КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

ІМЕНІ ТАРАСА ШЕВЧЕНКА

ФАКУЛЬТЕТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ

КАФЕДРА ІНФОРМАЦІЙНИХ СИСТЕМ ТА ТЕХНОЛОГІЙ

До захисту допущено
Завідувач кафедри ІСТ


Олександр КУЧАНСЬКИЙ
(підпис) (ім'я, ПРІЗВИЩЕ)

"___"_____ 2022р.

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

спеціальності 126 «Інформаційні системи та технології»

освітньої програми «Програмні технології інтернет речей»

на тему: «Розробка IOT системи керування дорожнім рухом»

Виконав: студент 4 курсу, групи IP-41

_____ Андрій РОТА _____ 

(шифр групи) (Ім'я, ПРІЗВИЩЕ) (підпис)

Керівник к. т. н., доцент Ольга КРАВЧЕНКО _____ 

(посада, науковий ступінь, вчене звання, Ім'я, ПРІЗВИЩЕ) (підпис)

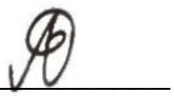
Консультант нормоконтроль к.т.н, доцент, Ростислав ЛІСНЕВСЬКИЙ __

(назва розділу) (посада, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Рецензент доцент кафедри ІСТ ЧДТУ, к.т.н., Анаіт КАРАПЕТЯН 

(посада, науковий ступінь, вчене звання, науковий ступінь, Ім'я, ПРІЗВИЩЕ) (підпис)

Засвідчую, що у пояснювальна записка не має запозичень з праць інших авторів без відповідних посилань.

Здобувач освіти 

(підпис)

Київ – 2022 року

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ
ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційні системи та технології
Освітній рівень Бакалавр
Спеціальність 126 Інформаційні системи та технології
Освітня програма Програмні технології інтернет речей

ЗАТВЕРДЖУЮ
Завідувач кафедри,
професор
Олександр КУЧАНСЬКИЙ
_____
«__» _____ 2022 року

ЗАВДАННЯ
НА ВИКОНАННЯ КВАЛІФІКАЦІЙНОЇ РОБОТИ БАКАЛАВРА

Здобувач освіти: Андрій РОТА

Група: ІР-41

1. **Тема кваліфікаційна робота бакалавра**: «Розробка IOT системи керування дорожнім рухом».
Затверджена протоколом засідання кафедри ІСТ №05/21_22 від 03.12.2021 року
2. **Строк подання студентом готової роботи** – «22» червня 2022 р.
3. **Вихідні дані до роботи**: дослідження в області управління та автоматизації дорожнього руху. Програмні рішення для системи автоматизованого керування дорожнім рухом. Дані про режим роботи світлофорів, стан на дорогах, швидкість машин, час простою у заторах, методи їх збору та збереження у базі даних.
4. Зміст роботи: РОЗДІЛ 1 АНАЛІЗ СФЕРИ ЗАСТОСУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ДОРОЖНІМ РУХОМ. ПОСТАНОВКА ЗАДАЧІ (проблематика старої системи управління дорожнім рухом, аналіз аналогічних існуючих рішень); РОЗДІЛ 2 ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ДОРОЖНІМ РУХОМ (проектування бази даних автоматизованої системи керування дорожнім рухом, проектування інтерфейсу панелі адміністратора CRM автоматизованої системи керування дорожнім рухом); РОЗДІЛ 3 ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ДОРОЖНІМ РУХОМ

(розробка Unity симуляції автоматизованої системи керування дорожнім рухом, розробка сцени, розміщення 3D-об'єктів, розробка скриптів для роботи автоматизованої системи управління дорожнім рухом, збірка Unity симуляції та її інтеграція в CRM систему).

5. **Перелік графічного матеріалу**: функціональна модель розробки IoT системи, концептуальна схема роботи IoT системи, контекстна діаграма процесу діяльності підприємства IoT системи, діаграми декомпозиції системи по відділам, діаграма дерева вузлів підприємства, діаграма прецедентів, діаграма діяльності, послідовності, класів, логічна модель бази даних, фізична модель бази даних, схема MVC роботи веб-додатку, схема обміну інформацією на ділянці вулиці.

6. **Календарний план виконання роботи:**

Етапи виконання кваліфікаційної роботи бакалавра	**Термін виконання**	**Результат виконання**
1. Вибір тематики кваліфікаційної роботи бакалавра	01.09.2021-01.10.2021	виконано
2. Наказ про затвердження тем кваліфікаційної роботи бакалавра та призначення керівників	03.12.2021	виконано
3. Розробка плану кваліфікаційної роботи бакалавра і його погодження з керівником	25.12.2021	виконано
4. Написання I розділу кваліфікаційної роботи	19.03.2022	виконано
5. Написання II розділу кваліфікаційної роботи	25.04.2022	виконано
6. Написання III розділу кваліфікаційної роботи	29.04.2022	виконано
7. Підготовка висновків і пропозицій	30.04.2022	виконано
8. Попередній захист кваліфікаційної роботи	12.05.2022	виконано
9. Перевірка на плагіат	13.05.2022-15.06.2022	виконано
10. Нормоконтроль	02.06.2022-06.06.2022	виконано
11. Рецензування кваліфікаційної роботи бакалавра і представлення роботи на кафедру в друкованому вигляді	15.06.2022	виконано
12. Захист кваліфікаційної роботи бакалавра	23.06.2021	

Дата видачі завдання «__» _____ 2022 р.

Керівник роботи: к. т. н., доцент Ольга КРАВЧЕНКО_____ (підпис)

Завдання прийняв до виконання:

Здобувач освіти на освітньому рівні «бакалавр» 4-го курсу групи ІР-41

Андрій РОТА _____

(Власне Ім’я, ПРІЗВИЩЕ) (підпис)

АНОТАЦІЯ

КИЇВСЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ ІМЕНІ ТАРАСА ШЕВЧЕНКА

Факультет інформаційних технологій

Кафедра Інформаційних систем та технологій

Освітня програма «Програмні технології інтернет речей»

Кваліфікаційна робота бакалавра Андрія РОТИ

Тема роботи: «Розробка IOT системи керування дорожнім рухом».

Мета кваліфікаційної роботи бакалавра – розробка автоматизованої системи управління дорожнім рухом з симуляцією потоку трафіку.

Об’єкт дослідження – використання автоматизованої системи управління дорожнім рухом для забезпечення розвантаження трафіку на дорогах.

Предмет дослідження – автоматизація управління дорожнім рухом.

Кваліфікаційна робота бакалавра складається зі змісту, вступу, основної частини, яка включає три розділи, висновків та списку використаних джерел. Всього 75 сторінок, 69 рисунків, 7 таблиць.

КЛЮЧОВІ СЛОВА: IoT, CRM-система, Unity, C#, PHP, JavaScript, HTML, CSS, ООП, MySQL, датчик руху, світлофор, мікроконтролер, 3D модель, симуляція, автоматизація.

ABSTRACT

TARAS SHEVCHENKO NATIONAL UNIVERSITY OF KYIV

Faculty of Information Technologies

Department of Information Systems and Technologies

Educational Program "Software Technologies of the Internet of Things"

Qualification work of master Andrey ROTA.

Work topic: "Automated traffic control system"

The purpose of the bachelor’s qualification work is to develop an automated traffic control system with traffic flow simulation.

The object of research use of an automated traffic control system to ensure the unloading of traffic on the roads.

The subject of research is traffic control automation.

The bachelor's qualification work consists of the content, introduction, main part, which includes three sections, conclusions and a list of sources used. Total 75 pages, 69 pictures, 7 tables.

KEY WORDS: IoT, CRM-system, Unity, C #, PHP, JavaScript, HTML, CSS, OOP, MySQL, motion sensor, traffic light, microcontroller, 3D model, simulation, automation.

ЗМІСТ

ВСТУП

Тема дорожнього руху є як ніколи актуальною у сучасному світі. Великі міста мають щорічний приріст кількості транспортних засобів у населення, що обумовлено поступовим зростанням їх доступності, а також процесам урбанізації. Зростання кількісті машин на дорогах тягне за собою зростання кількості автотранспортних пригод, викидів палива та зменшення комфортності пересування містом через постійні затори.

Не всі сучасні міста адаптувалися до сьогоденних реалій. Більшість з них функціонують за застарілою системою керування дорожнім рухом, яка не може повноцінно відповідати усім сучасним стандартам. Автотранспортні пригоди, кількість яких може зростати із року в рік, є прямим наслідком неузгодженої роботи світлофорів. Вулиці не можуть «зрозуміти» де і скільки часу виділяти на червоне, або зелене світло, тому ми маємо постійні затори.

Для забезпечення безпечного і комфортного пересування містом були розроблені автоматизовані системи управління дорожнім рухом. Такі системи дозволяють світлофорам аналізувати поточну ситуацію на дорогах та адаптувати режим своєї роботи для того, щоб розвантажити транспортний трафік. Досягається це за рахунок сучасних датчиків і контролерів, які дозволяють збирати інформацію про швидкість проїжджаючих машин в єдину систему, фіксувати порушників правил дорожнього руху та вчасно оповіщати відповідні диспетчерські відділи.

Автоматизація управління дорожнім рухом дозволяє отримувати оперативну обстановку на дорогах через систему датчиків і світлофорів пов’язаних між собою та об’єднаних мікроконтролерами. Також система включає в себе камери, які в купі з працюючим обладнанням, надають змогу фіксувати та протоколювати усі транспортні пригоди в реальному часі. Автоматизація дозволяє розвантажити не тільки дороги, але й робочий день диспетчерського відділу, який замість перегляду камер, може займатися вирішенням більш важливих питань. Такий підхід до управління та контролю ситуації на дорогах дозволяє зменшити кількість ДТП, зменшити забруднення

навколишнього середовища викидами палива, зменшити витрати на усунення наслідків неузгодженої роботи застарілої системи, а також підвищити комфортність пересування містом, що позитивно позначається на ефективності роботи міста в цілому.

Саме тому темою роботи обрано «Автоматизовану систему управління дорожнім рухом». Актуальність і значимість цієї теми є беззаперечною, як і користь від впровадження таких систем у великі міста.

Метою дослідження є створення автоматизованої системи управління дорожнім рухом з розробкою симуляції трафіку, що дозволить отримати модель для оцінки ефективності використання автоматизованих засобів замість застарілої моделі моніторингу доріг у містах.

Об’єктом дослідження є використання автоматизованої системи управління дорожнім рухом для забезпечення розвантаження трафіку на дорогах для зниження кількості заторів, викидів у повітря та кількості дорожньо-транспортних пригод.

Предметом дослідження є автоматизація управління дорожнім рухом, що покращить ефективність роботи диспетчерського відділу та зробить ситуацію на дорогах комфортнішою.

Методами дослідження є аналіз та порівняння аналогічних автоматизованих систем управління дорожнім рухом, розгляд окремих елементів таких систем (датчиків, мікроконтролерів, програмних засобів), моделювання ситуації на дорогах, а також експерименти на перевірку адаптації системи до поточного стану на дорогах.

Для виконання мети дослідження необхідно розробити CRM-систему, базу даних та симуляцію роботи автоматизованої системи управління дорожнім рухом для того, щоб мати візуальну модель працюючої системи, яка адаптується до поточного стану на дорогах та розвантажує затори на вулицях.

1. АНАЛІЗ СФЕРИ ЗАСТОСУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ДОРОЖНІМ РУХОМ. ПОСТАНОВКА ЗАДАЧІ

У наш час системам дорожнього руху категорично необхідно використовувати засоби автоматизації для забезпечення комфортного пересування містом. Людей стає більше, як і транспортних засобів, і вже важко уявити собі систему ручного керування усіма потоками транспорту. Більшість вулиць мають серйозні проблеми, такі як:

- неузгоджений контроль за світлофорами;
- недостатнє технічне обслуговування світлофорного обладнання;
- нескоординоване керування рухом.

Тому, в цій роботі розглядається саме розробка IOT системи керування дорожнім рухом, яка виконує такі функції:

- адаптивне централізоване та локальне управління транспортними потоками;
- збір та обробка статистичних даних про поточну ситуацію на дорогах;
- відео-контроль за станом вулиць;
- забезпечення розвантаження вулиць для комфортного пересування містом.

Заходи щодо покращення дорожнього руху:

- введення системи автоматизованого керування дорожнім рухом за допомогою комп’ютера;
- розгортання та встановлення системи регулювання дорожнім рухом;
- стандартизація системи дорожнього руху та її технічне обслуговування.

Цілі:

- зменшення кількості дорожньо-транспортних пригод;
- зниження кількості заторів;
- підвищення пропускної здатності доріг.

1.1. Проблематика старої системи управління дорожнім рухом

Розвиток транспортної системи у сучасному світі є доволі важливим питанням для кожної країни. Все більше людей мають власний транспортний засіб, що в рамках поточної ситуації на дорогах та перехрестях, може призводити до перевантаження трафіку. Перевантаження в свою чергу є наслідком застарілої системи керування дорожнім рухом, яка зберігається у більшості міст нашої країни та багатьох інших країн. Неможливість автоматично аналізувати навантаження на дорогах та керувати роботою світлофорів призводить до заторів. Затори та перевантаження в свою чергу можуть призвести до дорожньо-транспортних пригод. Безпека дорожнього руху і комфортне пересування містом є актуальним питанням сьогодення.

У великих містах, як, наприклад, у Києві, де «1.13 мільйонів автомобілів та щорічний приріст автівок на 2021 рік по зрівнянню з 2020 становив 21%» [1], дуже важлива автоматизація оперативно-диспетчерського управління рухом. Для цього треба розгорнути необхідне серверне та технічне обладнання, монітори, а також провести відповідні заходи для забезпечення вулиць датчиками та мікроконтролерами, які будуть працювати у певному радіоканалі, оповіщаючи один одного про стан на дорогах. Розробка IOT системи керування дорожнім рухом забезпечить безпеку на дорогах, мінімізацію заторів, розвантаження вулиць, а також зменшення викидів у повітря, що покращить екологічний та економічний стан у місті.

Основне завдання системи автоматизованого управління дорожнім рухом – це забезпечення розвантаження доріг за допомогою моніторингу стану трафіку у реальному часі та адаптації системи до поточного стану на дорогах.

Проблеми застарілої системи:

- велика кількість викидів пального у повітря;
- велика кількість заторів на ключових транспортних вузлах;
- фінансові втрати через неавтоматизоване та неоперативне реагування на дорожні проблеми.

Додаткові проблеми:

- відсутність моніторингу доріг у реальному часі;
- відсутність ідентифікації транспортних засобів у реальному часі;
- відсутність бази даних пересування автомобілей містом;
- відсутність таких важливих інформаційних журналів, як: журнал стану вулиць, журнал проїжджаючих машин, дані про стан світлофорів, камер, датчиків.

1.2. Аналіз аналогічних існуючих рішень

В даному розділі буде розглянуто три аналоги систем керування дорожнім рухом.

1.2.1. Програмне забезпечення АСКДР SEA TCS

Компанія SEA TCS [2] пропонує сучасну систему керування дорожнім рухом (АСКДР), яка надає можливість у реальному часі регулювати дорожній рух, підтримувати працездатність світлофорних об’єктів, контролювати вуличне освітлення, передивлятися стан доріг за допомогою відповідних камер тощо.

За допомогою впровадження автоматизованого комплексу засобів контролю дорожнього руху можна вирішити наступні завдання:

– зменшити завантаженість доріг та ймовірність появи заторів;

– забезпечити зв'язок зі світлофорними об'єктами та оперативний контроль їх функціонування;

– знизити рівень шуму та концентрацію вихлопних газів у місцях скупчення транспортних засобів;

– попередити надмірний знос дорожнього покриття;

– узгодити режими роботи світлофорів задля пониження інтенсивності транспортного потоку;

– значно підвищити рівень безпеки на дорогах і оперативність управління дорожнім рухом.

Доступ працівників диспетчерського центру до усіх можливостей системи автоматизованого керування дорожнім рухом АСКДР SEA TCS (Traffic Control

System) здійснюється через веб-інтерфейс робочих місць та дорожніх контролерів (рис. 1.1).

Основні функції:

– управління рухом транспорту по інтегрованим вулицям через CRM-систему;

– отримання інформації про стан світлофорних об’єктів у реальному часі;

– online моніторинг та діагностика стану дорожніх контролерів RTC, світлофорів, камер, детекторів та іншого обладнання;

– аналітичні відомості щодо роботи світлофорної сигналізації у реальному часі;

– динамічний режим "зелена хвиля", при якому забезпечується одночасне управління світлофорами, інформаційними табло, електронним обладнанням та освітленням пішохідних переходів.

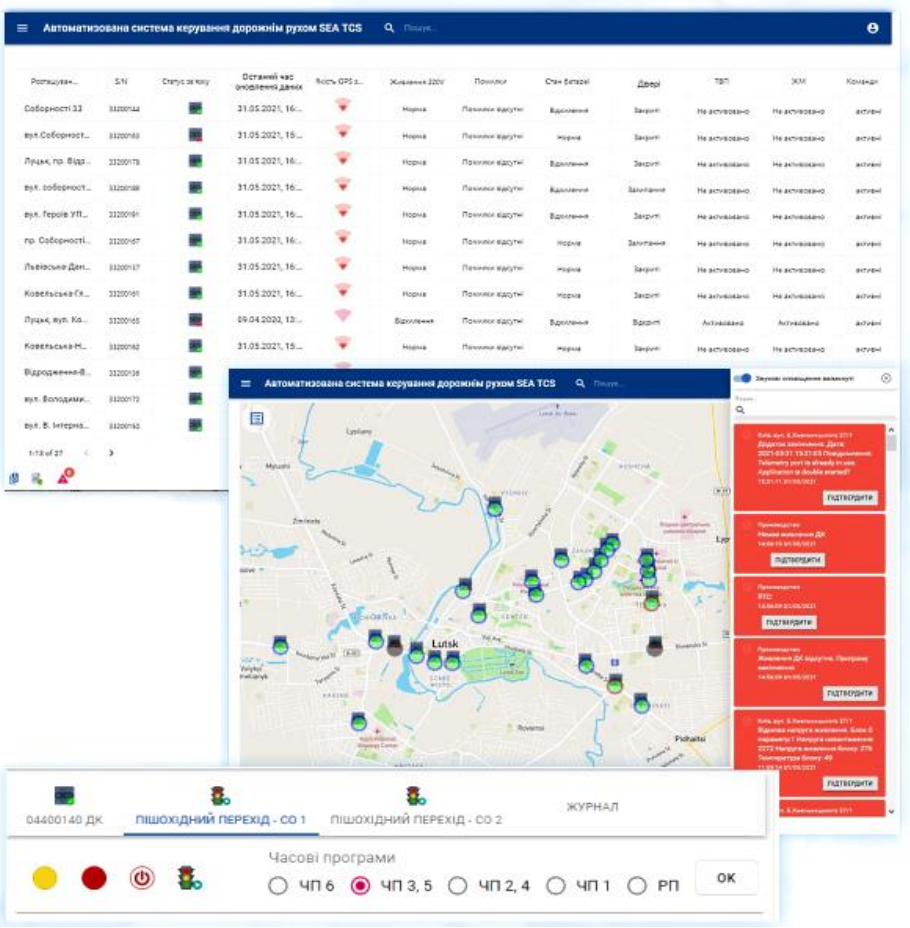


Рисунок 1.1 - Інтерфейс робочого місця диспетчера АСКДР

Виконання загальносистемних та серверних функцій, обробка та обмін даними, функціонування програмних модулів, WEB-сервера, дорожніх контролерів і системи в цілому підтримується програмним забезпеченням АСКДР SEA TCS, яке включає вбудоване ПЗ ДК та WEB-інтерфейс, сервер БД та Інтернет сервер, ПЗ WEB-порталу з інтерфейсами АРМ. Система зберігає усю інформацію на сервері БД і дозволяє авторизованим користувачам підключатися до АСКДР через Інтернет для моніторингу та керування контрольованими об'єктами вулично-дорожньої мережі.

Компанія вже має готові рішення по впровадженню автоматизованої системи управління дорожнім рухом на нову ділянку дороги. Налаштування системи для замовника здійснюється на придбаній ним серверній апаратурі. Для невеликих міст є рішення у вигляді інтеграції системи контролю на базі сервера у постачальника програмного забезпечення. Доступ до обладнання та програмних модулів компанії надається користувачам на основі абонентської плати.

1.2.2. АСКДР підприємства «Росток-ЕЛЕКОМ»

Підприємство “Росток-ЕЛЕКОМ” [3] працює з 1992 року, встановило перший дорожній контролер у 1995 році, а центр керування дорожнім рухом м.Києва працює з 1997 року. На сьогоднішній день обладнання підприємства завоювало довіру більш ніж 60 міст України та міст за кордоном, таких як Рига, Кишинев та Тбілісі. Всього підприємство випустило більш, як 4000 контролерів та 20 000 світлофорів. Обсяг експорту продукції складає 25%- 35%.

Ключові складові цієї АСКДР (рис. 1.2) на принципах інтелектуальних транспортних систем:

- створення нового проекту, редагування, видалення;
- перегляд дорожнього трафіку у реальному часі, аналіз ситуації на дорогах;
- пз для отримання математичної моделі міста;
- відео-сервер та програмне забезпечення відео-нагляду;

– відеокамери, контролери, світлофори, датчики, детектори транспорту;

– аналіз отриманих за час моніторингу даних та реалізація режимів «зелена вулиця» та «зелена хвиля»;

– адаптивне управління автоматизованою системою.

Рисунок 1.2 - АСКДР виробництва Росток-ЕЛЕКОМ у Тбілісі

Результати впровадження автоматизованої системи управління дорожнім рухом:

– збільшення середньої швидкості руху на 15%;

– зменшення аварійності на 20%;

– зменшення затримок транспорту на 18%;

– зменшення шкідливих викидів в повітря на 13%;

– зменшення витрат пального на 15%.

1.2.3. Сайт компанії, що виробляє пристрої для керування рухом, «АТІЛОС»

Товариство з обмеженою відповідальністю «Багатопрофільне підприємство» АТІЛОС» [4] займається розробкою і виробництвом високотехнологічної

продукції виробничо-технічного призначення. Підприємство є одним з лідерів по виробництву світлотехнічної продукції розробленої на основі світловипромінювальних діодів (рис. 1.3).

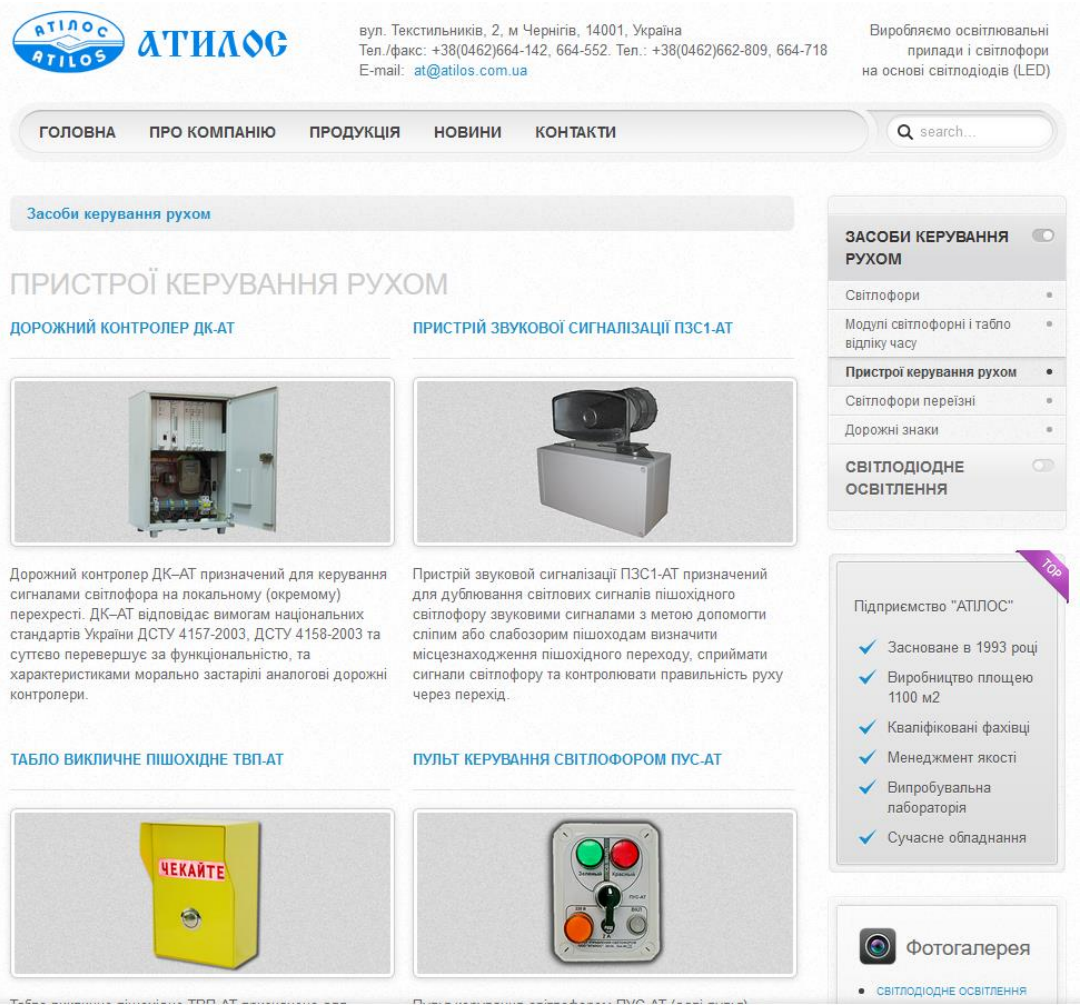


Рисунок 1.3 - Сторінка з засобами керування рухом

Вироби підприємства зарекомендували себе як високотехнологічну продукцію на базі сучасних електронних компонентів провідних світових виробників. У виробах підприємства використовуються сучасні цифрові технології на основі однокристальних мікро-ЕОМ, програмованих контролерів, застосовується технологія поверхневого монтажу. Для розробки продукції та документації використовуються сучасні комп'ютерні технології. Виробничі технології підприємства засновані на використанні сучасного обладнання фірм PACE, ERSA, TABAI, BOSH, DREMEL, а також сучасних матеріалів відомих виробників. Виробництво забезпечене необхідним контрольним, вимірювальним

та випробувальним обладнанням, використовуються автоматизовані робочі місця. На всі вироби розроблена необхідна конструкторська і технологічна документація, що відповідає вимогам стандартів ЄСКД і ЄСТД. Технологічні процеси виробництва є безвідходними. Продукція, технологічні процеси і діяльність підприємства безпечні для навколишнього середовища, не мають негативного впливу.

У відповідності, до розглянутих аналогів, систематизовано основні тези у таблиці порівняння аналогів 1.1 за обраними критеріями. Критерії характеризують призначення продукції, архітектуру програмного забезпечення, особливості рівня якості та відповідність продукції державним стандартам, а також ціну.

Таблиця 1.1 - Порівняння аналогів

Критерій	**Компанія**		
	СЕА	**Росток-ЕЛЕКОМ**	**АТІЛОС**
1	2	3	4
Продукт	АСКДР	АСКДР	Пристрої системи контролю дорожнього руху
Призначення	Створення і налагодження системи контролю дорожнього руху	Створення і налагодження системи контролю дорожнього руху	Продаж пристроїв для систем контролю дорожнього руху
Архітектура	Клієнт-сервер	Клієнт-сервер	Клієнт-сервер
Ціна	Ліцензія (на ПЗ або на SaaS)	Ліцензія	Безкоштовне використання, оплачується придбана продукція

При ознайомленні з інформацією, представленою компаніями на власних сайтах, була отримана інформація для подальшого виконання робіт, а саме:

– СЕА надала досить детальну інформацію про завдання, функції, компоненти та структуру, а також приклад лаконічного вигляду інтерфейсу АСКДР;

– у презентації «Росток-ЕЛЕКОМ» ґрунтовано описані етапи реалізації системи та різновид її модулів, що допомагає краще зрозуміти, який продукт має бути отриманий у кінці;

– сайт «АТІЛОС» є гарним прикладом інформаційної сторінки з короткими та повними описами різноманітних продуктів, дані про які можуть бути використані у подальшому для вибору елементів автоматизованої системи управління.

1.3. Аналіз платформи Unity як інструменту для реалізації симуляції автоматизованої системи управління дорожнім рухом

Unity (рис. 1.4) дає можливість створювати програми, що працюють на більш ніж на 20 різних платформах, що включають персональні комп'ютери, ігрові консолі, мобільні пристрої, інтернет-програми та інші [5].

Unity це дуже потужний, і водночас зручний та простий у вивченні ігровий движок. Зараз більшість розробників різного рівня пишуть свої ігри саме на цьому движку і на це є ряд причин.

Рисунок 1.4 - Інтерфейс Unity

Перше на що звертає увагу новачок в вивченні Unity, це те, що там існує спеціальний магазин різних ассетів та плагінів, які ви можете вбудовувати у свою гру. Тобто це повністі готові набори текстур, об’єкти оточующого середовища, анімації і так далі. Це дозволяє створювати ігри в рази швидше та продуктивніше, не відволікаючись на моделювання об’єктів. Також є платні набори ассетів, які відкривають ще більш широкий спектр різноманітних предметів для створення нової гри [6].

По-друге, програма має user-friendly графічний редактор, який дозволяє розставляти об'єкти, додавати ворогів, малювати красиві, великі та незвичайні карти місцевості та багато іншого. По суті графічний редактор дає розробнику доступ до найдрібнішого налаштування оточуючого світу та окремих його предметів. Це дуже сильно заощадує сили і час на розробку і є безумовною перевагою Unity над його аналогами.

Інтерфейс Untiy дозволяє відкривати та розташовувати робочі вікна як зручніше розробнику, завдяки чому можна проводити налагодження гри чи застосунка прямо в редакторі. Гра в Unity поділяється на сцени, або рівні в деяких іграх, які являють собою окремі файли з певним оточенням, своїм набором об'єктів, сценаріїв, і налаштувань. Рівні можуть містити в собі і об'єкти-

моделі (ландшафт, персонажі, предмети довкілля тощо), і порожні ігрові об'єкти, тобто такі, що не мають моделі, проте задають поведінку інших об'єктів (тригери подій, колайдери (рис. 1.5), точки збереження тощо).

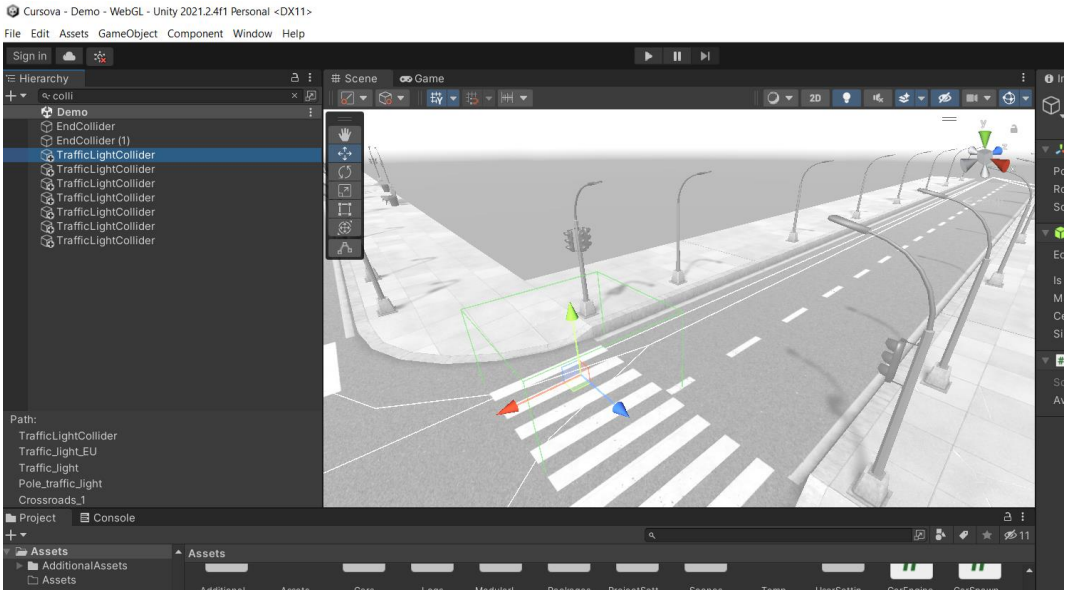


Рисунок 1.5 - Порожній об’єкт «Колайдер»

У Unity є вбудована фізика твердих тіл і тканини, а також Ragdoll (ганчіркова лялька) [7]. За допомогою системи дочірніх елементів у Unity об’єкти можуть повторювати всі зміни позиції, повороту і масштабу батьківського об'єкта. Скрипти у Unity додаються як окремі компоненти об’єкту, що дозволяє їм отримати доступ до усії властивостей цього об’єкту.

Unity реалізує оптимізацію виконання коду за допомогою стиснення текстур та міпмапінга. Також Unity має такий функціонал як: затінення навколишнього світла у екранному просторі, динамічні тіні за картами тіней, рендер у текстуру та повноекранні ефекти обробки зображення, такі як зернистість, глибина чіткості, розмиття в русі, відблиски віртуальних лінз або ореол навколо джерел світла.

У процесі розробки зроблено висновок, що Unity, як ігровий двигун, має безліч функціональних можливостей, які дозволили їх використати для виконання поставленного завдання з розробки симуляції автоматизованного управління дорожнім рухом. Серед найбільш важливіших функціональних можливостей, я би видлів: моделювання фізичних середовищ, карти нормалей,

динамічні тіні та багато іншого. Особливу увагу я би приділив реалізації тіней та світла, тому що в Unity це зроблено на вищому рівні. В порівнянні з його аналогами, в Unity не так помітне навантаження на компьютер при великій кількості світла на певній сцені. Це якраз завдяки добрій оптимізації коду розробниками платформи, яка була розглянута вище. Також, на відміну від багатьох ігрових движків, у Unity є ще дві основні переваги: наявність візуального середовища розробки та міжплатформова підтримка (рис. 1.6).

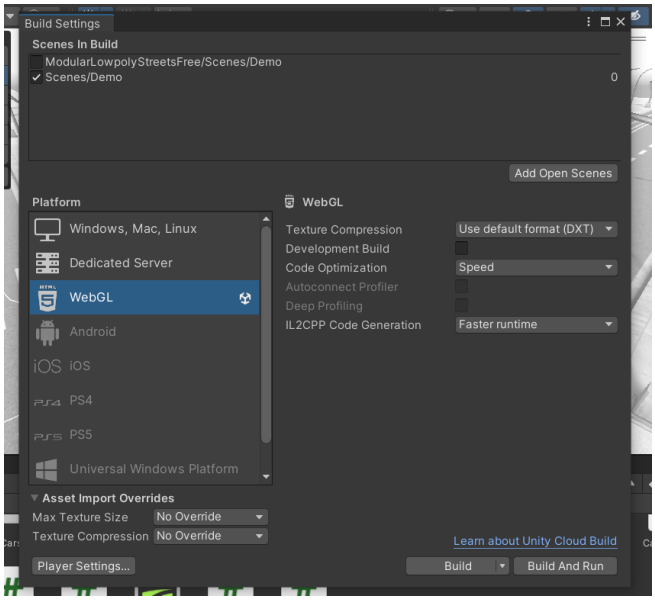


Рисунок 1.6 - Вибір платформи для розгортання Unity гри

Перший фактор включає не тільки інструментарій візуального моделювання, а й інтегроване середовище, що підвищує продуктивність розробників, зокрема етапів створення прототипів та тестування. Під міжплатформною підтримкою мається на увазі не тільки місця розгортання гри (на персональному комп'ютері, на мобільному пристрої, консолі тощо), але й наявність інструментарію розробки (інтегроване середовище може використовуватись під Windows та Mac OS).

Ще однією перевагою Unity є модульна система, за допомогою якої відбувається створення ігрових об'єктів, коли останні є комбінованими пакетами функціональних елементів. На відміну від механізмів успадкування, об'єкти в Unity створюються за допомогою об'єднання функціональних блоків, а не

поміщення у вузли дерева успадкування. Такий підхід полегшує створення прототипів, що є актуальним при розробці ігор.

Недоліками є обмеження візуального редактора під час роботи з багатокомпонентними схемами. Другим недоліком є відсутність підтримки Unity посилань на зовнішні бібліотеки, роботу з якими розробникам доводиться налаштовувати самостійно, що ускладнює командну роботу. Ще один недолік пов'язаний із використанням шаблонів екземплярів, або «префабів».

З одного боку, ця концепція Unity пропонує гнучкий підхід візуального редагування об'єктів, але з іншого боку, редагування таких шаблонів є складним. Також, WebGL-версія движка, яку я використовував для того, щоб відкривати Unity симуляцію у браузері в силу специфіки своєї архітектури має ряд невирішених проблем з продуктивністю, споживанням пам'яті і працездатністю на мобільних пристроях.

Можна зробити висновок, що Unity є чудовою програмою для новачків, які ще не мають команди, але хочуть почати створювати прості ігри та заробляти на них гроші (рис. 1.7).

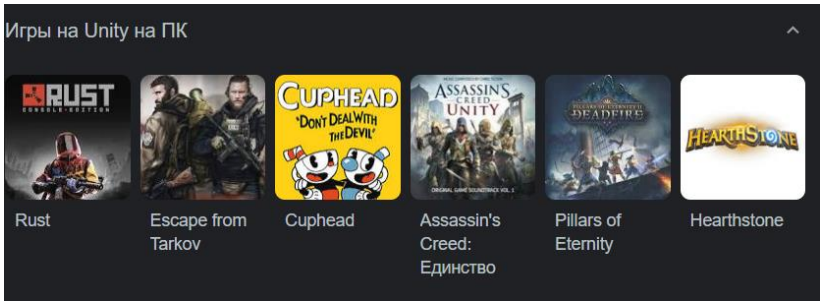


Рисунок 1.7 - Відомі ігри написані на Unity [8]

1.3.1. Аналіз аналогів Unity

Огляд аналогів розпочато з Unreal Engine Blueprints (рис. 1.8) [9]. Це ігровий двигун на основі якого були створені такі ігри як Fortnite Mobile, Life is Strange, Mortal Kombat, Pro Evolution Soccer 2020, Bright Memory. Він є головним конкурентом Unity по кількості розробників.

Рисунок 1.8 - Інтерфейс Unreal Engine

Переваги Unreal Engine:

- можна робити ігри без кодингу як такого. Для цього в Unreal Engine є візуальний редактор Blueprints, який автоматизує написання скриптів та налаштування поведінки ігрових об'єктів;
- велика кількість безкоштовних ассетів, що допоможуть у розробці;
- вбудований інструмент для оптимізації ігор для мобільних платформ.

Недоліки Unreal Engine:

- у двигуна погана оптимізація. Якщо додати на карту занадто багато об'єктів, особливо об’єктів освітлення, або спробувати створити гру з відкритим світом, така гра гальмуватиме. Справа в тому, що Unreal Engine промальовує всі елементи незалежно від того, чи потрапляють вони у поле зору гравця;
- інтерфейс розрахований на новачків, більшість дійсно потрібних кнопок швидкого доступу розташовані невдало;
- при створенні великих ігор розробникам потрібно серйозно оптимізувати свій код.

Крім цього, Unity має чимало інших аналогів: CryENGINE Free SDK, 3D Rad, Unreal Development Kit (UDK), Kodu Game Lab, NeoAxis 3D Engine, Construct 2, Game Editor, Autodesk 3ds Max, Game Maker, Clickteam Fusion.

CryENGINE Free SDK (рис. 1.9) [10] – потужна платформа, на якій були створені такі популярні шутери від першої особи, як Far Cry та Crisis. CryENGINE має великий набір вбудованих текстур, скриптів і об'єктів навколишнього середовища. Завдяки наявності інтуїтивно зрозумілого інтерфейсу програми та зручної системи підказок, можна дуже швидко оволодіти управлінням.

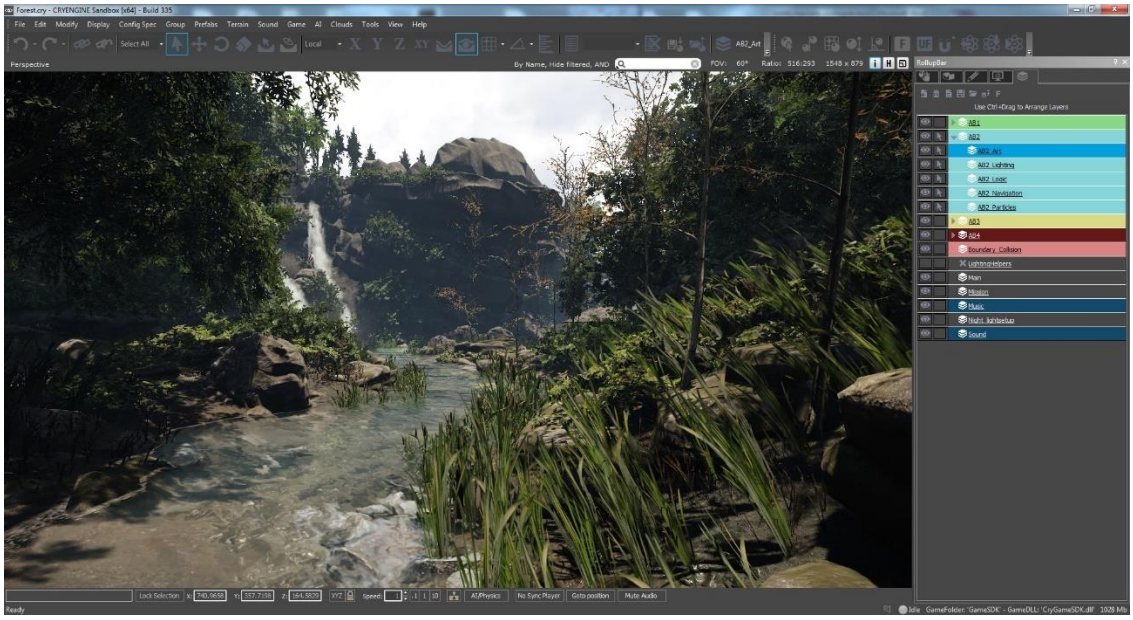


Рисунок 1.9 - Інтерфейс CryENGINE

Можливості CryENGINE:

- детальне промальовування рівнів та карт;
- можливість робити скріншоти та ділитися ними;
- імпорт об'єктів з графічних редакторів 3ds Max та Maya;
- можливість використовувати фотореалістичну графіку, елементи віртуальної реальності;
- наявність офіційного маркету з платними та безкоштовними моделями;
- функція попереднього перегляду локацій у режимі реального часу;
- ігри для комп’ютера можна портувати Sony PlayStation і Xbox 360.

Unreal Development Kit (UDK) (рис. 1.10) [11] - потужне середовище для створення ігор та програмування. Працює з відомими платформами Windows, Linux, Android, Xbox 360, PlayStation, PSP та іншими. Відрізняється від більшості

аналогів просунутою системою з налаштування фізики поведінки та взаємодії об’єктів навколишнього середовища. Дозволяє самостійно регулювати освітлення, тіні та ефекти.

Рисунок 1.10 - Інтерфейс Unreal Development Kit

Включає засоби Unreal Kismet, Cascade і Matinee, необхідні для якісної візуалізації ігрових подій. У розділі Epic Citadel можна знайти багато інтерактивного контенту для програми, яка створюється.

Сильні сторони:

- підтримка LAN та прямого підключення до IP;
- вбудована мова програмування UnrealScript;
- набори скриптів, спрайтів, текстур та звуків;
- можливість створення комп'ютерних та мобільних ігор;
- тестування проекту Unreal Engine у реальному часі;
- простий у використанні інтерфейс підказки для новачків.

Kodu Game Lab (рис. 1.11) [12] – відмінний варіант для створення комп'ютерних ігор різних жанрів без специфічних знань. Відрізняється наявністю потужного конструктора локацій, рівнів та об'єктів. Забезпечує комфортний процес розробки двовимірних та тривимірних моделей.

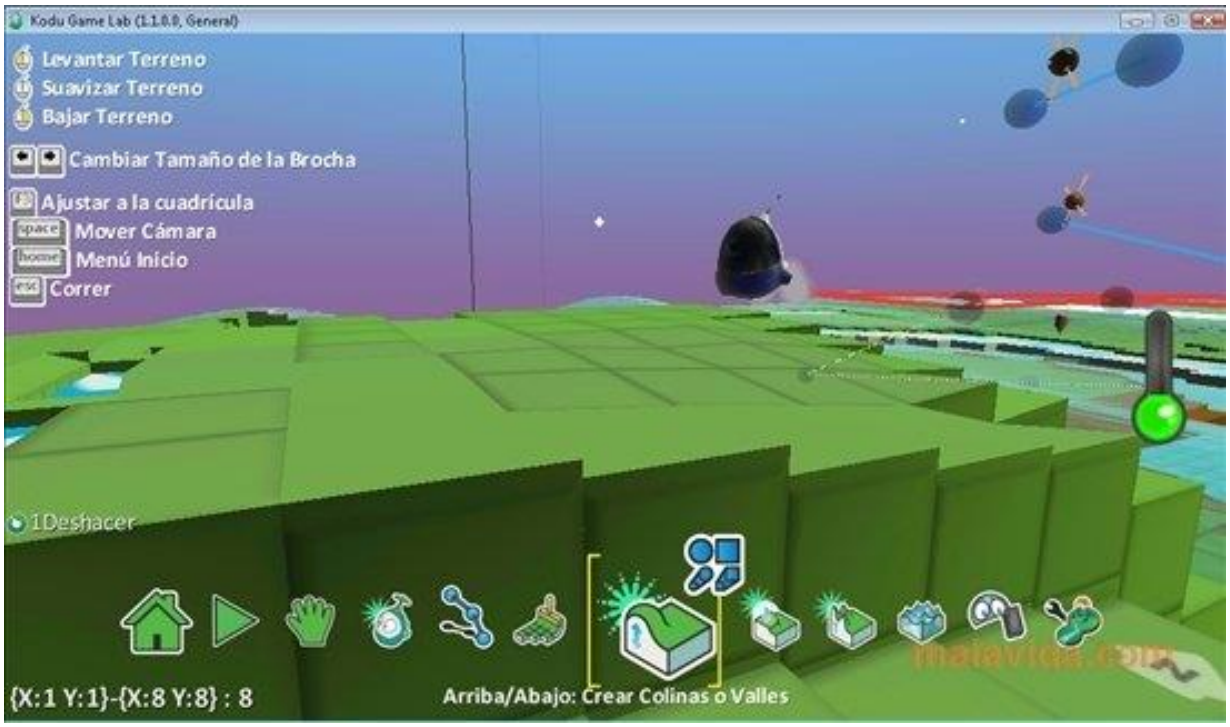


Рисунок 1.11 - Інтерфейс Kodu Game Lab

Програмне забезпечення стабільно отримує автоматичні оновлення від корпорації Майкрософт. Користувачі-початківці можуть пройти зручне навчання з основ роботи з програмою. Можливостей софту цілком вистачить на будівництво цілого ігрового світу, населеного різноманітними персонажами, які взаємодіятимуть друг з одним за заданими вами правилам.

Ключові переваги:

- інтеграція з Visual Studio;
- зберігання даних у хмарі Kodu;
- наявність зручної таблиці подій;
- має відкритий вихідний код;
- репост скріншотів у соціальні мережі;
- сучасний російськомовний інтерфейс;
- поворот камери (управління видом зверху, збоку);
- експорт мультиплатформенних ігор на PC, Xbox та Zune;
- використання високорозвиненого штучного інтелекту;
- підтримка сучасних технологій XNA Microsoft Game Studio.

NeoAxis 3D Engine (рис. 1.12) [13] надасть широкі можливості створення ігор для досвідчених програмістів. Перед вами відкриються такі інструменти, як: фізична система предметів, бібліотеки скриптів, засіб налаштування реакції об'єктів на певні дії та багато інших функцій.

Рисунок 1.12 - Інтерфейс NeoAxis 3D Engine

На окрему увагу заслуговує можливість складання високоякісних транспортних засобів та будівель. При побудові локації можна самостійно регулювати освітлення, деталізацію, рельєф карти та інші параметри.

Особливості софту:

- скриптинг на C# та .NET;
- вбудована підтримка мережі (онлайн ігри);
- набір готових дій для персонажів;
- підключення додаткових плагінів;
- сумісність із Windows Forms та Presentation Foundation.

1.4. Оцінювання тривалості світлофорного циклу за формулою Вебстера

Формула Вебстера використана для оцінки тривалості світлофорного циклу завдяки розповсюдженості закордоном, та в Україні зокрема. Вона дозволяє зрозуміти чи є оптимальною тривалість світлофорного циклу і на що треба звернути увагу задля покращення ситуації на дорогах.

Для оцінки взято період симуляції 15 хвилин. Формула Вебстера:

$$T_{\text{ц}} = \frac{1.5\sum_{i=1}^{n} t_{ni}+5}{1-\sum_{i=1}^{n} y_i}\text{с.}, \quad (1.1)$$

де tni – тривалість перехідного інтервалу в і-й фазі регулювання, с. yi – фазовий коефіцієнт і-ї фази, який визначають через співвідношення:

$$y_i = \frac{N_i}{S_i}, \quad (1.2)$$

де Ni – інтенсивність руху автівок у цьому напрямку в і-й фазі регулювання, авт./с. Si – потік насичення в цьому ж напрямку, авт./с.

За 42 фази регулювання, проїхали 232 машини, середній час простою 5.7887931034483 с., середня швидкість 48.793103448276 км/год, tni дорівнює 901, yi (1.2) дорівнює 12.328823895211. Тривалість світлофорного циклу за формулою Вебстера (1.1): 119.73881954096.

З точки зору безпеки, якщо тривалість циклу більше ніж 120, то така тривалість не є допустимою за Вебстером. Необхідно домогтися зниження тривалості циклу шляхом збільшення кількості смуг руху на підході до перехрестя, або заборони окремих маневрів, чи зниження кількості фаз регулювання, організації пропуску інтенсивних потоків протягом двох і більше фаз. Для повторної симуляції додано в формулу значення ще однієї полоси руху транспортних засобів. Як результат, вдалося зменшити тривалість світлофорного циклу за Вебстером до 57.338752018669. Але при регуляції роботи світлофорного циклу треба розуміти, що значення менше 25 також не є допустимими.

Висновок до розділу 1: у цьому розділі розглянуто сферу застосування автоматизованої системи управління дорожнім рухом, розглянуті основні проблеми, які тягне за собою неузгоджене ручне керування, описані функції і потреби до проектованої системи. Поставлені цілі розробки автоматизованої системи та описані заходи щодо покращення ситуації на дорогах. Крім того, розглянута більш детально проблематика старої системи управління, на основі якої зроблено відповідні висновки щодо покращення проектованої системи. Додатково розглянуто аналоги розроблюваної системи, такі як АСКДР SEA TCS, АСКДР підприємства «Росток-ЕЛЕКОМ» та «АТІЛОС». На основі розглянутих аналогів сформовано порівняльну таблицю аналогів для виявлення сильних і слабких сторін проектованої системи. Також проведено аналіз популярних платформ, аналогів Unity, які можуть забезпечити програмну і візуальну реалізацію симуляції трафіку. Розраховано і проведено оцінювання світлофорного циклу за формулою Вебстера.

2. ПРОЕКТУВАННЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ КЕРУВАННЯ ДОРОЖНІМ РУХОМ

Перший крок для проектування автоматизованої системи керування дорожнім рухом - це створення бази даних. База даних повинна включати в собі дані про камери, світлофори, датчики, їх поточний стан, режим роботи, та журнали пересування машин вулицями, а також журнал роботи світлофорів. Для цього розроблено концептуальну, контекстну, логічну та фізичну моделі бази даних.

2.1. Проектування концептуальної моделі бази даних

Концептуальна модель [14] бази даних це певна наочна діаграма, створена з використанням прийнятих позначень, і яка докладно показує зв'язок між об'єктами та його характеристиками. На концептуальній моделі у візуально зручному вигляді фіксуються зв'язки між об'єктами даних та їх характеристиками.

Перед тим, як почато безпосередньо проектування концептуальної моделі, розроблено функціональну модель системи, для того, щоб розуміти взаємозв’язок кожної окремої частини системи.

Функціональна модель розробки IoT систем керування світлофорами для нової ділянки дороги показана на рис. 2.1-2.3.

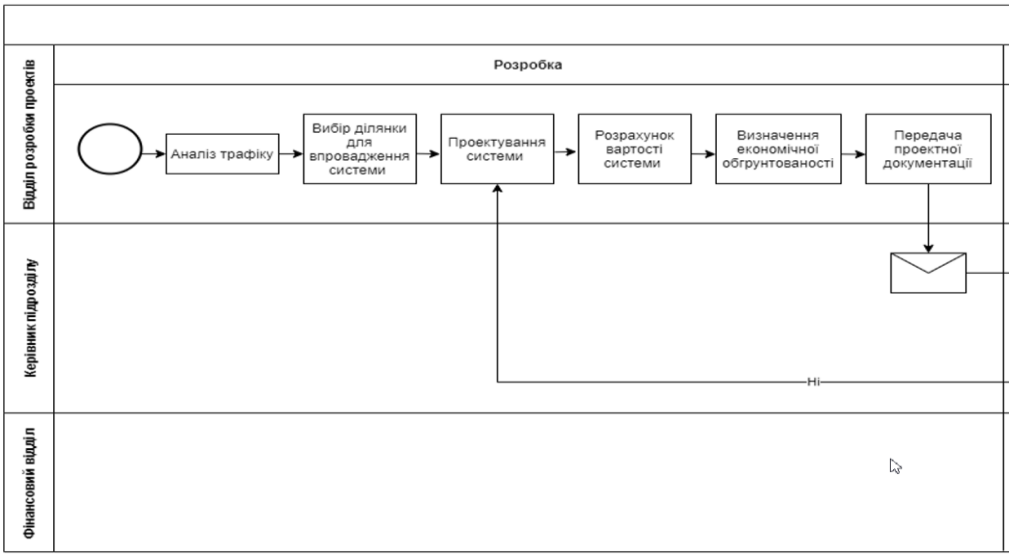


Рисунок 2.1 - Функціональна модель розробки IoT системи керування світлофорами для нової ділянки дороги частина 1

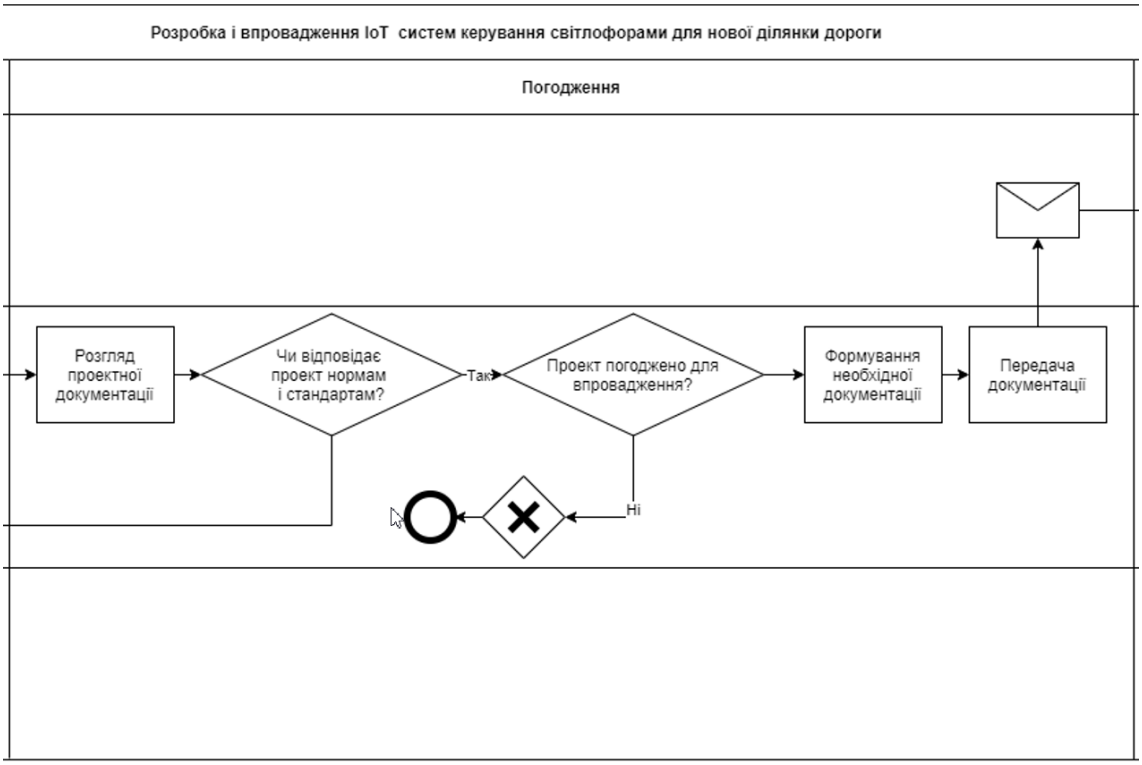


Рисунок 2.2 - Функціональна модель розробки IoT системи керування світлофорами для нової ділянки дороги частина 2

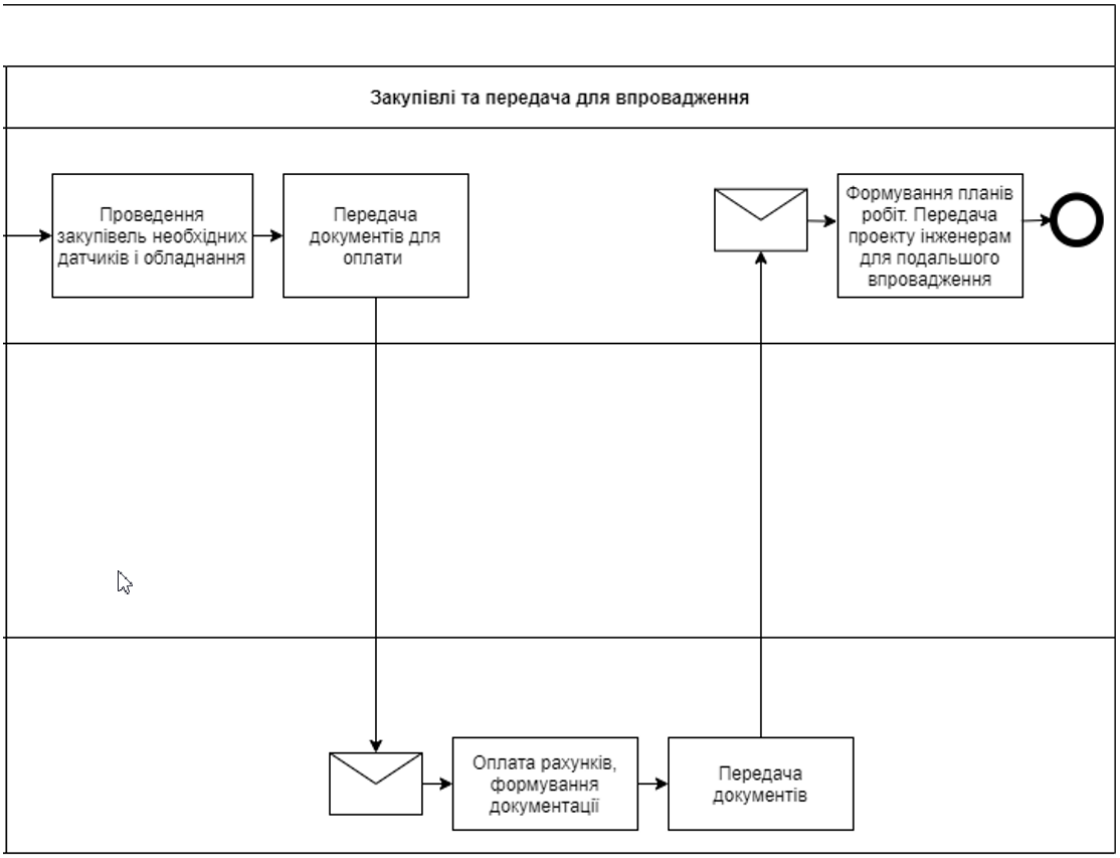


Рисунок 2.3 - Функціональна модель розробки IoT системи керування світлофорами для нової ділянки дороги частина 3

Як видно з функціональної моделі, впровадження автоматизації на нову ділянку дороги почато з аналізу трафіку на цій дорозі. Тобто, треба прорахувати чи потрібно взагалі вводити цю систему, які є «за» і «проти», які ризики, та яку вигоду буде отримано.

Відповідно, провівши аналіз, був отриманий висновок на яку вулицю потрібно інтегрувати систему автоматизації дорожнім рухом. Після цього, сплановано і проведено проектування самої системи. Треба розглянути усі аспекти робот, які будуть проводитися, і підійти до цього відповідально. Наприклад, якщо на певній вулиці планується заміна доріг і туди ще треба ввести автоматизовану систему керування дорожнім рухом, то це добра нагода для того, щоб використовувати петлеві датчики руху машин, які є дешевші і точніші ніж ультразвукові та можуть бути встановленими під дорогу. Це дозволить заощадити кошти і отримати більш точні статистичні дані.

Далі розраховано вартість цього проекту, скільки будуть коштувати додаткові комп'ютери для спеціалістів з дорожнього руху, матеріали, будівельно-монтажні роботи, датчики, камерне обладнання, кабелі, запасні частини, а також тренінги з користування системою. Якщо вартість проекту вкладається в бюджет і має відносно швидку окупність, то проект буде продовжуватися.

Після визначення економічної обґрунтованості, сформовано проектну документацію з описом робіт та їх вартості, термінами виконання робіт, з перерахування робітників і так далі. Далі керівник технічного відділу розглядає дану технічну документацію і переконується чи відповідає проект нормам і стандартам по виконанню аналогічних робіт. Якщо технічна документація відповідає усім зазначеним нормам, то далі формується уся наступна необхідна документація проекту, інакше проект закривається.

На основі документації, відділ, який займається безпосередньо розробкою проектів, виконує закупівлю усіх необхідних датчиків та обладнання. Для цього вони формують відповідні документи, які передають в фінансовий відділ. Фінансовий відділ оплачує рахунки і веде їх облік, а менеджер по закупівлях

виконує саму закупівлю і доставку обладнання. На основі проведених розрахунків і квитанцій формується остаточна документація з вказанням планів робіт і передача проекту інженерам для подальшого впровадження.

Наступним кроком, була розглянута ділова модель [15] підприємства, яка наведена у таблиці 2.1.

Таблиця 2.1 –

Ділова модель системи управління світлофорним регулюванням міста

Функції	**Класи**						
	Світлофори	**Датчики руху**	**Камери**	**Вулиці**	**Traffic Lights Modes**	**Traffic Lights Logs**	**Street Logs**
1	2	3	4	5	6	7	8
Дані про стан зберігаються в						*	*
Збирають дані		*	*				
Датчики та світлофори розміщено на				*			

Traffic Lights Modes, або режими роботи світлофорів, описують спосіб роботи відповідно світлофорів. Це зелений, червоний та жовтий кольори, а також час їх переключення.

Traffic Lights Logs, або Логи світлофорів, зберігають дані про стан системи. Тобто, логи світлофорів, - це певний журнал, де можна продивитися стан кожного світлофора на кожній вулиці, а також прослідити як змінювався його режим роботи протягом обраного періоду часу.

Street Logs, або Логи вулиць, зберігають дані про ситуацію на дорогах. Тобто, це журнал, де можна продивитися швидкість кожної машини, яка проїхала по певній вулиці, а також час, який ця машина стояла на червоному

світлі та у заторі. На основі цієї інформації система далі адаптується для забезпечення розвантаження потоку машин.

Збираються дані відповідно світлофорами, камерами і датчиками. За допомогою інтегрованих технічних засобів, ці компоненти об’єднанні в єдину систему автоматизованого керування.

Безпосередньо датчики, світлофори та камери розташовані на вулицях міста.

На основі описаної вище ділової моделі, розроблено концептуальну схему роботи системи управління світлофорним регулюванням міста. Ця схема представлена на рис. 2.4.

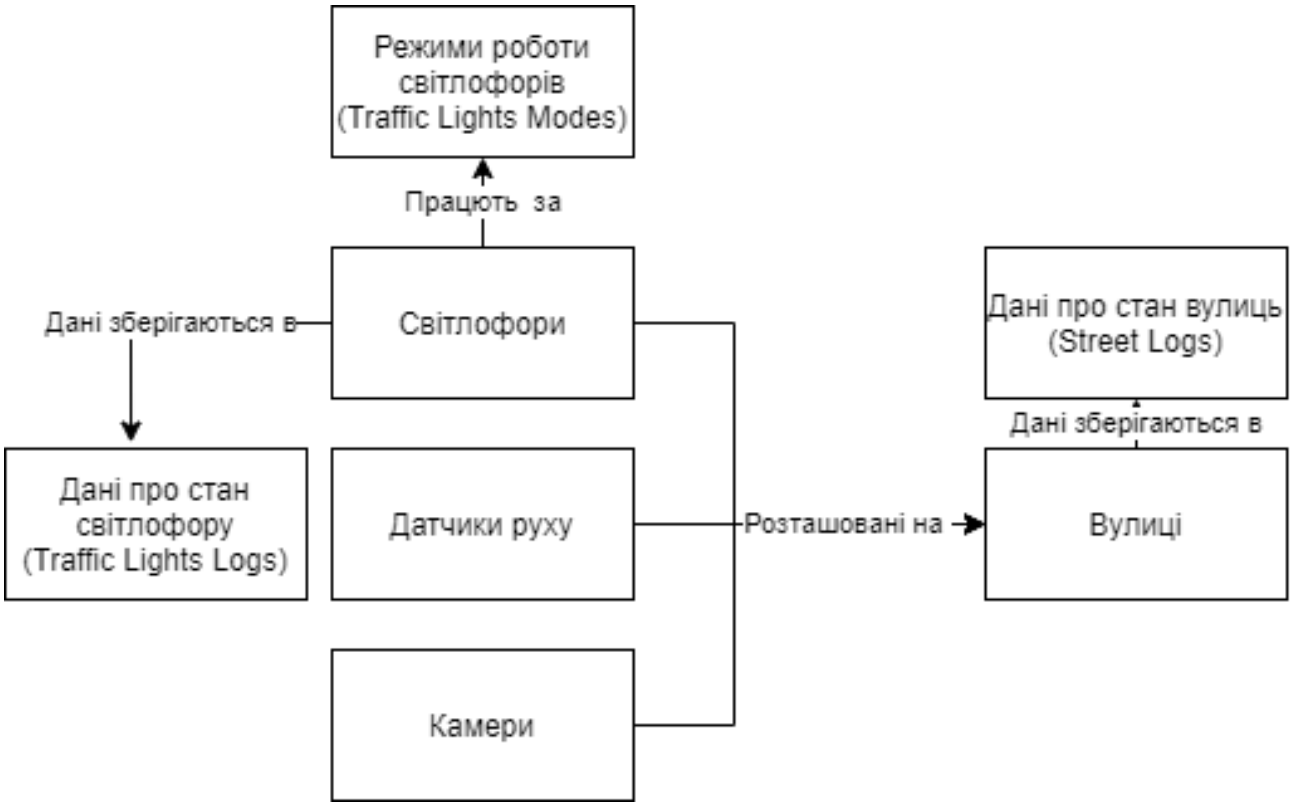


Рисунок 2.4 - Концептуальна схема роботи системи управління світлофорним регулюванням міста

Як видно зі схеми, перш за все система складається з світлофорів, датчиків і камер, з яких беруться дані про стан системи і зберігаються у журналі світлофорів (Traffic Lights Logs), які були описані вище. Світлофори в свою чергу працюють за «режимами роботи світлофорів». Усі елементи системи (датчики,

світлофори, камери) розташовані на улицях, дані про які зберігаються у журналі вулиць (Street Logs).

На основі концептуальної схеми був розроблений проект таблиць (таблиця 2.2), який містить базу даних системи автоматизованого керування дорожнім рухом.

Таблиця 2.2 – Структури таблиць для бази даних системи управління світлофорним регулюванням міста

Назва таблиці	**Назва стовпців**
1	2
Traffic Lights	id integer NOT NULL
	stateId (fk) integer NOT NULL
	modeId (fk) integer NOT NULL
	streetId (fk) integer NOT NULL
Traffic Lights Modes	id integer NOT NULL
	name varchar(6)
Traffic Lights States	id integer NOT NULL
	name varchar(14)
Traffic Lights Logs	currentStateId (fk) integer NOT NULL
	timeOfChange Datetime
	trafficLightId (fk) integer NOT NULL
Streets	id integer NOT NULL
	name varchar(32)
Streets Log	streetId (fk) integer NOT NULL
	timeOfArrived Datetime
	carSpeed float
	waitingTime integer
Cameras	id integer NOT NULL
	streetId (fk) integer NOT NULL
	stateId (fk) integer NOT NULL
Camera States	id integer NOT NULL
	name varchar(12)
Motion Sensors	id integer NOT NULL
	streetId (fk) integer NOT NULL

Продовження таблиці 2.2

Назва таблиці	Назва стовпців
1	2
Motion Sensors Types	typeId(fk) integer NOT NULL
	id integer NOT NULL
	name varchar(10)

Призначено id NOT NULL (для всіх ідентифікаторів).

Поле name в таблиці TrafficLightStates має обмеження на 14 символів (найбільше значення буде «OUT OF SERVICE»).

Для назв вулиць (поле name у таблиці Streets) призначено довжину рядка 32 символи.

Поле name таблиці MotionSensorTypes має обмеження на 10 символів. Поле буде мати два значення: loopback або ultrasonic.

Поле carSpeed таблиці StreetsLog має тип float, так як швидкість може бути не цілим значенням.

2.2. Проектування контекстної моделі бази даних

Контекстна діаграма моделі бази даних автоматизованої системи керування дорожнім рухом була розроблена у нотації IDEF0 [16] за допомогою програмного засобу AllFusion Process Modeler.

Перед тим як створити безпосередньо контекстну діаграму, були надані визначення стрілкам цієї діаграми (таблиця 2.3).

Таблиця 2.3 – Стрілки контекстної діаграми

Назва стрілки	Визначення стрілки	Тип стрілки
1	2	3
1. Міська транспортна система	На балансі підприємства перебуває близько 700 світлофорних об’єктів, більша половина яких підключена до центрального пункту керування, 600 пристроїв примусового зменшення швидкості, 54000 дорожніх знаків та 12 кольорових електронних табло	Input

Продовження таблиці 2.3

Назва стрілки	Визначення стрілки	Тип стрілки
1	2	3
2. Правила дорожнього руху, закони, норма та промислові стандарти	Діяльність підприємства регулюється законодавством, нормами та промисловими стандартами, що визначать характеристики системи в цілому та її окремих елементів.	Control
3. Прибуток	Діяльність підприємства спрямована на отримання прибутку шляхом - надання транспортних послуг, впровадження та експлуатація пристроїв примусового зниження швидкості та автоматизованих систем управління дорожнім рухом.	Output
4. Автоматична система управління світлофорним регулюванням міста	Система, що автоматично керує світлофорами на основі даних з датчиків руху, камер.	Output
5. Транспортні засоби	Транспортні засоби необхідні для надання спеціалізованих транспортних послуг, обслуговування транспортної системи, виконання будівельних, ремонтних та аварійно-відновлювальних робіт.	Mechanism
6. Працівники	Працівники виконують роботи в різних підрозділах підприємства.	Mechanism
7. Пристрої, датчики	Датчики та пристрої необхідні для роботи автоматичної система управління світлофорним регулюванням міста.	Mechanism

Контекстна діаграма (рис. 2.5) побудована на основі даних про підприємство, що містяться у таблиці 2.3.

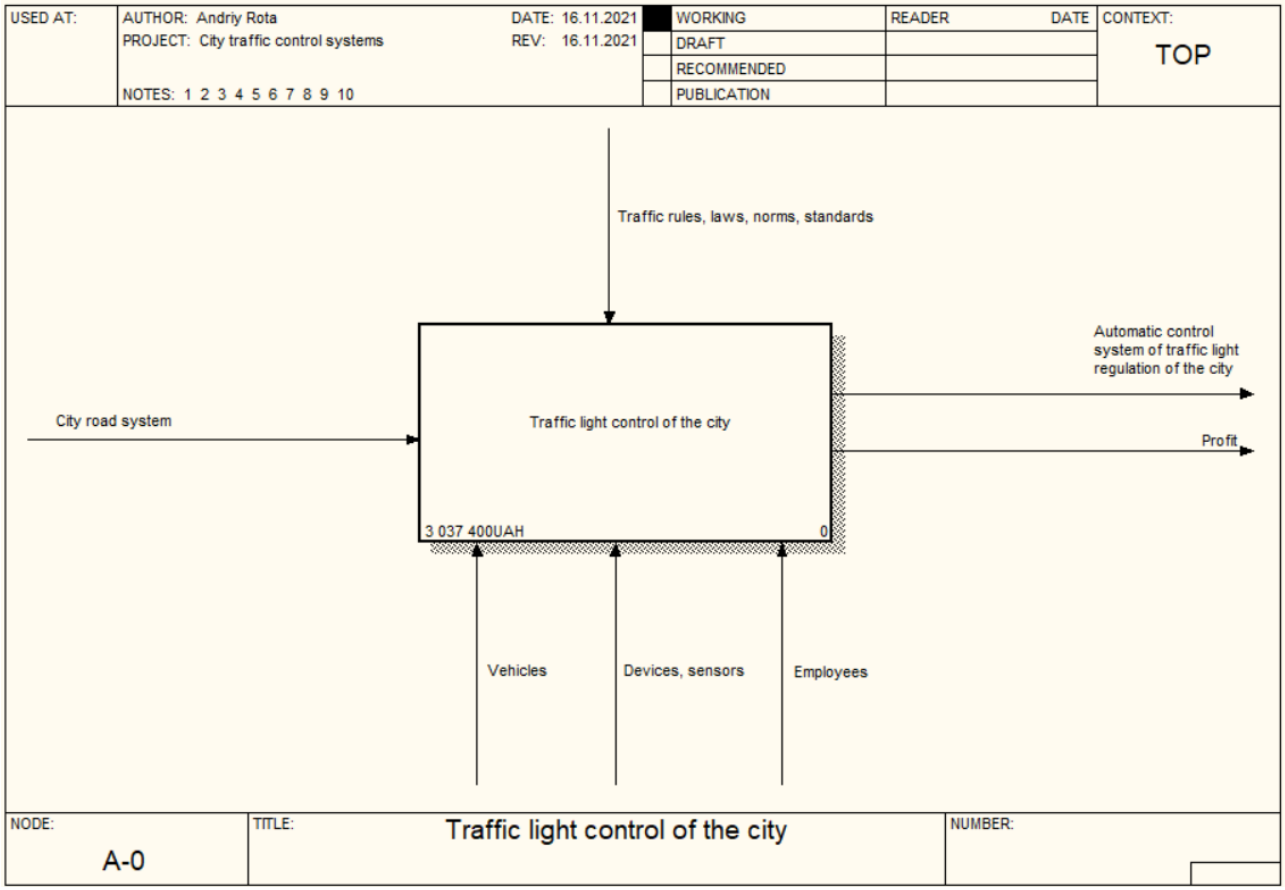


Рисунок 2.5 – Контекстна діаграма процесу діяльності підприємства, що реалізує системи управління світлофорним регулюванням міста в нотації IDEF0

Як видно з контекстної діаграми процесу діяльності підприємства, система приймає на вхід правила дорожнього руху, закони, нормативи, стандарті, систему доріг міста, транспортні засоби, датчики, світлофори, працівників, а на вихід автоматичний контроль і регуляцію потоку машин на вулицях, а також прибуток, який приносить ця система.

Для того, щоб більш детально розібрати усі аспекти і елементи, які забезпечують роботу системи, створено діаграму декомпозиції [17] основних напрямків діяльності системи (рис. 2.6) на основі описаних робіт відділів (таблиця 2.4).

Таблиця 2.4 – Роботи діаграми декомпозиції A0

Назва роботи	Визначення роботи
1	2
Діяльність технічних департаментів (Activities of technical departments)	Діяльність департаментів, що відповідають за виконання основних функцій підприємства (надання транспортних послуг, впровадження та експлуатація пристроїв примусового зниження швидкості та автоматизованих систем управління дорожнім рухом.).
Діяльність відділу реалізації (Sales activities)	Відділ відповідає за закупівлі та укладення договорів про надання послуг з сторонніми організаціями, фізичними та юридичними особами.
Фінансова діяльність (Financial activities)	Діяльність включає в себе ведення фінансового та податкового обліку, фінансове планування, виплату заробітної платні.
Процеси підтримки(Support process)	Об’єднують діяльність відділу кадрів та спеціалістів з охорони праці.

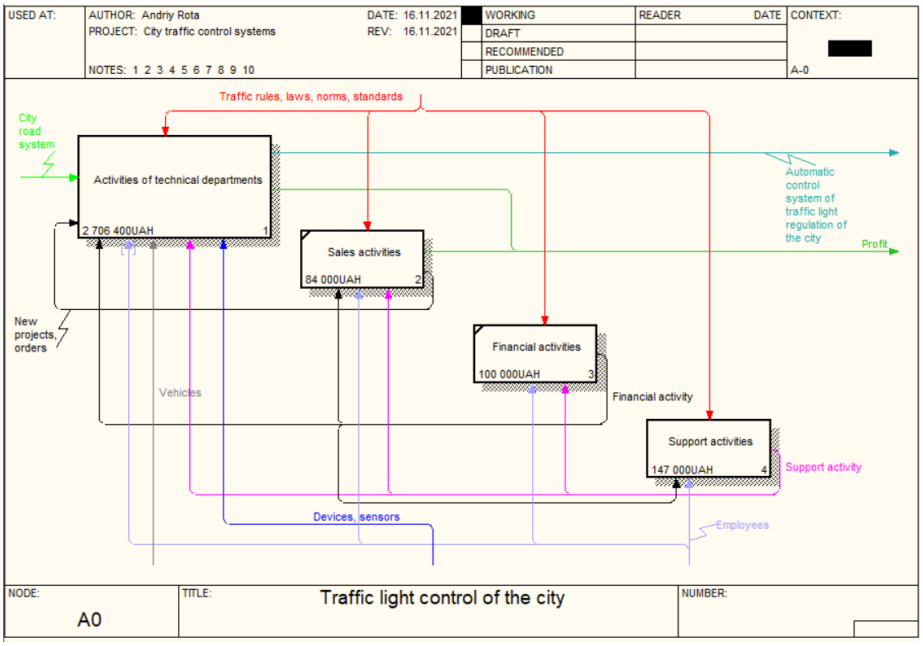


Рисунок 2.6 - Діаграма декомпозиції основних напрямів діяльності підприємства, що реалізує система управління світлофорним регулюванням міста в нотації IDEF0

Для повного розуміння основної діяльності роботи компанії була розроблена таблиця робіт технічного департаменту (таблиця 2.5) та на її основі створено діаграму декомпозиції діяльності технічних департаментів (Activities of technical departments) підприємства (рис. 2.7), що реалізують саму систему управління світлофорним регулюванням міста в нотації IDEF0.

Таблиця 2.5 – Роботи діаграми декомпозиції A0

Назва роботи	**Визначення роботи**
1	2
Діяльність інженерів, електромонтерів та інших технічних спеціалістів (Engineering activities)	Роботи з впровадження автоматичної системи управління світлофорним регулюванням міста та інші функції.
Автоматична система управління світлофорним регулюванням міста (Automatic control system of traffic light regulation activities)	Обслуговування та забезпечення роботи автоматичної системи управління світлофорним регулюванням міста
Транспортна діяльність (Logistic activities)	Надання спеціалізованих транспортних послуг, обслуговування транспортної системи, виконання будівельних, ремонтних та аварійно-відновлювальних робіт.

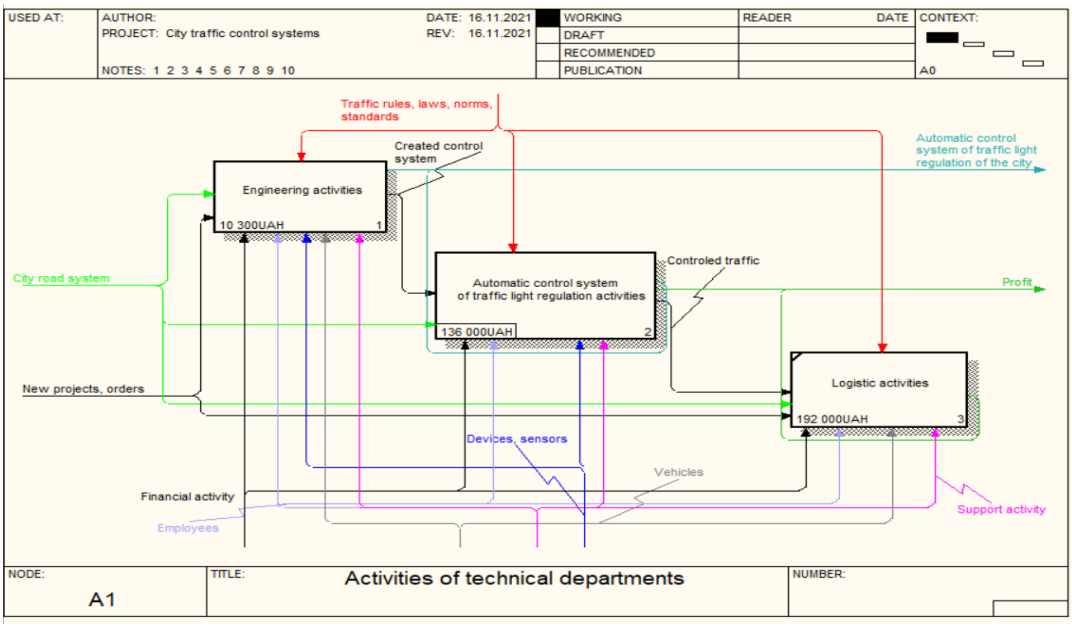


Рисунок 2.7 - Діаграма декомпозиції діяльності технічних департаментів, що реалізує системи управління світлофорним регулюванням міста в нотації IDEF0

Допоміжні процеси діяльності підприємства представлені у таблиці 2.6 та на їх основі була розроблена діаграма декомпозиції допоміжних процесів діяльності підприємства (рис. 2.8), що реалізує систему управління світлофорним регулюванням міста в нотації IDEF0.

Таблиця 2.6 – Роботи діаграми декомпозиції A0

Назва роботи	**Визначення роботи**
1	2
Відділ кадрів (Human resources activities)	Обмін відомостями щодо працівників компанії, звільнення та працевлаштування.
Спеціалістів з охорони праці (Labor protection activities)	Нагляд за дотриманням компанією вимог законодавства, нормативів тощо з охорони праці. Пропозиції щодо покращення рівня охорони праці.

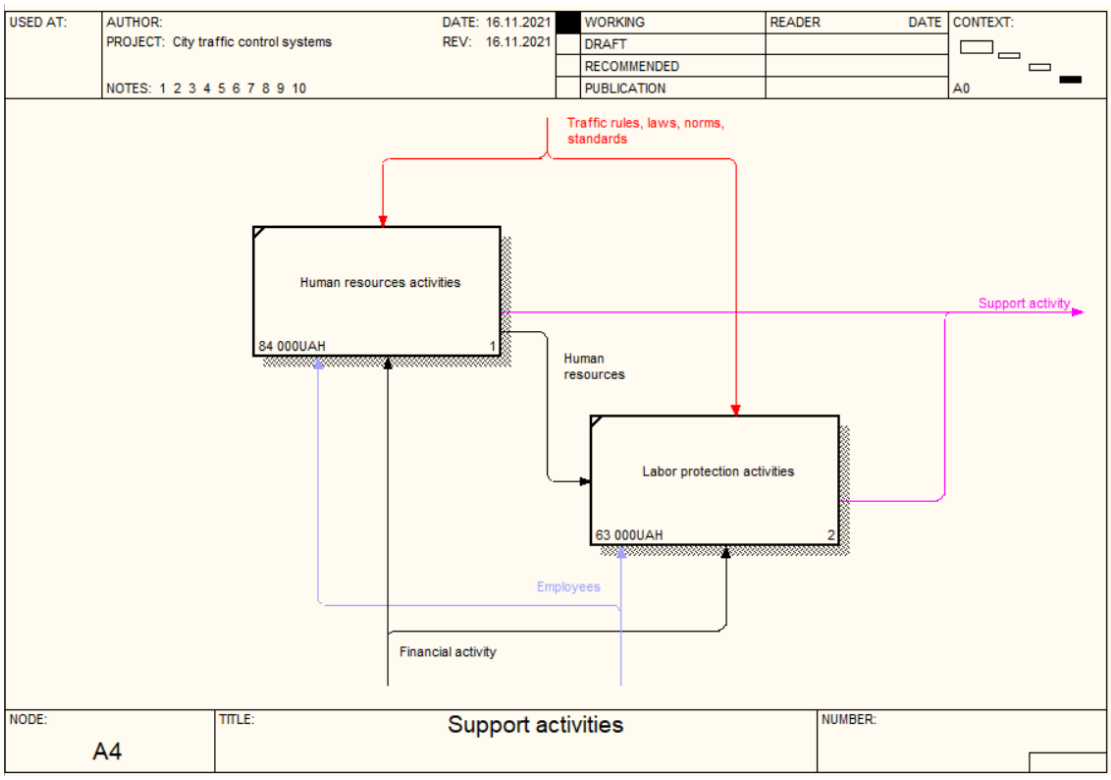


Рисунок 2.8 - Діаграма декомпозиції допоміжних процесів діяльності підприємства, що реалізує системи управління світлофорним регулюванням міста в нотації IDEF0

На основі вищеописаних діаграм отримано дерево вузлів підприємства [18], що реалізує системи управління світлофорним регулюванням міста представлене на рис. 2.9.

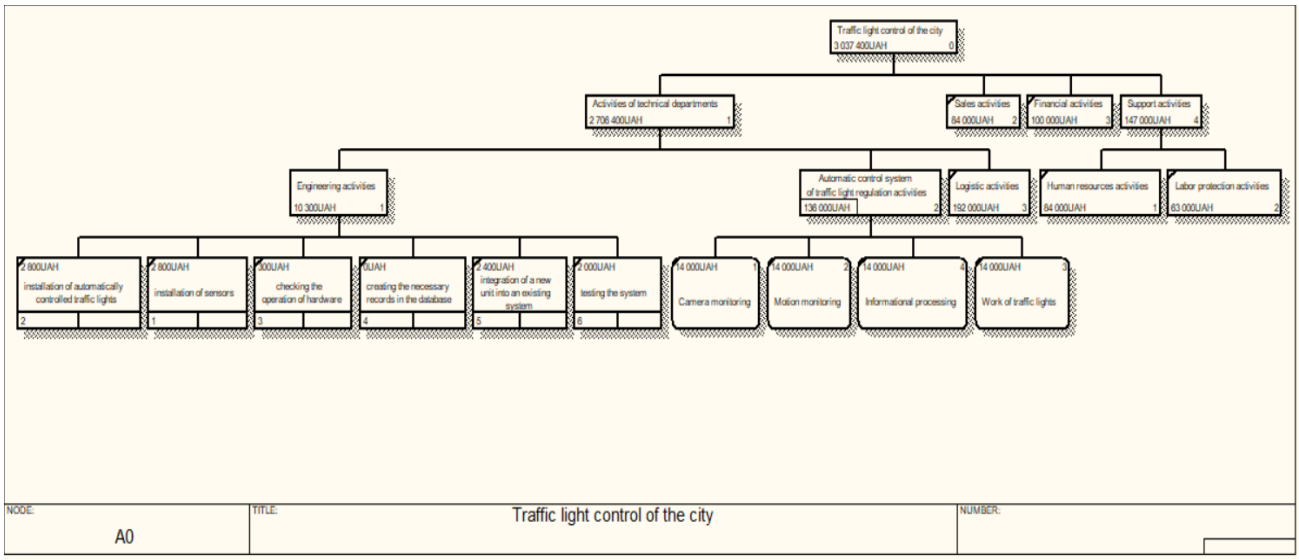


Рисунок 2.9 – Діаграма дерева вузлів підприємства, що реалізує системи управління світлофорним регулюванням міста

Діаграма дерева вузлів відображає діяльність усіх підрозділів компанії та їх підпорядкування. Зі схеми видно, що система автоматизованого управління дорожнім рухом включає в себе процеси технічного відділу, відділу продаж, фінансів та підтримки. Технічний відділ включає в себе інженерні процеси, процеси з автоматизації та логістики. Відділ підтримки включає в себе усі процеси пов'язані з людським ресурсом і захистом їх робочого процесу.

Наступним кроком розроблено модель системи у нотації DFD. Модель представлена на рис. 2.10.

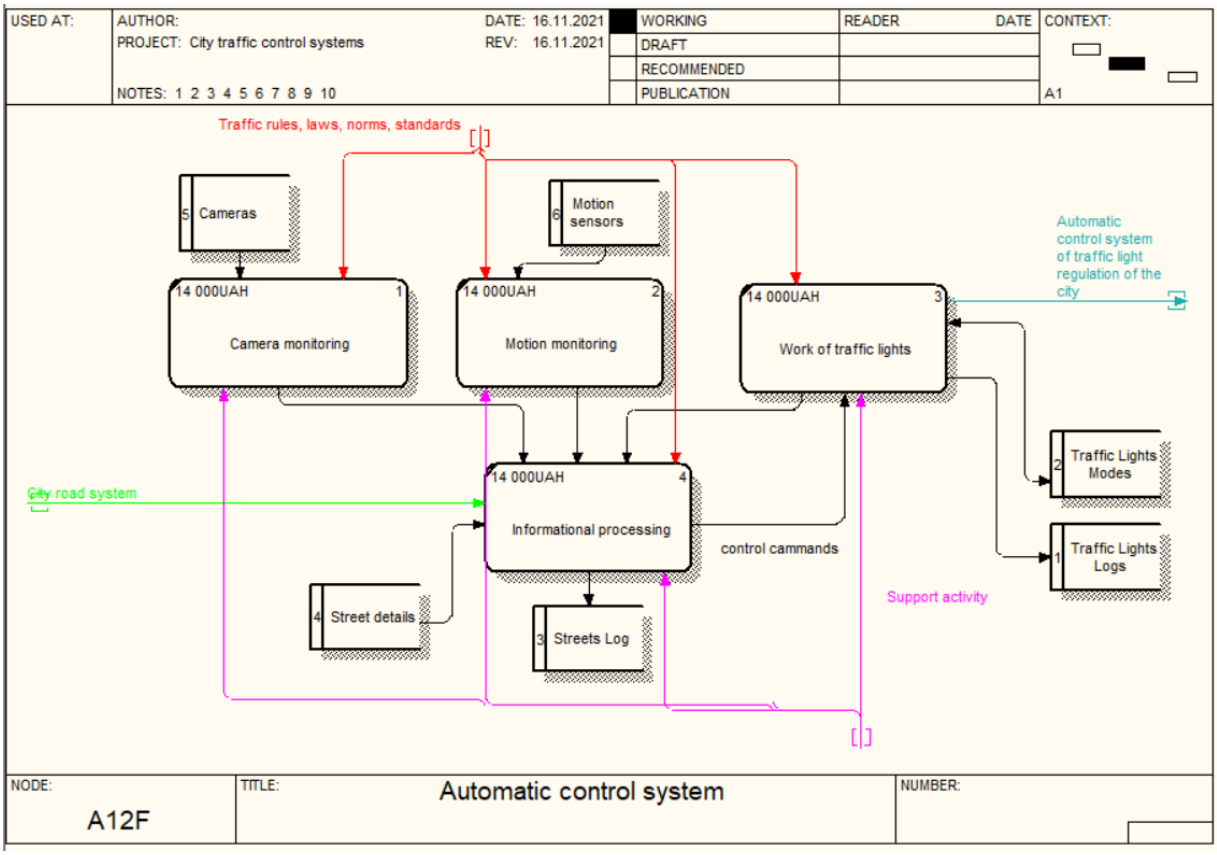


Рисунок 2.10 - Результат виконання нотації DFD

Модель системи відображає частини автоматизованої системи та їх взаємодію. Правила дорожнього руху, закони і стандарти є базою для роботи. Камери включають в себе процеси з моніторингу, які підпорядковуються інформаційним процесам, як і робота світлофорів та датчиків. Інформаційні процеси формують журнал вулиць, в якому ведуться записи машин. Відповідно процес функціонування світлофорів формує свій журнал, в якому відображено час переключення режимів роботи.

Для ілюстрації моделі процесу діяльності підприємства з точки зору процесів, їх взаємозв’язків і формування доходу, зроблено діаграму "тільки для експозиції" (FEO) [19] (рис. 2.11).

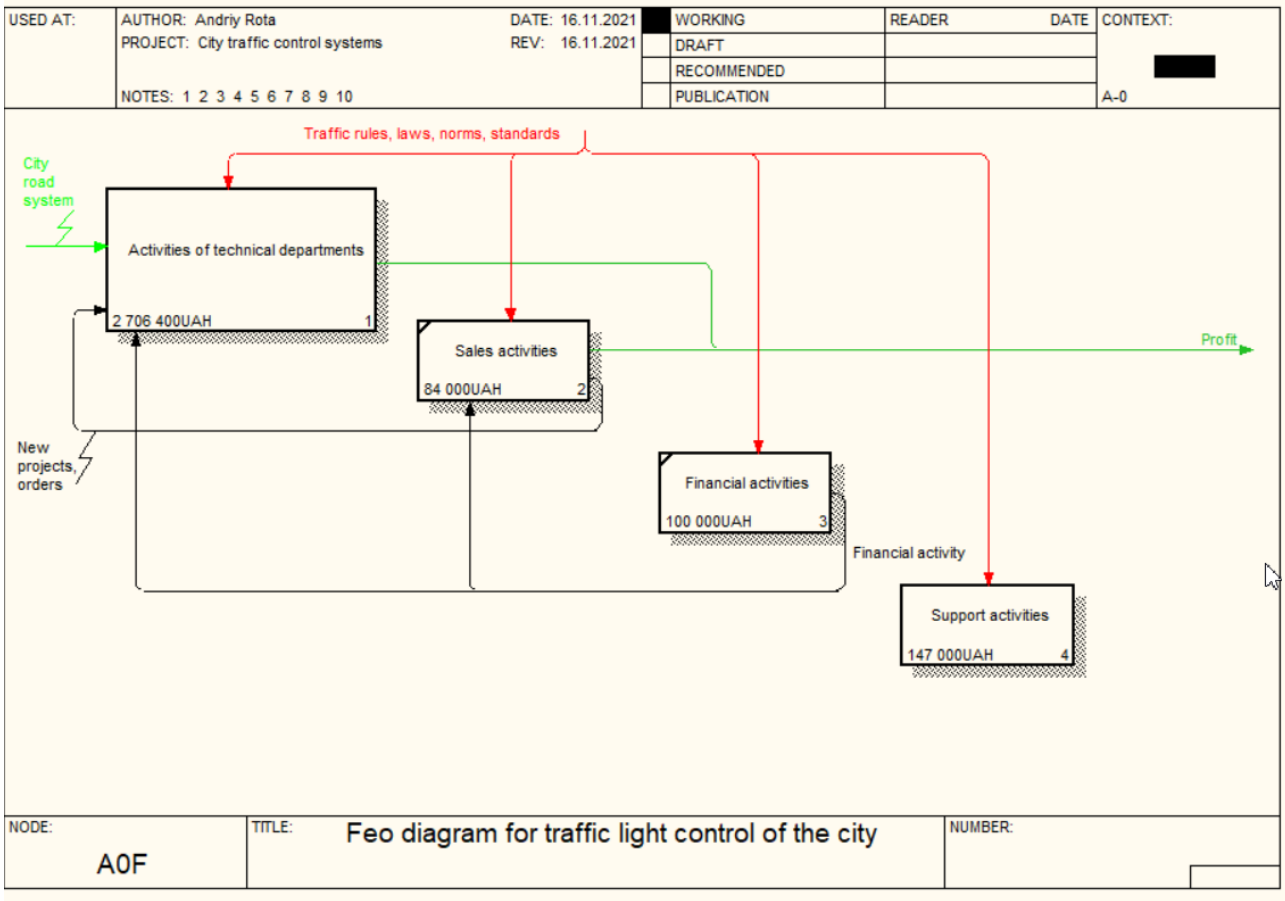


Рисунок 2.11 - FEO діаграма декомпозиції процесу основних напрямів діяльності підприємства, що реалізує системи управління світлофорним регулюванням міста в нотації IDEF0

У цій FEO діаграмі видно, що система дорожнього руху забезпечує виконання робіт технічного департаменту, що є логічним. Для виконання саме технічних робіт виділено 2 706 400 гривень. Також для роботи технічного відділу відповідно потрібні нові проекти (замовлення) і виконання повинно підлягати усім законам, нормам і правилам дорожнього руху та дорожнього проектування.

У цій діаграмі також представлені процеси з продажу, які підтримуються фінансовим відділом, який також забезпечує роботу технічного відділу, бо без закупівлі необхідного обладнання неможливо проводити технічні роботи.

На процеси з продажу виділені 84000 гривень, на фінансовий відділ

виділено 100 000 гривень, та 147 000 гривень виділено на процеси підтримки усіх інших процесів.

Далі розглянуто сценарій впровадження нової частини системи (тобто вулиці) на рис. 2.12.

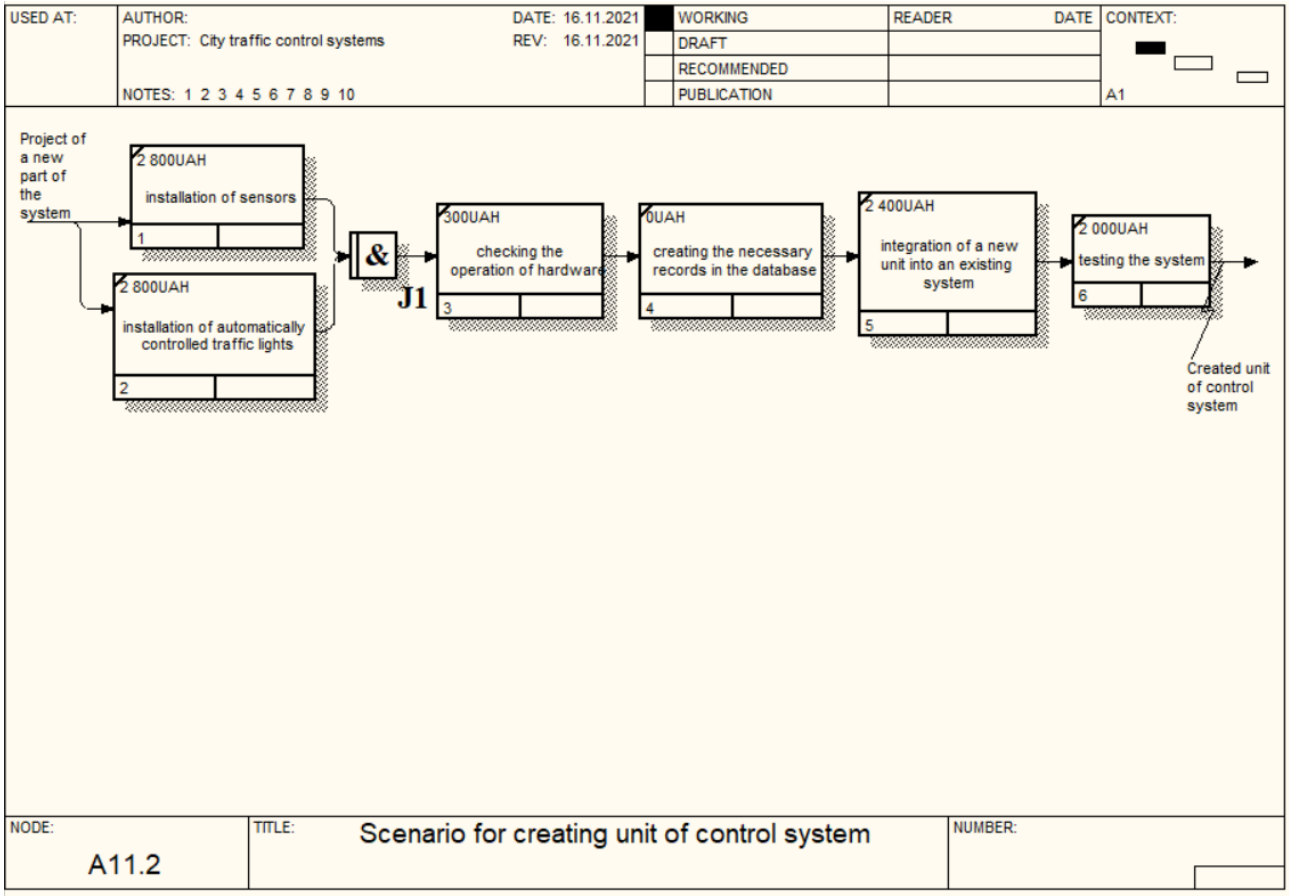


Рисунок 2.12 – Результат виконання сценарію впровадження автоматизованої системи керування світлофорами на нову вулицю

Установка нових датчиків та автоматизованих світлофорів є початком процесу впровадження системи на нову вулицю. Далі перевірено працездатність програмного забезпечення та проведено аналіз цілісності нової системи. Для того, щоб збирати статистичні дані, створено нові необхідні записи в базі даних та «заінтегрована» нова вулиця до існуючої системи. Фінальною частиною є тестування працездатності компонентів системи на новій вулиці.

2.3. Проектування логічної моделі бази даних

Після розробки контекстної моделі бази даних, спроектовано таблиці, які будуть входити до бази даних та описані усі рядки, що будуть містити ці таблиці

та обмеження на тип даних. Наступним кроком є створення логічної моделі[20].

Для правильної побудови логічної моделі відображено взаємозв’язки окремих таблиць бази між собою.

В першу чергу, перед безпосереднім проектуванням логічної моделі, були розглянуті варіанти використання системи, представлені на діаграмі прецедентів [21] (рис. 2.13).

Рисунок 2.13 - Діаграма прецедентів (use case diagram) діяльності підприємства, що реалізує системи управління світлофорним регулюванням міста

З діаграми видно, що є три «актора» - оператор системи автоматизованого керування, водії та технічний спеціаліст. Вони виконують обслуговування інформаційної системи, управління світлофорним регулюванням містом та обслуговування датчиків відповідно. На діаграмі показано, що вищевказані процеси включають в себе ще такі процеси, як розрахунок оптимального часу перемикання світла, визначення кількості автомобілів, часу простою, середньої швидкості і таке інше.

Далі розглянуто діаграму діяльності [21] системи автоматизованого керування дорожнім рухом (рис. 2.14).

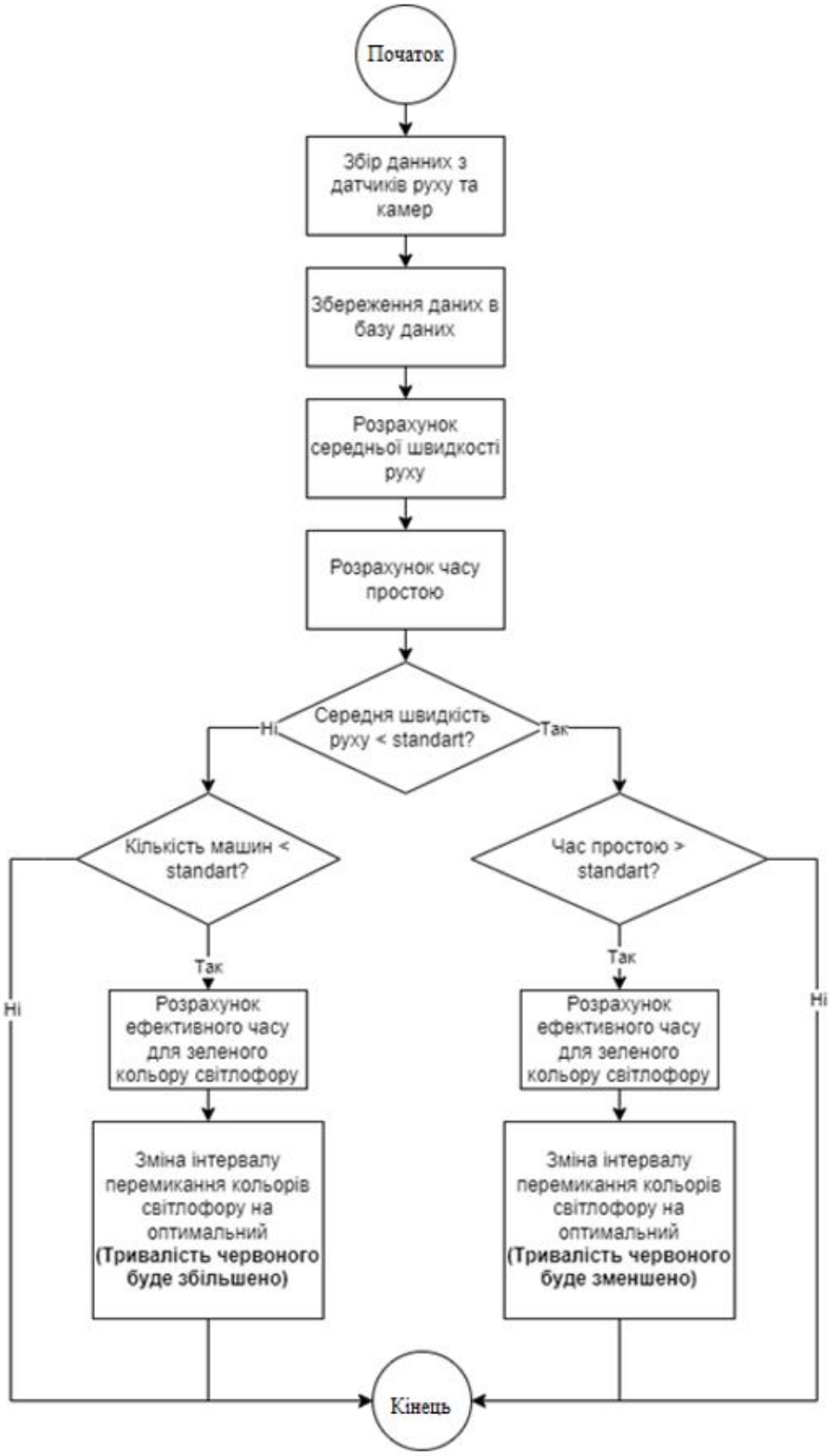


Рисунок 2.14 – Діаграма діяльності (activity diagram) для IoT системи управління світлофорним регулюванням міста

Схема відображає процес адаптації системи до поточного стану доріг. Якщо дороги навантаженні, тобто великий час простою машин, то треба зменшити час простою на червоному світлі, і навпаки, якщо певна вулиця

розвантажена для машин, то треба збільшити час роботи червоного кольору для транспортних засобів.

Наступним кроком перед тим, як побудувати логічну модель бази даних, є створення діаграми послідовності [21][22] роботи системи (рис. 2.15). Вона дає розуміння підпорядковування процесів у системі та їх взаємозв'язок.

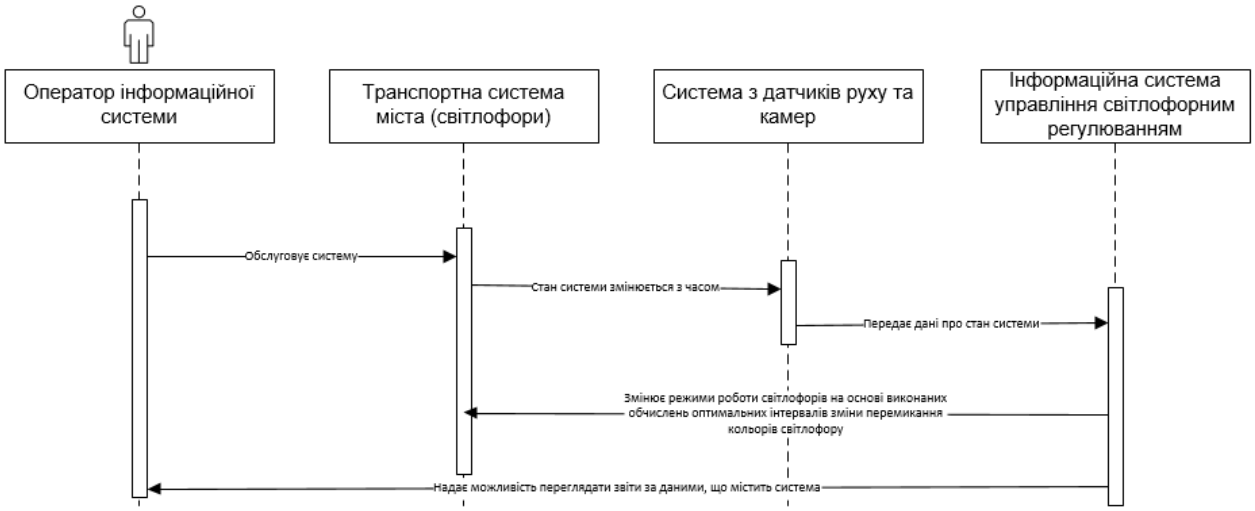


Рисунок 2.15 - Діаграма послідовності (sequence diagram) для інформаційної системи управління світлофорним регулюванням міста

Перш за все, оператор системи автоматизованого керування дорожнім рухом займається обслуговуванням самої системи, яка в свою чергу змінює свій стан з певним часом. Датчики та камери, які входять в транспортну систему, передають дані про свій стан у базу даних. Автоматизована система використовує отриманні з бази дані для того, щоб світлофори адаптувались до поточної ситуації на дорогах і змінили свій колір. Оператор в свою чергу отримує можливість переглядати звіти за даними, що містить система.

Для легшого розуміння як правильно побудувати логічну систему бази даних представлено усі таблиці майбутньої бази у вигляді діаграми класів [21][23] (рис. 2.16).

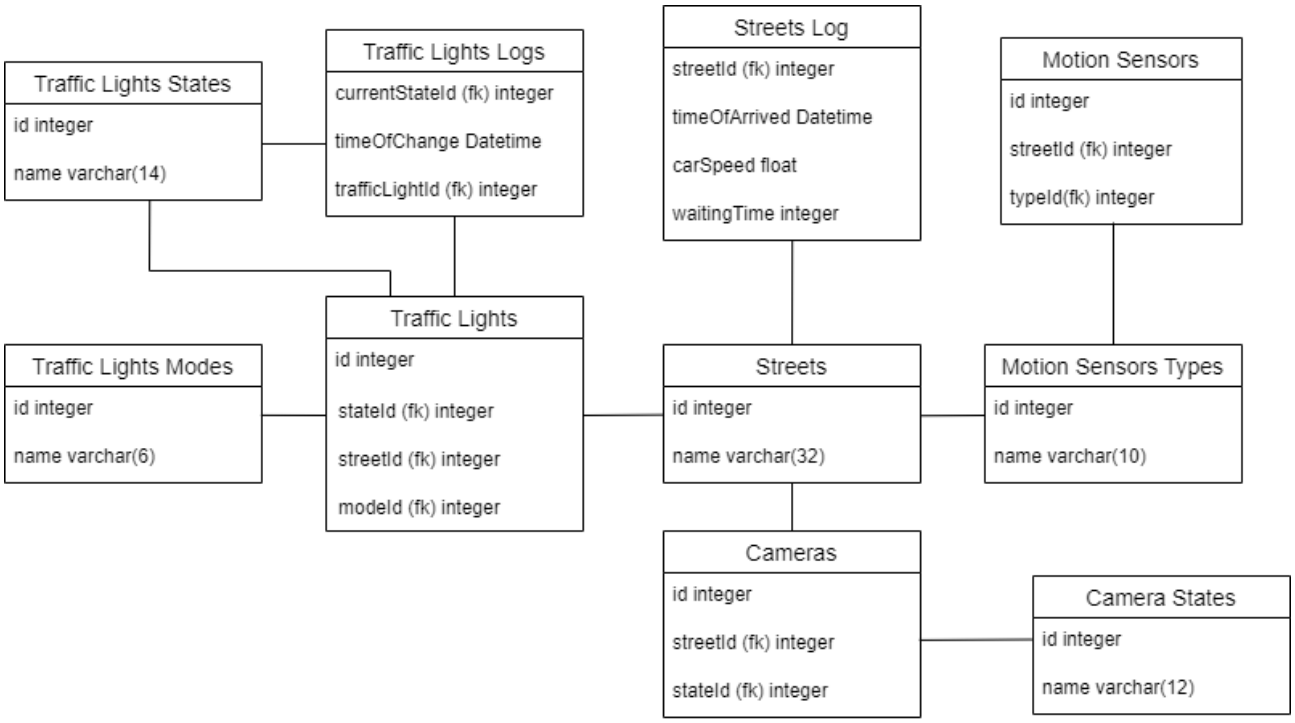


Рисунок 2.16 - Діаграма класів (class diagram) для інформаційної системи управління світлофорним регулюванням міста

Ієрархія таблиць у Erwin Data Modeler представлена на рис. 2.17. На основі діаграми класів розроблено логічну модель бази даних за допомогою Erwin Data Modeler 7.3 (рис. 2.18).

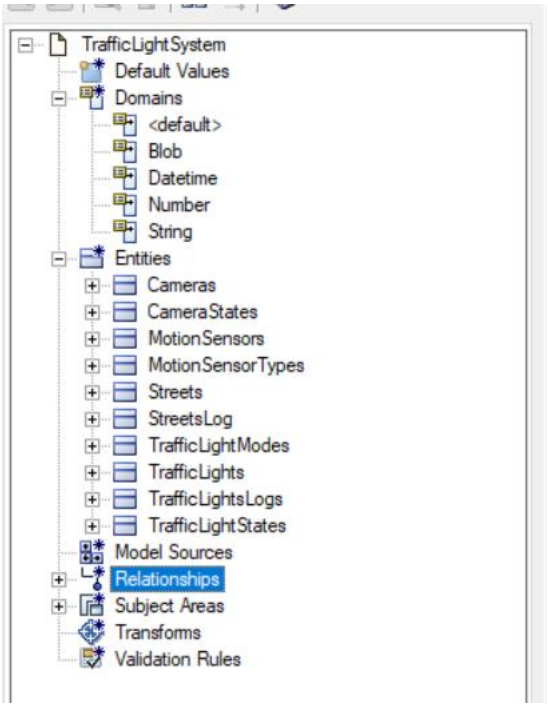


Рисунок 2.17 – Список таблиць бази даних автоматизованої системи керування дорожнім рухом

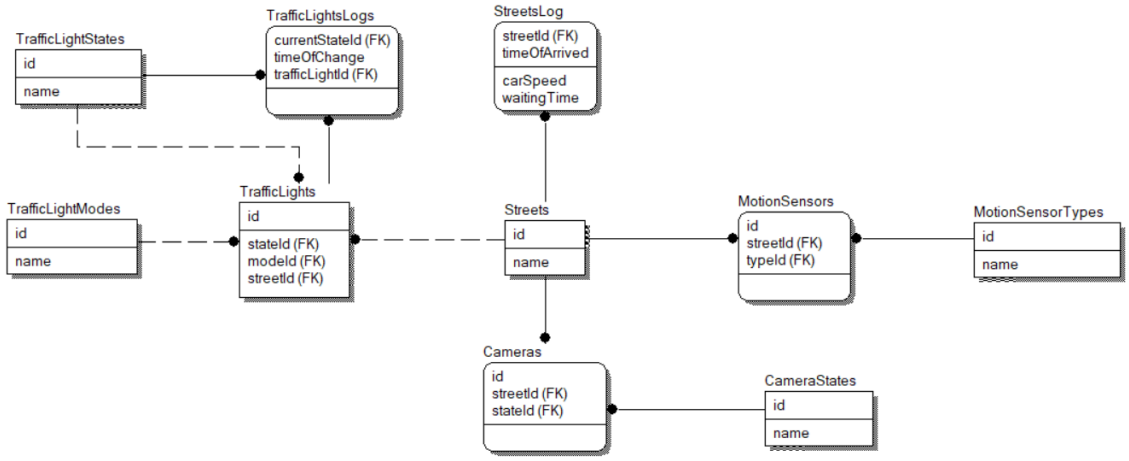


Рисунок 2.18 – Логічна модель бази даних автоматизованої системи керування дорожнім рухом

2.4. Проектування фізичної моделі бази даних

Для того, щоб розуміти усі моделі реляційних даних та окремі об’єкти бази даних використовується фізична модель [24]. Також ця модель використовується для того, щоб чітко оцінити розмір бази даних та спланувати потужності необхідні для її проектування, визначити обмеження даних, такі як їх розмір, безпека, і таке інше.

У цьому випадку фізична модель (рис. 2.19) була розроблена шляхом перетворення в неї логічної моделі за допомогою Erwin Data Modeler 7.3.

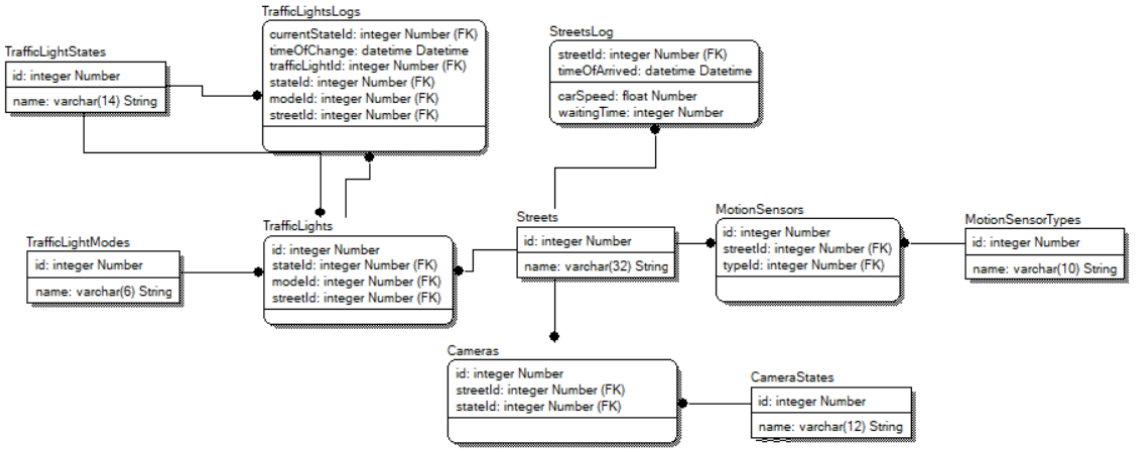


Рисунок 2.19 – Фізична модель бази даних автоматизованої системи керування дорожнім рухом

2.5. Проектування бази даних автоматизованої системи керування дорожнім рухом

База даних проектувалася у веб-додатку PhpMyAdmin[25].

Після авторизації в додаток диспетчер потрапляє на головну сторінку, де у лівому або верхньому меню можна обрати вкладку «Створити БД», або «Бази даних», відповідно. У цьому розділі є інструменти для налаштування назви бази даних, полів, а також для задання обсягу на довжини рядків, на їх тип, і так далі.

Готова база даних веб-додатку показана на рис. 2.20.

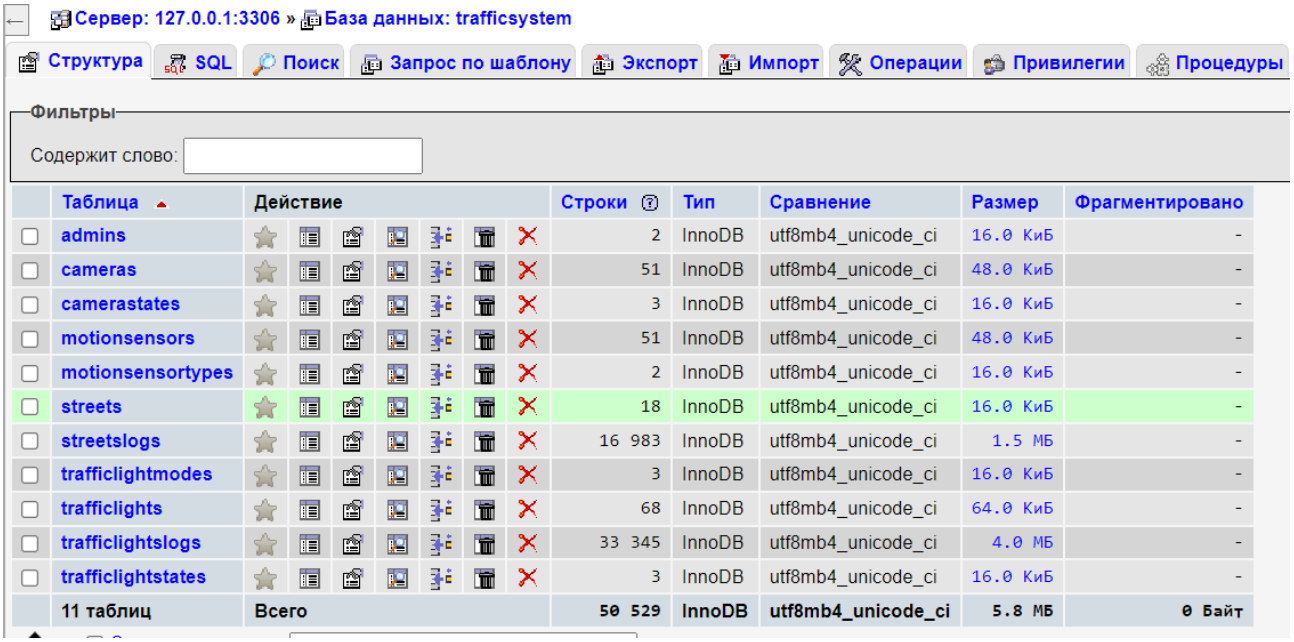


Таблица	Действие	Строки	Тип	Сравнение	Размер	Фрагментировано
admins		2	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
cameras		51	InnoDB	utf8mb4_unicode_ci	48.0 КиБ	-
camerastates		3	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
motionsensors		51	InnoDB	utf8mb4_unicode_ci	48.0 КиБ	-
motionsensortypes		2	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
streets		18	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
streetslogs		16 983	InnoDB	utf8mb4_unicode_ci	1.5 МБ	-
trafficlightmodes		3	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
trafficlights		68	InnoDB	utf8mb4_unicode_ci	64.0 КиБ	-
trafficlightslogs		33 345	InnoDB	utf8mb4_unicode_ci	4.0 МБ	-
trafficlightstates		3	InnoDB	utf8mb4_unicode_ci	16.0 КиБ	-
11 таблиц	Всего	50 529	InnoDB	utf8mb4_unicode_ci	5.8 МБ	0 Байт

Рисунок 2.20 – база даних веб-додатку trafficsystem

У базі даних представлені такі таблиці:

- **admins** – таблиця адміністраторів;
- **cameras** – таблиця камер;
- **camerastates** – таблиця станів камер;
- **motionsensors** – таблиця датчиків;
- **motionsensortypes** – таблиця типів датчиків, streets – таблиця вулиць;
- **streetslogs** – таблиця логів вулиць;
- **trafficlightmodes** – таблиця режимів роботи світлофора;
- **trafficlights** – таблиця світлофорів;
- **trafficlightslogs** – таблиця логів світлофорів;

- **trafficlightstates** – таблиця станів світлофорів.

Для прикладу зображено дані по вулицям на рис. 2.21.

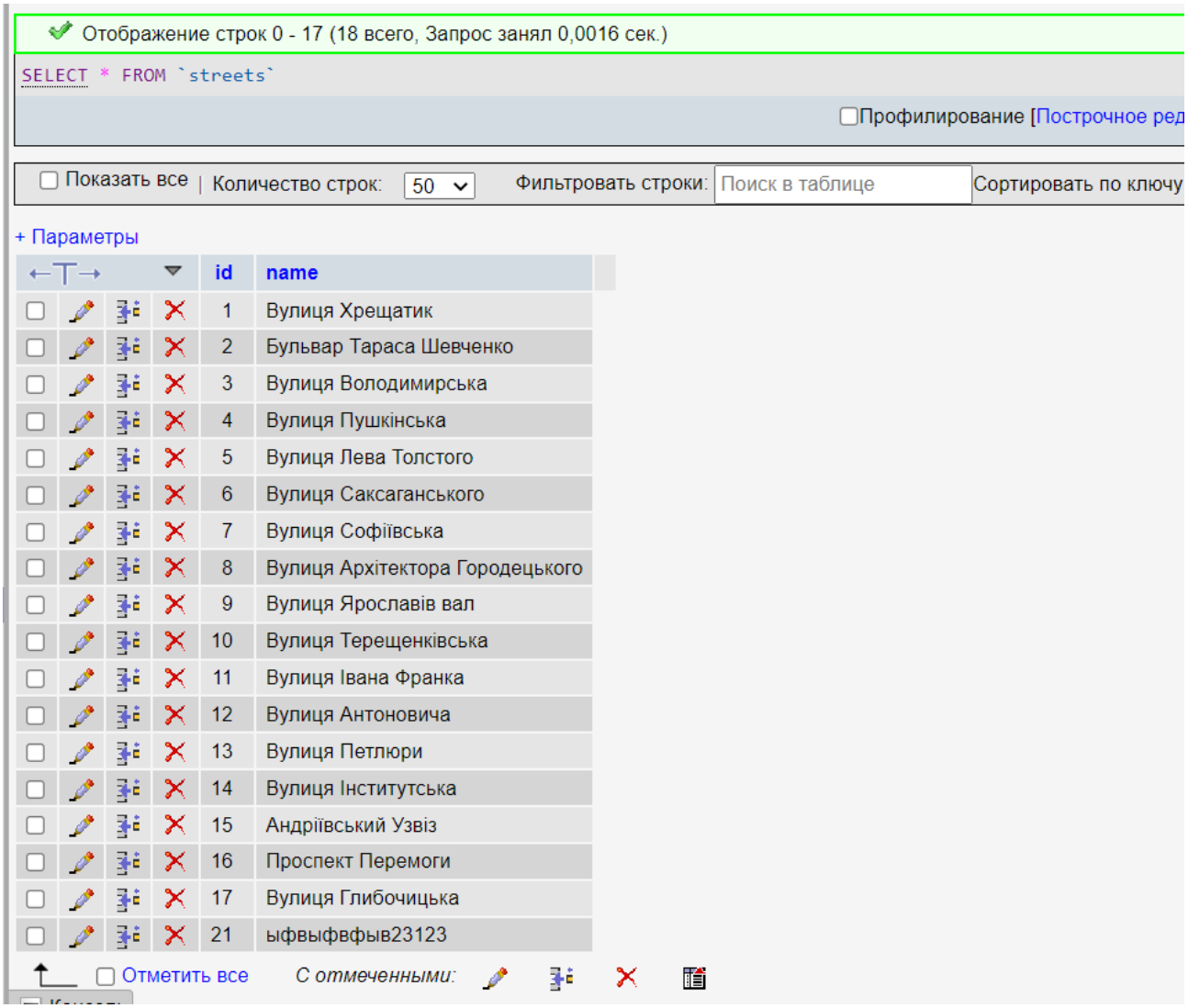


Рисунок 2.21 – Приклад наповнення таблиці вулиць бази даних веб-додатку

2.6. Проектування інтерфейсу панелі адміністратора CRM автоматизованої системи керування дорожнім рухом

Інтерфейс панелі адміністратора був розроблений за допомогою мови розміщення гіпертексту HTML5[26] та мови стилю сторінок CSS3[27].

Першим кроком створено сторінку авторизації (рис. 2.22) та безпосередньо форму авторизації (файл login.php).

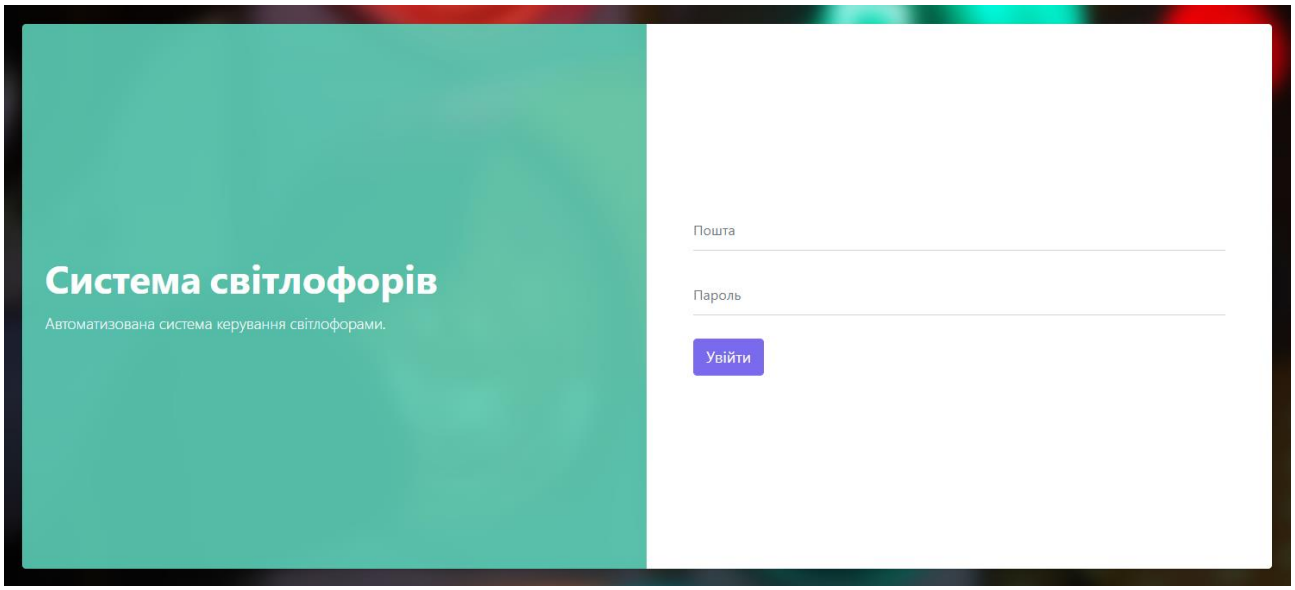


Рисунок 2.22 – Сторінка входу в CRM-систему

Наступним кроком створено головну сторінку (index.php) без наповнення (рис. 2.23). Створено верхнє меню з назвою CRM-системи та кнопкою виходу, створено бокове меню з розділами «Головна», «Вулиці», «Датчики руху», «Типи датчиків руху», «Камери», «Світлофори», «Симуляція трафіку», «Логи системи» з підпунктами «Логи світлофорів» та «Логи вулиць».

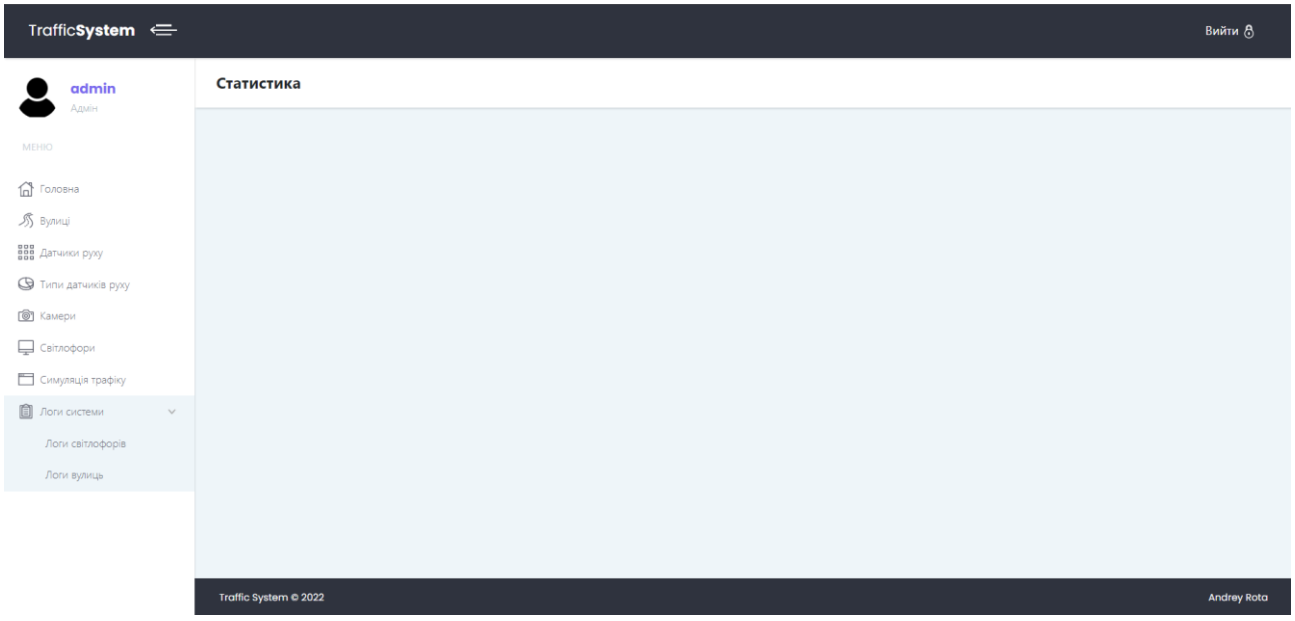


Рисунок 2.23 – Головна сторінка без наповнення даними

Далі до головної сторінки були додані панелі з статистичними даними (рис. 2.24) про кількість персоналу, поточні кольори роботи світлофорів, кількість датчиків, світлофорів та камер, а також графіки з середньою швидкістю машин

за сьогодні по годинам та середньою швидкістю по місяцям. Далі був доданим розділ з відсотком навантаження на дорогах за місяць. Під цими інформаційними блоками розміщено блок зі списком вулиць і їх навантаженням на поточний момент (рис. 2.25), а також з середнім часом простою машин на червоному світлі за цей місяць (рис. 2.26).

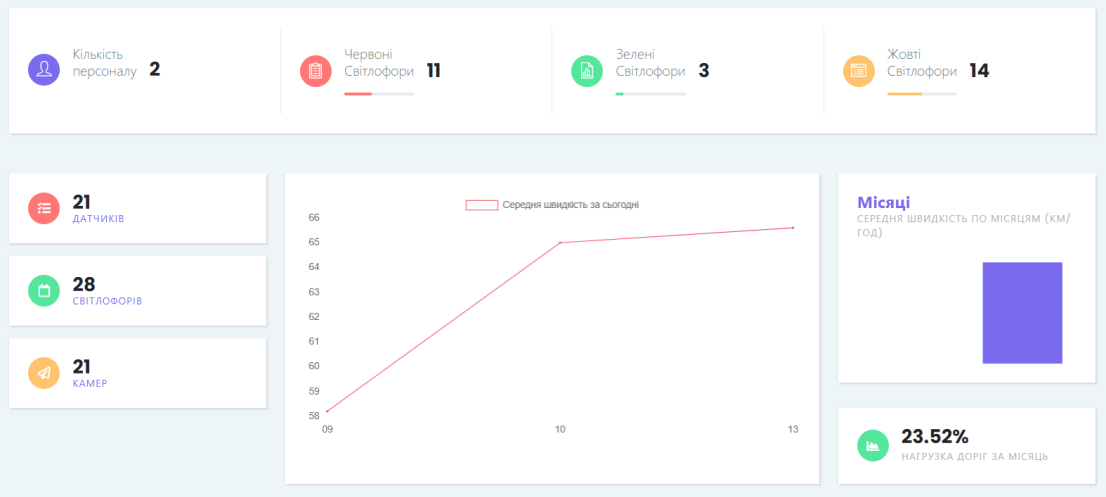


Рисунок 2.24 – Статистичні дані на головній сторінці

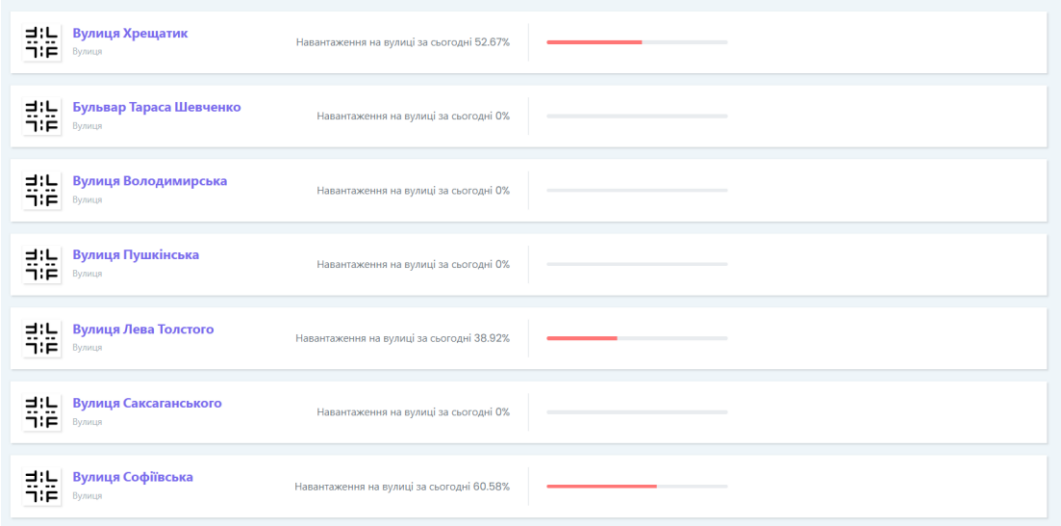


Рисунок 2.25 – Список вулиць і їх завантаження на головній сторінці

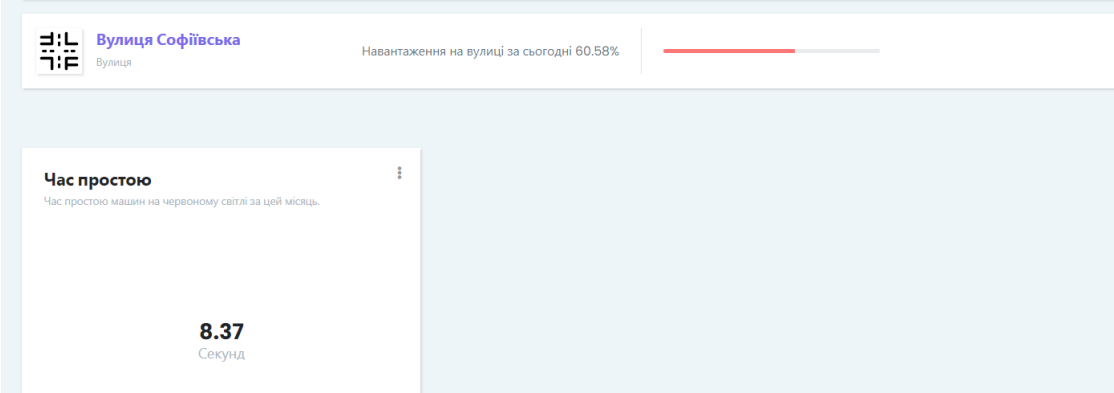


Рисунок 2.26 – Блок часу простою на головній сторінці

Наступною була створена сторінка з управління адміністраторами CRM-системи (admin.php), де будуть додаватися та видалятися диспетчери автоматизованої системи управління дорожнім рухом (рис. 2.27).

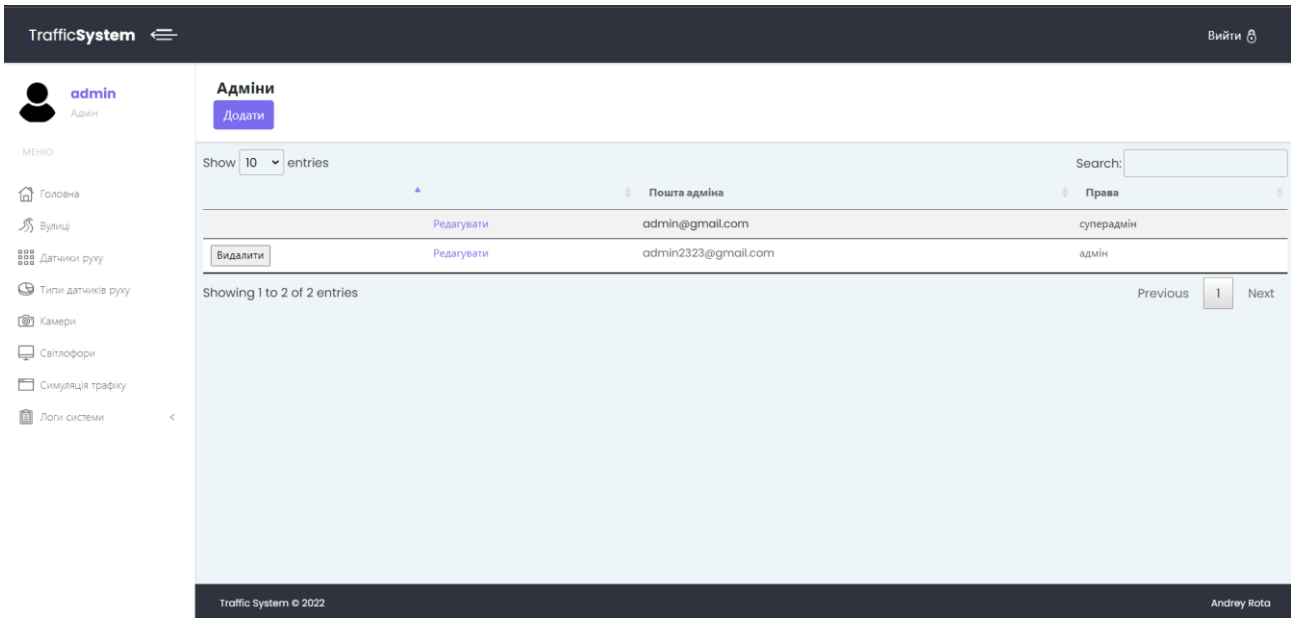


Рисунок 2.27 – Сторінка для керування адміністраторами системи

Аналогічно були створені наступні сторінки:

Вулиці на рис. 2.28.

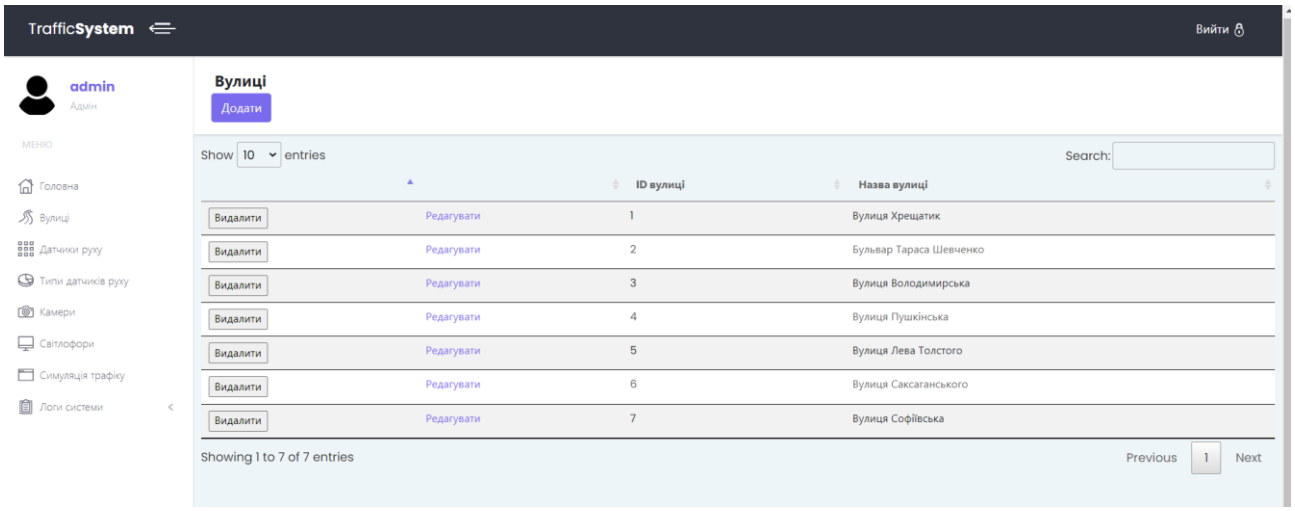


Рисунок 2.28 – Сторінка керування вулицями

Датчики руху на рис. 2.29.

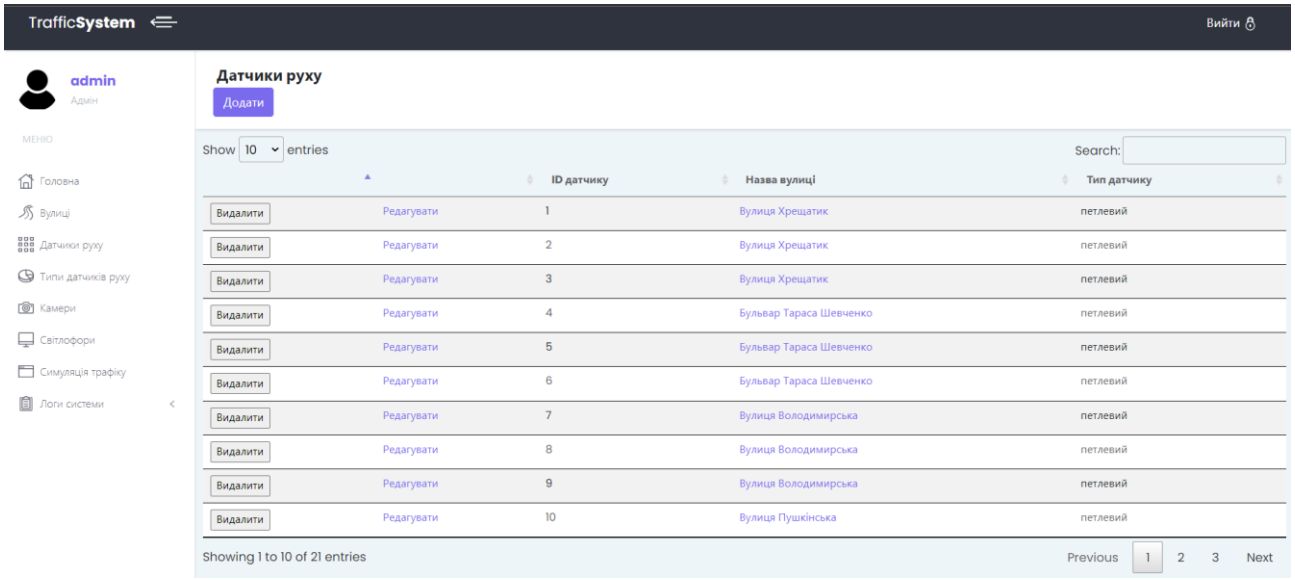


Рисунок 2.29 – Сторінка керування датчиками

Типи датчиків руху на рис. 2.30.

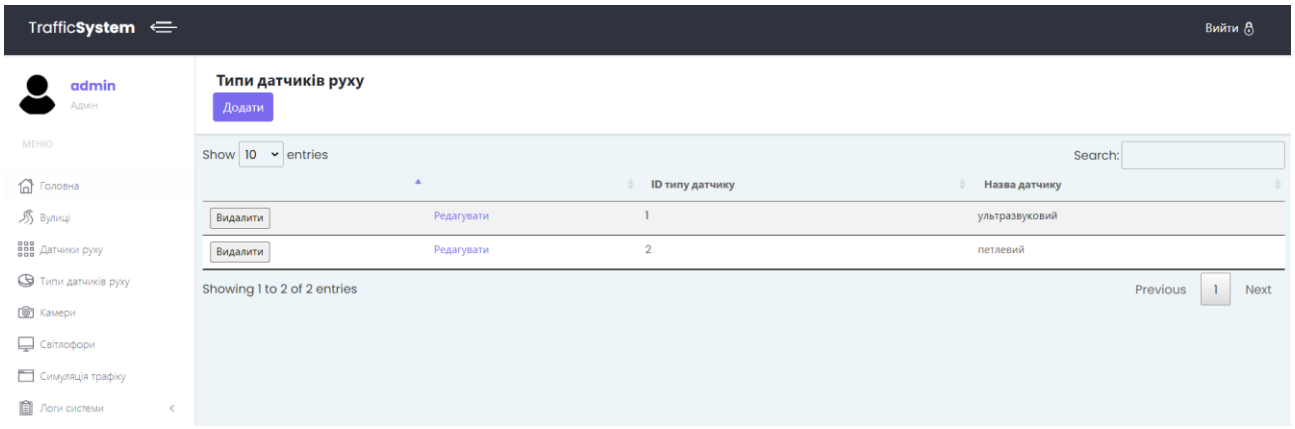


Рисунок 2.30 – Сторінка керування типами датчиків

Камери на рис. 2.31.

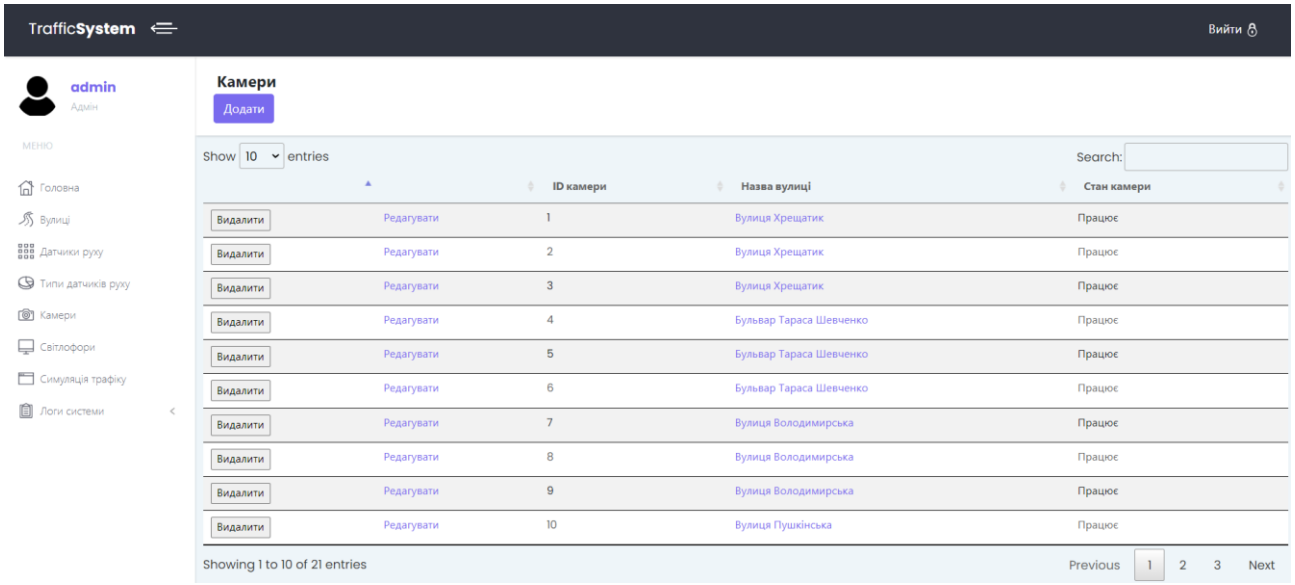


Рисунок 2.31 – Сторінка керування камерами

Світлофори на рис. 2.32.

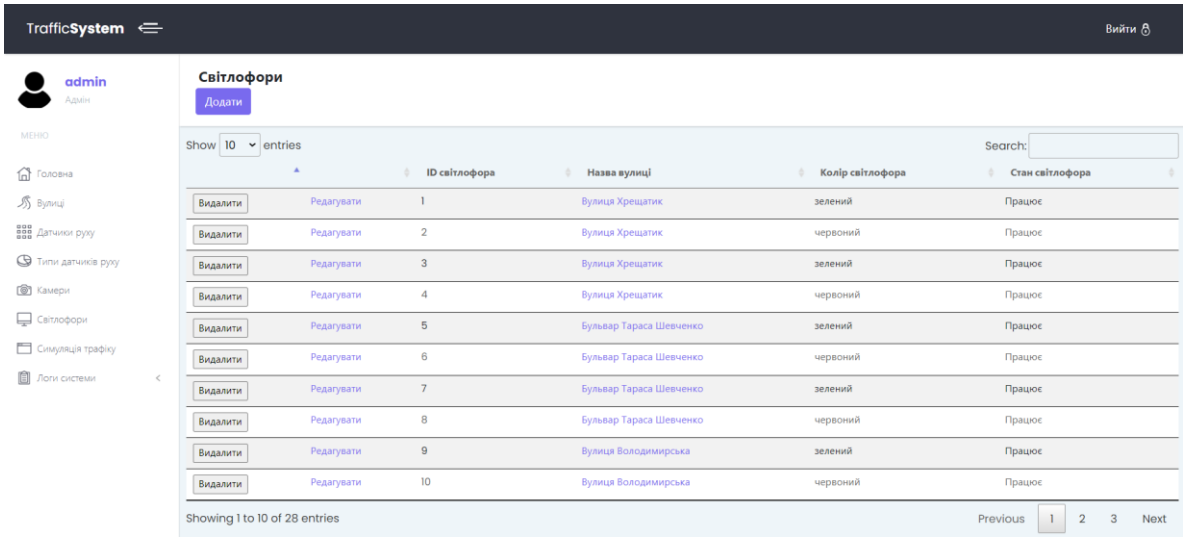


Рисунок 2.32 – Сторінка керування світлофорами

Логи світлофорів на рис. 2.33.

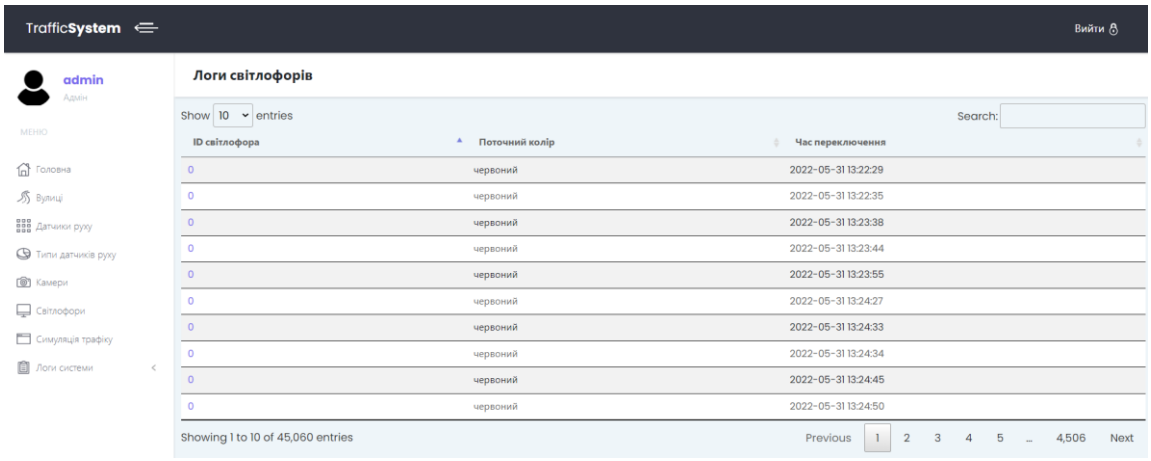


Рисунок 2.33 – Сторінка журналу роботи світлофорів

Логи вулиць на рис. 2.34.

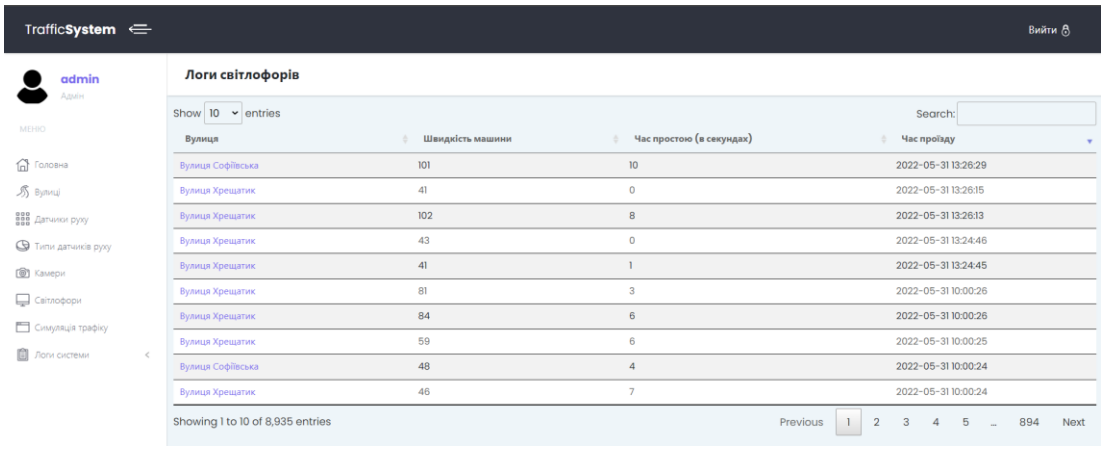


Рисунок 2.34 – Сторінка журналу пересування машин вулицями

Інструкція по користуванню CRM наведена у додатку Б.

Висновок до розділу 2:

- Спроектовано концептуальну модель бази даних на основі функціональної моделі для розуміння взаємозв’зків окремих частин системи.
- Розраховано вартість впровадження автоматизованої системи управління рухом на окрему частину дороги.
- Після визначення економічної обґрунтованості, сформовано проектну документацію з описом робіт та їх вартості.
- Наступним кроком спроектовано ділову модель підприємства для розуміння процесів окремих частин.

- На основі описаної вище ділової моделі, розроблено концептуальну схему роботи системи управління світлофорним регулюванням міста.
- На основі концептуальної схеми розроблено структуру таблиці бази даних, присвоєно типи даних та їх обмеження.
- Надано визначення стрілкам контекстної діаграми та створена сама контекстна діаграма.
- Далі спроектовано діаграму декомпозиції основних напрямів діяльності підприємства, що реалізує система управління світлофорним регулюванням міст.
- Розглянута також діаграма декомпозиції діяльності технічних департаментів, допоміжні процеси, а також діаграма дерева вузлів підприємства і модель самого підприємства.
- Для ілюстрації моделі процесу діяльності підприємства з точки зору процесів, їх взаємозв’язків і формування доходу, зроблено діаграму "тільки для експозиції".
- Далі розглянуто сценарій впровадження нової частини системи.
- Спроектовано логічну модель бази даних на основі створеної діаграми прецедентів. діаграми діяльності, діаграми послідовності і діаграми класів.
- Спроектовано фізичну модель бази даних, розроблено базу даних MySQL та спроектовано інтерфейс панелі адміністратора CRM автоматизованої системи керування дорожнім рухом за допомогою HTML5 та CSS3.

3. ПРОГРАМНА РЕАЛІЗАЦІЯ АВТОМАТИЗОВАНОЇ СИСТЕМИ УПРАВЛІННЯ ДОРОЖНІМ РУХОМ

3.1. Програмне забезпечення CRM автоматизованої системи управління дорожнім рухом

Для розробки системи автоматизованого керування дорожнім рухом обрано мову програмування PHP [28] та JavaScript [29] для написання бізнес-логіки самої системи. Даний «стек» мов програмування відповідає усім потребам до проектування веб-додатку системи.

Написання коду програми виконано на мові PHP за допомогою об’єктно-орієнтованої парадигми програмування без використання додаткових «бек-енд» фреймворків.

Для написання «фронт-енду» веб-додатку використано фреймфорк Bootstrap. Він надає бібліотеки для швидкого створення інтуїтивно зрозумілого інтерфейсу.

Для забезпечення швидкого пошуку та фільтрації даних на стороні клієнта використано безкоштовну JavaScript бібліотеку під назвою DataTables [30], для забезпечення роботи якої підключено бібліотеку JQuery, яка надає готовий набір програмних засобів для динамічної зміни інтерфейсу системи за потребою користувача.

Переваги використання даного стеку технологій полягають у наступному:

- популярність мов програмування PHP і JavaScript, що забезпечує актуальність системи та надає змогу швидше знайти спеціалістів для її підтримки;
- велика кількість готових рішень, які лежать у вільному доступі для кожного;
- популярність також забезпечує велику спільноту програмістів на цих мовах, які можуть допомогти у вирішенні проблем з написанням коду;

Переваги використання об’єктно-орієнтованої парадигми програмування:

- «природнє» представлення компонентів системи, що є більш

зрозумілим для розробника;

- масштабованість за рахунок того, що не треба переписувати код при введені нової частини системи, а можна використовувати готові шаблони проектування;
- збільшується повторне використання коду;
- можливість створення бібліотек класів для подальшого їх використання.

Переваги використання бібліотеки DataTables:

- гарне представлення даних у вигляді зручних таблиць;
- гнучкі можливості у фільтрації та сортуванні даних;
- можливості динамічного доповнення поточних таблиць;
- зменшення часу розробки коду;
- безкоштовність;
- постійна підтримка та доповнення функціоналу бібліотеки.

3.2. Опис програмної реалізації CRM-системи та бази даних

Для написання коду веб-додатку, як сказано вище, використано дві мови програмування – PHP та JavaScript. Для виконання обрано шаблон проектування MVC [31] – Model View Controller. Даний шаблон проектування використовується для того, щоб чітко розділити клієнтську частину, бізнес-логіку та обробку запитів в базу даних.

У цій роботі розроблено такі класи контролерів:

- **AdminController** – обробник запитів адміністраторської частини.
- **AuthController** – обробник запитів на авторизацію.
- **CamerasController** – оброник запитів камер.
- **MotionSensorsController** – обробник запитів датчиків.
- **StreetsController** – обробник запитів вулиць.
- **TrafficLightsController** - обробник запитів світлофорів.
- **StreetsLogsController** – обробник запитів на логи вулиць.
- **TrafficLightsLogsController** – обробник запитів на логи світлофорів.

У роботі розроблено такі класи моделей:

- **AdminController –** обробка запитів у таблицю адміністраторів бази даних.
- **CamerasModel –** обробка запитів по камерам до бази даних.
- **CamerasStatesModel –** обробка запитів по станам камер до бази даних.
- **MotionSensorsModel –** обробка запитів по датчикам руху до бази даних.
- **MotionSensorsTypesModel –** обробка запитів по типам датчиків руху до бази даних.
- **StreetsModel –** обробка запитів по вулицям до бази даних.
- **TrafficLightsModel –** обробка запитів по світлофорам до бази даних.
- **TrafficLightsModesModel –** обробка запитів по режимам роботи світлофорів до бази даних.
- **TrafficLightsStatesModel –** обробка запитів по станам роботи світлофорів до бази даних.
- **TrafficLightsLogsModel –** обробка запитів по логам світлофорів до бази даних.
- **StreetsLogsModel** – обробка запитів по логам вулиць до бази даних.

А також у роботі створено такі клієнтські частини: сторінка авторизації, головна сторінка, сторінка з інформацією про світлофори, про датчики, камери, вулиці, сторінка з управлінням персоналом, сторінка з управління типами датчиків, і дві сторінки з логами вулиць та світлофорів.

Повна ієрархія усіх файлів веб додатку представлена на рис. 3.1 та рис. 3.2. Написання програмного коду проведено у Visual Studio Code, редакторі коду для безпосереднього створення та редагування сучасних веб-застосунків.

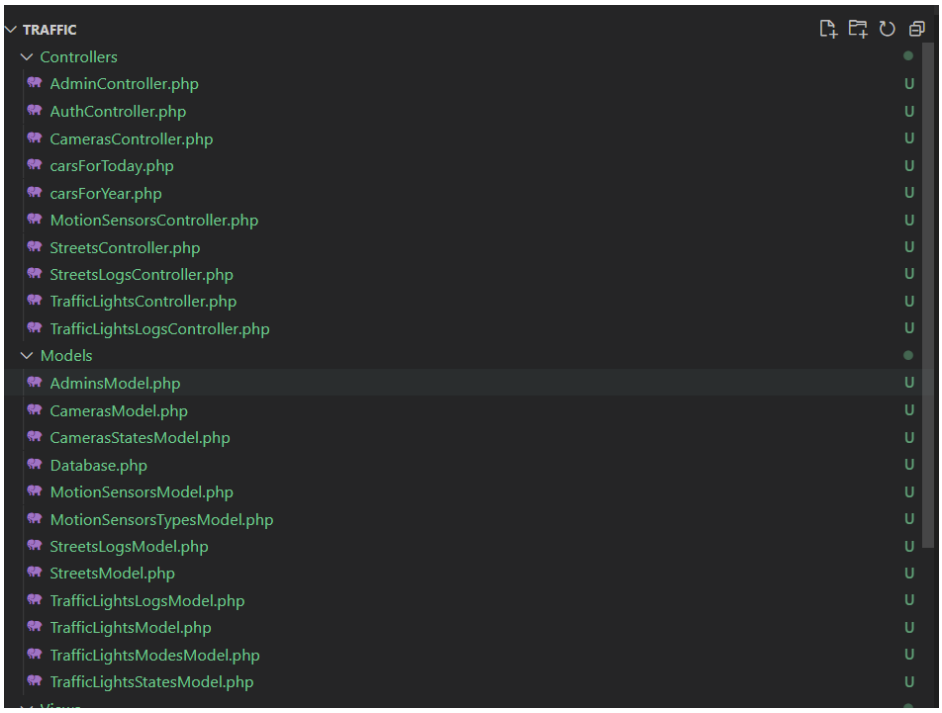


Рисунок 3.1 – ієрархія файлів веб-застосунку частина 1

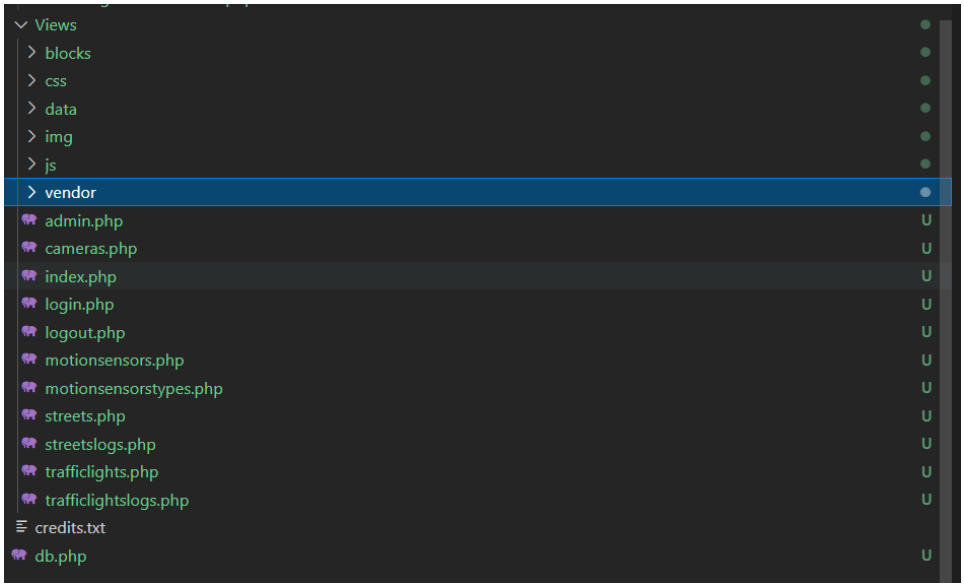


Рисунок 3.2 - ієрархія файлів веб-застосунку частина 2

Розглянуто приклад контролеру, який використовується у веб-додатку для авторизації адміністратора до адмін-панелі. Називається клас цього контролеру AuthController (рис. 3.3).

Рисунок 3.3 – Код класу AuthController, який виступає контролером авторизації користувачів до системи

В даному випадку, видно, що до класу AuthController підключається модель AdminsModel (рис. 3.4), яка забезпечує відправку от обробку запитів до бази даних. Контролер AuthController використовує метод login(), який звертається до однойменного методу з класу моделі AdminsModel і виконує чи не виконує запуск сесії користувача.

AuthController.php U | AdminsModel.php U

Models > AdminsModel.php

```php
<?php

require_once("Database.php");

class AdminsModel extends Database {

    public function __construct() {
        $this->pdo = parent::dbConnect("localhost", "root", "root", "trafficsystem");
    }

    public function login(string $login, string $password) {

        $password = $this->hashedPassword($password);

        $adminQuery = $this->pdo->query("SELECT * FROM `admins` WHERE `login`='".$login."' AND `password`='".$password."'");

        if($adminQuery->rowCount() > 0)
            return  $adminQuery->fetch(PDO::FETCH_ASSOC);
        else
            return false;
    }

    public function getAdmins() {
        $admins = $this->pdo->query("SELECT * FROM `admins` ORDER BY `id` DESC");

        return $admins;
    }

    public function getAdmin($adminId) {
        $admin = $this->pdo->query("SELECT * FROM `admins` WHERE `id`='".$adminId."'");

        return $admin->fetch(PDO::FETCH_ASSOC);
    }

```

Рисунок 3.4 – код класу моделі AdminsModel

У моделі AdminsModel ще є деякі додаткові методи, які використовувалися при написані програмної частини веб-додатку. Наприклад, метод getAdmins() видає список всіх адміністраторів і усі їх дані, а метод getAdmin() повертає одного конкретного адміністратора, ідентифікатор якого був переданим в якості аргументу методу. За допомогою цього методу працює авторизація на сторінці логіну (рис. 3.5).

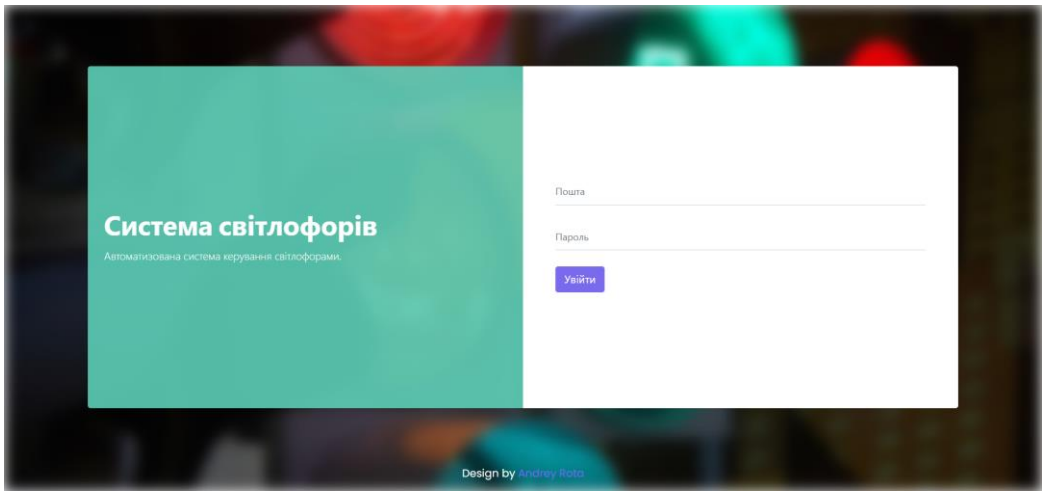


Рисунок 3.5 – Сторінка авторизації до веб-додатку системи керування дорожнім рухом

Вище показано приклад реалізації об’єктно-орієнтованої парадигми програмування разом із шаблоном проектування MVC (рис. 3.6). Дана зв’язка використовується і в інших частинах та модулях системи для забезпечення гнучкості розробки, масштабування, підвищенню читаності коду та обмеження доступу однієї частини програми до іншої. Дана практика наразі є найбільш поширеною серед розробників веб-застосунків, тому що добре зарекомендувала себе на великий дистанції серед тисяч розробників.

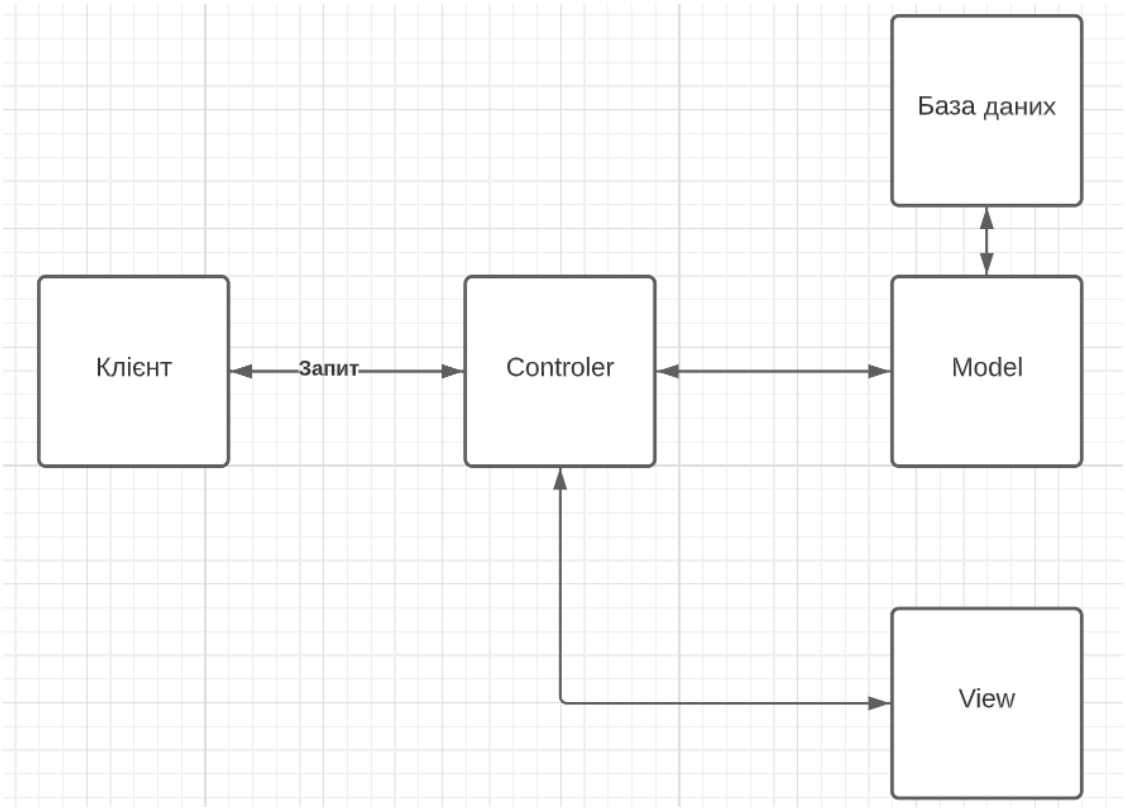


Рисунок 3.6 – принцип роботи веб-додатку за шаблоном проектування MVC

Для кращого розуміння роботи веб-додатку розглянуто схему взаємодії окремих частин: моделей, контролерів та уявлень. Схема представлена на рис. 3.6. Як можна побачити, користувач заходить до системи через певне уявлення – спочатку це сторінка авторизації, а потім головна сторінка системи, і після цього кожний запит користувача йде через певний контролер частини системи. Контролер в свою чергу оновлює дані в базі даних через відповідну йому модель, які користувач потім може побачити на певному уявлені, яке в свою чергу отримує інформацію безпосередньо з моделі.

3.3. Структурна схема зв’язку модулів системи.

Як видно зі структурної схеми (рис. 3.7), на вході системи є датчики, світлофори і камери, які об’єднані в одну систему, збирають дані, оброблюють і передають їх в базу даних. База даних приймає нові дані, записує інформацію про стан камер, світлофорів і датчиків, записує логи по кожній вулиці, записує усі перемикання кольорів світлофору, групує дані та передає їх за запитом користувача у автоматизовану систему керування дорожнім рухом.

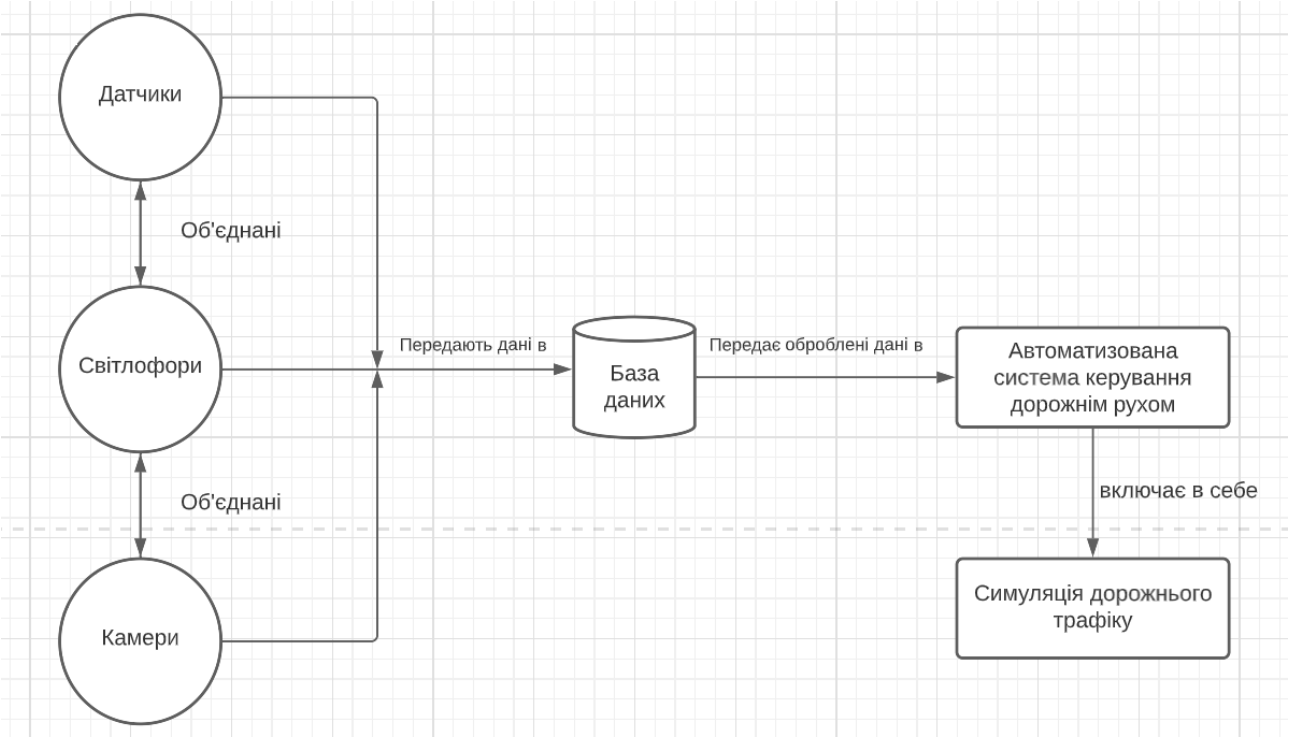


Рисунок 3.7 – структурна схема зв’язків модулів програми

Дані, які беруться з бази даних, далі перетворюються для користувача у зручні та інформативні звіти по котрим спеціаліст з дорожнього руху може зрозуміти оперативну обстановку на вулицях міста і прийняти рішення щодо зміни режиму роботи системи.

Симуляція є окремим модулем, який включається до автоматизованої системи керування дорожнім рухом. Даний модуль дає змогу провести симуляцію різних потоків машин та зрозуміти яке буде навантаження на кожну з вулиць, що дає можливість запобігти перенавантаження доріг та спланувати свої дії при екстрених ситуаціях.

3.4. Симуляція автоматизованої системи керування дорожнім рухом з точки зору безпеки

Для проектування симуляції автоматизованої системи управління дорожнім рухом обрано платформу для розробки ігор Unity 3D.

Симуляція автоматизованого управління дорожнім рухом реалізується в Unity за допомогою асетів самої вулиці та префабів машин, які рандомно генеруються на вулиці та слідують з рандомною швидкістю по випадковій траєкторії. Для того, щоб збирати дані про стан навантаження доріг у місті, треба перш за все розташувати датчики руху, камери і контролери на світлофори.

Наступним кроком є реалізація радіочастної передачі даних (рис. 3.8) між датчиками за допомогою контролерів. Передача даних має відбуватися у зашифрованому вигляді, мікроконтролери повинні мати у списках доступу тільки інші мікроконтролери, щоб третя особа не змогла до них підключитися.

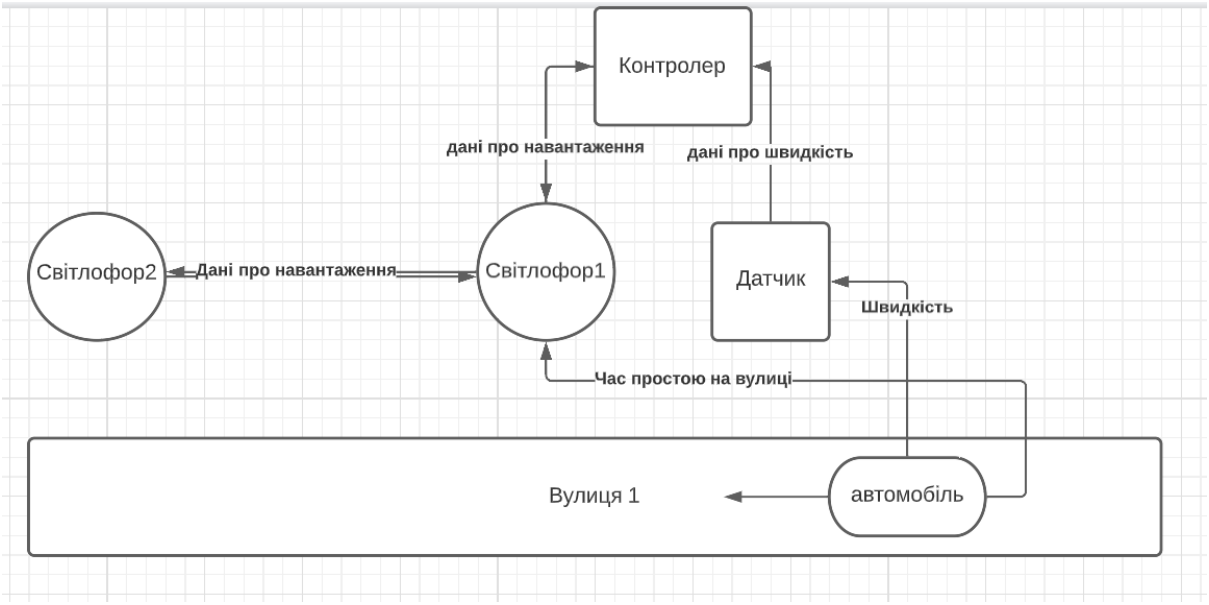


Рисунок 3.8 - Схема обміну інформацією на ділянці вулиці

Дані про швидкість транспортних засобів, які проїжджали по вулиці збираються датчиком за допомогою ультразвуку, який він посилає і за рахунок якого фіксує нові машини. Датчик передає дані до контролеру, який також збирає дані з світлофорів про час простою машин на вулиці. На основі отриманних данних контролер робить певні обчислення та змінює час роботи світлофорів. Усі відправлені запити повинні фільтруватися та оброблятися приймаючим

датчиком або контролером. Також датчики повинні використовувати шифрування даних.

3.5. Розробка Unity симуляції автоматизованої системи керування дорожнім рухом

Перед початком розробки був розглянутий робочий день диспетчера дорожнього руху для аналізу його потреб та виявлення ключових повсякденних завдань.

Робочий день у диспетчера дорожнього руху починається о 9 годині ранку та закінчується в 18 годин вечора. Працівник відповідає за моніторинг працездатності автоматизованої системи та за станом на дорогах. Через камери, диспетчер може виявляти проблемні участки дороги, або бачити дорожні пригоди, які тільки сталися, що дозволяє швидко на них реагувати.

Датчики та контролери збирають та оброляють інформацію, яка поступає в автоматизовану систему, в якій диспетчер може передивитися журнал подій по кожній вулиці, а також по кожному з світлофорів.

3.5.1. Розробка сцени, розміщення 3D-об'єктів

Першим кроком розробки сцени є знаходження безкоштовного набору об’єктів, так званих assets, який повністю задовольняв потребам побудови вулиць, перехресть та світлофорів.

Спершу на сцені було розміщено об’єкт Crossroads, тобто перехрестя (рис. 3.9).

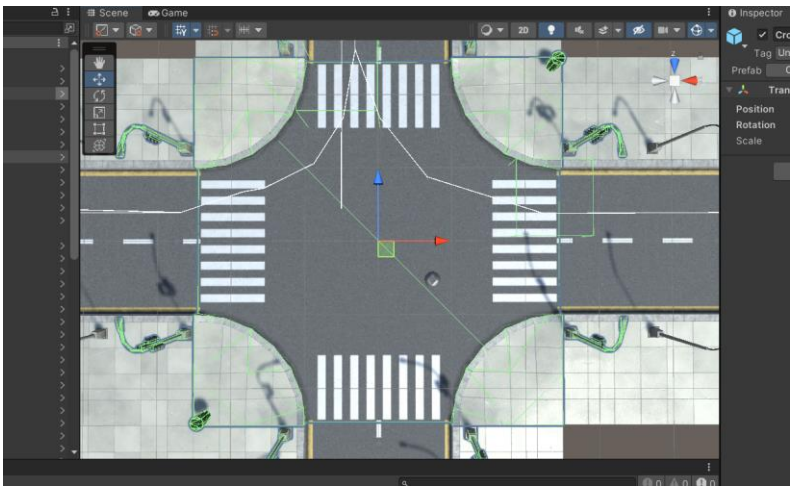


Рисунок 3.9 – Unity-об’єкт перехрестя

Далі до перехрестя додано об’єкти Road_line, тобто самі вулиці, по яким повинні переміщатися транспортні засоби (рис. 3.10).

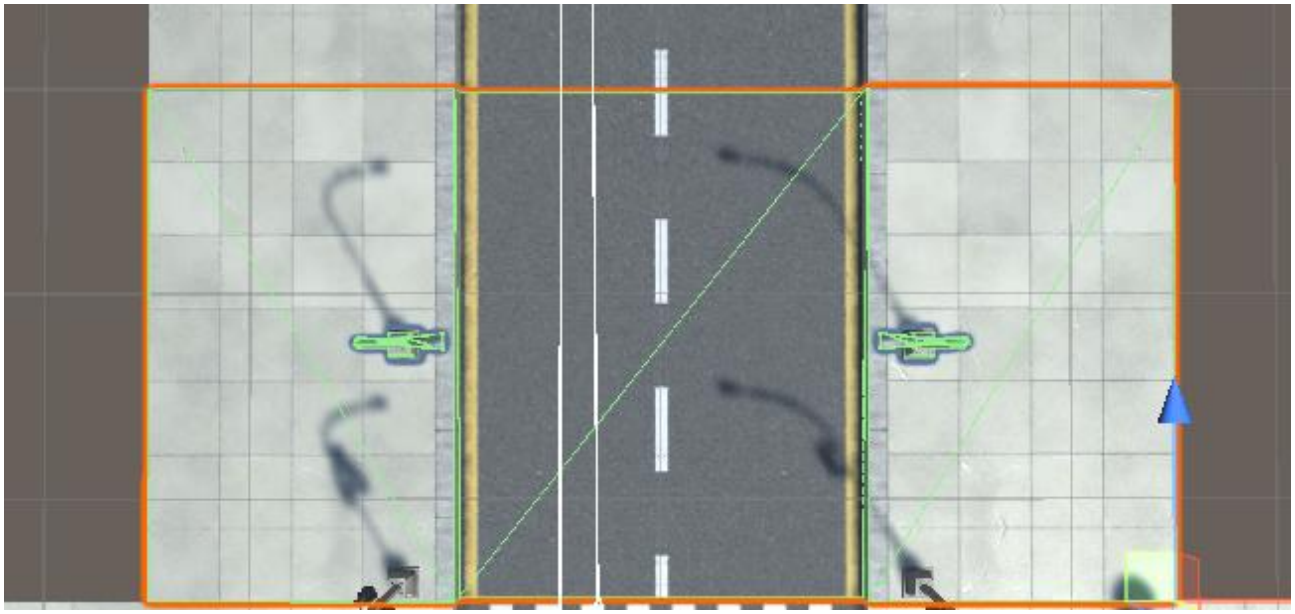


Рисунок 3.10 – Unity-об’єкт дорога

Наступним кроком на дорогах були розміщені вуличні ліхтарі, об’єкти під назвою Pole (рис. 3.11).

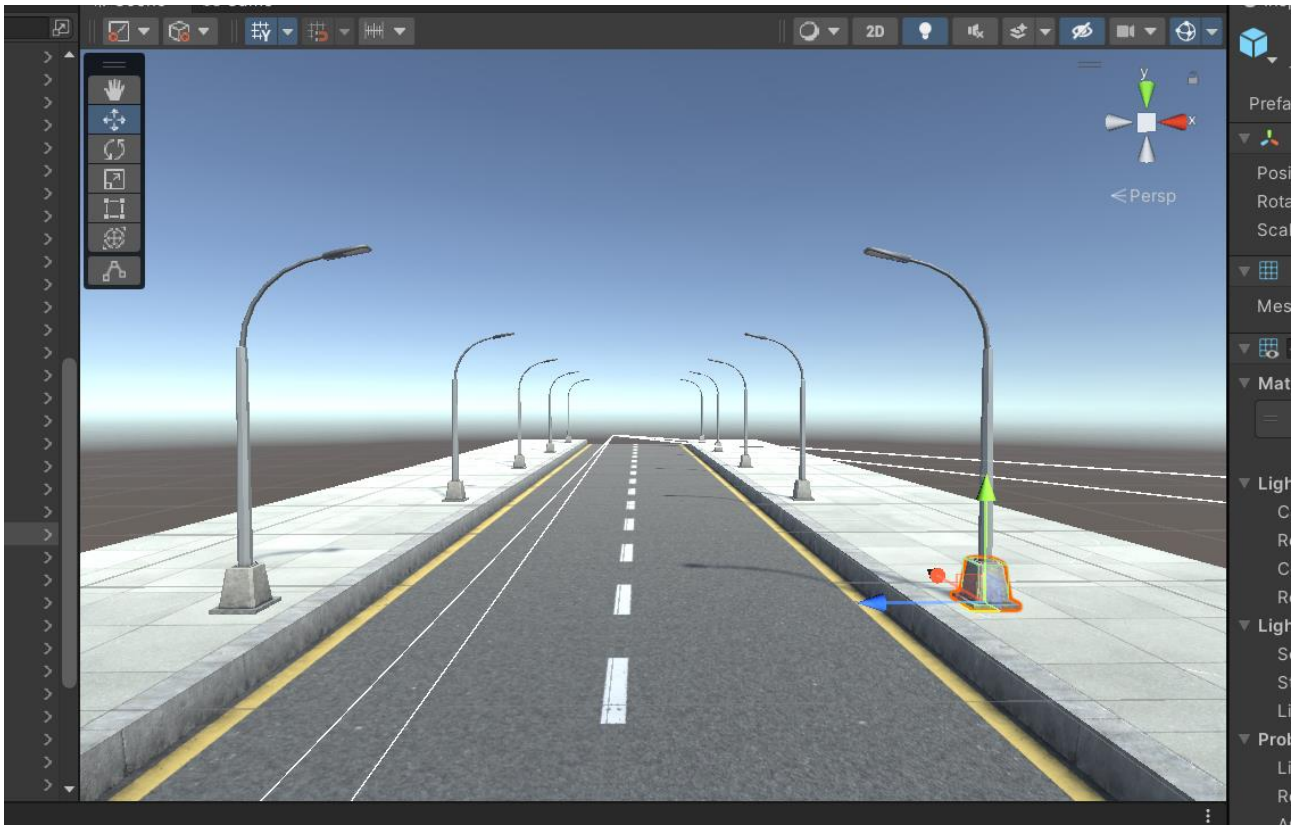


Рисунок 3.11 – Unity-об’єкти вуличні ліхтарі

На перехрестях були розміщені об’єкти Traffic_lights, тобто світлофори (рис. 3.12).

Рисунок 3.12 – Unity-об’єкти світлофори

Далі були розміщені об’єкти площі, каналізаційних люків, сміттєвих баків, дерев, столів, стільців та лавок (рис. 3.13).

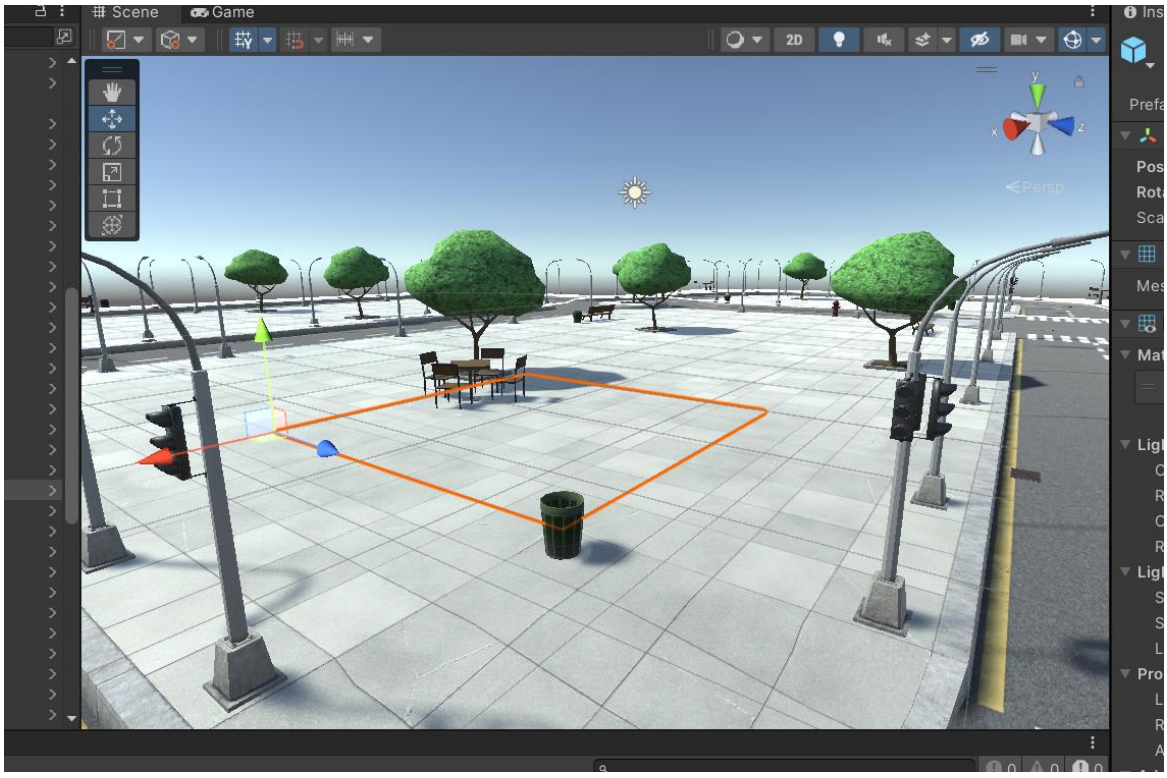


Рисунок 3.13 – Unity-об’єкти стільці, столи, лави, сміттєві баки, дерева

Наступним кроком обрано і налаштовано prefab (шаблон об’єкту) машини для генерації транспортних засобів на карті (рис. 3.14).

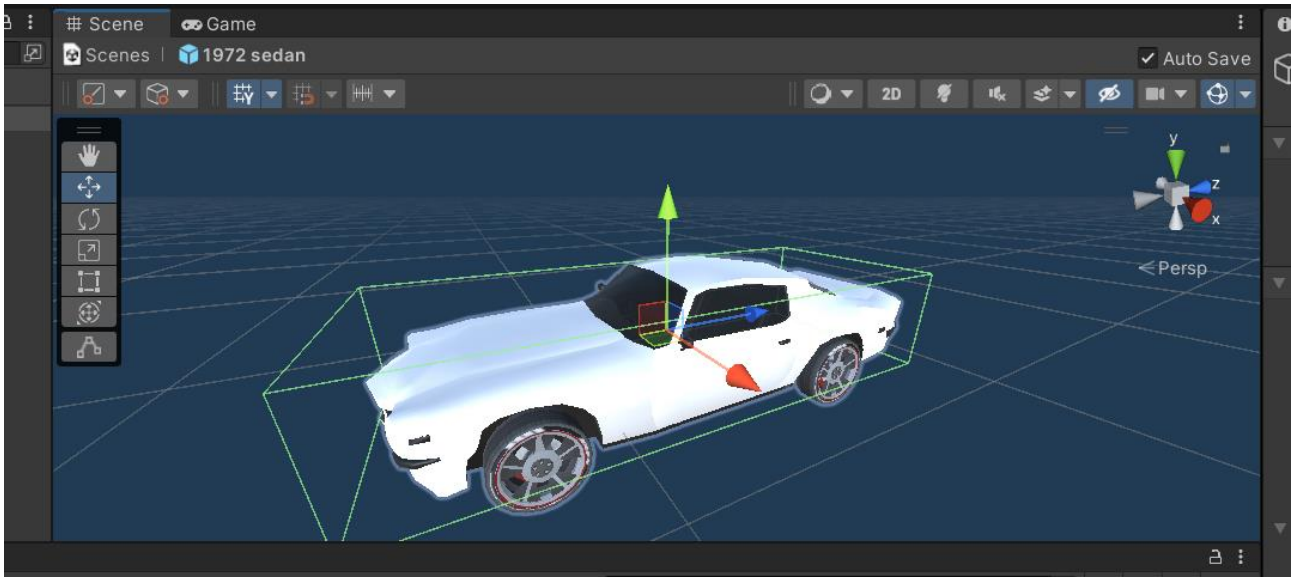


Рисунок 3.14 – Unity-об’єкт машина

Як результат отримано мапу з дорогами і перехрестями, на якій пересуваються машини (рис. 3.15).

Рисунок 3.15 – Пересування машин по симуляції

3.5.2. Розробка скриптів для роботи автоматизованої системи управління дорожнім рухом

Для забезпечення симуляції роботи автоматизованої системи я використовував такі класи:

CarEngine – цей клас відповідає за логіку пересування транспортних засобів, регулює швидкість, зупинку та продовження руху.

CarSpawner – це клас, який відповідає за логіку генерації нових транспортних засобів на карті. Він випадковим чином створює машину з єдиного шаблону (префабу), але присвоює їй унікальний колір, швидкість та траєкторію руху.

DeleteCar – це клас, який відповідає за знищення машини з карти, коли вона дійшла до кінцевої точки свого шляху.

DistanceCheker – це клас, який відповідає за перевірку відстані між машинами, щоб вони не проходили одна повзж іншу.

LightCheker – це клас колайдеру, який перевіряє колір світлофору та приймає рішення чи зупинити транспортний засіб, чи пропустити його. Також цей клас фіксує швидкість машини, час її простою на вулиці у заторі та відправляє ці дані у базу.

MoveMouse – цей клас відповідає за логіку повороту камери спостерігача для зручного просмотру симуляції.

Path – цей клас відповідає за створення нової траєкторії руху, яка потім присвоюється машинам.

TrafficLIghtSystem – цей клас відповідає за логіку роботи світлофорів, переключення кольорів, зміну часу роботи певного кольору і таке інше.

Опишемо логіку роботи скриптів.

З початку симуляції CarSpawner створює нові транспортні засоби, присвоює їм траєкторію, колір, швидкість і запускає на карту. Далі починає працювати клас CarEngine**,** який є у кожної машини. У ньому відбувається перевірка двох змінних canRide та canMove, які є бульовими, та відповідають за можливість пересування. Якщо canRide = false, то попереду є машина і треба зупинитися, а якщо canMove = false, то машина вже на світлофорі і його колір червоний. За значення цих змінних відповідають класи DistanceChecker та LightCheker відповідно. Крім цього, LightCheker фіксує значення швидкості та часу простою машини, коли вона проїжджає світлофор, а значить заїжджає або виїжджає з вулиці і ми можемо кінцеві дані додати у базу. Також LightCheker перевіряє навантаження на вулицях і на тих вулицях де машини стоять у заторах більше ніж 10 секунд (обраний такий час для пришвидення аналізу симуляції)

він змінює час роботи червоного і зеленого кольорів у класі TrafficLIghtSystem (рис. 3.16). Таким чином система адаптується до ситуації на вулицях. Транспортний засіб, закінчивши свій шлях, видаляється классом DeleteCar, розташованим на об'єкті EndCollider.

```
public class TrafficLIghtSystem : MonoBehaviour
{

    private bool colorCicle = false;
    public string currentColor;
    private IEnumerator trafficLightRoutine;

    public float startColorTime = 4f; // это время работы первого цвета
    public float endColorTime = 10f; // время работы второго цвета
    public float yellowTime = 1f;

    public string startStringColor = "green";
    public string endStringColor = "red";

    private Color startColor = Color.green;
    private Color endColor = Color.red;

    public string streetId = "";

    void Start()
    {


        if(gameObject.transform.parent.gameObject.name == "Traffic_light (2)") {
            startStringColor = "red";
            endStringColor = "green";

            startColor = Color.red;
            endColor = Color.green;

            startColorTime = 10f;
            endColorTime = 4f;
        }
```

Рисунок 3.16 - частина коду класу TrafficLIghtSystem

3.5.3. Інтеграція Unity симуляції із базою даних

Для забезпечення збереження даних про пересування транспортних засобів містом, даних про їх швидкість, час простою у заторах, а також даних про роботу світлофорів зроблено інтеграцію Unity симуляції з базою даних MySQL.

Оскільки розміщувати пряме підключення до бази даних у коді C# Unity не є безпечним через те, що проект можна досить легко декомпілювати і

передивитися увесь код, в тому числі логіни і паролі до бази даних, то частина роботи з базою була перенесена на серверну частину.

Відповідно, до коду CRM системи додано три файли addData.php, addTrafficLightsLog.php, getLoad.php.

Усі три файли приймають POST запроси, фільтрують і форматують вхідні дані для подальшого внесення в базу даних.

Файл appData.php (рис. 3.17) вносить дані про час простою машини у заторі (змінна waitingTime), швидкість машини (змінна carSpeed), ідентифікатор вулиці (змінна streetId) до таблиці streetslogs.

```php
<?php

require_once("Models/Database.php");

$database = new Database;

$pdo = $database->dbConnect("localhost", "root", "root", "trafficsystem");

$currentDate = date("Y-m-d H:i:s");

$waitingTime = floatval($_POST["waitingTime"]);
$carSpeed = floatval($_POST["carSpeed"]);
$streetId = intval($_POST["streetId"]);

$pdo->query("INSERT INTO `streetslogs`(`streetId`, `arriveTime`, `carSpeed`, `waitingTime`) VALUES('".$streetId."', '".$currentDate."', '".$carSpeed."', '".$waitingTime."
```

Рисунок 3.17 – Код файлу addData.php

У коді симуляції ці дані збираються і обробляються асинхронним методом appData класа LightCheker (рис. 3.18), який відповідає за логіку зупинки машин на світлофорах.

```
private IEnumerator addData(string streetId, string carSpeed, string waitingTime) {

    string url = "http://traffic/addData.php";

    WWWForm form = new WWWForm();

    form.AddField("streetId", streetId);
    form.AddField("carSpeed", carSpeed);
    form.AddField("waitingTime", waitingTime);

    WWW data = new WWW(url, form);

    yield return data;

    Debug.Log(data.text);

}
```

Рисунок 3.18 – Код методу симуляції addData

Файл addTrafficLightsLog.php (рис. 3.19) приймає змінні currentModeId (колір на який світлофор переключився) та trafficLightId (ідентифікатор

світлофору) та вносить ці дані у таблицю trafficlightslogs з вказанням часу переключення кольору.

Рисунок 3.19 – Код файлу addTrafficLightsLog.php

У коді симуляції ці дані збираються і обробляються асинхронним методом addTrafficLightsLog класа TrafficLIghtSystem, який відповідає за логіку роботи світлофорів.

Файл getLoad.php (рис. 3.20) дістає з таблиці streetslogs середній час простою машин у заторах по переданому йому ідентифікатору вулиці (streetId).

Рисунок 3.20 – Код файлу getLoad.php

На основі отриманого значення методом getTime (рис. 3.21) класу LightCheker Unity-симуляція робить висновок стосовно того, як змінити час роботи зеленого та червоного кольору світлофора для розвантаження заторів на дорогах.

```
private IEnumerator getTime(string streetId) {

    string url = "http://traffic/getLoad.php";

    WWWForm form = new WWWForm();

    form.AddField("streetId", streetId);

    WWW data = new WWW(url, form);

    yield return data;

    averageWaitingTime = data.text;

}
```

Рисунок 3.21 – Код методу симуляції getTime

3.5.4. Збірка Unity симуляції та її інтеграція в CRM систему

Для збірки Unity симуляції, був обраний режим WebGL [32] у розділі Bulid Settings (рис. 3.22). Цей режим забезпечує розгортання Unity-проекту у веб-браузері, завдяки чому його потім можна інтегрувати в CRM-систему.

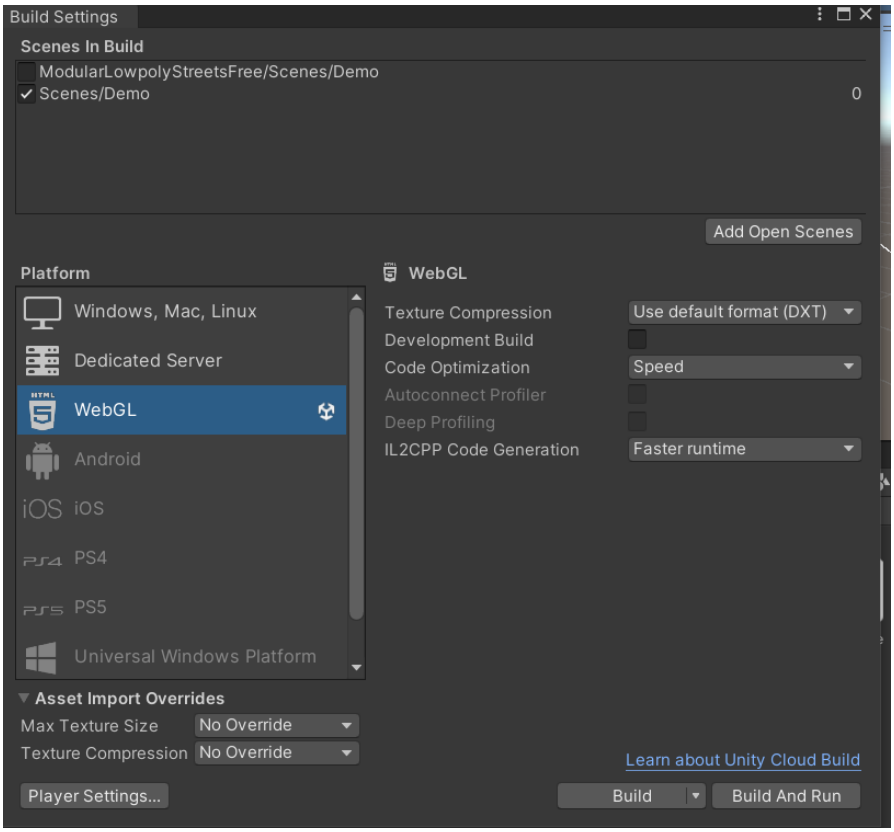


Рисунок 3.22 – збірка проекту Unity в режимі WebGL

Після збору симуляції, вона стала доступною на одному з вільних портів локального серверу. В даному випадку на порті 56838. Відповідно, до CRM

системи додано вкладку з симуляцією трафіку до файлу sidebar.php, який відповідає за ліву бокову панель (рис. 3.23).

```
<li class="sidebar-item"><a class="sidebar-link" href="http://localhost:56838/" target="_blank">
    <svg class="svg-icon svg-icon-sm svg-icon-heavy me-xl-2">
        <use xlink:href="#browser-window-1"> </use>
    </svg>Симуляція трафіку </a>
</li>
```

Рисунок 3.23 – код кнопки симуляції бокового меню

Висновок до розділу 3: розроблено програмну частину CRM-системи за допомогою мови програмування PHP. Забезпечено користувацьку взаємодію з базою даних за допомогою PHP, JavaScript та технології Ajax. Зроблено функціонал додавання, читання, редагування та видалення елементів системи. За допомогою Unity і мови програмування C# розроблено і заінтегровано в CRM-систему симуляцію автоматизованої системи управління дорожнім рухом. Розроблено для цього усі необхідні 3D моделі і забезпечено зв’язок симуляції з базою даних розробкою відповідного REST API на мові PHP.

ВИСНОВОК

У ході виконання роботи, розглянуто аналоги існуючих автоматизованих рішень, спроектовано і розроблено концептуальну, логічну і фізичну моделі бази даних автоматизованої системи управління дорожнім рухом. На основі побудованих моделей, розроблено базу даних MySQL за допомогою веб-додатку PhpMyAdmin.

Спроектовано і розроблено дизайн CRM-системи автоматизованого управління дорожнім рухом за допомогою поєднання мови гіпертексту HTML5 та мови стилю сторінок CSS3, а також фреймворку Bootstrap. Для забезпечення взаємодії користувача з базою даних за допомогою інтерфейсу використовувалися мови PHP та JavaScript, а також окремі бібліотеки JavaScript такі як JQuery та DataTables і технологія Ajax. Для забезпечення симуляції потоку дорожнього трафіку розроблено програму Unity на мові програмування C#. За допомогою створення REST API на мові PHP забезпечено взаємодію коду C# симуляції з базою даних MySQL.

Проведено тестування і поставлено експерименти, результатом яких стала адаптація автоматизованої системи до реальних умов на дорогах. Симуляція забезпечила адаптацію роботи світлофорів з ціллю розвантаження заторів. Усі журнали роботи світлофорів та проїжджаючих машин представлені у відповідних розділах CRM-системи. На головній сторінці системи представлена аналітика та звіти про поточну ситуацію на дорогах міста, де розташована IOT система керування дорожнім рухом.

Відповідно до мети дослідження створено автоматизовану систему управління дорожнім рухом з розробкою симуляції трафіку. Розробка дозволить отримати модель для моніторингу доріг у містах. Мета кваліфікаційної роботи бакалавра досягнута.

Перелік використаних інформаційних джерел

1. У Кличка порахували загальну кількість автомобілів у Києві [Електронний ресурс] - Режим доступу: https://www.epravda.com.ua/rus/news/2022/02/11/682305/ (дата звертання: 05.10.2022)
2. Сайт компанії «SEA» [Електронний ресурс] / SEA – Режим доступу: https://www.sea.com.ua/ua/svetofornaya-produkciya/programmnoe-obespecenie/
3. Сайт компанії «Росток-ЕЛЕКОМ» [Електронний ресурс] / ТОВ Росток Елеком – Режим доступу: http://rostok-elekom.com/
4. Сайт компанії «АТІЛОС» [Електронний ресурс] / АТІЛОС – Режим доступу: http://www.atilos.com.ua/uk/
5. Riccitiello, John (23 жовтня 2014) [Електронний ресурс] / Інтерв'ю з Dean Takahashi – Режим доступу: https://web.archive.org/web/20150117174953/http:/venturebeat.com/2014/10/23/john-riccitiello-sets-out-to-identify-the-engine-of-growth-for-unity-technologies-interview/
6. Джозеф Хокинг «Unity в дії. Мультиплатформенна розробка на C#», Питер, 2016. — 336 с.
7. Unity Manual Physics [Електронний ресурс] – Режим доступу: https://web.archive.org/web/20160824030644/http://docs.unity3d.com/540/Documentation/Manual/PhysicsSection.html
8. Ігри на Unity [Електронний ресурс] – Режим доступу: https://ru.wikipedia.org/wiki/%D0%9A%D0%B0%D1%82%D0%B5%D0%B3%D0%BE%D1%80%D0%B8%D1%8F:%D0%98%D0%B3%D1%80%D1%8B_%D0%BD%D0%B0_%D0%B4%D0%B2%D0%B8%D0%B6%D0%BA%D0%B5_Unity
9. Офіційний сайт Unreal Engine [Електронний ресурс] – Режим доступу: https://www.unrealengine.com/en-US
10. Офіційний сайт CryEngine [Електронний ресурс] – Режим доступу: https://www.cryengine.com/
11. Інформаційна сторінка Unreal Developer Kit (UDK) [Електронний ресурс] – Режим доступу: https://www.unrealengine.com/en-US/previous-versions

12. Офіційний сайт Kodu Game Lab [Електронний ресурс] – Режим доступу: https://kodugamelab.com/

13. Офіційний сайт NeoAxis Engine [Електронний ресурс] – Режим доступу: https://www.neoaxis.com/

14. Mylopoulos, J. «Conceptual modeling and Telos1». In Loucopoulos, P.; Zicari. «Conceptual Modeling, Databases, and Case An integrated view of information systems development», New York: Wiley. pp. 49-68.

15. Alt, R., and Zimmermann, H. (2001) «Introduction to special section – business models», Electronic Markets, 11, 1, 3-9.

16. IDEF0 діаграма: приклади і правила побудови [Електронний ресурс] / Codoschool – Режим доступу: https://codoschool.ru/uk/services/idef0-diagramma-primery-i-pravila-postroeniya-metodologii-modelirovaniya.html

17. Діаграми декомпозиції: приклади і правила побудови [Електронний ресурс] / Студопедія – Режим доступу: https://studopedia.com.ua/1_162873_diagrami-dekompozitsii.html

18. Діаграма дерева вузлів: приклади і правила побудови [Електронний ресурс] / Студопедія – Режим доступу: https://studopedia.com.ua/1_162876_diagrami-dereva-vuzliv-i-FEO.html

19. Fischer, R. A., Campbell, A. J., Shofner, G. A., Lord, O. T., Dera, P., & Prakapenka, V. B. (2011). [Електронний ресурс] / Equation of state and phase diagram of FeO. Earth and Planetary Science Letters, 304(3-4), 496-502. – Режим доступу: https://doi.org/10.1016/j.epsl.2011.02.025

20. Тіллманн, Джордж (червень 1995) [Електронний ресурс] / Building a Logical Data Model – Режим доступу: https://web.archive.org/web/20080509063521/http://www.dbmsmag.com/9506d16.html

21. James Rumbaugh, Ivar Jacobson, Grady Booch (1999) «The unified modeling language reference manual», University of Toronto.

22. Benedikt Bollig (2006) «Formal Models of Communicating Systems», «Message Sequence Charts» pp. 91-95.

23. Діаграма класів: приклад побудови [Електронний ресурс] / Державний університет телекомунікацій – Режим доступу: https://dut.edu.ua/ua/news-1-626-8002-zastosuvannya-uml-chastina-3-diagrama-klasiv----class-diagram_kafedra-kompyuternih-nauk-ta-informaciynih-tehnologiy

24. Енциклопедія кібернетики : у 2 т. / за ред. В. М. Глушкова. — Київ : Гол. ред. Української радянської енциклопедії, 1973.

25. Офіційна сторінка PhpMyAdmin [Електронний ресурс] / Bringing MySQL to the web – Режим доступу: https://www.phpmyadmin.net/

26. Джон Дакетт «HTML и CSS. Разработка и дизайн веб-сайтов» (2011).

27. David Sawyer McFarland «CSS3: The Missing Manual, 3rd Edition» (2012).

28. Майк МакГрат «PHP7 для початківців с покроковими інструкціями» (2018).

29. Девід Фленаган «JavaScript: The Definitive Guide, 5th Edition» (2008)

30. Офіційна сторінка DataTables [Електронний ресурс] / DataTables – Режим доступу: https://datatables.net/

31. Model-View-Controller [Електронний ресурс] / Вікіпедія – Режим доступу: https://ru.wikipedia.org/wiki/Model-View-Controller – Загол. з екрану

32. Сайт WebGL [Електронний ресурс] / WebGL Overview - The Khronos Group Inc – Режим доступу: https://www.khronos.org/webgl/

ДОДАТОК А

Код ключових елементів системи

Листів 9

Розробник Рота А.В.

Керівник Кравченко О.В.

Київ - 2022

Контролер адміністраторів системи

```
class AdminController {
   public function __construct() {
      $this->admin = new AdminsModel();
   }
   public function getAdminsCount() {
      return $this->admin->getAdmins()->rowCount();
   }
   public function getAdminPostname(string $email) {
      $split_post = explode("@", $email);
      return $split_post[0];
   }
   public function getAdmins() {
      $admins = $this->admin->getAdmins();
      return $admins->fetchAll();
   }
   public function getAdmin($adminId) {
      $admin = $this->admin->getAdmin($adminId);
      return $admin;
   }
   public function updateAdmin($adminId, $email) {
      $this->admin->updateAdmin($adminId, $email);
   }
   public function setNewPassword($adminId, $password) {
      $this->admin->setNewPassword($adminId, $password);
   }
   public function addAdmin($login, $password) {
      $this->admin->addAdmin($login, $password);
   }
   public function deleteAdmin($adminId) {
      $this->admin->deleteAdmin($adminId);
   }
   public function printAdminsTable() {
     $admins = $this->getAdmins();
      echo '
      <table id="datatable1" class="table table-striped table-hover">
         <thead>
         <tr>
            <th></th>
            <th></th>
            <th>Пошта адміна</th>
            <th>Права</th>
         </tr>
         </thead>
         <tbody>';
         foreach($admins as $admin) {
            if($this->admin->isSuperAdmin($admin["id"]))
               $adminType = "суперадмін";
            else
               $adminType = "адмін";
            echo '
            <tr>';
               if(!$this->admin->isSuperAdmin($admin["id"]))
                  echo '<td><input type="submit" name="delete" value="Видалити" id="'.$admin["id"].'"
class="deleteAdmin"></td>';
               else
                  echo '<td></td>';
               echo '<td><a href="admin.php?edit='.$admin["id"].'">Редагувати</a></td>
               <td>'.$admin["login"].'</td>
               <td>'.$adminType.'</td>
            </tr>
```

```
            ';
        }
        echo'</tbody>
    </table>
    ';
  }
}
```

Модель адміністраторів системи

```
class AdminsModel extends Database {

  public function __construct() {
    $this->pdo = parent::dbConnect("localhost", "root", "root", "trafficsystem");
  }
  public function login(string $login, string $password) {
    $password = $this->hashedPassword($password);
    $adminQuery = $this->pdo->query("SELECT * FROM `admins` WHERE `login`='".$login."' AND `password`='".$password."'");
    if($adminQuery->rowCount() > 0)
      return  $adminQuery->fetch(PDO::FETCH_ASSOC);
    else
      return false;
  }
  public function getAdmins() {
    $admins = $this->pdo->query("SELECT * FROM `admins` ORDER BY `id` DESC");
    return $admins;
  }
  public function getAdmin($adminId) {
    $admin = $this->pdo->query("SELECT * FROM `admins` WHERE `id`='".$adminId."'");
    return $admin->fetch(PDO::FETCH_ASSOC);
  }

  public function isSuperAdmin($adminId) {
    $superAdminQuery = $this->pdo->query("SELECT `id` FROM `admins` WHERE `id`='".$adminId."' AND `superUser`='1'");
    return $superAdminQuery->rowCount();
  }

  public function hashedPassword(string $password) {
    $password = md5("&B8t*b#Ta2N872S&zx".$password."&B8t*b#Ta2N872S&zx");
    return $password;
  }
  public function updateAdmin($adminId, $email) {
    $this->pdo->query("UPDATE `admins` SET `login`='".$email."' WHERE `id`='".$adminId."'");
  }
  public function setNewPassword($adminId, $password) {
    $password = $this->hashedPassword($password);
    $this->pdo->query("UPDATE `admins` SET `password`='".$password."' WHERE `id`='".$adminId."'");
  }
  public function addAdmin($login, $password) {
    $password = $this->hashedPassword($password);
    $this->pdo->query("INSERT INTO `admins`(`login`, `password`) VALUES('{$login}', '{$password}')");
  }
  public function deleteAdmin($adminId) {

    if(!$this->isSuperAdmin($adminId)) {
      $this->pdo->query("DELETE FROM `admins` WHERE `id`='".$adminId."'");
    }

  }

}
```

View адміністраторів системи

```
<?php
  session_start();
  ob_start();
  require_once("../Controllers/AdminController.php");
  $admin = new AdminController();
  if(!$_SESSION["is_super_admin"])
    header("Location: index.php");
?>
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <meta http-equiv="X-UA-Compatible" content="IE=edge">
    <title>Traffic System Адміни</title>
    <meta name="description" content="">
    <meta name="viewport" content="width=device-width, initial-scale=1, shrink-to-fit=no">
    <meta name="robots" content="all,follow">
    <link rel="stylesheet" href="https://fonts.googleapis.com/css?family=Poppins:300,400,700">
    <link rel="stylesheet" href="vendor/choices.js/public/assets/styles/choices.min.css">
    <link rel="stylesheet" href="css/style.default.css" id="theme-stylesheet">
    <link rel="stylesheet" href="css/custom.css">
    <link rel="shortcut icon" href="img/favicon.ico">
    <script src="https://ajax.googleapis.com/ajax/libs/jquery/2.2.0/jquery.min.js"></script>
    <script src="https://cdnjs.cloudflare.com/ajax/libs/chosen/1.5.1/chosen.jquery.min.js"></script>
<link rel="stylesheet" href="https://cdnjs.cloudflare.com/ajax/libs/chosen/1.5.1/chosen.min.css">
    <link rel="stylesheet" type="text/css" href="https://cdn.datatables.net/1.10.22/css/jquery.dataTables.css">
<script type="text/javascript" charset="utf8" src="https://cdn.datatables.net/1.10.22/js/jquery.dataTables.js"></script>
   <script type="text/javascript">
                        $(document).ready( function () {
                                $('#datatable1').DataTable();
                        } );
</script>
  </head>
  <body>
    <div class="page">
      <?php require_once("blocks/header.php"); ?>
      <div class="page-content d-flex align-items-stretch">
        <?php require_once("blocks/sidebar.php"); ?>
        <div class="content-inner w-100">
          <header class="bg-white shadow-sm px-4 py-3 z-index-20">
            <div class="container-fluid px-0">
              <h2 class="mb-0 p-1">Адміни</h2>
            </div><a href="admin.php?add=newAdmin"><button class="btn btn-primary">Додати</button></a>
          </header>
          <section class="pb-0" style="padding: 10px">
            <?php
            if(count($_GET) == 0) {
              $admin->printAdminsTable();
            }
            else if($_GET["sec"] == "asdasdaasdasd324124eredwsffdsf234rfdss" && !empty(trim($_GET["delete"]))) {
              $admin->deleteAdmin($_GET["delete"]);
              header("Location: admin.php");
            }
            else if(isset($_GET["add"]) && $_GET["add"] == "newAdmin") {
              ?>
               <div class="bg-white">
                <div class="container-fluid">
                  <nav aria-label="breadcrumb">
                    <ol class="breadcrumb mb-0 py-3">
                      <li class="breadcrumb-item"><a class="fw-light" href="admin.php">Адмін</a></li>
```

```
                <li class="breadcrumb-item active fw-light" aria-current="page">Новий адмін</li>
              </ol>
            </nav>
          </div>
        </div>
        <div class="card">
            <div class="card-header">
              <h3 class="h4 mb-0">Додавання адміна</h3>
            </div>
            <div class="card-body">
              <p class="text-sm">Введіть нові значення.</p>
              <form method="post">
              <div class="mb-3">
                 <label class="form-label" for="exampleInputEmail1">Пошта</label>
                 <input type="email" name="login" id="exampleInputEmail1" required>
                 <label class="form-label" for="password">Пароль</label>
                 <input type="password" name="password" id="password" required>
               </div>
               <input type="submit" name="add" value="Додати" class="btn btn-primary">
              </form>
            </div>
          </div>
        <?php
        if(isset($_POST["login"]) && isset($_POST["password"])) {
           $admin->addAdmin($_POST["login"], $_POST["password"]);
           header("Location: admin.php");
        }
       }
       else if(isset($_GET["edit"])) {
         $currentAdmin = $admin->getAdmin($_GET["edit"]);
        ?>
        <div class="bg-white">
          <div class="container-fluid">
            <nav aria-label="breadcrumb">
              <ol class="breadcrumb mb-0 py-3">
                <li class="breadcrumb-item"><a class="fw-light" href="admin.php">Адмін</a></li>
                <li class="breadcrumb-item active fw-light" aria-current="page">Редагування адміну
<?=$currentAdmin["login"]?></li>
              </ol>
            </nav>
          </div>
        </div>
        <div class="card">
            <div class="card-header">
              <h3 class="h4 mb-0">Редагувати адміна №<?=$currentAdmin["id"]?></h3>
            </div>
            <div class="card-body">
              <p class="text-sm">Введіть нові значення.</p>
              <form method="post">
                <div class="mb-3">
                   <label class="form-label" for="exampleInputEmail1" required>Пошта</label>
                   <input type="email" name="login" id="exampleInputEmail1" value="<?=$currentAdmin["login"]?>">
                   <label class="form-label" for="password">Новий пароль</label>
                   <input type="password" name="password" value="******" required>
                </div>
                <input type="submit" name="edit" value="Редагувати" class="btn btn-primary">
              </form>
            </div>
          </div>
        <?php
        if(isset($_POST["login"]) && isset($_POST["password"])) {
            $admin->updateAdmin($_GET["edit"], $_POST["login"]);
```

```
            if($_POST["password"] != "******" && !empty(trim($_POST["password"]))) {
               $admin->setNewPassword($_GET["edit"], $_POST["password"]);
            }
            header("Location: admin.php?edit=".$_GET["edit"]);

          }
         }
         ?>
        </section>
        <?php require_once("blocks/footer.php"); ?>
       </div>
     </div>
   </div>
   <script src="js/DataTables/jquery.dataTables.min.js"></script>
                    <script src="js/DataTables/extensions/ColVis/js/dataTables.colVis.min.js"></script>
                    <script src="js/DataTables/extensions/TableTools/js/dataTables.tableTools.min.js"></script>
   <script src="vendor/bootstrap/js/bootstrap.bundle.min.js"></script>
   <script src="vendor/just-validate/js/just-validate.min.js"></script>
   <script src="vendor/choices.js/public/assets/scripts/choices.min.js"></script>
   <script src="js/front.js"></script>
   <script>
      $(".select").chosen();
      $(".deleteAdmin").click(function() {
        if(confirm("Ви дійсно хочете видалити цього адміна?")) {
          var id = $(this).attr("id");
          window.location.href = "admin.php?delete=" + id + "&sec=asdasdaasdasd324124eredwsffdsf234rfdss";
        }
       });
     function injectSvgSprite(path) {
        var ajax = new XMLHttpRequest();
        ajax.open("GET", path, true);
        ajax.send();
        ajax.onload = function(e) {
          var div = document.createElement("div");
          div.className = 'd-none';
          div.innerHTML = ajax.responseText;
          document.body.insertBefore(div, document.body.childNodes[0]);
        }
     }
     injectSvgSprite('https://bootstraptemple.com/files/icons/orion-svg-sprite.svg');
   </script>
   <link rel="stylesheet" href="https://use.fontawesome.com/releases/v5.7.1/css/all.css" integrity="sha384-
fnmOCqbTlWIlj8LyTjo7mOUStjsKC4pOpQbqyi7RrhN7udi9RwhKkMHpvLbHG9Sr" crossorigin="anonymous">
  </body>
</html>
```

Код роботи світлофорів

```
using System.Collections;
using System.Threading;
using System.Collections.Generic;
using UnityEngine;
public class TrafficLIghtSystem : MonoBehaviour
{
   private bool colorCicle = false;
   public string currentColor;
   private IEnumerator trafficLightRoutine;
   public float startColorTime = 4f;
   public float endColorTime = 10f;
   public float yellowTime = 1f;
   public string startStringColor = "green";
   public string endStringColor = "red";
   private Color startColor = Color.green;
```

```
private Color endColor = Color.red;
public string streetId = "";
private IEnumerator addTrafficLightsLog(string currentModeId, string trafficLightId) {
    string url = "http://traffic/addTrafficLightsLog.php";
    WWWForm form = new WWWForm();
    form.AddField("currentModeId", currentModeId);
    form.AddField("trafficLightId", trafficLightId);
    WWW data = new WWW(url, form);
    yield return data;
    Debug.Log(data.text);
}
void Start()
{
    if(gameObject.transform.parent.gameObject.name == "Traffic_light (2)") {
        startStringColor = "red";
        endStringColor = "green";
        startColor = Color.red;
        endColor = Color.green;
        startColorTime = 10f;
        endColorTime = 4f;
    }
    trafficLightRoutine = trafficLight();
    StartCoroutine(trafficLightRoutine);
}
private void setBlack() {
    transform.Find("Traffic_light_EU_red").GetComponent<Renderer>().material.color = Color.black;
    transform.Find("Traffic_light_EU_green").GetComponent<Renderer>().material.color = Color.black;
    transform.Find("Traffic_light_EU_yellow").GetComponent<Renderer>().material.color = Color.black;
}
private void setColor(string colorName, Color color) {
    transform.Find($"Traffic_light_EU_{colorName}").GetComponent<Renderer>().material.color = color;
}
private IEnumerator trafficLight() {
    colorCicle = false;
    yield return addTrafficLightsLog(startStringColor, gameObject.tag);
    setBlack();
    setColor(startStringColor, startColor);
    currentColor = startStringColor;
    yield return new WaitForSeconds(startColorTime);
    yield return addTrafficLightsLog("yellow", gameObject.tag);
    setBlack();
    setColor("yellow", Color.yellow);
    currentColor = "yellow";
    yield return new WaitForSeconds(yellowTime);
    yield return addTrafficLightsLog(endStringColor, gameObject.tag);
    setBlack();
    setColor(endStringColor, endColor);
    currentColor = endStringColor;
    yield return new WaitForSeconds(endColorTime);
    yield return addTrafficLightsLog("yellow", gameObject.tag);
    setBlack();
    setColor("yellow", Color.yellow);
    currentColor = "yellow";
    yield return new WaitForSeconds(yellowTime);
    colorCicle = true;
}
private void Update() {
    if(colorCicle) {
        StopCoroutine(trafficLightRoutine);
        trafficLightRoutine = trafficLight();
        StartCoroutine(trafficLightRoutine);
    }
```

```
    }
}
```

Код роботи машини

```
using System.Collections;
using System.Collections.Generic;
using UnityEngine;
public class CarEngine : MonoBehaviour
{
    public Transform path;
    [SerializeField]
    private List<Transform> nodes;
    [SerializeField]
    private int currentNode;
    [SerializeField]
    private float speed;
    public bool canMove = true;
    public bool canRide = true;
    private int stepBeforeStop = 0;
    private float sumSpeed = 0;
    public float averageSpeed = 0;
    public int waitingTime = 0;
    private int rotationAngle = 0;
    void Start() {
        speed = Random.Range(1, 4);
        Transform[] pathTransforms = path.GetComponentsInChildren<Transform>();
        nodes = new List<Transform>();
        for(int i = 0; i < pathTransforms.Length; i++) {
            if(pathTransforms[i] != path.transform) {
                nodes.Add(pathTransforms[i]);
            }
        }
    }
    private void Drive() {
        speed = speed + (speed / 1000);
        transform.LookAt(nodes[currentNode]);
        transform.rotation = Quaternion.LookRotation(transform.position - nodes[currentNode].position);
        float step =  speed * Time.deltaTime; // calculate distance to move
        stepBeforeStop = stepBeforeStop + 1;
        sumSpeed = sumSpeed + speed;
        transform.position = Vector3.MoveTowards(transform.position, nodes[currentNode].position, step);
    }
    private void CheckWaypointDistance() {
        if(Vector3.Distance(transform.position, nodes[currentNode].position) < 0.5f) {
            if(currentNode == nodes.Count - 1) {
                currentNode = 0;
            }
            else {
                currentNode++;
            }
        }
    }
    // Update is called once per frame
    void FixedUpdate()
    {
        if(canMove && canRide) {
            averageSpeed = sumSpeed / stepBeforeStop;
            Drive();
            CheckWaypointDistance();
        }
        else {
            waitingTime = waitingTime + 1;
```

```
            speed = Random.Range(1, 4);
            averageSpeed = sumSpeed / stepBeforeStop;
        }
    }
}
```

Код зупинки автомобілей на світлофорах

```
using System.Collections;
using System.Collections.Generic;
using System.Threading;
using UnityEngine;
public class LightCheker : MonoBehaviour
{
    private bool timeChecked = false;
    private IEnumerator trafficLightRoutine;
    private IEnumerator trafficLightTime;
    public string averageWaitingTime = "0";
    private IEnumerator addData(string streetId, string carSpeed, string waitingTime) {
      string url = "http://traffic/addData.php";
       WWWForm form = new WWWForm();
       form.AddField("streetId", streetId);
       form.AddField("carSpeed", carSpeed);
       form.AddField("waitingTime", waitingTime);
       WWW data = new WWW(url, form);
       yield return data;
       Debug.Log(data.text);
    }
    private IEnumerator getTime(string streetId) {
       string url = "http://traffic/getLoad.php";
       WWWForm form = new WWWForm();
       form.AddField("streetId", streetId);
       WWW data = new WWW(url, form);
       yield return data;
       averageWaitingTime = data.text;
    }
    private void OnTriggerEnter(Collider other) {
       timeChecked = false;
       if(other) {
          if(GetComponentInParent<TrafficLIghtSystem>() != null) {
             var color = GetComponentInParent<TrafficLIghtSystem>().currentColor;
             trafficLightTime = getTime(gameObject.name);
             StartCoroutine(trafficLightTime);
             //Debug.Log("waiting streetId_" + gameObject.name + ": " +  averageWaitingTime);
             averageWaitingTime = averageWaitingTime.Replace(".",",");
             float averageFloatWaitingTime = float.Parse(averageWaitingTime);
             var trafficLightSystem = GetComponentInParent<TrafficLIghtSystem>();
             if(averageFloatWaitingTime > 9) {
                if(trafficLightSystem.startStringColor == "red") {
                   trafficLightSystem.startColorTime = trafficLightSystem.startColorTime - 1;
                   trafficLightSystem.endColorTime = trafficLightSystem.endColorTime + 1;
                   if(trafficLightSystem.endColorTime > 25)
                      trafficLightSystem.endColorTime = 25;
                   if(trafficLightSystem.startColorTime < 3)
                     trafficLightSystem.startColorTime = 3;
                }
                else {
                   trafficLightSystem.startColorTime = trafficLightSystem.startColorTime + 1;
                   trafficLightSystem.endColorTime = trafficLightSystem.endColorTime - 1;
                   if(trafficLightSystem.startColorTime > 25)
                      trafficLightSystem.startColorTime = 25;
                   if(trafficLightSystem.endColorTime < 3)
                     trafficLightSystem.endColorTime = 3;
```

```
                }
            }
            else {
                if(trafficLightSystem.startStringColor == "red") {
                    trafficLightSystem.startColorTime = trafficLightSystem.startColorTime + 1;
                    trafficLightSystem.endColorTime = trafficLightSystem.endColorTime - 1;
                    if(trafficLightSystem.startColorTime > 25)
                        trafficLightSystem.startColorTime = 25;
                    if(trafficLightSystem.endColorTime < 3)
                      trafficLightSystem.endColorTime = 3;
                }
                else {
                    trafficLightSystem.startColorTime = trafficLightSystem.startColorTime - 1;
                    trafficLightSystem.endColorTime = trafficLightSystem.endColorTime + 1;
                    if(trafficLightSystem.endColorTime > 10)
                        trafficLightSystem.endColorTime = 10;
                    if(trafficLightSystem.startColorTime < 4)
                      trafficLightSystem.startColorTime = 4;
                }
            }
            if(color == "red" || color == "yellow") {
                var engine = other.GetComponent<CarEngine>();
                if(engine != null)
                    engine.canMove = false;
            }
            else {
                if(!timeChecked) {
                    var engine = other.GetComponent<CarEngine>();
                    trafficLightRoutine = addData(gameObject.name, (engine.averageSpeed * 30).ToString("0.00"),
(engine.waitingTime / 60).ToString("0.00"));
                    StartCoroutine(trafficLightRoutine);
                    engine.waitingTime = 0;
                    timeChecked = true;
                }
            }
        }
    }
  }
  private void OnTriggerStay(Collider other) {
    if(GetComponentInParent<TrafficLIghtSystem>() != null) {
      var color = GetComponentInParent<TrafficLIghtSystem>().currentColor;
       if(color == "green") {
          var engine = other.GetComponent<CarEngine>();
          if(engine != null)
            engine.canMove = true;
          if(!timeChecked) {
            trafficLightRoutine = addData(gameObject.name, (engine.averageSpeed * 30).ToString("0.00"),
(engine.waitingTime / 60).ToString("0.00"));
            StartCoroutine(trafficLightRoutine);
            engine.waitingTime = 0;
            timeChecked = true;
          }
        }
    }
  }
}
```

ДОДАТОК Б

Інструкція користувачеві веб-додатком

Листів 4

Розробник Рота А.В.

Керівник Кравченко О.В.

Київ - 2022

Перше, що треба зробити для того, щоб мати змогу отримати доступ до функціоналу веб-додатка, - це авторизуватися. Авторизація відбувається на сторінці логіну.

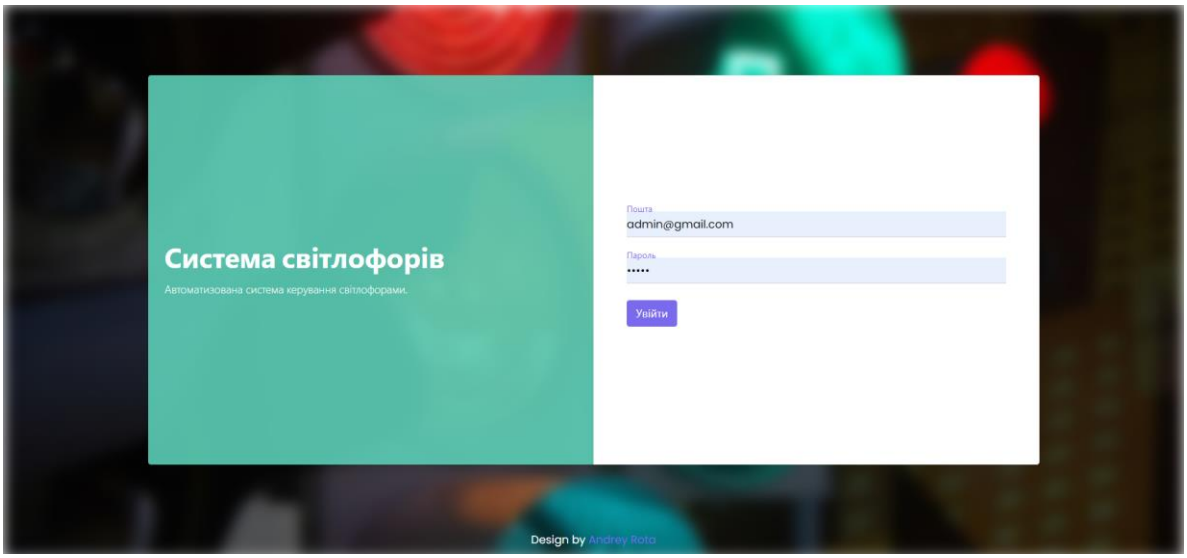


Рисунок 1 - Сторінка авторизації

Якщо ви ввели коректну пошту и правильний пароль, то наступним кроком ви попадете на головну сторінку автоматизованої системи керування дорожнім рухом. На головній сторінці ви побачите розділи з кількістю датчиків, світлофорів, камер, персоналу, графіки, які показують середню швидкість машин за день та за місяць, завантаженість доріг і так інше. Ця інформація може бути використана вами для того, щоб правильно оцінити ситуацію на вулицях міста і вчасно спланувати заходи щодо покращення пропускної здатності.

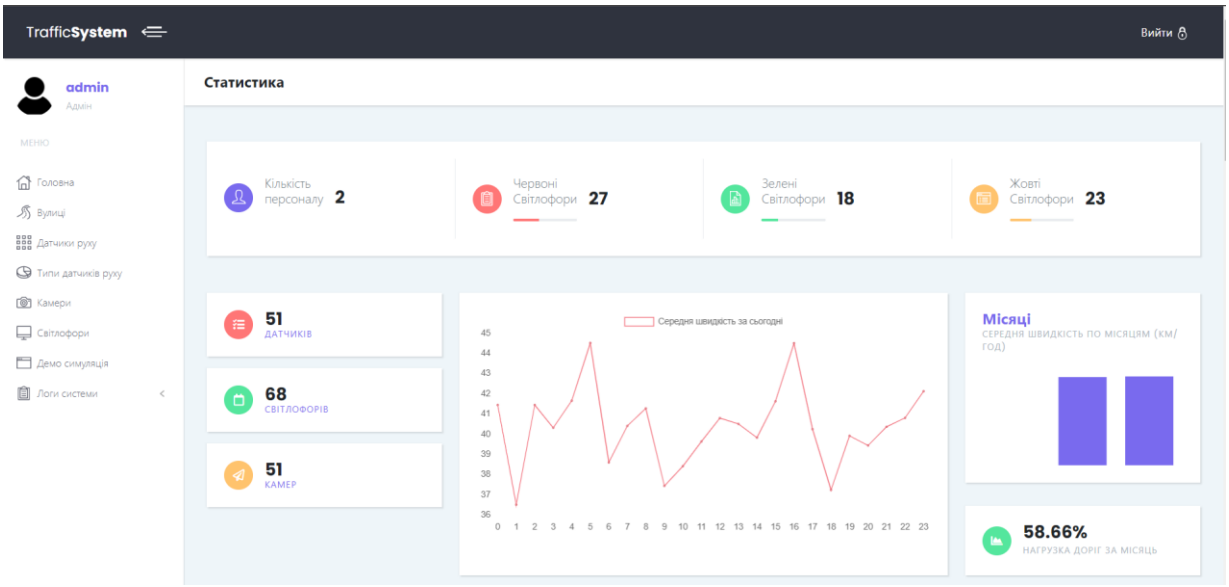


Рисунок 2 – Головна сторінка

Якщо ви хочете додати нову вулицю, або новий датчик / тип датчику, чи світлофор, то вам треба перейти на відповідну вкладку зображену на лівому меню системи.

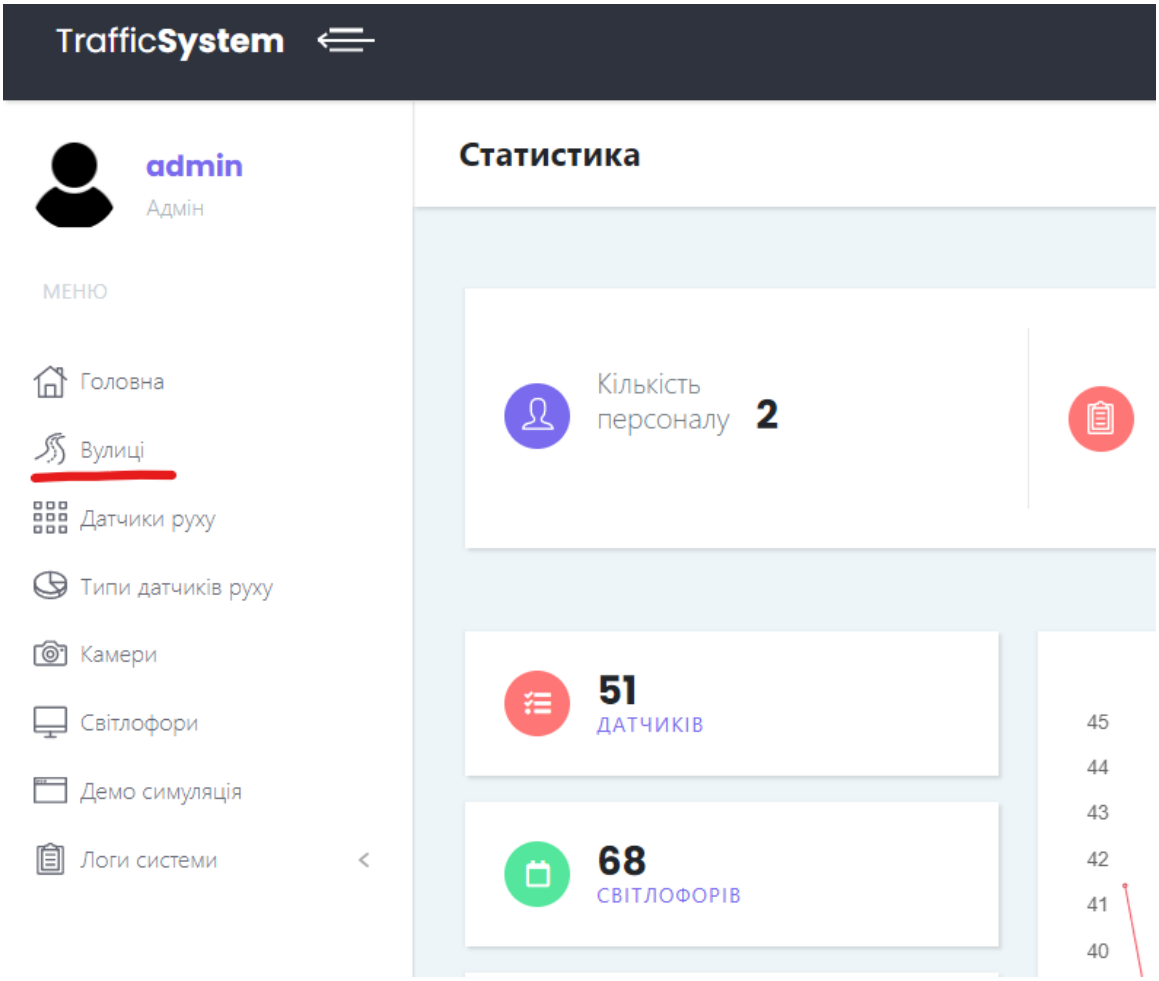


Рисунок 3 – Пункти меню

Кожна з цих вкладок дасть вам змогу переглянути дані, фільтрувати їх, шукати необхідні дані і таке інше. На представленій таблиці ви можете натиснути кнопку «Видалити» для того, щоб видалити запис з бази даних, або натиснути кнопку «Редагувати», щоб потрапити на сторінку редагування вибраного поля.

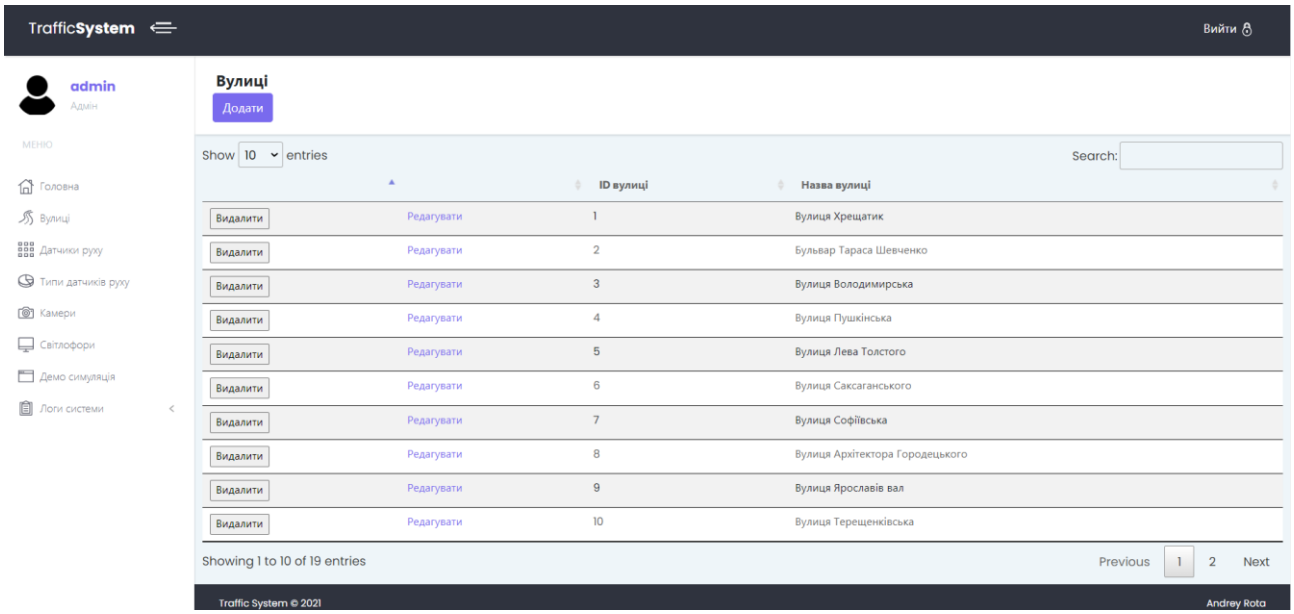


Рисунок 4 – сторінка зі списком вулиць

Для того, щоб переглянути логи системи треба натиснути на пункт лівого меню «Логи системи» і там вибрати або пункт «Логи світлофорів», або «Логи вулиць». Вам відкриється сторінка з логами системи, де буде видно час переключення режимів роботи світлофорів, чи дані про проїжджаючі машини відповідно.

Для того, щоб знайти необхідне значення, треба просто вписати його в пошук.

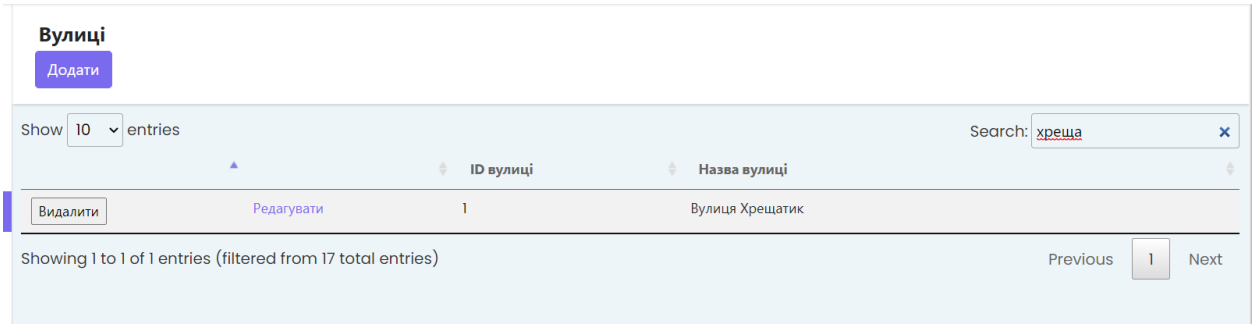


Рисунок 5 – пошук даних в таблиці

Для того, щоб відсортувати значення вам потрібно натиснути на назву потрібного стовпця.

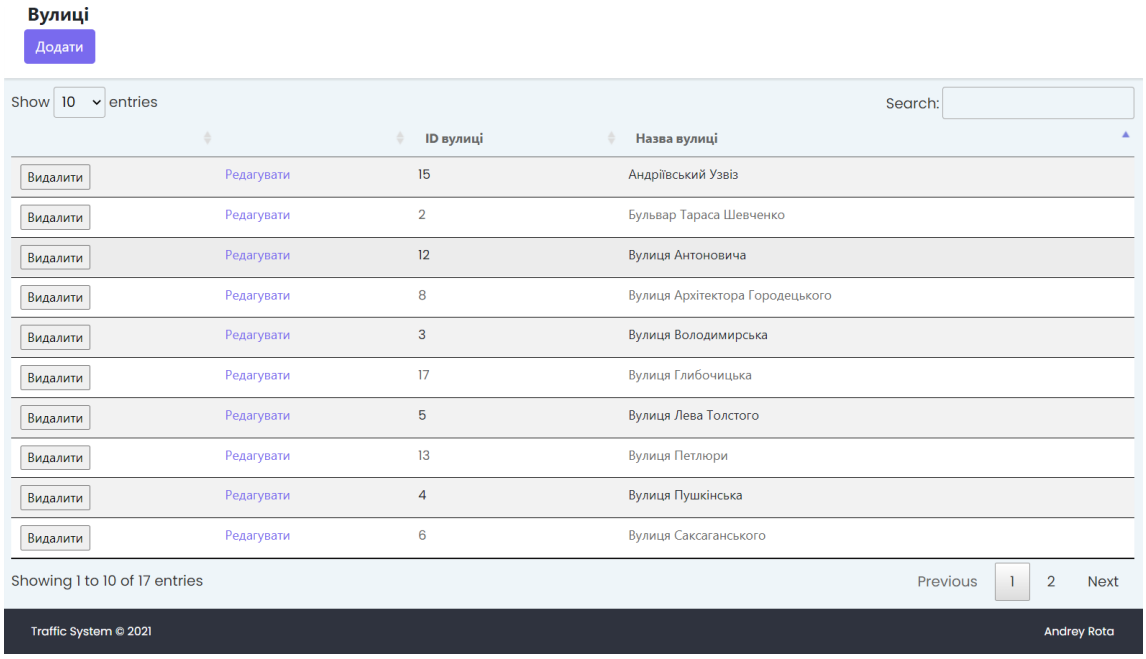


Рисунок 6 – сортування даних

Для того, щоб додати нового адміністратора треба натиснути на ваш поточний логін у лівому меню і ви потрапите на сторінку редагування адміністраторів системи.

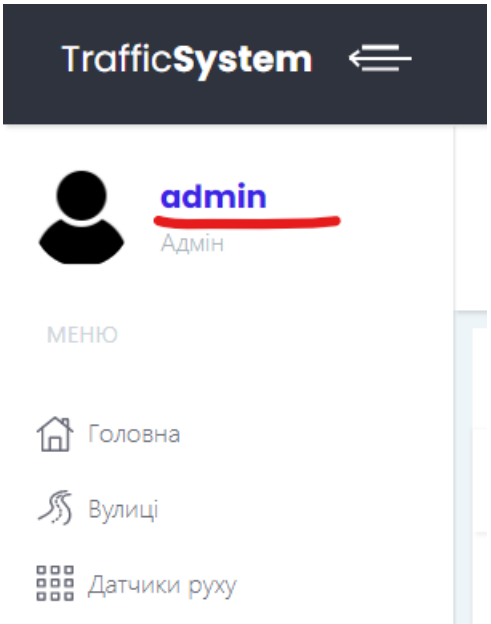


Рисунок 7 – поточний логін у лівому меню

Для того, щоб вийти із системи, вам потрібно натиснути на кнопку «Вийти» у верхньому меню.

ДОДАТОК В

Основні слайди презентації

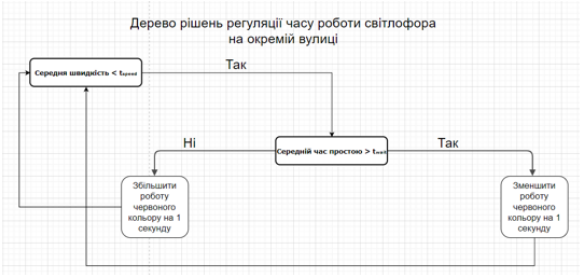
Листів 1

Розробник Рота А.В.

Керівник Кравченко О.В.

Київ - 2022

Алгоритм адаптації симуляції



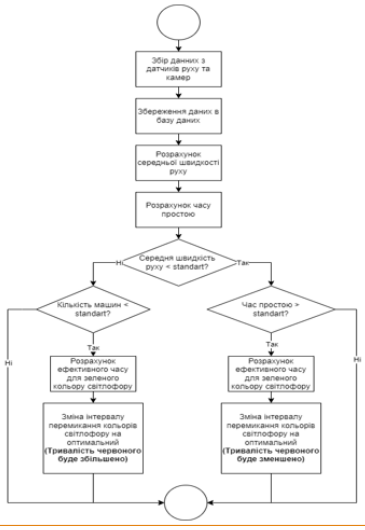


Рисунок 1 – Алгоритм адаптації симуляції

Оцінювання алгоритму формулою Вебстера для оцінки тривалості світлофорного циклу

$$T_{ц} = \frac{1{,}5\sum_{i=1}^{n} t_{ni} + 5}{1 - \sum_{i=1}^{n} y_i},\ \text{с},$$

де t_{ni} – тривалість перехідного інтервалу в i-й фазі регулювання, с; y_i – фазовий коефіцієнт i-ї фази, який визначають через співвідношення $y_i = \frac{N_i}{S_i}$

де N_i – інтенсивність руху ТЗ у цьому напрямку в i-й фазі регулювання, авт./с

S_i – потік насичення в цьому самому напрямку, авт./с.

Деякі дані з вибірки:

Машина зі швидкістю: 41 проїхала в 2022-06-07 09:13:08
Час простою: 5

Машина зі швидкістю: 40 проїхала в 2022-06-07 09:13:09
Час простою: 5

Період: 2022-06-07 09:13:12 - 2022-06-07 09:13:38
Період (червоний-жовтий-зелений-червоний) в сек: 26
Машина зі швидкістю: 52 проїхала в 2022-06-07 09:13:23
Час простою: 14

Машина зі швидкістю: 47 проїхала в 2022-06-07 09:13:25
Час простою: 11

Машина зі швидкістю: 52 проїхала в 2022-06-07 09:13:24
Час простою: 9

Машина зі швидкістю: 67 проїхала в 2022-06-07 09:13:24
Час простою: 6

Обраний період інтенсивної симуляції: 2022-06-07 09:09:26 - 2022-06-07 09:24:27

Тривалість світлофорного циклу за формулою Вебстера: 119.73881954096

З точки зору безпеки, якщо тривалість циклу > 120, то така тривалість не є допустимою за Вебстером. Необхідно домогтися зниження тривалості циклу шляхом збільшення кількості смуг руху на підході до перехрестя, заборони окремих маневрів, зниження кількості фаз регулювання, організації пропуску інтенсивних потоків протягом двох і більше фаз.

Симуляція Unity розрахована на 1 смугу руху. При повторних підрахунках з урахуванням ще однієї доданої смуги, отримуємо:

Тривалість світлофорного циклу за формулою Вебстера: 57.338752018669

Рисунок 2 – Оцінювання алгоритму формулою Вебстера для оцінки тривалості світлофорного циклу

Початкова адаптація

Симуляція також може працювати в режимі «початкової адаптації», коли вона дивиться по своїй базі якими є оптимальний час роботи зеленого і червоного кольорів. Приклад коду:

```
$yellow_query = $pdo->query("
        SELECT `timeOfChange` FROM `trafficlightslogs`
        WHERE `timeOfChange` >= '".$traffic_lights_periods[$i]["timeOfChange"]."' AND `currentModeId`='3' ORDER BY `timeOfChange` ASC LIMIT 1
    ");
```

Отримуємо: 4,42 секунди – оптимальний час для роботи червоного кольору, щоб виконувалася норма за формулою Вебстера. Відповідно при зміні інтенсивності потоку транспортних засобів, система буде і далі адаптувати час своєї роботи.

Рисунок 3 – початкова адаптація