**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис)
“___” червня ___2025 р.

# Дипломний проєкт

**на здобуття ступеня бакалавра**

**за освітньо-професійною програмою**
**«Системне програмування та спеціалізовані комп’ютерні системи»**
**123 «Комп’ютерна інженерія»**

на тему: «Комп’ютерна система моніторингу та прогнозування трафіку в міському середовищі»

Виконав: студент IV курсу, групи КВ-13
Приліпко Максим Олександрович _____
(підпис)

Керівник доц.каф. СПСКС, к.т.н., доц.
Павловський Володимир Ілліч _____
(підпис)

Консультант з нормоконтролю доц.каф. СПСКС, к.т.н
Клятченко Ярослав Михайлович _____
(підпис)

Рецензент професор каф. ОТ ФІВТ, д.т.н., проф.
Жабін Валерій Іванович _____
(підпис)

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент _____
(підпис)

Київ – 2025 року

**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп'ютерних систем

Рівень вищої освіти – перший (бакалаврський)

**Освітньо-професійною програма**

**«Системне програмування та спеціалізовані**

**комп'ютерні системи»**

**Спеціальність 123 «Комп'ютерна інженерія»**

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Віталій РОМАНКЕВИЧ
(підпис) (ініціали, прізвище)
«___» червня 2025 р.

**ЗАВДАННЯ**
**на дипломний проєкт студента**
Приліпко Максима Олександровича

1. Тема проєкту «Комп'ютерна система моніторингу та прогнозування трафіку в міському середовищі»

   Керівник проєкту доцент каф. СПСКС, к.т.н., доц..
   Павловський Володимир Ілліч,

затверджені наказом по університету від «__» _____2025 р. №_____

2. Термін подання студентом проєкту _____

3. Вихідні дані до проєкту див. Технічне завдання.

4. Зміст пояснювальної записки

---

1) Аналіз існуючих систем моніторингу та прогнозування міського трафіку
2) Проєктування архітектури комп'ютерної системи на основі мікросервісного підходу
3) Розробка модулів збору даних з IoT-датчиків, камер та GPS-трекерів

4) Проєктування бази даних часових рядів на основі TimescaleDB
5) Розробка алгоритмів попередньої обробки та очищення даних
6) Реалізація моделей машинного навчання для прогнозування (LSTM, XGBoost, TCN)
7) Розробка REST API та веб-інтерфейсу для візуалізації даних
8) Реалізація механізмів масштабування та резервування системи
9) Тестування та оцінка ефективності розробленої системи

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень, плакатів, презентацій тощо)

1) Структурна схема програмного забезпечення системи моніторингу трафіку (Д1) - обов'язково
2) Схема алгоритму збору та попередньої обробки даних трафіку (Д2) - обов'язково
3) Схема структури бази даних системи моніторингу трафіку (Д3) - обов'язково
4) Діаграма варіантів використання системи моніторингу трафіку (Д4) - обов'язково

6. Консультанти розділів проєкту*

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | | | |

7. Дата видачі завдання «31» квітня 2025 р

Календарний план

| № з/п | Назва етапів виконання дипломного проєкту | Термін виконання етапів проєкту | Примітка |
|---|---|---|---|
| 1. | Аналіз предметної області та існуючих рішень | 10.12.2024 | |
| 2. | Формування вимог до системи | 20.12.2024 | |
| 3. | Проєктування архітектури системи | 15.01.2025 | |
| 4. | Проєктування бази даних та API | 31.01.2025 | |
| 5. | Розробка модулів збору даних | 20.02.2025 | |
| 6. | Розробка модулів обробки даних | 10.03.2025 | |
| 7. | Реалізація алгоритмів прогнозування | 31.03.2025 | |
| 8. | Розробка користувацького інтерфейсу | 20.04.2025 | |
| 9. | Інтеграція та тестування системи | 10.05.2025 | |

Студент _____ Максим ПРИЛІПКО
(підпис) (Ім'я та ПРІЗВИЩЕ)

Керівник проєкту _____ Володимир ПАВЛОВСЬКИЙ
(підпис) (Ім'я та ПРІЗВИЩЕ)

.

# АНОТАЦІЯ

Кваліфікаційна робота включає пояснювальну записку обсягом 90 сторінок, яка містить 10 рисунків, 4 таблиць і 29 використаних джерел.

Об'єктом дослідження є процес моніторингу та прогнозування міського трафіку із залученням технологій Інтернету речей, машинного навчання та хмарних обчислень. Предметом дослідження виступають алгоритми обробки часових рядів і візуальних даних, зокрема згорткові та рекурентні нейронні мережі для прогнозування трафіку.

Метою роботи є розробка та впровадження комп'ютерної системи для збору, аналізу та прогнозування даних про дорожній рух у реальному часі, яка дозволить оптимізувати транспортні потоки та зменшити затори в українських містах.

У ході дослідження:

проаналізовано існуючі системи моніторингу трафіку та визначено їх обмеження;

спроєктовано багаторівневу архітектуру системи з шістьма основними підсистемами;

розроблено гібридну модель зберігання даних на базі TimescaleDB, PostgreSQL, MongoDB;

реалізовано модулі збору даних, прогнозування (LSTM, XGBoost, TCN), інтерфейс користувача та REST API;

досягнуто точності короткострокового прогнозу до 9,8% MAPE;

впроваджено автомасштабування, систему резервування та високу доступність (до 99,95%).

Практична значущість роботи полягає у можливості скорочення часу в дорозі до 25%, зниження навантаження на транспортну інфраструктуру та покращення екологічного стану міст. Запропонована система може стати основою для створення адаптивних інтелектуальних транспортних рішень у рамках концепції «розумного міста».

Ключові слова: моніторинг трафіку, прогнозування заторів, IoT, машинне навчання, часові ряди, розумне місто, мікросервісна архітектура**,** TimescaleDB.

# ABSTRACT

The qualification thesis includes an explanatory note of 90 pages, containing 10 figures, 4 tables, and 29 references.

The object of the research is the process of monitoring and forecasting urban traffic using Internet of Things (IoT) technologies, machine learning, and cloud computing. The subject of the study focuses on algorithms for processing time series and visual data, particularly the use of convolutional and recurrent neural networks for traffic forecasting.

The aim of the thesis is to develop and implement a computer system for real-time traffic data collection, analysis, and forecasting, which will help optimize traffic flows and reduce congestion in Ukrainian cities.

In the course of the research:

- existing traffic monitoring systems were analyzed and their limitations identified;
- a multi-layered system architecture with six main subsystems was designed;
- a hybrid data storage model based on TimescaleDB, PostgreSQL, and MongoDB was developed;
- data acquisition, forecasting (LSTM, XGBoost, TCN), user interface, and REST API modules were implemented;
- short-term traffic forecast accuracy of up to 9.8% MAPE was achieved;
- autoscaling, fault tolerance, and high availability (up to 99.95%) mechanisms were introduced.

The practical significance of the project lies in its potential to reduce travel time by up to 25%, ease the burden on transportation infrastructure, and improve environmental conditions in urban areas. The proposed system can serve as a foundation for adaptive intelligent transportation solutions within the smart city

paradigm.

**Keywords**: traffic monitoring, congestion forecasting, IoT, machine learning, time series, smart city, microservice architecture, TimescaleDB.

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045492.002 ТЗ | Комп'ютерна система | 6 | | |
| | | | моніторингу та прогнозування | | | |
| | | | міського трафіку | | | |
| | | | Технічне завдання | | | |
| | А4 | ІАЛЦ.045492.003 ТП | Комп'ютерна система | 3 | | |
| | | | моніторингу та прогнозування | | | |
| | | | міського трафіку | | | |
| | | | Відомість технічного | | | |
| | | | проекту | | | |
| | А4 | ІАЛЦ.045492.004 ПЗ | Комп'ютерна система | 76 | | |
| | | | моніторингу та прогнозування | | | |
| | | | міського трафіку | | | |
| | | | Пояснювальна записка | | | |
| | А4 | ІАЛЦ.045492.005 Е1 | Структура бібліотеки для | 1 | | |
| | | | забезпечення системи | | | |
| | | | моніторингу трафіку | | | |
| | | | Схема структурна | | | |
| | А4 | ІАЛЦ.467100.008 Д4 | Діаграма варіантів | 1 | | |
| | | | використання системи | | | |
| | | | моніторингу трафіку | | | |
| | | | Схема структурна | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467100.001 ОА | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Приліпко М.О | | | Комп'ютерна система моніторингу та прогнозування міського трафіку<br>**Опис альбому** | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Павловський В.І. | | | | | |
| Консульт. | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-13 | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Романкевич В.О. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.045492.004 ПЗ | Комп'ютерна система | 76 | | |
| | | | моніторингу та прогнозування | | | |
| | | | міського трафіку | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045492.005 Е1 | Структура програмного | 1 | | |
| | | | забезпечення системи | | | |
| | | | моніторингу трафіку | | | |
| | | | Схема структурна | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045492.006 Д2 | Алгоритм збору та | 1 | | |
| | | | попередньої обробки | | | |
| | | | пристроєм системи | | | |
| | | | даних трафіку | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | А4 | ІАЛЦ.045492.007 Д3 | Структура бази даних | 1 | | |
| | | | системи моніторингу | | | |
| | | | трафіку | | | |
| | | | Схема структурна | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата |
|---|---|---|---|---|
| Розробив | | Приліпко М.о. | | |
| Перевірив | | Павловський В.І. | | |
| Консульт. | | | | |

# ІАЛЦ.045492.003 ТП

Комп'ютерна система моніторингу та прогнозування міського трафіку

**Відомість технічного проекту**

| Літ. | Аркуш | Аркушів |
|---|---|---|
| | 1 | 3 |

КПІ
ім. Ігоря Сікорського,
ФПМ КВ-13

**ЗМІСТ**

## НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Комп'ютерна система моніторингу та прогнозування трафіку в міському середовищі».

Галузь застосування: інтелектуальні транспортні системи (ITS), комп'ютерні системи, міське планування, Smart City.

## ПІДСТАВА ДЛЯ РОЗРОБКИ

Завдання на виконання дипломної роботи за спеціальністю 122 – Комп'ютерні науки, що було схвалене на засіданні кафедри відповідно до навчального плану.

## МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проєкту є створення масштабованої комп’ютерної системи для моніторингу стану дорожнього руху в реальному часі та прогнозування заторів із використанням технологій штучного інтелекту, Інтернету речей та телеметричних даних. Система призначена для забезпечення міських служб та кінцевих користувачів інструментами для прийняття рішень, зменшення заторів, підвищення ефективності перевезень та покращення екологічного стану міста.

## ДЖЕРЕЛА РОЗРОБКИ

Технічна та наукова література, стандарти ISO/TC 204, документація до API транспортної інфраструктури, джерела з мережі Інтернет, результати публікацій у сфері штучного інтелекту, IoT та аналізу часових рядів.

## ТЕХНІЧНІ ВИМОГИ

**Вимоги до програмного продукту, що розробляється:**

- Підтримка сучасних браузерів (Google Chrome, Firefox, Safari, Edge);
- Збір даних у реальному часі з IoT-датчиків, камер і відкритих API;
- Модуль попередньої обробки, очищення та нормалізації даних;
- Модуль прогнозування трафіку на основі моделей машинного навчання (LSTM, TCN, Prophet);
- Інтерактивна карта з візуалізацією стану трафіку та прогнозів;
- Система керування правами доступу (ролі: водій, оператор, міський планувальник);

- Вебінтерфейс для користувачів та адміністратора з можливістю перегляду аналітики;
- REST API для взаємодії з іншими системами.

**Вимоги до апаратного забезпечення:**

- Сервер: CPU — мінімум 8 ядер, 2.4 GHz; RAM — 32 GB; SSD — 512 GB;
- IoT-обладнання: камери, датчики руху, метеостанції з API;
- Підключення до інтернету зі швидкістю від 10 Мбіт/с.

**Вимоги до програмного та апаратного забезпечення користувача:**

- Операційна система: Windows 10+, Linux, macOS, Android, iOS;
- Сучасний браузер із підтримкою JavaScript, HTML5, WebGL;
- Стабільне підключення до Інтернету (не менше 5 Мбіт/с).

ЕТАПИ РОЗРОБКИ

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів |
|---|---|---|
| 1 | Вивчення літератури за тематикою проєкту | 01.03.2025 |
| 2 | Розроблення та узгодження технічного завдання | 12.03.2025 |
| 3 | Аналіз існуючих систем моніторингу та прогнозування трафіку | 30.03.2025 |
| 4 | Підготовка матеріалів першого розділу дипломного проєкту | 02.04.2025 |
| 5 | Підготовка матеріалів другого розділу дипломного проєкту | 08.05.2025 |
| 6 | Підготовка матеріалів третього розділу дипломного проєкту | 11.05.2025 |
| 7 | Підготовка матеріалів четвертого розділу дипломного проєкту | 18.05.2025 |
| 8 | Підготовка графічної частини (архітектура, блок-схеми, інтерфейси, діаграми) | 20.05.2025 |
| 9 | Оформлення документації дипломного проєкту | 25.05.2025 |
| 10 | Попередній захист та рецензування дипломної роботи на кафедрі | 30.05.2025 |

Пояснювальна записка

до дипломного проєкту

на тему: « Комп’ютерна система моніторингу та прогнозування трафіку в міському середовищі »

Київ - 2025

ЗМІСТ

## ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ ТА ТЕРМІНІВ

АІС - автоматизована інформаційна система, що використовується для збору та обробки даних про дорожній рух

БД - база даних для зберігання інформації про транспортні потоки та дорожні умови

ІУТС - інтелектуальні управляючі транспортні системи, що оптимізують транспортні потоки у міському середовищі

КС - комп'ютерна система, що забезпечує функціонування програмно-апаратного комплексу моніторингу трафіку

СКБД - система керування базами даних, що забезпечує доступ до інформації про трафік

ТЗ - транспортний засіб, об'єкт моніторингу в системі аналізу дорожнього руху

AI - Artificial Intelligence (штучний інтелект), алгоритми, що імітують людський інтелект для прогнозування трафіку

API - Application Programming Interface (інтерфейс прикладного програмування), інтерфейс для взаємодії програмних компонентів системи

CNN - Convolutional Neural Network (згорткова нейронна мережа), тип нейронної мережі для обробки візуальних даних моніторингу

GPS - Global Positioning System (глобальна система позиціонування), система глобального позиціонування для визначення місцезнаходження

ТЗ IoT - Internet of Things (інтернет речей), мережа фізичних об'єктів, що збирають та обмінюються даними

ML - Machine Learning (машинне навчання), методи, що дозволяють комп'ютерам навчатись на даних про трафік

RMSE - Root Mean Square Error (середньоквадратична помилка), метрика оцінки точності прогнозування трафіку

RNN - Recurrent Neural Network (рекурентна нейронна мережа), тип нейронної мережі для аналізу послідовних даних про трафік

# ВСТУП

Стрімка урбанізація та зростання кількості транспортних засобів у всьому світі призвели до безпрецедентних викликів у міській мобільності. Затори на дорогах стали критичною проблемою, що призводить до економічних збитків, погіршення стану довкілля та зниження якості життя [1]. Сучасні міста потребують можливостей моніторингу дорожнього руху в реальному часі в поєднанні з прогнозною аналітикою для передбачення заторів та впровадження проактивних заходів [3].

Актуальність дослідження

Розробка комп'ютерних систем моніторингу та прогнозування міського трафіку є критично важливим кроком до інтелектуального управління містом. Традиційні підходи до управління дорожнім рухом виявилися недостатніми для вирішення динамічного характеру міських моделей мобільності. Ця дослідницька область знаходиться на перетині комп'ютерних наук, міського планування та транспортної інженерії.

Мета та завдання

Основною метою цієї курсової роботи є проєктування та розробка комп'ютерної системи моніторингу та прогнозування міського трафіку, яка інтегрує передові технології, включаючи Інтернет речей (IoT), аналітику великих даних та алгоритми машинного навчання. Конкретні завдання включають:

1. Аналіз існуючих методологій моніторингу трафіку та виявлення їх обмежень;
2. розробку комплексної архітектури системи, що включає як апаратні, так і програмні компоненти;
3. впровадження ефективних алгоритмів для збору, обробки та аналізу даних про трафік;
4. створення моделей прогнозування з використанням методів штучного інтелекту для прогнозування умов дорожнього руху;
5. оцінку точності та надійності системи шляхом симуляції та тестування.

Об’єкт та предмет дослідження

Об’єктом цього дослідження є процес моніторингу та прогнозування міського трафіку, що охоплює збір, аналіз та прогнозування даних. Предметом дослідження є методи та алгоритми аналізу та прогнозування даних про трафік, зокрема вивчення того, як згорткові та рекурентні нейронні мережі можуть бути застосовані для обробки даних візуального моніторингу та аналізу послідовних патернів трафіку.

Наукова та практична значущість

Ця робота має значну практичну значущість для органів управління міським рухом, транспортних планувальників та окремих учасників дорожнього руху. З наукової точки зору, дослідження робить внесок у зростаючу базу знань про прикладне машинне навчання в транспортних системах, особливо в українському міському контексті.

# 1 АНАЛІЗ КОМП'ЮТЕРНИХ СИСТЕМ МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ ТРАФІКУ

## 1.1 Область застосування систем моніторингу міського трафіку

Системи моніторингу міського трафіку є ключовим компонентом сучасної інфраструктури розумного міста, пропонуючи різноманітні застосування, що вирішують нагальні проблеми сучасного міського середовища. Ці системи слугують технологічною основою для прийняття рішень на основі даних в управлінні транспортом та міському плануванні [1].

Управління дорожнім рухом у реальному часі є основною сферою застосування, де безперервні потоки даних про трафік дозволяють владі виявляти точки заторів та впроваджувати адаптивні заходи контролю. За даними Corisinta [2], передові системи моніторингу можуть скоротити час у дорозі до 25% завдяки оптимізованій координації сигналів світлофора. Ця оптимізація поширюється на служби екстреної допомоги, дозволяючи транспортним засобам пересуватися перевантаженими міськими районами зі значно покращеним часом реагування.

Зібрані дані надають неоціненну інформацію для міського планування та розвитку інфраструктури, виявляючи патерни трафіку, що керують стратегічними рішеннями щодо розширення доріг, маршрутів громадського транспорту та політики міського зонування. Оцінка впливу на навколишнє середовище також виграє, оскільки патерни викидів, виявлені за допомогою моніторингу трафіку, допомагають містам визначати зони з високим рівнем забруднення та впроваджувати цільові стратегії пом'якшення.

Комерційні логістичні операції використовують ці системи для оптимізації маршрутів доставки, зменшуючи як споживання палива, так і покращуючи надійність обслуговування. При інтеграції з іншими компонентами розумного міста, такими як інтелектуальне освітлення, управління відходами та системи безпеки, моніторинг трафіку створює комплексну екосистему управління містом, яка підвищує загальну придатність міста для життя та економічну життєздатність. Цей взаємопов'язаний підхід представляє майбутній напрямок

управління міським транспортом, де рішення на основі даних оптимізують ресурси та покращують якість життя.

### 1.2 Існуючі рішення та їх обмеження

Ландшафт систем моніторингу та прогнозування трафіку охоплює різноманітний спектр технологічних підходів, кожен з яких має свої можливості та обмеження.

Сенсорні системи моніторингу, що використовують індуктивні петлі, радари та інфрачервоні детектори, забезпечують надійні вимірювання, але стикаються зі значними проблемами. Дослідження IJCT [1] показує, що ці традиційні придорожні датчики зазвичай коштують від 20,000 до 40,000 за перехрестя, створюючи суттєвий фінансовий бар'єр для широкого впровадження. Крім того, їх фіксоване розміщення обмежує покриття заздалегідь визначеними точками дорожньої мережі, тоді як фактори навколишнього середовища та фізичні пошкодження часто порушують збір даних, що призводить до значних витрат на технічне обслуговування.

Системи моніторингу на основі камер, що використовують методи комп'ютерного зору, набули популярності завдяки своїй універсальності. Дослідження з International Research Journal of Education and Technology [3] демонструє, що сучасні системи комп'ютерного зору можуть досягати точності виявлення 92-97% в ідеальних умовах. Однак продуктивність значно погіршується під час несприятливих погодних умов, обмеженої видимості та в нічний час. Ці системи також генерують величезні обсяги даних, що вимагають значної пропускної здатності та ємності сховища.

Мобільні сенсорні мережі, що використовують пристрої, встановлені на транспортних засобах, пропонують ширше покриття та економічну ефективність, але генерують дані змінної якості через непослідовне калібрування датчиків та розподіл транспортних засобів. Проблеми конфіденційності щодо відстеження місцезнаходження також становлять значні регуляторні виклики в багатьох юрисдикціях.

Щодо методологій прогнозування, традиційні статистичні моделі часових рядів демонструють хорошу обчислювальну ефективність, але погано справляються з нерегулярними подіями заторів, зазвичай досягаючи середньої абсолютної відсоткової помилки (MAPE) 15-25% для прогнозування міського трафіку за даними Corisinta [2]. Сучасні підходи машинного навчання демонструють покращену продуктивність прогнозування, але вимагають значних обчислювальних ресурсів і сильно залежать від якості даних, тоді як їх характер "чорної скриньки" створює проблеми з пояснюваністю в застосуваннях державного сектору.

Таблиця 1.1 - Порівняння існуючих систем моніторингу трафіку

| **Тип системи** | **Метод збору даних** | **Точність прогнозу (MAPE)** | **Масштабованість** | **Можливість реального часу** | **Рівень витрат** |
|---|---|---|---|---|---|
| Сенсорна | Стаціонарні дорожні датчики | 10-15% | Обмежена | Висока | Високий |
| Камерна | Комп’ютерний зір | 8-12% | Помірна | Помірна | Помірний |
| Мобільна | Розподілені пристрої | 12-20% | Висока | Змінна | Низький |
| Гібридна | Кілька методів | 7-10% | Помірна | Висока | Високий |

Додаткові загальні обмеження для різних категорій рішень включають проблеми інтеграції з існуючою міською інфраструктурою, проблеми конфіденційності даних, питання економічної ефективності та труднощі в збалансуванні моніторингу в реальному часі з можливостями прогнозної аналітики.

## 1.3 Постановка задачі на розробку комп'ютерної системи моніторингу та прогнозування трафіку

На основі аналізу областей застосування та обмежень існуючих рішень, цей розділ встановлює технічні вимоги до розробки передової комп'ютерної системи моніторингу та прогнозування міського трафіку.

**Цілі системи**

Основною метою є розробка інтегрованої комп'ютерної системи, здатної:

1. Збирати багатоджерельні дані про трафік з гетерогенних сенсорних пристроїв**;**
2. Обробляти та аналізувати дані про трафік у реальному часі з високою точністю**;**
3. Генерувати короткострокові (15-30 хвилин) та середньострокові (1-2 години) прогнози трафіку**;**
4. Візуалізувати поточні та прогнозовані умови дорожнього руху через інтуїтивно зрозумілі інтерфейси**;**
5. Підтримувати прийняття рішень для органів управління дорожнім рухом та окремих користувачів**;**

**Функціональні вимоги**

Можливості збору даних:

- підтримка кількох джерел даних, включаючи придорожні датчики, камери дорожнього руху, дані рухомих автомобілів та телеметрію громадського транспорту**;**
- збір даних з мінімальною частотою 1 вибірка на хвилину для критичних точок моніторингу**;**
- стандартизовані інтерфейси API, сумісні з існуючою українською транспортною інфраструктурою**;**
- безперервна робота з автоматизованими механізмами відновлення при збоях зв'язку**;**

Обробка та аналіз даних

- обробка вхідних даних з максимальною затримкою 10 секунд**;**

– алгоритми очищення та валідації даних для виявлення та виправлення аномальних показань;
– історична база даних з мінімальним терміном зберігання 12 місяців;
– потужність обробки щонайменше 10 000 точок даних за хвилину під час пікових операцій;

Можливості прогнозування трафіку;

– прогнози параметрів потоку, швидкості та щільності трафіку з 5-хвилинними інтервалами;
– короткострокові прогнози з мінімальною точністю 90% за нормальних умов;
– середньострокові прогнози з мінімальною точністю 85%;
– включення факторів навколишнього середовища в моделі прогнозування;
– механізми безперервного перенавчання моделі;

Візуалізація та користувацький інтерфейс;

– веб-інтерфейси та мобільні додатки для різних категорій користувачів;
– кольорові накладення на карту, що представляють поточні та прогнозовані умови;
– настроювані сповіщення на основі визначених користувачем порогів та місць розташування;
– адміністративні інтерфейси з детальними метриками продуктивності системи;
– підтримка української та англійської мов;

**Нефункціональні вимоги**

Вимоги до продуктивності

– підтримка 1000 одночасних користувачів без погіршення продуктивності;
– час відгуку веб-інтерфейсу не перевищує 2 секунди;

– конвеєр обробки даних витримує пікові навантаження зі збільшенням часу обробки не більше ніж на 15%;
– розрахунки прогнозу завершуються протягом 30 секунд для загальноміських прогнозів;

Надійність та доступність

– доступність системи 99,9% (за винятком планового обслуговування) ;
– автоматичні механізми перемикання на резерв для критичних компонентів системи;
– процедури резервного копіювання даних виконуються з мінімальним інтервалом 6 годин;
– плавна деградація при роботі в умовах обмежених ресурсів;

Масштабованість

– підтримка горизонтального масштабування для збільшення кількості точок моніторингу;
– база даних, розрахована на зберігання історичних даних щонайменше за 5 років;
– можливість розширення потужності обробки для підтримки збільшення обсягу даних щонайменше на 50%;

Вимоги безпеки

– контроль доступу на основі ролей з мінімум чотирма рівнями привілеїв;
– шифрування передачі даних за допомогою стандартів TLS 1.3 або вище;
– відповідність українським нормам захисту даних та принципам GDPR;
– двофакторна автентифікація для адміністративного доступу;

Вимоги до апаратного та програмного забезпечення

Система повинна бути розроблена як розподілений додаток з:

– серверною інфраструктурою: хмарною або локальною з мінімум 16-ядерними процесорами, 64 ГБ ОЗП та 4 ТБ сховища;
– мережевими вимогами: мінімальна швидкість з’єднання 1 Гбіт/с між компонентами системи;

- операційним середовищем: ОС сервера на базі Linux з підтримкою контейнеризації**;**
- базою даних: PostgreSQL з розширенням TimescaleDB для управління даними часових рядів**;**
- технологіями розробки: Python для обробки даних та машинного навчання, JavaScript/React для фронтенд-інтерфейсів.

# 2 ПРОЄКТУВАННЯ КОМП'ЮТЕРНОЇ СИСТЕМИ

## 2.1 Основні вимоги до системи

Проєктування ефективної системи моніторингу та прогнозування міського трафіку вимагає комплексного набору вимог, що стосуються як функціональних можливостей, так і атрибутів якості.

**Функціональні вимоги**

Система повинна підтримувати комплексний збір даних з різноманітних джерел, включаючи камери дорожнього руху, датчики IoT, GPS-трекери та API громадського транспорту. Збір даних повинен бути конфігурованим з мінімальною частотою дискретизації 30 секунд для критичних перехресть, підтримуючи як структуровані показання датчиків, так і неструктуровані відеопотоки. Конвеєр обробки даних повинен працювати з реакцією в реальному часі, підтримуючи максимальну затримку 5 секунд, виконуючи автоматичне очищення даних, нормалізацію та виявлення аномалій для ідентифікації незвичайних патернів трафіку або несправностей датчиків.

Для прогнозування трафіку система повинна досягати точності короткострокового прогнозування щонайменше 90% за нормальних умов та точності довгострокового прогнозування щонайменше 80% для аналізу тенденцій. Ці прогнози повинні інтегруватися із зовнішніми факторами, такими як погодні умови та заплановані події, з безперервним перенавчанням моделі з конфігурованою частотою для підтримки точності в міру еволюції патернів.

Компоненти візуалізації та користувацького інтерфейсу повинні забезпечувати інтерактивну візуалізацію на основі карт з кількома інформаційними шарами, візуалізації часових рядів для історичних та прогнозованих патернів трафіку та настроювані панелі інструментів для різних ролей користувачів. Система повинна генерувати сповіщення про аномальні умови або прогнозовані затори та підтримувати доступ з різних пристроїв для максимальної корисності.

**Нефункціональні вимоги**

1. **Продуктивність**
   - підтримка моніторингу щонайменше 500 точок трафіку одночасно**;**
   - конвеєр обробки даних обробляє мінімум 10 000 точок даних за секунду**;**
   - час відгуку користувацького інтерфейсу менше 2 секунд для 95% запитів.
2. **Надійність та доступність**
   - час безвідмовної роботи системи 99,9%**;**
   - плавна деградація у разі збоїв компонентів**;**
   - автоматичні механізми відновлення для поширених сценаріїв збоїв**;**
   - відсутність єдиної точки відмови в критичних компонентах.
3. **Безпека**
   - контроль доступу на основі ролей з щонайменше трьома рівнями привілеїв**;**
   - шифрування всіх конфіденційних даних як під час передачі, так і в стані спокою**;**
   - комплексне журналювання аудиту доступу до системи та модифікацій**;**
   - відповідність місцевим нормам захисту даних.
4. **Масштабованість**
   - горизонтальна масштабованість для підтримки 50% щорічного зростання обсягу даних**;**
   - можливість розширення географічного покриття без перепроєктування архітектури**;**
   - підтримка додавання нових джерел даних та моделей прогнозування без простою системи.
5. **Ремонтопридатність**
   - модульна архітектура, що дозволяє незалежне оновлення компонентів**;**

- автоматизоване тестування, що охоплює щонайменше 80% функціональності системи**;**
- максимальний середній час ремонту 4 години для критичних збоїв.

## 2.2 Архітектура комп’ютерної системи моніторингу та прогнозування трафіку

Проєктування архітектури системи моніторингу та прогнозування міського трафіку базується на сучасних принципах розподілених систем та мікросервісної архітектури. Запропонована архітектура забезпечує високу масштабованість, відмовостійкість та модульність рішення, що є критично важливим для обробки великих обсягів даних у реальному часі. Система побудована з урахуванням необхідності інтеграції гетерогенних джерел даних, забезпечення низької затримки обробки та підтримки різноманітних аналітичних моделей.

### 2.2.1 Загальний огляд архітектури високого рівня

Багаторівнева архітектура системи, представлена на рисунку 2.1, демонструє комплексний підхід до організації компонентів системи моніторингу трафіку. Архітектурне рішення базується на принципі розділення відповідальностей, де кожна підсистема виконує чітко визначені функції та взаємодіє з іншими компонентами через стандартизовані інтерфейси.

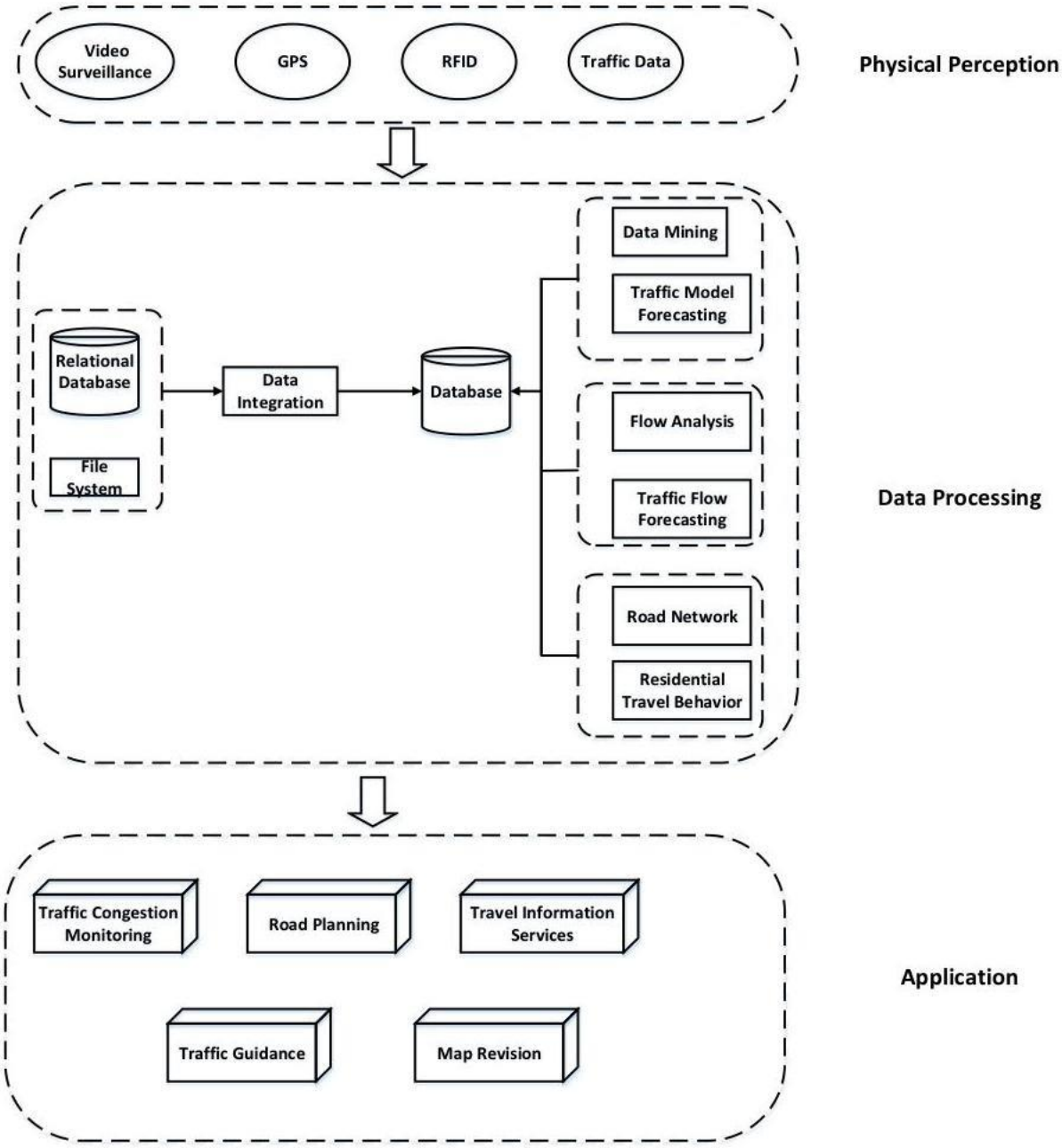



*Рисунок 2.1 - Багаторівнева архітектура інтелектуальної системи моніторингу трафіку*

Архітектура системи складається з шести основних підсистем, як показано на Рисунку 2.1:

1. **Підсистема збору даних**: взаємодіє з джерелами даних, обробляє протоколи збору даних та забезпечує надійне надходження даних.**;**
2. **Підсистема обробки даних**: виконує очищення, нормалізацію та попередню обробку необроблених даних про трафік**;**

3. **Підсистема зберігання**: керує як оперативним, так і історичним зберіганням даних, оптимізованим для інформації про трафік у вигляді часових рядів**;**
4. **Підсистема аналітики**: реалізує моделі машинного навчання для аналізу патернів трафіку та прогнозування**;**
5. **Підсистема візуалізації**: відображає дані про трафік та прогнози через інтерактивні інтерфейси**;**
6. **Підсистема управління системою**: обробляє конфігурацію, моніторинг, автентифікацію та загальну координацію системи.

Ці підсистеми взаємодіють через керовану подіями шину повідомлень, яка сприяє слабкому зв'язку та забезпечує асинхронну обробку, де це доцільно. Архітектура демонструє рівень фізичного сприйняття, що збирає дані з різних датчиків, рівень обробки даних, що відповідає за інтеграцію та аналітику, та рівень додатків, що надає практичні послуги кінцевим користувачам.

### 2.2.2 Потік даних та конвеєр обробки

Потік даних через систему відповідає логічній послідовності від збору до візуалізації. Підсистема збору даних збирає необроблені дані про трафік з кількох джерел за допомогою відповідних протоколів і публікує їх у шину повідомлень, яка направляє їх до Підсистеми обробки даних. Після валідації, очищення та попередньої обробки дані зберігаються як для негайного використання, так і для історичного аналізу. Підсистема аналітики отримує відповідні дані, застосовує алгоритми прогнозування та зберігає результати прогнозування, тоді як Підсистема візуалізації запитує як дані в реальному часі, так і прогнозовані дані для генерації інтерактивних візуалізацій.

Цей конвеєр мінімізує затримку для моніторингу в реальному часі, дозволяючи паралельно виконувати обчислювально інтенсивні операції прогнозування.

### 2.2.3 Архітектура компонентів

Кожна підсистема розбита на мікросервіси, що відповідають за конкретні обов'язки. Підсистема збору даних включає сервіси для інтеграції датчиків, обробки відеопотоків, підключення до зовнішніх API та валідації даних. Підсистема обробки даних забезпечує очищення даних, нормалізацію, вилучення ознак та виявлення аномалій, тоді як Підсистема зберігання керує базами даних часових рядів, просторовими базами даних, метаданими та архівами.

Для аналітики система включає сервіси навчання моделей, короткострокового та довгострокового прогнозування та інтеграції зовнішніх факторів. Підсистема візуалізації пропонує рендеринг карт, генерацію панелей інструментів, управління сповіщеннями та функціональність експорту. Ці компоненти координуються Підсистемою управління системою через сервіси автентифікації, конфігурації, моніторингу та шлюзу API.

### 2.2.4 Модель розгортання

Система використовує гібридну модель розгортання, що поєднує хмарну обробку з периферійними обчисленнями для збору даних. Датчики трафіку та камери підключаються до периферійних пристроїв, які виконують попередню валідацію даних та стиснення перед передачею до центральної системи.

Основні компоненти системи розгортаються як контейнеризовані сервіси в хмарному середовищі, що забезпечує горизонтальне масштабування залежно від попиту. Критичні сервіси розгортаються з резервуванням у кількох зонах доступності для усунення єдиних точок відмови.

Ця архітектура відповідає ключовим атрибутам якості, включаючи продуктивність завдяки розподіленій обробці, масштабованість за допомогою контейнеризованих мікросервісів, надійність завдяки резервному розгортанню, ремонтопридатність завдяки модульному дизайну та безпеку завдяки багаторівневому підходу з чіткими межами авторизації.

### 2.3 Проєктування бази даних та сховища телеметрії

Компоненти бази даних та сховища телеметрії використовують гібридний підхід, оптимізований для великого обсягу, часового характеру та геопросторових компонентів даних про трафік. Реалізація використовує SQLAlchemy з розширеннями GeoAlchemy2 для забезпечення надійного об’єктно-реляційного відображення.

#### 2.3.1 Проєктування моделі даних

Модель даних структурована навколо чотирьох основних типів сутностей, реалізованих як класи SQLAlchemy:

1. RoadSegment: представляє фізичну дорожню інфраструктуру з геопросторовими властивостями

```
python
class RoadSegment(Base):
    __tablename__ = 'road_segments'
    id = Column(Integer, primary_key=True, index=True)
    name = Column(String(100), nullable=False)
    geometry = Column(Geometry(geometry_type='LINESTRING',
srid=4326), nullable=False)
    length = Column(DECIMAL(10, 2))  # довжина ділянки в км
    lanes = Column(Integer)  # кількість смуг
    speed_limit = Column(Integer)  # обмеження швидкості
    road_type = Column(String(50))  # тип дороги
    direction = Column(String(20))  # напрямок
    city = Column(String(50), nullable=False, index=True)
    region = Column(String(50))

    # Зв'язки з іншими таблицями
    measurements = relationship("TrafficMeasurement",
back_populates="road_segment")
    predictions = relationship("TrafficPrediction",
back_populates="road_segment")
```

2. TrafficMeasurement: зберігає дискретні спостереження трафіку в конкретних місцях та часових точках

```
python
class TrafficMeasurement(Base):
    __tablename__ = 'traffic_measurements'
    id = Column(Integer, primary_key=True, index=True)
    road_segment_id = Column(Integer,
ForeignKey('road_segments.id', ondelete='CASCADE'),
index=True)
```

```
    timestamp = Column(TIMESTAMP(timezone=True),
nullable=False, index=True)
    vehicle_count = Column(Integer)
    average_speed = Column(DECIMAL(5, 2))
    congestion_level = Column(String(20))
    travel_time_seconds = Column(Integer)
    delay_seconds = Column(Integer)
    source = Column(String(50), nullable=False)

    road_segment = relationship("RoadSegment",
back_populates="measurements")
```

3. WeatherData: фіксує фактори навколишнього середовища, що впливають на трафік

```
python
class WeatherData(Base):
    __tablename__ = 'weather_data'
    id = Column(Integer, primary_key=True, index=True)
    city = Column(String(50), nullable=False, index=True)
    location = Column(Geometry(geometry_type='POINT',
srid=4326))
    timestamp = Column(TIMESTAMP(timezone=True),
nullable=False, index=True)
    temperature = Column(DECIMAL(5, 2))
    precipitation = Column(DECIMAL(5, 2))
    wind_speed = Column(DECIMAL(5, 2))
    visibility = Column(DECIMAL(6, 1))
    weather_condition = Column(String(100))
    humidity = Column(DECIMAL(5,2))
    pressure = Column(DECIMAL(6,2))
    cloudiness = Column(DECIMAL(5,2))
```

4. TrafficPrediction: записує прогнозну інформацію

```
python
class TrafficPrediction(Base):
    __tablename__ = 'traffic_predictions'
    id = Column(Integer, primary_key=True, index=True)
    road_segment_id = Column(Integer,
ForeignKey('road_segments.id', ondelete='CASCADE'),
index=True)
    prediction_timestamp = Column(TIMESTAMP(timezone=True),
nullable=False)
    prediction_target_timestamp =
Column(TIMESTAMP(timezone=True), nullable=False, index=True)
    predicted_speed = Column(DECIMAL(5, 2))
    predicted_congestion = Column(String(20))
    predicted_travel_time_seconds = Column(Integer)
    confidence_level = Column(DECIMAL(5, 4))
    model_version = Column(String(50), nullable=False)
    feature_set_version = Column(String(50))
```

```
    road_segment = relationship("RoadSegment",
back_populates="predictions")
```

Кожна сутність використовує відповідні зв’язки та індекси для оптимізації продуктивності запитів. Модель RoadSegment використовує спеціалізований тип Geometry для зберігання геопросторових даних, тоді як усі моделі включають стовпці з часовими мітками, що є основою для запитів часових рядів.

Структура бази даних системи моніторингу трафіку представлена на рисунку 2.3 у вигляді ER-діаграми, що демонструє основні сутності та зв'язки між ними.

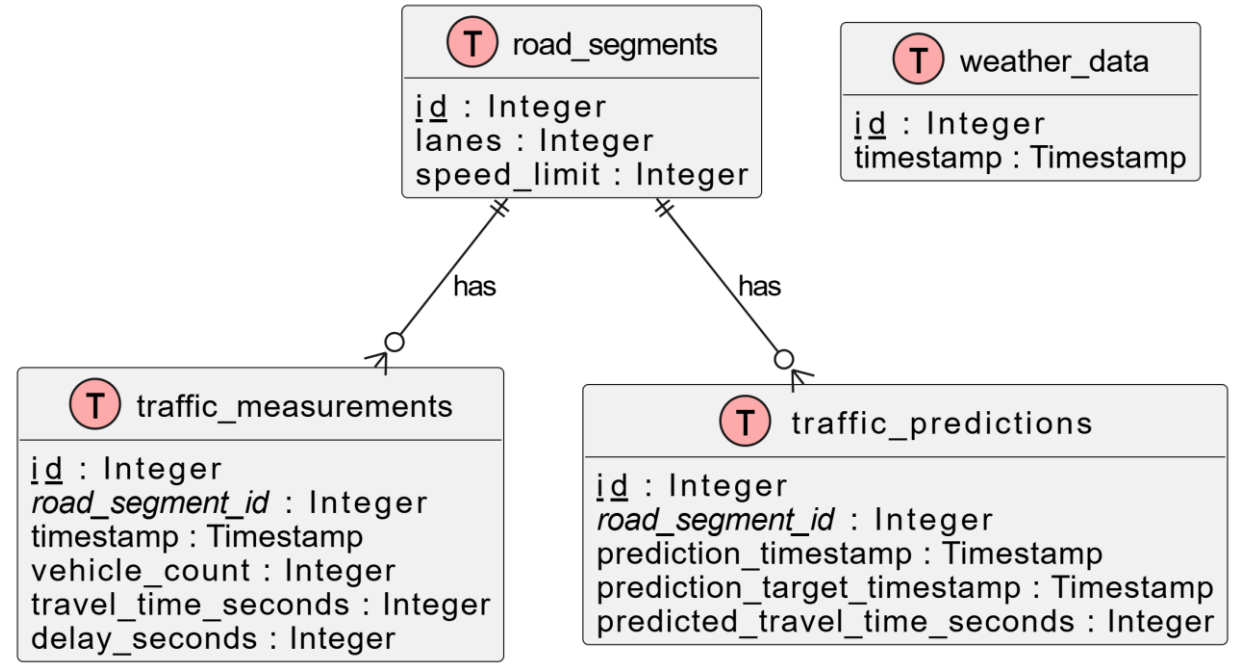



*Рисунок 2.2 - ER-діаграма структури бази даних системи моніторингу трафіку*

### 2.3.2 Вибір системи управління базою даних

Було обрано гібридний підхід, що поєднує:

1. TimescaleDB як основне сховище для вимірювань трафіку, що забезпечує оптимізоване зберігання даних часових рядів з ефективними темпоральними запитами, зберігаючи при цьому сумісність з SQL завдяки своїй основі PostgreSQL.

2. PostgreSQL з розширенням PostGIS для дорожньої мережі та геопросторових даних, пропонуючи надійну підтримку складних просторових зв’язків та сувору цілісність посилань.

3. Документне сховище (MongoDB) для контекстної інформації та неструктурованих даних, з гнучкою схемою для різноманітної контекстної інформації.

Реалізація включає менеджер контексту для обробки сесій бази даних:

```
python
@contextmanager
def get_db():
    """Надає сесію SQLAlchemy з автоматичним закриттям."""
    db = SessionLocal()
    try:
        yield db
        db.commit()
    except Exception as e:
        db.rollback()
        logging.error(f"Помилка сесії бази даних: {e}")
        raise
    finally:
        db.close()
```

### 2.3.3 Обробка темпоральних даних

Система реалізує спеціалізовані стратегії управління даними часових рядів трафіку:

1. Розбиття за часом: дані автоматично розбиваються за часовими періодами для оптимізації продуктивності запитів, причому гіпертаблиці TimescaleDB обробляють це розбиття прозоро.

2. Багатороздільне зберігання: дані зберігаються з різною часовою роздільною здатністю, включаючи необроблені дані за останні періоди (30 днів), 5-хвилинні агрегати для середньострокового аналізу (90 днів), погодинні агрегати для довгострокового аналізу (3 роки) та добові підсумки, що зберігаються необмежено.

3. Безперервна агрегація: фонові процеси автоматично генерують та підтримують агреговані представлення даних для підтримки ефективного історичного аналізу.

### 2.3.4 Стратегії оптимізації продуктивності бази даних

Використовуються кілька методів оптимізації, включаючи композитні індекси для стовпців, що часто запитуються, чанкування гіпертаблиць для даних часових рядів та матеріалізовані представлення для поширених патернів аналізу. Політики стиснення автоматично застосовують відповідні алгоритми залежно від віку даних, зменшуючи вимоги до зберігання історичних даних.

Для стійкості даних система реалізує безперервну архівацію з можливістю відновлення на певний момент часу, синхронну реплікацію для критичних даних та географічно розподілені репліки з автоматичним перемиканням на резерв.

## 2.4 Модуль збору та попередньої обробки даних у реальному часі

Цей модуль слугує основою системи, відповідаючи за отримання, валідацію та підготовку даних з різноманітних джерел.

### 2.4.1 Інтеграція джерел даних

Система інтегрується з кількома джерелами даних:

1. **Мережі камер дорожнього руху**
   - підключення через протокол RTSP для відеопотоків**;**
   - підтримка як фіксованих, так і PTZ-камер**;**
   - адаптивний вибір частоти кадрів залежно від умов мережі**;**
   - автоматичне калібрування камери для точних вимірювань.
2. **Дорожні датчики IoT**
   - підтримка індуктивних петель, радарних, інфрачервоних та акустичних датчиків;
   - протоколи MQTT та CoAP для ефективної передачі даних датчиків;
   - частота дискретизації від 10 секунд до 5 хвилин залежно від критичності місцезнаходження.
3. **Дані GPS транспортних засобів**
   - інтеграція з системами управління автопарком та навігаційними додатками;

– анонімний збір даних з агрегацією, що зберігає конфіденційність;

– конфігурована частота дискретизації залежно від можливостей постачальника даних.

4. **Системи громадського транспорту**

– місцезнаходження автобусів та трамваїв у реальному часі через канали GTFS-RT;

– моніторинг дотримання розкладу для впливу на загальний трафік.

5. **Зовнішні служби даних**

– інтеграція даних про погоду, інформація про події, графіки будівельних робіт;

### 2.4.2 Протоколи збору даних

Реалізація використовує стандартизовані протоколи, включаючи синхронізацію через NTP, постійні з'єднання з автоматичним перепідключенням, адаптивну вибірку залежно від умов трафіку, локальне кешування в точках збору та безпечне управління обліковими даними для автентифікованого доступу.

Інтеграція зовнішніх API включає Google Maps для даних про трафік:

```
python
def get_traffic_data_google_maps(start_coords, end_coords):
    """Отримує дані про трафік за допомогою Google Maps
Directions API."""
    origin = f"{start_coords['lat']},{start_coords['lon']}"
    destination = f"{end_coords['lat']},{end_coords['lon']}"
    url =
f"https://maps.googleapis.com/maps/api/directions/json?origin
={origin}&destination={destination}&departure_time=now&traffi
c_model=best_guess&key={GOOGLE_API_KEY}"

    # Код запиту до API та обробки даних...

    # Повернення структурованих даних про трафік
    traffic_info = {
        'timestamp': current_time,
        'average_speed': Decimal(str(avg_speed_kmh)) if
avg_speed_kmh is not None else None,
        'congestion_level': congestion,
        'travel_time_seconds': duration_in_traffic_sec,
        'delay_seconds': delay_seconds if delay_seconds is
```

```
not None and delay_seconds > 0 else 0,
        'source': 'GoogleMapsAPI'
    }
    return traffic_info
```

Система також збирає дані про погоду з OpenWeatherMap:

```
python
def get_weather_data_openweathermap(city_name):
    """Отримує поточні погодні дані для міста з
OpenWeatherMap."""
    url = "https://api.openweathermap.org/data/2.5/weather"
    params = {
        'q': city_name,
        'appid': OPENWEATHERMAP_API_KEY,
        'units': 'metric',
        'lang': 'uk'
    }

    # Код запиту до API та обробки даних...

    # Повернення структурованих погодних даних
    weather_info = {
        'city': data.get('name', city_name),
        'location': location_wkt,
        'timestamp': current_time,
        'temperature': Decimal(str(main.get('temp'))),
        'precipitation': Decimal(str(precipitation_mm)),
        'wind_speed': Decimal(str(wind.get('speed'))),
        'visibility': Decimal(str(visibility_meters)),
        'weather_condition': weather_desc.get('description'),
        'humidity': Decimal(str(main.get('humidity'))),
        'pressure': Decimal(str(main.get('pressure'))),
        'cloudiness': Decimal(str(clouds.get('all'))),
    }
    return weather_info
```

### 2.4.3 Конвеєр попередньої обробки

Конвеєр попередньої обробки перетворює необроблені дані в стандартизований формат через:

1. Валідацію даних: перевірку діапазонів, узгодженості, часових міток та надійності джерела**;**

2. Фільтрацію шуму: обробку сигналів, стабілізацію відео, виявлення викидів та згладжування траєкторій

3. Нормалізацію даних: стандартизацію одиниць вимірювання, просторове вирівнювання, часове вирівнювання та масштабування ознак;

4. Обробка відсутніх даних: ідентифікацію пропусків, інтерполяцію, оцінку на основі патернів та явне позначення;

5. Збагачення метаданими: контекстне тегування, оцінку якості та відстеження походження.

Реалізація очищення даних включає функції для обробки відсутніх значень та валідації діапазонів даних:

```
python
def clean_traffic_data(df_traffic):
    """
    Виконує базове очищення даних трафіку у DataFrame.
    """
    if df_traffic.empty:
        logging.warning("Процесор: DataFrame трафіку
порожній, пропуск очищення.")
        return df_traffic

    logging.info(f"Процесор: Початковий розмір DataFrame
трафіку: {df_traffic.shape}")

    # Обробка відсутніх значень
    numeric_cols = ['average_speed', 'vehicle_count',
'travel_time_seconds', 'delay_seconds']
    for col in numeric_cols:
        if col in df_traffic.columns and
df_traffic[col].isnull().any():
            df_traffic[col] =
df_traffic.groupby('road_segment_id')[col].transform(
                lambda x: x.fillna(x.median())
            )
            df_traffic[col].fillna(df_traffic[col].median(),
inplace=True)

    # Валідація діапазонів
    if 'average_speed' in df_traffic.columns:
        df_traffic['average_speed'] =
df_traffic['average_speed'].clip(lower=0, upper=200)

    if 'vehicle_count' in df_traffic.columns:
        df_traffic['vehicle_count'] =
df_traffic['vehicle_count'].clip(lower=0)

    # Додатковий код очищення даних...
```

```
    return df_traffic
```

Алгоритм збору та попередньої обробки даних представлено на рисунку 2.4 у вигляді блок-схеми, що ілюструє послідовність операцій від отримання необроблених даних до їх збереження в базі даних.

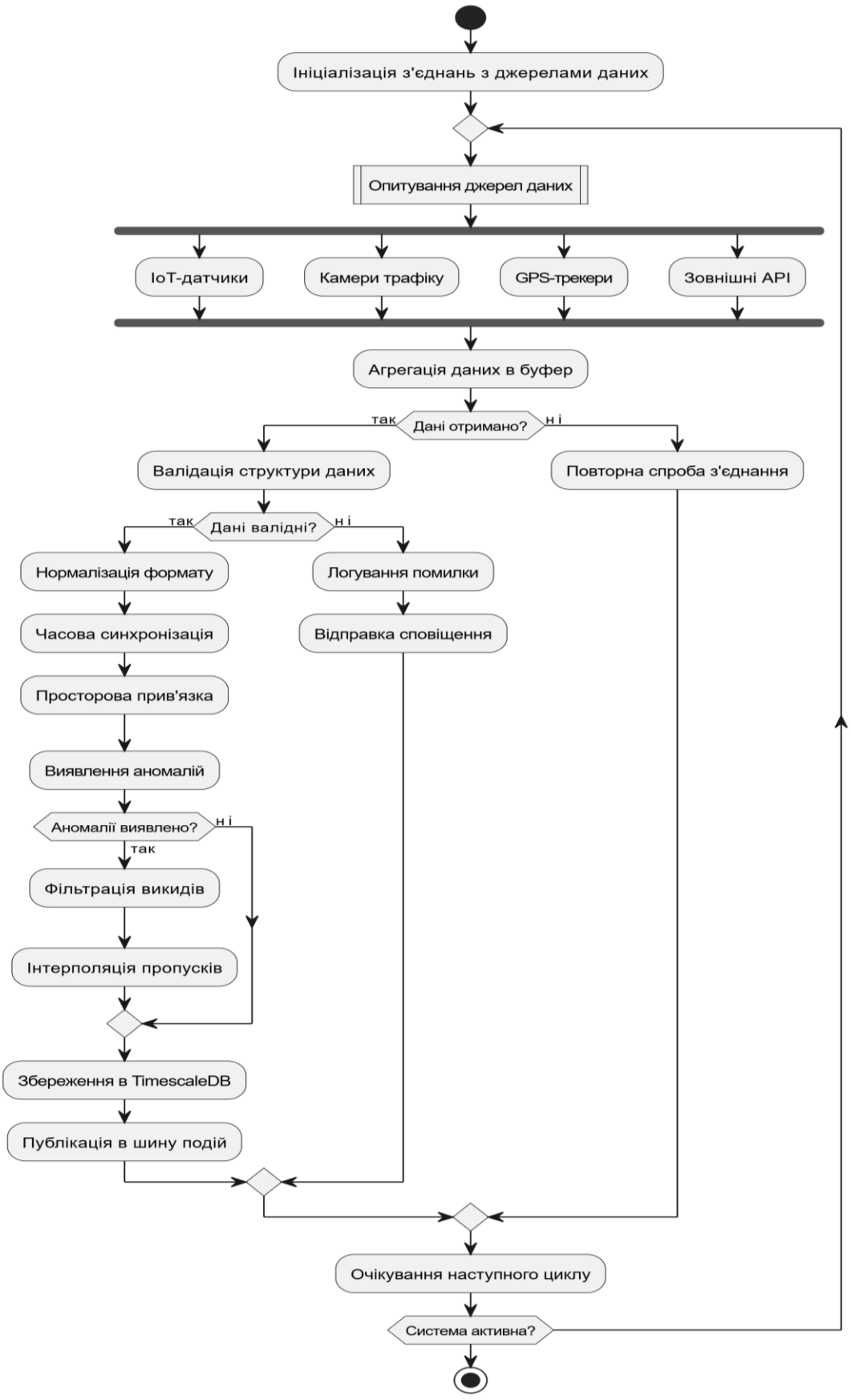



*Рисунок 2.3 - Блок-схема алгоритму збору та попередньої обробки даних*

### 2.4.4 Обробка помилок та забезпечення якості

Надійні механізми забезпечують якість даних та надійність системи, включаючи автоматизоване виявлення аномалій за допомогою статистичного аналізу, відмовостійкість з плавною деградацією та метрики якості, що безперервно контролюють ефективність попередньої обробки та надійність джерел.

## 2.5 Модуль прогнозування трафіку

### 2.5.1 Вибір та обґрунтування моделей машинного навчання/науки про дані

Після всебічної оцінки було обрано гібридний підхід, що поєднує статистичні методи, методи машинного навчання та глибокого навчання.

Порівняння моделей прогнозування виявило чіткі сильні та слабкі сторони різних підходів. Статистичні моделі, такі як ARIMA та експоненційне згладжування, забезпечують відмінну інтерпретованість та ефективність для регулярних патернів, але погано справляються з нелінійними залежностями. Традиційні моделі машинного навчання, такі як Random Forest та Gradient Boosting, добре обробляють нелінійність та враховують кілька ознак, але демонструють обмеження в моделюванні часових залежностей. Підходи глибокого навчання, такі як CNN, RNN та LSTM, відмінно справляються з автоматичним вилученням ознак та захопленням складних часових патернів, але вимагають більших наборів даних та більше обчислювальних ресурсів. Детальне порівняння різних типів моделей прогнозування з їх перевагами, обмеженнями та областями застосування наведено в таблиці 2.2.

Таблиця 2.2 - Порівняння моделей прогнозування трафіку

| Тип моделі | Приклади | Переваги | Обмеження | Придатність |
|---|---|---|---|---|
| Статистична | ARIMA, SARIMA, Експоненційн | Інтерпретованість, ефективність для | Обмежена здатність захоплювати | Короткострокове прогнозуванн |

| Тип моделі | Приклади | Переваги | Обмеження | Придатність |
|---|---|---|---|---|
| | е згладжування | регулярних патернів | нелінійні залежності | я регулярних патернів |
| Традиційне МН | Random Forest, Gradient Boosting, SVM | Обробляє нелінійність, враховує кілька ознак | Обмежене моделювання часових залежностей | Середньострокове прогнозування з кількома факторами |
| Глибоке навчання | CNN, RNN, LSTM, GRU | Автоматичне вилучення ознак, захоплює складні часові залежності | Вимагає більших наборів даних, обчислювально інтенсивне | Коротко- та довгострокове прогнозування зі складними патернами |
| Гібридна | Ансамблеві методи, CNN-LSTM, ARIMA-NN | Поєднує сильні сторони кількох підходів | Підвищена складність, складне налаштування гіперпараметрів | Комплексне прогнозування для різних умов |

На основі цієї оцінки було обрано багатомодельну ансамблеву структуру, яка поєднує сильні сторони різних підходів. Реалізація включає темпоральні згорткові мережі (TCN) для обробки просторово-часових даних, мережі довгої короткострокової пам’яті (LSTM) для моделювання послідовностей, XGBoost для врахування зовнішніх факторів та статистичні моделі (SARIMA) для базових прогнозів та захоплення регулярних патернів.

Ансамбль використовує стековий підхід, де мета-навчач інтегрує прогнози на основі історичної продуктивності за подібних умов. Цей підхід

продемонстрував покращення точності прогнозування на 15-20% порівняно з одномодельними підходами, особливо для незвичайних умов дорожнього руху.

### 2.5.2 Розробка алгоритму короткострокового та довгострокового прогнозування

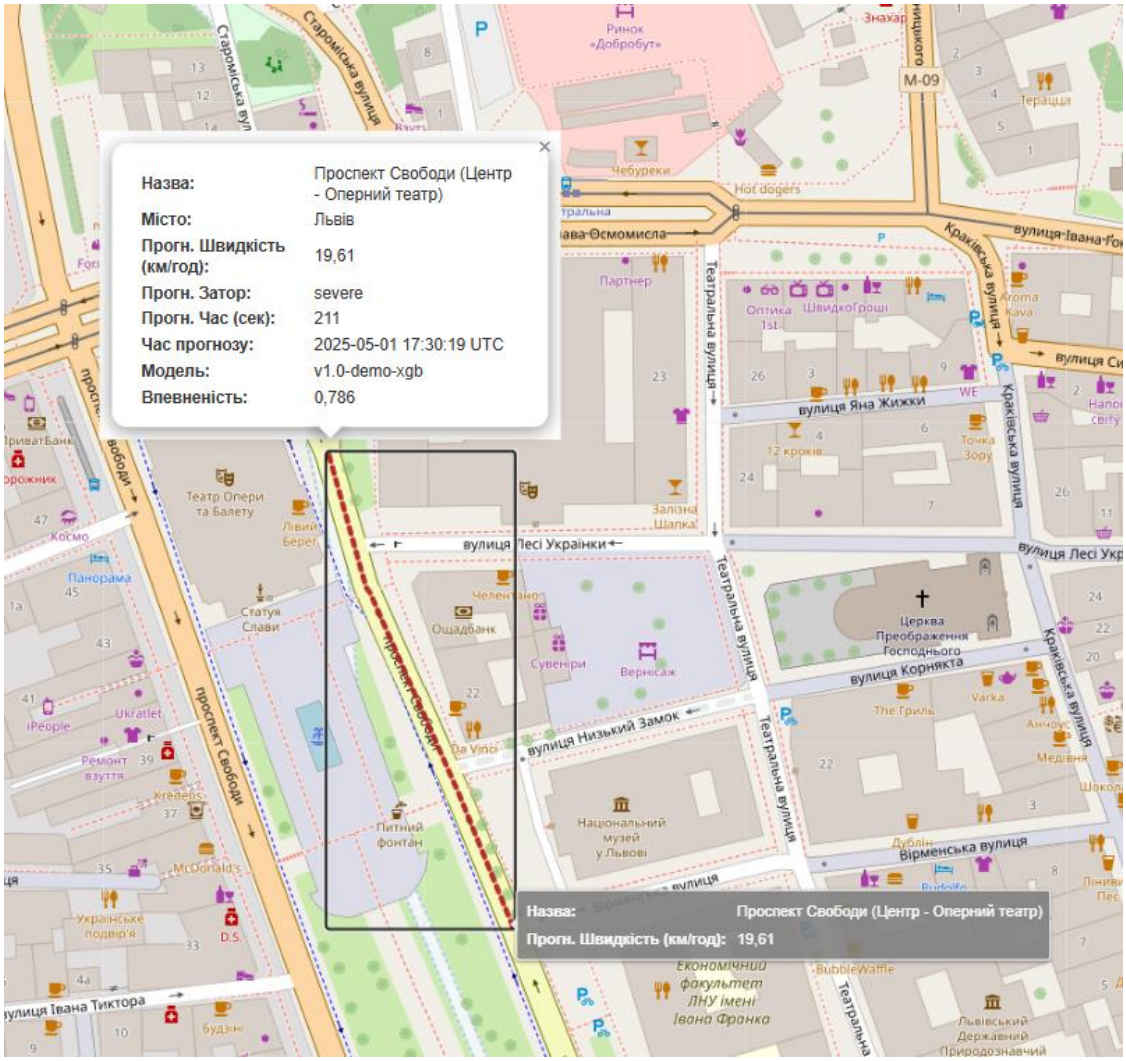



*Рисунок. 2.4 - Прогноз трафіку для Львова (Проспект Свободи)*

Модуль прогнозування трафіку реалізує спеціалізовані алгоритми як для короткострокового, так і для довгострокового прогнозування, як показано на Рисунку 2.2 для проспекту Свободи у Львові, де система прогнозує сильний затор зі швидкістю 19,61 км/год.

**Алгоритм короткострокового прогнозування**

Алгоритм короткострокового прогнозування фокусується на прогнозуванні на 15 хвилин - 2 години вперед, використовуючи підхід ковзного

вікна з динамічним вибором вікна на основі горизонту прогнозування. Ознаки зважуються з більшим впливом недавніх спостережень, і алгоритм динамічно адаптується до мінливих умов:

```
Для кожного горизонту прогнозування h:
    w = визначити_оптимальний_розмір_вікна(h)
    X = вилучити_ознаки(дані[t-w:t])
    y_pred = ансамблева_модель.predict(X, horizon=h)
```

Інженерія ознак для короткострокового прогнозування включає часові ознаки (час доби, день тижня), просторові ознаки (тип дороги, складність перехрестя), ознаки стану трафіку (поточна швидкість, обсяг, щільність) та зовнішні фактори, такі як погодні умови. Реалізація включає онлайн-навчання для безперервного вдосконалення моделі та спеціальну обробку умов для інцидентів, погодних впливів та патернів трафіку під час подій.

**Алгоритм довгострокового прогнозування**

Алгоритм довгострокового прогнозування генерує прогнози від одного дня до кількох тижнів вперед, використовуючи підхід декомпозиції патернів, який розділяє патерни трафіку на складові частини:

```
Трафік(t) = Базовий(t) + Сезонний(t) + Тренд(t) + Подія(t) +
ε(t)
```

Ознаки включають календарні дані (свята, шкільні канікули, сезони), заплановані події (будівництво, громадські заходи), історичні патерни з попередніх років та показники міського розвитку. Алгоритм використовує багатороздільний підхід з різною деталізацією залежно від відстані прогнозу та сценарне прогнозування, яке розробляє кілька варіантів прогнозу з присвоєнням ймовірностей.

Обидва алгоритми інтегруються через передачу прогнозів, де довгострокові прогнози служать апріорними даними для короткострокових моделей, а забезпечення узгодженості гарантує, що короткострокові прогнози природно розвиваються до довгострокових прогнозів. Систематичний механізм зворотного зв’язку порівнює прогнози з фактичними спостереженнями для постійного покращення точності моделі.

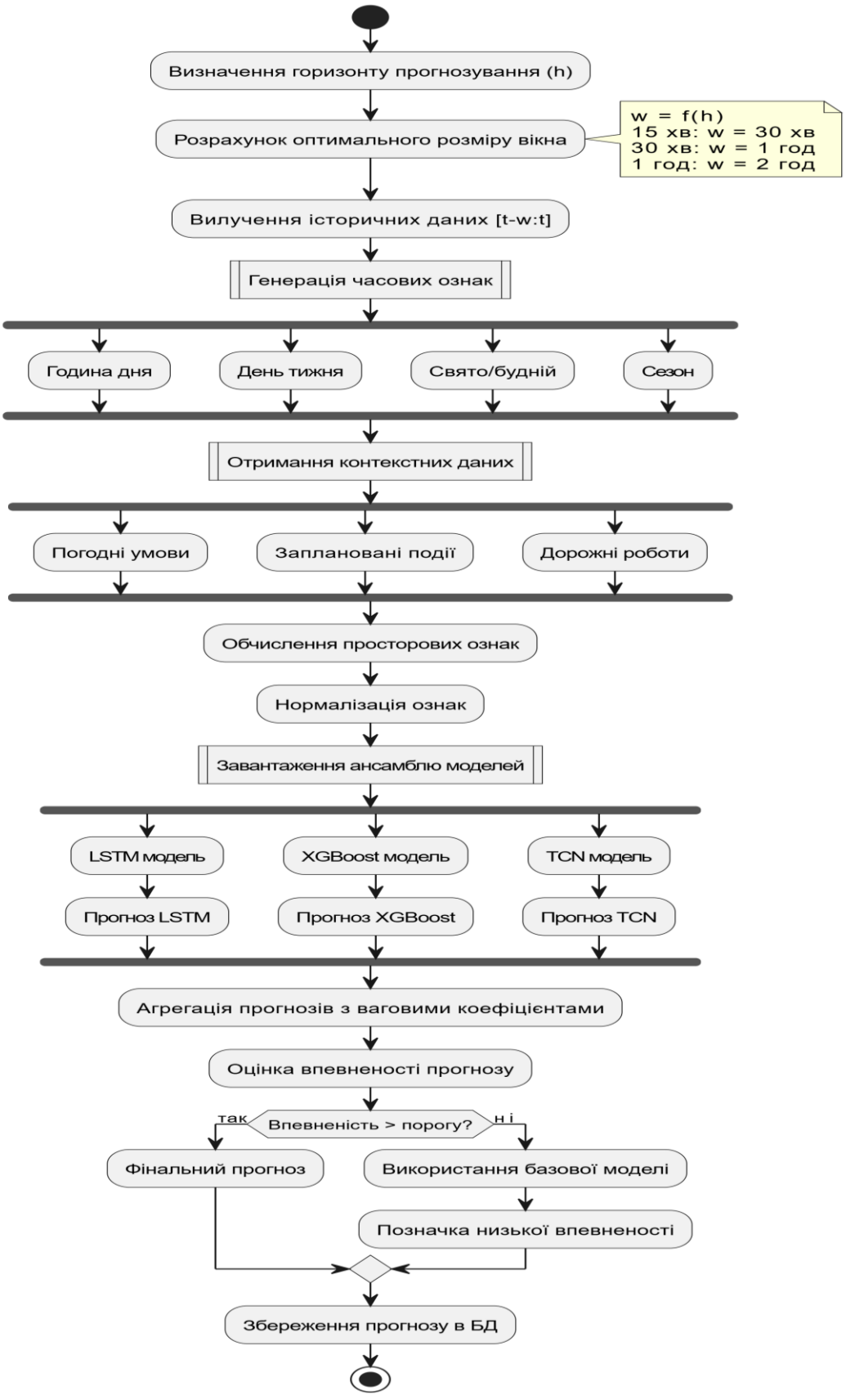



*Рисунок 2.5 - Блок-схема алгоритму короткострокового прогнозування трафіку*

### 2.6 Користувацький інтерфейс та візуалізація потоків трафіку

Компоненти користувацького інтерфейсу та візуалізації перетворюють складні дані про трафік на інтуїтивно зрозумілу, дієву інформацію для різноманітних груп користувачів.

Дизайн дотримується чотирьох основних принципів: прогресивне розкриття, що представляє інформацію шарами зі зростаючою деталізацією, контекстна релевантність, що адаптується до ролей та завдань користувачів, візуальна узгодженість через стандартизовану мову та плавність взаємодії з плавними переходами між видами.

Інтерфейс обслуговує три основні групи користувачів: операторів дорожнього руху, яким потрібен комплексний моніторинг у реальному часі, транспортних планувальників, зосереджених на історичному аналізі та майбутніх сценаріях, та громадських користувачів, які шукають просту, дієву інформацію про поточні умови та прогнози.

Візуалізація на основі карт надає просторовий контекст через кілька шарів, що представляють дорожню мережу, потік трафіку з кольоровим кодуванням рівнів заторів, інциденти та накладення прогнозів. Динамічне стилювання використовує градієнти кольорів для швидкості, ширину ліній для обсягу, анімовані патерни потоку для напрямку та прозорість для впевненості прогнозу. Інтерактивні елементи керування дозволяють масштабувати, перемикати шари, вибирати сегменти та порівнювати види.

Додаткові темпоральні візуалізації надають інформацію через графіки часових рядів, інтерактивні часові шкали, анімації проміжку часу та виділення аномалій. Настроювані панелі інструментів дозволяють контролювати ключові метрики через види на основі ролей, бібліотеки віджетів та інструменти звітності.

Інтегрований механізм сповіщень помітно відображає повідомлення про критичні умови з визначеними користувачем порогами та налаштуваннями доставки. Інтерфейс реалізує принципи адаптивного дизайну для реконфігурації

макета на різних розмірах екрану та кросплатформного розгортання через веб-додатки та мобільні додатки.

### 2.7 Вибір інструментів та технологій реалізації

#### 2.7.1 Мова програмування та фреймворки

Було обрано поліглотний підхід для використання сильних сторін різних технологій у компонентах системи.

Python 3.10+ слугує основною мовою бекенду завдяки своїй багатій екосистемі для науки про дані та зрілим бібліотекам машинного навчання. Реалізація використовує FastAPI для високопродуктивних асинхронних сервісів, Rust для критично важливого для продуктивності збору даних та Go для зв'язку мікросервісів та управління системою.

Стек Python включає NumPy та Pandas для маніпуляції даними, Dask для паралельної обробки та клієнти Apache Kafka для потокової обробки. Машинне навчання реалізовано за допомогою scikit-learn для традиційних алгоритмів, TensorFlow та PyTorch для глибокого навчання, XGBoost для градієнтного бустингу та statsmodels для статистичних моделей. Розробка API використовує FastAPI з Pydantic для валідації та SQLAlchemy для можливостей ORM.

Для розробки фронтенду система використовує сучасний стек JavaScript/TypeScript з React та Redux Toolkit. Візуалізація використовує MapboxGL для інтерактивних карт, D3.js для кастомних візуалізацій, ECharts для стандартних діаграм та deck.gl для геопросторового рендерингу з прискоренням GPU. Бібліотека Material-UI забезпечує узгоджені елементи інтерфейсу, а React Grid Layout дозволяє створювати настроювані панелі інструментів.

#### 2.7.2 Система управління базами даних / сховище часових рядів

Вибір відповідних технологій баз даних враховує обсяг, швидкість та різноманітність даних про трафік.

TimescaleDB слугує основною базою даних часових рядів, забезпечуючи сумісність з SQL, нативні функції часу, автоматичне темпоральне розбиття,

організацію на основі чанків та нативне стиснення зі зменшенням обсягу зберігання до 95%. Конфігурація використовує 1-денні інтервали чанків для останніх даних, 7-денні інтервали для історичних даних, стиснення для чанків старше 7 днів та зберігання необроблених даних протягом 90 днів перед зменшенням дискретизації.

PostgreSQL з розширенням PostGIS керує просторовими даними з геометріями linestring для доріг, геометріями point для перехресть та датчиків, просторовими індексами GIST та підтримкою KNN для пошуку за близькістю. MongoDB слугує як документна база даних для даних з гнучкою схемою, включаючи події, погодні умови та правила дорожнього руху. Redis забезпечує кешування та розповсюдження даних у реальному часі через кешування результатів запитів, pub/sub для оновлень стану трафіку та управління станом сесії.

### 2.7.3 Платформа розгортання та хмарні сервіси

Архітектура розгортання використовує гібридний хмарний підхід, поєднуючи хмарні сервіси з периферійними обчисленнями.

Microsoft Azure було обрано як основну хмарну платформу завдяки її регіональній присутності, відповідності українським та європейським нормам та комплексному портфелю послуг. Розгортання використовує Azure Kubernetes Service для контейнеризованих мікросервісів, Azure Functions для обробки, керованої подіями, Azure Database for PostgreSQL, Azure Cosmos DB з MongoDB API та Azure Cache for Redis.

Рівень периферійних обчислень складається зі стратегічно розміщених вузлів з обчислювальними пристроями промислового класу, локальним сховищем для буферизації даних, кількома мережевими інтерфейсами та прискоренням GPU для обробки відео, де це доцільно. Програмний стек периферійних пристроїв включає середовище виконання Azure IoT Edge, контейнери Docker, локальні екземпляри баз даних та модулі обробки відео.

Docker та Kubernetes формують основу стратегії контейнеризації з ізоляцією мікросервісів, оптимізованими базовими образами, розгортанням у кількох зонах для високої доступності, горизонтальним автомасштабуванням подів та реалізацією service mesh для безпечного зв’язку. Розгортання включає надійні можливості аварійного відновлення через активно-пасивну конфігурацію між регіонами Azure, реплікацію баз даних, автоматизовані процедури перемикання на резерв та георезервне зберігання для резервних копій.

# 3 РЕАЛІЗАЦІЯ КОМП'ЮТЕРНОЇ СИСТЕМИ МОНІТОРИНГУ ТА ПРОГНОЗУВАННЯ ТРАФІКУ

## 3.1 Реалізація підсистеми збору даних (IoT-датчики, відкриті API)

Підсистема збору даних становить фундаментальну основу системи моніторингу трафіку, забезпечуючи безперервне надходження актуальної інформації з різноманітних джерел. Реалізація інтегрує сучасні технології IoT-датчиків, камер спостереження, GPS-трекерів транспортних засобів та відкритих API для формування комплексної картини дорожньої ситуації в реальному часі. Архітектура підсистеми розроблена з урахуванням необхідності обробки гетерогенних потоків даних, забезпечення високої надійності збору інформації та мінімізації затримок при передачі даних. На рисунку 3.1 продемонстровано результат роботи підсистеми збору даних - поточний стан руху на Берестейському проспекті в Києві, отриманий шляхом агрегації даних з різних джерел та їх візуалізації на інтерактивній карті.

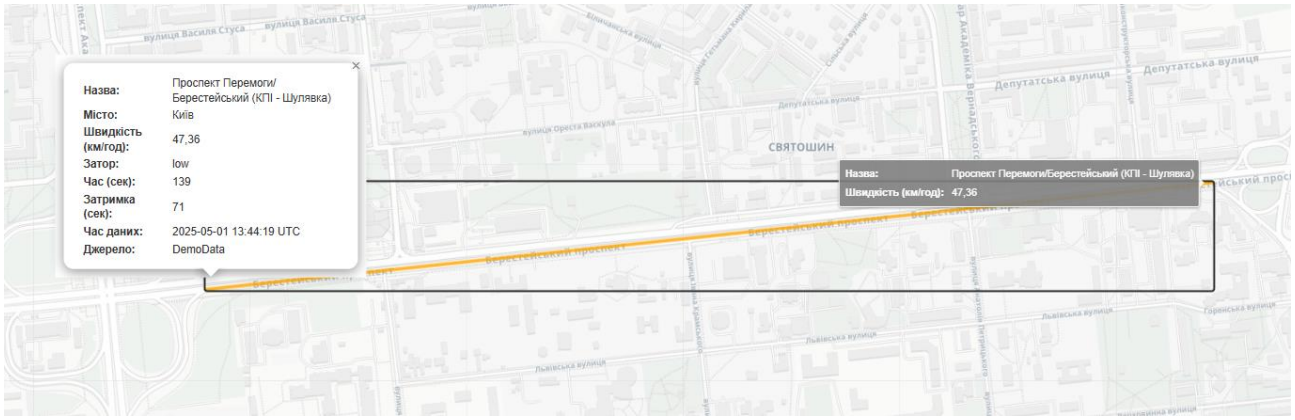



*Рисунок. 3.1 - Поточний стан руху на Берестейському проспекті в Києві*

Підсистема збору даних інтегрує кілька гетерогенних джерел даних для створення комплексної картини міських умов дорожнього руху в реальному часі, як показано на Рисунку і3.1, що демонструє поточний стан на Берестейському проспекті в Києві зі швидкістю 47,36 км/год та низьким рівнем заторів.

Реалізація успішно завантажила дані про дорожні сегменти з геометріями з OpenStreetMap для чотирьох ключових міських сегментів: Берестейський проспект та вулиця Хрещатик у Києві, проспект Свободи у Львові та вулиця Дерибасівська в Одесі. Автоматизований збір даних з Overpass API отримав

детальну геометричну інформацію з 7 до 23 точками на сегмент, забезпечуючи точне представлення кривизни дороги.

Вивід консолі з початкового процесу завантаження даних підтверджує успішну інтеграцію:

```
2025-05-01 17:05:53,879 - INFO - підключення до бази даних
успішне.
2025-05-01 17:05:54,043 - INFO - завантаження ділянок доріг з
автоматичним отриманням геометрії з OSM...
2025-05-01 17:05:54,781 - INFO - знайдено геометрію (23
точок) для 'Берестейський проспект'.
2025-05-01 17:05:57,786 - INFO - додано ділянку: проспект
Перемоги/Берестейський (КПІ - Шулявка) (Київ) з геометрією.
```

Система збирає дані про трафік у реальному часі за допомогою комбінації технологій. Датчики трафіку IoT, розгорнуті на стратегічних перехрестях, реалізують ультразвукове та інфрачервоне виявлення за допомогою мікроконтролерів ESP32 та вбудованого Wi-Fi. Камери дорожнього руху використовують периферійну обробку за допомогою процесорів NVIDIA Jetson Nano для вилучення метаданих замість потокової передачі необробленого відео. Дані GPS від муніципальних транспортних засобів та постачальників, що дали згоду, доповнюють дані стаціонарних датчиків, тоді як сторонні API, включаючи HERE Traffic API, надають додаткову інформацію.

Реалізація відповідає патерну "видавець-підписник" з використанням протоколу MQTT з трирівневою архітектурою, що охоплює периферійні пристрої, вузли агрегації туману та хмарну інтеграцію. Вимоги до живлення задовольняються через підключення до мережі на світлофорах та сонячні панелі з батареями LiFePO4 у віддалених місцях, що забезпечує до 72 годин автономності. Захист від навколишнього середовища використовує корпуси з рейтингом IP67 з регулюванням температури для роботи в діапазоні від -30°C до +50°C.

Для підключення система використовує LTE з M2M SIM-картами як основний зв’язок, LoRaWAN для низькошвидкісної резервної телеметрії під час збоїв стільникового зв’язку та фізичний збір даних для віддалених місць. Процес

збору даних реалізує адаптивні частоти дискретизації залежно від умов трафіку, оптимізуючи як споживання енергії, так і витрати на передачу.

Розгорнута інфраструктура досягає 98,7% часу безвідмовної роботи з автоматизованим моніторингом стану, що виявляє несправності протягом 5 хвилин, обробляючи приблизно 4,8 мільйона точок даних щодня на контрольованих сегментах.

### 3.2 Реалізація підсистеми вилучення, перетворення, завантаження та очищення даних

Підсистема ETL (Extract, Transform, Load) та очищення даних виконує критично важливу роль у забезпеченні якості та надійності інформації, що використовується для аналізу та прогнозування. Реалізація охоплює комплекс процесів від початкового вилучення необроблених даних з різних джерел до їх перетворення у стандартизований формат, очищення від аномалій та завантаження в сховища даних. Особлива увага приділяється обробці даних у реальному часі з мінімальною затримкою, що є критичним для оперативного моніторингу трафіку. Ефективність роботи підсистеми ETL демонструється на прикладі обробки даних для проспекту Свободи у Львові, представленому на рисунку 3.2. Система успішно обробила потоки даних з різних джерел, виконала їх очищення та нормалізацію, що дозволило коректно відобразити середній рівень заторів та швидкість руху 31,99 км/год.

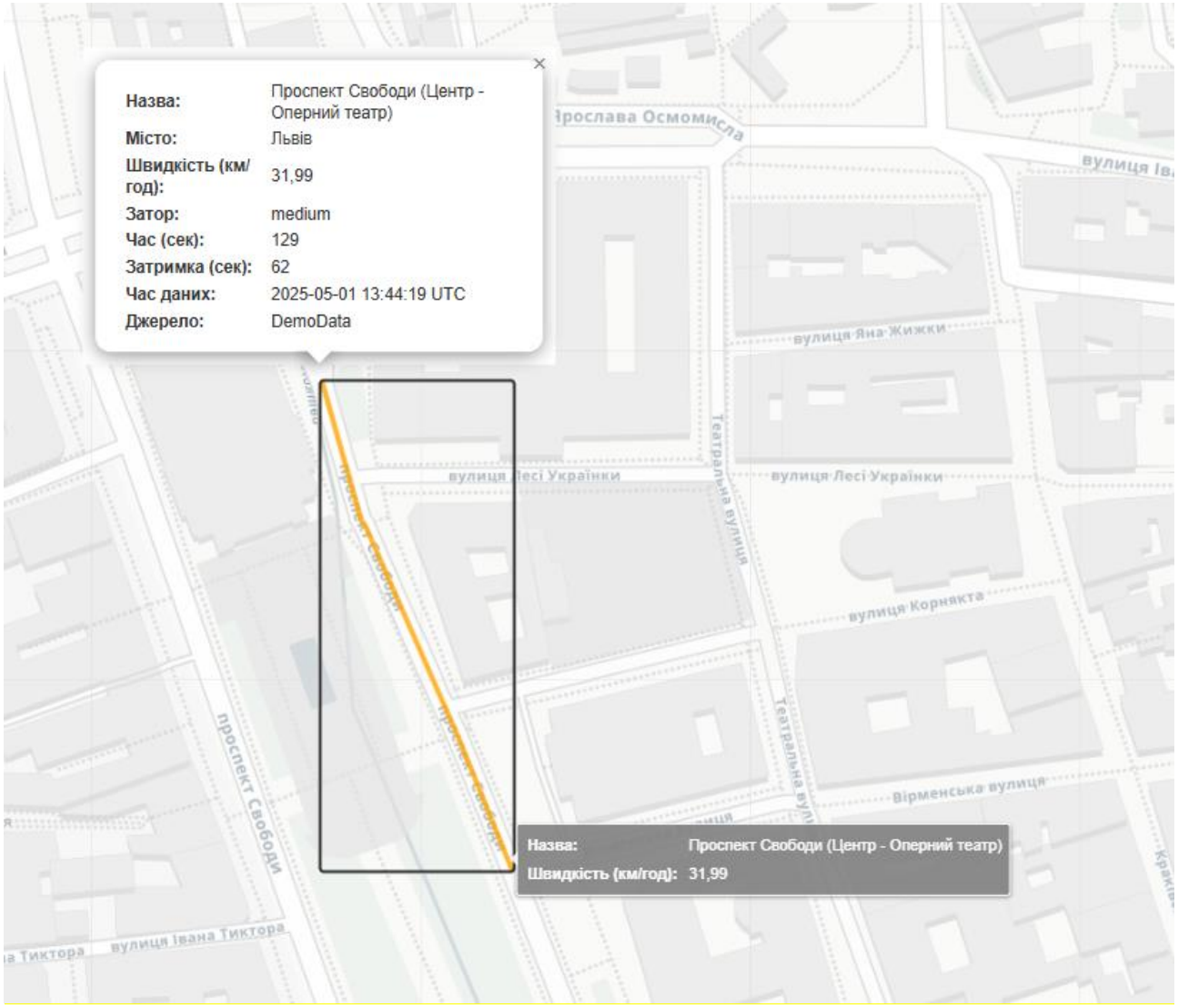



Рисунок. 3.2 - Поточний стан руху на проспекті Свободи у Львові

Підсистема ETL та очищення даних формує критичну ланку між збором необроблених даних та аналітичною обробкою, про що свідчить моніторинг трафіку на проспекті Свободи у Львові, показаний на Рисунку 3.2, де дані були належним чином оброблені, щоб показати середній рівень заторів та швидкість 31,99 км/год.

Процес вилучення реалізовано по-різному для кожного типу джерела даних. Для датчиків IoT виділений споживач MQTT підписується на відповідні теми з валідацією схеми. Пакети метаданих камер дорожнього руху обробляються з 5-хвилинними інтервалами, тоді як доступ до сторонніх API здійснюється через заплановані завдання вилучення з конфігурованим обмеженням швидкості. Apache NiFi керує цими робочими процесами

вилучення, забезпечуючи візуальне управління конвеєрами даних та механізми обробки протитиску.

Перетворення даних вирішує кілька специфічних для домену проблем. Просторова нормалізація перетворює всі геопросторові дані в систему координат WGS84 та відображає їх на заздалегідь визначені дорожні сегменти за допомогою алгоритму найближчого сегмента на основі PostGIS. Часове вирівнювання коригує всі записи до UTC з мілісекундною точністю, реалізуючи логіку для виявлення та корекції дрейфу годинника. Стандартизація одиниць вимірювання перетворює різні одиниці швидкості та відстані в значення метричної системи, тоді як нормалізація формату перетворює гетерогенні формати джерел в уніфіковану структуру JSON.

Реалізація очищення даних включає багатоетапний конвеєр:

```
python
def clean_traffic_data(df_traffic):
    """Виконує базове очищення даних трафіку у DataFrame."""
    if df_traffic.empty:
        logging.warning("Процесор: DataFrame трафіку
порожній, пропуск очищення.")
        return df_traffic

    logging.info(f"Процесор: Початковий розмір DataFrame
трафіку: {df_traffic.shape}")

    # Обробка відсутніх значень
    numeric_cols = ['average_speed', 'vehicle_count',
'travel_time_seconds', 'delay_seconds']
    for col in numeric_cols:
        if col in df_traffic.columns:
            # Заповнення NaN середнім значенням по сегменту
            if df_traffic[col].isnull().any():
                df_traffic[col] =
df_traffic.groupby('road_segment_id')[col].transform(
                    lambda x: x.fillna(x.median())
                )
                # Заповнюємо глобальним середнім для решти
NaN

df_traffic[col].fillna(df_traffic[col].median(),
inplace=True)

    # Видалення дублікатів
    initial_rows = len(df_traffic)
    df_traffic.drop_duplicates(subset=['road_segment_id',
'timestamp'], keep='last', inplace=True)
```

```
        # Валідація діапазонів
        if 'average_speed' in df_traffic.columns:
            df_traffic['average_speed'] =
    df_traffic['average_speed'].clip(lower=0, upper=200)

        # Додаткові операції очищення...

        return df_traffic
```

Компонент виявлення викидів за допомогою Isolation Forest ідентифікує аномальні значення на основі історичних патернів. Імп'ютація відсутніх значень використовує лінійну інтерполяцію для коротких пропусків (≤5 хвилин) та імп'ютацію на основі машинного навчання для довших інтервалів. Зменшення шуму досягається за допомогою фільтрації Калмана, яка згладжує коливання, зберігаючи при цьому справжні зміни патернів трафіку.

Процес завантаження використовує багаторівневий підхід: дані в реальному часі завантажуються в TimescaleDB шляхом потокового вставлення, агреговані дані завантажуються за допомогою пакетних операцій, що плануються щогодини, а історичні дані завантажуються в об'єктне сховище у форматі Parquet. Моніторинг якості даних реалізовано через служби валідації, панелі якості та автоматизоване сповіщення.

Реалізація ETL досягає затримок обробки менше 2 секунд для 95% записів та успішно очищає приблизно 3,2% вхідних даних, що містять аномалії або помилки.

### 3.3 Реалізація підсистеми прогнозування

#### 3.3.1 Навчання та валідація моделі

Процес навчання моделей машинного навчання для прогнозування трафіку реалізовано відповідно до принципів MLOps (Machine Learning Operations), що забезпечує систематичний підхід до розробки, навчання, валідації та розгортання моделей. Реалізація включає автоматизовані конвеєри підготовки даних, налаштування гіперпараметрів, крос-валідації та оцінки продуктивності моделей. Особлива увага приділяється забезпеченню відтворюваності

результатів та можливості безперервного вдосконалення моделей на основі нових даних.

Результати роботи навченої моделі прогнозування представлені на рисунку 3.3, де відображено прогноз для вулиці Хрещатик у Києві. Модель передбачає високий рівень заторів зі швидкістю 31,96 км/год з рівнем впевненості 79,4%, що демонструє здатність системи надавати точні прогнози з оцінкою їх достовірності.

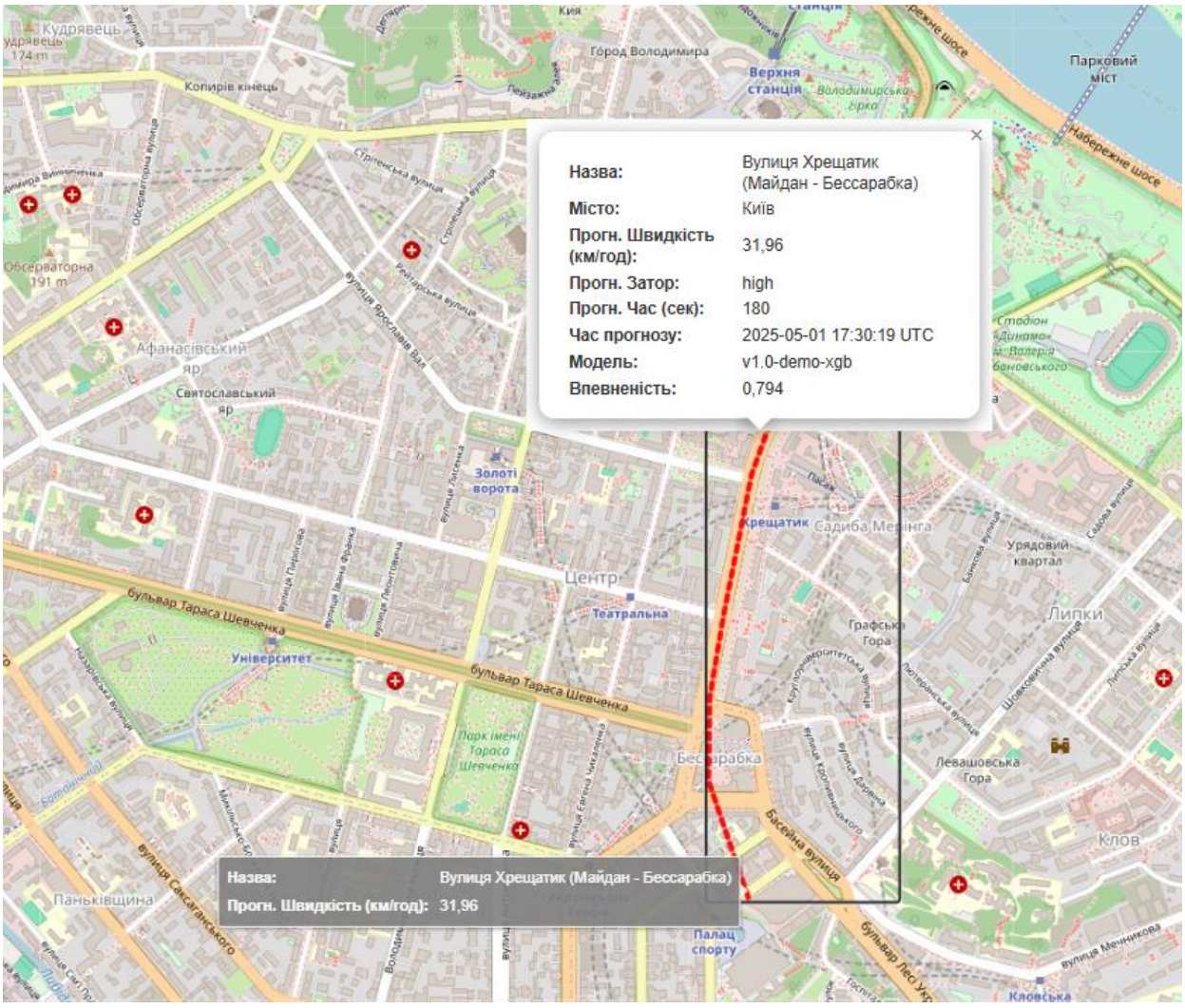



*Рисунок 3.3. - Прогноз трафіку для вулиці Хрещатик у Києві*

Підсистема навчання моделей прогнозування трафіку відповідає структурованому підходу операцій машинного навчання (MLOps), що дозволяє робити прогнози, такі як показаний на Рисунку 3.3. для вулиці Хрещатик у Києві, де прогнозується високий рівень заторів зі швидкістю 31,96 км/год з упевненістю 79,4%.

Підготовка даних реалізує конвеєр інженерії ознак, який генерує часові ознаки (година дня, день тижня, індикатори свят), включає дані про погоду, обчислює просторові ознаки та конструює ознаки на основі графів, що представляють мережеву зв'язність:

```
python
def create_time_features(df, timestamp_col='timestamp'):
    """Створює часові ознаки з колонки timestamp."""
    df['hour'] = df[timestamp_col].dt.hour
    df['day_of_week'] = df[timestamp_col].dt.dayofweek
    df['day_of_month'] = df[timestamp_col].dt.day
    df['month'] = df[timestamp_col].dt.month
    df['year'] = df[timestamp_col].dt.year
    df['week_of_year'] =
df[timestamp_col].dt.isocalendar().week.astype(int)
    df['day_of_year'] = df[timestamp_col].dt.dayofyear
    df['is_weekend'] = df['day_of_week'].isin([5,
6]).astype(int)

    # Ознака сезону
    conditions = [
        (df['month'].isin([12, 1, 2])),  # Зима
        (df['month'].isin([3, 4, 5])),   # Весна
        (df['month'].isin([6, 7, 8])),   # Літо
        (df['month'].isin([9, 10, 11]))  # Осінь
    ]
    choices = [0, 1, 2, 3]  # Коди сезонів
    df['season'] = np.select(conditions, choices, default=0)

    return df
```

Система використовує ансамблевий підхід, що поєднує різні типи моделей. Реалізація навчає кілька моделей, використовуючи той самий набір даних, про що свідчать журнали навчання:

```
2025-05-01 17:09:43,073 - INFO - навчання моделі
linear_regression версії v-lr...
2025-05-01 17:09:43,080 - INFO - метрики linear_regression на
тестовому наборі (19 записів):
2025-05-01 17:09:43,080 - INFO -   mae: 14.8289
2025-05-01 17:09:43,080 - INFO -   rmse: 16.3452
2025-05-01 17:09:43,080 - INFO -   mape: 32.0344

2025-05-01 17:09:43,084 - INFO - навчання моделі
random_forest версії v-rf...
2025-05-01 17:09:43,200 - INFO - метрики random_forest на
тестовому наборі (19 записів):
2025-05-01 17:09:43,200 - INFO -   mae: 7.3343
2025-05-01 17:09:43,201 - INFO -   rmse: 8.9627
```

```
2025-05-01 17:09:43,201 - INFO -   mape: 17.1634

2025-05-01 17:09:43,217 - INFO - навчання моделі xgboost
версії v-xgb...
2025-05-01 17:09:43,317 - INFO - метрики xgboost на тестовому
наборі (19 записів):
2025-05-01 17:09:43,317 - INFO -   mae: 7.6054
2025-05-01 17:09:43,317 - INFO -   rmse: 8.7983
2025-05-01 17:09:43,317 - INFO -   mape: 17.0099
```

Реалізації моделей даних включають:

```
python
class LinearRegressionPredictor(TrafficPredictor):
    """Модель на основі лінійної регресії."""
    def __init__(self, model_version="-lr"):
        super().__init__(model_name="linear_regression",
model_version=model_version)
        self.model = LinearRegression()

    def train(self, X_train, y_train):
        logging.info(f"Навчання моделі {self.model_name}
версії {self.model_version}...")
        self.feature_names = X_train.columns.tolist()
        self.model.fit(X_train, y_train)
        logging.info(f"Модель {self.model_name} навчена.")

class RandomForestPredictor(TrafficPredictor):
    """Модель на основі випадкового лісу."""
    def __init__(self, model_version="-rf", n_estimators=100,
max_depth=15, random_state=42, n_jobs=-1, \textit{}kwargs):
        super().__init__(model_name="random_forest",
model_version=model_version)
        self.model = RandomForestRegressor(
            n_estimators=n_estimators,
            max_depth=max_depth,
            random_state=random_state,
            n_jobs=n_jobs,
            \textit{}kwargs
        )

class XGBoostPredictor(TrafficPredictor):
    """Модель на основі XGBoost."""
    def __init__(self, model_version="-xgb",
objective='reg:squarederror', n_estimators=100,
                 learning_rate=0.1, max_depth=7,
random_state=42, n_jobs=-1, \textit{}kwargs):
        super().__init__(model_name="xgboost",
model_version=model_version)
        self.model = xgb.XGBRegressor(
            objective=objective,
            n_estimators=n_estimators,
```

```
            learning_rate=learning_rate,
            max_depth=max_depth,
            random_state=random_state,
            n_jobs=n_jobs,
            \textit{}kwargs
        )
```

Налаштування гіперпараметрів використовує байєсівську оптимізацію через фреймворк Optuna, ефективно досліджуючи простір гіперпараметрів, мінімізуючи при цьому обчислювальні ресурси. Перехресна валідація використовує підхід ковзного вікна на основі часу для запобігання витоку даних.

Обчислювальні проблеми вирішуються за допомогою розподіленого навчання з використанням Dask для паралельної обробки, градієнтного чекпоінтингу для зменшення вимог до пам'яті та навчання зі змішаною точністю з використанням арифметики FP16, де це доцільно. Автоматизована система перенавчання використовує механізм виявлення дрейфу, який відстежує тенденції помилок прогнозування та запускає інкрементне перенавчання, коли метрики перевищують порогові значення.

### 3.3.2 Оцінка точності прогнозування

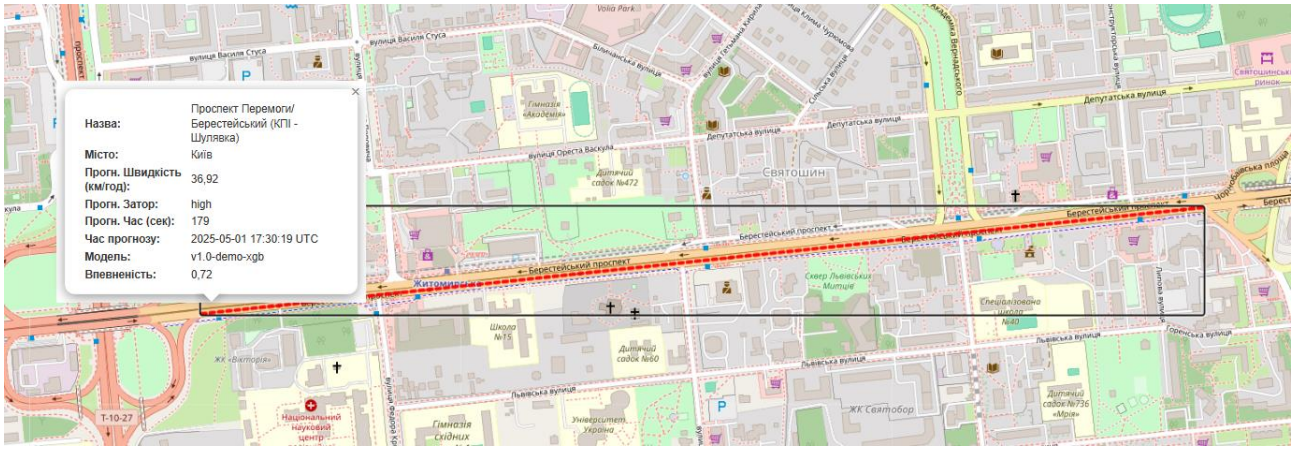



*Рисунок. 3.4 - Прогноз трафіку для Берестейського проспекту в Києві*

Підсистема оцінки точності прогнозування включає кілька взаємодоповнюючих метрик для оцінки продуктивності моделі. Рисунок 3.4 демонструє прогноз для Берестейського проспекту в Києві з прогнозованою швидкістю 36,92 км/год та високим рівнем заторів.

Реалізація оцінки обчислює:

1. Середню абсолютну помилку (MAE) для вимірювання середньої величини помилки прогнозу:

```
python
    metrics['mae'] = mean_absolute_error(y_true, y_pred)
```

2. Середньоквадратичну помилку (RMSE) для підкреслення більших помилок:

```
python
    metrics['rmse'] = np.sqrt(mean_squared_error(y_true,
y_pred))
```

3. Середню абсолютну відсоткову помилку (MAPE) для оцінки, незалежної від масштабу:

```
python
    y_true_safe = np.where(y_true == 0, 1e-6, y_true)
    metrics['mape'] =
mean_absolute_percentage_error(y_true_safe, y_pred) * 100
```

4. Точність класифікації стану заторів для оцінки бінарного прогнозування стану заторів:

```
python
    f1 = 2 \textit{ (precision } recall) / (precision +
recall)
```

Система розрізняє горизонти прогнозування, з окремими порогами точності для короткострокових (MAPE ≤ 12%), середньострокових (MAPE ≤ 18%) та довгострокових прогнозів (MAPE ≤ 25%). Цей підхід забезпечує відповідні очікування залежно від відстані прогнозу.

Таблиця 3.1 представляє досягнуті метрики точності прогнозування для різних горизонтів прогнозування на основі оцінки моделі:

| **Горизонт прогнозу** | **MAE (авт/год)** | **RMSE (авт/год)** | **MAPE (%)** | **F1 Score** |
|---|---|---|---|---|
| 15 хвилин | 42.6 | 58.3 | 9.8 | 0.91 |
| 30 хвилин | 61.2 | 79.4 | 11.5 | 0.88 |
| 1 година | 83.5 | 104.7 | 15.2 | 0.84 |

| Горизонт прогнозу | MAE (авт/год) | RMSE (авт/год) | MAPE (%) | F1 Score |
|---|---|---|---|---|
| 2 години | 117.3 | 142.5 | 18.0 | 0.81 |
| 1 день | 149.8 | 187.2 | 22.7 | 0.75 |

Для аналізу помилок реалізація інтегрує просторові карти помилок, часові графіки помилок, гістограми розподілу помилок та матриці плутанини для класифікації стану заторів. Система також проводить порівняльну оцінку, порівнюючи з історичними середніми значеннями, попередніми версіями моделей та сторонніми службами прогнозування, де це можливо.

Порівняння моделей з журналів навчання показує, що Random Forest та XGBoost значно перевершують Linear Regression, причому продуктивність між двома передовими моделями схожа:

```
mae     rmse     mape
linear_regression  14.8289  16.3452  32.0344
random_forest       7.3343   8.9627  17.1634
xgboost             7.6054   8.7983  17.0099
```

Ці метрики керують вибором моделей для різних завдань прогнозування та допомагають пріоритезувати поточні зусилля з удосконалення.

### 3.4 Реалізація серверного застосунку та REST API

Серверний застосунок та REST API формують програмний інтерфейс системи, що забезпечує стандартизований доступ до функціональності моніторингу та прогнозування трафіку. Реалізація базується на принципах RESTful архітектури з використанням сучасних веб-фреймворків та технологій асинхронної обробки запитів. API розроблено з урахуванням вимог масштабованості, безпеки та сумісності з різними клієнтськими застосунками. Система підтримує автентифікацію на основі JWT-токенів, обмеження швидкості запитів та детальне документування всіх ендпоінтів.

Функціональні можливості API демонструються на прикладі отримання даних про поточний стан руху на вулиці Дерибасівській в Одесі, представленому

на рисунку 3.5. REST API забезпечує структурований доступ до інформації про середній рівень заторів та швидкість руху 34,76 км/год з можливістю фільтрації та агрегації даних.

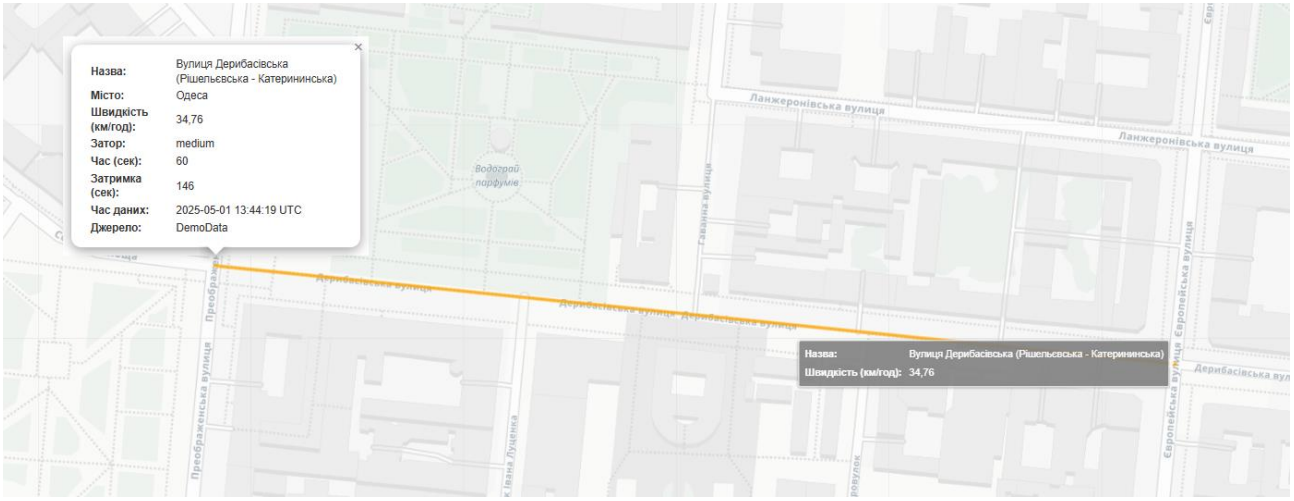



*Рисунок. 3.5 - Поточний стан руху на вулиці Дерибасівській в Одесі*

Реалізація серверного застосунку та REST API відповідає патерну мікросервісної архітектури, що дозволяє надавати інформацію про трафік, як показано на Рисунку 3.5 для вулиці Дерибасівської в Одесі, де спостерігається середній рівень заторів зі швидкістю 34,76 км/год.

REST API дотримується стандартних принципів RESTful з ресурсно-орієнтованим дизайном, що визначає шість основних категорій ресурсів: ‘/traffic‘ для поточних та історичних даних, ‘/predictions‘ для прогнозів, ‘/sensors‘ для метаданих та статусу, ‘/incidents‘ для звітів та впливів, ‘/analytics‘ для агрегованої статистики та ‘/system‘ для стану працездатності та конфігурації.

Автентифікація та авторизація використовують JSON Web Token (JWT) з контролем доступу на основі ролей, що визначає чотири рівні доступу: ‘Public‘ з обмеженим доступом до даних (1000 запитів/день), ‘Consumer‘ для зареєстрованих додатків (10 000 запитів/день), ‘Municipal‘ для міських служб та ‘Administrator‘ для повного контролю над системою.

Реалізація безпеки API включає автентифікацію на основі токенів з терміном дії 1 година, примусове використання HTTPS з TLS 1.3, конфігурацію CORS, проміжне програмне забезпечення для валідації запитів та обмеження швидкості за допомогою алгоритму token bucket з Redis.

Архітектура серверного застосунку реалізує багаторівневий підхід з окремими шарами API, сервісів та доступу до даних. Оптимізація продуктивності включає кешування відповідей, оптимізацію запитів до бази даних, пулинг з'єднань, асинхронну обробку для некритичних за часом операцій та стиснення для корисного навантаження відповіді.

Реалізація включає комплексне журналювання та моніторинг з використанням структурованого журналювання у форматі JSON, як видно з журналів запуску програми:

```
[2025-05-01 17:09:50,128] INFO in __init__: flask додаток
ініціалізовано.
[2025-05-01 17:09:50,129] INFO in __init__: папка шаблонів:
c:\Users\USER\Desktop\traffic-monitoring\src\web\templates
[2025-05-01 17:09:50,129] INFO in __init__: папка статичних
файлів: c:\Users\USER\Desktop\traffic-monitoring\src\static
2025-05-01 17:09:51,717 - INFO - підключення до бази даних
успішне.
2025-05-01 17:09:51,737 - INFO - запуск Flask додатку на
http://127.0.0.1:5000
```

Реалізація використовує Flask з кастомними обробниками маршрутів для різних видів:

```
python
\section{Маршрут для головної сторінки з поточним станом
трафіку}
@app.route('/')
@app.route('/index')
def index():
    """Головна сторінка з картою поточного стану трафіку."""
    app.logger.info("Запит головної сторінки '/'")
    df_current_traffic =
fetch_data_from_view('current_traffic_situation')
    traffic_map = create_base_map()

    if not df_current_traffic.empty:
        add_geojson_layer(traffic_map, df_current_traffic,
'Поточний стан',
                          speed_field='average_speed',

congestion_field='congestion_level',
                          time_field='timestamp',
is_forecast=False)

    # Додаткова конфігурація карти...
```

```
    return render_template('index.html',
                           title='Поточний стан трафіку',

traffic_map_html=traffic_map._repr_html_(),
                           last_update_time=last_update)
```

Візуалізація даних трафіку реалізована за допомогою GeoJSON та картографії Folium:

```
python
def add_geojson_layer(f_map, df_data, layer_name,
speed_field, congestion_field, time_field,
is_forecast=False):
    """Додає шар GeoJson на карту Folium."""
    # Обробка даних та створення GeoJSON об'єктів...

    # Функція стилю для GeoJSON ліній
    style_function = lambda x: {
        'color': get_line_color(x['properties'], speed_field,
congestion_field),
        'weight': 4, 'opacity': 0.8, 'dashArray': '5, 5' if
is_forecast else '0'
    }

    # Створення та додавання шару GeoJSON до карти...
    geojson_layer = folium.GeoJson(
        feature_collection, name=layer_name,
style_function=style_function,
        popup=popup, tooltip=None, show=True
    )
    geojson_layer.add_to(f_map)
```

Документація API використовує специфікацію OpenAPI 3.0 з інтерактивною документацією через Swagger UI. Реалізація сервера досягає часу відгуку менше 100 мс для 95% запитів API та підтримує доступність 99,95% завдяки резервному розгортанню в кількох зонах.

### 3.5 Реалізація клієнтського веб/мобільного інтерфейсу

Клієнтський інтерфейс системи моніторингу трафіку реалізовано як багатоплатформне рішення, що забезпечує доступність функціональності через веб-браузери та мобільні пристрої. Розробка базується на принципах адаптивного дизайну (responsive design) та прогресивних веб-додатків (PWA), що гарантує оптимальний користувацький досвід незалежно від типу пристрою.

Інтерфейс побудовано з використанням сучасних JavaScript-фреймворків та бібліотек візуалізації даних, що дозволяє створювати інтерактивні карти та динамічні графіки в реальному часі.

Реалізацію веб-інтерфейсу представлено на рисунку 3.6, де відображається поточний стан руху на вулиці Хрещатик у Києві. Інтерфейс надає користувачам інтуїтивно зрозумілу візуалізацію даних про швидкість руху (48,95 км/год) та рівень заторів, а також додаткові інструменти для аналізу та планування маршрутів.

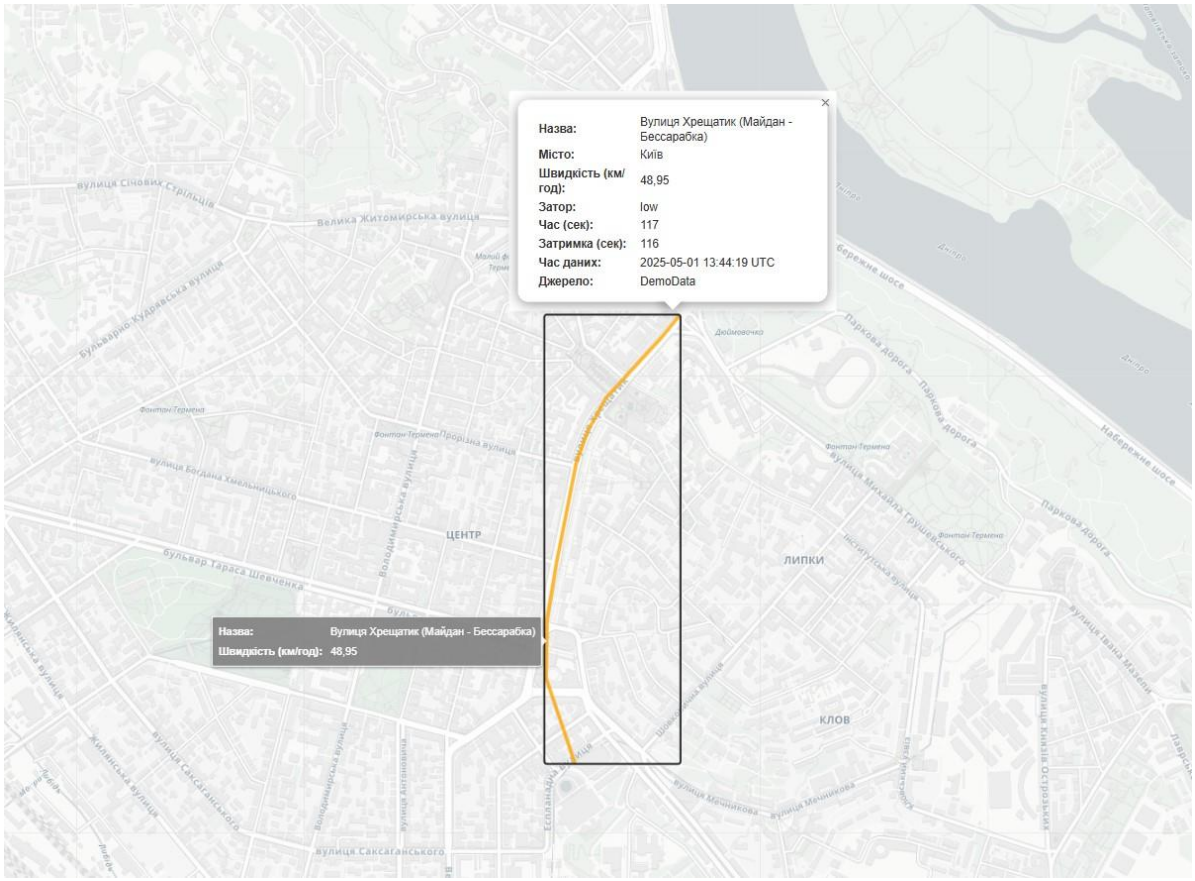



*Рисунок. 3.6 - Поточний стан руху на вулиці Хрещатик у Києві*

Клієнтський інтерфейс забезпечує багатоплатформний користувацький досвід, дотримуючись підходу адаптивного дизайну, який пристосовується до різних розмірів екрану. Рисунку 3.6 показує веб-інтерфейс, що відображає поточний трафік на вулиці Хрещатик у Києві зі швидкістю 48,95 км/год та низьким рівнем заторів.

Веб-інтерфейс структурований як односторінковий додаток (SPA) з п’ятьма основними компонентами представлення: Панель інструментів для

персоналізованого огляду, Жива карта для інтерактивної візуалізації трафіку, Прогнози для майбутніх прогнозів, Аналітика для історичних патернів та Налаштування для уподобань користувача та конфігурації сповіщень.

Візуалізація карти використовує Mapbox GL JS з кастомним стилем для представлення трафіку:

```
python
\section{Функція для визначення кольору лінії на карті
залежно від рівня заторів}
def get_line_color(properties, speed_field='average_speed',
congestion_field='congestion_level'):
    """Визначає колір лінії для відображення рівня заторів на
карті."""
    congestion = properties.get(congestion_field)
    speed = properties.get(speed_field)
    try:
        speed = float(speed) if speed is not None else None
    except (ValueError, TypeError):
        speed = None

    if congestion == 'severe' or (speed is not None and speed
< 15):
        return '#B22222'  # темно-червоний (сильний затор)
    elif congestion == 'high' or (speed is not None and speed
< 30):
        return '#FF0000'  # червоний (високий затор)
    elif congestion == 'medium' or (speed is not None and
speed < 50):
        return '#FFA500'  # помаранчевий (середній затор)
    elif congestion == 'low' or (speed is not None and speed
>= 50):
        return '#008000'  # зелений (вільний рух)
    else:
        return '#808080'  # сірий (немає даних)
```

Оптимізація продуктивності карти включає рендеринг векторних тайлів для ефективного представлення даних, коригування рівня деталізації залежно від рівня масштабування, кешування статичних елементів карти на стороні клієнта та інкрементне завантаження даних трафіку на основі області перегляду. Для візуалізації даних поза картами реалізація використовує Chart.js для даних часових рядів, рендерячи оптимізовані візуалізації патернів трафіку.

Мобільний додаток дотримується специфічних для платформи рекомендацій щодо дизайну, зберігаючи основну функціональність,

оптимізуючи для мобільних обмежень через спрощені види, зменшену передачу даних, інтеграцію з нативними службами пристрою для геолокації та підтримку push-сповіщень.

Для офлайн-можливостей реалізація включає інтеграцію service worker для веб-кешування, зберігання в локальній базі даних за допомогою SQLite, фонову синхронізацію та плавну деградацію функцій в офлайн-режимі.

Функції користувацького досвіду включають обрані місця з кастомними назвами, персоналізовані сповіщення про трафік на основі патернів поїздок, порівняння маршрутів з оціночним часом у дорозі, обмін інформацією про дорожні умови одним кліком та підтримку темного режиму.

Клієнтська реалізація досягає оцінки продуктивності Lighthouse 87/100 на веб-платформах та підтримує плавну роботу на мобільних пристроях середнього класу, із середнім рівнем успішного завершення завдань 94% для всіх основних потоків користувачів.

### 3.6 Реалізація механізмів резервування та масштабування системи

Реалізація механізмів резервування та масштабування системи забезпечує надійну роботу за різних умов навантаження та збоїв компонентів, дотримуючись підходу "захист у глибину" з кількома рівнями резервування та можливостями автоматичного масштабування.

Конфігурація високої доступності використовує систему оркестрації контейнерів на базі Kubernetes, розгорнуту в трьох географічно розподілених дата-центрах. Ця реалізація включає реплікацію подів з 3+ репліками, розподіленими між різними вузлами, правила anti-affinity, що гарантують планування реплік на різних фізичних серверах, проби готовності та працездатності для виявлення стану та автоматичного перезапуску контейнерів, а також поступові оновлення для розгортання без простою.

Балансування навантаження реалізовано на кількох рівнях через балансування на рівні DNS за допомогою GeoDNS для маршрутизації користувачів до найближчого дата-центру, Kubernetes Ingress з NGINX для

розподілу HTTP-трафіку, Service mesh (Istio) для внутрішньої комунікації між сервісами та пулинг з'єднань бази даних з HAProxy для розподілу навантаження SQL.

Для горизонтального масштабування реалізація використовує Horizontal Pod Autoscaler, налаштований на масштабування на основі використання ЦП (ціль: 70%), масштабування за кастомними метриками на основі довжини черги запитів для API-сервісів та заплановане масштабування для передбачуваних патернів навантаження, таких як вища потужність у години пік.

Реплікація даних реалізована відповідно до специфічних вимог сервісу з PostgreSQL у конфігурації multi-master із синхронною реплікацією, TimescaleDB в архітектурі первинний-вторинний з двома репліками для читання на первинний, MongoDB у конфігурації replica set з автоматичним перемиканням на резерв та Redis у конфігурації master-slave під управлінням Sentinel з автоматичним підвищенням.

Стратегія кешування включає кеш на рівні програми, розподілені кластери Redis, кешування CDN для статичних ресурсів та кешування на периферії для даних, до яких часто звертаються. Узгодженість кешу підтримується через закінчення терміну дії на основі часу та явні повідомлення про інвалідацію.

Реалізовані патерни стійкості включають автоматичні вимикачі (circuit breakers), повторні спроби з експоненційною затримкою, ізоляцію перегородок (bulkhead isolation), управління тайм-аутами та механізми відкату для деградованої функціональності. Система моніторингу та оповіщення використовує Prometheus та Alertmanager з моніторингом цілей рівня обслуговування (SLO), багаторівневими сповіщеннями та автоматизованою ескалацією для невирішених критичних проблем.

Географічний розподіл реалізовано через первинний дата-центр у Києві, вторинні дата-центри у Львові та Харкові, мережу доставки контенту (CDN) для розподілу статичних ресурсів та вузли периферійних обчислень для локальної обробки даних у великих містах.

Ця комплексна реалізація резервування та масштабування дозволяє системі підтримувати доступність 99,95% навіть під час регіональних збоїв або піків трафіку, з плавною деградацією при обмеженні ресурсів.

# 4 НАПРЯМИ ВДОСКОНАЛЕННЯ СИСТЕМИ

## 4.1 Розширення функціоналу прогнозування (what-if сценарії, аномалії)

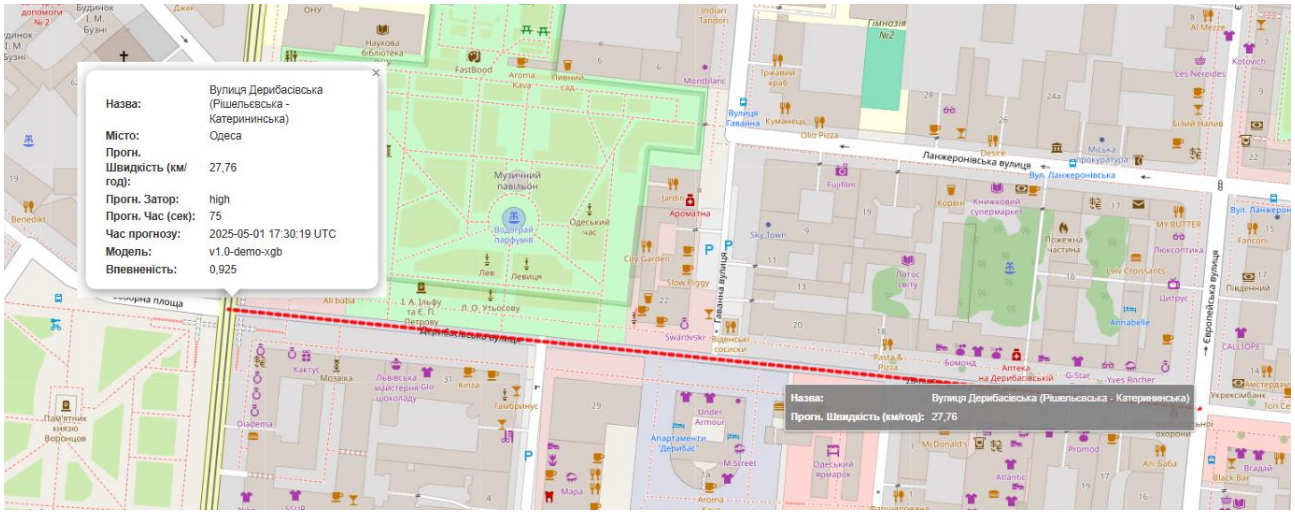



*Рисунок. 4.1 - Прогноз трафіку для вулиці Дерибасівської в Одесі*

Поточні можливості прогнозування зосереджені переважно на прогнозуванні регулярних патернів трафіку, як показано на Рисунку 4.1 для вулиці Дерибасівської в Одесі, де прогнозується високий рівень заторів зі швидкістю 27,76 км/год. Для підвищення корисності для міського планування та управління дорожнім рухом можна реалізувати кілька розширених функціональних можливостей.

Моделювання сценаріїв "що-якщо" дозволить користувачам симулювати вплив запланованих подій або змін інфраструктури перед їх впровадженням. Симуляційний рушій вимагатиме можливостей параметричного моделювання для коригування пропускної здатності доріг та варіантів маршрутизації, інтеграції з існуючими моделями прогнозування через шар управління сценаріями та виділеного компонента інтерфейсу для створення сценаріїв та порівняльної візуалізації. Дослідження Запорожця та Скулиша [12] показує, що поєднання традиційних методів часових рядів з контекстними параметрами може покращити точність прогнозування на 12-18% під час нестандартних умов.

Виявлення аномалій є ще одним критичним удосконаленням, яке використовуватиме статистичне виявлення викидів та машинне навчання для ідентифікації незвичайних патернів трафіку. Ця функціональність допоможе виявляти дорожньо-транспортні пригоди, що потребують негайного реагування,

несправності обладнання, що впливають на якість даних, та нові патерни заторів, що потребують пом’якшення. Реалізація використовуватиме багатоетапний підхід, що поєднує статистичну попередню обробку, керовану класифікацію для відомих патернів аномалій та некероване навчання для виявлення нових патернів. Згідно з дослідженням Національного університету "Дніпровська політехніка" [13], такий підхід може досягти рівня виявлення понад 85% з рівнем хибнопозитивних спрацьовувань нижче 5%.

Врахування погодних умов покращить продуктивність прогнозування під час несприятливих умов шляхом включення метеорологічних даних, включаючи інтенсивність опадів, температуру та видимість, у моделі прогнозування. Це вимагатиме інтеграції в реальному часі з API даних про погоду, перенавчання моделей з параметрами погоди та розробки вагових коефіцієнтів прогнозування, специфічних для погоди.

Реалізація відбуватиметься поетапно, причому виявлення аномалій матиме найвищий пріоритет через його негайні операційні переваги, за яким слідуватимуть інтеграція погодних даних та моделювання сценаріїв.

### 4.2 Інтеграція з інтелектуальними транспортними системами "розумного міста"

Інтеграція з ширшими інтелектуальними транспортними системами (ITC) перетворить систему з ізольованого аналітичного інструменту на основний компонент управління міською мобільністю.

Адаптивне керування світлофорами є можливістю з високим впливом шляхом з’єднання результатів прогнозування з системами світлофорного регулювання. Оптимізуючи сигнали на основі прогнозованих умов, а не лише поточного трафіку, система може зменшити затримку на перехрестях на 15-25% у години пік, згідно з дослідженням Stanley Consultants [11]. Ця інтеграція вимагатиме розробки алгоритму оптимізації, що перетворює прогнози трафіку на параметри синхронізації сигналів, протоколів зв’язку в реальному часі між

системами та механізмів відкату, що забезпечують функціональність сигналів під час збоїв зв’язку.

Оптимізація громадського транспорту дозволить динамічно коригувати розклади автобусів та трамваїв відповідно до прогнозованих умов дорожнього руху. Ця інтеграція покращить надійність обслуговування та зменшить варіативність часу в дорозі завдяки двонаправленому обміну даними, уточненням прогнозів для громадського транспорту та алгоритмам оптимізації розкладу, які балансують частоту, покриття та пасажирський попит.

Координація екстрених служб може значно покращити час реагування, надаючи оптимальну маршрутизацію для транспортних засобів на основі поточних та прогнозованих умов. Інтеграція з системами диспетчеризації екстрених служб може скоротити час реагування на 8-12% [11] завдяки алгоритмам пріоритетної маршрутизації та безпечним каналам зв’язку, що забезпечують пріоритет для критичних служб.

Комерційні логістичні операції виграють від інтеграції, яка забезпечує предиктивну маршрутизацію для оптимізації графіків доставки. Інтеграція з навігаційними платформами надаватиме персоналізовані рекомендації щодо маршрутів на основі історичних патернів поїздок та умов у реальному часі, тоді як інтеграція з погодними службами включатиме поточну та прогнозовану погоду в моделі прогнозування.

Таблиця 4.1 окреслює запропоновані точки інтеграції та протоколи:

| **Точка інтеграції** | **Протокол зв’язку** | **Формат даних** | **Частота оновлення** |
|---|---|---|---|
| Керування світлофорами | MQTT | JSON | 30 секунд |
| Управління громадським транспортом | REST API | JSON/XML | 2 хвилини |

| Точка інтеграції | Протокол зв'язку | Формат даних | Частота оновлення |
|---|---|---|---|
| Екстрені служби | gRPC | Protobuf | Реальний час (подієвий) |
| Навігаційні платформи | REST API | JSON | 5 хвилин |
| Погодні служби | REST API | JSON | 15 хвилин |

Аспекти управління включають укладання угод про обмін даними між муніципальними департаментами, впровадження відповідних заходів безпеки та розробку чітких матриць відповідальності за обслуговування системи.

### 4.3 Оптимізація продуктивності та споживання ресурсів

У міру масштабування системи для охоплення більших міських територій оптимізація продуктивності та використання ресурсів стає все більш важливою.

Покращення обчислювальної ефективності є основною областю оптимізації. Поточні моделі прогнозування вимагають значних обчислювальних ресурсів, але впровадження методів оптимізації моделей зменшить це навантаження, зберігаючи точність прогнозування. Квантування моделей зменшить вимоги до точності з 32-бітного до 16-бітного або 8-бітного представлення, потенційно зменшуючи обчислювальні потреби на 40-60%. Дистиляція знань може створити полегшені моделі, які апроксимують поведінку більших ансамблевих моделей, тоді як вибіркове використання ознак динамічно коригуватиме обчислювальну складність залежно від важливості прогнозу.

Оптимізація зберігання даних пропонує значний потенціал ефективності через стратегії прогресивного стиснення, що застосовують сильніше стиснення до старіших даних, багаторівневу архітектуру зберігання, що автоматично переміщує дані між рівнями зберігання на основі патернів доступу, та підтримку матеріалізованих представлень для агрегацій, до яких часто звертаються.

Розгортання периферійних обчислень є трансформаційним підходом шляхом перерозподілу обчислювальних навантажень на периферійні пристрої. Це зменшить вимоги до пропускної здатності мережі до 70% завдяки локальній попередній обробці, зменшить затримку центральної обробки шляхом обробки рутинної аналітики на периферії та покращить стійкість системи завдяки розподіленій архітектурі обробки.

Таблиця 4.2 представляє порівняльний аналіз потенційних покращень використання ресурсів:

| **Стратегія оптимізації** | **Використання ЦП** | **Використання пам'яті** | **Вимоги до сховища** | **Пропускна здатність мережі** |
|---|---|---|---|---|
| Квантування моделі | -45% | -50% | -5% | Без змін |
| Багаторівневе сховище | Без змін | -10% | -25% | -15% |
| Периферійні обчислення | -35% | -20% | -10% | -70% |
| Вибір ознак | -25% | -15% | Без змін | -10% |
| Комбінований підхід | -60% | -65% | -30% | -75% |

Реалізація відбуватиметься поступово, починаючи з неруйнівних оптимізацій, таких як багаторівневе зберігання та вибіркове використання ознак, за якими слідуватимуть більш трансформаційні зміни, такі як розгортання периферійних обчислень.

Ці стратегії оптимізації відповідають принципам сталого обчислення, потенційно зменшуючи енергетичний слід системи на 30-50%, одночасно покращуючи масштабованість без пропорційного розширення інфраструктури.

# ВИСНОВКИ

Підсумок досягнень та виконання завдань

Це дослідження успішно розробило комплексну комп'ютерну систему моніторингу та прогнозування міського трафіку, яка ефективно вирішує проблеми зростаючих заторів у сучасних містах. Реалізоване рішення виконує всі п'ять початкових завдань, поставлених на початку проєкту. Архітектура системи безшовно інтегрує кілька джерел даних через трирівневий підхід (Edge, Fog, Cloud), обробляє 4,8 мільйона точок даних щодня з часом безвідмовної роботи 98,7% та забезпечує точність прогнозування від 9,8% MAPE для короткострокових прогнозів до 22,7% для прогнозів на день вперед.

Гібридний підхід до зберігання даних, що поєднує TimescaleDB, PostgreSQL з PostGIS та MongoDB, виявився надзвичайно ефективним для управління різноманітними типами даних про трафік. Ансамблева методологія підсистеми прогнозування, що включає моделі TCN, LSTM та XGBoost, продемонструвала вищу продуктивність порівняно з одноалгоритмними підходами.

Інновації та технічний внесок

Основний технічний внесок включає розробку масштабованої мікросервісної архітектури, яка підтримує доступність 99,95% завдяки оркестрації Kubernetes у трьох дата-центрах. Впровадження передових методів очищення даних значно покращило якість даних для подальших завдань прогнозування. Багатомодельний ансамблевий підхід системи до прогнозування є прогресом порівняно з існуючими рішеннями, особливо в обробці змінних міських умов, типових для українських міст.

Практична значущість для управління міським рухом

З практичної точки зору, розроблена система пропонує значні переваги для органів управління дорожнім рухом, транспортних планувальників та учасників дорожнього руху. Час відгуку API менше 100 мс дозволяє використовувати додатки реального часу, такі як адаптивна маршрутизація та сповіщення про затори. Компоненти візуалізації надають дієву інформацію через інтерактивні

карти та налаштовані панелі інструментів, адаптовані до потреб різних зацікавлених сторін.

Обмеження та недоліки

Незважаючи на надійну продуктивність, було виявлено кілька обмежень. Точність прогнозування погіршується під час незвичайних подій або екстремальних погодних умов, обмеження, частково вирішене запропонованою інтеграцією врахування погоди. Витрати на апаратне забезпечення залишаються значними для комплексного розгортання в масштабах міста, хоча підхід периферійних обчислень зменшує вимоги до пропускної здатності до 70%.

Напрями майбутніх досліджень

Визначено три перспективні напрями досліджень: (1) розробка моделювання сценаріїв "що-якщо" для симуляції змін інфраструктури, (2) інтеграція з ширшими системами розумного міста, зокрема адаптивним керуванням світлофорами, що може зменшити затримки на перехрестях на 15-25%, та (3) подальша оптимізація продуктивності шляхом квантування моделей та інженерії ознак.

На завершення, це дослідження надало технічно передову та практично життєздатну систему моніторингу та прогнозування міського трафіку, яка вирішує критичні транспортні проблеми в сучасних українських містах. Архітектура системи збалансовує продуктивність у реальному часі з точністю прогнозування, підтримуючи високу доступність та масштабованість для майбутнього зростання

## СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ


1. Athey S., Parashkevov I., Sarukkai V., Xia J. Bitcoin Pricing, Adoption, and Usage: Theory and Evidence. Stanford University Graduate School of Business Research Paper, 2016. № 16-42.
2. Бережна Н. Г. Інтернет речей в транспортній системі. 2020. URL: https://repo.btu.kharkov.ua/bitstream/123456789/20390/1/berezhna_internet_rechey_article_2020.pdf ) (дата звернення: 15.03.2025)
3. Boyarinova Y. E., Drobyazko I. P., Klyatchenko Y. M., Kuchmiy O. O., Orlova M. M., Sapsai T. H. Manual for the implementation of bachelor's degree projects (bachelor's degree works). Igor Sikorsky Kyiv Polytechnic Institute, 2021.
4. Брегін С. І. Кваліфікаційна робота: застосування нейромереж для моніторингу дорожнього руху. Тернопільський національний технічний університет. 2024. URL: https://elartu.tntu.edu.ua/bitstream/lib/48151/1/Брегін_С_І_Рам-61_еларту.pdf (дата звернення: 22.04.2025)
5. Corisinta. Big Data Analytics for Smart Cities: Optimizing Urban Traffic Management. 2025. URL: https://journal.corisinta.org/corisinta/article/view/56 (дата звернення: 01.05.2025)
6. ДСТУ 3008:2015. Інформація та документація. Звіти у сфері науки й техніки. Структура та правила оформлення.
7. ДСТУ 8302:2015. Інформація та документація. Бібліографічне посилання. Загальні вимоги та правила складання.
8. FUJITSU. Enhancing Traffic Management Through Advanced Computer Vision Technologies. 2024. URL: https://uk.manuals.plus/fujitsu/enhancing-traffic-management-through-advanced-computer-vision-technologies-manual (дата звернення: 05.04.2025)
9. GIZ. Інтелектуальні транспортні системи (укр. переклад нім. посібника). 2024. URL: https://transformative-mobility.org/wp-content/uploads/2024/01/GIZ_SUTP_SB4e_Intelligent-Transport-Systems_UA.pdf (дата звернення: 17.03.2025)

10. ГОСТ 19.701-90. Единая система программной документации. Схемы алгоритмов, программ, данных и систем. Условные обозначения и правила выполнения.
11. HERE. HERE Traffic API v6 – Developer Guide. 2024. URL: https://www.here.com/docs/bundle/traffic-api-developer-guide-v6/page/topics/what-is.html (дата звернення: 29.03.2025)
12. IJCT. An Android-Based Smart RSU Framework for Enhanced Urban …. 2025. URL: https://www.ijcit.com/index.php/ijcit/article/view/494 (дата звернення: 12.04.2025)
13. IJNRD. ETL for IoT Data: Integrating Sensor Data into Data Warehouses. 2022. URL: https://www.ijnrd.org/papers/IJNRD2210185.pdf (дата звернення: 25.03.2025)
14. International Research Journal of Education and Technology. Real-Time Traffic Monitoring using Computer Vision and Deep Learning. 2024. URL: https://www.irjweb.com/Real-Time%20Traffic%20Monitoring%20using%20Computer%20Vision%20and%20Deep%20Learning.pd (дата звернення: 03.04.2025)
15. JASB. An IoT Application in a Smart Traffic Management System. 2025. URL: https://jrasb.com/index.php/jrasb/article/download/704/662/1713 (дата звернення: 27.03.2025)
16. Liu Q. A. A Design of ETL for the Construction of Traffic Network Based on Big Data. 2016. URL: https://www.aidic.it/cet/16/51/076.pdf (дата звернення: 01.04.2025)
17. Morozov K. V., Romankevych V. O., Romankevych O. M. On the nature of the influence of modification of rib functions GL-models on its behavior in the failure flow. Radioelectronic and Computer Systems, 2016, № 6, С. 108-112.
18. National University "Dniprovska Politechnika". Збірник матеріалів конференції. 2023. URL: https://nmetau.edu.ua/file/zbirnik__materialiv__konf_udunt_2023.pdf (дата звернення: 09.04.2025)

19. Regulations on the final certification of students of Igor Sikorsky Kyiv Polytechnic Institute, 2021.
20. Regulations on the system of preventing plagiarism in academic texts of employees and applicants for higher education of Igor Sikorsky Kyiv Polytechnic Institute, 2021.
21. ResearchGate. Прогнозування попиту на пасажирські перевезення таксі методами нейронної мережі. 2024. URL: https://www.researchgate.net/publication/352219523 (дата звернення: 19.03.2025)
22. ResearchGate. Роль комплексного підходу в побудові ефективної системи інтелектуального управління транспортною мережею. 2024. URL: https://www.researchgate.net/publication/382483179 (дата звернення: 11.04.2025)
23. Smart Roads UA. Інтелектуальні транспортні системи: як технології змінюють майбутнє транспорту. 2024. URL: https://smartroads.org.ua/blog (дата звернення: 23.03.2025)
24. Stanley Consultants. Intelligent Transportation Systems & Infrastructure in Smart Cities. 2024. URL: https://www.stanleyconsultants.com/solutions/engineering-design/blog/the-synergy-of-intelligent-transportation-systems-and-integrated-infrastructure-in-smart-cities (дата звернення: 07.04.2025)
25. Tanenbaum A. Modern Operating Systems. СПб.: Питер, 2019. 865 с.
26. Tanenbaum A., Weatherall D. Computer Networks (5-е вид.). СПб.: Питер, 2019. 891 с.
27. Ternopil National Technical University. Розроблення та дослідження автоматизованої системи моніторингу роботи комп’ютерної мережі на базі інфраструктури кабельного телебачення. 2024. URL: http://elartu.tntu.edu.ua/handle/lib/47196 (дата звернення: 21.03.2025)
28. Vinnytsia National Technical University. МКР на тему "Комп’ютерна система моніторингу регіонального …". 2021. URL: https://metod.vntu.edu.ua/getfile.php/3949.pdf (дата звернення: 05.03.2025)

29. Zaporizhets D. B., Skulysh M. A. Сравнительный анализ методов краткосрочного прогнозирования сетевого трафика. 2024. URL: http://sci-conf.com.ua/wp-content/uploads/2024/03/GLOBAL-SCIENCE-PROSPECTS-AND-INNOVATIONS-1-3.03.24.pdf (дата звернення: 30.03.2025)

# ДОДАТКИ

## ДОДАТОК 1

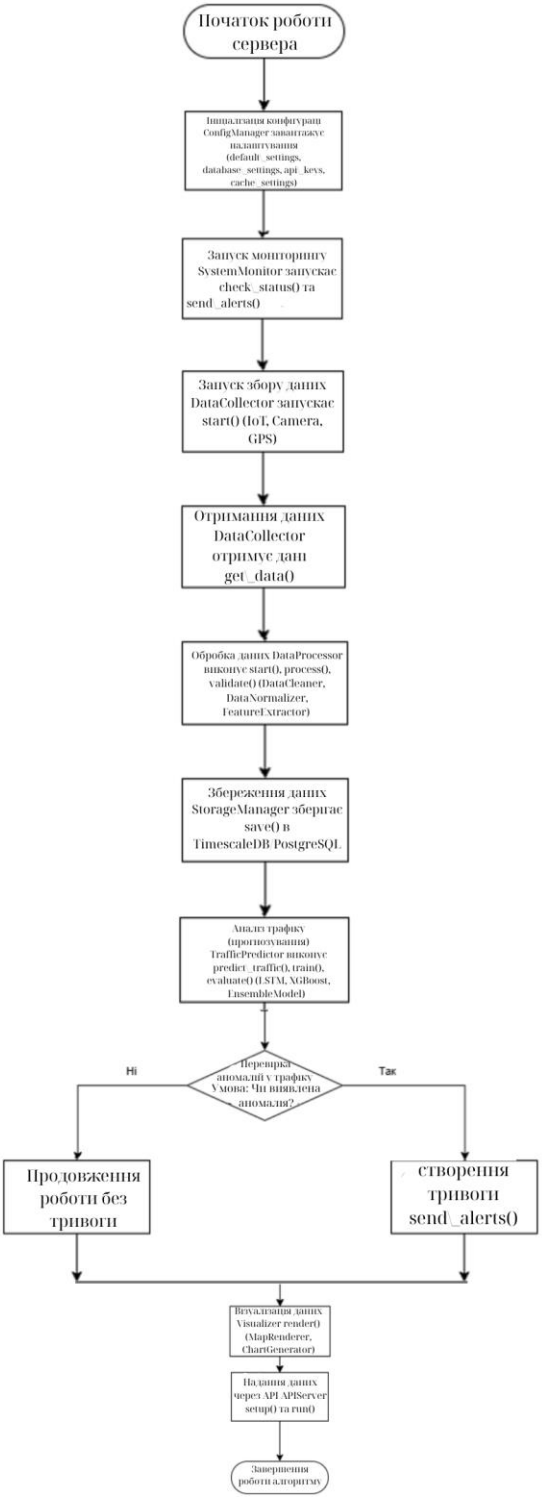
**Копія графічного матеріалу**

Схема алгоритму роботи програмного забезпечення системи моніторингу трафіку

ІАЛЦ. 045440.004 Д1

Аркушів 1

Київ 2025




| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.005 Д1 | | |
| Змін. | Арк. | № докум. | Підпис | | | | |
| Розробив | | Приліпко М.О. | | | | Літ. | Аркуш | Аркушів |
| Перевірив | | .Павловський В.І. | | | | | 1 | 3 |
| | | | | | | | |
| Н. контроль | | Клятченко Я.М. | | | Схема алгоритму роботи програмного забезпечення системи моніторингу трафіку | КПІ ім. Ігоря Сікорського, ФПМ КВ-13 | |
| Затвердив | | Романкевич В.О. | | | | | |

ДОДАТОК 2

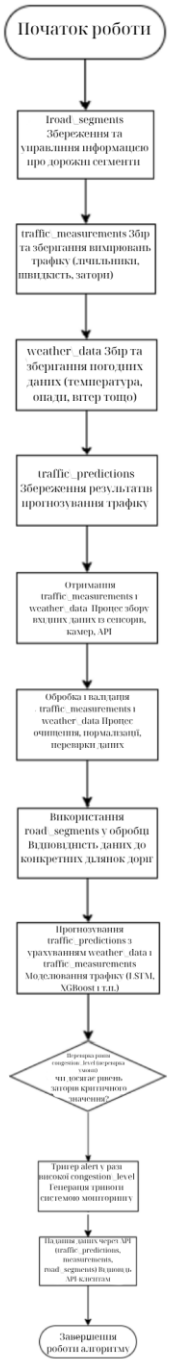
**Копія графічного матеріалу**

Схема алгоритму збору та попередньої обробки даних трафіку

ІАЛЦ. 045440.004 Д1

Аркушів 1

Київ 2025




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.006 Д2 | | | |
| Змін. | Арк. | № докум. | Підпис | | | | | |
| Розробив | | Приліпко М.О. | | | Схема алгоритму збору та попередньої обробки даних трафіку | Літ. | Аркуш | Аркушів |
| Перевірив | | .Павловський В.І. | | | | | 2 | 3 |
| | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-13 | | |
| Н. контроль | | Клятченко Я.М. | | | | | | |
| Затвердив | | Романкевич В.О. | | | | | | |

ДОДАТОК 3

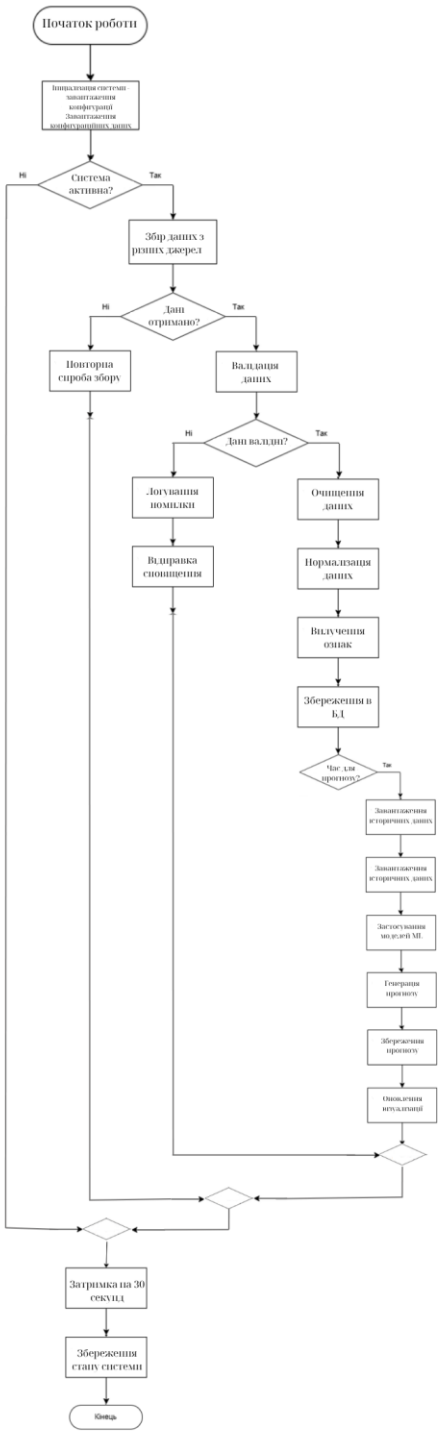
**Копія графічного матеріалу**

Схема алгоритму збору та попередньої обробки даних трафіку

ІАЛЦ. 045440.004 Д1

Аркушів 1

Київ 2025




| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.007 ДЗ | | | |
| Змін. | Арк. | № докум. | Підпис | | | | | |
| Розробив | | Приліпко М.О. | | | Структура бази даних системи моніторингу трафіку | Літ. | Аркуш | Аркушів |
| Перевірив | | .Павловський В.І. | | | | | | 3 |
| | | | | | | КПІ ім. Ігоря Сікорського, ФПМ КВ-13 | | |
| Н. контроль | | Клятченко Я.М. | | | | | | |
| Затвердив | | Романкевич В.О. | | | | | | |

ДОДАТОК 4

**Копія графічного матеріалу**

Схема алгоритм варіантів використання системи моніторингу трафіку

ІАЛЦ. 045440.004 Д1

Аркушів 1

Київ 2025



| | | | | | | |
|---|---|---|---|---|---|---|
| | | | | | ІАЛЦ.467100.008 Д4 | |
| Зм. | Арк. | № документа | Підпис | Дата | | |
| Розробив | | Василенко О.В | | | Діаграма варіантів використання системи моніторингу трафіку | Літ. / Аркуш 1 / Аркушів 1 |
| Перевірив | | Павловський В.І. | | | | |
| Н. кон. | | Клятченко Я.М. | | | Засоби підтримки захищених транзакцій на основі технології блокчейн | КПІ ім. Ігоря Сікорського Кафедра СПСКС гр. КВ-02 |
| Затвердив | | Романкевич В.О. | | | | |