояния от водителя такси к заказчику что повысит эффективность работы диспетчера такси и уменьшит расход топлива водителями такси. Для этого необходимо применить алгоритм Дейкстры для поиска кратчайшего пути.

Целью дипломного проекта является разработка математического и программного обеспечения для автоматизации работы диспетчера такси.

С развитием компьютерных информационных технологий возникает много новых возможностей для расширения бизнеса, для увеличения доходов фирмы и расширения клиентской базы. Одним из способов внедрения информационных технологий в процесс является создание собственных web-сайтов. Это поможет обслуживать больше клиентов и позволит избежать ошибок из-за «человеческого фактора» (например, диспетчер при приеме заказа).

Поэтому актуальной является разработка web-ориентированной информационной системы для автоматизации работы диспетчера такси.

Ключевые слова: Class-based queueing, язык программирования PHP, оформление заказа, диспетчер такси, алгоритм Дейкстры.

Zdyrko V. Taxi dispatcher work automatization with the help of Software and mathematical support.

The current Diploma project is dedicated to solving actual problems concerning automatic ordering, given the distance from the customer to the taxi driver which will increase the efficiency of the taxi dispatcher and reduce fuel consumption by taxi drivers. Applying Dijkstra's algorithm will help in finding the shortest path.

The aim of the diploma project is to develop mathematical and software for taxi dispatcher work automation.

With the development of computer information technology, there are many new opportunities for the creation of new business types, aimed to increase the profits of the company and expand its customer base. One of the methods of information technologies implementation is to create own web-sites. This will help serve more customers and will avoid errors due to "human factor" (eg dispatcher when taking the order).

That is why optimization of web-oriented information system for automation of dispatcher taxi is important.

Keywords: Class-based queueing, programming language PHP, ordering, taxi dispatcher, Dijkstra's algorithm

РОЗДІЛ 1

ОСОБЛИВОСТІ ФОРМУВАННЯ ЗАМОВЛЕННЯ ЗА ДОПОМОГОЮ АЛГОРИТМУ ПОШУКУ НАЙКОРОТШОГО ШЛЯХУ

1.1 Коротка характеристика об’єкту дослідження

З розвитком комп'ютерних-інформаційних технологій з'являється багато нових можливостей для розширення бізнесу, задля збільшення прибутків фірми та розширення клієнтської бази. Одним із способів впровадження інформаційних технологій у процес роботи підприємства є створення власних веб-сайтів. По-перше, це допоможе обслуговувати більше клієнтів, по-друге, дозволить уникнути помилок через «людський фактор».

Для служби таксі при створенні власного веб-сайту, особливо актуальною є реалізація функції «онлайн замовлення», за допомогою якої клієнт зможе зробити замовлення без участі диспетчера.

Розглянемо процес транспортних перевезень клієнтів службою таксі. Він здійснюється наступним чином: кожен водій, заступаючи на зміну, зв'язується з диспетчером, той в свою чергу, вносить його в список доступних водіїв. Водії спілкуються з диспетчером по радіо. Варто зазначити, що зазвичай клієнти здійснюють замовлення за допомогою дзвінка до диспетчера служби таксі. Диспетчер записує необхідні дані, та зв'язується по рації з доступними у даний момент водіями, один із яких приймає замовлення. Після виконання замовлення водій звітує перед диспетчером, який вносить його в список доступних водіїв. Після кожної зміни диспетчер формує звіт, який показує, скільки замовлень надійшло, скільки із них виконалось, скільки було скасовано, і їх загальну вартість. Ці звіти надходять до менеджера, на основі чого формується загальний звіт (зведений звіт) за певну дату чи місяць загалом.

Для оптимізації процесу прийому замовлень слід застосувати алгоритми для знаходження найкоротшого шляху від водія таксі до замовника, щоб зменшити кількість затраченого пального.

Аналіз процесу роботи середньостатистичної служби таксі показав, що основними напрямами її діяльності є:

- перевезення пасажирів по місту;
- міжміські перевезення пасажирів;
- перевезення вантажів по місту;
- міжміські перевезення вантажів ;
- зустріч пасажирів на ЖД вокзалах з табличкою;
- доставка кур'єром;
- обслуговування урочистих подій (надання авто VIP або бізнес класу).

Загальну схему організаційної структури служби таксі проілюстровано на рисунку 1.1.

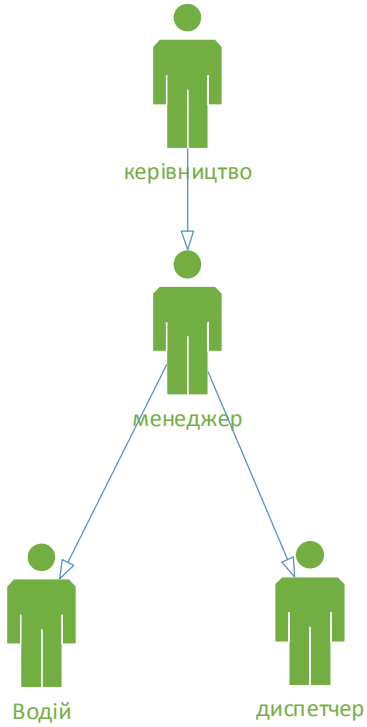

Рисунок 1.1 – Організаційна структура служби таксі

Розглянемо детальніше функціональні обов’язки кожної із ланок

організаційної структури служи таксі (рисунок 1.1).

Менеджер таксі виконує такі обов'язки :

- укладає договори про прийом на роботу;
- контролює виконання роботи диспетчерів;
- вирішую проблеми з замовниками, якщо вони виникли;
- звільняє працівників;
- формує звіт, який потім надає керівництву таксі;
- несе персональну відповідальність за виконання покладених на нього обов’язків;

Диспетчер таксі виконує такі обов'язки :

- приймає замовлення;
- заповняє дані про замовлення;
- віддає замовлення водіям;
- формує звіт за свою зміну.

Директор служби таксі є найвищою ланкою управління та виконує наступні обов’язки:

- без доручення діє від імені підприємства, представляє його інтереси в усіх установах незалежно від форми власності;
- укладає угоди;
- несе персональну відповідальність за виконання покладених на нього обов’язків;
- видає в межах компетенції накази по підприємству;
- розпоряджається коштами в межах кошторису витрат;
- відкриває рахунки в банках;
- приймає інші рішення по питаннях роботи підприємства, що не суперечать чинному законодавству України.

Проаналізувавши предметну область, виділено три основні бізнес-процеси, які необхідно буде автоматизувати за допомогою розроблюваної програмної системи:

- прийом заявки;
- обробка заявки;
- пошук оптимального вибору машини;
- формування звітів.

Прийом заявки – процес, що включає в себе отримання та збереження даних на сервері (рисунок 1.2).

Рисунок 1.2 – Дерево функцій бізнес-процесу «Прийом заявки»

Характеристика цього бізнес-процесу представлена в таблиці 1.1.

Таблиця 1.1

Характеристика бізнес-процесу «Прийом заявки»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Прийом заявки
Основні учасники	Сервер, браузер, клієнт
Вхідна подія	Відправлення даних про заявку
Вхідні документи	Немає
Вихідна подія	Немає
Вихідні документи	Немає

Продовження таблиці 1.1

Клієнт бізнес-процесу	Сервер

Обробка даних є процесом, що відбувається на сервері, в ньому формується відповідь клієнту і зберігаються дані (рисунок 1.3).

Рисунок 1.3 – Дерево функцій бізнес-процесу «Обробка даних»

Характеристика цього бізнес-процесу представлена в таблиці 1.2.

Таблиця 1.2

Характеристика бізнес-процесу «Обробка даних»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Обробка даних
Основні учасники	Сервер, браузер, користувач
Вхідна подія	Відправка даних форми
Вхідні документи	Немає
Вихідна подія	Збереження даних
Вихідні документи	Немає
Клієнт бізнес-процесу	Сервер

Пошук оптимального вибору машини є процесом, що відбувається на

сервері, в ньому відбувається пошук машини яка знаходится найближче до замовника (рисунок 1.4).

Рисунок 1.4 – Дерево функцій бізнес-процесу «Пошук оптимального вибору машиши»

Характеристика цього бізнес-процесу представлена в таблиці 1.3.

Таблиця 1.3

Характеристика бізнес-процесу «Пошук оптимального вибору машиши»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Пошук оптимального вибору машиши
Основні учасники	Сервер, браузер, користувач
Вхідна подія	Відправка даних форми
Вхідні документи	Немає
Вихідна подія	Вибір машини
Вихідні документи	Немає
Клієнт бізнес-процесу	Сервер

Наступним бізнес-процесом є формування звітів у доступному для користувача вигляді.

Рисунок 1.5 – Дерево функцій бізнес-процесу «Формування звітів»

Характеристика цього бізнес-процесу представлена в таблиці 1.4.

Таблиця 1.4

Характеристика бізнес-процесу «Формування звітів»

Назва характеристики	Значення характеристики
Ім’я бізнес-процесу	Формування звітів
Основні учасники	браузер, користувач
Вхідна подія	Запит на сервер
Вхідні документи	Немає
Вихідна подія	Звіт
Вихідні документи	Немає
Клієнт бізнес-процесу	Користувач

1.2 Процес пошуку найкоротшого шляху

Задача про найкоротший шлях полягає в знаходженні такого шляху між двома вершинами (або вузлами) графу, що сума ваг ребр з яких він складається мінімальна [13].

Завдяки великому спектру використання, алгоритми знаходження

найкоротшого шляху стрімко розвиваються. Ця задача є життєво необхідною адже використовується в різних сферах життя, починаючи від прокладення оптимального шляху між об’єктами (наприклад, найкоротший шлях від автомобіля до його кінцевої точки), в військових цілях та перевезеннях. Найкоротший шлях представлений як певний математичний об’єкт (графом). Існують два найбільш популярних і ефективних алгоритми знаходження оптимального шляху:

- алгоритм Дейкстри (Знаходить найкоротший шлях від однієї вершини графа до всіх інших вершин)[13];
- алгоритм Флойда-Воршелла (використовується для розв'язання задачі про найкоротший шлях у зваженому графі з додатними або від'ємними вагами ребр)[13].

Ці алгоритми мають найбільшу продуктивність при не великій кількості вершин у графі. При більшій кількості вершин завдання по пошуку найкоротшого шляху стає складнішим. Інформаційні технології підвищили можливості пошуку оптимальних шляхів адже вони дозволяють моделювати об’єкти, явища і процеси, незалежно від того чи є вони природні чи штучні. Кількість об’єктів постійно збільшується, а отже моделювання за допомогою природних матеріал стає не вигідним. Тому почали активно використовувати математику.

Математична модель - система математичних співвідношень, які описують досліджуваний процес або явище[13]. Реалізувати великі можливості математичного моделювання важко без потужної обчислювальної техніки (серверів або комп’ютерів). Завдяки великій популярності інформаційних технологій впроваджуються методи і засоби моделювання, які вирішують складні практичні завдання, такі як управління складними системами, прогнозуванню явищ, моделювання систем різного призначення, проектування.

Комп'ютерна модель є представленням об'єкту, системи чи поняття у

формі, відмінній від реальної, але наближеною до алгоритмичного опису, що включає і набір даних, що характеризують властивості системи та динаміку їх змін з часом[14]. Для створення комп'ютерної моделі необхідно вивчати та розуміти основні принципи програмування. Без неї комп'ютер не є корисним адже не може виконати поставлене перед ним завдання.

Розглянемо детальніше алгоритм Дейкстри.

Найефективніший алгоритм на графах відкритий нідерландським вченим Е. Дейкстром у 1959 році. Цей алгоритм знаходить найкоротшу відстань від однієї вершини графу до всіх решти. Алгоритм працює лише для графів у яких ребра не мають від'ємної ваги [13].

Приклад використання :

Необхідно знайти відстань від вершини 1 до інших вершин рисунок 1.6.

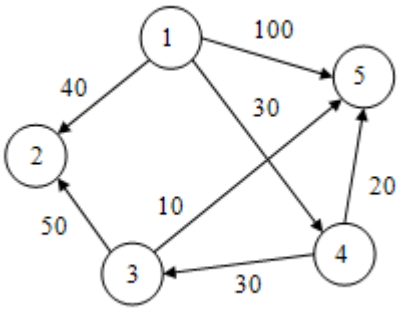

Рисунок 1.6 – Початковий граф

Від вершини 1 до вершин 2, 4, 5 існує орієнтоване ребро. Обчислимо мітки для даних вершин і занесемо їх в таблицю 1.5.

Таблиця 1.5

Номер вершини	Мітка	Статус мітки
1	[0,-]	Постійна
2	[40,1]	Тимчасова

Продовження таблиці 1.5

4	[30,1]	Тимчасова
5	[100,1]	Тимчасова

Дальше потрібно вибрати мітку відстань до якої є найменшою, відповідно це мітка [30,1] і вершина 4. Від вершини під номером 4 існує 2 орієнтованих ребра до вершини 3 і 5. Обчислимо мітки для даних вершин і занесемо їх в таблицю 1.5. Також необхідно змінити статус мітки 4 на постійний.

Таблиця 1.6

Номер вершини	Мітка	Статус мітки
1	[0,-]	Постійна
2	[40,1]	Тимчасова
4	[30,1]	Постійна
5	[30+20,4] = [50,4]	Тимчасова
3	[30+30,4] = [60,4]	Тимчасова

Аналогічно вибираємо мітку з найменшою відстанню, маркуємо її в постійну і обраховуємо знову відстані через постійні точки.

Таблиця 1.7

Номер вершини	Мітка	Статус мітки
1	[0,-]	Постійна
2	[40,1]	Постійна
4	[30,1]	Постійна

Продовження таблиці 1.7

5	[30+20,4] = [50,4]	Тимчасова
3	[30+30,4] = [60,4]	Тимчасова

Від вершини 5 не існує орієнтованих ребр тому таблиця залишається без змін, тільки статус цієї вершини міняється в постійний.

Таблиця 1.8

Номер вершини	Мітка	Статус мітки
1	[0,-]	Постійна
2	[40,1]	Постійна
4	[30,1]	Постійна
5	[30+20,4] = [50,4]	Постійна
3	[30+30,4] = [60,4]	Тимчасова

Оскільки залишилась лише вершина номер 3 з тимчасовим статусом і з неї можна лише потрапити на вершини 2 і 5 можна завершити обрахунок.

Результатом обрахунків є оптимальний шлях по ребрах графа зображений на рисунку 1.7.

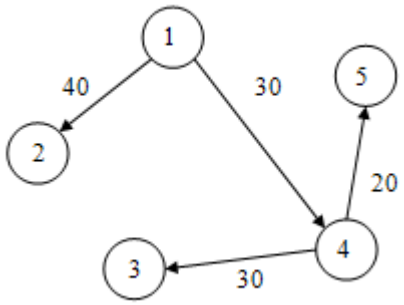

Рисунок 1.7 – Кінцевий граф

Розглянемо детальніше алгоритм Флойда-Воршелла.

Це алгоритм динамічного програмування для знаходження найкоротших відстаней між усіма вершинами важеного орієнтованого графа. Розроблений в 1962 році Робертом Флойдом і Стівеном Воршеллом [13].

Динамічне програмування – це альтернативне вирішення задач за допомогою жадний алгоритм або перебором. Використовується там, де оптимальне вирішення під задачі меншого розміру може бути використано для вирішення основної задачі. Робота алгоритму має такий вигляд:

1. Декомпозиція задачі на менші.

2. Рекурсивне вирішення під задач.

3. Обробка результатів виконання під задач для вирішення основної.

Завдяки цього задача стає модульною і її можна розпаралелювати, а отже обрахунок шляху відбувається паралельно.

Приклад

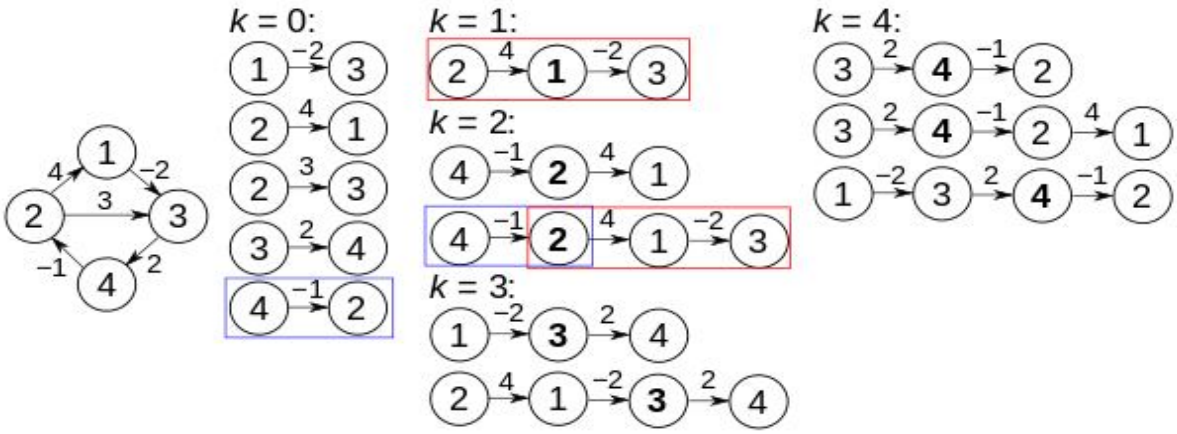

Рисунок 1.8 – Кінцевий граф

Перед першою ітерацією циклу k=0 і відомі шляхи відповідають одиночним ребрам у графі. Коли k=1, знайдено шляхи, яку проходять через вершину 1, зокрема: шлях 2→1→3 , замінить шлях 2→3, що проходить через меншу кількість ребер, але є довшим. При k=2, знаходяться шляхи, що проходять через вершини {1,2} . Червоні і голубі квадратики показують, як шлях 4→2→1→3 складається з 4→2 і 2→1→3, визначеними на попередніх ітераціях. Шлях 4→2→3 не розглядається, бо 2→1→3 поки що найкоротший

шлях. При k=3, знаходяться шляхи, що проходять через {1,2,3}. Нарешті, при k=4, знайдено всі найкоротші шляхи[13].

1.3 Порівняльна характеристика існуючих систем

Розглянемо наступні існуючі системи для обслуговування таксі:

- "Шанстаксі" (режим доступу: http://shans.te.ua/);
- " VIP таксі" (режим доступу: http://viptaxi.te.ua/);
- " Єдине таксі" (режим доступу: http://www.edynetaxi.te.ua/).

Усі перелічені системи розроблені з метою спрощення процесу прийому заявок.

У таблиці 1.9 наведено порівняльну характеристику розглянутих програмних продуктів (аналогів).

Таблиця 1.9

Порівняльна характеристика програмних продуктів

Фірма-розробник	Gryvik	vlasne.info	vlasne.info
Назва програмного продукту	VIP таксі	Шанстаксі	Єдине таксі
Версії продукту	Не вказано	Не вказано	Не вказано

Продовження таблиці 1.9

Функціональність	- замовлення таксі онлайн; - бюро знахідок; - перегляд вакансій; - перегляд послуг та тарифів; - перегляд новин - реєстрація	- перегляд новин; - онлайн замовлення; - тарифи; - реєстрація	- бюро знахідок; - онлайн замовлення; - перегляд інформації про місце знаходження таксі;
Інтерфейс користувача	Web-система	Web-система	Web-система
Допомога користувачу	Вкладка "Корисна інформація"	Немає	Немає

Розглянувши вище перелічені системи можна зробити наступні висновки: веб-сайт служби "Єдине таксі" містить досить цікаву функцію, а саме перегляд місця розташування таксі; системи "Шанс", як і "VIP таксі" мають функцію реєстрації, проте "VIP таксі" здійснюється тільки за допомогою дзвінка до диспетчера, який повинен вносити усі дані в систему, система "Єдинетаксі" немає функції реєстрації, а отже неможливо отримати дані про кількість клієнтів.

1.4 Застосування стратегії голубого океану

1. Ідентифікація власної галузі, товари замінники та альтернативні товари.

Транспортування — процес переміщення людей, вантажів, сигналів та інформації з одного місця в інше. В логістиці транспортування розглядається, як процес переміщення людей та вантажів, або перевезення.

Розглянемо процес транспортних перевезень клієнтів службою таксі. Він здійснюється наступним чином: кожен водій, заступаючи на зміну, зв'язується з диспетчером, той в свою чергу, вносить його в список доступних водіїв. Водії спілкуються з диспетчером по радіо. Варто зазначити, що зазвичай клієнти здійснюють замовлення за допомогою дзвінка до диспетчера служби таксі. Диспетчер записує необхідні дані, та зв'язується по рації з доступними у даний момент водіями, один із яких приймає замовлення. Після виконання замовлення водій звітує перед диспетчером, який вносить його в список доступних водіїв. Після кожної зміни диспетчер формує звіт, який показує, скільки замовлень надійшло, скільки із них виконалось, скільки було скасовано, і їх загальну вартість. Ці звіти надходять до менеджера, на основі чого формується загальний звіт (зведений звіт) за певну дату чи місяць загалом.

Альтернативні товари:

Пасажирські перевезення можуть виконуються також транспортом загального користування. Ці перевезення можуть відбуватися за допомогою такого транспорту як маршрутні автобуси, автобуси які були надані державою та підприємствами або громадянами які уклали договори на перевезення пасажирів.

Також транспортування може відбуватися за допомогою автомобілів таксі індивідуального користування які зазвичай знаходятся в місцях великого скупчення людей. До таких товарів можна віднести сервіси

компаній “Таксі шанс” та “Vip taxi”

2. Визначення ключових фактори, які впливають на вибір споживачів даної групи продуктів таблиця 1.9.

Споживачами даного типу продуктів є перш за все звичайні люди, які в більшості не мають технічної освіти. Отже освоїти даний продуктому повинен бути взмозі кожний користувач. Інтерфейс продукту повинен бути чітко зрозумілий та максимально швидким.

Таблиця 1.9

Фактори	Таксі шанс	Vip taxi
Ціна	5	5
Якість сервісу	5	6
Швидкість сервісу	6	8
Якість транспорту	5	5
Зручність використання	6	6

3. Стратегічна канва зображена на рисунку 1.9.

Рисунок 1.9 – Стратегічна канва

Головним диференціатором продукту Taxi є те, що даний продукт дасть можливість покращити роботу диспетчера таксі та збільшить кількість замовлень онлайн. Також при розробці даного продукту значна увага приділена функції пошуку водія який знаходится найближче до замовника, на цього водія буде потрапляти замовлення задля зменшення витрат пального.

4. Ідентифікація точок болю при користуванню сервісом

Перед:

- Вибір сервісу;

- Ціна;

- Оформлення;

Під час:

- Очікування замовлення;

- Неприємні враження від покупки;

По завершенню:

- Негативний відгук;

- Незадоволеність сервісом;

- Незадоволеність обслуговуванням;

5. Тенденції і попит

- розробка мобільних додатків;

- удосконалення перевірки і контролю замовлення;

- відміна замовлення;

- перегляд вільних водіїв і оформлення замовлення на найближчого з них;

6. Матриця змін таблиця 1.11.

Таблиця 1.11

Усунути	Зменшити
1)Оформлення замовлення за допомогою диспетчера 2)Очікування з'єднанням з диспетчером	1)Кількість використовуваних модулів 2)Негативні враження від оформлення замовлення
Створити	Збільшити
1)Алгоритм формування замовлення на основі черг з пріоритетом та пошуком найближчого водія	1)Кількість замовлень 2)Кількість користувачів

7. Таблиця цінностей та переваг

Таблиця 1.12

Feature	Benefit	Advantage
Онлайн новини	1.Користувач завжди може переглянути інформацію про акції та інші події.	1.Користувач завжди може переглянути інформацію про акції та інші події.
Онлайн оформлення замовлення	1.Користувач може оформити замовлення без з'єднанням з диспетчером.	1.Більше замовлень.
Адмін панель	1.Адміністратор може легко переглядати замовлення і стан сервісу.	1.Збільшення швидкодії сервісу.

Застосувавши стратегію голубого океану випливає, що розроблюваний

продукт повинен реалізувати основний мінімальний функціонал необхідний для системи прийому та оформлення замовлення в службі таксі, забезпечити зручне та надійне користування системою.

Висновки до першого розділу:

1. Розглянуто алгоритми для покращення формування заявки.
2. Проведено порівняльний аналіз трьох програмних систем аналогів: ”Шанс таксі”, ”Єдинетаксі”, ”Viртаксі”. На основі визначених переваг та недоліків розглянутих програмних систем аналогів, досліджено можливості використання досвіду провідних фірм-розробників програмних продутів.

3. Застосовану стратегію голубого океану для оцінки конкурентів

РОЗДІЛ 2
Проектування

2.1 Алгоритм пошуку найкоротшого шляху

Задача про знаходження найкоротшого шляху появляється при проектуванні і покращенні мереж транспорту, знаходженні ділянок з великою завантаженістю доріг. Задача ускладнюється додатково через те що потрібно врахувати спецефічні особливості такі як:

1) при пересуванні між двома точками може виникнути необхідність змінити маршрут;

2) ділянки доріг можуть буду заблоковані через ДТП;

3) ділянки доріг можуть буду заблоковані через ремонтні роботи;

4) навантаження певних ділянок під час години пік;

Завдання по знаходженню найкоротшого шляху відноситься до класу NP-повних. Вона може бути виконана лише шляхом повного перебору всіх можливих варіантів. Вся складність полягає в тому що при зміні будь-якій зміні маршруту всі найкоротші шляхи потріно повністю перерахувати. Отже щоб отримати приблизного розв’язку за допомогою перебору залежить від обчислювальної ефективності алгоритму знаходження найкоротшого шляху на мережі доріг. Алгоритм побудований на принципі розширення множини вершин графа транспортної мережі і додавання у нього додаткових ребер. Пошук найкоротшого шляхяху на графі на сьогоднішній день найкраще виконується алгоритмом Дейкстри, складність якого є полономіальною у найкращому випадку O + (log) n n m , де n – кількість вершин графа, а m – кількість його ребер. Якщо позначити як R наявну кількість маршрутів, а s – середню довжину маршруту, то розширений граф маршрутної мережі матиме n = R× s вершин. Це значення набагато перевищує кількість вершин транспортної мережі і швидко зростає зі збільшенням кількості маршрутів та

їх середньої довжини. Таким чином, реалізація алгоритму Дейкстри для розширеного графа транспортної мережі потребує значних обчислювальних потужностей.

Опис алгоритму.

Позначемо через u_i найкоротша відстань від вершини 1 до вершини *i*, через d_{ij} — довжину ребра *(i, j)*. Тоді для вершини *j* мітка $[u_j, i]$ визначається наступним чином:

$$[u_j, i] = [u_i + d_{ij}, i], d_{ij} \geq 0$$

Мітки в алгоритмі Дейкстри можуть бути двох типів: тимчасові або постійні. Тимчасова мітка може бути замінена на нову також тимчасову, якщо в процесі виконання алгоритму буде знайдено більш короткий маршрут до даної вершини. Якщо ж в процесі виконання алгоритму виявиться, що більш короткого маршруту від початкової до даної вершини не існує, то змінюємо статус тимчасової мітки на постійну.

Розв'язок за алгоритмом Дейкстри складається з наступних етапів:

1. Початковій вершині (вершина під номером 1) присвоюємо постійну мітку $[0, -]$ і покладаємо $i = 1$.
2. Для всіх вершин *j*, які поєднані орієнтованим ребром з вершиною *i*, і не мають міток, статус яких становить «постійна», обчислюємо тимчасові мітки $[u_i + d_{ij}, i]$. Якщо для вершина *j* вже існує деяка мітка $[u_j, k]$, то слід провірити виконання умови $u_i + d_{ij} < u_j$. Виконання даної умови говорить про те, що необхідно мітку $[u_j, k]$ замінити на $[u_i + d_{ij}, i]$.

Даний процес продовжуємо до тих пір, поки всім вершинам графа не буде присвоєно мітку зі статусом — постійна.

Приклад використання :

Є кілька способів реалізації даного рішення, а також їх доповнення. Деякі з них покращували продуктивність, а інші лише вказували на недоліки рішення Дейкстри, оскільки воно працює тільки з позитивно зваженими графами - такими, де немає негативних ваг у зв'язків. Для прикладу візьмемо

наступний позитивний граф і представимо його через список, де вкажемо зв'язку вузлів між собою:

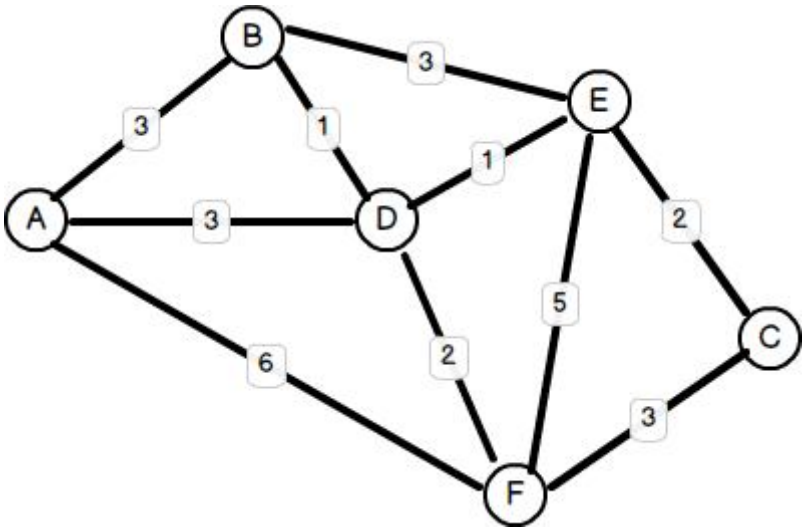

Рисунок 2.1 – Граф

Базова реалізація на мові php.

Ініціалізація графу

```
$graph = [
    'A' => ['B' => 3, 'D' => 3, 'F' => 6],
    'B' => ['A' => 3, 'D' => 1, 'E' => 3],
    'C' => ['E' => 2, 'F' => 3],
    'D' => ['A' => 3, 'B' => 1, 'E' => 1, 'F' => 2],
    'E' => ['B' => 3, 'C' => 2, 'D' => 1, 'F' => 5],
    'F' => ['A' => 6, 'C' => 3, 'D' => 2, 'E' => 5]
];
```

Клас алгоритму

```
class Algorithm
{
    protected $graph;

    public function __construct($graph) {
        $this->graph = $graph;
    }

    public function shortestPath($source, $target) {
        $shortestWay = [];
        $beforePoint = [];
        $queue = new SplPriorityQueue();
```

```
        foreach ($this->graph as $node => $adj) {
            $shortestWay[$node] = INF;
            $beforePoint[$node] = null;
            foreach ($adj as $w => $distance) {
                $queue->insert($w, $distance);
            }
        }

        $shortestWay[$source] = 0;
        while (!$queue->isEmpty()) {

            $minimumDistance = $queue->extract();
            if (!empty($this->graph[$minimumDistance])) {
                foreach ($this->graph[$minimumDistance] as $node => $distance) {
                    $alt = $shortestWay[$minimumDistance] + $distance;
                    if ($alt < $shortestWay[$node]) {
                        $shortestWay[$node] = $alt;
                        $beforePoint[$node] = $minimumDistance;
                    }
                }
            }
        }

        $stack = new SplStack();
        $minimumDistance = $target;
        $dist = 0;

        while (isset($beforePoint[$minimumDistance]) &&
$beforePoint[$minimumDistance]) {
            $stack->push($minimumDistance);
            $dist += $this-
>graph[$minimumDistance][$beforePoint[$minimumDistance]];
            $minimumDistance = $beforePoint[$minimumDistance];
        }

        if ($stack->isEmpty()) {
            echo "Нема шляху з $source в $targetn";
        } else {
            $stack->push($source);
            echo "$dist:";
            $sep = '';
            foreach ($stack as $node) {
                echo $sep, $node;
                $sep = '->';
            }
```

```
            echo "\n";
        }
    }
}
```

Виклик алгоритму

```
$algorithm = new Algorithm($graph);
$algorithm->shortestPath('D', 'C');  // 3:D->E->C
```

Обчислювальна ефективність алгоритму Дейкстри.

1. Даний алгоритм добре розкриває свій потенціал при багаторазовому використанні, адже під час кожного розрахунку він додатково обчислює відстані до інших. У прикладі, при пошуку найкоротшого шляху між вершинами D та C, додатково обраховуються відстані від вершини D до всіх інших вершин графу. Це дозволить заощадити час при повторному перерахунку відстаней від точки D до інших вершин графу.

2. Для реалізації алгоритму необхідно зберігати і сортування у порядку зростання оцінок всіх нерозгалужених вершин (вершин які мають лише одне ребро) в процесі розв’язку. Для зберігання даних найкраще підійде бінарна купа адже вона виконує функції запису і вичитки за за логарифмічний час O n (log). При обрахунку кожна розгалужена вершина в купу не записується, це дозволяє зменшити час визначення найкоротшого шляху.

3. Алгоритм можна змінити практично для всіх додаткових умов які необхідні для отримання найкращого результату. Наприклад, заборонити побудову маршруту через ребро яке заблоковане.

2.2 Розробка архітектури програмної системи

Клієнт-серверна архітектура – тип архітектури програмної системи, в якій обробка даних ділиться на дві на сторони: клієнта і сервера. Клієнт в ній використовується для відправки запитів на сервер та отримання даних від нього. Це означа, що клієнт встановлює зв'язок з сервером, за допомогою HTTP протоколу, та обмінюється з ним інформаційними пакетами.

Архітектура системи зображена на рисунку 2.2.

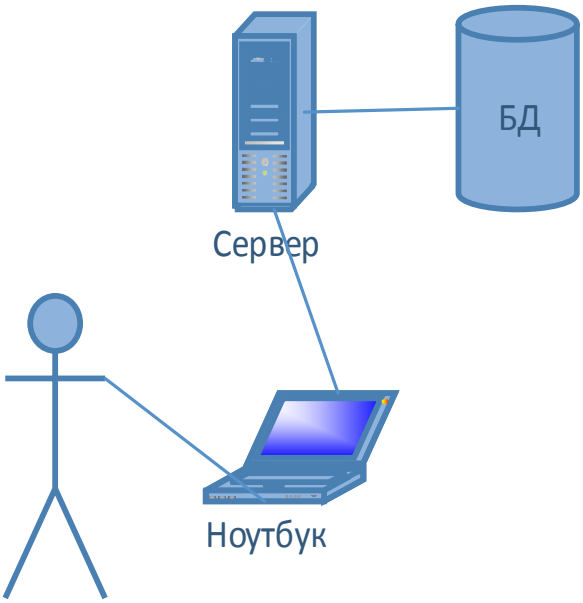

Рисунок 2.2 – Архітектура «клієнт-сервер»

Далі проведемо об’єктно-орієнтовний аналіз предметної області та побудуємо діаграми класів проекту. На рисунку 2.3 проілюстровано діаграму класів рівня сайту.

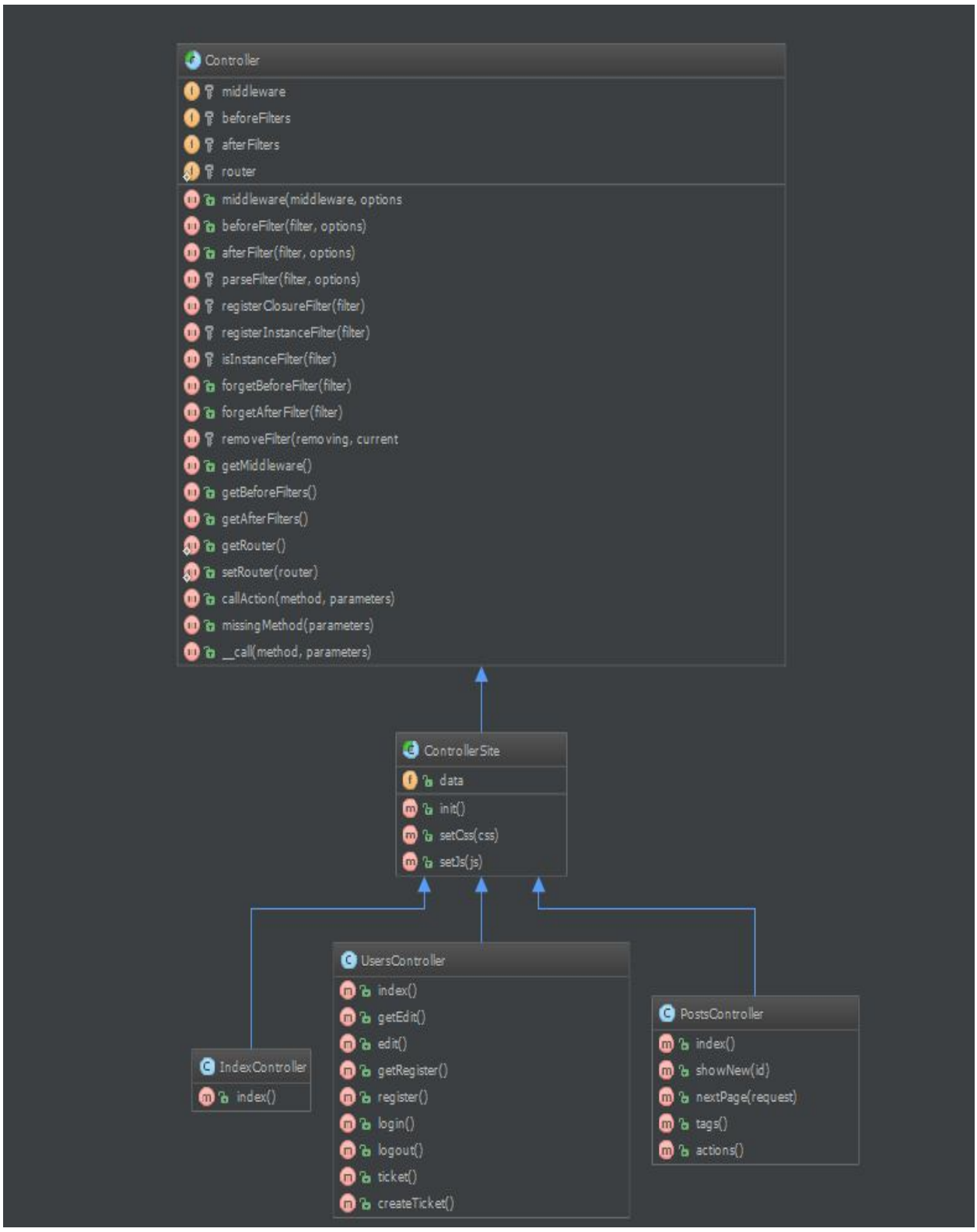

Рисунок 2.3 – Діаграма класів рівня сайту

Розглянемо детальніше Клас ControllerSite із рисунку 3.2.

Клас ControllerSite – базовий всіх класів, які будуть використовуватися при розробці системи. Атрибут data призначений для зберігання даних про css і js файли. Функція init() –оголошує css і js файли. Також є встановлювачі

setCss() і setJs() – вони призначені для розширення даних про css і js файли.
Клас IndexController – має такі функції :
index() – генерує сторінку на якій виводяться головна сторінка.

Розглянемо детальніше Клас IndexController із рисунку 3.2. Клас IndexController – має такі функції :
index() – генерує сторінку на якій виводяться інформація користувача .
getEdit() – генерує сторінку з полями для редагування інформації користувача.
edit()–зберігає відредаговані дані користувача.
getRegister ()–генерує сторінку реєстрації користувача.
ticket()–генерує сторінку оформлення замовлення.
login()–вхід користувача.
logout()–вихід користувача.
createTicket()– створює замовлення.
register()–реєструє користувача.

Далі, розглянемо детальніше Клас PostController із рисунку 2.3, він містить наступні методи:
index() – генерує сторінку на якій виводяться новини.
showNew()– генерує сторінку з інформацією про новину за id.
nextPage() –генерує новини за номером пагінації.
actions() – генерує сторінку акціями.

На рисунку 2.4 проілюстровано діаграму класів представницького рівня адмін-панелі.

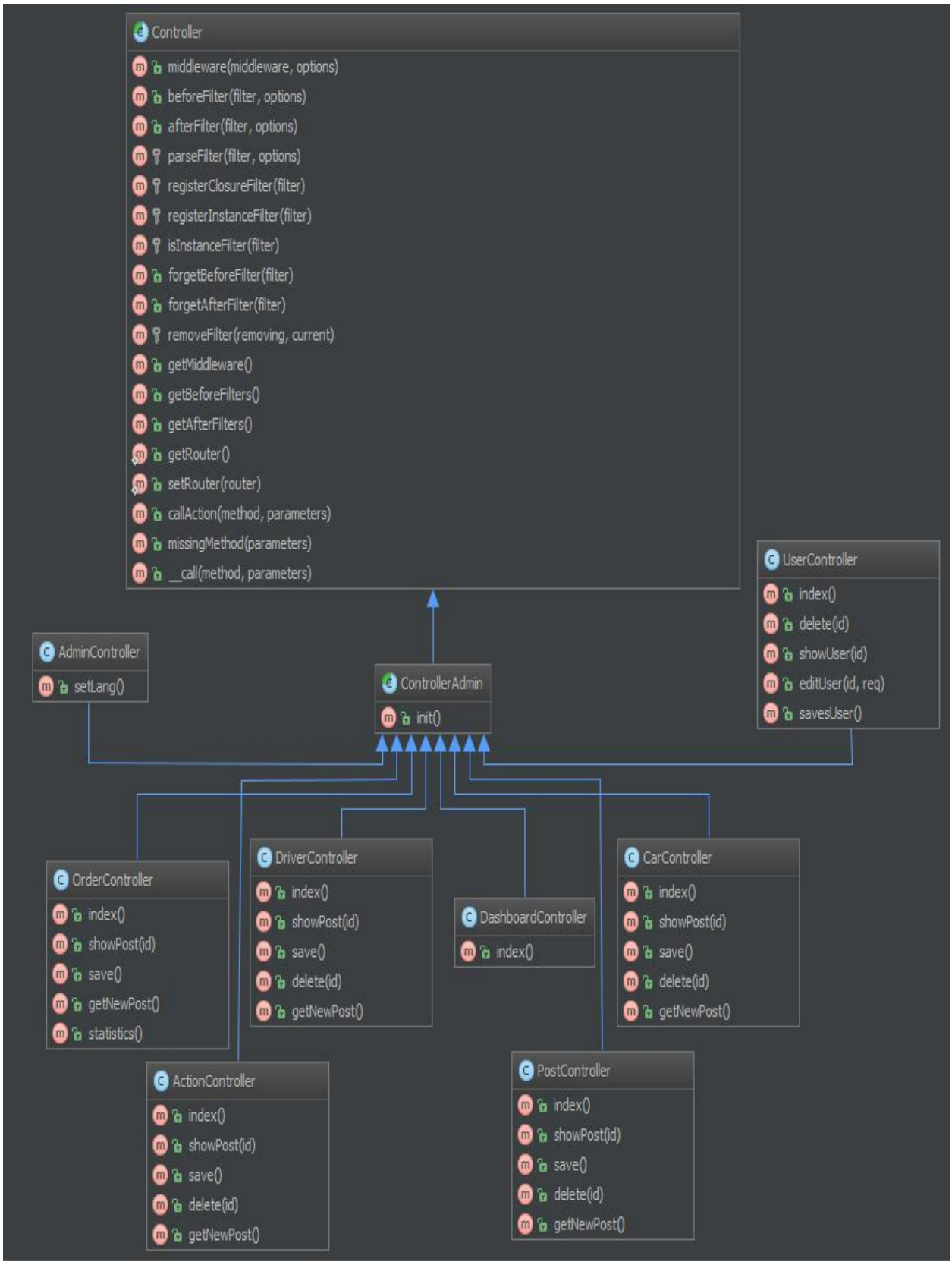

Рисунок 2.4 – Діаграма класів рівня адмін-панелі

Розглянемо детальніше Клас ControllerAdmin із рисунку 2.4.

Клас ControllerAdmin буде виконувати роль базового класу для усіх класів, які будуть використовуватися при розробці системи. Він має атрибут data, що задає масив змінних які будуть зберігати послання на css та js файли. Функція init() – ініціалізує css та js файли, які будуть використовуватися, для формування HTML коду сторінки. Методи setCss() і setJs() – дають змогу доповнювати перелік css і js файлів в конкретному класі.

Розглянемо детальніше Клас AuthController із рисунку 2.4.

Клас AuthController – містить в собі змінні auth і registrar,вони вміщують в собі об’єкти класів фреймворку.

Клас OrderController – має такі функції (рисунок 2.4):

index() – генерує сторінку на якій виводяться всі замовлення.

showPost() – генерує замовлення по id.

save() – зберігає дані про замовлення.

delete() – видаляє замовлення по id.

getNewPost() – генерує сторінку заповнення нового замовлення.

Далі, розглянемо детальніше методи класу UserController із рисунку 3.3:

index() – генерує сторінку на якій виводяться всі користувачі.

showUser () – генерує сторінку з інформацію користувача по id.

savesUser() – зберігає дані про користувача.

editUser() – оновлює дані користувача по id.

delete() – видаляє дані користувача по id.

Розглянемо детальніше Клас ActionController із рисунку 2.4. Клас ActionController– має такі функції:

index() – генерує сторінку, на якій виводяться всі акції.

showPost() – генерує сторінку з інформацією про акцію по id.

save() – зберігає дані акцію.

delete() – видаляє акцію по id.

getNewPost() –генерує сторінку додавання нової акції.

Розглянемо детальніше Клас DriverController із рисунку 2.4.

Клас DriverController– має такі функції :

index() – генерує сторінку на якій виводяться всі водії.

showPost() – генерує сторінку з інформацією про водія по id.

save() – зберігає дані водія.

delete() – видаляє водія по id.

getNewPost() –вертає сторінку додавання нового водія.

Далі, розглянемо детальніше Клас CarController із рисунуку 2.4. Клас CarController– має такі функції :

index() – генерує сторінку на якій виводяться всіводії.

showPost() – генерує сторінку з інформацією про машину по id.

save() – зберігає дані про машину.

delete() – видаляє дані машини по id.

getNewPost() –генерує сторінку додавання нової машини.

Наступниним розглянемо Клас PostController із рисунуку 2.4, він містить наступні методи:

index() – генерує сторінку на якій виводяться всі новини.

showPost() – генерує сторінку з інформацією про новину по id.

save() – зберігає дані про новину.

delete() – видаляє дані про новину по id.

getNewPost() –генерує сторінку додавання нової новини.

Для кращого розуміння логіки роботи програмної системи для прийому заявок таксі, наведемо діаграми діяльності основних варіантів використання розроблюваної програмної системи.

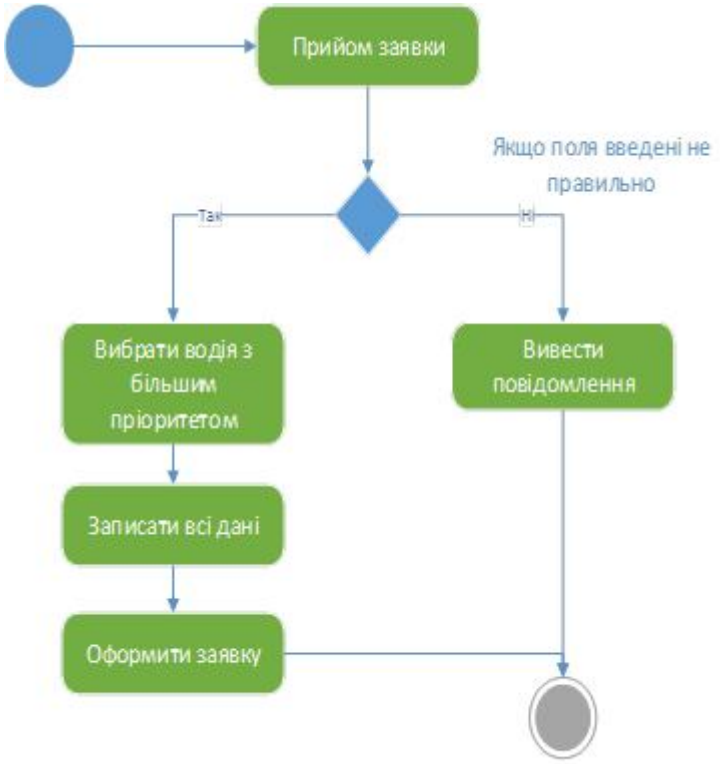

Рисунок 2.5 – Діграма діяльності варіанту використання «Прийом заявки»

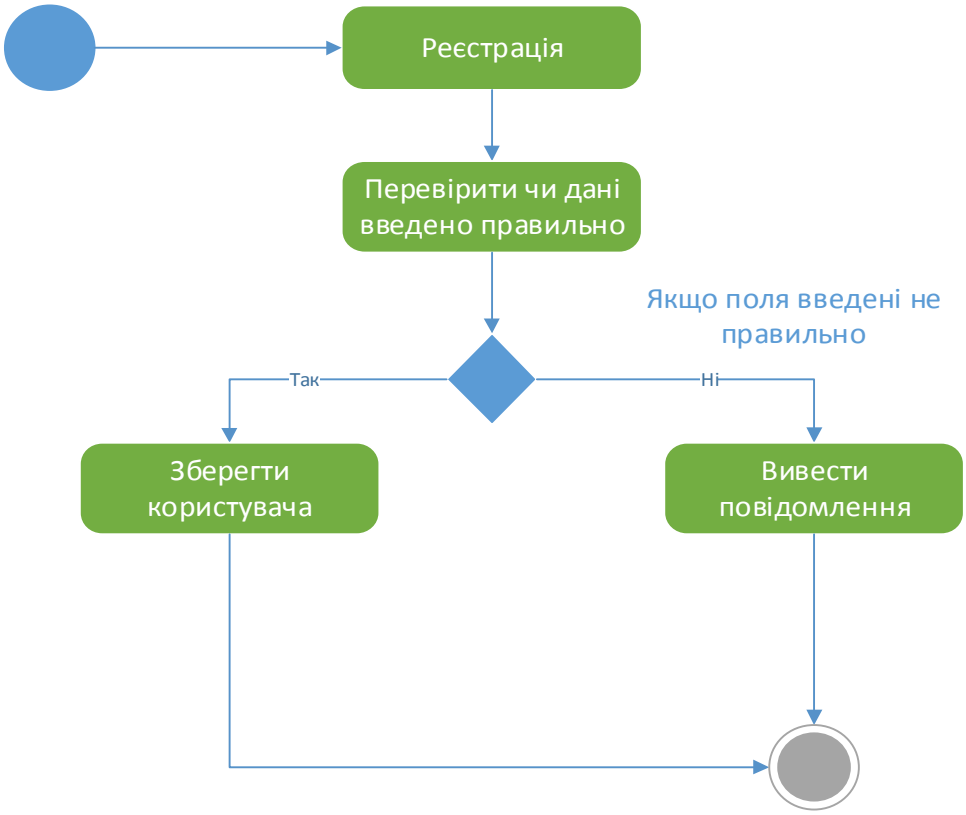

Рисунок 2.6 – Діграма діяльності варіанту використання «Реєстрація»

Рисунок 2.7 – Діграма діяльності варіанту використання «Перегляд статистика»

2.3. Проектування структури бази даних

Згідно вербального опису розроблюваної системи наведеного у першому розділі дипломної роботи, було створено діаграму елементів і зв’язків системи прийому заявок таксі, показану на рисунку 2.8.

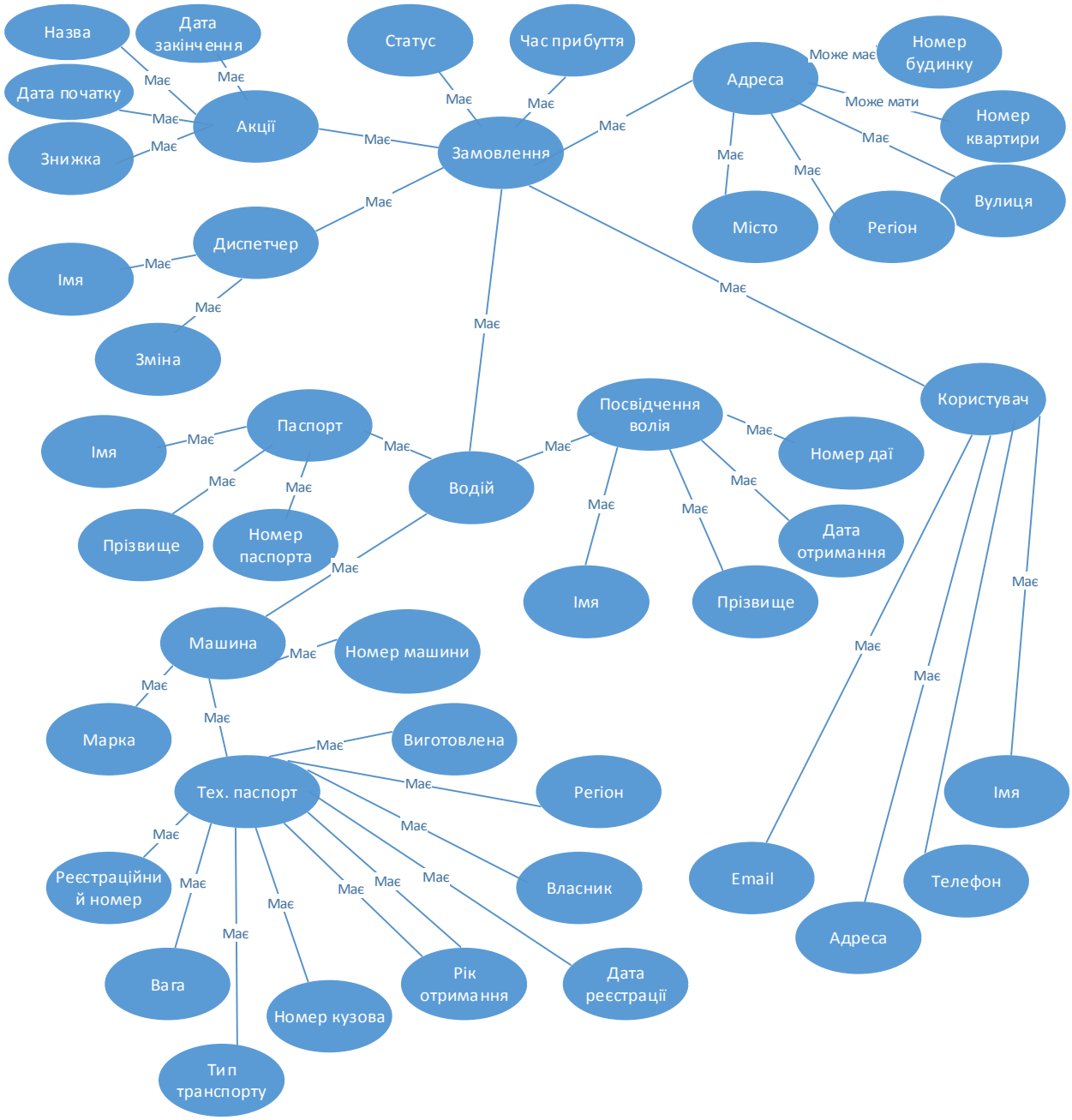

Рисунок 2.8 – Діаграма елементів і зв’язків

На основі діаграми елементів і зв’язків було побудовано ER діаграму. Варто зазначити, що ER діаграм відображає модель взаємозвязку даних [10].

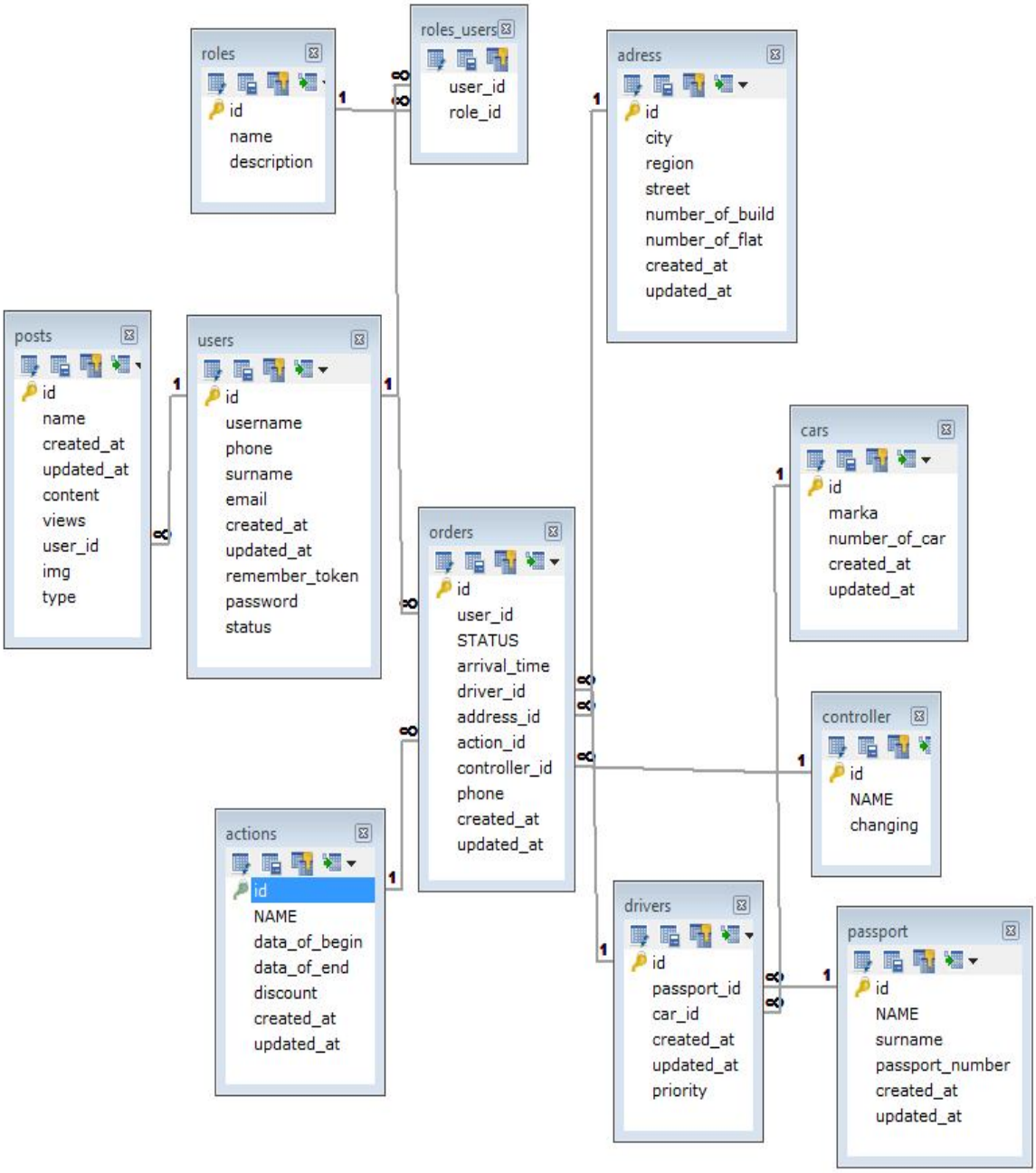

Рисунок 2.9 – ER діаграма бази даних

Відношення, проілюстровані на рисунку 2.9, призначені для зберігання даних системи прийому заявок таксі. Далі, наведемо таблицю ідентифікаторів, яка містить дані про назву елементу та його ідентифікатор.

Таблиця 2.1

Ідентифікатори атрибутів відношень

№ п/п.	Вхідний елемент	Ідентифікатор
1.	Назва	Name
2.	Дата початку	Date_of_begin
3.	Знижка	Discount
4.	Дата закінчення	Date_of_end
5.	Акції	Actions
6.	Замовлення	Orders
7.	Статус	Status
8.	Час прибуття	Arrival_time
9.	Адрес	Address
10.	Місто	City
11.	Регіон	Region
12.	Вулиця	Street
13.	Номер квартири	Number_of_flat
14.	Номер будинку	Number_of_build
15.	Ім’я	Name
16.	Прізвище	Surname
17.	Водії	Drivers
18.	Користувачі	Users
19.	Телефон	Phone
20.	Email	Email
21.	Диспетчер	Controller
22.	Змінна	Changing
23.	Номер паспорта	Passport_number
24.	Машини	Cars
25.	Марка	Marka
26.	Номер Машини	Number_of_car
27.	Вага	Weight
28.	Перегляди	Views
29.	Зображення	Img
30.	Наповнення	Content

Продовження таблиці 2.1

31.	Створено	Created_at
32.	Оновлено	Updated_at

Далі, наведемо таблицю індексів, яка містить дані про назву атрибуту, назву таблиці якій належить цей атрибут та його опис.

Таблиця 2.2

Індекси

Назва таблиці	Атрибут	Опис
Orders	Address_id	Зовнішній ключ
Orders	Driver_id	Зовнішній ключ
Orders	User_id	Зовнішній ключ
Orders	Action_id	Зовнішній ключ
Orders	Controller_id	Зовнішній ключ
Drivers	Passport_id	Зовнішній ключ
Drivers	Car_id	Зовнішній ключ
Posts	User_id	Зовнішній ключ
Posts	User_id	Зовнішній ключ
Roler_Users	User_id	Зовнішній ключ
Roler_Users	Role_id	Зовнішній ключ

Далі, наведемо таблицю типів, яка містить дані про назву вхідного елементу, його тип та розмірність.

Таблиця 2.3

Типи атрибутів відношень

№ п/п.	Вхідний елемент	Тип	Розмірність
1.	Назва	varchar	25
2.	Дата початку	date	
3.	Знижка	float	8
4.	Дата закінчення	date	
5.	Статус	varchar	10
6.	Час прибуття	date	
7.	Адрес	varchar	20
8.	Місто	varchar	20
9.	Регіон	varchar	20
10.	Вулиця	varchar	20
11.	Номер квартири	int	4
12.	Номер будинку	varchar	10
13.	Ім’я	varchar	25
14.	Прізвище	varchar	25
15.	Дата отримання	date	
16.	Телефон	varchar	20
17.	Email	varchar	30
18.	Номер паспорта	varchar	20
19.	Марка	varchar	20
20.	Номер Машини	varchar	20
21.	Наповнення	text	
22.	Картинка	varchar	20
23.	Створено	varchar	20
24.	Оновлено	varchar	20

Висновки до другого розділу:

1. Розроблено діаграму елементів і зв’язків, на якій зображені всі елементи бази даних.
2. Розроблено таблиці індетифікаторів, типів та індексі. У таблиці типів описано назви атрибутів усіх відношень, їх тип та розмірність. Таблиця індексів містить дані про назву атрибуту, назву відношення, якому належить цей атрибут та його опис.Таблиця індетифікаторів містить дані про назву елементу та його ідентифікатор.
3. Розроблено ER-діаграму та зпроектовано структуру бази даних web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі .

РОЗДІЛ 3
РЕАЛІЗАЦІЯ СИСТЕМИ

3.1 Програмна реалізація проекту

PHP — мова, функції якої використовують при формуванні html-коду сторіноки, які обробляються PHP-інтерпретатором. Велика різноманітність функцій PHP дозволяє уникнути написання багатострічкових власних функцій, сам інтерпритатом побудований на основі мови C [1].

PHP — мова, яка обєднує переваги Perl і C і орієнтована на роботу в Інтернеті. І хоча PHP є досить молодою мовою, вона є популярною серед web-програмістів і є найпопулярнішою для створення веб-застосунків (скриптів) [2].

Laravel є вільним, відкритим фреймворком PHP, призначений для розробки веб-додатків наступного за Model-View-Controller (MVC). Особливості Laravel включають його виразний синтаксис, модульну систему упаковки з менеджером виділеної залежності, різні способи для доступу до реляційних баз даних, і різні утиліти, які допомагають у розгортанні додатків та їх обслуговуванні.

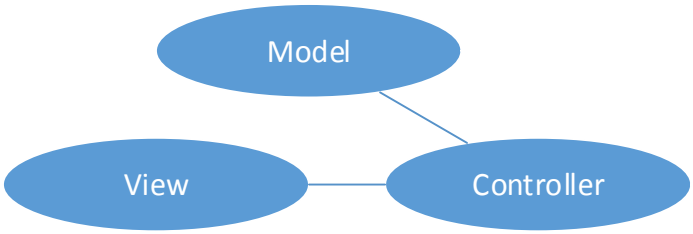

Рисунок 3.1 –Архітектура MVC

Приклад програмної реалізації контролера, для перегляду новин, наведено нижще:

```
<?php namespace App\Http\Controllers\Site;
use App\Http\Controllers\ControllerSite;
use App\Models\Posts;
```

```
use App\Models\Action;
use Illuminate\Routing\Route;
use Illuminate\Support\Facades\Request;
use Input;

class PostsController extends ControllerSite {
            public function index()
            {
        $this->init();
        $confValue = 6;
        $count = Posts::all()->count();
        $posts =  Posts::where('type','=','post')->orderBy('created_at','desc')->take($confValue)->get();
                        return view('site/posts/posts')->with(array(
          'data' => $this->data,
          'posts' => $posts,
          'pagination' => ceil($count/$confValue)
        ));
            }
    public  function  showNew($id){
        $post =  Posts::find($id);
        $views = $post->views;
        $post->views = $views +1;
        $post->save();
        return view('site/posts/post')->with(array(
          'data' => $this->data,
          'post' => $post
        ));
    }
    public  function  nextPage(Request $request){
        $confValue = 6;
        $data = Input::all();
        $posts = Posts::where('type','=','post')->orderBy('created_at','desc')->skip($confValue*($data['page']-1))-
>take($confValue)->get();
        return view('site/posts/pagination')->with(array(
             'posts' => $posts,
          ));
    }
    }
}
```

Далі, наведемо короткий опис усіх функцій, які фігурують у наведеній частині коду програми:

function index () – генерує код сторінки з новинами;

function showNew () – вертає код сторінки з новининою використовуючи ідентифікатор новини;

function nextPage () – генерує сторінку з наступними шістьма новинами.

Повний лістинг коду програмної системи наведено в додатку Б. Для реалізації функцій входу в системи та роботою з модальними вікнами програмної системи було використано JavaScript.

JavaScript(JS) — динамічна, об'єктно-орієнтована мова програмування. Реалізація стандарту ECMAScript. Найчастіше використовується на стороні браузера, що надає можливість коду на стороні клієнта, взаємодіяти з користувачем, керувати браузером, асинхронно обмінюватися даними з сервером, змінювати структуру та зовнішній вигляд веб-сторінки. Bootstrap є найпопулярнішим фреймворком для створення гнучких веб – сайтів, щостало основною причиною його використання у цьому дипломному проекті.

Приклад реалізації кілької функцій програмної системи на JavaScript наведено нижще.

```
$(document).ready(function () {
   var modal;
   $.ajaxSetup({
      headers: {
         'X-CSRF-TOKEN': $('meta[name="csrf-token"]').attr('content')
      }
   });
   $(document).on('click','.next_page',function(){
      var active = $(this).closest('ul').find('.active');
      if(!$(active).next().hasClass('next_page'))
         $(active).next().find('a').trigger('click');
   });
   $(document).on('click','#login',function(){
    modal =  $('#modal_login').modal();
   });
   $(document).on('click','#subm',function(){
      var request_data = {
        email : $('#login_au').val(),
        password : $('#pass_au').val()
      };
```

```
    $.ajax({
        url : '/user/login',
        type : "POST",
        dataType: "JSON",
        data : request_data,
        success : function(response){
        }
    });
    setTimeout(function(){window.location.reload()},3000);
});
        }
});});
```

Далі, наведемо короткий опис усіх функцій, які використовуються у наведеному скрипті:

$.ajaxSetup – встановлює токен для ajax запитів;

$(document).on('click','#login',function(){}) – відкриває модальне вікно для входу в систему;

$(document).on('click','#subm',function(){}) – відправляє дані для входу по натиску кнопки.

Для продуктивного функціонування програмної системи для прийому заявок таксі, висуваються наступні вимоги до апаратного забезпечення.

Системні вимоги до сервера:

1. Оперативна пам’ять - 2Гб.
2. Об’єм дискового простору - 1Гб.
3. Операційна система - OS Windows XP і вище.
4. MySQL.
5. Sqlyog community edition.
6. Apache.

Системні вимоги до клієнта:

1. Браузер: Firefox 37 версії, Google chrome 43 версії .

3.2 Програмна реалізація бази даних

База даних веб-орієнтованої програмної системи для прийому заявок таксі реалізовано із використанням СКБД MySQL версії 5.6. MySQL є найкращим рішенням для проектів такого типу, адже має ряд переваг:

- простота у встановленні та використанні;
- підтримується необмежена кількість користувачів, що одночасно працюють із БД;
- висока швидкість виконання команд;
- наявність простої і ефективної системи безпеки.

Першим кроком у програмній реалізації бази даних є її створення. Далі, наведено DDL-код створення бази даних:

```
CREATE DATABASE /*!32312 IF NOT EXISTS*/`back` /*!40100 DEFAULT
```

Після цього, створюємо відношення у базі даних веб-орінтованої програмної системи для прийому заявок таксі, які були спроектовані та показані на рисунку 2.8 (ERD). DDL-код створення відношень:

- Відношення drivers:

```
DROP TABLE IF EXISTS `drivers`;
CREATE TABLE `drivers` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
  `passport_id` int(11) NOT NULL,
  `car_id` int(11) NOT NULL,
  `created_at` datetime DEFAULT NULL,
  `updated_at` datetime DEFAULT NULL,
  `priority` int(10) DEFAULT '0',
  PRIMARY KEY (`id`),
  KEY `passport_id` (`passport_id`),
  KEY `car_id` (`car_id`),
  CONSTRAINT `drivers_ibfk_1` FOREIGN KEY (`passport_id`) REFERENCES `passport` (`id`),
  CONSTRAINT `drivers_ibfk_2` FOREIGN KEY (`car_id`) REFERENCES `cars` (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

- Відношення cars:

```
 CREATE TABLE `cars` (
  `id` int(10) NOT NULL AUTO_INCREMENT,
```

```
  `marka` varchar(10) COLLATE utf8_bin NOT NULL,
  `number_of_car` varchar(20) COLLATE utf8_bin NOT NULL,
  `created_at` datetime DEFAULT NULL,
  `updated_at` datetime DEFAULT NULL,
  PRIMARY KEY (`id`)
) ENGINE=InnoDB AUTO_INCREMENT=3 DEFAULT CHARSET=utf8 COLLATE=utf8_bin;
```

Завершальним етапом програмної реалізації бази даних веб-орінтованої програмної системи для прийому заявок таксі є додавання кортежів у створені вище відношення.

Приклад, DDL-коду для додавання одного кортежу у відношення «Cars»:

```
insert into `cars`(`id`,`marka`,`number_of_car`,`created_at`,`updated_at`) values (1,'nissan','9945',NULL,NULL);
```

Приклад, DDL-коду для додавання одного кортежу у відношення «Drivers»:

```
insert into `drivers`(`id`,`passport_id`,`car_id`,`created_at`,`updated_at`,`priority`) values (1,1,1,NULL,NULL,0);
```

Повний DDL-код для сворення бази даних наведено в додатку В.

3.3 Тестування

Тестування — це процес дослідження системи, призначений для перевірки якості продукту керуючись документацією і вимогами [5].

У роботі було проведено GUI та USABILITY тестування програмної системи, яке виконувалося з метою перевірки зручності розроблюваної системи у процесі її використання. Для проведення зазначеного типу тестування було залучено користувачів системи у ролі тестувальників, далі було детально проаналізовано отримані результати.

Аналіз отриманих результатів GUI та USABILITY тестування розробленої програмної системи, показав, що:

- ✓ зручність використання розроблюваної системи для пересічного користувача знаходиться на високому рівні;

- ✓ дизайн розробленої програмної системи є простим та зрозумілим для пересічного користувача;
- ✓ розроблена веб-система є ефективною, так як вона забезпечує результативність для кінцевого користувача, і при цьому не вимагає виконання надлишкових дій чи операцій;
- ✓ розроблена система є продуктивною, тому що швидкість завантаження сторінок користувацького інтерфейсу не перевищує 0.2 сек;
- ✓ користувацький інтерфейс системи забезпечує для пересічного користувача можливість роботи з нею на інтуїтивному зрозумілому рівні;
- ✓ розроблена система характеризується цілісністю інтерфейсу, що явно сприяє кращому розумінню пересічного користувача.

Наступним етапом, було проведено функціональне тестування web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі. Результати функціонального тестування за варіантами використання наведено у таблиці 3.1.

Таблиця 3.1

Функціональні тестові випадки

Варіант використання	Тестові випадки	Тестові дані
Вхід	4	12
Реєстрація	1	25
Перегляд звітів	3	43
Перегляд замовлень	3	34
Перегляд користувачів	3	24
Перегляд машин	4	11
Перегляд водіїв	1	15
Перегляд новин	2	17

Продовження таблиці 3.1

Перегляд акцій	1	42
Зміна паролю	4	31
Зробити замовлення	2	42
Загалом	48	296

47 із 48 розроблених тестових випадків наведених у таблиці 4.1 (295 із 296 множин тестових даних пройшли) пройшли успішно, отже функціональне тестування розглядається як частково успішне, адже 97,9166% тестових випадків пройшли, 99,6621 % наборів тестових даних пройшли.Однак при виконанні одного тестового випадку не пройшли функціональне тестування один набір тестових даних, через дефект програмної системи допущений у процесі її розробки, розглянемо його детальніше.

Дефект

Коли користувач входить в систему, то його запис показується тільки після перезавантаження сторінки.

Дефект полягає в тому, що коли користувач входить в систему, то його запис показується тільки після перезавантаження сторінки.

Опис

Відповідно до тестового випадку «Вхід» обліковий запис має показуватися відразу.

Подолання дефекту

Цей дефект можливо виправити, використовуючи Jquery бібліотеку, яка асинхронно відправляє запити на сервер і логінить користувача.

Варто зазначити, що після виправлення відповідних помилок в коді програми усі тестові дані пройшли. Отже, розроблена програмна система показала хороші показники якості під час функціонального тестування. Тест-кейси наведено в додатку Г.

3.4 Інструкція користувача

Щоб встановити сервер Apache і MySQL я використовую збірку XAMPP, яка безкоштовно скачується і швидко розгортається. Під час встановлення цієї збірки повинно бути відмічено обов'язково два компонента, а саме Apache сервер і MySQL сервер. Після завершення встановлення в панелі керування потрібно запустити ці два сервера. За допомогою Sqlyog потрібно розгорнути базу даних. Та закинути файли проекту в папку за адресою «xampp/htdocs» яка знаходится на диску С. Відкрити браузер та перейти по адресі «localhost».

Копія екранної форми із головною сторінка веб-сайту таксі проілюстрована на рисунку 3.2. На головній сторінці веб-сайту розміщено наступні пункти меню із посиланнями на відповідні веб-сторінки: головна, новини, онлайн замовлення, акції, про проект.

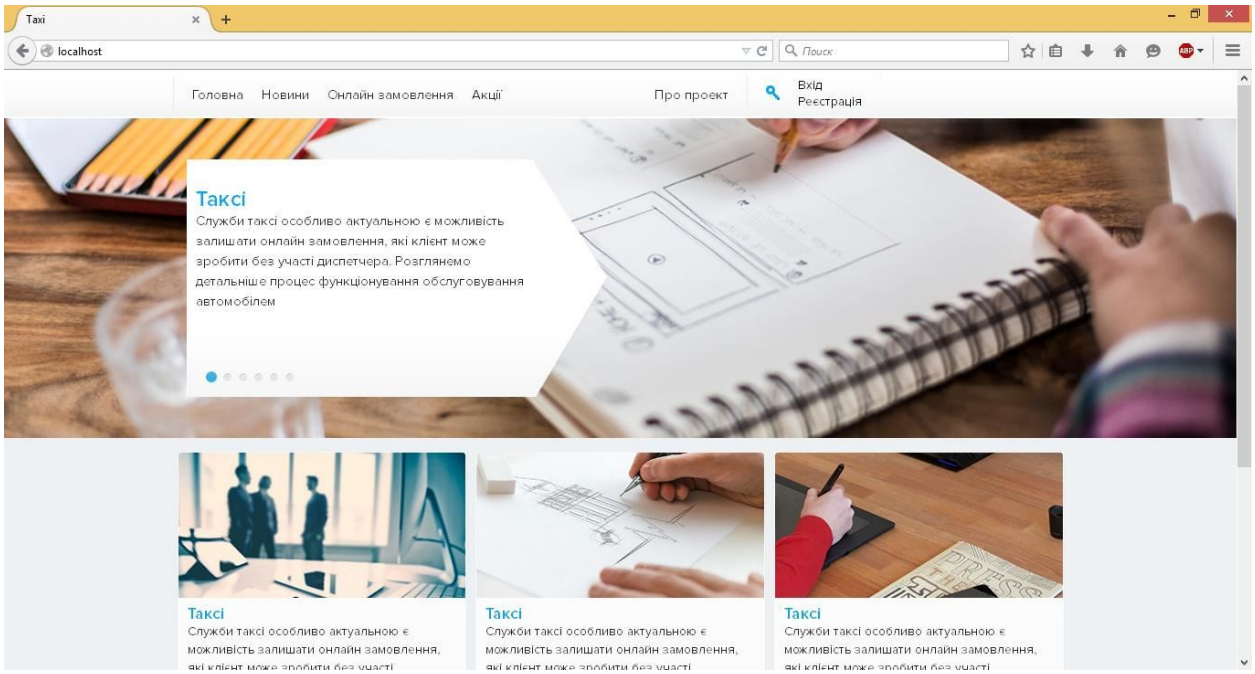

Рисунок 3.2 – Головне вікно сайту

Після вибору пункту меню «Новини», користувачу завантажується сторінка зображена на рисунку 3.3. На ній відображаються назви новин, їх короткий опис та кількість переглядів.

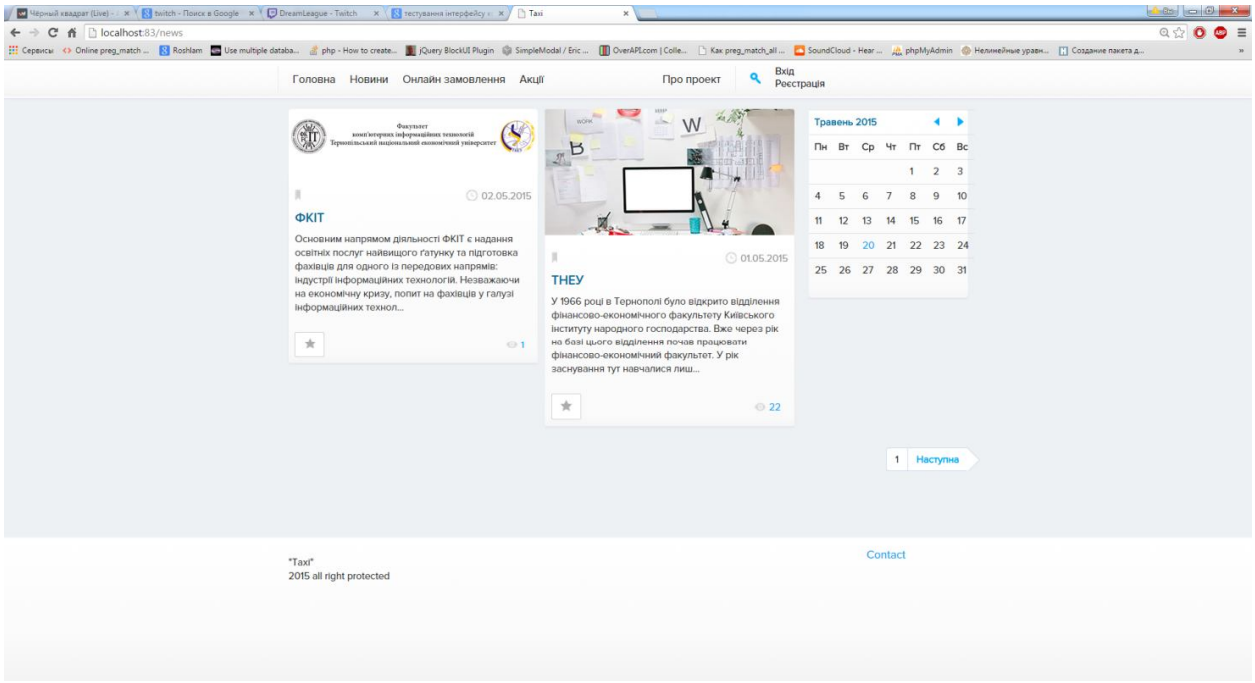

Рисунок 3.3– Вікно новин сайту

Після вибору пункту меню «Акції», користувачу завантажується сторінка зображена на рисунку 3.4. На цій веб-сторінці можна ознайомитися із умовами актуальних акцій сайту.

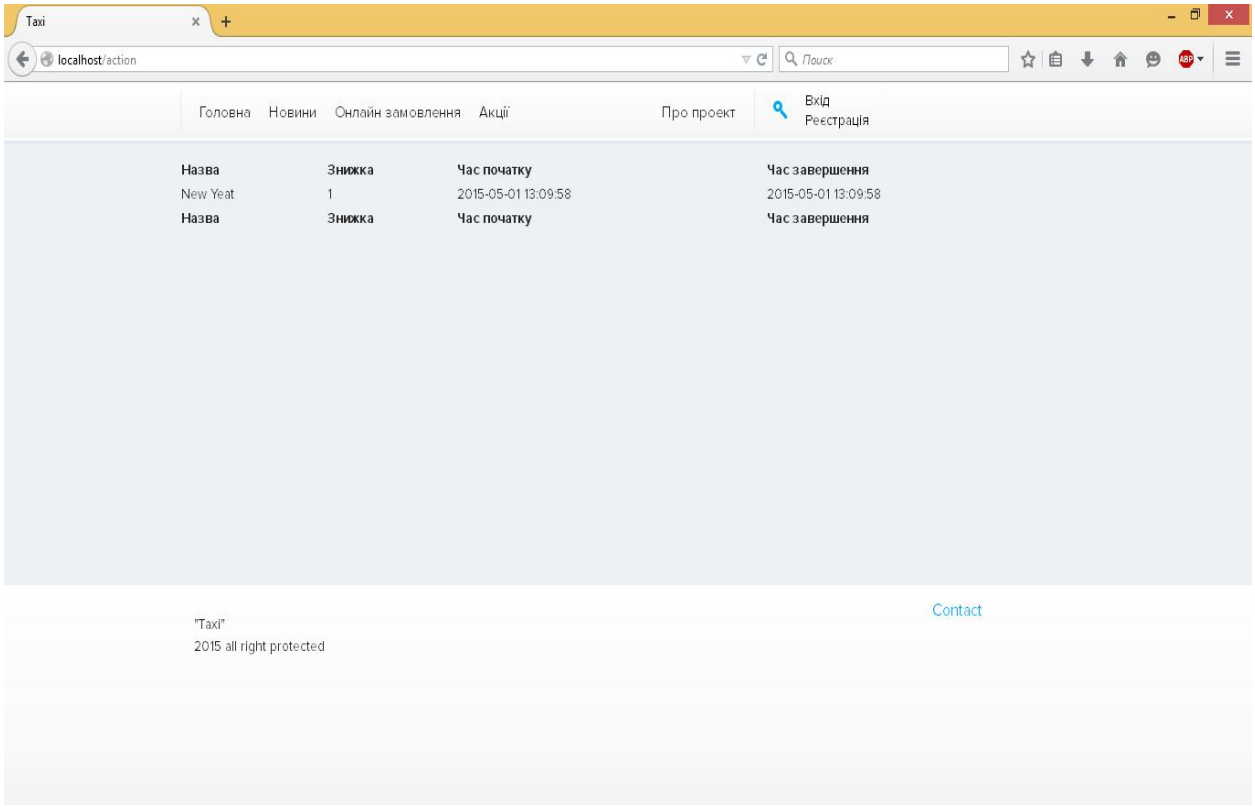

Рисунок 3.4 – Вікно з акціями сайту

Щоб зробити онлайн замовлення потрібно натиснути кнопку «Онлайн замовлення» (рисунок 3.5). Після натискання зазначеної кнопки завантажується сторінка з формою для заповнення деталей замовлення.

Після натиску на кнопку «Створити замовлення» замовлення буде відправлено на сервер, після опрацювання на сервері на екран буде виведено повідомлення про статус оформлення.

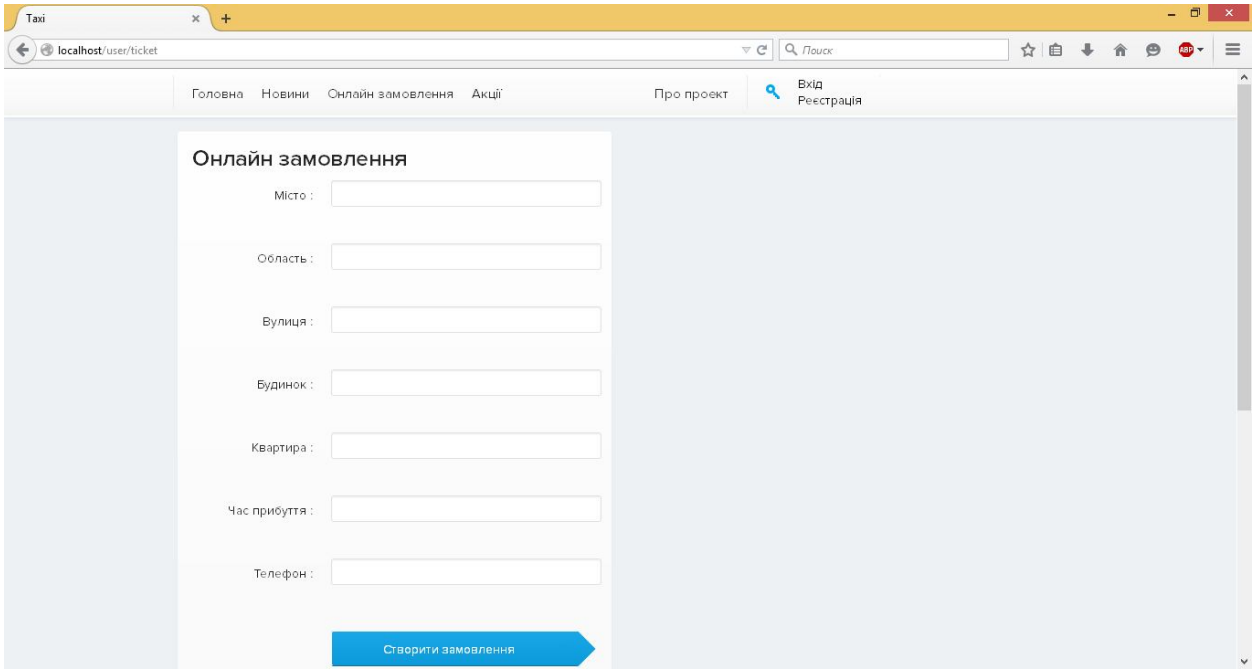

Рисунок 3.5 – Вікно онлайн замовлення

Щоб увійти в систему потрібно натиснути кнопку «Вхід», ввести логін та пароль, далі – натиснути кнопку «Увійти» (рисунок 3.6).

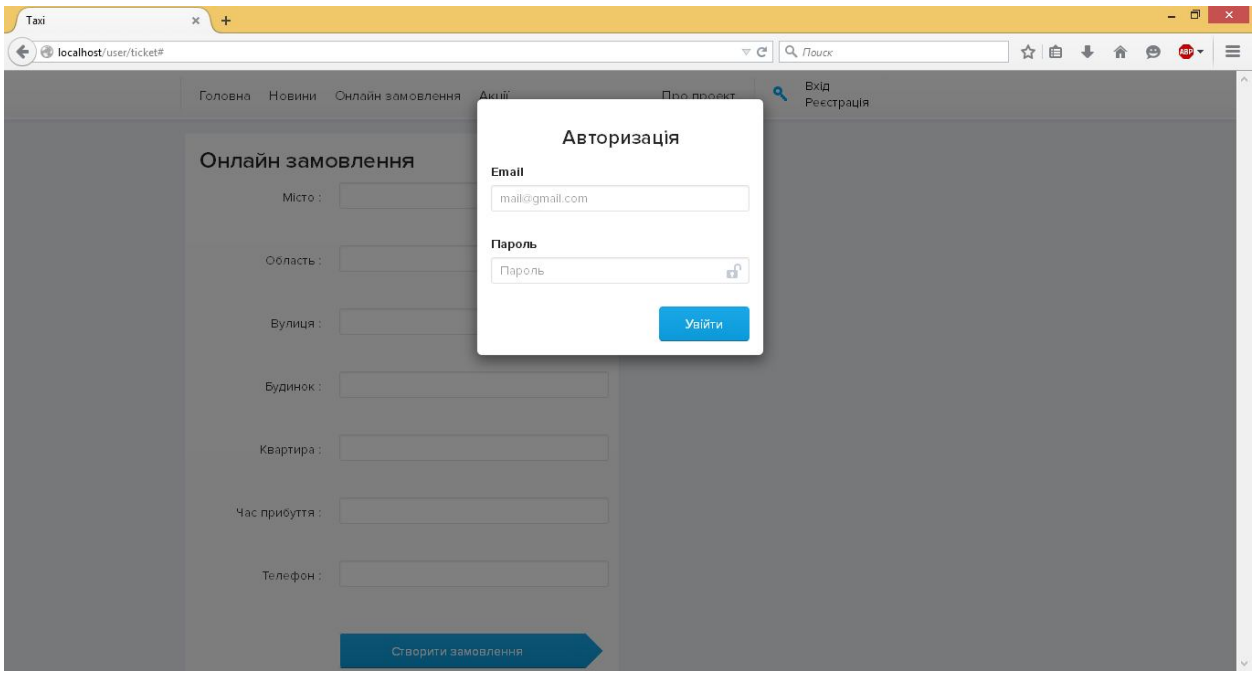

Рисунок 3.6 – Вікно авторизації

Для реєстрації у системі необхідно завантажити відповідну форму, для цього користувачу потрібно натиснути кнопку «Реєстрація», що розміщена у верхньому правому кутку вікна. Загальний вигляд сторінки із формою реєстрації проілюстровано на рисунку 3.6.

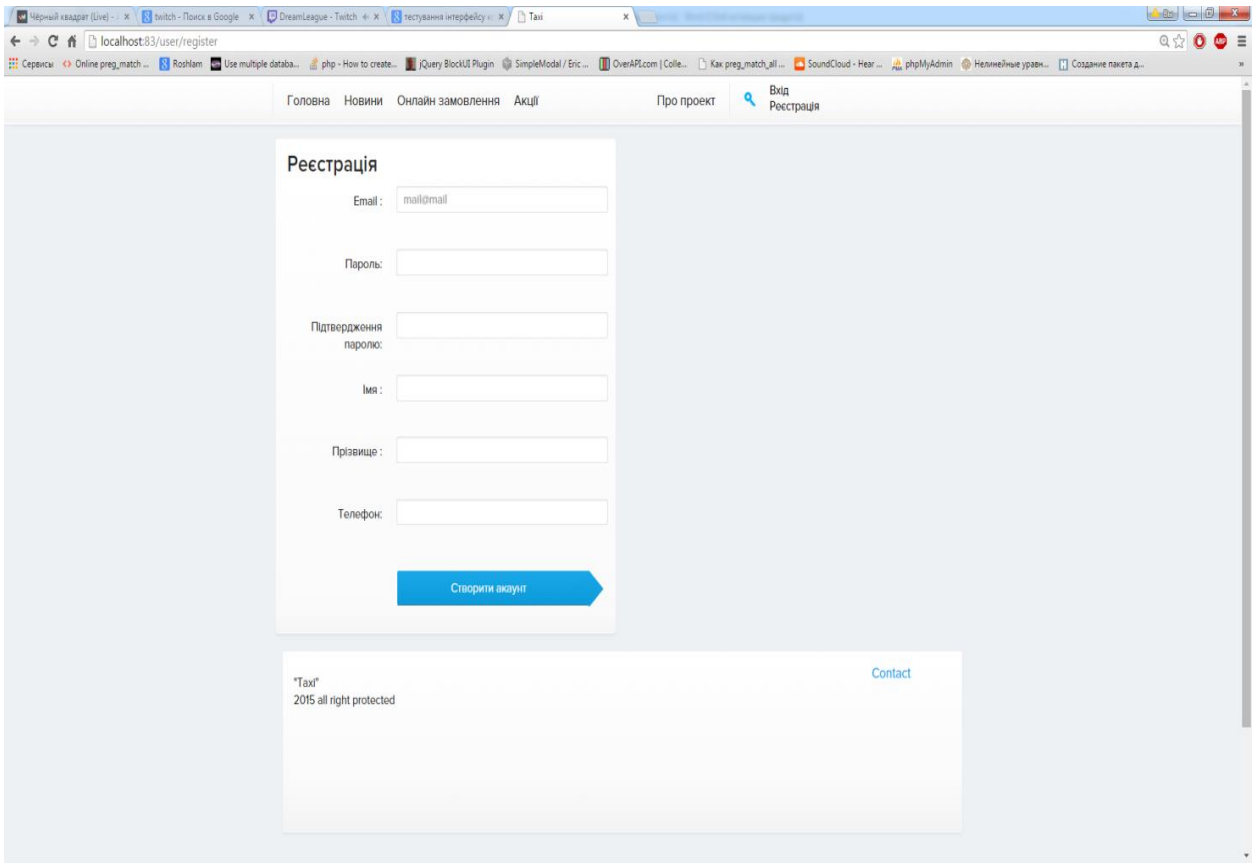

Рисунок 3.6 – Вікно реєстрації

Щоб відкрити сторінку особистого кабінету потрібно натиснути на кнопку із назвою свого профілю що розміщена у верхньому правому кутку вікна (рисунок 3.7).

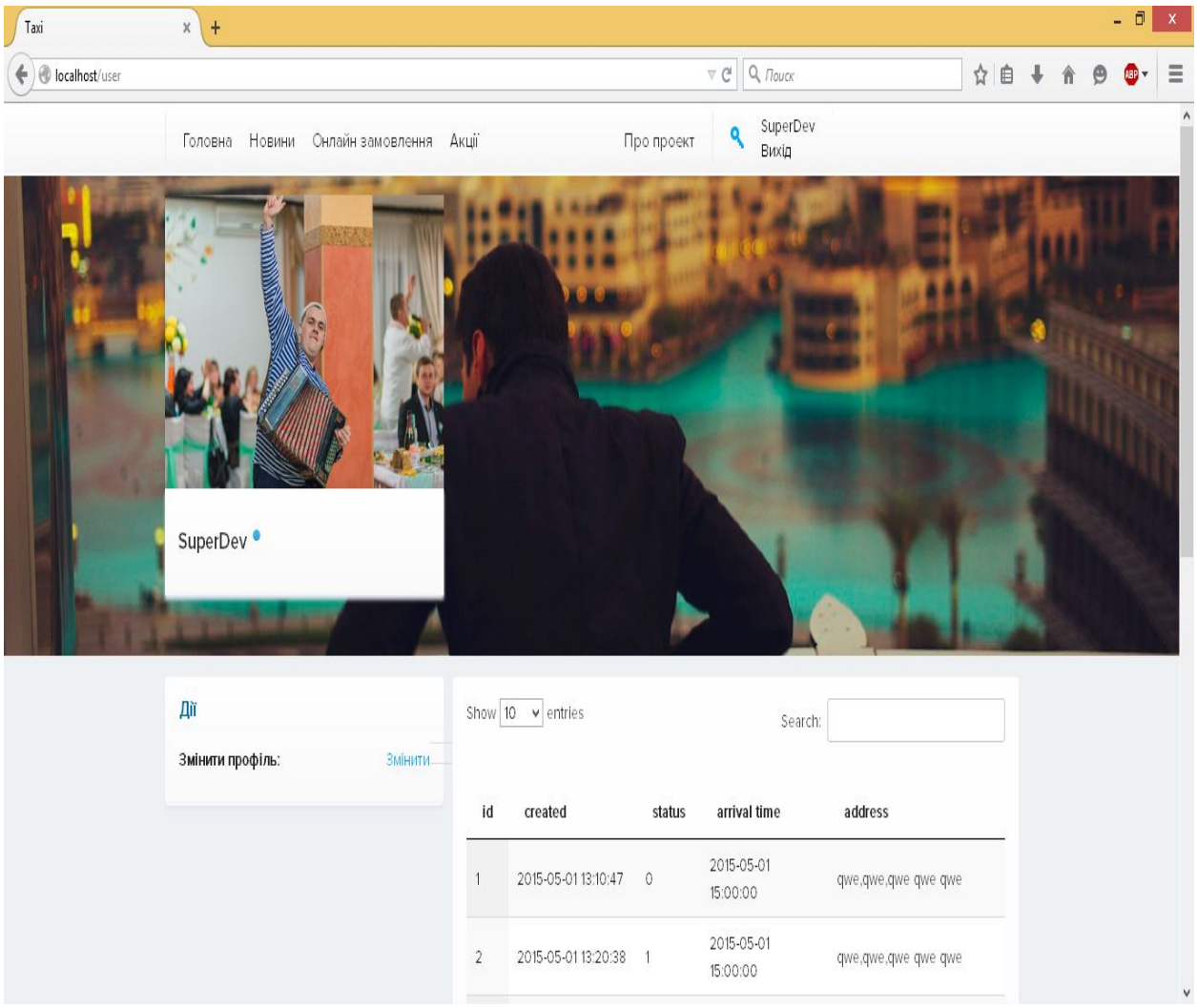

Рисунок 3.7 – Вікно особистого кабінету

Для отримання доступу до адмін-панелі веб-сайту, потрібно авторизуватися у системі під правами адміністратора. Загальний вигляд основної сторінки адмін-панелі веб-сайту показано на рисунку 3.8. Після завантаження адмін-панелі веб-сайту, адміністратору будуть доступні наступні вкладки:

- Головна;
- Новини;
- Машини;
- Замовлення;
- Водії;
- Користувачі.

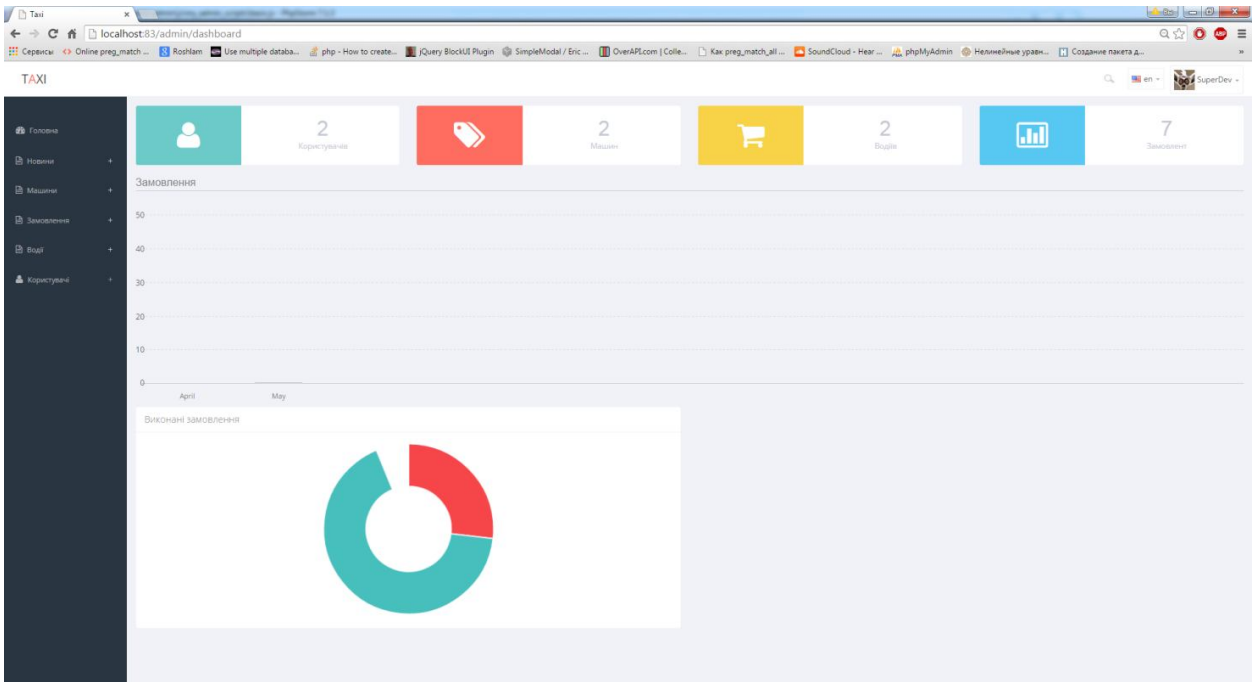

Рисунок 3.8 – Головна сторінка адмін-панелі

Після вибору вкладки «Новини», адміністратору завантажиться сторінка для роботи з новинами, загальний вигляд якої проілюстровано на рисунку 3.9.

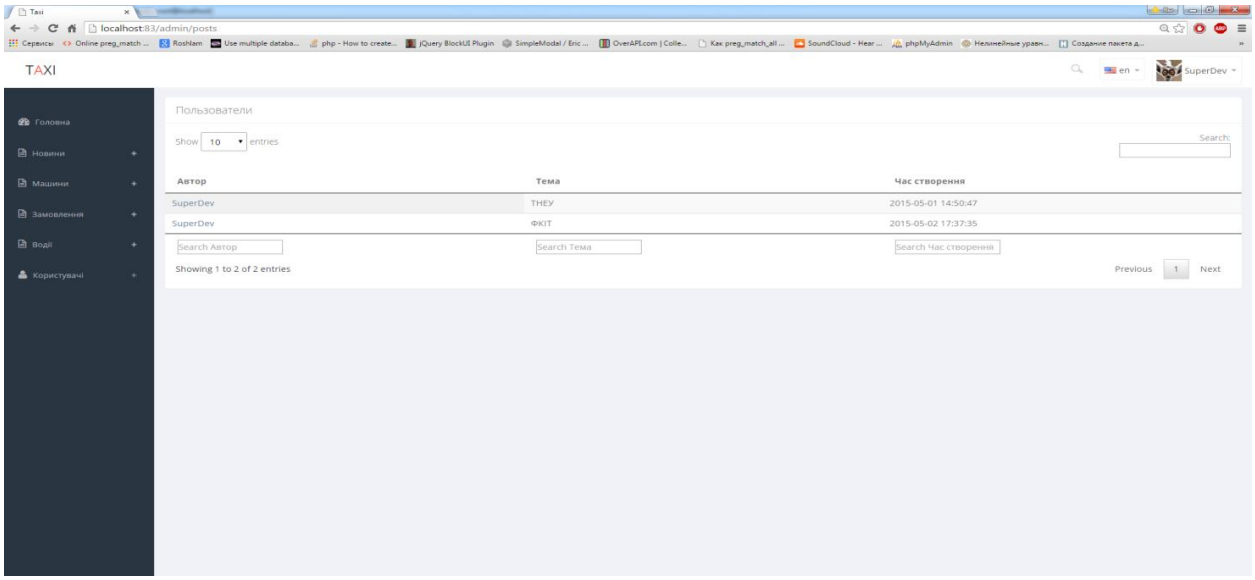

Рисунок 3.9 – Сторінка переліку новин

Якщо адміністратор системи обере вкладку «Машини», йому завантажиться сторінка для роботи з машинами. На цій сторінці можна додати дані про нову машину, відредагувати дані про уже існуючої машини

чи видалити дані машину. Загальний вигляд сторінки для проведення маніпуляцій з даними про машинами проілюстровано на рисунку 3.10.

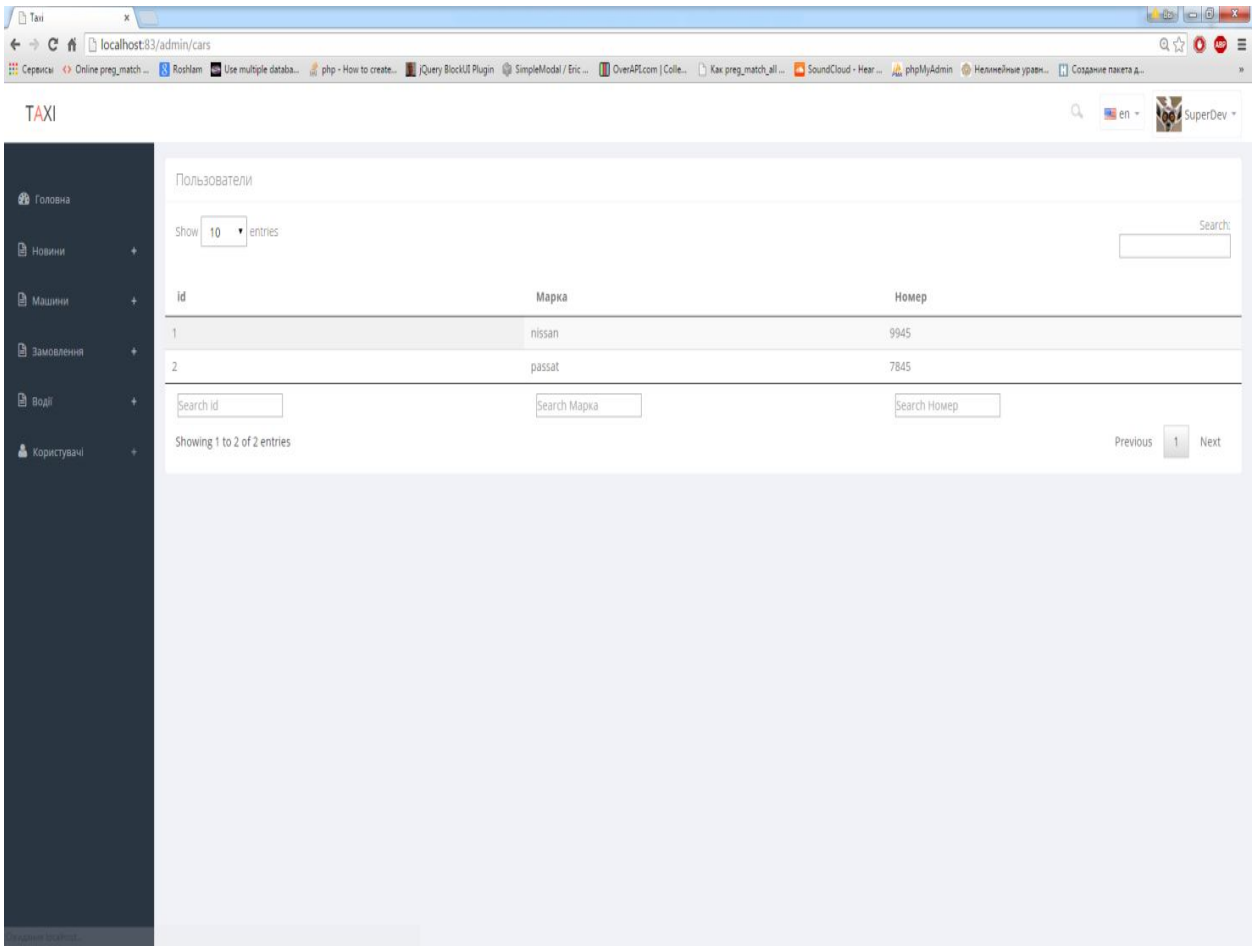

Рисунок 3.10 – Сторінка переліку машин

Якщо адміністратор системи обере вкладку «Замовлення», йому завантажиться сторінка для роботи з замовленнями. На цій сторінці можна додати дані про нове замовлення, відредагувати дані про уже існуючого замовлення чи видалити дані замовлення. Загальний вигляд сторінки для проведення маніпуляцій з даними про замовленнями проілюстровано на рисунку 3.11.

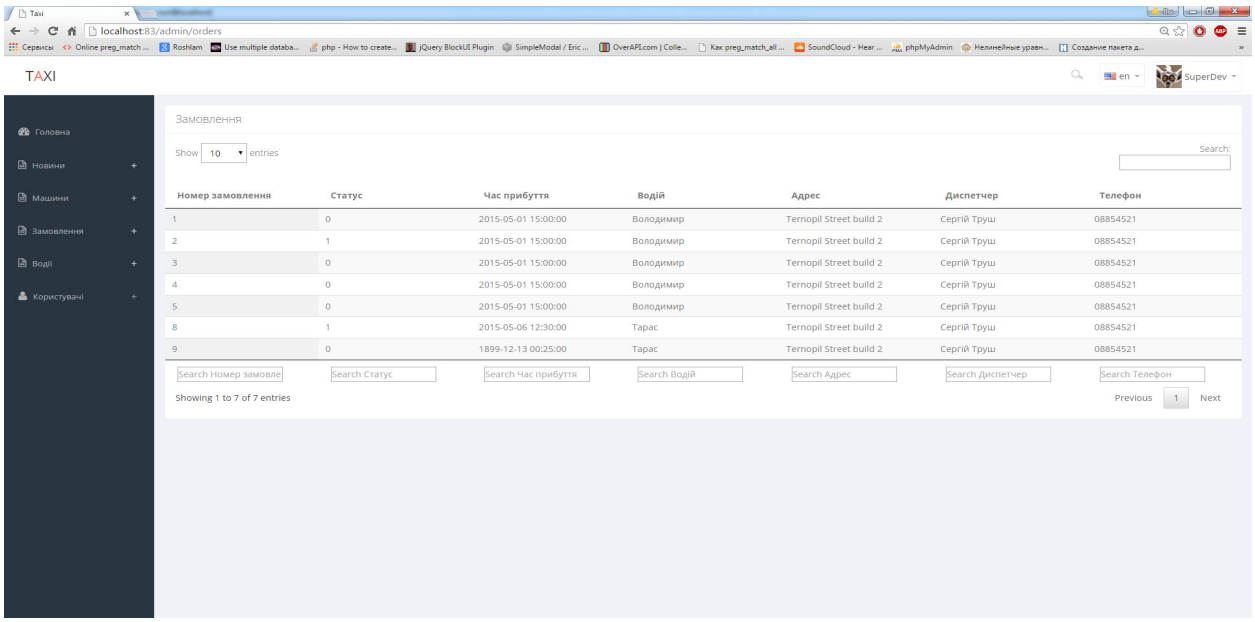

Рисунок 3.11 – Сторінка переліку замовлень

Якщо адміністратор системи обере вкладку «Користувачі», йому завантажиться сторінка для роботи з користувачами. На цій сторінці можна додати дані про нового користувача, відредагувати дані про уже існуючого користувача чи видалити дані про користувача. Загальний вигляд сторінки для проведення маніпуляцій з даними про користувача проілюстровано на рисунку 3.12.

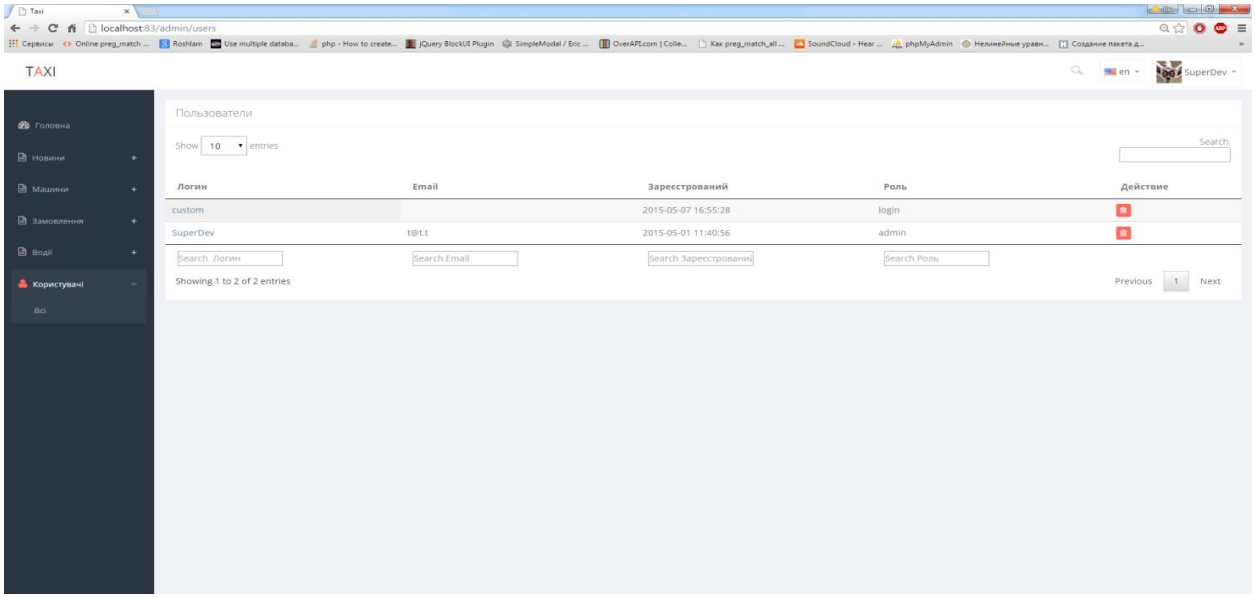

Рисунок 3.12 – Сторінка переліку користувачів

Після вибору вкладки «Водії», завантажиться сторінка системи для роботи з водіями. На цій сторінці можна додати дані про нового водія, відредагувати дані про уже існуючого водія чи видалити їх. Загальний вигляд сторінки для проведення маніпуляцій з даними про водія проілюстровано на рисунку 3.13.

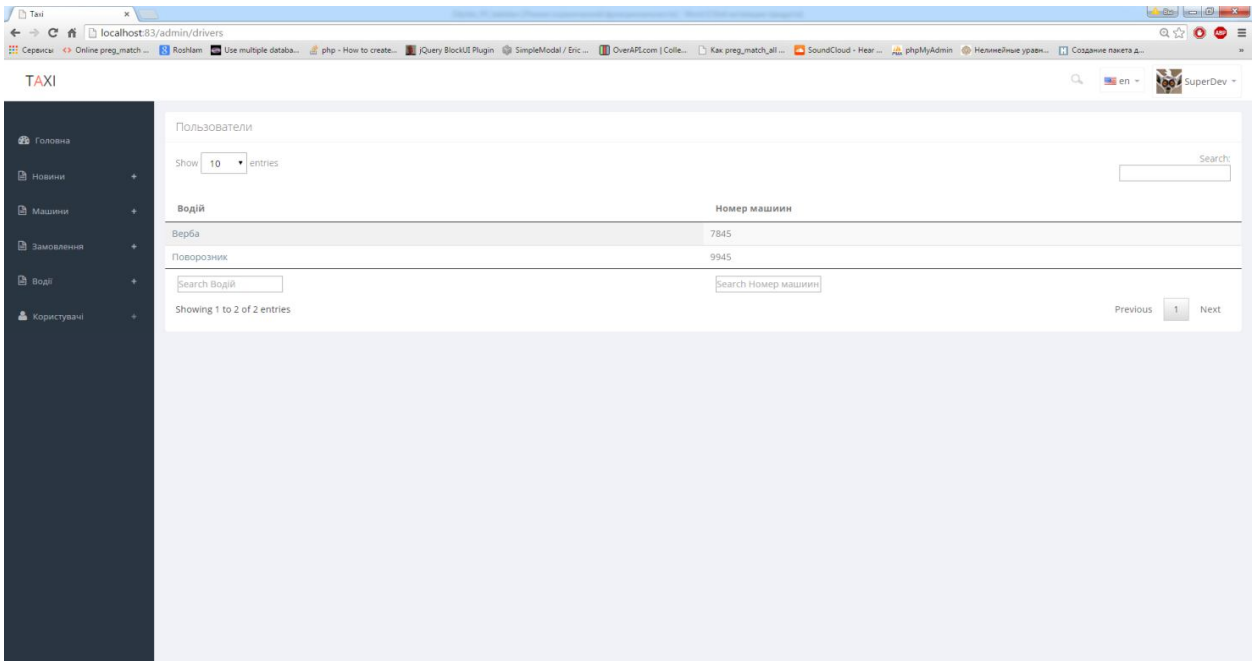

Рисунок 3.13 – Сторінка переліку водіїв

Висновки до третього розділу:

4. Розроблено діаграму елементів і зв’язків, на якій зображені всі елементи бази даних.
5. Розроблено таблиці індетифікаторів, типів та індексі. У таблиці типів описано назви атрибутів усіх відношень, їх тип та розмірність. Таблиця індексів містить дані про назву атрибуту, назву відношення, якому належить цей атрибут та його опис.Таблиця індетифікаторів містить дані про назву елементу та його ідентифікатор.
6. Розроблено ER-діаграму та зпроектовано структуру бази даних web-орієнтованої інформаційної системи для автоматизації роботи диспетчера таксі .

7. У звязку з тим, що мова PHP і сервер бази даних MySQL є кросплатформленим, їх було для робробки цієї системи. Також я вибрав фреймворк для написання «Laravel» і реалізовавав на ньому вебсайт.
8. Розроблені у другому розділі об’єктно-орієнтовані діаграми класів було трансльовано в програмний код, на основі фреймворку «Laravel».
9. Було написоно DDL код для побудови бази даних із взаємозвязками та запити для її тестового заповнення.
10. На основі специфікації вимог до програмної системи, розробленої у першому розділі роботи, було створено тестові випадки для проведення функціонального тестування ситеми.
11. На основі розроблених тестових випадків було проведено функціональне тестування, результати, якого показали, що програмна система для прийому заявок таксі працює коректно.
12. Із залученням користувачів системи було проведено GUI та USABILITY тестування розробленої програмної системи, результати якого показали, що зручність використання розроблюваної системи для пересічного користувача знаходиться на високому рівні.
13. Створено інструкцію користувача.

Список використаної літератури

1. Джон К. ВанДик , Мэт Вестгейт. Pro Drupal 7 Development: Third Edition / Todd Tomlinson . John K. VanDyk - Apress, 2010 .
2. Стивен Хольцнер . PHP в примерах. / Стивен Хольцнер . М.: 000 «Бином-Пресс», 2007 г. Пер. с англ. 352 с
3. Ларри Ульман. Ульман Л. Основы программирования на PHP:/Ларри Ульман. Пер. с англ. -М.: ДМК Пресс, 2001. -288 с.: ил. (Самоучитель).
4. Александр Мазуркевич. MB PHP: настольная книга программиста /Александр Мазуркевич, Дмитрий Еловой. — Мн.: Новое знание, 2003. — 480 с.: ил
5. Томсон Лаура. Разработка Web-приложений на PHP и MySQL: Пер. с англ. /Лаура Томсон, Люк Вел-
6. линг. — 2-е изд., испр. — СПб: ООО «ДиаСофтЮП», 2003. — 672 с.
7. Гутманс Э., Баккен С, Ретанс Д. PHP 5. Профессиональное программирование./ Пер. с англ. СПб: Символ- Плюс, 2006. 704 с., ил.
8. М. Дубаков. Веб-мастеринг. / Санкт-Петербург, BHV, 2002.
9. Колисниченко Д. Н. PHP 5/6 и MySQL 6. Разработка Web-приложений / Денис Колисниченко. – СПб: Вильямс, 2009. - 607 с. : ил., табл.
10. Прохоренок Н. HTML, JavaScript, PHP и MySQL. Джентльменский набор Web-мастера / Николай Прохоренок. – СПб: БХВ-Петербург, 2010. - 900 с.
11. Томас А. Пауэлл. Ajax. Настольная книга программиста / Томас А. Пауэлл. – М: Эксмо, 2009. - 720 стр.
12. Геронимус Б.Л. Экономико-математические методы в планировании на автомо- бильном транспорте / Б.Л. Геронимус, Л.В. Царфин. – М.: Транспорт, 1988. – 192 с
13. https://uk.wikipedia.org/
14. Ю.В. Нікольський. Дискретна математика. Київ. Видавнича група BHV. 2007 р