МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

_____ проф. Приходько С.Б.

«___» _____ 2023 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка програмного забезпечення для автоматизації роботи диспетчера служби таксі

Виконав: студент групи 4151

_____ Кардач Є.О.

(підпис, ПІБ)

Керівник роботи:

доцент кафедри ПЗАС, к.т.н., доцент

(посада, науковий ступень вчене звання)

_____ Макарова Л.М.

(підпис, ПІБ)

Миколаїв – 2023 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)
«___» _____ 2023 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Кардач Євгенію Олександровичу _____
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка програмного забезпечення для автоматизації роботи диспетчера служби таксі _____

Керівник роботи Макарова Л.М. _____

Затверджені наказом ректора №257-уч від «17» березня 2023 р.

2. Термін подання роботи: 29.05.2023 р. _____

3. Вихідні дані по роботі: _____

_____

4. Перелік питань, що належать до розробки (найменування розділів)_____

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (при необхідності); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів _____

Титульний слайд, Мета та завдання кваліфікаційної роботи, Аналіз вимог до ПЗ, Архітектура ПЗ, Діаграма варіантів використання, Сценарії варіантів використання, Концептуальна модель даних, Діаграма класів, Діаграми взаємодії, Логічна модель бази даних, Технічний проект ПЗ, Результати розробки, Висновки, Заключний слайд. _____

_____

_____

_____

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 20.03.2023 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1.	Підготовка розділу ВСТУП	24.03.2023	ПП*
2.	Аналіз практичної задачі інженерії ПЗ	31.03.2023	ПП*
3.	Аналіз вимог до ПЗ	07.04.2023	ПП*
4.	Розробка архітектури ПЗ	21.04.2023	
5.	Розробка проекту ПЗ	05.05.2023	
6.	Реалізація та тестування ПЗ	12.05.2023	
7.	Підготовка розділу Результати розробки ПЗ	17.05.2023	
8.	Підготовка розділу з охорони праці	19.05.2023	ПП*
9.	Підготовка розділу ВИСНОВКИ	24.05.2023	
10.	Оформлення списку використаних джерел та додатків	26.05.2023	
11.	Подання на кафедру ПЗАС тексту остаточного варіанту роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (Додатки 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів/ідентичності/схожості із використанням програмно-технічних засобів», який введений в дію наказом ректора НУК за №20 від 20.01.2020 р.)	29.05.2023	не пізніше, ніж за 14 діб до захисту (згідно п.4.1 зазначеного Порядку)
12.	Підготовка презентації та доповіді	02.06.2023	
13.	Попередній захист роботи на засіданні кафедри ПЗАС	05.06.2023	
14.	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файлу-опису кваліфікаційної роботи (згідно Додатку до наказу ректора НУК за №287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	12.06.2023	

* - за результатами переддипломної практики (ПП), яка була з 01.02.2023 до 19.03.2023 р.

Студент _____ Кардач Є.О.
(підпис) (ПІБ)

Керівник роботи _____ Макарова Л.М.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – розробці програмного забезпечення для автоматизації роботи диспетчера служби таксі, що дозволить спростити роботу диспетчера та налагодити комунікацію між диспетчером та водієм.

У кваліфікаційній роботі представлені аналіз практичної задачі інженерії програмного забезпечення за темою кваліфікаційної роботи, аналіз існуючих програмних рішень та постановка задачі на розробку програмного забезпечення. Розроблено ескізний, технічний та робочий проекти програмного забезпечення, представлено результати розробки та розділ з охорони праці.

Кваліфікаційна робота викладена на 75 сторінках машинописного тексту та містить: 15 рисунків, 14 таблиць, 5 додатків, список використаних джерел з 17 найменувань.

ABSTRACT

The qualification work is dedicated to solving the practical task of software engineering, characterized by complexity and uncertainty of conditions – software development to automate the work of taxi service dispatcher, which will simplify the dispatcher's work and establish communication between the dispatcher and the driver.

In qualification work presents analysis of the practical task of software engineering on the topic of qualification work, analysis of existing software solutions and formulation of the problem for software development. The sketch, technical and working projects of the software, results of development and the section on labor protection are developed.

The qualification work is presented on 75 pages of typewritten text and contains: 15 figures, 14 tables, 5 appendices, list of references from 17 names.

ЗМІСТ

ВСТУП

Таксі – популярний громадський транспорт, який використовується для перевезення пасажирів у будь-яку зазначену точку. Служба таксі оперує великою кількістю інформації і для ефективної роботи потребує можливості постійної комунікації між працівниками. Популярність служби таксі серед населення обумовлена можливістю доставки в будь-яку частину міста в будь-який час в найкоротші терміни і цілодобовий графік роботи.

У наш час автоматизація роботи служби таксі є актуальною темою і розробки у цій галузі знаходять широке застосування у полегшенні бізнес-процесів. Автоматизація надає можливість здійснювати контроль над процесом виконання замовлень, дозволяє налагодити комунікацію між диспетчером та водієм, спрощує документообіг.

Кваліфікаційна робота присвячена розв’язанню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов – розробці програмного забезпечення для автоматизації роботи диспетчера служби таксі.

Програмне забезпечення, що розробляється, призначене для працівників приватного підприємства, яке надає населенню послуги таксі, і дозволить їм підвищити ефективність своєї роботи за рахунок систематизації та швидкого пошуку необхідної їм інформації. Це спростить роботу персоналу, полегшить комунікацію між працівниками та спростить складання звітності. Дружній користувацький інтерфейс дозволить користувачу використовувати програмне забезпечення без додаткового навчання.

Для досягнення поставленої мети необхідно:

- виконати аналіз практичної задачі інженерії програмного забезпечення з теми роботи, а саме проаналізувати організаційну структуру служби таксі та визначити специфіку її роботи з точки зору інформаційних технологій, які застосовуються, розглянути існуючі програмні рішення, здійснити постановку

задачі на розробку програмного забезпечення для автоматизації роботи диспетчера служби таксі;

- на основі постановки задачі розробити технічне завдання на розробку програмного забезпечення;

- виконати аналіз вимог до програмного забезпечення для автоматизації роботи диспетчера служби таксі;

- розробити архітектуру програмного забезпечення для автоматизації роботи диспетчера служби таксі;

- розробити ескізний, технічний та робочий проєкти програмного забезпечення для автоматизації роботи диспетчера служби таксі;

- здійснити тестування та випробування розробленого програмного забезпечення;

- розробити необхідну програмну документацію.

Об'єктом кваліфікаційної роботи є процес розробки програмного забезпечення для автоматизації роботи диспетчера служби таксі.

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРА СЛУЖБИ ТАКСІ

1.1 Аналіз практичної задачі інженерії програмного забезпечення

Перш ніж розпочати проєктування програмного забезпечення для автоматизації роботи диспетчера служби таксі, потрібно виконати аналіз практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, та визначити специфіку застосування інформаційних технологій в роботі диспетчерської служби таксі. Також для вдалого проєктування програмного забезпечення потрібно провести аналіз існуючих програмних рішень та виконати постановку задачі на розробку програмного забезпечення.

Таксі (від фр. taximètre «лічильник ціни») – громадський транспорт, як правило, автомобіль, що використовується для перевезення пасажирів та рідше вантажів, у будь-яку заздалегідь зазначену точку призначення з оплатою проїзду по лічильнику – таксометру [1]. Іноді ціна призначається за договором з водієм. Особливою характерною рисою для таксі є так звані "Шашки", виконані в жовтій прямокутній формі, які прикріплюють на дах таксі [2]. Найчастіше в якості автомобілів таксі використовуються седани або мінівени.

Вартість поїздки на таксі, як правило, не залежить від кількості людей і вантажу, додаткова оплата йде тільки за простій. У місцях, де попит на таксі великий, наприклад, в аеропортах вночі, таксист може взяти того пасажира, хто запропонує більшу ціну. Клієнт може відразу дізнатися вартість замовлення.

Існують також маршрутні таксі, які здійснюють перевезення за певними лініях - маршрутами. Від автобуса маршрутні таксі відрізняються невеликою кількістю пасажирів.

Крім власне таксі – легкового візництва - служби таксі можуть надавати й інші послуги:

– оренда автомобіля представницького класу;
– вантажне таксі для перевезення габаритних вантажів;
– "драйвер" – підвезення п'яного водія на його ж машині.

Зазвичай, водії таксі координують свої дії з диспетчером таксі, який може передавати водіям відомості про замовлення, зазвичай за допомогою сучасного смартфону з встановленим відповідним програмним забезпеченням.

Дзвінок пасажира або ж заповнена заявка на сайті з урахуванням пріоритету обслуговування клієнта, наприклад, ціни, потрапляє в інформаційну систему. В результаті є можливість підбору машини по тарифу чи іншими критеріями і значно більша ймовірність швидкої подачі машини.

Інформаційна взаємодія працівників служби таксі організована таким чином: кожен водій, який заступає на зміну, повинен зв'язатися з диспетчером, після чого диспетчер вносить його до картки працюючих в даний момент водіїв. Замовлення фіксуються в базу замовлень, дата та час надходження замовлення фіксуються автоматично.

Зі списку замовлень водії, відмічені як працюючі в даний момент, обирають замовлення. Водій автоматично звітує після виконання замовлення. Після кожної зміни диспетчер формує звіт, який включає в себе інформацію про те, скільки замовлень надійшло, скільки замовлень виконано, скільки було скасовано, їх загальну вартість і таке інше. Ці звіти надходять до адміністратора, на підставі чого адміністратор формує загальний (зведений) звіт за певну дату.

1.2 Аналіз існуючих програмних продуктів

На сьогоднішній день існує досить широке розмаїття програмного забезпечення для автоматизації роботи диспетчера служби таксі, яке працює як

із встановленням на власні апаратні платформи, так і з використанням інтернет-технологій без купівлі дорогого програмного забезпечення та обладнання.

До першої групи програм відноситься, зокрема, "АРМ диспетчера таксі". Розробник: "A123 Software" [3]. Програмне забезпечення призначене для прийняття замовлень та їх обробки з подальшим оформленням звітів.

Переваги: простота роботи, низькі системні вимоги, простий інтерфейс, який не потребує спеціальної підготовки. Розроблено для операційної системи Windows.

Недоліки: поєднує в собі функції диспетчера і частину функцій адміністратора, немає належного захисту даних, немає можливості зведених звітів.

Аналогічним є програмний комплекс "Диспетчер таксі" від розробника F-Group Software. Але на думку відвідувачів спеціалізованого "Форуму Таксі Сервис - все про таксі в Україні. Форум водіїв таксі та керівників в Україні!", особливо з невеликих міст, при використанні таких програм найбільш проблемними питаннями є оновлення програмного забезпечення, наявність великої кількості зайвих функцій, "відставання у розвитку" вже після виходу програм та орієнтація на фірми-агрегатори [4].

До другої групи програм відносяться, зокрема, програми e(taxi) або "Таксі Навігатор (Евос)".

Основні можливості програми e(taxi): обробка більше, ніж 5000 заказів на добу; робота з водіями через Java або Android клієнт; білінг з водіями та клієнтами; інтеграція з call-центрами та іншими диспетчерськими таксі.

Програма передається безоплатно, але є абонплата за користування та підтримку. Система працює на захищених серверах, які розміщені в дата-центрах Європи [5].

Аналогічно працює і програмний комплекс "Такси Навигатор" від Евос – IT-рішення для автоматизації різних процесів в роботі диспетчерської таксі [6].

При очевидних перевагах - можливості швидко почати роботу диспетчерської таксі при мінімальних початкових витратах, такі рішення мають

і певні недоліки. А саме, неможливість продовжувати роботу при відмові в обслуговуванні від оператора віддаленого програмного комплексу та можлива втрата даних через несанкціонований доступ до сервера чи програмного забезпечення [7].

1.3 Постановка задачі на розробку програмного забезпечення для автоматизації роботи диспетчера служби таксі

В результаті аналізу практичної задачі інженерії програмного забезпечення, що стосується розробки програмного забезпечення для автоматизації роботи диспетчера служби таксі, для полегшення і прискорення роботи диспетчерської служби таксі, було прийнято рішення розробити програмне забезпечення, призначенням якого є автоматизація роботи диспетчерської служби таксі, покращення та легкість ведення та обробки даних, полегшення комунікації між працівниками. В рамках даної кваліфікаційної роботи буде розроблено додаток диспетчера.

Додаток диспетчера – це додаток для персонального комп’ютера, який призначений для автоматизації роботи диспетчера, шляхом надання наступного функціоналу:

- авторизація користувача;
– реєстрація нового водія;
– перегляд списку водіїв;
– автоматичне додавання нового замовлення;
– перегляд списку замовлень;
– автоматичне формування звітів.

Було виділено вимоги до організації вхідних та вихідних даних. Введення даних повинно здійснюватися за допомогою пристроїв введення даних (клавіатури та миши). Дані або повинні вводитися вручну, або обиратися із представлених списків.

Виведення даних повинно здійснюватися за допомогою пристроїв виведення даних (монітор та принтер).

Обмін інформацією між додатком і сервером, а також між сервером і базою даних повинні відбуватися за допомогою файлів з довільним доступом, з розширенням .json.

Вхідні дані

Дані для авторизації: логін/пароль.

Дані про замовлення та його статус:

- дата/час;
- адреса (вулиця, будинок, під'їзд);
- виконавець;
- телефон;
- статус виконання (не виконується, у процесі, виконано).

Вихідні дані

Дані про водіїв та їх зайнятість:

- ПІБ;
- телефон;
- посвідчення;
- номер автомобіля;
- марка автомобіля;
- колір автомобіля;
- зайнятість.

Дані про замовлення:

- дата/час;
- телефон замовника;
- адреса (вулиця, будинок, під'їзд);
- дані про виконавця;
- статус виконання (не виконується, у процесі, виконано);
- звітність.

Було виділено вимоги до складу та параметрів технічних засобів. Система повинна задовольняти наступним вимогам в мінімальній комплектації:

Апаратне забезпечення серверної частини:

– CPU: Intel Core i7 або еквівалентного рівня;

– RAM: 4 GB;

– HDD: 100 GB.

Апаратне забезпечення клієнтської частини:

– CPU: Intel Core i5 або еквівалентного рівня);

– RAM: 2 GB;

– HDD: 50 GB.

Також є вимоги до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows версії не нижче 7.

Детальніша постановка задачі представлена у технічному завданні, яке наведено у додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРА СЛУЖБИ ТАКСІ

2.1 Аналіз вимог до програмного забезпечення

2.1.1 Побудова діаграми варіантів використання

Діаграма варіантів використання – це діаграма, на якій зображуються відносини між діючими особами і варіантами використання [8].

Кожен варіант використання задає сценарій взаємодії системи з дійовими особами або ролями, що дає в підсумку значущий для них результат. Дійовими особами можуть бути не тільки люди, але й інші системи або підсистеми, які взаємодіють з ними. Для програмного забезпечення для автоматизації роботи диспетчера служби таксі виявлено одного актора - диспетчера, який буде взаємодіяти з цим програмним забезпеченням.

Для фіксації вимог було виконано побудову діаграми варіантів використання, яка представлена на рисунку 2.1.

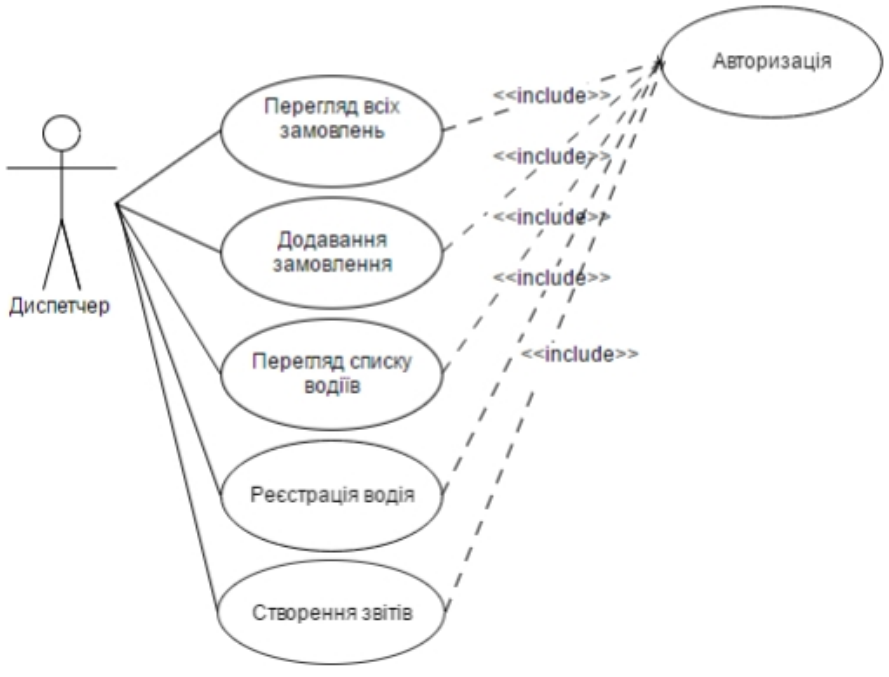

Рисунок 2.1 – Діаграма варіантів використання програмного забезпечення для автоматизації роботи диспетчера служби таксі

Діаграма варіантів використання є основою для проектування та оцінки готовності системи до впровадження. Далі необхідно провести специфікацію виявлених варіантів використання.

2.1.2 Специфікація варіантів використання

Більш детально кожен варіант використання із наведених на рисунку 2.1 розглянуто в таблицях 2.1 - 2.6.

Таблиця 2.1 – Опис варіанту використання "Авторизація"

Складова	Опис
Короткий опис	Вхід до системи
Передумови	Запуск програми
Основний потік подій	Актор вводить логін та пароль та підтверджує вхід. Система перевірить дані на достовірність, якщо логін та пароль достовірний, то актор входить до системи.
Альтернативний потік подій	Якщо логін або пароль не достовірні, то система повідомить про помилку, та запропонує ввести ще раз логін та пароль.
Постумови	Отримання доступу до системи

Таблиця 2.2 – Опис варіанту використання "Перегляд всіх замовлень"

Складова	Опис
Короткий опис	Перегляд списку всіх замовлень диспетчером
Передумови	Даний варіант використання буде доступний після успішного проходження авторизації диспетчера та вибору диспетчером опції «Перегляд списку замовлень».
Основний потік подій	Система відобразить таблицю з полями Дата, Телефон замовника, Адреса (вулиця, будинок, під’їзд), Виконавець, Статус виконання.
Альтернативний потік подій	-
Постумови	-

Таблиця 2.3 – Опис варіанту використання "Додавання замовлення"

Складова	Опис
Короткий опис	Додавання нового замовлення до системи.
Передумови	Даний варіант використання буде доступний після успішного проходження авторизації диспетчера та вибору опції «Додати замовлення».
Основний потік подій	Система відобразить вікно з полями Дата, Телефон замовника, Адреса (вулиця, будинок, під'їзд), Виконавець, Статус виконання. Диспетчер заповнивши все поля, здійснює підтвердження та збереження даних в системі. Система перевіряє коректність введення даних, якщо дані введено корректно, система видасть повідомлення про успішне додавання замовлення та збереження даних в системі.
Альтернативний потік подій	Якщо дані введено не коректно, система видасть помилку та запропонує змінити раніше введені дані та зберегти їх і диспетчер остаточно підтверджує їх введення
Постумови	Дані про замовлення з'являються у системі

Таблиця 2.4 – Опис варіанту використання "Перегляд списку водіїв"

Складова	Опис
Короткий опис	Перегляд списку водіїв диспетчером
Передумови	Даний варіант використання буде доступний після успішного проходження авторизації диспетчера та вибору диспетчером опції «Перегляд списку водіїв».
Основний потік подій	Система відобразить таблицю з полями ПІБ, Телефон, Посвідчення, Марка автомобіля, Колір автомобіля, Зайнятість
Альтернативний потік подій	-
Постумови	-

Таблиця 2.5 – Опис варіанту використання "Реєстрація водія"

Складова	Опис
Короткий опис	Додавання нового водія до системи.
Передумови	Даний варіант використання буде доступний після успішного проходження авторизації диспетчера та вибору опції «Реєстрація водія».
Основний потік подій	Система відобразить вікно з полями ПІБ, Телефон посвідчення, Марка автомобіля, Колір автомобіля Адміністратор заповнивши все поля, здійснює підтвердження та збереження даних в системі. Система перевіряє коректність введення даних, якщо дані введено корректно, система видасть повідомлення про успішне додавання замовлення та збереження даних в системі.
Альтернативний потік подій	Якщо дані введено не коректно, система видасть помилку та запропонує змінити раніше введені дані та зберегти їх і диспетчер остаточно підтверджує іх введення.
Постумови	Дані про водія з’являються у системі.

Таблиця 2.6 – Опис варіанту використання "Створення звіту"

Складова	Опис
Короткий опис	Створення звіту за певний звітний період.
Передумови	Даний варіант використання буде доступний після успішного проходження авторизації диспетчера та вибору опції «Створення звіту».
Основний потік подій	Додаток робить запит до сервера для отримання звіту за певний період.
Альтернативний потік подій	-
Постумови	Система відобразить форму звіту.

Застосування наведених варіантів використання програмного забезпечення для автоматизації роботи диспетчера служби таксі є дієвим засобом для послідовного уточнення вимог до прогрманого забезпечення та дозволить досягти необхідного рівня уніфікації позначень для подальшого проєктування. Також на рівні окремих операцій класу можна організувати зворотний зв'язок та уточнення відповідного варіанту використання розроблюваного програмного забезпечення.

2.2 Архітектура програмного забезпечення

Для побудови UML діаграми архітектурним стилем було обрано архітектуру програмного забезпечення клієнт-сервер, тому що вона має наступні переваги:

– дозволяє організовувати мережі зі значною кількістю клієнтів;

– спрощує адміністрування за рахунок забезпечення централізованого управління обліковими записами користувачів;

– організовує ефективний доступ до ресурсів;

– для доступу до ресурсів користувачу потрібен лише один логін/пароль [9].

Архітектура програмного забезпечення для автоматизації роботи диспетчера служби таксі зображена на рисунку 2.2.

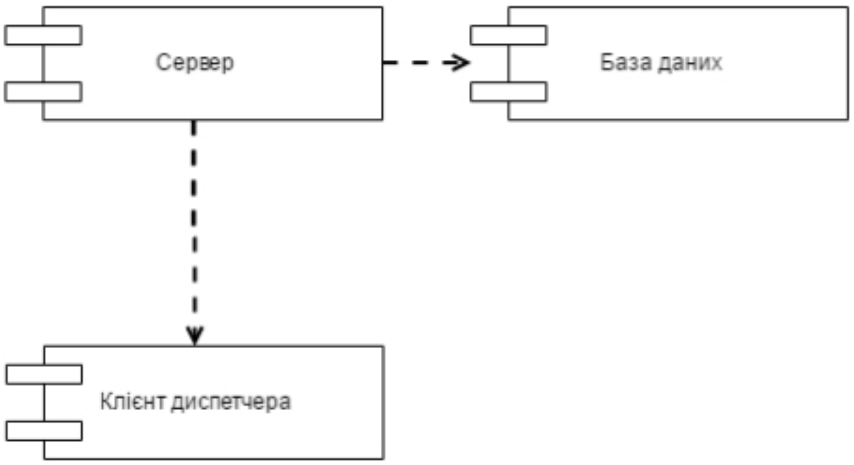

Рисунок 2.2 – Архітектура програмного забезпечення для автоматизації роботи диспетчера служби таксі

Детальний опис елементів архітектури:

– База даних – централізоване сховище даних.

– Сервер – обробляє запити з клієнтських додатків.

– Клієнт диспетчера забезпечує необхідний диспетчеру функціонал шляхом здійснення запитів до сервера.

Побудова діаграми розгортання

Діаграма розгортання – це UML-діаграма, яка відображає обчислювальні вузли під час роботи програмного забезпечення, а також об'єкти та компоненти, які виконуються на цих вузлах [8]. Діаграма розгортання програмного забезпечення для автоматизації роботи диспетчера служби таксі представленна на рисунку 2.3.

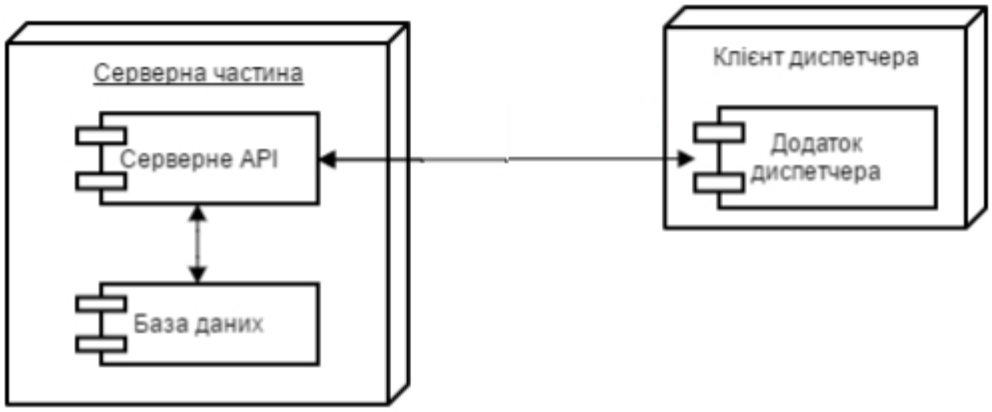

Рисунок 2.3 – Діаграма розгортання програмного забезпечення для автоматизації роботи диспетчера служби таксі

В якості компонентів виступають представлення робочих екземплярів одиниць коду. Компоненти, що не мають представлення під час роботи програми на таких діаграмах не відображаються.

2.3 Проект програмного забезпечення

2.3.1 Ескізний проект програмного забезпечення

Побудова діаграми діяльності

Діаграми діяльності - це UML-діаграми, які використовуються для моделювання процесу виконання операцій в програмному забезпеченні. Застосовуване в них графічне представлення дещо схоже на представлення діаграми станів, тому що на цих діаграмах теж є і позначення станів, і позначення переходів. Однак, по-перше, стани використовуються для уявлення дій, а не діяльностей, а по-друге, у відсутності сигнатури подій на переходах [8].

На основі описаних ключових варіантів використання були побудовані діаграма діяльності для деяких варіантів використання програмного забезпечення для автоматизації роботи диспетчера служби таксі, зображені на рисунках 2.3 – 2.5.

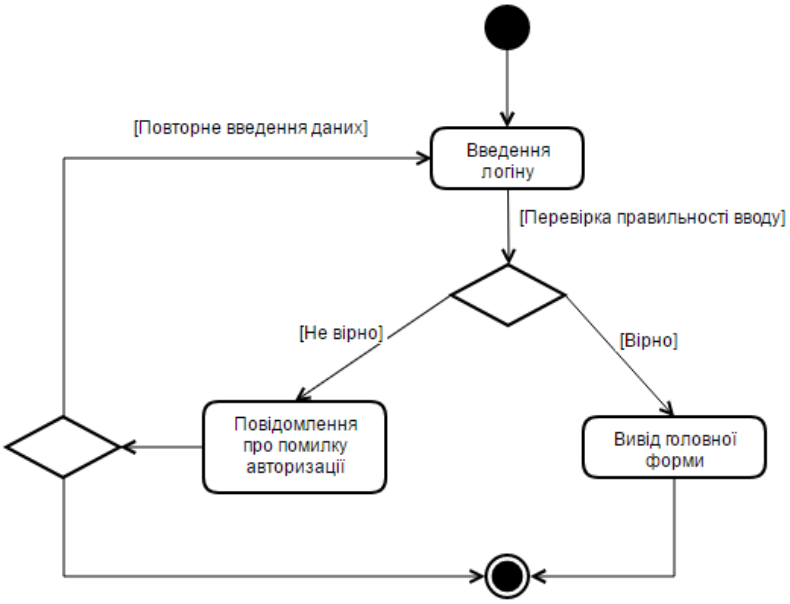

Рисунок 2.3 – Діаграма діяльності варіанту використання «Авторизація»

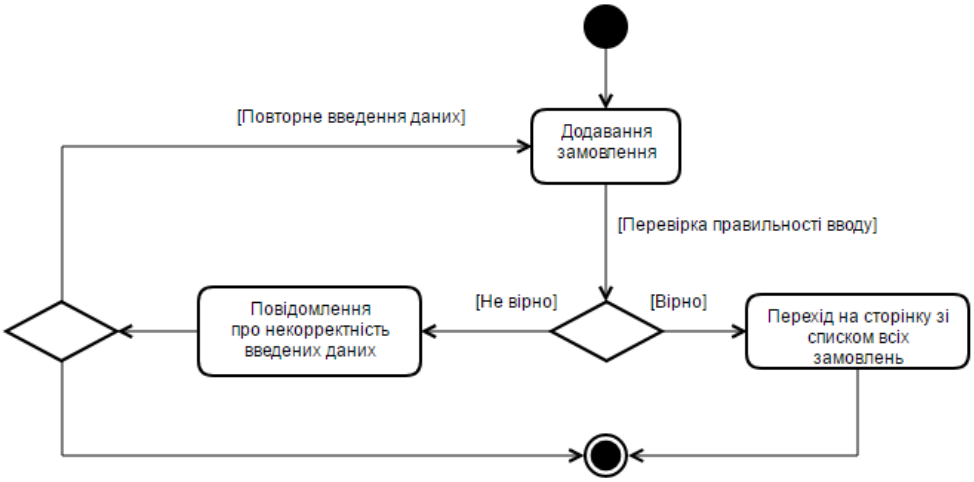

Рисунок 2.4 – Діаграма діяльності варіанту використання «Додання замовлення»

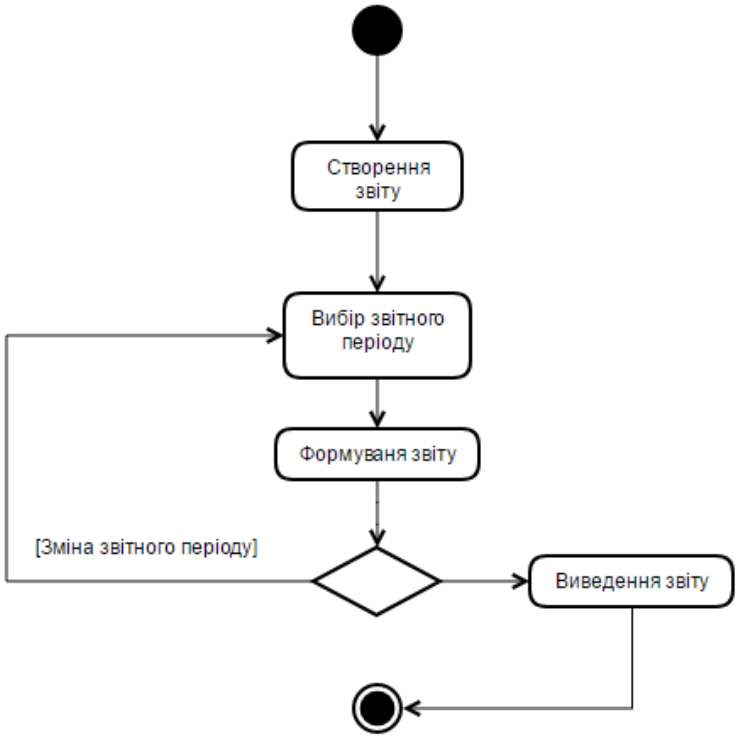

Рисунок 2.5 – Діаграма діяльності варіанту використання «Створення звіту»

Кожна із наведених діаграм діяльності має єдиний початковий стан та єдиний кінцевий стан. Зазвичай, діаграма діяльності розташовується так, що дії, які на ній представлені, слідують зверху вниз. Тоді початковий стан діаграми

діяльності зображуватиметься у верхній частині, а кінцевий – у нижній частині діаграми.

Розробка концептуальної моделі даних

Модель «сутність - зв'язок» (ER-модель, або ER-діаграма) – це така модель даних, що описує концептуальну схему даних за допомогою узагальнених блоків. Модель «сутність - зв'язок» являє собою засіб опису моделей даних, тобто мета-модель даних.

Взагалі для подання даних існує цілий ряд моделей. Однією з таких моделей для уніфікованого подання даних, є модель «сутність - зв'язок» (entity - relationship model, ER - model), яка є незалежною від програмного забезпечення, що далі реалізує цю модель [10].

Модель «сутність - зв'язок» базується на інформації про реальний світ та призначена для первинного логічного подання даних. Вона подає значення даних в контексті їх взаємозв'язку з іншими даними. Концептуальна модель даних програмного забезпечення для автоматизації роботи диспетчера служби таксі зображена на рисунку 2.6.

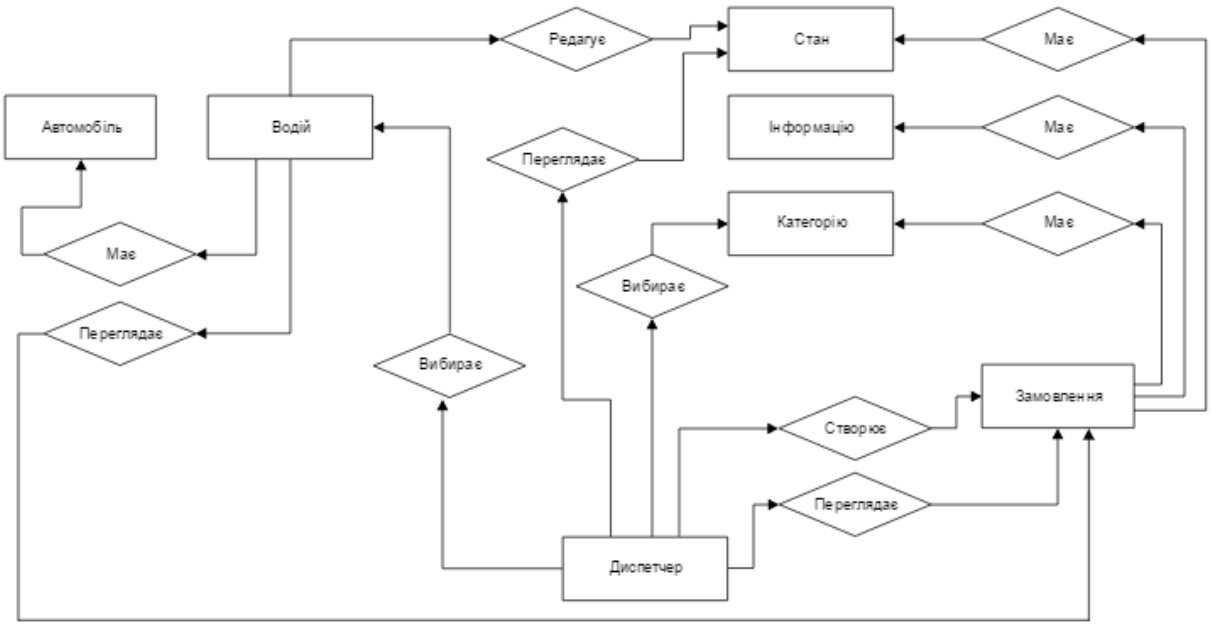

Рисунок 2.9 – Концептуальна модель програмного забезпечення для автоматизації роботи диспетчера служби таксі

ER - модель виявляється найбільш зручною при проектуванні по-перше, інформаційних систем та баз даних, а, по-друге, архітектур комп'ютерного програмного забезпечення. Ця модель допомагає виділити найсуттєвіші елементи моделі даних та встановити зв'язки між ними.

Розробка прототипу інтерфейсу користувача

Зазвичай, взаємодія між комп’ютером та користувачем відбувається у інтерфейсі користувача, який ще має назву Human-Computer Interaction, або скорочено HCI. Його основною метою є налагодження взаємодії між комп’ютером та користувачем, він робить комп’ютер більш сприйнятливим до потреб користувача.

Зазвичай, ефективність використання функціоналу, який закладено до програмного забезпечення, а також ефективність роботи з розробленим програмним забезпеченням визначається тим, як побудований інтерфейс користувача.

З урахуванням всіх вимог до інтерфейсу користувача було спроектовано ескізи майбутнього програмного забезпечення для автоматизації роботи диспетчера служби таксі, зображені на рисунках 2.7 – 2.8.

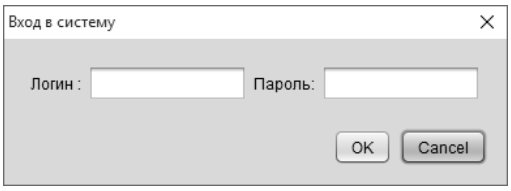

Рисунок 2.7 – Ескіз вікна авторизації програмного забезпечення для автоматизації роботи диспетчера служби таксі

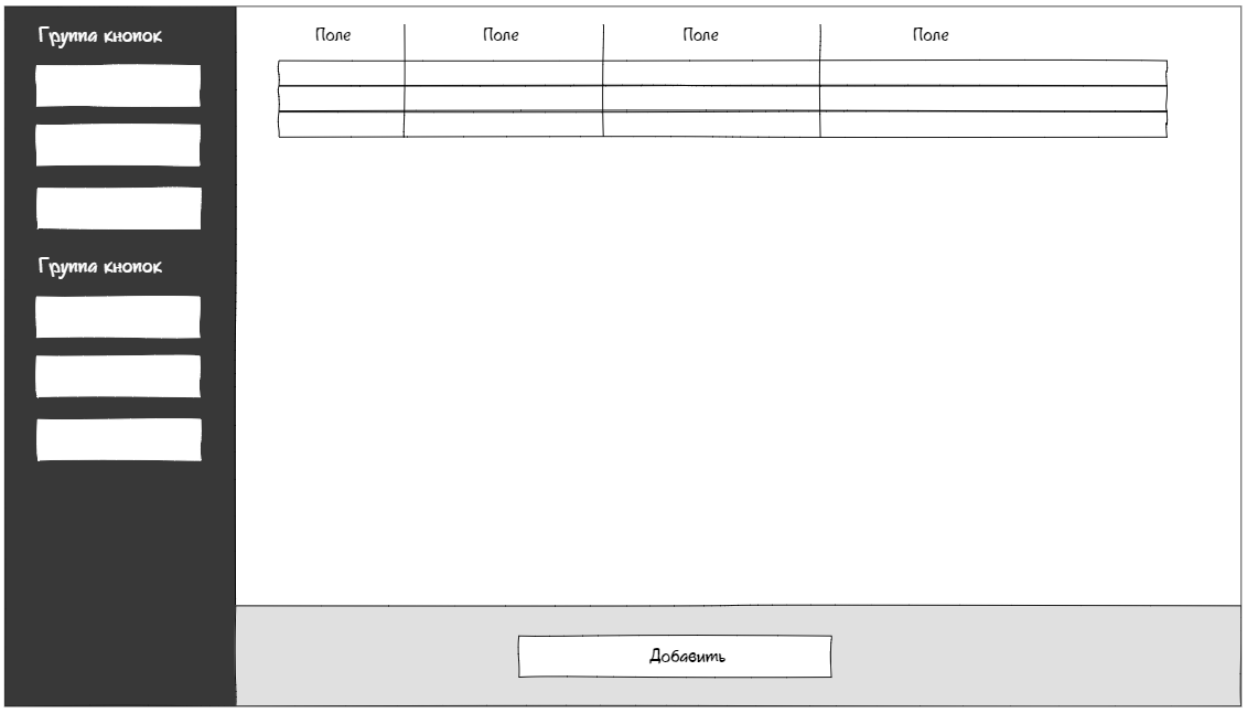

Рисунок 2.8 – Ескіз головного вікна програмного забезпечення для автоматизації роботи диспетчера служби таксі

Розроблений прототип інтерфейсу програмного забезпечення задовольняє поставленим вимогам та забезпечить зручну роботу користувачам.

2.3.2 Технічний проект програмного забезпечення

Побудова діаграми класів

Діаграма класів програмного забезпеченя являє собою статичну діаграму, на якій представлена сукупність декларативних або статичних елементів програмного забезпечення, таких як, класи та відношення, що їх з’єднують. Класи позначаються з їх атрибутами та операціями [11]. Діаграми класі реалізації деяких основних варіантів використання програмного забезпечення для автоматизації роботи диспетчера служби таксі наведено на рисунках 2.9 – 2.10.

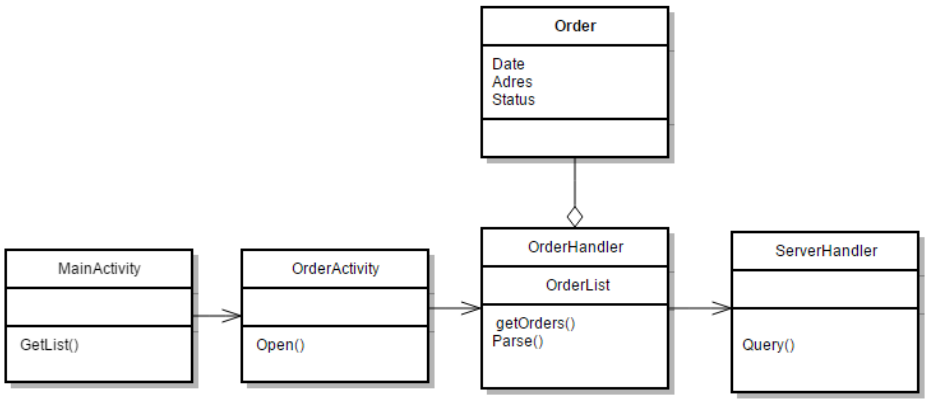

Рисунок 2.9 – Діаграма класів реалізації варіанту використання «Перегляд всіх замовлень»

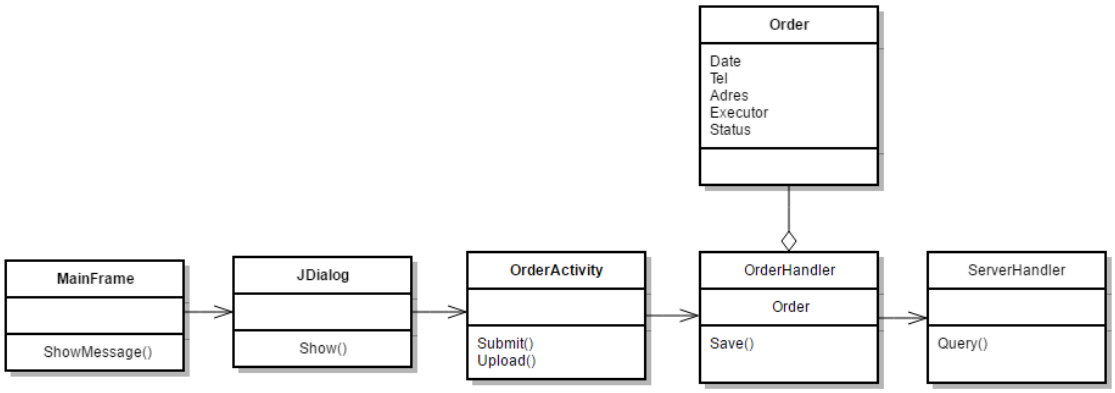

Рисунок 2.13 – Діаграма класів реалізації варіанту використання «Додавання замовлення»

Діаграма класів програмного забезпеченя, що розробляється, призначена для подання статичної структури цього програмного забезпечення в термінології об’єктно-орієнтованого програмування. Однак на цій діаграмі не наводиться інформація про часові аспекти функціонування програмного забезпечення.

Побудова діаграми взаємодії

Діаграма взаємодії – це одна з UML-діаграм, що використовується для опису поведінки груп об'єктів, які взаємодфють між собою. Діаграми взаємодії реалізації деяких основних варіантів використання програмного забезпечення

для автоматизації роботи диспетчера служби таксі наведено на рисунках 2.11 – 2.12.

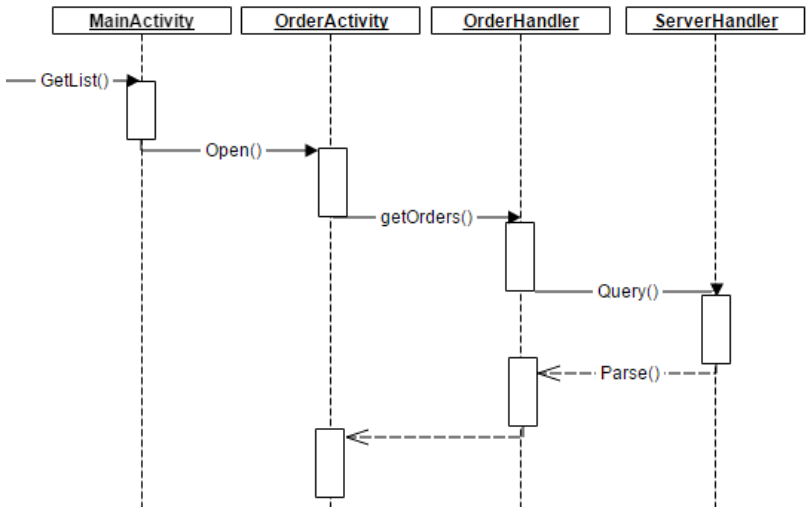

Рисунок 2.11 – Діаграма взаємодії реалізації варіанту використання «Перегляд всіх замовлень»

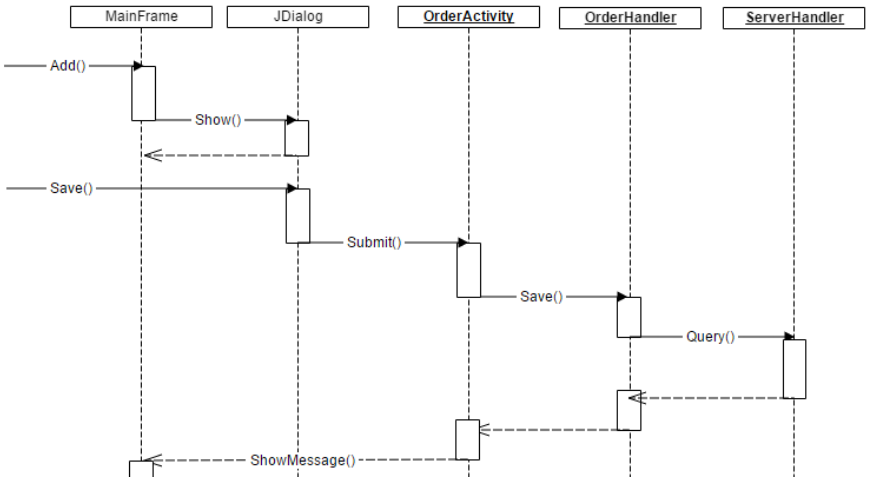

Рисунок 2.12 – Діаграма взаємодії реалізації варіанту використання «Додавання замовлення»

За звичай, кожна діаграма взаємодії демонструє поведінку об`єктів в межах одного варіанту використання. Така діаграма відображає екземпляри об'єктів та повідомлення, якими ці об'єкти обмінюються між собою у межах одного варіанту використання [8].

Розробка логічної моделі бази даних

Логічна модель бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі представлена на рисунку 2.13. Ця модель є початковим прототипом бази даних і описує на логічному рівні її інформаційний зміст. Логічна модель описує всю базу даних як єдине ціле і будується без прив'язки до конкретної СУБД.

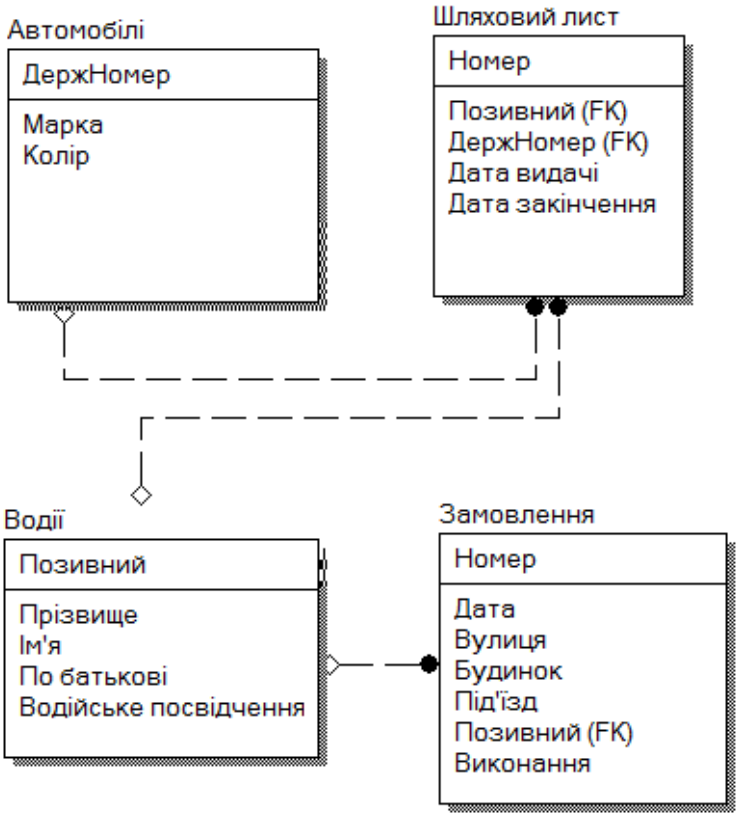

Рисунок 2.10 – Логічна модель бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі

Опис типів даних атрибутів сутностей логічної моделі бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі представлено у таблицях 2.7 – 2.10

Таблиця 2.7 – Опис атрибутів сутності «Автомобілі»

Ім’я	Тип
Держ. номер	Строкові дані
Марка	Строкові дані
Колір	Строкові дані

Таблиця 2.8 – Опис атрибутів сутності «Водії»

Ім’я	Тип
Позивний	Строкові дані
Прізвище	Строкові дані
Ім'я	Строкові дані
По батькові	Строкові дані
Водійське посвідчення	Строкові дані

Таблиця 2.9 – Опис атрибутів сутності «Шляховий лист»

Ім’я	Тип
Номер	Цілий
Позивний	Строкові дані
Держ. Номер	Строкові дані
Дата видачі	Дата/час
Дата закінчення	Дата/час

Таблица 2.10 – Опис атрибутів сутності «Замовлення»

Ім’я	Тип
Номер	Цілий
Дата	Дата/час
Вулиця	Строкові дані
Будинок	Строкові дані
Позивний	Строкові дані
Виконання	Строкові дані

Логічна модель бази даних відображає узагальнений погляд на вихідні дан в конкретній предметній галузі, уточненням логічної моделі є фізична модель бази даних.

2.3.3 Робочий проект програмного забезпечення

Вибір інструментарію розробки

Для реалізації програмного забезпечення для автоматизації роботи диспетчера служби таксі було обрано мову програмування Java, тому що вона є зручною для побудови мережевих додатків та забезпечує незалежність від платформи.

Мова програмування Java відноситься до групи об'єктно-орієнтованих мов програмування. Була розроблена як основний компонент платформи Java компанією «Sun Microsystems» у 1995 році. Компанія «Oracle» в 2009 році придбала компанію «Sun Microsystems» і з цього року підтримує та розвиває мову Java. Особливістю, яка підтримує платформну незалежність мови Java є те, що програми, написані мовою Java, компілюються у байт-код, який далі при виконанні цієї програми інтерпретується віртуальною Java-машиною вже для конкретної платформи.

Мова Java призначена в тому числі для створення програмного забезпечення, яке працює на базі протоколів TCP/IP в розподіленому Internet-середовищі. Крім того, в програмі, яка написана мовою Java, присутня можливість передачі повідомлень в межах внутрішнього адресного простору. Все це дозволяє забезпечити віддалене виконання процедур. Наявність цих можливостей привносить високий рівень абстракції в розробку програмного забезпечення, яке призначене для клієнт-серверного середовища [12].

Система, що розробляється, має клієнт-серверну архітектуру. Сервер має забезпечити швидкий обмін даними між пристроями на яких встановлені додатки диспетчера, при цьому він повинен бути не надто вимогливим до апаратного забезпечення. Для задоволення цих вимог було обрано серверну платформу Node.js.

Серверна платформа Node.js являє собою платформу з відкритим вихідним кодом для виконання високопродуктивних мережевих застосунків, розроблених з використанням мови програмуванн JavaScript. Використання

серверної платформи node.js перетворило мову програмування JavaScript з мови, яка використовувалась в основному в браузерах на мову загального використання, яка вже має велику спільноту розробників.

Серверна платформа Node.js характеризується такими властивостями як: асинхронна однопотокова модель виконання запитів; неблокуючий ввід/вивід; система модулів CommonJS; рушій JavaScript Google.

Для керування модулями використовується пакетний менеджер node package manager. Серверна платформа Node.js призначена для відокремленого виконання високопродуктивних мережних застосунків, розроблених мовою JavaScript [13].

JavaScript – динамічна, об'єктно-орієнтована мова програмування. Зазвичай використовується як частина браузера та надає можливість виконання програмного коду на стороні клієнта, наприклад, взаємодіяти з користувачем, керувати браузером, обмінюватися даними з сервером в асинхронном режимі, динамічно змінювати подання веб-сторінки. Мова програмування JavaScript також може використовуватися для програмування на стороні сервера, розробки ігор, стаціонарних та мобільних програмних застосунків, сценаріїв в прикладному програмному забезпеченні, наприклад, всередині документів формату .pdf.

Мова програмування JavaScript має властивості об'єктно-орієнтованої мови програмування, але підтримка об'єктів в ній дещо відрізняється від традиційних мов об'єктно-орієнтованого програмування завдяки концепції прототипів. До того ж, мова програмування JavaScript має і ряд властивостей, притаманних функціональним мовам програмування: функції як об'єкти першого класу, об'єкти як списки, анонімні функції і таке інше. Все це додає мові JavaScript додаткової гнучкості [14].

MySQL представляє собою систему управління реляційними базами даних, дані у яких зберігаються в окремих таблицях, що дозволяє досягти значного виграшу у швидкості та гнучкості. Таблиці у такій базі даних зв'язуються між собою за допомогою відносин по ключовим полям, що

забезпечує можливість поєднання даних з декількох таблиць при виконанні запитів до бази даних. Мову SQL як частина СУБД MySQL можна характеризувати як мову структурованих запитів, яка використовується для доступу до даних.

СУБД MySQL являє собою програмне забезпечення з відкритим кодом, тобто застосовувати та модифікувати його під свої потреби може будь-який розробник програмного забезпечення. СУБД MySQL можна отримати за допомогою Internet та використовувати безоплатно [15].

Розробка фізичної моделі бази даних

Фізична модель бази даних описує дані засобами конкретної СУБД і показує спосіб їх зберігання в базі.

Фізична модель бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі зображена на рисунку 2.14.

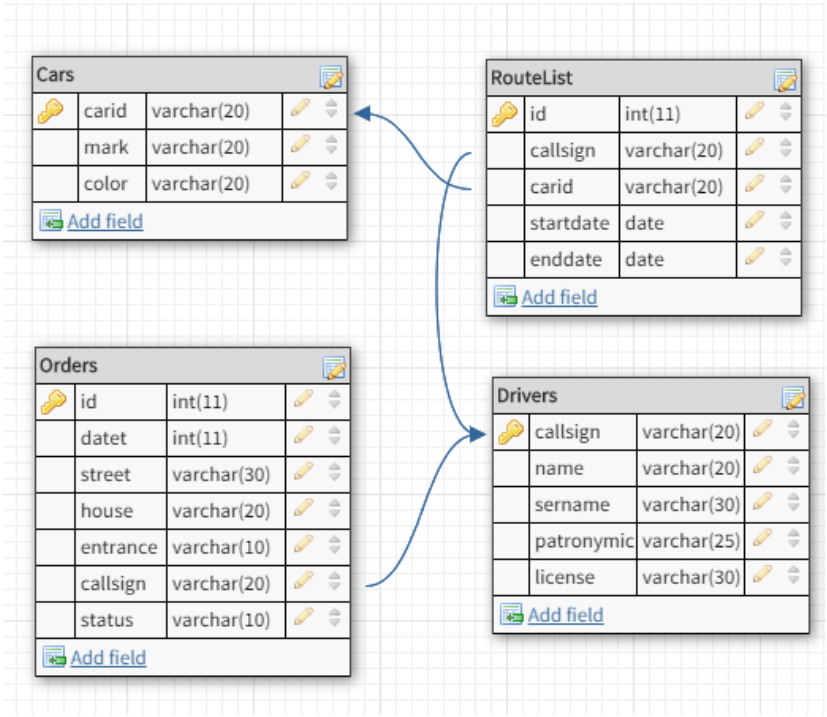

Рисунок 2.14 – Фізична модель бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі

Нижче представлено опис таблиць фізичної моделі бази даних та їх відповідність логічній (див. табл. 2.11 – 2.14).

Таблиця 2.11 – Опис атрибутів таблиці «Автомобілі»

Ім’я	Тип	NULL	Ключ
carid	varchar(20)	No	PK
mark	varchar(20)	No	–
color	varchar(20)	No	–

Таблиця 2.12 – Опис атрибутів таблиці «Водії»

Ім’я	Тип	NULL	Ключ
callsign	varchar(20)	No	PK
name	varchar(20)	No	–
sername	varchar(20)	No	–
patronymic	varchar(20)	No	–
license	varchar(20)	No	–

Таблиця 2.13 – Опис атрибутів таблиці «Шляховий лист»

Ім’я	Тип	NULL	Ключ
id	int(11)	No	PK
callsign	varchar(20)	No	FK
carid	varchar(20)	No	FK
startdate	date	No	–
enddate	date	No	–
entrance	varchar(10)	No	–
callsign	varchar(20)	No	FK
status	varchar(10)	No	–

Таблиця 2.14 – Опис атрибутів таблиці «Замовлення»

Ім’я	Тип	NULL	Ключ
id	int(11)	No	PK
date	date	No	–
street	varchar(30)	No	–
house	varchar(20)	No	–

Розроблена фізична модель бази даних програмного забезпечення для автоматизації роботи диспетчера служби таксі дозволить легко реалізувати базу даних для застосування у програмному забезпеченні.

Тестування програмного забезпечення

Під час тестування було проведено повне функціональне тестування, а також навантажувальне тестування і тестування на відмову всього програмно-апаратного комплексу.

Всі виявлені недоліки програмного забезпечення були зафіксовані в протоколах і усунуті до моменту впровадження програмного забезпечення у дію.

Методи тестування

Тестування було проведено за стратегією «чорного ящика». Через велику кількості функцій, які потрібно було випробовувати, представлено результати випробувань основних функцій додатку диспетчера служби таксі.

Протокол тестування

1. Проведено перевірку програми на відповідність технічному завданню:

– перевірка роботи функції «Авторизація»

Після запуску програми у вікні авторизації було введено логін і пароль, після чого було успішно пройдено авторизацію у додатку.

– перевірка роботи функції «Перегляд списку всіх замовлень»

Після проходження авторизації на головному вікні натиснули кнопку «Управление заказами». З’явилась таблиця, що відображає список всіх замовлень у системі.

– перевірка роботи функції «Додавання нового замовлення»

Після проходження авторизації на головному вікні натиснули на кнопку «Управление заказами». Після чого натиснули на кнопку «Добавить заказ». У вікні додавання замовлення було введено інформацію про замовлення, після чого було перевірено правильність адресу та додано замовлення в систему шляхом натискання кнопки «Добавить».

– перевірка роботи функції «Перегляд списку водіїв»

Після проходження авторизації на головному вікні натиснули кнопку «Управление водителями». З’явилась таблиця, яка відображає список всіх водіїв у системі.

– перевірка роботи функції «Реєстрація водія»

Після проходження авторизації на головному вікні натиснули на кнопку «Управление водителями». Після чого натиснули на кнопку «Регистрация». У вікні реєстрації було введено інформацію про водія, після додано в систему шляхом натискання кнопки «Ok».

– перевірка роботи функції «Формування звітів»

Після проходження авторизації на головному вікні натиснули на кнопку «Отчеты». Після чого вибрали необхідний тип звіту. Далі отримали необхідний звіт.

2. Проведено перевірку програми на її програмно-апаратну сумісність: виконано тестування на апаратно-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені в технічному завданні; виконано перевірку на наявність і усунення всіх помилок, зазначених у п.1.

3. Проведено візуальний перегляд програми: виконано остаточне налагоджено на предмет виявлення та усунення дрібних помилок.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ ДИСПЕТЧЕРА СЛУЖБИ ТАКСІ

Результатом кваліфікаційної роботи є розв’язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов - розроблене програмне забезпечення для автоматизації роботи диспетчера служби таксі. Було проаналізовано практичну задачу інженерії програмного забезпечення з теми роботи, а саме проаналізовано організаційну структуру служби таксі та визначено специфіку її роботи з точки зору інформаційних технологій, які застосовуються, розглянуто існуючі програмні рішення та здійснено постановку задачі на розробку програмного забезпечення.

Виконано аналіз вимог до програмного забезпечення для автоматизації роботи диспетчера служби таксі. Розроблена архітектура програмного забезпечення для автоматизації роботи диспетчера служби таксі. Розроблені ескізний, технічний та робочий проєкти програмного забезпечення для автоматизації роботи диспетчера служби таксі. Здійснено тестування та випробування розробленого програмного забезпечення.

Згідно з вимогами до програмного забезпечення, наведеними у технічному завданні (Додаток А), програмне забезпечення надає такі функції:

- авторизація користувача;

– реєстрація нового водія;

– перегляд списку водіїв;

– автоматичне додавання нового замовлення;

– перегляд списку замовлень;

– автоматичне формування звітів.

Вікно "Управління заказами" програмного забезпечення для автоматизації роботи диспетчера служби таксі наведено на рис. 3.1.

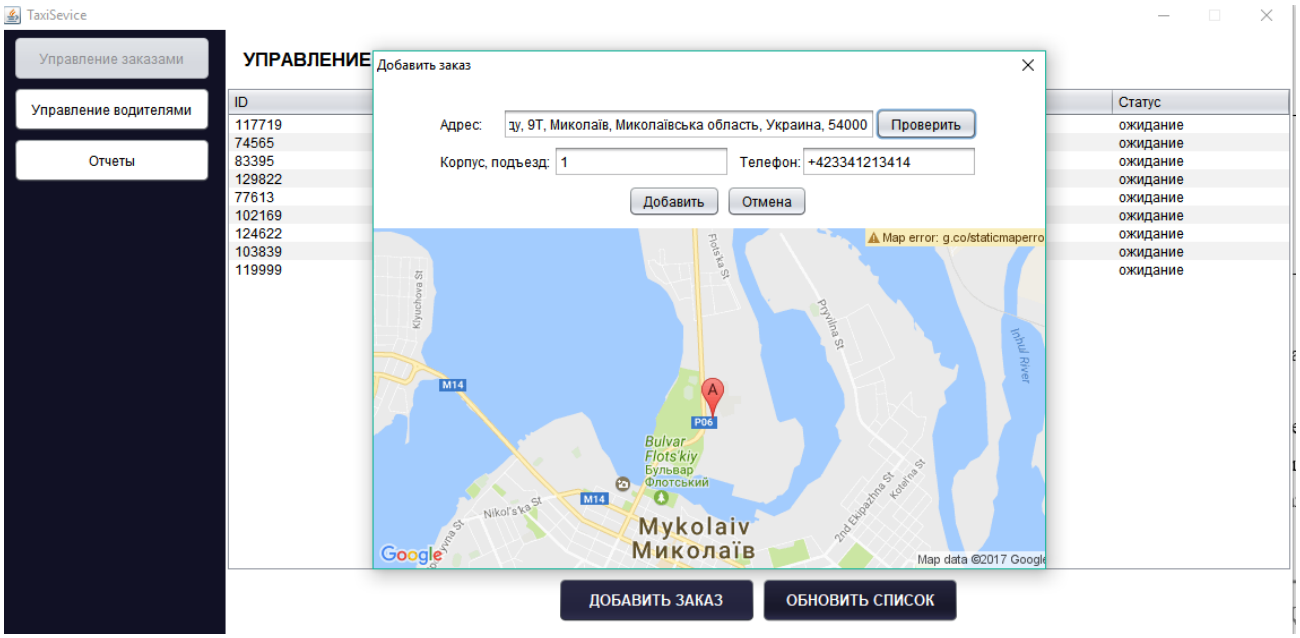

Рисунок 3.1 – Вікно "Управління заказами" програмного забезпечення для автоматизації роботи диспетчера служби таксі

Програмне забезпечення повністю відповідає вимогам до організації вхідних та вихідних даних, вимогам до надійності та іншим вимогам, зазначеним в технічному завданні.

Також була розроблена наступна програмна документація:

- технічне завдання (Додаток А);
- текст програми (Додаток Б);
- опис програми (Додаток В)
- інструкція користувача (Додаток Г);
- програма та методика випробування програмного забезпечення (Додаток Д).

Програмне забезпечення для автоматизації роботи диспетчера служби таксі реалізовано за допомогою середовища розробки IntelliJ IDEA, програмне забезпечення було написане з використанням технологій NodeJS, Java, JavaScript. У якості СУБД було обрано MySQL.

Розроблене програмне забезпечення для автоматизації роботи диспетчера служби таксі було протестоване на відповідність технічному завданню. Під час тестування програмне забезпечення показало повну працездатність.

Розроблене програмне забезпечення для автоматизації роботи диспетчера служби таксі може бути використано користувачем, не здатним до програмування. Інтерфейс програмного забезпеченняє інтуїтивно зрозумілим.

4 ОХОРОНА ПРАЦІ

4.1 Вступ

Охорона праці – це система правових, соціально-економічних, організаційно-технічних, санітарно-гігієнічних і лікувально-профілактичних заходів та засобів, спрямованих на збереження життя, здоров'я і працездатності людини у процесі трудової діяльності [16].

Метою політики охорони праці е зведення до мінімуму показників виробничого травматизму та професійних захворювань. Ця мета набула нових форм у ЄС протягом останніх років і поширилася сьогодні до пропаганди "добробуту на роботі", що означає моральний, фізичний та соціальний добробут, а не лише відсутність нещасних випадків та професійних захворювань.

Крім того, необхідно також досягти низки допоміжних цілей:

- профілактика соціальних ризиків (стресів, домагань на робочому місці, депресій та роздратування, а також ризиків, які пов'язані з алкогольною, наркотичною залежністю);

- аналіз ризиків, пов'язаних із роботою, а також ергономічні, психологічні та соціальні ризики;

- урахування змін у формах зайнятості, організації роботи та робочого часу працівників з нестандартною та тимчасовою зайнятістю;

- урахування розмірів підприємства (конкретні заходи щодо інформування, підвищення рівня обізнаності, програм попередження ризиків на малих та середніх підприємствах, приватних підприємців, домашньої обслуги тощо);

- інтенсивна профілактика професійних захворювань (спричинених азбестом, втрата слуху, проблеми опорно-рухового апарату);

- урахування демографічних змін.

Основними завданнями управління охороною праці є:

1) опрацювання заходів щодо здійснення державної політики з охорони праці на регіональному та галузевому рівнях;

2) підготовка, прийняття та реалізація заходів, спрямованих на забезпечення:

- належних, безпечних і здорових умов праці;

- утримання в належному стані виробничого устаткування, будівель і споруд, інженерних мереж, безпечного ведення технологічних процесів;

- необхідних засобів індивідуального захисту для працівників;

- організації і проведення навчання працівників з питань охорони праці;

- пропаганди охорони праці;

- обліку, аналізу та оцінки стану умов і безпеки праці;

- професійного добору працівників окремих спеціальностей;

- страхування працівників від нещасного випадку на виробництві та профзахворювань;

3) організаційно-методичне керівництво на регіональному та галузевому рівнях;

4) стимулювання інтеграції управління охороною праці в єдину систему загального управління організацією виробництва;

5) широке впровадження позитивного досвіду у сфері охорони праці.

За умови економічної, екологічної та демографічної кризи в Україні, подій на Сході України, склалася надзвичайна ситуація з безпекою та умовами праці на більшості підприємств, особливо середнього і малого бізнесу [17].

Таку оцінку ФПУ оприлюднила у другій національній профспілковій доповіді Президенту України, у проекті Стратегії поліпшення стану охорони праці в Україні, Концепції Загальнодержавної програми поліпшення стану безпеки, гігієни праці та виробничого середовища.

4.2 Аналіз шкідливих та небезпечних факторів при роботі з персональним комп’ютером

До фізичних шкідливих і небезпечних чинників відносяться:

- підвищені рівні електромагнітного, рентгенівського, ультрафіолетового і інфрачервоного випромінювання;
- підвищений рівень статичної електрики і запилене повітря робочої зони;
- підвищений вміст позитивних іонів і понижений вміст негативних іонів в повітрі робочої зони;
- підвищений рівень яскравості;
- нерівномірність розподілу яскравості в полі зору;
- підвищене значення напруги в електричному ланцюзі, замикання якого може статися через тіло людини.

Хімічні шкідливі і небезпечні чинники наступні: підвищений вміст в повітрі робочої зони двоокису вуглецю, озону, аміаку, фенолу і формальдегіду.

Психофізіологічні шкідливі і небезпечні фактори:

- напруга зору і уваги;
- інтелектуальні, емоційні і тривалі статичні навантаження;
- монотонність праці;
- великий обсяг інформації, що обробляється в одиницю часу;
- нераціональна організація робочого місця.

Типовими відчуттями, які відчувають до кінця робочого дня оператори персональних комп’ютерів, є: перевтома очей, головний біль, тягучий біль в м'язах шиї, рук і спини, зниження концентрації уваги.

Уже в перші роки комп'ютеризації було відзначено специфічне зорове стомлення у користувачів дисплеїв, що отримало загальну назву «комп'ютерний зоровий синдром». Однією з причин є те, що сформована за мільйони років еволюції зорова система людини пристосована для сприйняття об'єктів у відбитому світлі (друковані тексти, малюнки тощо), а не для роботи за дисплеєм. Зображення на дисплеї принципово відрізняється від звичних оку

об'єктів спостереження – воно світиться, мерехтить, складається з дискретних точок, а кольорове комп'ютерне зображення не відповідає природним кольорам. Але не тільки особливості зображення на екрані викликають зорове стомлення. Велике навантаження орган зору відчуває при введенні інформації, так як користувач змушений часто переводити погляд з екрану на текст і клавіатуру, що знаходяться на різній відстані і по-різному освітлені. Зорове стомлення проявляється скаргами на затуманення зору, труднощі при перенесенні погляду з ближніх предметів на дальні і з далеких на ближні, що здаються зміни забарвлення предметів, їх двоїння, відчуття печіння, «піску» в очах, почервоніння повік, болі при рухanні очима.

Тривала і інтенсивна робота на комп'ютері може стати джерелом важких професійних захворювань, таких, як травма повторюваних навантажень (ТПН), що представляє собою поступове накопичення нездужань, котрі переходять у захворювання нервів, м'язів і сухожиль руки.

До професійних захворювань, пов'язаних з ТПН, відносяться:

- тендовагініт – запалення сухожиль кисті, зап'ястя, плеча;

- тендосиновіт – запалення синовіальної оболонки сухожильного основи кисті і зап'ястя;

- синдром зап'ястного каналу (СЗК) – викликається утиском серединного нерва в зап'ястному каналі. Травма викликає утворення продуктів розпаду в області зап'ястного каналу, в результаті чого спочатку виникає набряк, а потім СЗК.

З'являються скарги на пекучий біль і поколювання в зап'ясті, долоні, а також пальцях, крім мізинця. Спостерігається хворобливість і оніміння, ослаблення м'язів, що забезпечують рух великого пальця.

Ці захворювання зазвичай настають в результаті безперервної роботи на неправильно організованому робочому місці.

Механізм порушень, що відбуваються в організмі під впливом електромагнітних полів, обумовлений їх специфічною (нетепловою) і тепловою дією.

4.3 Розробка заходів щодо зменшення впливу шкідливих та небезпечних факторів при роботі з персональним комп’ютером

Розглянемо детальніше чинники, котрі призводять до проблем із зором при роботі за персональним комп’ютером.

Основні чинники, які призводять до виникнення проблем із зоровим апаратом:

- недостатнє освітлення робочого місця;
- високий контраст між монітором та оточуючим середовищем.
- відблиски на моніторі;
- мала дистанція між користувачем на монітором;
- мала частота кліпання повіками.

Для покращення освітлення робочого місця зазвичай використовують одну настільну лампу – лампу розжарювання чи компактну ртутну ЛДС, рідше світлодіодну. Але одне джерело світла забезпечує нерівномірне освітлення робочого місця та часто створює відблиски на екрані монітору. Щоб забезпечити рівномірне освітлення також використовують два когерентні джерела світла, наприклад: дві ртутні ЛДС, частоти випромінювання яких є когерентні і через це, вони посилюють одна одну. Але приведений метод, дозволяє тільки частково зменшити контраст між працюючим екраном та оточуючим середовищем і є не економічним.

Принцип дії запропонованого методу досить простий і полягає в розміщенні 4 напрямлених перпендикулярно до користувача джерел світла на корпусі монітору для тилового підсвічування. В якості джерел світла були використані світлодіодні стрічки. Колір випромінювання був обраний спектрально-оптимальний – білий «теплий» (2700–3200 К). А залежно від моделі LED, мають кут розсіювання від 30 до 270 градусів, що дозволяє використовувати їх як для акцентного підсвічування, так і для загального

освітлення. Варто також зауважити, що світлодіоди не несуть ніякої загрози навколишньому середовищу.

Проведено експеримент з фотоапаратом та опитування серед програмістів. За допомогою цифрового фотоапарата було створено серію фотознімків у «ручному» режимі, тобто без автоматичного налаштування яскравості, витримки та балансу білого, з тиловою підсвіткою та без неї. Дане рішення дозволяє помітно зменшити контраст, різкий перепад яскравості; завдяки використання LED-стрічки – отримати мініатюрні габарити, низьку собівартість, високу тривалість роботи та безпечність джерел освітлення.

Вимоги безпеки під час роботи з комп’ютером

Щодня перед початком роботи оператор повинен:

- оглянути своє робоче місце;

- про виявлення ознак пошкодження обладнання інформувати свого безпосереднього керівника;

- відрегулювати освітленість на робочому місці, переконатися в відсутності відблисків на екрані комп’ютера, відсутності зустрічного світла;

- перевірити правильність підключення обладнання ЕОМ до електромережі;

- очистити екран комп’ютера від пилу та інших забруднень;

- перевірити правильність організації робочого місця й за необхідності провести відповідні корегування.

Оператор під час роботи зобов'язаний:

- виконувати тільки ту роботу, яку йому було доручено;

- підтримувати порядок і чистоту на робочому місці;

- тримати відкритими всі вентиляційні отвори обладнання;

- коректно закрити всі активні завдання у разі припинення роботи з комп’ютером;

- негайно відключити комп’ютером від електричної мережі у разі виникнення аварійної ситуації.

У ході виконання робіт оператор комп’ютера повинен:

- витримувати відстань від очей до екрана комп’ютером в межах 60-70 см;

- дотримуватися змінного режиму праці та відпочинку, регламентованих перерв у роботі;

- для розробників програм – тривалістю 15 хвилин через кожну годину роботи;

- для інших категорій працівників – тривалістю 15 хвилин через кожні дві години роботи;

- для операторів комп'ютерного набору – тривалістю 10 хвилин, після кожної години роботи.

Під час регламентованих перерв рекомендується виконувати комплекси вправ для очей, рук, хребта, поліпшення мозкового кровообігу тощо. Про виявлення несправності обладнання або інших факторів, які створюють загрозу для життя або здоров'я працівників, необхідно негайно інформувати свого безпосереднього керівника.

Не допускається:

- виконання ремонту та налагодження комп’ютерної техніки безпосередньо на робочому місці оператора;

- зберігання біля комп’ютера паперу, дискет, інших носіїв інформації, запасних блоків, деталей тощо, якщо вони не використовуються для поточної роботи;

- відключення захисних пристроїв, самочинні зміни в конструкції комп’ютера;

- використання комп’ютерів, на екранах яких під час роботи з'являються нехарактерні сигнали, нестабільне зображення на екрані тощо;

- доторкання до задньої панелі системного блоку при включеному живленні;

- вимикання живлення під час виконання активного завдання;

- попадання вологи на поверхню системного блоку, монітора, клавіатури, дисководів, принтерів та інших пристроїв;

- приймання напоїв та їжі на робочому місці.

Після закінчення роботи з використанням необхідно дотримуватися такої послідовності вимикання обладнання:

- закрити всі активні завдання;

- використавши опцію "Завершення роботи" у меню "Пуск", вимкнути живлення системного блоку;

- вимкнути живлення всіх комп’ютерів;

- вимкнути блок аварійного живлення (за наявності);

- відключити комп’ютер від електромережі, при цьому забороняється тягнути штепсельну вилку за дріт.

У випадку виникнення аварійної ситуації оператор зобов'язаний:

- у всіх випадках виявлення пошкодження проводів електричного живлення, несправності заземлення та інших пошкодженнях електрообладнання, виникненні запаху гарі, диму - негайно вимкнути електричне живлення і повідомити про аварійну ситуацію свого безпосереднього керівника й чергового електрика;

- при попаданні людини під електричну напругу негайно звільнити її від дії струму шляхом вимкнення електричного живлення, до прибуття лікаря надати потерпілому долікарську медичну допомогу;

- при будь-яких випадках порушень роботи технічного обладнання або програмного забезпечення негайно викликати представника технічної служби з питань експлуатації обчислювальної техніки;

- у випадку виникнення різі в очах, різкого погіршення зору, виникнення головного болю, больових відчуттів у пальцях та кистях рук, посилення серцебиття - негайно припинити роботу з використанням ЕОМ, повідомити про те, що сталося, свого безпосереднього керівника й звернутися до медичної установи;

- при загорянні обладнання негайно відключити його від електромережі;

- про загорання повідомити свого безпосереднього керівника, оперативного чергового, пожежну службу; ужити заходів щодо ліквідації вогню за допомогою вуглекислотного або порошкового вогнегасника.

ВИСНОВКИ

В результаті кваліфікаційної роботи було розв`язано практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов та розроблено програмне забезпечення для автоматизації роботи диспетчера служби таксі. Розроблене програмне забезпечення задовольняє поставленим вимогам, надає можливість здійснювати контроль над процесом виконання замовлень, спрощує документообіг.

Для досягнення поставленої мети було виконано аналіз практичної задачі інженерії програмного забезпечення, що стосується теми роботи, проаналізовано організаційну структуру служби таксі та визначено специфіку її роботи з точки зору інформаційних технологій, які застосовуються, розглянуто існуючі програмні рішення, здійснено постановку задачі на розробку програмного забезпечення. Виконано аналіз вимог до програмного забезпечення для автоматизації роботи диспетчера служби таксі. Розроблена архітектура програмного забезпечення для автоматизації роботи диспетчера служби таксі. Розроблені ескізний, технічний та робочий проєкти програмного забезпечення для автоматизації роботи диспетчера служби таксі. Здійснено тестування та випробування розробленого програмного забезпечення, які показали його працездатність та відповідність поставленим вимогам. Було розроблено документацію до програмного забезпечення.

Також в кваліфікаційній роботі були розглянуті спеціальні питання з охорони праці.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Fierro A., Laffont R. Histoire et Dictionnaire de Paris. 2007. 1166 p. ISBN 2-221-07862-4

2. Про автомобільний транспорт: Закон України від 05.04.2001 р. № 2344-III. Дата оновлення: 23.03.2023. URL: https://zakon.rada.gov.ua/laws/show/2344-14#Text (дата звернення: 30.03.2023).

3. АРМ диспетчера таксі URL: https://www.updatestar.com/ru/detail/арм-диспетчер-такси-4-0b (дата звернення: 30.03.2023).

4. Тема: TaxiAdmin. Спустя полтора года использования. URL: https://forum.taxiservice.com.ua/taksi-admin/taxiadmin-spustya-poltora-goda-ispolzovaniya/ (дата звернення: 30.03.2023).

5. Организация диспетчерской такси на программе e(taxi). URL: http://www.etaxi.ua/ (дата звернення: 30.03.2023).

6. EVOS. If you dream - Dream big. URL: http://www.evos.in.ua/ (дата звернення: 30.03.2023).

7. Mobile Taxi (Евос). Обсуждение программы Mobile Taxi. Тонкости, настройки, обновления и другие вопросы связанные с Mobile Taxi. URL: https://forum.taxiservice.com.ua/taksi-navigator-(evos)/ (дата звернення: 30.03.2023).

8. Діаграми UML для моделювання процесів і архітектури ПЗ. URL: https://evergreens.com.ua/ua/articles/uml-diagrams.html (дата звернення: 15.04.2023).

9. Соммервилл И. Инженерия программного обеспечения. Вильямс, 2002. 624 с.

10. Елементи моделі «сутність-зв'язок». URL: http://easy-code.com.ua/2012/09/elementi-modeli-sutnist-zvyazok-integraciya-dodatkiv-i-danix-bazi-danix-statti/ (дата звернення: 15.04.2023).

11. Діаграма класів. URL:

http://moodle.ipo.kpi.ua/moodle/mod/resource/view.php?id=28088 (дата звернення: 15.04.2023).

12. Oracle Java is the #1 programming language and development platform. URL: https://www.oracle.com/uk/java/ (дата звернення: 10.05.2023).

13. Node.js® is an open-source, cross-platform JavaScript runtime environment. URL: https://nodejs.org/en (дата звернення: 10.05.2023).

14. JavaScript - Dynamic client-side scripting. URL: https://developer.mozilla.org/en-US/docs/Learn/JavaScript (дата звернення: 10.05.2023).

15. MySQL. The world's most popular open source database. URL:https://www.mysql.com/ (дата звернення: 10.05.2023).

16. Про охорону праці: Закон України від 14.10.1992 р. № 2694-XII. Дата оновлення: 31.03.2023. URL: https://zakon.rada.gov.ua/laws/main/2694-12#Text. (дата звернення: 10.05.2023).

17. Поняття, мета і завдання охорони праці. URL: http://ncpn.net.ua/oxrana_truda.html (дата звернення: 10.05.2023).

Додаток А

Технічне завдання

Вступ

Назва розроблюваного проекту: «Програмне забезпечення для автоматизації роботи диспетчера служби таксі». Область застосування – підприємство, яке надає послуги таксі.

1. Підстави для розробки

Підставою для розробки даного програмного забезпечення є наказ на затвердження тем кваліфікаційних робіт бакалаврів зі спеціальності 121 Інженерія програмного забезпечення в Національному університеті кораблебудування ім. адмірала Макарова.

2. Призначення розробки

2.1 Експлуатаційне призначення

Експлуатаційним призначенням програмного забезпечення є спрощення бізнес-процесів служби таксі, покращення та легкість ведення та обробки даних, полегшення комунікації між працівниками.

2.2 Функціональне призначення програми

Функціональним призначенням програмного забезпечення є автоматизація процесів:

- реєстрації та авторизації користувачів;
- прийому та передачі замовлень виконавцю;
- контролю за ходом виконання замовлень;
- складання звітності.

3. Вимоги до програмного забезпечення

3.1 Вимоги до функціональних характеристик

3.3.1 Вимоги до складу виконуваних функцій

Програмне забезпечення повинно виконувати наступні функції:

– авторизація користувача;

– реєстрація нового водія;

– перегляд списку водіїв;

– автоматичне додавання нового замовлення;

– перегляд списку замовлень;

– автоматичне формування звітів.

3.1.2 Вимоги до організації вхідних та вихідних даних

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатура та миша). Дані або вводяться вручну, або обираються із представлених списків.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

Обмін інформацією між додатком і сервером, а також між сервером і базою даних відбувається за допомогою файлів з довільним доступом, з розширенням .json.

Вхідні дані:

Дані для авторизації: логін/пароль.

Дані про замовлення та його статус:

- дата/час;
- адреса (вулиця, будинок, під’їзд);
- виконавець;
- телефон;
- статус виконання (не виконується, у процесі, виконано).

Вихідні дані:

Дані про водіїв та їх зайнятість:

- ПІБ;

- телефон;
- посвідчення;
- номер автомобіля;
- марка автомобіля;
- колір автомобіля;
- зайнятість.

Дані про замовлення:

- дата/час;
- телефон замовника;
- адреса (вулиця, будинок, під'їзд);
- дані про виконавця;
- статус виконання (не виконується, у процесі, виконано);
- звітність.

3.2 Вимоги до надійності

Програмний продукт повинен нормально функціонувати при безперебійній роботі персонального комп'ютера. При виникненні збоїв в роботі, відновлення нормальної роботи повинне проводитися після перезавантаження програми.

Для забезпечення надійності інформації повинна використовуватися система керування базою даних, що забезпечує цілісність транзакцій і цілісність інформації.

У разі введення користувачем некоректної інформації система повинно повідомити про помилку і надати можливість виправити її.

3.3 Вимоги до умов експлуатації

Необхідний рівень підготовки користувачів: мінімальні навички в користуванні комп'ютером. Комп'ютер призначений для роботи в закритому опалювальному приміщені.

3.4 Вимоги до складу та параметрів технічних засобів

Система повинна задовольняти вказаним мінімальним вимогам:

Апаратне забезпечення серверної частини:

– CPU: Intel Core i7 або еквівалентного рівня;

– RAM: 4 GB;

– HDD: 100 GB.

Апаратне забезпечення клієнтської частини:

– CPU: Intel Core i5 або еквівалентного рівня);

– RAM: 2 GB;

– HDD: 50 GB.

3.5 Вимоги до інформаційної та програмної сумісності

Також є вимоги до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows версії не нижче 7.

3.6 Вимоги до маркування та пакування

Додаток диспетчера буде упаковано в електронний архів і мати найменування disptaxi.rar.

4. Вимоги до програмної документації

Склад програмної документації повинен включати в себе:

- технічне завдання
- текст програми
- опис програми
- інструкція користувача
- програма і методика випробувань програмного забезпечення

5. Техніко-економічні показники

Не розраховуються в зв’язку з тим, що продукт розробляється в рамках кваліфікаційної роботи.

6. Стадії та етапи розробки представлені нижче в таблиці А.1.

Таблиця А.1 – Стадії та етапи розробки проекту

№ п/п	Назва етапів роботи	Терміни виконання
1.	Аналіз практичної задачі інженерії ПЗ	31.03.2023
2.	Аналіз вимог до ПЗ	07.04.2023
3.	Розробка архітектури ПЗ	21.04.2023
4.	Розробка проекту ПЗ	05.05.2023
5.	Реалізація та тестування ПЗ	12.05.2023

7. Порядок контролю та прийому

Контроль за аналізом та проектуванням кожної окремої частини програмного забезпечення для автоматизації диспетчерської служби таксі відбувається на кожному етапі з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Прийом готового програного проекту документують за допомогою протоколу проведення випробувань. У разі знаходження помилок під час прийому програмного виробу складається акт про знайдені помилки, який підписується представниками замовника і розробника і затверджується керівниками організації - замовника та організації - розробника. Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки, і оповістити замовника про повторне проведення перевірки.

Додаток Б

Текст програми

Код модуля server.js

```
"use strict";
var http = require('http');
var path = require('path');
var async = require('async');
var socketio = require('socket.io');
var express = require('express');
let bodyParser = require("body-parser");
var router = express();
var server = http.createServer(router);
var io = socketio.listen(server);
var utils = require('./utils.js');
router.use(express.static(path.resolve(__dirname, 'client')));
/**
 * ======================  Начальная настройка
========================
 * */
router.use(bodyParser.urlencoded({
    extended: true
}));
router.use(bodyParser.json());
/**
 * ======================   Роутинг ========================
 * */


router.get('/', function (req, res) {
    res.set('Content-Type', 'text/plain');
    res.send(``);
});
// IMPORTANT: Your application HAS to respond to GET /health with
status 200
```

```
//            for OpenShift health monitoring
// Проверка сервера
router.get('/health', function (req, res) {
    res.sendStatus(200);
});

// Вывод списка заказов
router.get('/orders', function (req, res) {
    res.writeHead(200, {"Content-Type": "text/json"});
    utils.getOrders(
        function(result)
        {
            res.write(JSON.stringify(result));
            res.end();
        }
    );
});
// Вывод списка водителей
router.get('/drivers', function (req, res) {
    res.writeHead(200, {"Content-Type": "text/json"});
    utils.getDrivers(
        function(result)
        {
            res.write(JSON.stringify(result));
            res.end();
        }
    );
});
// Добавление нового заказа
router.post('/addorder', function (req, res) {
    utils.addOrder(req.body);
    console.log("added: \n" + JSON.stringify(req.body));

public class Order {
```

```
    @SerializedName("_id")
    private String _id;
    @SerializedName("date")
    private String date;
    @SerializedName("address")
    private String address;
    @SerializedName("phone")
    private String phone;

    @SerializedName("executor")
    private String executor;
    @SerializedName("additional")
    private String additional;
    @SerializedName("status")
    private String status;
    @Override
    public String toString() {
        return "Order{" +
                "id=" + _id +
                ", date='" + date + '\'' +
                ", address='" + address + '\'' +
                ", phone='" + phone + '\'' +
                ", executor='" + executor + '\'' +
                ", status='" + status + '\'' +
                '}';
    }
    public Order(String _id, String date, String address, String
phone, String executor, String additional, String status) {
        this._id = _id;
        this.date = date;
        this.address = address;
        this.phone = phone;
        this.executor = executor;
        this.status = status;
        this.additional = additional;
    }
```

```
public String getPhone() {
    return phone;
}
public void setPhone(String phone) {
    this.phone = phone;
}
public String getId() {
    return _id;
}
public void setId(String _id) {
    this._id = _id;
}
public String getDate() {
    return date;
}
public void setDate(String date) {
    this.date = date;
}
public String getAddress() {
    return address;
}
public void setAddress(String address) {
    this.address = address;
}
public String getExecutor() {
    return executor;
}
public void setExecutor(String executor) {
    this.executor = executor;
}
public String getStatus() {
    return status;
}
public void setStatus(String status) {
    this.status = status;
```

```
    }
    public String getAdditional() {
        return additional;
    }
    public void setAdditional(String additional) {
        this.additional = additional;
    }
}
```

Код модуля OrderView.java

```
package view;

import com.google.maps.model.GeocodingResult;
import com.sun.org.apache.xpath.internal.operations.Or;
import model.Order;
import util.RequestUtils;

import javax.imageio.ImageIO;
import javax.swing.*;
import javax.swing.border.EmptyBorder;
import javax.swing.table.DefaultTableModel;
import java.awt.*;
import java.awt.event.ActionEvent;
import java.awt.event.ActionListener;
import java.io.IOException;
import java.net.URL;
import java.text.SimpleDateFormat;
import java.util.Date;



 class OrdersView extends JPanel {

    private JTable ordersTable;
    private String[] columnNames = {"ID", "Дата", "Адрес",
"Телефон", "Исполнитель", "Статус"};
    private JFrame parent;


    OrdersView(JFrame parent)
    {
        this.setLayout(new BorderLayout());
        this.setBackground(Color.white);
        Object[][] data = {};

        ordersTable = new JTable(data, columnNames);
      //  ordersTable.setPreferredSize(new
```

```
Dimension(this.getWidth(),this.getHeight()));
        ordersTable.setBorder( new EmptyBorder( 9, 9, 9, 9 ) );
        ordersTable.setPreferredSize(new Dimension(500,400));
        ordersTable.setEnabled(false);
        this.add(new JScrollPane(ordersTable),
BorderLayout.NORTH);

        JPanel buttonPanel = new JPanel();
        JButton addButton = new JButton("ДОБАВИТЬ ЗАКАЗ");
        addButton.setBackground(new Color(5, 5, 10));
        addButton.setForeground(Color.WHITE);
        addButton.setFocusPainted(false);
        addButton.setPreferredSize(new Dimension(175,40));
        addButton.setFont(new Font("Helvetica", Font.BOLD, 13));
        addButton.addActionListener(new ActionListener() {
            @Override
            public void actionPerformed(ActionEvent e) {
                addOrder();
            }
        });
        buttonPanel.add(addButton);

        JButton updateButton = new JButton("ОБНОВИТЬ СПИСОК");
        updateButton.setBackground(new Color(5, 5, 10));
        updateButton.setForeground(Color.WHITE);
        updateButton.setFocusPainted(false);
        updateButton.setPreferredSize(new Dimension(175,40));
        updateButton.setFont(new Font("Helvetica", Font.BOLD,
13));
        updateButton.addActionListener(new ActionListener() {
        @


 ();


        });
        buttonPanel.add(updateButton);
        buttonPanel.setBackground(this.getBackground());

        this.add(buttonPanel, BorderLayout.CENTER);
        this.updateTable();
    }

    private void updateTable()
    {
        new Thread(() -> {
            Order ordersArray[] = RequestUtils.getOrders();
            Object[][] data = new Object[ordersArray.length][6];
            int i = 0;
            for(Order order : ordersArray)
            {
                data[i][0] = order.getId();
                data[i][1] = order.getDate();
```

```
                data[i][2] = order.getAddress();
                data[i][3] = order.getPhone();


                data[i][4] = order.getExecutor();
                data[i][5] = order.getStatus();
                i++;
            }
            DefaultTableModel model = new DefaultTableModel(data,
columnNames);
            ordersTable.setModel(model);
        }).start();
    }

    private void addOrder()
    {
        OrderDialog orderDialog = new OrderDialog();
    }



    class OrderDialog extends JDialog implements ActionListener
    {
        private JButton checkBtn;
        private JButton okBtn;
        private JButton cancelBtn;

        private JTextField addressField;
        private JTextField telephoneField;
        private JTextField additionalField;

        private JLabel mapView;

        OrderDialog()
        {



            super(parent, "Добавить заказ", true);


this.setDefaultCloseOperation(WindowConstants.DISPOSE_ON_CLOSE);

            int width = 600, height = 460;
            this.setPreferredSize(new Dimension(width, height));
            this.setModal(true);
            this.getContentPane().setBackground(Color.white);
            this.getContentPane().setLayout(new BorderLayout());

            addressField = new JTextField();
            additionalField = new JTextField();
            telephoneField = new JTextField();
```

```
            JLabel
                    adressLabel = new JLabel("   Адрес:      "),
                    additionLabel = new JLabel("   Корпус,
подъезд: "),
                    telephoneLabel = new JLabel("   Телефон:");

            checkBtn = new JButton("Проверить");
            okBtn = new JButton("Добавить");
            cancelBtn = new JButton("Отмена");

            checkBtn.


            okBtn.addActionListener(this);

            JPanel buttonPanel = new JPanel();
            buttonPanel.setBackground(Color.white);
           // buttonPanel.setLayout(new BoxLayout(buttonPanel,
BoxLayout.LINE_AXIS));
            buttonPanel.add(Box.createHorizontalStrut(10));
            buttonPanel.add(okBtn);
            buttonPanel.add(cancelBtn);

            JPanel inputPanel = new JPanel();
            inputPanel.setLayout(new BoxLayout(inputPanel,
BoxLayout.PAGE_AXIS));
            inputPanel.add(Box.createVerticalStrut(25));
            inputPanel.setBackground(Color.white);


            JPanel inputAddress = new JPanel();
            inputAddress.setLayout(new BoxLayout(inputAddress,
BoxLayout.LINE_AXIS));
            inputAddress.add(Box.createHorizontalStrut(50));
            inputAddress.add(adressLabel);
            inputAddress.add(addressField);
            inputAddress.add(getCheckBtn());
            inputAddress.add(Box.createHorizontalStrut(60));
            inputAddress.setBackground(Color.white);
            inputPanel.add(inputAddress);




            inputPanel.add(Box.createVerticalStrut(5));



                else
                {
                     addressField.setText("Адрес не найден");
```

```
                }
            }
            else if(e.getSource() == cancelBtn)
            {
                System.out.print("1");
                this.dispose();
            }
            else if(e.getSource() == okBtn)
            {
                final double flat = lat, flng = lng;
                this.dispose();
                new Thread(() -> {
                    RequestUtils.postOrder(new Order(
                            "",
                            new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(new Date()),
                            this.getAddressField().getText(),
                            this.getTelephoneField().getText(),
                            "не назначен",

his.getAdditionalField


                    ));
                }).start();
            }

        }

        private void setMapImage( Image image)
        {
            mapView.setIcon(new ImageIcon((
                    new ImageIcon(image))
                    .getImage().getScaledInstance(600, 300,
Image.SCALE_SMOOTH)));
        }

        JButton getCheckBtn() {
            return checkBtn;
        }

        JButton getOkBtn() {
            return okBtn;
        }




        JButton getCancelBtn() {
            return cancelBtn;
        }
```

```
            inputTelephone.add(additionLabel);
            inputTelephone.add(additionalField);
            inputTelephone.add(telephoneLabel);
            inputTelephone.add(telephoneField);
            inputTelephone.setBackground(Color.white);
            inputTelephone.add(Box.createHorizontalStrut(60));

            inputPanel.add(inputTelephone);
            inputPanel.add(Box.createVerticalStrut(2));


            mapView = new JLabel();
            mapView.setBackground(Color.white);
            Image image =
RequestUtils.getImageFromURL("https://taxi-
users.io/staticmap.png");
            if(image != null)
            {
                setMapImage(image);
            }



            this.getContentPane().add(inputPanel,
BorderLayout.NORTH);
       /* frame.getContentPane().add(new JLabel(new ImageIcon((new
ImageIcon("image.jpg")).getImage().getScaledInstance(600, 300,
                java.awt.Image.SCALE_SMOOTH))),
BorderLayout.SOUTH);
*/
            this.setLocationRelativeTo(parent);
            this.setLocation(this.getX() - (width/2),this.getY() -
(height/2));

            this.getContentPane().add(buttonPanel,
BorderLayout.CENTER);
            this.getContentPane().add(mapView,
BorderLayout.SOUTH);
            this.pack();
            this.setResizable(false);
            this.setVisible(true);
        }

        @Override
        public void actionPerformed(ActionEvent e) {
            GeocodingResult geocodingResult;
            double lat = 46.,
                    lng = 32.;
            if(e.getSource() == checkBtn){
                geocodingResult =
RequestUtils.getGeocoding(addressField.getText());
```

```
                if(geocodingResult != null)
                {


addressField.setText(geocodingResult.formattedAddress);
                    Image image =
RequestUtils.getStaticMap(geocodingResult.geometry.location.toStri
ng());
                    if(image != null)
                        this.setMapImage(image);
                }
                else
                {
                     addressField.setText("Адрес не найден");
                }
            }
            else if(e.getSource() == cancelBtn)
            {
                System.out.print("1");
                this.dispose();
            }
            else if(e.getSource() == okBtn)
            {
                final double flat = lat, flng = lng;
                this.dispose();
                new Thread(() -> {
                    RequestUtils.postOrder(new Order(
                            "",
                            new SimpleDateFormat("yyyy-MM-dd
HH:mm:ss").format(new Date()),
                            this.getAddressField().getText(),
                            this.getTelephoneField().getText(),
                            "не назначен",


his.getAdditionalField


                    ));
                }).start();
            }

        }

        private void setMapImage( Image image)
        {
            mapView.setIcon(new ImageIcon((
                    new ImageIcon(image))
                    .getImage().getScaledInstance(600, 300,
Image.SCALE_SMOOTH)));
        }

        JButton getCheckBtn() {
            return checkBtn;
```

```
        }

        JButton getOkBtn() {
            return okBtn;
        }




        JButton getCancelBtn() {
            return cancelBtn;
        }




    }


public static GeocodingResult getGeocoding(String address)
{
    GeoApiContext context = new
GeoApiContext().setApiKey("AIzaSyDU_18hXT3Z7ji9_-
6jTe17N3XOU4PKE5s");

    GeocodingResult result = new GeocodingResult();
    try {

        GeocodingApiRequest apiRequest = new
GeocodingApiRequest(context);
        apiRequest.address(address);
        result = apiRequest.await()[0];


    } catch (Exception e) {
        e.printStackTrace();
    }

    return result;

}

public static Image getImageFromURL(String urlString)
{
    try {
        URL url = new URL(urlString);
        return ImageIO.read(url);
    } catch (Exception e) {
        return null;
    }
}
```

```
public static Image getStaticMap(String coord)
{
    return
getImageFromURL("https://maps.googleapis.com/maps/api/staticmap?"
+
            "center=7C" + URLEncoder.encode(coord) +
"&zoom=13&size=600x300&" +
            "markers=color:red%7Clabel:A%7C" +
URLEncoder.encode(coord) + "&key=" +
            API_KEY);

}
}
```

Додаток В

Опис програми

1 Загальні відомості

Розроблене програмне забезпечення для автоматизації роботи диспетчера служби таксі надає можливість здійснювати контроль над процесом виконання замовлень та спрощує документообіг.

Програма призначена для спрощення бізнес-процесів служби таксі, покращення та легкості ведення та обробки даних, полегшення комунікації між працівниками.

Для розробки програмного забезпечення були обрані наступні засоби:

– мови програмування Java та Javascript;

– СКБД MySQL;

– середовище розробки IntelliJ IDEA.

2 Функціональне призначення

Функціональним призначенням програмного забезпечення є автоматизація процесів:

- реєстрації та авторизації користувачів;
- прийому та передачі замовлень виконавцю;
- контролю за ходом виконання замовлень;
- складання звітності.

3 Опис логічної структури

Програма розроблена на модульній основі, тобто кожний компонент (модуль) програми відповідає за конкретну дію, наприклад: формування звіту, додавання замовлення. Дані, необхідні для роботи програми, зберігаються у нормалізованих таблицях бази даних.

4 Технічні та програмні засоби

Система повинна задовольняти вказаним мінімальним вимогам:

Апаратне забезпечення серверної частини:

– CPU: Intel Core i7 або еквівалентного рівня;

– RAM: 4 GB;

– HDD: 100 GB.

Апаратне забезпечення клієнтської частини:

– CPU: Intel Core i5 або еквівалентного рівня);

– RAM: 2 GB;

– HDD: 50 GB.

Також є вимоги до інформаційної та програмної сумісності. Для повноцінного функціонування системи необхідно використовувати операційну систему Windows версії не нижче 7.

5 Виклик та завантаження

Додаток диспетчера встановлюється безпосередньо на комп’ютер користувача. Програмне забезпечення відкривається або за допомогою ярлика, який знаходиться на робочому столі, або за допомогою вибору відповідного пункту меню «Пуск» - «Програми». Програма має один режим використання.

Після запуску з’являється форма авторизації користувача, де пропонується ввести логін та пароль для входу в систему. Якщо дані користувача ведені вірно, відкривається головна форма програми.

6 Вхідні та вихідні дані

Введення даних здійснюється за допомогою пристроїв введення даних (клавіатура та миша). Дані або вводяться вручну, або обираються із представлених списків.

Виведення даних здійснюється за допомогою пристроїв виведення даних (монітор та принтер).

Додаток Г

Інструкція користувача

Авторизація

Для авторизації у системі необхідно після запуску програми в вікні авторизації (див. рис. Г.1) ввести логін і пароль, після чого можна отримати доступ до програми шляхом натискання кнопки «ОК».

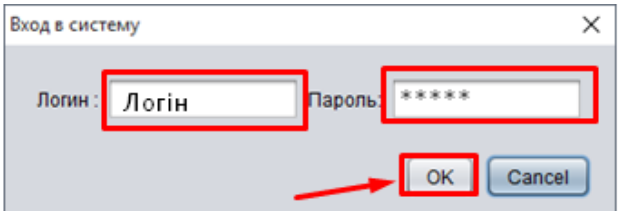

Рисунок Г.1 – Форма авторизації

Реєстрація водія

Для реєстрації водія необхідно натиснути на кнопку «Регистрация» на сторінці зі списком водіїв. У вікні реєстрації (див. рис. Г.2) необхідно ввести інформацію про водія, після чого додати в систему шляхом натискання кнопки «Ok».

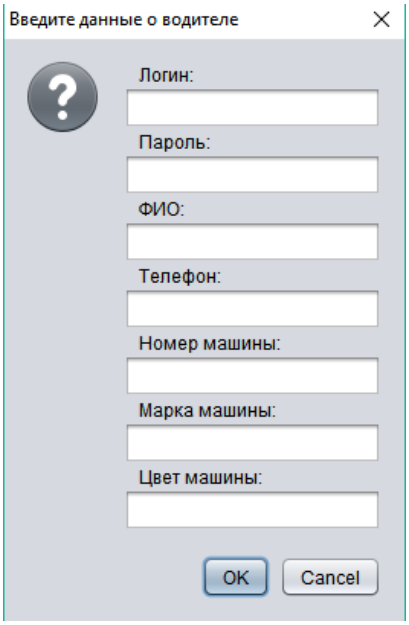

Рисунок Г.2 – Вікно реєстрації водія

Перегляд списку водіїв

Для перегляду списку водіїв після проходження авторизаціїї на головному вікні необхідно натиснути кнопку «Управление водителями». З’явилась таблиця яка відображає список всіх водіїв у системі (див. рис. Г.3).

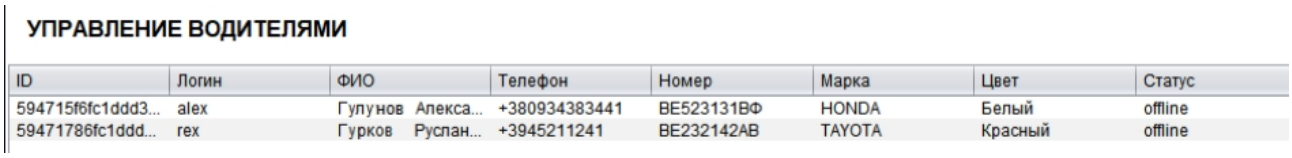

Рисунок Г.3 – Перегляд списку водіїв

Додавання нового замовлення

Для додавання нового замовлення після проходження авторизації на головному вікні необхідно натиснути на кнопку «Добавить заказ». У вікні додавання замовлення необхідно ввести інформацію про замовлення, після чого перевірити правильність адреси шляхом натискання кнопки «Проверить» та додати замовлення в систему шляхом натискання кнопки «Добавить» (див. рис. Г.4, Г.5).

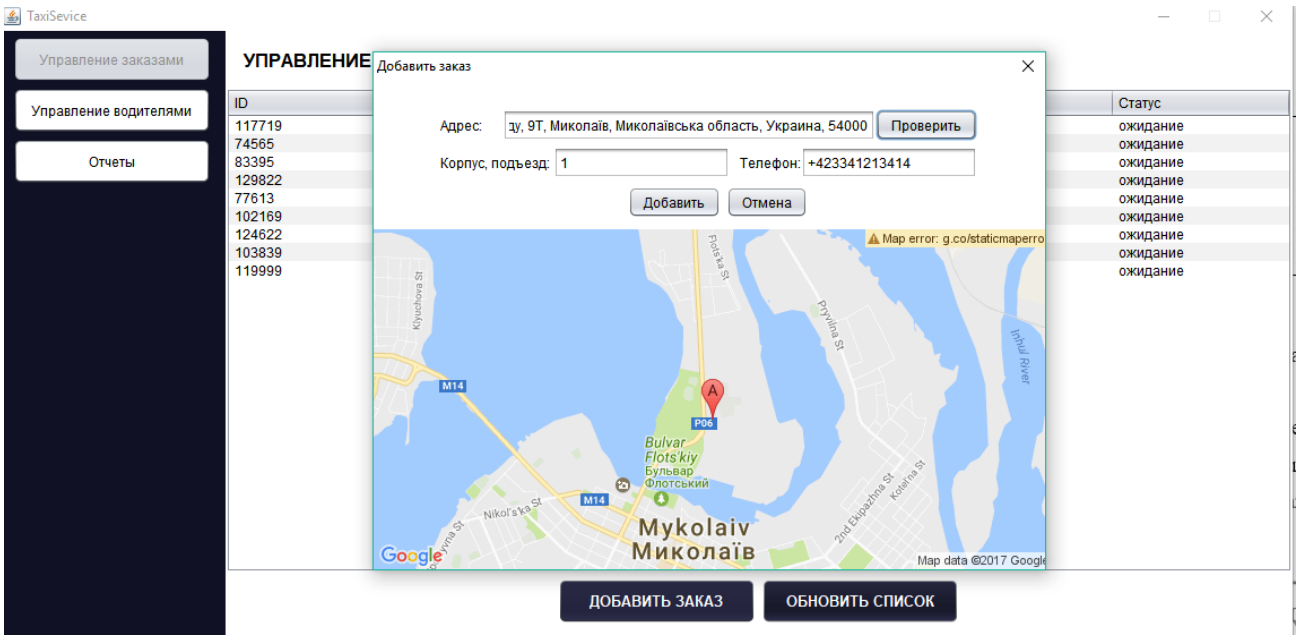

Рисунок Г.4 – Вікно «Добавить заказ»

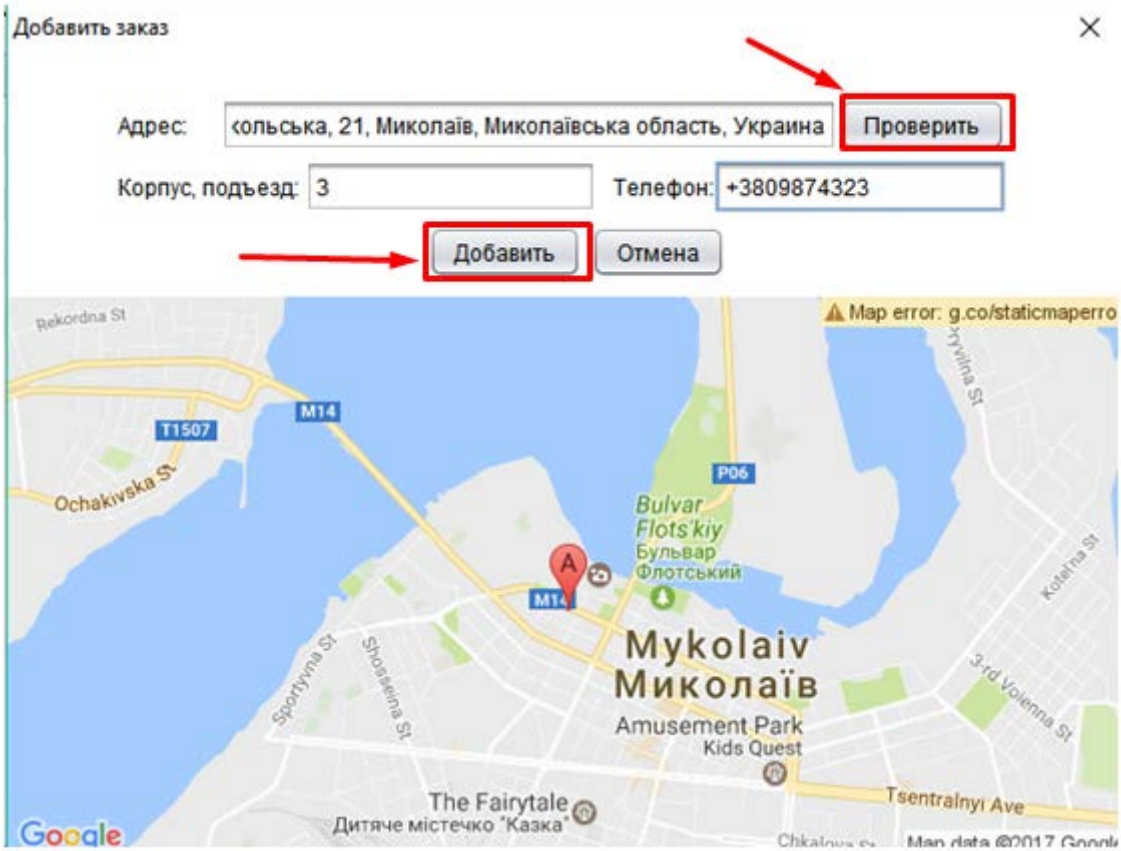

Рисунок Г.5 – Збільшений фрагмент вікна «Добавить заказ»

Перегляд списку замовлень

Після проходження авторизації на головному вікні відобразиться таблиця, яка відображає список всіх замовлень у системі (див. рис. Г.6).

УПРАВЛЕНИЕ ЗАКАЗАМИ

ID	Дата	Адрес	Телефон	Исполнитель	Статус
59471500fc1ddd3a5556...	2023-06-19 03:04:12	Nikol's'ka St, 21, Mykolaiv...	+389234234234	не назначен	ожидание
59471522fc1ddd3a5556...	2023-06-19 03:04:46	Tsentralnyi Ave, 15, Mykol...	421443412412	не назначен	ожидание
59471594fc1ddd3a5556...	2023-06-19 03:06:40	Pohranychna St, 15, Myk...	+380945425235	не назначен	ожидание
594715b7fc1ddd3a5556...	2023-06-19 03:07:15	Kosmonavtiv St, 35, Myko...	12412512412	не назначен	ожидание

Рисунок Г.6 – Перегляд списку замовлень

Формування звітів

Для формування звітів необхідно натиснути на кнопку «Отчеты» на головній формі (див. рис. Г.7).

ОТЧЕТЫ

ID	Дата	Адрес	Телефон	Исполнитель	Статус
59471500fc1ddd3a5556...	2023-06-19 03:04:12	Nikol's'ka St, 21, Mykolaiv...	+389234234234	не назначен	ожидание
59471522fc1ddd3a5556...	2023-06-19 03:04:46	Tsentralnyi Ave, 15, Mykol...	421443412412	не назначен	ожидание
59471594fc1ddd3a5556...	2023-06-19 03:06:40	Pohranychna St, 15, Myk...	+380945425235	не назначен	ожидание
594715b7fc1ddd3a5556...	2023-06-19 03:07:15	Kosmonavtiv St, 35, Myko...	12412512412	не назначен	ожидание

Рисунок Г.7 – Формування звітів

Додаток Д

Програма та методика випробування програмного забезпечення

1. Об’єкт випробувань

1.1 Назва об’єкту

Повна назва об’єкта випробувань: «Програмне забезпечення для автоматизації роботи диспетчера служби таксі». Коротка назва: програма.

1.2 Область застосування

Область застосування – підприємство, яке надає населенню послуги таксі.

1.3 Вимоги до складу виконуваних функцій

- авторизація користувача;

– реєстрація нового водія;

– перегляд списку водіїв;

– автоматичне додавання нового замовлення;

– перегляд списку замовлень;

– автоматичне формування звітів.

2 Мета випробувань

Мета проведення випробувань - оцінка експлуатаційних характеристик програми, перевірка і підтвердження працездатності програми в умовах, максимально наближених до умов реальної експлуатації.

3 Вимоги до програми

Програма має реалізовувати функції, описані в технічному завданні, яке наведено в Технічному завданні.

4 Вимоги до програмної документації

Для проведення тестування повинна бути надана наступна документація:

– технічне завдання

– текст програми

– опис програми;

– інструкція користувача;

- програма та методика випробування програмного забезпечення.

5 Склад, порядок та методи випробувань

5.1 Перевірка програми на відповідність технічному завданню:

– перевірка роботи функції «Авторизація користувача»;

– перевірка роботи функції «Реєстрація нового водія»;

– перевірка роботи функції «Перегляд списку водіїв»;

– перевірка роботи функції «Додавання нового замовлення»;

– перевірка роботи функції «Перегляд списку замовлень»;

– перевірка роботи функції «Формування звітів».

5.2 Перевірка програми на її програмно-апаратну сумісність: тестування на апаратних-програмних платформах різної конфігурації, але не нижче за мінімальні вимоги, наведені у технічному завданні; перевірка на наявність і усунення всіх помилок, зазначених у п. 5.1.

5.3 Візуальний перегляд програми: остаточне налагодження на предмет виявлення та усунення дрібних помилок.