Хмельницький національний університет
Факультет інформаційних технологій
Кафедра комп'ютерної інженерії та інформаційних систем

# КВАЛІФІКАЦІЙНА РОБОТА

бакалавр
Освітній рівень

Кіберфізична система відстеження транспортних засобів в реальному часі
Назва теми

КвРКІ. 190240.19.02.31 ПЗ
Шифр

Галузь знань 12 «Інформаційні технології»
Шифр, назва

Спеціальність 123 «Комп'ютерна інженерія»
Шифр, назва

Освітня програма «Комп'ютерна інженерія та програмування»
Назва

Виконав: студент IV курсу, група КІ2-19-2 _____ Д. М. Рибанчук
Підпис Ініціали, прізвище

Керівник _____ Т.М. Кисіль
Підпис, дата Ініціали, прізвище

Нормоконтролер _____ С.М. Лисенко
Підпис, дата Ініціали, прізвище

До захисту допускаю:
Зав. кафедри комп'ютерної інженерії та інформаційних систем _____ Т.О. Говорущенко
Підпис Ініціали, прізвище

«7» червня 2023 р.

Хмельницький 2023

ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет Інформаційних технологій

Кафедра Комп'ютерної інженерії та інформаційних систем

Освітній рівень рівень бакалавр

Галузь знань 12 Інформаційні технології

Спеціальність 123 Комп'ютерна інженерія

Освітня програма «Комп'ютерна інженерія та програмування»

ЗАТВЕРДЖУЮ

Зав. кафедри Т.О.Говорущенко

"11" 01 2023 р.

**ЗАВДАННЯ**
**НА КВАЛІФІКАЦІЙНУ РОБОТУ БАКАЛАВРА**

Рябінчуку Дмитру Миколайовичу

Прізвище, ім'я, по батькові студента

1. Тема проекту (роботи) Кіберфізична система відстеження транспортних засобів в реальному часі

Керівник проекту (роботи) Кисіль Т.М., к.ф. –м.н, доц.

Прізвище, ім'я, по батькові, науковий ступінь, вчене звання

Затверджена наказом ректора університету від 01.03.2023 р. № 5

2. Строк подання студентом проекту (роботи) на кафедру 01.06.2023 р.

3. Вихідні дані до проекту (роботи) Завдання на дипломне проектування

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Огляд відомих методів та їх рішення

аналіз наявного програмного забезпечення системи моніторингу транспортних засобів у реальному часі

програмна реалізація та тестування програмно-технічного засобу та висновки

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслень)

Блок-схеми "kafka"

Блок-схеми "Apache Strom та загальна"

Користувацький інтерфейс (1)

Користувацький інтерфейс (2)

6. Консультанти розділів дипломного проекту (роботи)

| Розділ | Прізвище, ініціали та посада консультанта | Підпис, дата | |
|---|---|---|---|
| | | завдання видав | завдання прийняв |
| Нормоконтроль | Лисенко С.М., професор кафедри КІС | | |
| Антиплагіат | Нічепорук А.О., доцент кафедри КІС | | |

7. Дата видачі завдання « 11 » 01 2023 р.

## КАЛЕНДАРНИЙ ПЛАН

| №з/п | Назва етапів (розділів) дипломного проекту (роботи) | Термін виконання етапів проекту (роботи) | Примітка |
|---|---|---|---|
| 1 | Вибір напряму дослідження та узгодження тематики кваліфікаційної роботи з керівником | 11.01.2023 | виконано |
| 2 | Ознайомлення з предметною областю; формулювання мети та задач дослідження; визначення об'єкта та предмета дослідження | 01.02.2023 | виконано |
| 3 | Робота над розділом 1 – Огляд відомих методів та їх рішення | 01.03.2023 | виконано |
| 4 | Робота над розділом 2 – Аналіз наявного програмного забезпечення системи моніторингу транспортних засобів у реальному часі | 01.04.2023 | виконано |
| 5 | Робота над розділом 3 – Програмна реалізація та тестування програмно-технічного засобу та висновки | 30.04.2023 | виконано |
| 6 | Оформлення пояснювальної записки згідно вимог | 11.05.2023 | виконано |
| 7 | Попередній захист ВКР | 26.05.2023 | виконано |
| 8 | Захист ВКР на засіданні ЕК | Червень 2023 року | |

**Студент** _____ Підпис Д.М.Рибанчук Ініціали, прізвище

**Керівник проекту (роботи)** _____ Підпис Т.М.Кисіль Ініціали, прізвище

| № рядка | формат | Позначення | Найменування | Кіл. листів | № екз | Примітка |
|---|---|---|---|---|---|---|
| | | | Текстові документи | | | |
| 1 | А4 | КвРКІ 190240.19.02.31 ПЗ | Пояснювальна записка | 54 | | |
| | | | Графічні матеріали | | | |
| 2 | А2 | КвРКІ 190240.19.02.31 Е8 | Блок-схеми | 1 | | |
| 3 | А2 | КвРКІ 190240.19.02.31 Е8 | Користувацький | 1 | | |
| | | | інтерфейс | | | |
| 4 | А2 | КвРКІ 190240.19.02.31 Е8 | Користувацький | 1 | | |
| | | | інтерфейс | | | |
| 5 | А2 | КвРКІ 190240.19.02.31 Е8 | Блок-схеми | 1 | | |

КвРКІ 190240.19.02.31 ВП

| Зм | Арк | № докум | Підпис | Дата |
|---|---|---|---|---|
| Розробив | | Рибачук | | |
| Перевір. | | Кисіль | | |
| Н.контр. | | Лисенко | | |
| Затв. | | Гончаренко | | 07.06 |

Відомість проекту

| Літера | Аркуш | Аркушів |
|---|---|---|
| У | 1 | 1 |

ХНУ, КІ2-19-2

# АНОТАЦІЯ

Тема кваліфікаційної роботи: Кіберфізична система відстеження транспортних засобів у реальному часі

Автор роботи: Рибанчук Дмитро Миколайович

Керівник роботи: Кисіль Тетяна Миколаївна.

Пояснювальна записка: 54 с., 19 рис., 12 табл., 60 джерел.

Графічна частина:4 креслення

Кіберфізична система відстеження транспортних засобів у реальному часі

Метою дослідження полягає у проєктуванні та розробці програмно–апаратного пристрою для кіберфізичної системи відстеження транспортних засобів у реальному часі.

Об'єктом дослідження є програмне забезпечення для кіберфізичної системи відстеження транспортних засобів у реальному часі

Предметом дослідження є Кіберфізична система відстеження транспортних засобів у реальному часі.

Практичне цінність полягає у проєктуванні та реалізації Кіберфізична система відстеження транспортних засобів у реальному часі

Підпис студента

Дата

**ЗМІСТ**

# ВСТУП

Кіберфізична система відстеження транспортних засобів у реальному часі - це комплексна технологія, що поєднує в собі елементи кібернетики та фізичних процесів. Вона дозволяє відслідковувати рух транспортних засобів у реальному часі та забезпечує надійну та точну інформацію про їх місцезнаходження.

Одним із головних компонентів кіберфізичної системи є сенсори, які встановлюються на транспортні засоби та забезпечують збір різноманітної інформації про них, таку як швидкість руху, відстань пройдену засобом, кількість палива в баку тощо. Також до складу системи входять GPS-трекери, що дозволяють визначати місцезнаходження транспортних засобів з високою точністю.

Зібрана інформація передається до центральної бази даних, де проводиться її обробка та аналіз. Централізована система дозволяє в режимі реального часу контролювати місцезнаходження транспортних засобів, їхню швидкість руху та діяти в разі виникнення непередбачуваних ситуацій.

В данній роботі буде досліджено принципи роботи кіберфізичної системи відстеження транспортних засобів у реальному часі, проаналізовано її основні компоненти та функції.

Основна мета дипломної роботи - проаналізувати переваги та можливості кіберфізичних систем відстеження транспортних засобів у реальному часі та їхній вплив на розвиток транспортної інфраструктури та економіку в цілому. Також будуть розглянуті можливості використання таких систем для покращення безпеки на дорогах та якості обслуговування пасажирів.

# 1. ОГЛЯД ВІДОМИХ МЕТОДІВ ТА ЇХ РІШЕННЯ

## 1.1 Формування понятєвої бази щодо систем відстеження транспортних засобів у реальному часі

Кіберфізична система моніторингу транспорту в реальному часі - це системний підхід до обробки та моніторингу великих обсягів географічних даних, що надходять з великої кількості автомобілів, з використанням розподіленого середовища для паралельної обробки та збереження даних, що дозволяє ефективно і безперешкодно вирішувати різні проблеми, з якими стикаються автономні сервери додатків, з низькою затримкою, підвищеною пропускною здатністю і надійністю.

Системи моніторингу транспорту існують вже більше десяти років. Раніше це були пристрої на основі GPS, які встановлювалися в машину. Пристрій періодично отримував інформацію про географічне розташування (широта, довгота) і надсилав її на сервер.

Сервер складався з апаратного забезпечення з фіксованими компонентами, такими як пам'ять, жорсткий диск та процесор. Пристрій з підтримкою GPS використовувався для надсилання Http запиту на певний порт сервера. На сервері встановлюється певна програма та скрипти, розгорнуті на певній мові програмування, такій як Perl, ruby, PHP або Java, які використовуються для отримання сигналу http-запиту. Потім запит обробляється, модифікується, якщо необхідно, і перенаправляється в інше місце для подальшої обробки або зберігається в деяких реляційних базах даних. Доступ до збереженої інформації з баз даних здійснюється за допомогою деяких мов високого рівня для отримання або маніпулювання інформацією та відображення різних результатів моніторингу за допомогою інтерфейсів користувача на основі браузера та інших видів звітів, що генеруються у текстових файлах, таблицях Excel або форматах PDF.

Транспортні засоби увійшли в кожну сферу життя, чи то шкільний автобус, який підвозить чи відвозить ваших дітей, чи то ваш щоденний графік життя - похід в офіс, відвідування родичів чи купівля товарів для щоденного вжитку. Жодна частина життя не залишається недоторканою. Саме тому кількість транспортних засобів зростає з кожним днем. Понад 1,2 мільярда транспортних засобів одночасно їздять по дорогах, і це при мінімальній кількості людей, які використовують транспорт. На цьому тлі для державних організацій, що обслуговують транспортні засоби, або будь-яких корпоративних організацій, стало обов'язковим піклуватися про безпеку людей та їх життя. Для приватних організацій і компаній відстеження транспортних засобів стало обов'язковим для того, щоб підвищити продуктивність працівників і довіру до них.

Ця система добре підходить для моніторингу кількох сотень автомобілів на день. Але як тільки навантаження починає зростати, скажімо, до двадцяти тисяч автомобілів, система починає працювати дуже повільно. Оскільки система має обмежений обсяг пам'яті та швидкість процесора, моніторинг транспортних засобів, кількість яких перевищує кілька сотень, стає справжньою проблемою. Вертикальне масштабування обмежене певним порогом, скажімо, оперативна пам'ять пристрою не може вийти за межі певних гігабайт, так само як і швидкість процесора. Очевидно, що виникає потреба в горизонтальному масштабуванні. Система повинна складатися з багатьох підсистем, які б працювали паралельно, рівномірно розподіляючи навантаження на всі підсистеми і безперешкодно виконували обчислення на звичайному апаратному забезпеченні.

Інший аспект моніторингу транспорту полягає в тому, що існує величезний попит на моніторинг транспортних засобів протягом певного періоду часу. Це призводить до пікового попиту на обчислювальні ресурси в один ,конкретний , час. Це призводить до пікового попиту на обчислювальні ресурси в один час, а не в інший. В такому випадку один потужний сервер з дуже великою вартістю і високими обчислювальними ресурсами є безглуздим і непродуктивним, коли є лише невелика кількість транспортних засобів, що підлягають моніторингу.

Паралельна розподілена архітектура системи моніторингу транспорту дає перевагу додавання додаткового вузла або апарата під час пікових навантажень, а коли навантаження є меншим, їх можна звільнити і використовувати для інших цілей.

Відмовостійкість ще один важливий аспект, який турбує багато підприємств. Запуск єдиної апаратної машини високого класу для моніторингу сотень клієнтів - це добре, поки вона не виходить з ладу. Як тільки збій трапляється, це майже глухий кут для всіх даних та інформації, що зберігаються на машині. Не факт, що система відновлення може бути реалізована або механізми резервного копіювання можуть бути впроваджені в єдиній ізольованій системі, однак ймовірність ризику та втрат в такому випадку є вищою. Належним чином реалізована розподілена система дозволить вам слідкувати за станом машини незалежно від того, на якій машині виконуються обчислення.

Це дозволяє нам мати розподілені координуючі сервіси, головний вузол відстежує всі підлеглі вузли. Це дозволяє ізолювати вузли даних від обчислювальних вузлів, щоб забезпечити безпеку даних, навіть якщо обчислювальний вузол вийде з ладу. Крім того, вузли даних можуть бути налаштовані на створення декількох копій, щоб у разі виходу з ладу вузла даних головний вузол міг призначити обчислювальні задачі, які зараз виконуються на цьому вузлі, з реплікованого вузла.

Як показано на рисунку 1.1, транспортні засоби оснащені пристроями на основі GPS. Супутник GPS передає сигнали на GPS-приймач. Ці приймачі пасивно приймають супутникові сигнали; вони не передають сигнали і потребують безперешкодного огляду неба, щоб їх можна було ефективно використовувати на відкритому повітрі. Колись ця технологія мала проблеми в таких місцях, як густі ліси або високі будівлі. Однак останнім часом ця технологія набула значного розвитку.

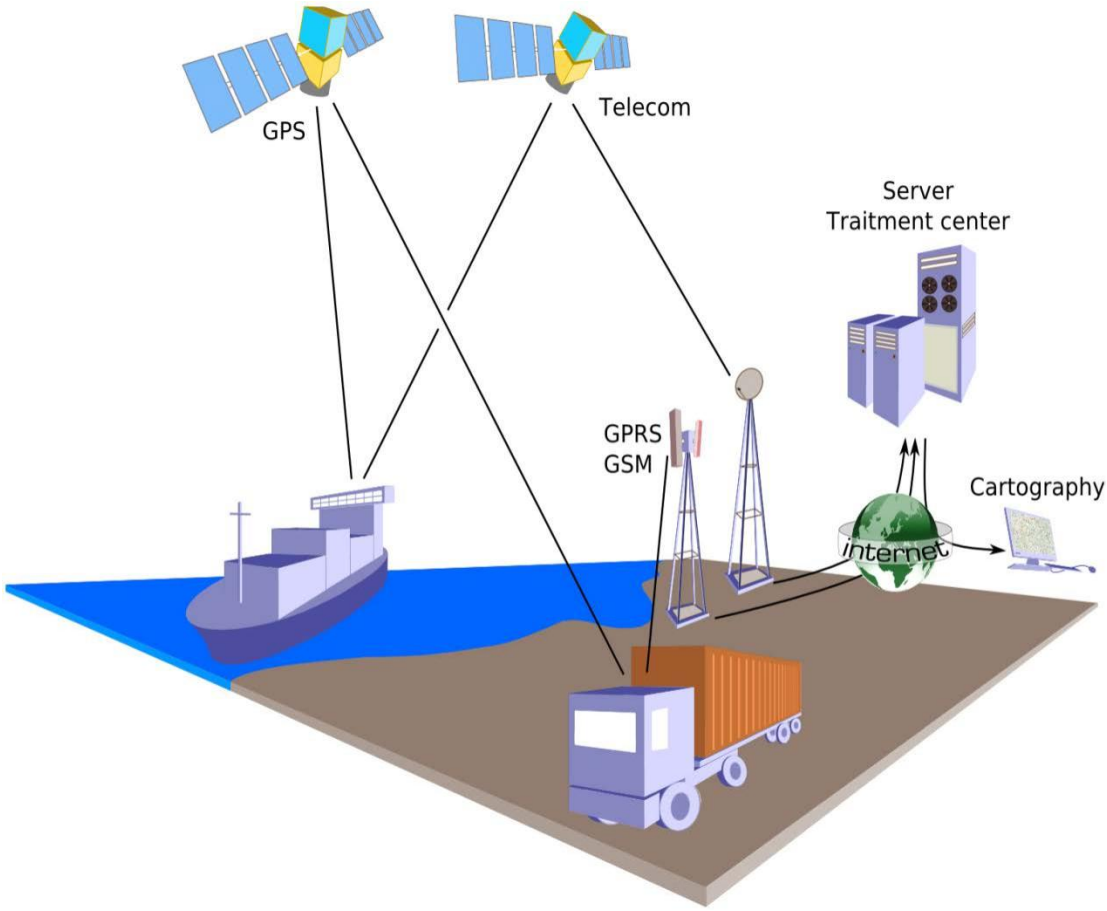



Рисунок 1.1 Моніторинг транспорту за допомогою GPS/GSM-технологій

Перша супутникова навігаційна система Transit, використовувана ВМС США, була успішно випробувана в 1960 році. Вона використовувала угруповання з п'яти супутників, і могла забезпечити фіксацію навігаційних даних приблизно раз на годину. Під час холодної війни і гонки озброєнь, ядерна загроза існуванню Сполучених Штатів була єдиною причиною, яка дозволила Конгресу США підтримати програму супутникової навігації. Під час холодної війни і гонки озброєнь, ядерна загроза існуванню Сполучених Штатів була єдиною причиною, яка дозволила Конгресу США підтримати програму супутникової навігації. 1 травня 2000-го року було прийнято рішення зняти обмеження з цивільної версії , що привело до підвищення точності цивільних GPS від 100 метрів до 20 метрів. За останнє десятиліття в США реалізовані декілька поліпшень служби GPS, у тому числі нові сигнали для цивільного використання, підвищення точності для усіх користувачів, зберігаючи при цьому сумісність з існуючим обладнанням GPS.

Модернізація системи GPS на даний момент проводиться постійно з додаванням нових можливостей для задоволення зростаючих військових, цивільних і комерційних потреб

У той же час, моніторинг транспорту можливий і через постачальників телекомунікаційних мереж завдяки використанню технологій GPRS і GSM. GPRS вбудовує технологію обробки інтернет-протоколу (IP) в сучасну мобільну мережу GSM. Серії пакетів маршрутизуються по декількох шляхах через мережу, а не як безперервний потік бітів через виділену комутовану мережу.

GSM (Global System for Mobile Communication) - це стандарт, розроблений ETSI (Європейським інститутом телекомунікаційних стандартів) для опису протоколів для цифрових стільникових мереж другого покоління (2G), що використовуються мобільними телефонами. GSM використовує різновид множинного доступу з часовим розділенням каналів (TDMA) і є найпоширенішою з трьох технологій цифрової бездротової телефонії (TDMA, GSM і CDMA). GSM оцифровує і стискає дані, а потім надсилає їх по каналу з двома іншими потоками даних користувача, кожен у своєму часовому інтервалі. Він працює в діапазоні частот 900 МГц або 1800 МГц.

Останнім часом технологічний прогрес є досить стрімким.

Технологія 2G перетворилася на високошвидкісні та високопродуктивні технології 3G і 4G.Також достатньо масово розпочалися тестування 5G.

Таблиця 1.1 підсумовує тенденції, що спостерігаються в розвитку комунікаційних технологій.

Розвиток комунікаційних технологій має позитивний зв'язок з розвитком систем моніторингу транспорту. Раніше системи моніторингу були призначені лише для збору інформації про географічне розташування, такої як широта, довгота і висота, через обмеження швидкості зв'язку.

Таблиця 1.1 порівняльна характеристика комунікаційних технологій.

| Покоління | Швидкість | Технологія | Характеристики |
|---|---|---|---|
| 2G | 9.4/14.4 Kbps | TDMA, CDMA | Кілька користувачів працюють на одному каналі за допомогою мультиплексування. 2G Дозволяє використовувати мобільні телефони для передачі даних разом з голосовим зв'язком. |
| 3G | 3.1 Mbps(Peak) 500-700 Kbps | CDMA 2000 (1XRTT, EVDO) UMTS, EDGE | Високошвидкісний Інтернет-браузинг, відеодзвінки, потокове відео |
| 3.5G | 14.4Mbps(Peak) 1-3 Mbps | HSPA | Підтримує вищу швидкість і забезпечує більші потреби в даних. |
| 4G | 100-1000Mbps(Peak) 10-50Mbps | WiMAX LTE | Дуже висока швидкість, дозволяє передавати потокове відео у форматі HD. |
| 5G | 1-10Gps 50+Mbps | OFDM | Покращений мобільний широкосмуговий зв'язок, Масовий IoT |

Останнім часом, коли швидкість зв'язку значно зросла,як показо на рисунку 1.2, системи моніторингу не тільки збирають географічну інформацію, але й передають на сервер багато інших метаданих, таких як погодні умови, температурна статистика, інформація про паливо, інформація про маршрут та інша різноманітна інформація. Крім того, на сервер також надсилаються відео та зображення маршруту і автопарку в режимі реального часу.

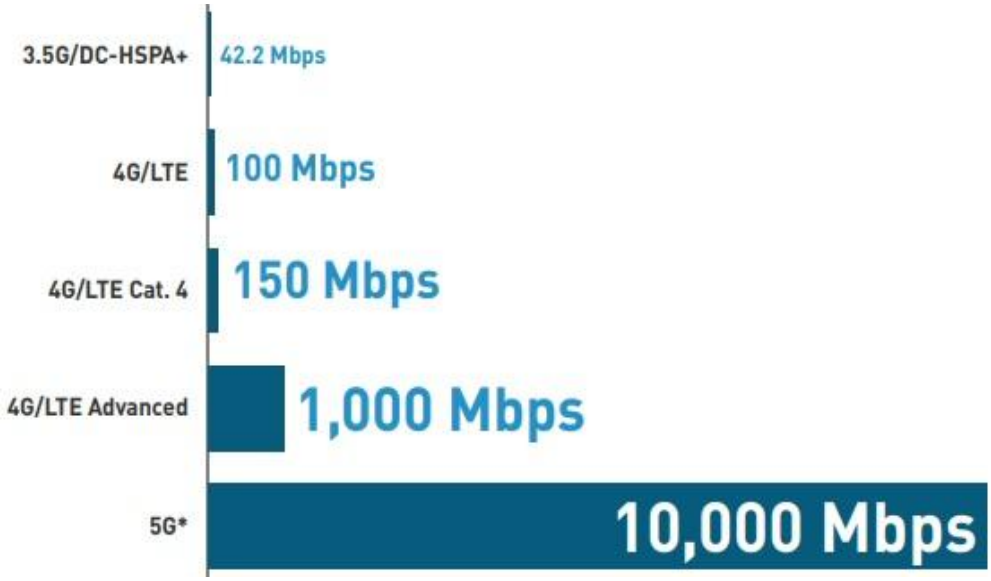



Рисунок 1.2 − Порівняння швидкості зв'язку різних технологій стільникового зв'язку

## 1.2 Потреби для використання системи моніторингу

Використання транспортних засобів стрімко зростає з кожним днем, наприклад для перевезення вантажів, подорожей і туризму, підвезення учнів до школи або для будь-яких інших особистих потреб, таких як похід в офіс, відвідування родичів, торгового центру тощо... Залишаючи транспортний засіб у невідомому місці, природним є занепокоєння про безпеку транспортних засобів від крадіжки пального або інших небажаних дійств з їх авто.

Таким чином, існує величезний попит на системи моніторингу транспортних засобів. Існуючі на сьогоднішній день системи моніторингу є досить дорогими, що не по кишені простому користувачеві. Висока вартість системи моніторингу в першу чергу пов'язана з тим, що традиційна технологія, яка використовується для моніторингу, потребує занадто багато зусиль для підтримки, модернізації та розгортання функцій. Висока доступність і надійність системи не може бути реалізована без суворого моніторингу впровадженої системи, що призводить до неекономічності системи для звичайної групи

користувачів. Користувачі в першу чергу потребують наступних елементів як основної частини хорошого прикладного програмного забезпечення:

1. Простота і зручність у використанні: Додаток для моніторингу має бути досить простим в експлуатації, його легко візуалізувати та отримувати статистику в режимі реального часу.
2. Висока надійність і доступність: Додаток повинен працювати з високою доступністю та відмовостійкістю.
3. Економічність: Додаток повинен бути дешевим для експлуатації.

Вищезазначені потреби можуть бути ефективно реалізовані за допомогою системи моніторингу транспортних засобів. Ця система моніторингу дозволила б нам забезпечити локалізацію даних таким чином, що дані користувача з будь-якої точки світу, не потрібно було б зберігати на централізованому сервері, розташованому,наприклад , в Києві, однак, систему можна було б легко контролювати за потреби з офісу як показано на рисунку 1.3.

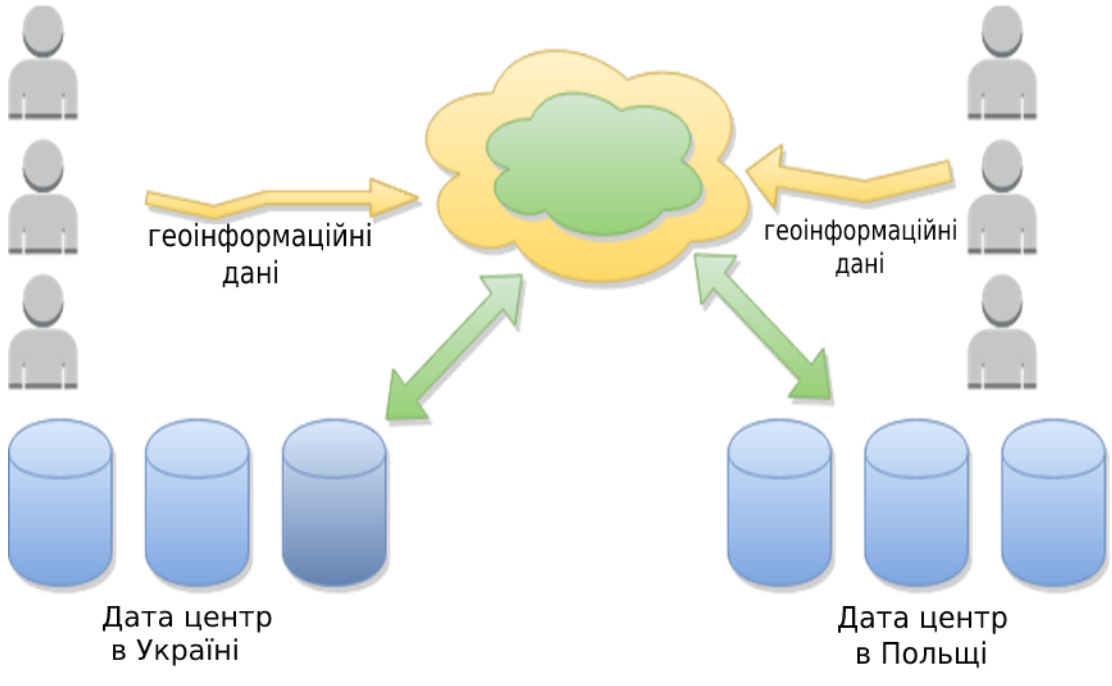



Рисунок 1.3 – Локалізація даних

Розташування даних в Україні дає ще одну важливу перевагу в зниженні витрат на зв'язок в системі. Весь вхідний трафік з моніторингового парку в Україні спрямовується до локального вузла обробки даних в країні, а не в умовний основний вузол. Це значно зменшує робоче навантаження і покращує час відгуку всієї системи. Мільярди транспортних засобів можна ефективно контролювати з високою доступністю, масштабованістю та високим ступенем відмовостійкості.

## 1.3 Моніторинг в реальному часі

Моніторинг автотранспорту в режимі реального часу - це справді цікава сфера технологій та їхнього розвитку. Це дає багато переваг у багатьох дисциплінах. Система відстеження дозволяє прораховувати найефективніші маршрути , зменшити кількість простоїв на дорогах та планувати подорожі більш ефективно . Також система може дозволити простежувати споживання палива/заряду що позволяє оптимізувати витрати та продовжити термін експлуатації транспортного засобу ,як показано на рисунку 1.4.


Рисунок 1.4 – Застосування моніторингу транспорту для відображення швидкості та напрямку

Також вона сприяє забезпеченню покращення безпеки на дорогах.Система може проводити моніторинг пересування транспортних засобі та виявляти потенційно небезпечні ситуації за рахунок опрацювання таких параметрів як : швидкість ,напрям руху ,час та місцеположення транспорту. Це допоможе уникнути небажаних аварій , знайти втрачений або викрадений автомобіль та контролювати дотримання правил дорожнього руху водіями,а також це допоможе в покращенні транспортної інфраструктури в цілому(зменшити кількість заторів до можливого мінімуму за рахунок аналізу пересування транспорту і належного конструювання ремонту доріг).

Система моніторингу транспорту застосовується в багатьох сферах як показано на рисунку 1.5


Рисунок 1.5 – Сфери застосування моніторингу автомобільного транспорту

За рахунок обробки вище названих даних ,система також може моніторити пересування пасажирського транспорту тим самим оптимізуючи процес формування розкладу руху. Пасажири зможуть в режимі реального часу відслідковувати пересування автобусів/трамваїв тощо і бачити прогнозований час прибуття ,що ,в свою чергу, дозволить пасажирам зменшити час очікування на зупинках і витратити звільнений час з користю.

З вищесказаного можна зробити висновок що система відстеження транспортних засобів є доволі потужним інструментом ,що може допомогти в вирішенні великої кількості проблем ,напряму пов'язаних з життям громадян ,та підвищити ефективність транспортної інфраструктури

## 1.4 Висновки. Постановка задачі

Система моніторингу транспорту в реальному часі має основний фокус на обробці великих обсягів даних про географічне розташування з використанням розподіленої кластерної технології. Вона продемонструє здатність безперешкодно обробляти велику кількість запитів на визначення географічного розташування, що надходять з різних GPS-пристроїв, таких як мобільні телефони, GPS-датчики тощо.

Через часові обмеження та технічні проблеми, система зосереджується лише на реалізації на стороні сервера. Реалізація на стороні клієнта, тобто збір інформації про географічне розташування, не реалізована, однак для демонстрації дані про географічне розташування будуть змодельовані з клієнтської програми з використанням навчального набору даних про географічне розташування.

З боку юзер інтерфейсу буде розроблено просту інтерактивну веб-сторінку на основі карти Google. Інтерфейс ,в свою чергу,буде простий, проте надаватиме основні типи фільтрів, такі як фільтр на основі діапазону дат, фільтр на основі географічного радіусу, фільтр на основі відстані та фільтр на основі адреси. Цього

було б достатньо, щоб продемонструвати моніторинг транспорту в реальному часі.

Технічно проект буде побудований на основі повноцінної хмаро орієнтованої веб-архітектури, щоб проект залишався відкритим для розвитку..

# 2. АНАЛІЗ НАЯВНОГО ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ СИСТЕМИ МОНІТОРИНГУ ТРАНСПОРТНИХ ЗАСОБІВ У РЕАЛЬНОМУ ЧАСІ

Основною метою цього проекту є вивчення та аналіз існуючих рішень, доступних для моніторингу автомобільного транспорту. Воно визначає різні проблеми моніторингу транспорту в режимі реального часу, існуючі обмеження існуючих рішень, їх надійність, масштабність, вартість, а також різні вузькі місця, притаманні цим рішенням, які зберігаються в них.

## 2.1 Огляд можливих рішень

Допустимо понад 1,2 мільярда транспортних засобів одночасно експлуатуються по всьому світу. Уявіть, що з 1,2 мільярда транспортних засобів потрібно моніторити сотні тисяч автомобілів. Як ви їх відстежуєте? Розглянемо простий приклад. Кожен автомобіль оснащений системою GPS-моніторингу з функцією зйомки відео з періодичністю 5 хвилин і функцією зйомки геолокації з періодичним інтервалом 15 секунд. Тоді кількість відеозаписів за годину становить 12, а кількість подій геолокації за годину - 240.

Допустимо понад 1,2 мільярда транспортних засобів одночасно експлуатуються по всьому світу. Уявіть, що з 1,2 мільярда транспортних засобів потрібно моніторити сотні тисяч автомобілів. Як ви їх відстежуєте? Розглянемо простий приклад. Кожен автомобіль оснащений системою GPS-моніторингу з функцією зйомки відео з періодичністю 5 хвилин і функцією зйомки геолокації з періодичним інтервалом 15 секунд. Тоді кількість відеозаписів за годину становить 12, а кількість подій геолокації за годину - 240.

Тепер, після отримання статистичних даних, нас цікавлять, насамперед, 2 фактори:

1.Масштабованість: Чи буде система працювати безперебійно, коли кількість транспортних засобів, що надсилають геоінформацію, збільшиться до 80 000 з початкових 20 000.

2.Продуктивність: Як система поводитиметься під великим навантаженням? Чи буде вона швидко реагувати на запит користувача, чи зависне на кілька годин, а потім відключиться?

Що ж, виглядає цікавою задачею. Я підійшов до неї традиційним способом. Використовуючи реляційну БД - MYSQL і Apache Tomcat сервер працював виключно добре, коли кількість одночасних запитів до сервера була близько 20 000. Коли я збільшив навантаження понад цю межу, з'явилися непередбачувані результати. Сервер став не реагувати на запити, а час відповіді виявився занадто високим. Публікація одного запиту займає більше секунд часу. Деякі запити не виконуються, а деякі виконуються з помилками. Беручи до уваги всі ці фактори, стало необхідним подумати про альтернативний підхід, який міг би впоратися з кількістю одночасних запитів, що перевищують цей ліміт.

Розподілена система моніторингу автомобілів в режимі реального часу призначена для моніторингу великої кількості машин в режимі реального часу шляхом збору інформації про географічне розташування, зокрема, широту, довготу та поточний час. Сервер отримуватиме ці дані і зберігатиме їх у певному постійному сховищі. Збережена інформація буде використовуватися користувачем на передній консолі для моніторингу поточного місцезнаходження автопарку на картографічному сервісі Google.

Оскільки транспортні засоби знаходяться під моніторингом, можна відстежити небезпечне водіння, перевищення швидкості та всі інші ризиковані ситуації. Це допомагає підвищити безпеку людей, навчаючи водіїв різним правилам безпеки. Крім того, шахрайські дії, випадкові пошкодження тощо... можна уникнути за допомогою належних заходів безпеки, наприклад, шляхом надсилання тривожного сигналу з пристрою на сервер.

На ринку існує багато рішень для моніторингу автопарку. Вони чудово справляються з моніторингом транспортних засобів на основі рішень на базі GPS. Однак жодна з організацій не має рішення, яке б відповідало вимогам високого навантаження на сервер. Поточні рішення працюють на одних традиційних http-серверах високого класу, таких як Apache tomcat, Jetty, IBM Http-сервер і т.д., що працюють з реляційними базами даних, такими як MYSQL або Oracle. Коли сервер стикається з одночасними запитами, що перевищують двадцять тисяч за один раз, він починає демонструвати непередбачувану поведінку зі збоями та затримками.

На ринку існує багато рішень для моніторингу автопарку. Вони чудово справляються з моніторингом транспортних засобів на основі рішень на базі GPS. Однак жодна з організацій не має рішення, яке б відповідало вимогам високого навантаження на сервер. Поточні рішення працюють на одних традиційних http-серверах високого класу, таких як Apache tomcat, Jetty, IBM Http-сервер і т.д., що працюють з реляційними базами даних, такими як MYSQL або Oracle. Коли сервер стикається з одночасними запитами, що перевищують двадцять тисяч за один раз, він починає демонструвати непередбачувану поведінку зі збоями та затримками

Teletrack, піонер у сфері моніторингу автотранспорту зазначає, що відстежувати автомобілі в межах певного автопарку майже неможливо, тому в такі програми, як GPS-моніторинг, варто інвестувати, щоб гарантувати, що безпека залишається пріоритетом для цих підприємств, зменшуючи високі витрати та підвищуючи безпеку дорожнього руху.

Рішення, які вони пропонують, занадто дорогі і не гарантують 100% відстеження транспорту на дорозі. За відстеження одного транспортного засобу вони стягують плату в доларах за день. Це не може бути доступним рішенням для тих, хто економічно менш стабільний, щоб платити таку високу ціну, як, наприклад, державні організації, що надають послуги. Звідси випливає необхідність скорочення витрат за рахунок оптимізованого впровадження

технологічних досягнень, які забезпечили б роботу рішення з використанням звичайного апаратного забезпечення, майже не крихкого і з дуже низькими витратами на обслуговування.

Intel представила рішення в якості підтвердження концепції в своїй технічній записці "Інтелектуальна система управління автопарком Intel Atom E3827 [IFMSI]", технологія якої підвищує ефективність управління автопарком за рахунок зменшення робочого навантаження та загальних обсягів перевезень. Вона використовує процесор Intel E3827 та GPS-пристрій, який підтримує управління даними, телематику, мобільні додатки, рекламу на основі визначення місцезнаходження тощо. Однак, це рішення не є життєздатним для економічно нестабільних організацій, оскільки це інтегроване рішення, яке коштує досить дорого.

Bynx, компанія-постачальник рішень для транспортних засобів, у своєму технічному документі наводить чудовий приклад використання "Клубів автопрокату" - наступної революції в управлінні транспортними засобами (TNRVM), яка завдяки вдосконаленню системи відстеження транспортних засобів дозволила з'явитися автомобільним клубам у Великобританії та Австралії. Оператори клубів можуть відстежувати використання автомобілів членами клубу і розраховувати оплату відповідно до того, скільки вони проїхали.

У статті "Система відстеження та блокування транспортних засобів на основі GSM та GPS [VTLS]" представлено роботу з відстеження викраденого транспортного засобу за допомогою GPS та GSM-технологій.

Багато рішень, які були впроваджені до цього часу, або в першу чергу призначені для вирішення окремих випадків використання на стороні клієнта, наприклад, трекер з підтримкою GPS та напрямки його вдосконалення, або вони в першу чергу намагаються вирішити проблему малого бізнесу, яка добре зарекомендувала себе в галузях з рішеннями на основі реляційних баз даних. Оскільки обсяги даних зростають з кожним днем, попередні тенденції рішень для моніторингу на стороні сервера стають застарілими і потребують окремого

сервера. Це змінило тенденцію до розробки рішень для моніторингу у світі великих даних на основі існуючих реляційних баз даних

Ефективна реалізація розподіленої архітектури та паралельних обчислень вимагає більш концентрованих зусиль, часу та висококваліфікованих ресурсів. Початкова вартість розробки данної системи набагато вища, ніж у централізованих рішень для моніторингу. Вона не підходить для невеликих організацій, що мають парк з десятків або двадцяти комп'ютерів. Її реальна цінність може бути реалізована тільки на великих підприємствах і в державних або громадських організаціях, де одночасно збираються величезні обсяги геолокаційних даних з сотень автопарків.

Час відгуку та затримка є ще одним важливим фактором, який необхідно враховувати при реалізації системи. Рішення для моніторингу даремне, якщо час відгуку не може бути досягнутий в межах, близьких до реального часу. Зі збільшенням кількості геолокаційних даних, що збираються в секунду з N автомобілів, завжди бажано, щоб час відгуку для передачі геоданих на сервер або отримання поточної статистики моніторингу на консолі був меншим, однак це є основною проблемою, з якою повинна справлятися розподілена система.

У статті "Система відстеження та блокування транспортних засобів на основі GSM та GPS [VTLS]" представлено роботу з відстеження викраденого транспортного засобу за допомогою GPS та GSM-технологій.

Багато рішень, які були впроваджені до цього часу, або в першу чергу призначені для вирішення окремих випадків використання на стороні клієнта, наприклад, трекер з підтримкою GPS та напрямки його вдосконалення, або вони в першу чергу намагаються вирішити проблему малого бізнесу, яка добре зарекомендувала себе в галузях з рішеннями на основі реляційних баз даних. Оскільки обсяги даних зростають з кожним днем, попередні тенденції рішень для моніторингу на стороні сервера стають застарілими і потребують окремого сервера. Це змінило тенденцію до розробки рішень для моніторингу у світі великих даних на основі існуючих реляційних баз даних

Ефективна реалізація розподіленої архітектури та паралельних обчислень вимагає більш концентрованих зусиль, часу та висококваліфікованих ресурсів. Початкова вартість розробки данної системи набагато вища, ніж у централізованих рішень для моніторингу. Вона не підходить для невеликих організацій, що мають парк з десятків або двадцяти комп'ютерів. Її реальна цінність може бути реалізована тільки на великих підприємствах і в державних або громадських організаціях, де одночасно збираються величезні обсяги геолокаційних даних з сотень автопарків.

Час відгуку та затримка є ще одним важливим фактором, який необхідно враховувати при реалізації системи. Рішення для моніторингу даремне, якщо час відгуку не може бути досягнутий в межах, близьких до реального часу. Зі збільшенням кількості геолокаційних даних, що збираються в секунду з N автомобілів, завжди бажано, щоб час відгуку для передачі геоданих на сервер або отримання поточної статистики моніторингу на консолі був меншим, однак це є основною проблемою, з якою повинна справлятися розподілена система.

1. Неоднорідність – Система повинна обробляти дані, що надходять з різних джерел. Джерелами геолокаційних даних можуть бути різні мобільні платформи, такі як Android, IPhone або BlackBerry, пристрої з підтримкою GPS на автомобілях, домашні планшети і т.д.

2. Прозорість – Система повинна приховувати фонову складність системи, додаючи рівень абстракції. Користувач не повинен знати про фонові процеси, що виконуються .

3. Паралелізм – Коли декілька користувачів отримують доступ до одного і того ж ресурсу, система повинна бути достатньо розумною, щоб система не потрапляла в стан "мертвої точки" і не ставала нестабільною.

4. Безпека – Надійна система автентифікації та авторизації повинна бути ввімкнена на рівні операційної системи та на рівні додатків. Всі підлеглі та головні вузли повинні бути захищені, щоб запобігти крадіжці, втраті або пошкодженню даних.

5. Маштабованість – Коли навантаження зростає з великими обсягами геоданих, система повинна автоматично масштабуватися. У системі повинен існувати балансувальник навантаження, який буде контролювати процесор, пам'ять та інші ресурси.

Кіберфізична система моніторингу транспорту в реальному часі складається з різних інструментів і компонентів, які інтегровані між собою для виконання поставленого завдання. Механізм розподіленої черги - Kafka здатний ставити в чергу безліч користувачі та обслуговувати потоки з черги. Система обчислень у реальному часі - Apache Storm здатна обчислювати і обробляти необмежену послідовність потоків паралельно в режимі, близькому до реального часу. Розподілена пошукова система - Elastic search це ефективний інструмент на основі Lucene, який може отримувати, зберігати, оновлювати або видаляти велику кількість запитів за один раз. Крім того, разом з ним використовується багато інших інструментів і компонентів, які можна узагальнити як показано нижче.

Метрики є корисними показниками для вимірювання якості роботи. Вони дозволяють виявити недоліки в поточній роботі та з'ясувати важливу кореляцію між якістю та показниками, але це не є єдиним достатнім фактором для оцінки. Під час розробки проекту та після його реалізації будуть збиратися такі метрики

Таблиця 2.1 – опис програмних утиліт

| Інструменти/компоненти | Короткий опис |
|---|---|
| Kafka | Високопродуктивний брокер повідомлень. Розподілена система черг з узгодженістю та високою доступністю. Краща за AcitveMQ в контексті забезпечення високої пропускної здатності. |
| Apache storm | Розподілена обчислювальна система реального часу. Заявлено, що вона може обробляти мільйони запитів за секунду.Це високопродуктивний, надійний обчислювальний рушій реального часу |

Продовження таблиці 2.1

| | |
|---|---|
| Zookeeper | Це централізований сервіс для обслуговування інформації конфігурації. Забезпечує розподілену синхронізацію. Використовується як розподілений координаційний сервіс для підтримки станів Kafka та Apache storm у кластері. |
| Elastic Search | Це вільне від схем сховище документів на основі JSON, яке діє як пошуковий сервер. Надає RESTFUL-інтерфейси на основі Java для різних CRUD-функцій. Він працює в кластерному або локальному режимі. Час відгуку на запити досить швидкий. |
| Http Server Apache Tomcat | Це веб-сервер з відкритим вихідним кодом і контейнер сервісів, розроблений компанією Apache Software Foundation. Це найнадійніший і найпростіший у використанні http-сервер. Веб-сервіси RESTFUL будуть розгорнуті та доступні через цей сервер. |
| Google Maps | Google maps - це технологія веб-картографії, яка використовується в браузері для відображення географічного розташування різних інтерактивних наборів маркерів, шляхів та вмісту інформаційних вікон. Альтернативою могли б бути карти Bing, однак через їхню більшу популярність, простоту у використанні та відкриті API java-скриптів, було обрано саме її. |
| Основна мова програмування | JAVA |

Кінець таблиці 2.1

| | |
|---|---|
| Мова програмування інтерфейсу | JAVASCRIPT, JQUERY |
| JConsole | Це графічний інструмент моніторингу на основі Java, який використовується для моніторингу віртуальної машини Java та Java-додатків на віддалених або локальних машинах |

1. Пропускна здатність: Вимірює кількість відправлених/оброблених запитів за секунду через систему. В основному, збираються дані про кількість запитів, що надсилаються за секунду від Кафки, і запитів, що обробляються за секунду від шторму.

2. Використання процесора: Відсоток навантаження на процесор буде відстежуватися під час максимального навантаження. Середнє навантаження на процесор, мінімальне навантаження на процесор, максимальне навантаження на процесор і т.д.. Статистика буде збиратися та зберігатися.

3. Мережа та вхід/вихід: Для оптимізації продуктивності буде збиратися середня швидкість вводу/виводу, використання дисків, пам'яті та мережевого трафіку в різних системах.

4. Практичність: Передня частина інтерфейсу користувача періодично аналізується, щоб зробити його зручним для користувача, що складається з чітких шляхів, кольорів і маркерів на маршруті карти для моніторингу транспорту.

5. Витрати, час та ресурси: Запропонована робота потребує кропіткої роботи, глибоких знань в архітектурі розподілених систем, невеликої кількості ресурсів в розподілених системах та виділеного періоду часу протягом семестру. Вся статистика, пов'язана з витратами, часом і ресурсами, буде періодично зведена в таблицю і оцінена.

Як підсумок, розподілені рішення для моніторингу транспорту та пов'язані з ними роботи є обмеженими в дослідженнях. Реалізовані до цього часу рішення базуються на традиційній системі моніторингу транспорту, яка залежить

виключно від одного автономного сервера з різними притаманними йому обмеженнями, такими як масштабованість і надійність.

Існують різні ключові комунікаційні протоколи, які використовуються при розробці будь-якого рішення в розподіленому середовищі. Обробка великої кількості запитів одночасно та їх паралельна обробка є складною задачею. Для вирішення цієї задачі вводяться системи черги повідомлень. Повідомлення, що стоять в черзі, повинні оброблятися з низькою затримкою, для чого необхідна паралельна обчислювальна система, і, зрештою, розподілена система зберігання даних, що дозволить реалізувати різні переваги розподіленої системи зберігання даних.

Система керування чергою повідомлень надає різноманітні послуги, такі як створення повідомлень, створення черг, ініціалізація процесів відправника та одержувача, а також надання засобів для надсилання та отримання повідомлень. Вона полегшує створення повідомлення і заповнення його даними. Повідомлення складається із заголовної інформації, такої як розмір повідомлення, час закінчення терміну дії повідомлення та пріоритет повідомлення. Він також надає засоби для створення черги і, за бажанням, для асоціювання параметрів з чергою, таких як довжина черги (тобто, максимальна кількість повідомлень, яку може утримувати черга). Механізм зв'язку може бути синхронним або асинхронним.

В синхронному режимі одержувач чекає на повідомлення від відправника, яке є заблокованим по своїй суті. В той час як при асинхронному протоколі зв'язку одержувач продовжує виконувати іншу задачу і отримує сповіщення про отримання повідомлень асинхронно.

Система черг застосовується таким чином, що коли велика кількість транспорту одночасно надсилає інформацію про географічне розташування на сервер, всі вони не можуть бути оброблені одночасно. Таким чином, нам потрібен механізм розподіленої черги, щоб їх можна було поставити в чергу на сервер і обробити.

Основними частинами системи керування чергою є:

1. Клієнтська частина черги повідомлень .
2. Об'єкти, керовані чергою повідомлень .
3. Адміністрування черги повідомлень .

Сервер черги повідомлень складається з брокера, який надає послуги передачі повідомлень для системи обміну повідомлень у черзі . Доставка повідомлень покладається на низку допоміжних компонентів, які забезпечують з'єднання, маршрутизацію та відправку повідомлень, стійкість, безпеку та ведення журналів. Сервер повідомлень може використовувати один або декілька екземплярів брокера у кластері з декількома брокерами

Брокер черги повідомлень підтримує зв'язок як з клієнтами програми черги повідомлень, так і з клієнтом керування чергою повідомлень. Кожен сервіс визначається типом і протоколом обслуговування

Після встановлення з'єднання між мережею та брокером за допомогою сервісів з'єднання, маршрутизатор повідомлень приступає до відправки повідомлень. Існує два підходи до відправки повідомлень: один з них - це базова відправка, при якій не береться до уваги відправка повідомлень-підтверджень. З іншого боку, підтвердження і транзакції працюють надійно

Apache Kafka - це розподілена видавнича система обміну повідомленнями за підпискою. Kafka - це швидкий, масштабований, розподілений за своєю структурою, розділений і реплікований сервіс журналу фіксацій.

Apache Kafka відрізняється від існуючих традиційних систем обробки повідомлень наступним чином:

1. Вона розроблена як розподілена система, яку дуже легко масштабувати.
2. Вона пропонує високу пропускну здатність як для публікації, так і для підписки.
3. Підтримує декілька передплатників і автоматично балансує споживачів під час збоїв.

4. Вона зберігає повідомлення на диску і, таким чином, може використовуватися для пакетного споживання, наприклад, ETL, на додаток до додатків реального часу.

Кафка підтримує стрічку повідомлень у розділах, які називаються темами. Процеси, які публікують повідомлення в темах Kafka, називаються продюсерами, а процеси, які підписуються на теми і обробляють стрічку опублікованих повідомлень, називаються споживачами. Kafka працює як кластер, що складається з одного або декількох серверів, кожен з яких називається брокером

Zookeeper - це централізований сервіс для зберігання інформації про конфігурацію, іменування, забезпечення розподіленої синхронізації та надання групових послуг. Kafka використовує Zookeeper для координаційних служб.

Zookeeper - це централізований сервіс для зберігання інформації про конфігурацію, іменування, забезпечення розподіленої синхронізації та надання групових послуг. Kafka використовує Zookeeper для координаційних служб.

На відміну від традиційних ітераторів, ітератор потоку повідомлень ніколи не завершується. Якщо в даний момент немає жодного повідомлення для споживання, ітератор блокується до тих пір, поки не будуть опубліковані нові повідомлення в цій темі. Kafka підтримує як модель доставки "точка-точка", у якій декілька споживачів спільно використовують одну копію повідомлення у черзі, так і модель "публікація-підписка", у якій декілька споживачів отримують власну копію повідомлення.

Кластер Kafka буде керувати сотнями або тисячами розділів. Розділи збалансовано розподіляються в кластері по колу, щоб уникнути зосередження всіх розділів з великими обсягами тем на невеликій кількості вузлів. Так само збалансовано лідерство, щоб кожен вузол був лідером для пропорційної частки своїх розділів.

Ущільнення журналу гарантує, що будь-який споживач, який перебуває в межах голови журналу, побачить кожне записане повідомлення; ці повідомлення

будуть мати послідовні зсуви. Порядок повідомлень завжди зберігається. Ущільнення ніколи не змінює порядок повідомлень, лише видаляє деякі з них.

Відправник надсилає дані безпосередньо брокеру, який є провідним для розділу, без жодного проміжного рівня маршрутизації. Щоб допомогти відправнику зробити це, всі вузли Kafka можуть відповісти на запит метаданих про те, які сервери активні і де знаходяться провідні для розділів теми в будь-який момент часу, щоб дозволити йому відповідним чином спрямувати свої запити. Клієнт контролює, до якого розділу відправляти повідомлення. Це може бути зроблено випадковим чином, реалізуючи своєрідне випадкове балансування навантаження, або це може бути зроблено за допомогою функції семантичного розбиття. Ми надаємо інтерфейс для семантичного розбиття на розділи, дозволяючи користувачеві вказати ключ для розбиття і використовувати його для хешування до розділу (також є можливість перевизначити функцію розбиття, якщо це необхідно). Наприклад, якщо вибраним ключем є ідентифікатор користувача, то всі дані для даного користувача будуть надіслані до одного розділу. Це, в свою чергу, дозволить споживачам робити припущення про локальність споживання. Цей стиль розбиття на розділи явно призначений для того, щоб дозволити отримувачам обробляти дані з урахуванням місцезнаходження.

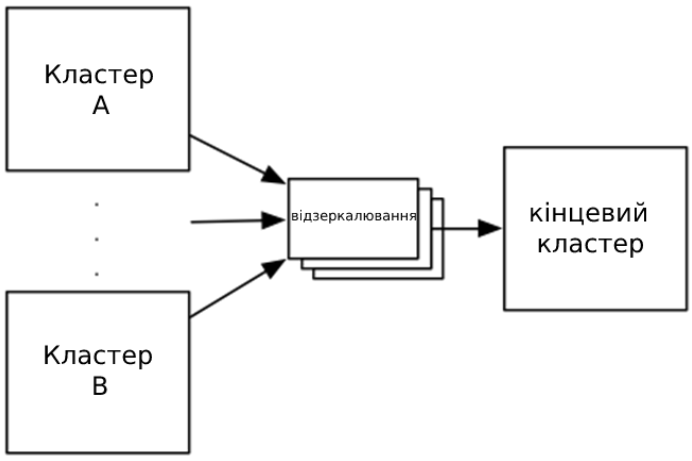



Рисунок 2.1 – відзеркалювання данних в kafka

Процес реплікації даних між кластерами Kafka називається "дзеркалюванням". Рушій Kafka складається з інструменту, який зчитує дані з одного або декількох кластерів-джерел і записує їх у кластер-приймач, як показано на рисунку 2.1.

Найпоширенішим випадком використання цього типу дзеркального відображення є створення копії в іншому центрі обробки даних. Для збільшення пропускної здатності та відмовостійкості можна запустити багато процесів дзеркалювання (якщо один процес зупиниться, інші візьмуть на себе додаткове навантаження).

Ефективна обробка та збереження даних є критично важливими в розподіленому середовищі. Завжди варто обробляти запити з низькою затримкою та високою пропускною здатністю. Для досягнення цієї мети потрібен ефективний механізм обчислень у реальному часі.

Обчислювальні системи реального часу стають все більш поширеними з кожним днем. Деякі з прикладів обчислювальних систем реального часу - це системи управління повітряним трафіком, навігаційні системи, мережеві мультимедійні системи, системи командного управління, авіаційні системи і т.д.. Розглянемо приклад системи прогнозування погоди - вона надсилає мільярди сигналів в один момент для того, щоб відстежувати погодні звіти в реальному часі по всьому світу. Однак всі вони не можуть бути оброблені миттєво через обмеженість обчислювальних ресурсів та інші технологічні фактори. Нам потрібна модель системи, яка може обробляти велику кількість запитів майже в реальному часі. Загалом системи реального часу можна розділити на 2 типи - системи жорсткого та м'якого режиму роботи. Пропуск строків у системах жорсткого режиму є катастрофічним, а в системах м'якого режиму може призвести до значних збитків. Система моніторингу транспорту в реальному часі орієнтована на безперебійну обробку геоінформації, отриманої в режимі реального часу від мільярдів автомобілів, що працюють в цей момент.

Storm - це фреймворк розподілених обчислень для обробки потоків даних про події, започаткований компанією з маркетингової аналітики, яку Twitter купив у 2011 році. Невдовзі Twitter виклав проект з відкритим вихідним кодом на GitHub, але згодом Storm переїхав до інкубатора Apache і став проектом вищого рівня Apache у вересні 2014 р. Іноді Storm також називають платформою Hadoop для обробки даних у реальному часі. Storm дозволяє легко і надійно обробляти необмежені потоки даних, роблячи для обробки в реальному часі те, що Hadoop робив для пакетної обробки.

Twitter використовує Hadoop для пакетної обробки та Storm для обробки в режимі реального часу. Storm був розроблений для виконання досить складної агрегації даних, які надходять через потік, перш ніж вони будуть відправлені назад до пакетної системи для подальшого аналізу.

Storm підходить для комплексної обробки. За допомогою Storm можна виконувати широкий спектр складних агрегацій, поки дані проходять через систему. Простий приклад - визначення найбільш популярних тем у твіттері. За допомогою подієво-керованого підходу ми можемо припустити, що маємо поточний стан, і просто змінити його, щоб оновити список популярних тем. На противагу цьому, пакетна система повинна зчитувати весь набір слів, перераховувати і переупорядковувати слова для кожного оновлення. Це робить обчислення набагато швидшими, ніж у системах пакетної обробки.

Асинхронні події, що надходять до Storm, розпаралелюються для обчислень, що допомагає Storm обробляти події майже в реальному часі.

Storm розроблений для масового масштабування, підтримує відмовостійкість з підходом до процесів "швидкий збій, автоматичний перезапуск" і пропонує надійну гарантію того, що кожен кортеж буде оброблено. За замовчуванням Storm гарантує обробку повідомлень "принаймні один раз", але також пропонує можливість реалізувати обробку "точно один раз". Storm має багато варіантів використання: аналітика в реальному часі, машинне навчання онлайн, безперервні обчислення, розподілений RPC, ETL, моніторинг драйверів

та багато іншого. Storm швидкий: за результатами бенчмарку він обробляє понад мільйон кортежів за секунду на кожен вузол. Він масштабований, відмовостійкий, гарантує, що ваші дані будуть оброблені, простий в налаштуванні та експлуатації. Storm кластер працює на топологічній основі. Топологія обробляє повідомлення або потоки даних вічно або до тих пір, поки не буде зупинена.

У кластері Storm є два типи вузлів: головний вузол і робочі вузли. На головному вузлі працює робоча програма Nimbus, яка відповідає за розподіл коду по кластеру, призначення завдань машинам і моніторинг збоїв.

На кожному робочому вузлі запускається робот, який називається " Supervisor". Супервізор стежить за роботою, призначеною для його машини, і запускає та зупиняє робочі процеси за необхідності, виходячи з того, що йому призначив Nimbus. Кожен робочий процес виконує свою підмножину топології; запущена топологія складається з багатьох робочих процесів, розподілених на багатьох машинах.

Zookeeper є координатором кластера Storm. Взаємодія між nimbus та робочими вузлами здійснюється через Zookeeper

Http-сервери - це засоби обробки запитів за допомогою протоколу HTTP, який є основним мережевим протоколом, що використовується для розповсюдження інформації у всесвітній мережі. Http-сервери обробляють, зберігають чи доставляють запити на інформацію клієнту. Клієнтом може бути простий браузер або пошуковик http. Оператор користувача на клієнті ініціює зв'язок, роблячи запит до певного ресурсу Http-сервера, а сервер відповідає змістом запиту або повідомленням про помилку.

Система використовує RESTFUL API на основі java spring для зв'язку з ресурсами сервера за допомогою Apache Tomcat - Http-сервера.

HTTP-сервер Apache - це HTTP-сервер з відкритим вихідним кодом для сучасних операційних систем, включаючи UNIX і Windows NT. Це безпечний, ефективний і гнучкий сервер, який надає HTTP-сервіси, синхронізовані з поточними стандартами HTTP.

Tomcat розроблений як швидка та ефективна реалізація специфікації сервлетів java. При цьому значну увагу було приділено продуктивності Tomcat, і зараз він знаходиться на одному рівні з іншими сервлет-контейнерами, в тому числі комерційними.

Сервер Apache tomcat складається з наступних компонентів:

1. Сервер представляє весь контейнер. Tomcat надає реалізацію інтерфейсу сервера за замовчуванням, яку користувачі рідко налаштовують.

2. Сервіс – це проміжний компонент, який знаходиться всередині сервера і пов'язує один або кілька коннекторів з одним рушієм.

3. Рушій являє собою конвеєр обробки запитів до сервісів. Сервіс може мати декілька коннекторів, на основі яких рушій отримує та обробляє всі запити від цих коннекторів і повертає відповідь на відповідний коннектор для передачі клієнту.

4. Хост – це зв'язок мережевого імені, наприклад, www.rtms.com, з сервером Tomcat. Движок може містити декілька хостів, а елемент Host також підтримує мережеві псевдоніми, такі як rtms-test.com

5. Коннектори відповідають за зв'язок з декількома клієнтами. Коннектор HTTP використовується для більшості HTTP-трафіку, особливо коли Tomcat працює як автономний сервер, а коннектор AJP, який реалізує протокол AJP, використовується при підключенні Tomcat до веб-сервера, такого як HTTPD-сервер Apache.

6. Контекст представляє собою веб-додатки або сервіси, розміщені в контейнері. Хост може містити декілька контекстів, кожен з яких має унікальний маршрут.

Простіше кажучи, хмарні обчислення - це спосіб зберігання та доступу до даних і програм через Інтернет замість жорсткого диска вашого комп'ютера.На рисунку 2.2 ілюстрована система хмарного сховища відносно пристроїв Хмара – це місце, де програми та сервіси можна безпечно зберігати та завантажувати. Хмарні обчислення мають важливе значення в наш час, оскільки:

1. Немає прямої потреби до обслуговування та керування з вашого боку

2. Фактично нескінченний за розміром , з чого слідує відсутність потреби турбувати про те ,що він вичерпає свій ресурс

3. Можливість отримати доступ з будь-якого місця де є достатньо сильне покриття для мобільного інтернету

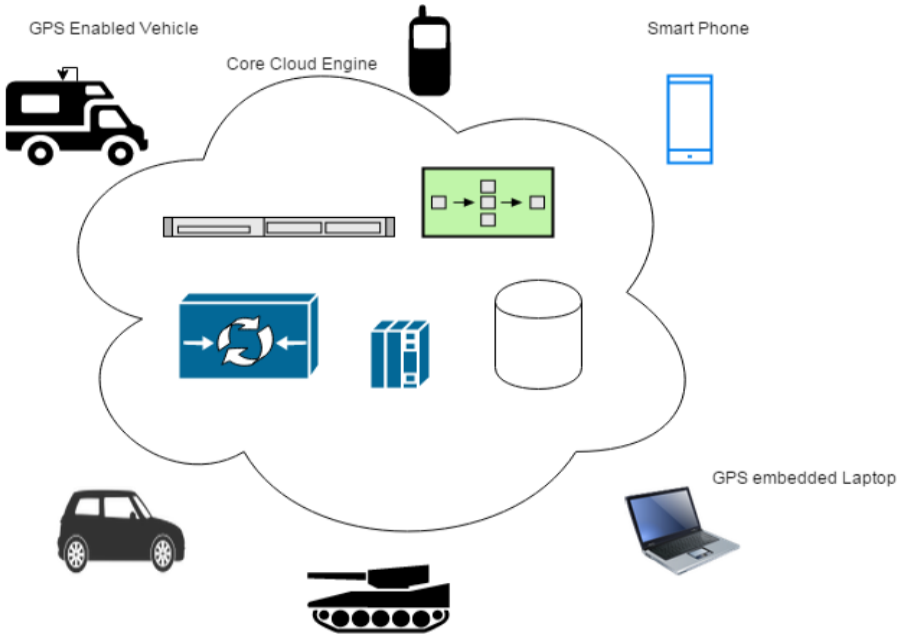



Рисунок 2.2 – Хмарне сховище

Кажучи про хмарні обчислення, виділяють три основні підходи:

1. Хмарні додатки (або програмне забезпечення як послуга) працюють на віддалених комп'ютерах "у хмарі", які належать іншим особам і підключаються до комп'ютерів користувачів через Інтернет і, відповідно, веб-браузер.

2. Платформа як послуга надає хмарне середовище з усім необхідним для підтримки повного життєвого циклу створення та реалізації веб-додатків (в хмарі) - без витрат та складнощів пов'язаних з придбанням та управлінням базовим обладнанням, софтом, підтримкою та хостингом.

3. Інфраструктура як послуга надає компаніям обчислювальні ресурси, включаючи сервери, мережі, сховища та місця в центрах обробки даних на умовах оплати за використання.

Переваги хмарних обчислень можна підсумувати наступним чином:

1. Досягнення ефекту масштабу - збільшення обсягів виробництва або продуктивності за допомогою меншої кількості людей. Витрати на одиницю продукції, проект або продукт знижуються.

2. Скорочення витрат на технологічну інфраструктуру. Підтримка легкого доступу до вашої інформації з мінімальними стартовими витратами. Оплата по мірі використання (щотижня, щокварталу або щороку), виходячи з попиту.

3. Глобалізація вашої робочої сили за низькою ціною. Люди з усього світу можуть отримати доступ до хмари, якщо у них є стабільне підключення до Інтернету.

4. Оптимізація процесів. Виконуйте більше роботи за менший час з меншою кількістю людей.

5. Зменшення капітальних витрат. Не потрібно витрачати значні кошти на обладнання, програмне забезпечення або ліцензійні платежі.

6. Покращення доступності. Ви маєте доступ у будь-який час і в будь-якому місці, що значно полегшує ваше життя!

7. Моніторинг проєктів більш ефективний. Залишайтеся в рамках бюджету та випереджайте терміни завершення циклу.

8. Необхідно менше витрат на навчання персоналу. Для виконання більшого обсягу роботи в хмарі потрібно менше людей, з мінімальною тривалістю навчання з питань апаратного та програмного забезпечення.

9. Мінімізуйте ліцензування нового програмного забезпечення. Ростіть та розвивайтеся без необхідності купувати дорогі ліцензії на програмне забезпечення або програми

10. Підвищення гнучкості. Ви можете змінювати напрямок без серйозних людських чи фінансових проблем на виході.

## 2.2 Висновок

Зважаючи на зростаючу тенденцію хмарних технологій та їхні переваги, система змодельована як розширювана та гнучка для підтримки архітектури хмарних обчислень. Система реалізована на основі архітектури SAAS(Software as service, варіант 1) ,тому розширюваність, масштабованість, доступність та моніторинг ніколи не будуть з обмеженнями.

Технологія SAAS в системі забезпечує більшу функціональність, надаючи послуги API на основі геолокації з хмаро орієнтованої архітектури. Оскільки сервіси розгорнуті в хмаро орієнтованій архітектурі, кінцевому користувачеві не потрібно знати або мати справу зі складністю внутрішніх базових сервісів системи моніторінгу. Кінцевому користувачеві надаються прості кінцеві точки на основі REST API для зв'язку з цими сервісами через Http. Таким чином, доступ до всіх сервісів здійснюється через Http-зв'язок. Будь-який кінцевий користувач, що підтримує Http-зв'язок, тепер може бути безпосередньо інтегрований з системою.

Наприклад, автомобіль з підтримкою GPS може надсилати інформацію про геолокацію з підключеного GPS-пристрою через HTTP. Так само користувач смартфона може відстежувати пристрій, написавши просту GPS-програму для надсилання інформації про геолокацію до хмарного сервісу ситеми через HTTP.

# 3. ПРОГРАМНА РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ПРОГРАМНО-ТЕХНІЧНОГО ЗАСОБУ ТА ВИСНОВКИ

## 3.1 Основна проблематика

Існує два основних проблемних моменти, які виникають при впровадженні рішення для моніторингу транспорту за допомогою існуючого традиційного підходу. Обробка великої кількості паралельних запитів та їх обробка в режимі реального часу та зберігання даних в якомусь постійному місці, яке є масштабним та ефективним при обробці запитів та повернення відповіді.

Необхідно підійти до розподіленої системи повідомлень з публікацією та відправкою повідомлень, яка має бути швидкою, розширюваною та стабільною для обробки великої кількості запитів, що надходять одночасно. Отже, для цієї мети найкраще підходить розподілений, секційний сервіс журналу реєстрації комітів, відомий як Kafka.

Система використовує Kafka для збереження каналів повідомлень у категоріях, відомих як томи. Процеси, які публікують повідомлення у томі Kafka, називаються відправниками. Процеси, які отримують повідомлення з томів і обробляють їх, називаються приймачами. Kafka працює як кластер, що складається з одного або декількох номерних серверів, які називаються брокерами.

Таким чином, на високому рівні відправники надсилають дані про геолокацію через мережу до кластеру Kafka, а отримувач їх обробляє. На діаграмі нижче показано високий рівень архітектури Kafka.

Що ж, Kafka вирішив нашу проблему, тримаючи мільйони потоків у реальному часі в черзі, які раніше не працювали через велику кількість одночасних запитів. Тепер наступний крок : Як нам отримувати та обробляти ці потоки в режимі реального часу? Нам потрібна розподілена система обчислень у реальному часі, яка дозволяє легко і надійно обробляти необмежені потоки даних.

Найнадійніший та найефективніший движок для обчислень у реальному часі, Apache Storm, є чудовим вибором для обробки мільйонів кортежів за секунду.

Система спирається на можливості Apache Storm, яка є масштабною, безвідмовною розподіленою обчислювальною системою з відкритим вихідним кодом, що дозволяє нам обробляти великі обсяги даних у реальному часі. Наш сценарій використання найкраще підходить для Storm, оскільки є велика кількість одночасних потоків даних про географічне розташування, отриманих від автомобілів з GPS в один момент. Він інтегрується з чергою, щоб отримувати повідомлення та обробляти їх у реальному часі.

Обчислення в реальному часі на Storm здійснюється за допомогою концепції, яка називається топологія. Топологія складається з схеми обчислень. Кожен вузол топології складається з обчислювальної логіки та зв'язку між різними вузлами, який вказує на потік даних між вузлами. Основною частиною абстракції в Storm є потік - необмежена послідовність потоків. Потік даних перетворюється на новий потік у вигляді розподіленого та надійного потоку за допомогою каналів та вузлів . Дані геолокації з сервісу обміну повідомленнями Kafka зчитуються з каналу. Вузол отримує вхідний потік данних, що надходить через канал або інший вузол. Вузол складається з логіки обробки та інших функцій, таких як фільтрація , агрегація , об'єднання потоків, зв'язок з базою даних і збереження результату в базі даних, тощо.

За потреби до системи можна підключити пересилання відео інформацію . В такому випадку виникає проблема в зберіганні та ефективному відтворенні файлів розміром в діапазоні від терабайт до петабайт. Час відгуку має бути мінімальним, незалежно від того, скільки даних зберігається на хмарі. Тому очевидним вибором для цього є вибір бази даних для зберігання документів NOSQL Виходячи з різних параметрів продуктивності, гнучкості та вимоги, щоб час відгуку був якомога меншим, найнадійнішим виявився Elastic search.

Система використовує Elastic search представляє собою високомасштабовану повнотекстову пошуково-аналітичну систему, яка дозволяє

зберігати, шукати та аналізувати великі обсяги даних майже в реальному часі. Це розподілена пошукова система та сховище документів, яка рівномірно розподіляє навантаження між кількома вузлами в кластерному середовищі. Кластер складається з ще одного вузла, який зберігає всі дані та забезпечує можливості об'єднаного індексування та пошуку по всіх вузлах(приклад архітектури наведено на рисунку 3.1). Він складається з таких основних компонентів:

1. Індекс – це набір документів зі схожими характеристиками. У нашому випадку геолокація - це індекс для даних про геолокацію.

2. Тип – кожен індекс може мати один або декілька типів, які вказують на логічну категорію індексу. У нашому випадку ми можемо вказати тип як розташування, щоб інформація про місцезнаходження для декількох держав потрапляла у власний логічний блок. Це значно покращує час пошуку за допомогою фільтра обмежень запиту, наприклад, тип = місцезнаходження XYZ.

3. Документ – це основна одиниця інформації, яка може бути проіндексована в кластері. Документ виражається у форматі JSON (Java script object notation), що робить його дуже гнучким в роботі, оскільки він є поширеним форматом.

4. Формат обміну даними в Інтернеті.

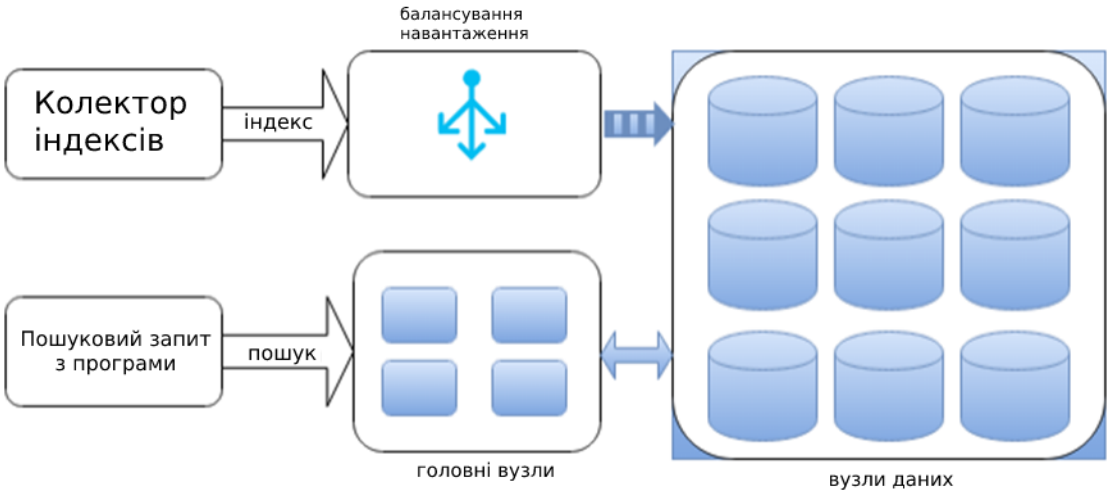



Рисунок 3.1 – Приклад архітектури “elastic search”

3.2 Реалізація про програмного засобу

Система складається з трьох основних компонентів :

1. Kafka.
2. Storm.
3. Elastic storm.

Механізм розподіленої черги повідомлень Kafka відповідає за формування черги з великої кількості запитів на інформацію про транспортні засоби, отриманих від SAAS комунікатора, і подає їх до механізму обробки в реальному часі. Storm вбудований в систему RTDFMS для обробки запитів на інформацію про транспортні засоби в режимі реального часу, аналізуючи геолокаційні дані. Він обробляє запит на геолокацію і отримує різні результати обчислень, такі як назва місця розташування з широти-довготи, вимірювання швидкості транспортного засобу і генерування робочих логічних символів. Результати обчислень, оброблені штормом, потім надсилаються на комунікатор SAAS, який, зрештою, зберігається в розподіленому сховищі Elastic Search.

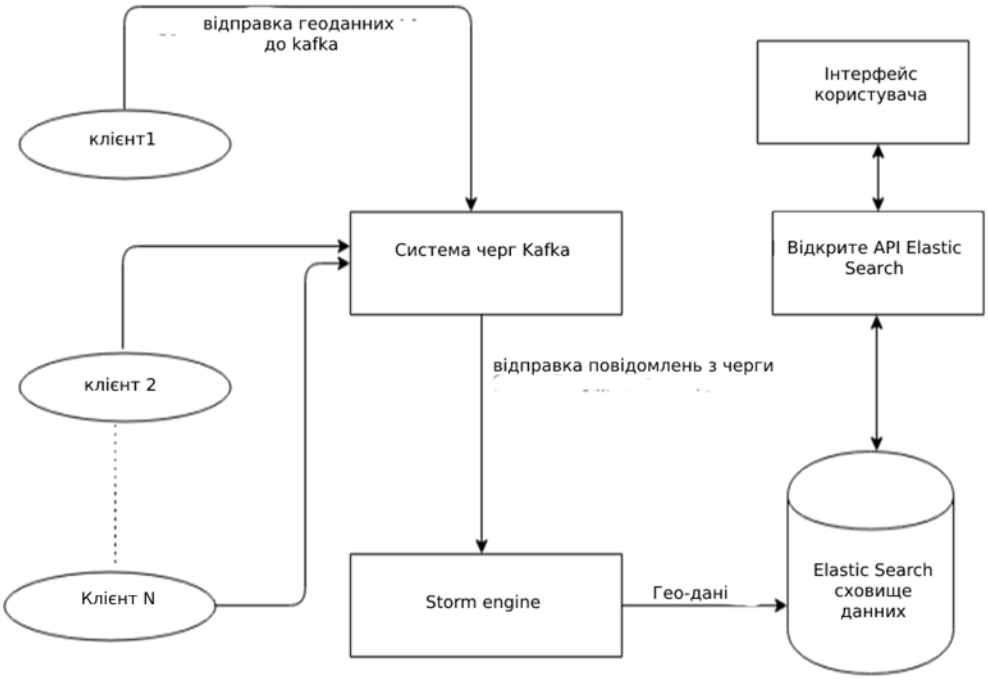



Рисунок 3.2 принцип роботи механізму системи

Як показано на схемі вище(рисунок 3.2), клієнт, тобто транспортний засіб з включеною системою GPS, періодично з певним інтервалом часу (наприклад, 15 секунд) надсилатиме інформацію про геолокацію до черги повідомлень - Kafka. Служба черги Kafka підтверджуватиме повідомлення і зберігатиме його у розподіленому кластері черги. Повідомлення з черги Kafka надходять паралельно через один або декілька каналів Strom Engine. Strom обробляє ці повідомлення у вигляді потоків і зберігає їх у сховищі Elastic search. Сховище доступне за допомогою різних REST API, які використовуються для запитів, отримання або оновлення даних геоінформації з метою відображення або обробки різної інформації про події в інтерфейсі користувача.

3.3 блоки майбутньої системи.

Цей розділ містить детальну інформацію про різні Rest API для реалізації системи . Ці блоки дозволяють клієнтській програмі на пристрої зберігати та маркувати відео заздалегідь визначеною метаінформацією, такою як номер машини, часова мітка, координати та будь-які події, що мають відношення до нього. Вони також дозволяють користувачеві шукати відео за метаданими та переглядати його вміст за допомогою централізованої консолі інтерфейсу користувача(таблиця 3.1, таблиця 3.2).

Таблиця 3.1 –додавання маршруту

| **Request URL:** /driver/route |
|---|
| Request Method: POST |
| Content-Type: Application/JSON |

Кінець таблиці 3.1 – додавання маршруту

Request Body:

```
{
  "customerId" : "1",
  "customerName" : "Dmytro Ryb",
  "address" : вулиця Козацька, 71-59, Славута, Хмельницька область, 30000",
  "pin" : "301209",
  "country" : "Ukraine",
  "city" : "Slavuta",
  "geoTag" : "50.297568, 26.869612",
  "pickupTime" : "1432368022000",
  "fleetNumber" : "1"
}
```

Таблиця 3.2 – додавання інформації про транспортні засоби

| **Request URL:** /driver/info |
|---|
| Request Method: POST |
| Content-Type: Application/JSON |

Request Body:

```
{
  "fleetNumber": "1",
  "driverName": "Petro P",
  "vehicleNumber": "AA0852ZA"
}
```

Таблиця 3.3 –завантаження відео з транспорту

| **Request URL:** /uploadvideo |
|---|
| Request Method: POST |

Content-Type: Application/JSON

Request Body:

```
{
    “fleetNumber” : “1”,
    “timestamp” : 1481213700,
    “latitude” :  50.297568,
    “longitude” : 26.869612,
    “videoName” : “1.avi”,
“eventType” : 1,
    “content” : ${BASE_64_ENCODED_VIDEO_DATA}
}
```

Таблиця 3.4 – Параметри до таблиці 3.3

| змінна | тип | опис |
|---|---|---|
| fleetNumber | string | Використовується для ідентифікації номеру транспорту, для якого завантажується відео. |
| timestamp | long | Цей параметр відображає час, коли було записано відео. |
| latitude | double | Цей параметр містить географічну широту місця, де було записано відео. |
| longitude | double | Цей параметр містить географічну довготу місця, де було записано відео. |
| videoName | double | Цей параметр містить назву відео. Формат відео – “мітка часу.avi” |
| eventType | integer | Цей параметр містить будь-яку подію, пов'язану з відео. Допустимі події: 1 - спокій, 2 - перевищення швидкості, 3 - затримка. |

Кінець таблиці 3.4 – Параметри до таблиці 3.3

| | | |
|---|---|---|
| content | string | Цей параметр містить фрагмент відео, закодований у base64. Пристрій повинен виконати кодування відеоконтенту в base64, перш ніж завантажувати його на сервер. |

Цей блок(таблиця 3.3) дозволяє додатку на пристрої завантажувати відео на сервер. Додаток також має надавати метадані до кожного відео. Ці метадані згодом використовуються для пошуку потрібного відео.

Доступ можна отримати за наступною адресою ${Project_BASE}/uploadvideo за допомогою методу http POST. API очікує в тілі http-запиту документ у форматі json, що містить метадані та відео.

Таблиця 3.5 – Завантаження інформації з транспорту

| **Request URL:** /uploadevent |
|---|
| Request Method: POST |
| Content-Type: Application/JSON |
| Request Body:<br>{<br>“fleetNumber” : “1”,<br>“timestamp” : 1481213700,<br>“latitude” : 50.297568,<br>“longitude” : 26.869612,<br>“eventType” : 1<br>} |

Таблиця 3.6 – параметри до таблиці 3.5

| змінна | тип | опис |
|---|---|---|
| fleetNumber | string | Використовується для ідентифікації номеру транспорту, для якого завантажується відео. |
| timestamp | long | Цей параметр відображає час, коли було записано відео. |
| latitude | double | Цей параметр містить географічну широту місця, де було записано відео. |
| longitude | double | Цей параметр містить географічну довготу місця, де було записано відео. |
| eventType | integer | Цей параметр містить будь-яку подію, пов'язану з відео. Допустимі події: 1 - спокій, 2 - перевищення швидкості, 3 - затримка |

Цей блок дозволяє програмі на пристрої публікувати події на сервері.(табилця 3.5). Події публікуються незалежно від відео, які також позначаються відповідними тегами, щоб ці події можна було використовувати для запуску будь-якої іншої дії на стороні сервера або для відображення в окремій консолі подій.

Додаток також повинен виводити метадані стосовно кожної події.

Доступ можна отримати за наступною адресою ${Project_BASE}/uploadevent за допомогою методу http POST. API очікує в тілі http-запиту документ у форматі json, що містить метадані та відео.

Таблиця 3.7 пошук

| **Request URL:** /search | | |
|---|---|---|
| Request Method: GET | | |
| Параметри URL: | | |
| параметр | тип | опис |
| fleetNumber | string | Цей параметр використовується для визначення номера автомобіля, який потрібно шукати.Якщо номер автомобіля не вказано, то будуть показані результати пошуку для всіх автомобілів, які є в наявності. |
| fromTime | | Цей параметр використовується для задавання часу з якого повинно починатися відтворення відеоряду |
| toTime | | Цей параметр використовується для задавання часу з якого повинно закінчуватися відтворення відеоряду |
| address | | Цей параметр використовується для вказівки адреси, за якою потрібно шукати автомобіль. API використовує параметр radius для зменшення маштабів пошуку |
| Radius | | Цей параметр використовується для вказівки межі радіусу, в межах якого потрібно шукати інформацію про транспорт |
| eventType | | Цей параметр використовується для позначення подій та ,відповідно, їх подальшої фільтрації |

Цей блок дозволяє здійснювати пошук відео на основі метаданих.(таблиця 3.7) Цей API буде використовуватися інтерфейсом консолі адміністратора, який дозволить користувачеві здійснювати пошук відео. Користувач зможе шукати за номером автомобіля, датою, часом, радіусом, геолокацією та маркером події.

Доступ до API можна отримати за наступною адресою URL${RTDFMS_BASE}/search, використовуючи http GET-метод. Необхідно щоб записані параметри були в форматі string .

Наприклад:

GET/search?fleetNumber=1&fromTime=01-Dec-2022 00:00:00&toTime=01-Dec-2022 23:59:59&address=&radius=&eventType=1

Таблиця 3.8 – пошук авто

| **Request URL:** /fleet/${fleet-id} |
|---|
| Request Method: POST |
| Content-Type: Application/JSON |
| Response Body:<br>{<br>“fleetid” : 1,<br>“driverName” : “Petro”,<br>“vehicleNumber” : “AA 8591 BA”<br>} |

Цей блок дозволяє користувачеві отримувати інформацію про конкретний транспортний засіб(таблиця 3.8). Наразі система зберігає ім'я водія та номер транспортного засобу як метадані для пошуку

Таблиця 3.9 – пошук відео

| **Request URL:** /fleet/${video-id} |
|---|
| Request Method: GET |

| Content-Type: Application/JSON |
|---|
| Response Body:<br>“fleetNumber” : “1”,<br>“timestamp” : 1418213700,<br>“latitude” : 50.297568,<br>“longitude” : 26.869612,<br>“videoName” : “1418212700.avi”,<br>“eventType” : 1,<br>“content” : ${BASE64DECODED_CONTENT}<br>} |

Цей блок дає можливість консолі адміністратора витягнути певне відео для перегляду.(таблиця 3.9)

## 3.4 Географічна картографія

Інтерфейс користувача для системи розроблений з використанням простої інтерактивної картографічної технології для моніторингу та перегляду деталей геоданних для окремих авто. Інтерфейс було зроблено динамічним та інтерактивним з використанням нанесення інформації про геолокацію на карту в реальному часі, а також відеозаписів та інших подій, що надсилаються через клієнтську частину , встановлену в мережі. Карти Google були інтегровані для реалізації інтерфейсу за допомогою JavaScript та Html5.

Google Maps - це технологія веб-картографії, розроблена компанією Google. Вона забезпечує високочутливий, інтуїтивно зрозумілий картографічний інтерфейс із вбудованими детальними даними вулиць та аерокосмічних знімків. Крім того, в програму можна вбудувати елементи керування картою, щоб надати

користувачам повний контроль над навігацією по карті та відображенням даних вулиць та знімків.

Карти Google пропонують кілька сервісів, які можна інтегрувати у великі веб-додатки, такі як:

1. Планувальник маршрутів - це вказівки для водіїв, велосипедистів, пішоходів та користувачів громадського транспорту, які хочуть дістатися з одного місця в інше.

2. Програмний інтерфейс (API) Карт Google дає змогу адміністраторам веб-сайтів вбудовувати Карти Google у власний сайт, наприклад, у путівник по Києву або на сторінку громади.

3. Карти Google для мобільних пристроїв - це служба визначення місцезнаходження для автомобілістів, яка використовує дані Глобальної системи позиціонування (GPS) про місцезнаходження мобільного пристрою (за наявності), а також дані з бездротових і стільникових мереж.

4. Google Street View дозволяє користувачам переглядати і переміщатися по горизонтальних і вертикальних панорамних зображеннях вулиць різних міст по всьому світу.

Елементи Google Maps:

1. Маркер – ідентифікує місце розташування на карті. За замовчуванням маркер використовує стандартне зображення. Маркери можуть відображати користувацькі зображення, в такому випадку їх зазвичай називають " іконками". Маркери та іконки є об'єктами типу Маркер.

2.Інформаційне вікно – відображає вміст (зазвичай текст або зображення) у спливаючому вікні над мапою в заданому місці. Інформаційне вікно має область вмісту та суцільну основу. Вона прикріплена до вказаного місця на карті.

## 3.5 Інтерфейс користувача

Нижче наведено різні сценарії, в яких здійснюється моніторинг транспортних засобів

На рисунку 3.3 наведено домашню сторінку системи . На головній сторінці показані фіксовані маршрути всіх авто і динамічні маршрути авто на поточний день.

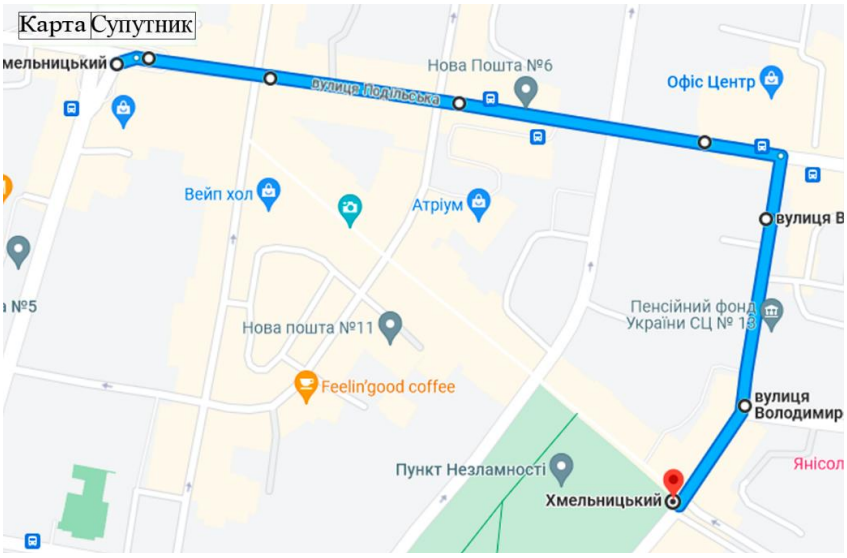



Рисунок 3.3 – Головна сторінка

Кожен автомобіль має постійний маршрут, по якому він повинен рухатися для виконання певного завдання. Система надає можливість додавати цей маршрут і відстежувати, чи працює машина по ньому, чи ні.(рисунок 3.4)

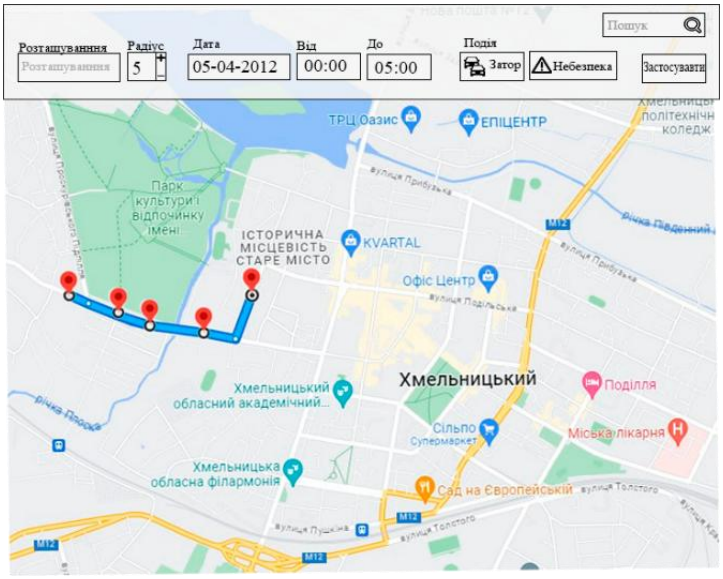



Рисунок 3.4 – Фіксований маршут

Знімок екрану нижче (рисунок 3.5) показує маршрут, пройдений транспортом в реальному часі. Кожна машина має початкове місцезнаходження, позначене червоним колом, і поточне місцезнаходження в реальному часі, позначене зеленим колом. Під час подорожі, окрім інформації про місцезнаходження, машина може фіксувати різні події, наприклад, відео з видом на дорогу, обідню перерву, поломку тощо.

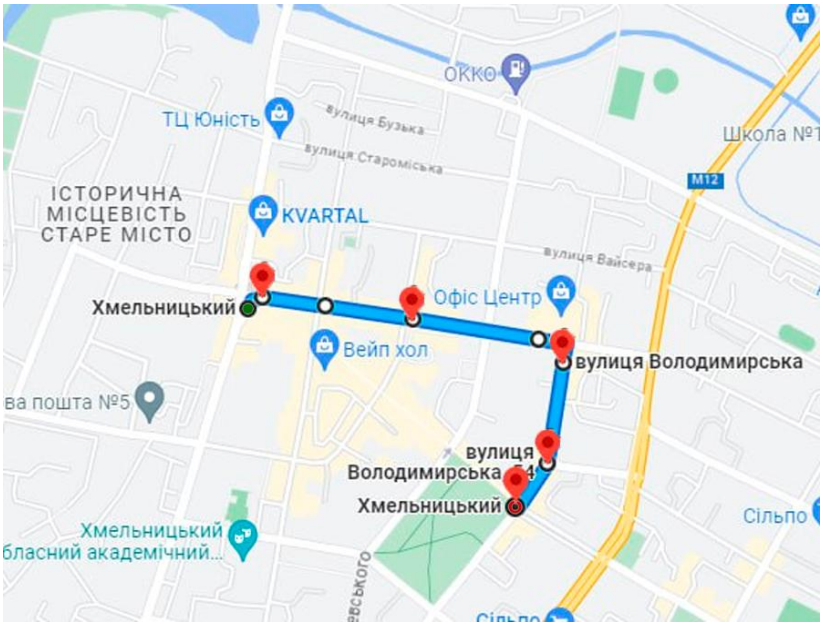



Рисунок 3.5 – Відстеження динамічного маршруту

Інтерфейс системи дає змогу відображати випадки простоїв, виявлені детектором під час керування автомобілем. Відповідна іконка вказує на данну подію. Для того, щоб переглянути події на мапі, необхідно активувати відповідну кнопку(як на рисунку 3.6).

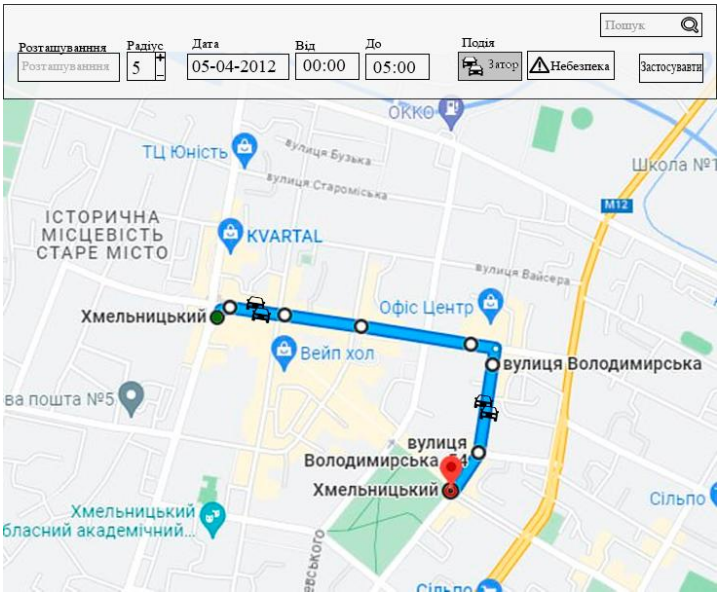



Рисунок 3.6 Подія – простій(затор)

Перевищення швидкості транспортного засобу може бути виявлено на основі геолокаційної інформації, зафіксованої протягом певного проміжку часу(рисунок 3.7)

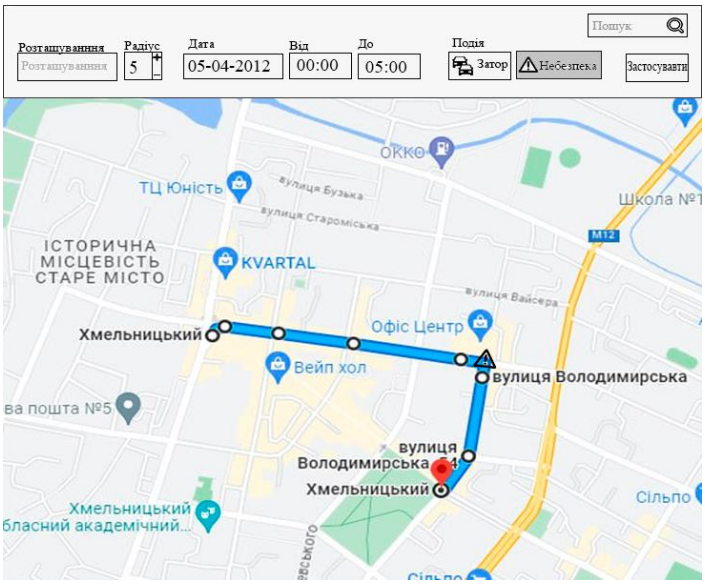



Рисунок 3.7 – Небезпечене керування авто

На наведеному нижче знімку екрана (рисунок 3.8) показано декілька фільтрів застосованих до одного маршруту.

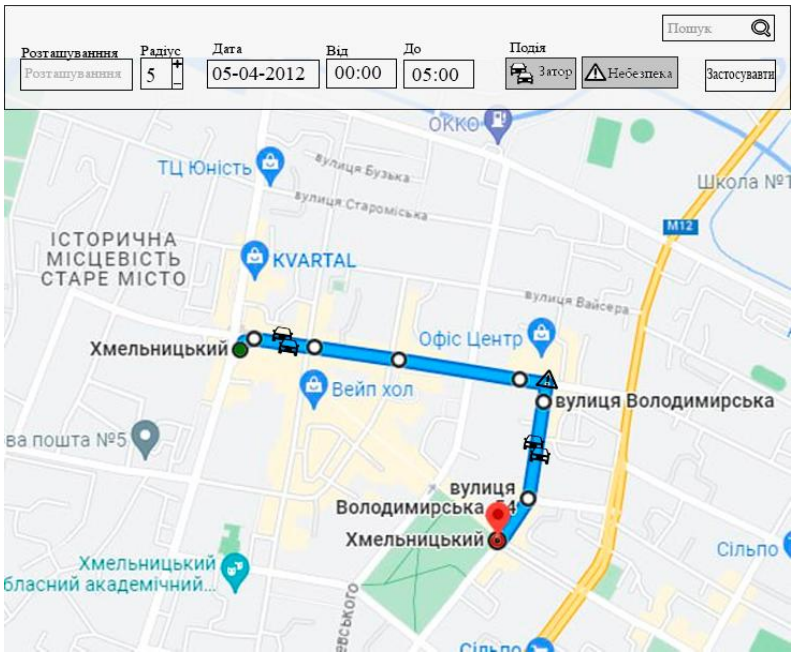



Рисунок 3.8 – Застосування двох фільтрів одночасно

Інформація про різноманітні деталі, пов'язані з подією, зберігається за допомогою системи. Наприклад, час, привязаний до місцезнаходження транспортного засобу, ім'я водія, номер машини, широта та довгота тощо(рисунок 3.9)

За потреби ,окрім вище названих даних ,система може фіксувати відео (з реєстратора наприклад) та виводити його в відповідне вікно(рисунок 3.10).

Також в данному вікні відображається інформація про події ,тобто затримки поломки тощо.

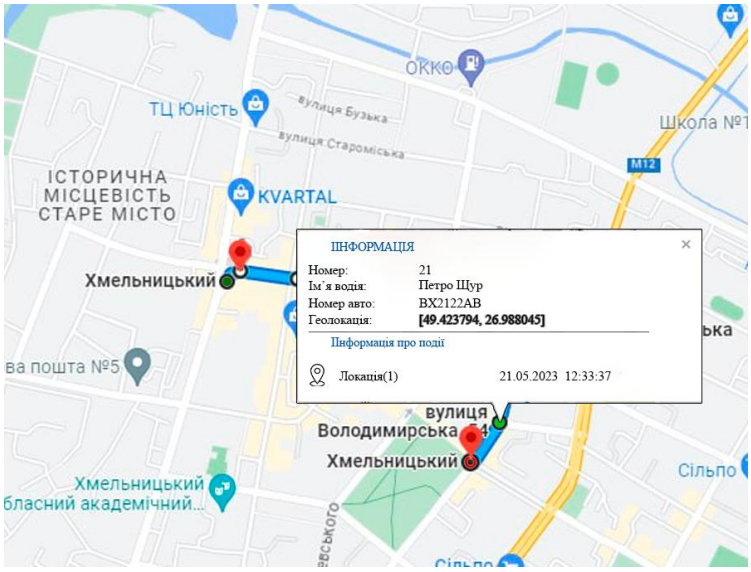



Рисунок 3.9 – Деталі події

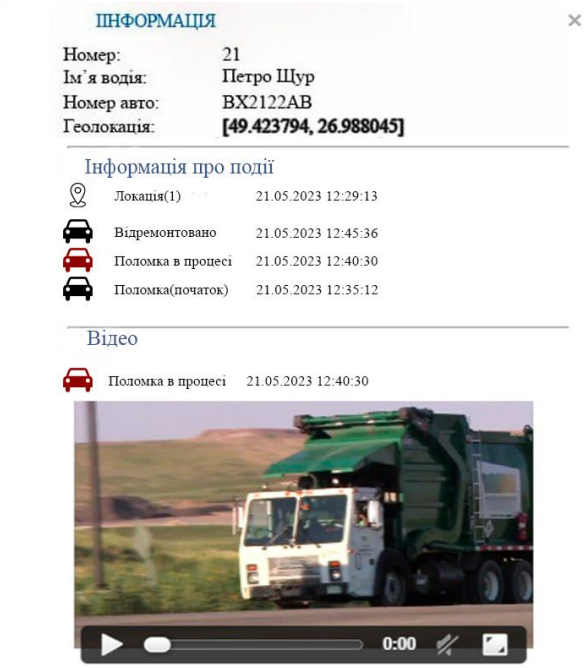



Рисунок 3.10 – Вікно з детальною інформацією та відео

Також в системі реалізована система пошуку (за допомогою elastic search) . Пошук орієнтується на такі параметри як порядковий номер авто в системі (fleetID) , дата, місце на мапі та радіус пошуку.

На наведеному нижче знімку екрану показано результат пошуку транспорту за допомогою порядкового номеру в системі(ID) автомобіля(рисунок 3.11).

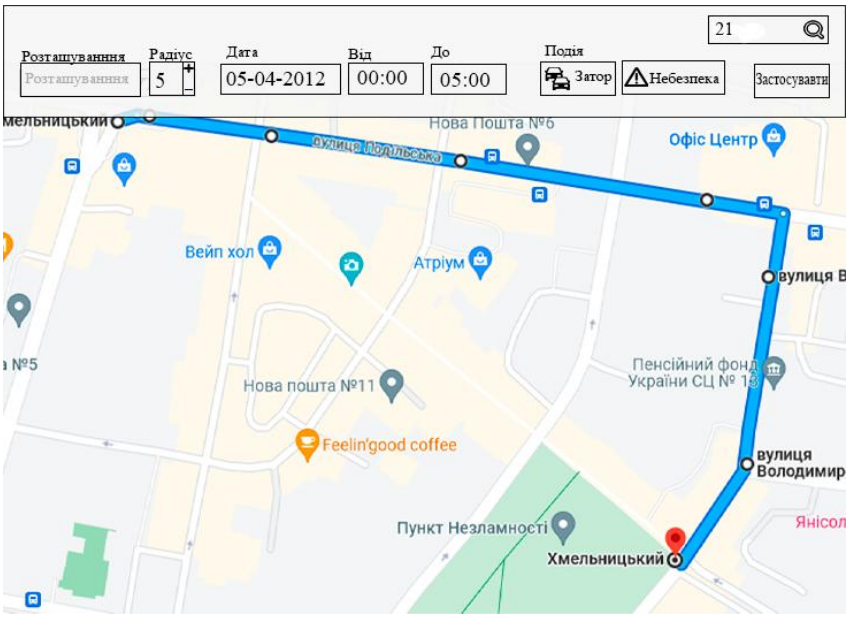



Рисунок 3.11 – Пошук по ID.

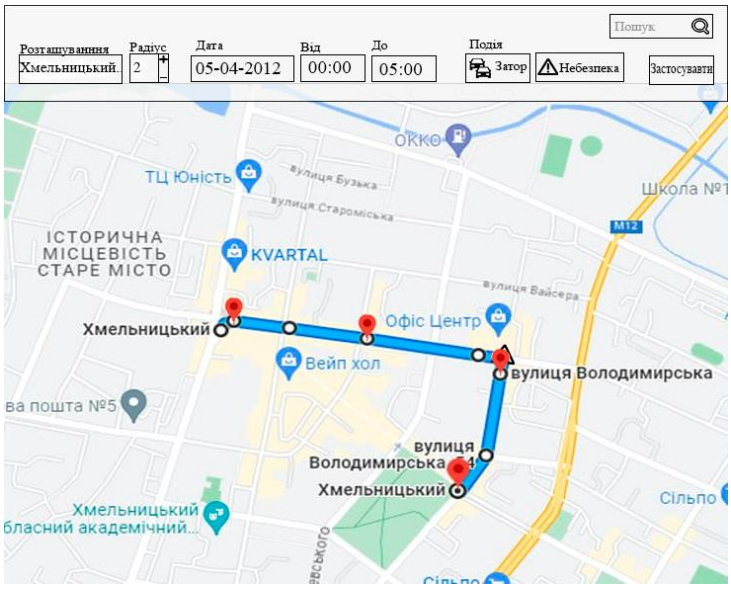



Рисунок 3.12 – пошук за параметром “розташування” “радіус”та “дата”

Знімок екрану вище(рисунок 3.12) показує пошук транспорту за місцем розташування та радіусом у заданій даті , а також із застосуванням інших фільтрів

В цілому, система .є інтеграцією декількох основних компонентів, починаючи від кластерного механізму розподіленої черги, механізму обробки в реальному часі, механізму розподіленого зберігання даних і закінчуючи простою

інтерактивною консоллю моніторингу транспорту в реальному часі на основі Google Maps. Реалізовані інтерфейси API на основі REST є основою динамічної взаємодії в реальному часі між кінцевим користувачем та системою.

### 3.6 Висновок

В цілому,проект .є інтеграцією декількох основних компонентів, починаючи від кластерного механізму розподіленої черги, механізму обробки в реальному часі, механізму розподіленого зберігання даних і закінчуючи простою інтерактивною консоллю моніторингу транспорту в реальному часі на основі Google Maps. API на основі REST, що використовуються в реалізації, є основою активної взаємодії в реальному часі між кінцевим користувачем та ядром системи . Для цього було реалізовано простий інтерфейс задля спрощення взаємодії.

# ВИСНОВКИ

Система використовує зразок набору геолокаційних даних для оцінки продуктивності обробки геолокаційної інформації в реальному часі. До розподіленого механізму черг подається велика кількість геолокаційних данних, які ставлять у чергу інформацію в межах певної геолокаційної області.

Ця велика кількість інформації в черзі подається паралельно до механізму обробки в реальному часі storm. Вузол Storm отримує ці повідомлення і надсилає їх на обробку до системи обробки даних . Система витягує дані, оптимізує їх і виконує різні обчислювальні завдання, такі як розрахунок відстані за широтою-довготою, аналіз того, чи не перевищує транспортний засіб швидкість, генерує сигнал небезпеки і т.д..

Нарешті, оброблена інформація зберігається в elastic search - розподіленому сховищі даних.

Кіберфізична система моніторингу транспорту в режимі реального часу стала б передовим технологічним переходом від традиційної системи моніторингу транспорту до високомасштабованої, ефективної та надійної системи моніторингу транспорту, яка б задовольняла попит на дуже великі обсяги даних про геолокацію транспорту. Це забезпечить значну користь великим підприємствам, допомагаючи їм знизити витрати та максимізувати продуктивність.

Хоча система спочатку потребуватиме значних витрат на встановлення апаратних ресурсів та залучення експертів з технології паралельних обчислень, однак подальші процеси підтримки та нарощування додаткових пристроїв і т.д. будуть досить простими та необтяжливими. Це значно зменшить майбутні витрати для підприємців, а також надасть економічно ефективне рішення для моніторингу кінцевому користувачеві. Система підійде для всіх видів бізнесу, оскільки її можна буде адаптувати до індивідуальних потреб різних організацій, починаючи від бізнесу та науки і закінчуючи сферою туризму та подорожей.

Існує величезний простір для досліджень та інновацій у сфері моніторингу, відстеження та управління ідентифікацією, а також безпеки людей та майна. Система може бути доповнена до наступного рівня розумної інформаційної системи з різними типами вбудованого штучного інтелекту.

Сфера машинного навчання та штучного інтелекту вже не є чимось далеким та неосяжним. Таким чином, систему можна розглядати як приклад використання в контексті дослідження різних видів штучного інтелекту, які можна застосувати до системи. Наприклад, враховуючи безпеку водія, систему можна зробити достатньо розумною, щоб вона помічала, коли водій перевищує швидкість тощо. Аналогічно, система могла б автоматично піднімати тривогу в громадському транспорті до найближчої служби безпеки, коли вона виявляє надзвичайні події в транспортному засобі, такі як пограбування, на основі певних ознак, таких як надмірний шум або виявлення зброї на відео, тощо. Щодо подорожей і туризму - систему можна зробити настільки розумною, щоб вона надавала історичну інформацію про місця, які відвідує турист, про найближчі ресторани, лікарні та інші служби екстреної допомоги. Це створить відчуття безпеки, що в кінцевому підсумку сприятиме розвитку цієї галузі.

## ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАНЬ


1. Todd Palino. Kafka: The Definitive Guide: Real-Time Data and Stream Processing at Scale. 2021. 485 с.
2. Bill Bejeck. Kafka Streams in Action: Real-time apps and microservices with the Kafka Streams API. 2018. 280 с.
3. Neha Narkhede. APACHE KAFKA: INVENT THE FUTURE. 2021. 297 с.
4. Emil Koutanov Effective Kafka: A Hands-On Guide to Building Robust and Scalable Event-Driven Applications with Code Examples in Java Kindle.. 2020. 468 с.
5. Dylan Scott Kafka in Action. 2022. 272 с.
6. Raoul-Gabriel Urma. Modern Java in Action: Lambdas, streams, functional and reactive programming. 2018 592 с.
7. Mitch Seymour Mastering Kafka Streams and ksqlDB: Building Real-Time Data Systems by Example). 2021. 432 с.
8. Alex S The Ultimate Guide to Event-Driven Programming with Kafka 2023. 78 с.
9. Brindha Priyadarshini Jeyaraman Real-Time Streaming with Apache Kafka, Spark, and Storm: Create Platforms That Can Quickly Crunch Data and Deliver Real-Time Analytics to Users 2021 182 с.
10. Benjamin Schmeling Kubernetes Native Development: Develop, Build, Deploy, and Run Applications on Kubernetes 2022 416 с.
11. Nishant Garg Learning Apache Kafka 2018 112 с.
12. Raúl Estrada Apache Kafka Quick Start Guide: Leverage Apache Kafka 2.0 to simplify real-time data processing for distributed applications 2018 188 с.
13. Raul Estrada. Mastering Apache Storm: Real-time big data streaming using Kafka, Hbase and Redis. 2019. 582 с.
14. Matthew Jankowski. Storm Applied: Strategies for real-time event processing. 2019. 280 с.
15. Antti Laaksonen. Guide to Competitive Programming: Learning and Improving Algorithms Through Contests 2020. 328 с.

16. Saurav Haloi. Apache ZooKeeper Essentials 2021. 253 с.

17. Flavio Junqueira ZooKeeper: Distributed Process Coordination. 2019. 246 с.

18. Nathan Marz Big Data: Principles and best practices of scalable realtime data systems 2018 328 с.

19. Tom White. Hadoop: The Definitive Guide . 2018. 688 с.

20. Asjad Athick Getting Started with Elastic Stack 8.0: Run powerful and scalable data platforms to search, observe, and secure your organization 2022 474 с.

21. Alberto Paro Elasticsearch 8.x Cookbook: Over 180 recipes to perform fast, scalable, and reliable searches for your enterprise, 5th Edition 2022 750 с.

22. Pranav Shukla. Learning Elastic Stack 6.0: A beginner's guide to distributed search, analytics, and visualization using Elasticsearch, Logstash and Kibana. 2018. 434 с.

23. Dan Noble Elastic Search Monitoring 2018. 326 с.

24. Doug Turnbull. Relevant Search: With applications for Solr and Elasticsearch. 2018. 360 с.

25. Huseyin Akdogan. Elasticsearch Indexing. 2019. 274 с.

26. Tommaso Teofili Deep Learning for Search 2019 328 с.

27. Pranav Shukla Learning Elastic Stack 7.0: Distributed search, analytics, and visualization using Elasticsearch, Logstash, Beats, and Kibana 2019 476 с.

28. Wai Tak Wong Advanced Elasticsearch 7.0: A practical guide to designing, indexing, and querying advanced distributed search engines 2019 560 с.

29. Alberto Paro Elasticsearch 7.0 Cookbook: Over 100 recipes for fast, scalable, and reliable search for your enterprise 2019 726 с.

30. Louis Rosenfeld Search Analytics for Your Site: Conversations with Your Customers 2021 224 с.

31. Thomas Erl Cloud Computing: Concepts, Technology & Architecture (The Pearson Service Technology Series from Thomas Erl) 2018 528 с.

32. Stephen J. Phillips Elastic Architecture: Frederick Kiesler and Design Research in the First Age of Robotic Culture (The MIT Press) 2018 384 с.

33. Anurag Srivastava Elasticsearch 7 Quick Start Guide: Get up and running with the distributed search and analytics capabilities of Elasticsearch 2019 188 c.

34. Brent Chaters Mastering Search Analytics: Measuring SEO, SEM and Site Search 2021 400 c.

35. Raoul-Gabriel Urma Modern Java in Action: Lambdas, streams, functional and reactive programming 2018 592 c.

36. Luciano Manelli Beginning Jakarta EE Web Development: Using JSP, JSF, MySQL, and Apache Tomcat for Building Java Web Applications 2020 424 c.

37. Apache Software Foundation Apache Tomcat 7 User Guide 2018 254 c,.

38. Aleksa Vukotic Apache Tomcat 7 2021 509 c.

39. Tanuj Khare Apache Tomcat 7 Essentials 2022 435 c.

40. Christian Wenz Apache Tomcat Bible 2019 840 c.

41. Vivek Chopra Professional Apache Tomcat 6 2018 672 c.

42. Etienne Langlet Apache Tomcat 5 - Java application server - Windows or Linux administration 2017 573 c.

43. SciMat Notes Developer Tears: Hardcover Journal Notebook for Quality Assurance Engineers and Programmers 2021 200 c.

44. Jon Duckett JavaScript and jQuery: Interactive Front-End Web Development 2018 640 c.

45. Jon Duckett Web Design with HTML, CSS, JavaScript and jQuery Set 2020 1152 c.

46. David Sawyer McFarland JavaScript & jQuery: The Missing Manual 2019 686 c.

47. Mary Delamater Murach's JavaScript and jQuery 2020 752 c.

48. Jon Duckett Front-End Back-End Development with HTML, CSS, JavaScript, jQuery, PHP, and MySQL 2022 938 c.

49. Robin Nixon Learning PHP, MySQL & JavaScript: With jQuery, CSS & HTML5 (Learning Php, Mysql, Javascript, Css & Html5) 2018 812 c.

50. Jon Raaschc JavaScript and jQuery for Data Analysis and Visualization 2019 480 с.

51. Dane Cameron A Software Engineer Learns HTML5, JavaScript and jQuery: A guide to standards-based web applications 2020 257 с.

52. Dayley Brad AngularJS, JavaScript, and jQuery All in One, Sams Teach Yourself 2019 2440 с.

53. Brad Dayley jQuery and JavaScript in 24 Hours, Sams Teach Yourself (Sams Teach Yourself in 24 Hours) 2019 656 с.

54. Jay Panseriya JavaScript and JQuery: Interactive Front–End Web Development (Best Coding Tutorials) 2020 431 с.

55. Larry Sanchez Web Programming with HTML, CSS, Bootstrap, JavaScript, jQuery, PHP, and MySQL Second Edition 2019 318 с.

56. David Flanagan jQuery Pocket Reference: Read Less, Learn More 2021 160 с.

57. Brad Dayley jQuery and JavaScript Phrasebook (Developer's Library 2019 873 с.

58. Addy Osmani Learning JavaScript Design Patterns: A JavaScript and jQuery Developer's Guide 2019 254 с.

59. Earle Castledine jQuery: Novice to Ninja: Novice to Ninja 2018 480 с.

60. Bear Bibeault jQuery in Action 2020 504 с.

**Додаток А**

(обов’язковий)

Лістинг коду для Elasting Search з боку сервера

```
      {
{
    "projectsevent": {
  "mappings": { "event":
              {
        "properties": {
          "eventType": {
            "properties": {
              "data": {
                "type": "object" },
              "event": {
                "type": "integer" }
            } },
          "fleetNumber": {
            "type": "string",
            "index": "not_analyzed" },
          "geoTag": {
            "type": "geo_point" },
          "timestamp": {
            "type": "long"
          } }
      } }
 },  "rtdfms": {
    "mappings": {
      "video": {
        "_all": { "enabled":
          false
        },  "properties": {
          "content": { "type":
            "string", "index": "no"
          },  "eventType": {
            "type": "nested",
            "properties": {
              "data": {
                "properties": {
                  "customerId": {
                    "type": "string" }
                } },
              "event": {
                "type": "integer" }
            } },
          "fleetNumber": {
            "type": "string",
            "index": "not_analyzed" },
          "geoTag": {
            "type": "geo_point" },
          "timestamp": {  "type":
            "long", "index":
```

```
            "analyzed"
          },  "videoName": {
            "type": "string",  "index":
            "not_analyzed"
          } }
      } }
},  "driver": {
  "mappings": {
    "route": {
      "_all": { "enabled":
        false
      },  "properties": {
        "address": { "type":
          "string",
          "index": "not_analyzed",
          "store": true
        },  "city": {
          "type": "string",   "index":
          "not_analyzed", "store": true
        },  "country": {
          "type": "string",   "index":
          "not_analyzed", "store": true
        },  "customerId": {
          "type": "string",   "index":
          "not_analyzed", "store": true
        },  "customerName": {
          "type": "string",   "index":
          "not_analyzed", "store": true
        },  "fleetNumber": {
          "type": "string",
          "store": true
        },  "geoTag": {
          "type": "geo_point",
          "store": true
        },  "pickupTime": {
          "type": "long", "index":
          "analyzed", "store": true
        },  "pin": {
          "type": "string",   "index":
          "not_analyzed", "store": true
        } }
    } }
},  "fleet": {
  "mappings": {
    "info": {
      "properties": {
        "driverName": {
          "type": "string", "analyzer":
          "standard", "path":
          "just_name", "fields": {
            "fullDriverName": {
```

```
                "type": "string",
                "analyzer": "lowercase_analyzer" }
            }
          },  "fleetNumber": {
            "type": "string",   "index":
            "not_analyzed", "store": true
          },  "vehicleNumber": {
            "type": "string",   "index":
            "not_analyzed", "store": true
          } }
      } }
  } }
```

Завантаження відео на стороні клієнта

```
URL: http://localhost:8080/project/uploadvideo
Content-Type: Application/JSON
Request Method: POST {
   "fleetNumber": "2",  "timestamp":
   "1436500631000", "latitude": 28.5700,
   "longitude": 77.3200, "videoName":
   "blackberry1.mp4", "content": "",
   "eventType": [ {
          "event": "1",
          "data": {
             "customerId": "1" }
       }, {
          "event": "2",
          "data": null
       } ]
}
```

Запит події на стороні клієнта

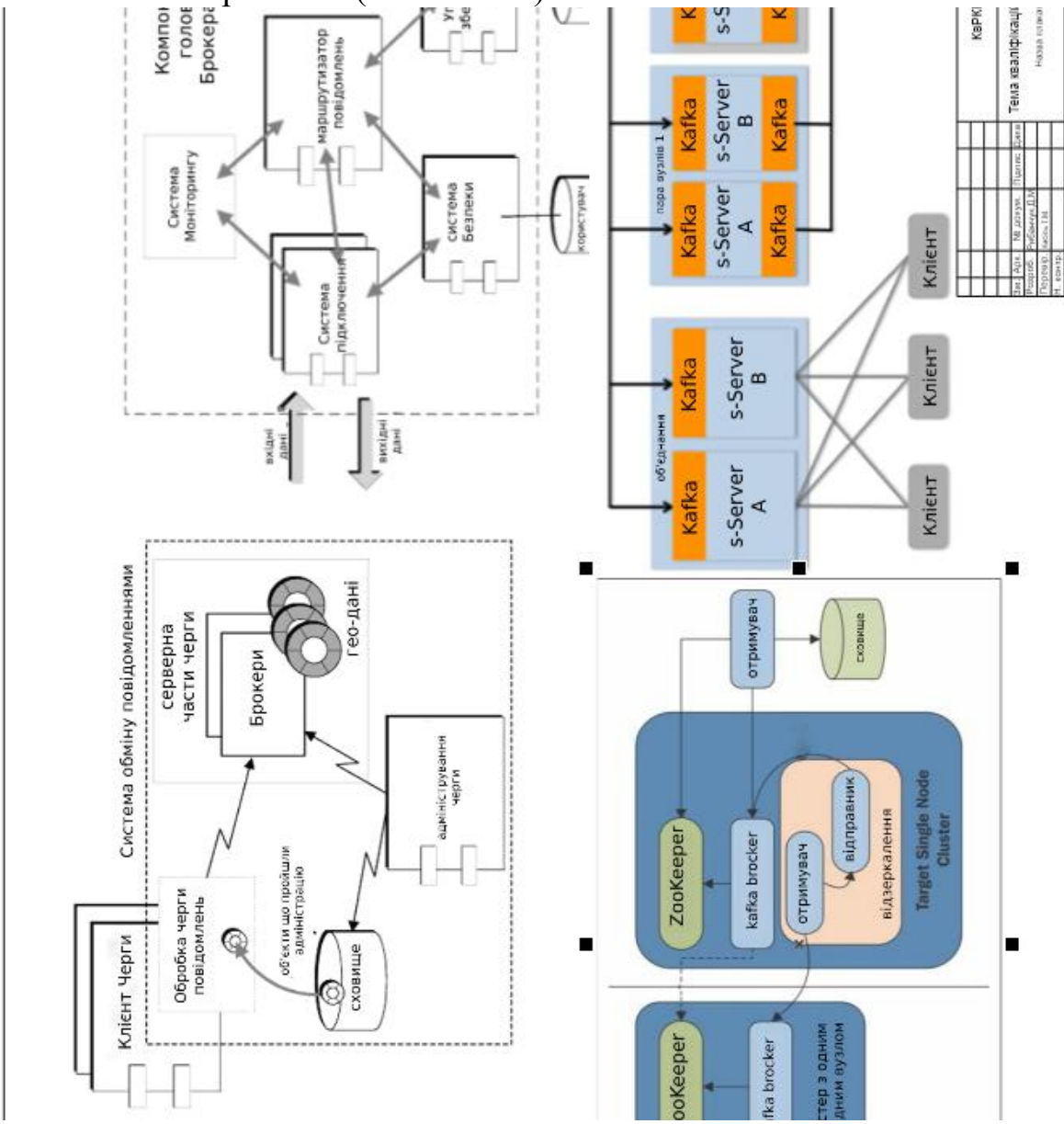
```
URL: http://localhost:8080/project/uploadevent
Content-Type: Application/JSON
Request Method: POST {
  "timestamp": "1424686295011",
  "fleetNumber": "22",
  "latitude": "40.728998",
  "longitude": "-74.081687",
  "eventType" : {
               "event": "3",
              "data": {
                     "customerId": "1" }

          }   }
```

**Додаток Б**

(обов’язковий)

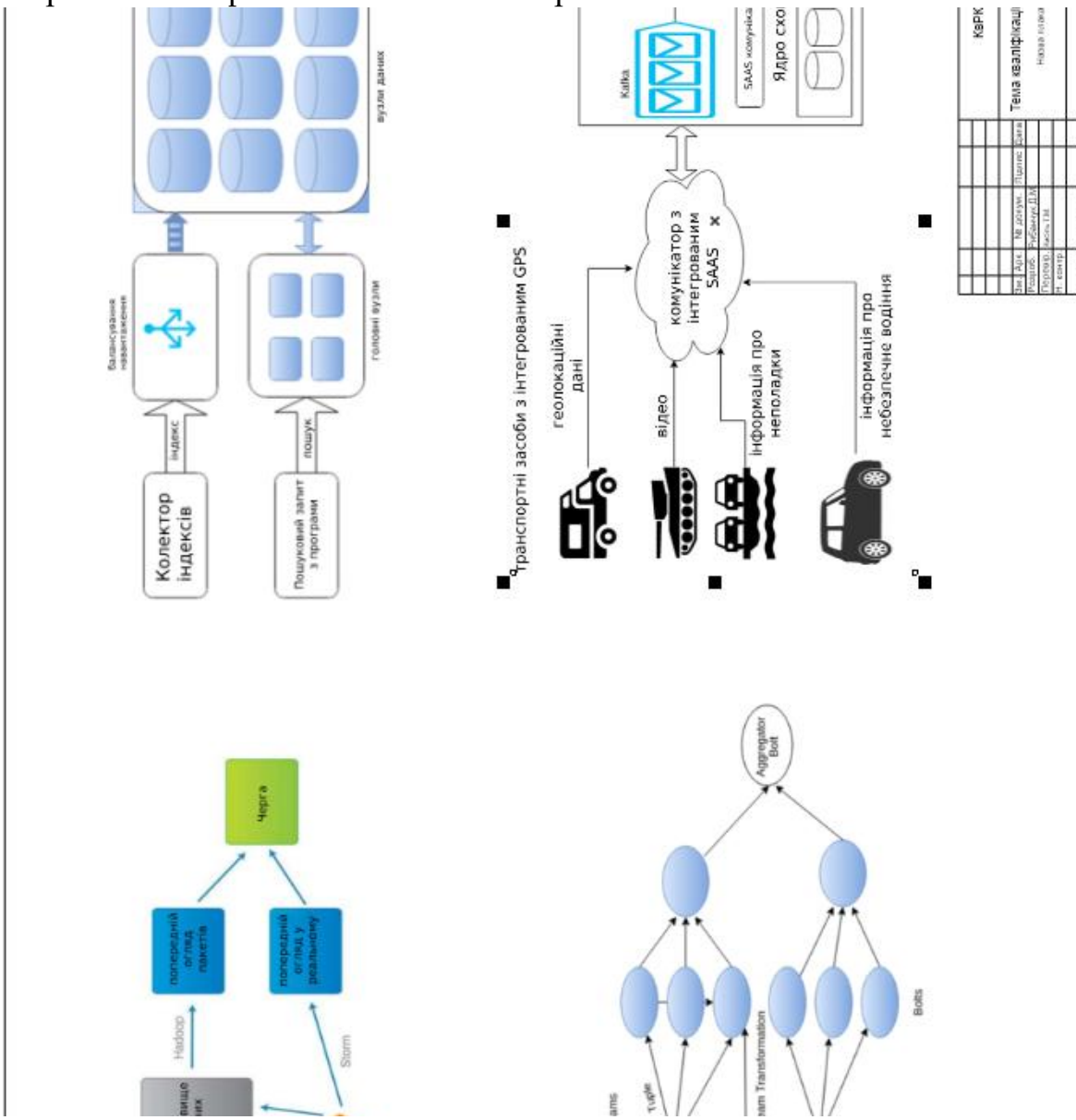
Копія креслення(блок-схеми)“Kafka”



**Додаток В**

(обов’язковий)

Копія креслення “Apache Storm+загальна робота системи”

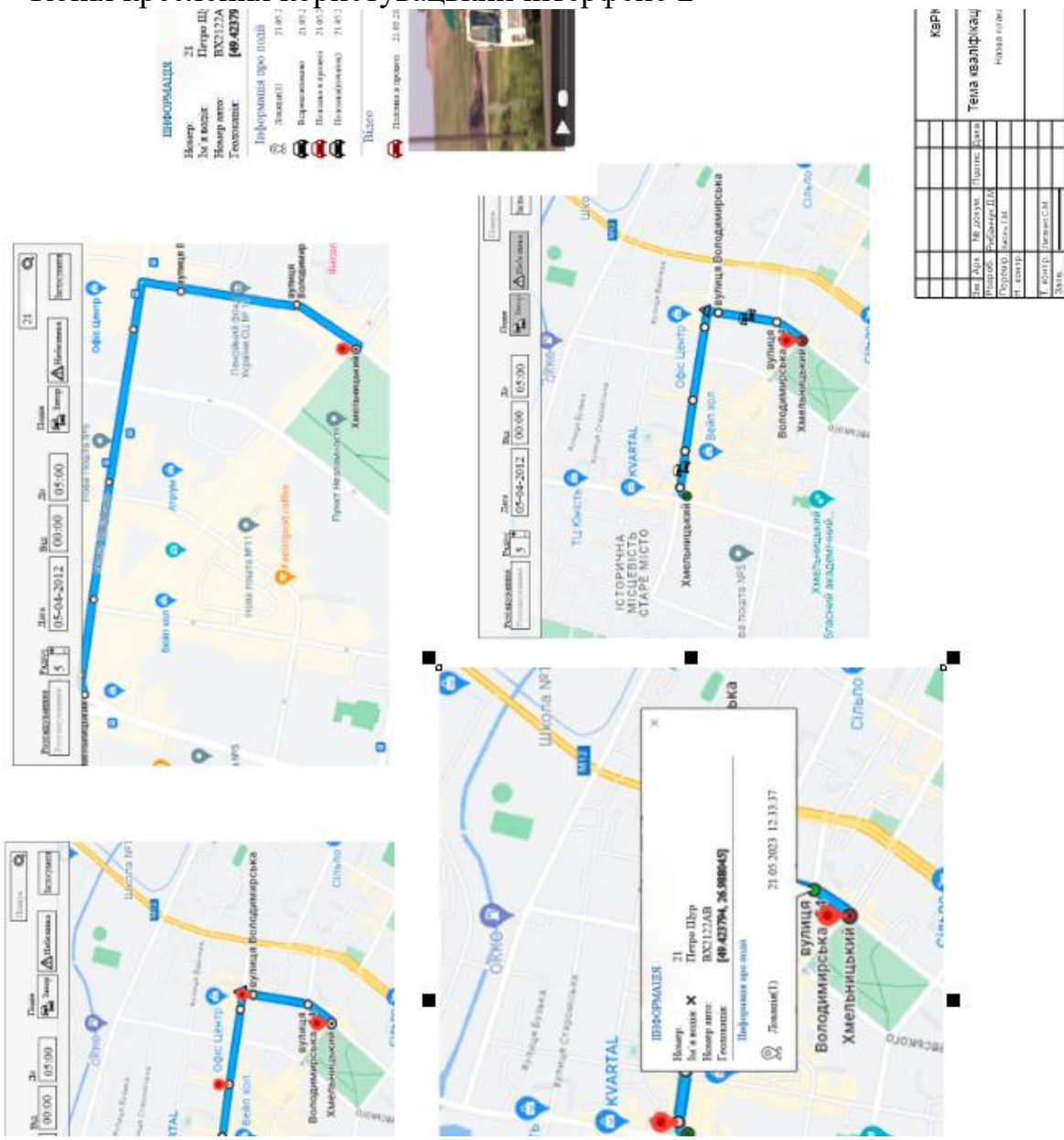


**Додаток Г**

(обов'язковий)

Копія креслення користувацький інтерфейс 1

**Додаток Д**
(обов’язковий)
Копія креслення користувацький інтерфейс 2



UNICHECK

Ім'я користувача:
**Кафедра КІ**

ID перевірки:
**1015469232**

Дата перевірки:
**06.06.2023 20:27:15 EEST**

Тип перевірки:
**Doc vs Internet + Library**

Дата звіту:
**06.06.2023 20:27:39 EEST**

ID користувача:
**100005591**

Назва документа: **Рибанчук_Кіберфізична система відстеження транспортних засобів в реальному часі**

Кількість сторінок: **64** Кількість слів: **11120** Кількість символів: **85000** Розмір файлу: **6.44 MB** ID файлу: **1015127760**

## 4.22%
## Схожість

Найбільша схожість: **1.39%** з Інтернет-джерелом (http://www.itrack.com.ua/ua/support/docs/historyofgps)

| | | |
|---|---|---|
| 4.03% Джерела з Інтернету | 73 | Сторінка 66 |
| 1.02% Джерела з Бібліотеки | 75 | Сторінка 67 |

## 0.26% Цитат

| | | |
|---|---|---|
| Цитати | 5 | Сторінка 68 |
| Посилання | 1 | Сторінка 68 |

## 0%
## Вилучень

Немає вилучених джерел

## Модифікації

Виявлено модифікації тексту. Детальна інформація доступна в онлайн-звіті.

| | |
|---|---|
| Замінені символи | 1 |

Tue Jun 06 19:32:29 EEST 2023, Медзатий Дмитро Миколайович, Хмельницький національний університет, ХНУ

# Anti-Plagiarism v-15.257

**Максимальне співпадіння з одним документом 0.0%**

Словники перевірки: en_US, ru_RU, ua_UA. **Помилок в документах: 8%**

| ID: 114993<br>Назва: БКР Кіберфізична система відстеження транспортних засобів в реальному часі<br>Додано в БД: 2023-06-06<br>Автора: Д. М. Рибанчук<br>Керівники: Т.М. Кисіль<br>Консультанти:<br>Опоненти: | Документ | | Сумарний збіг по Базі Даних | |
|---|---|---|---|---|
| | Символи | Лексеми | Символи | Лексеми |
| | 75834 | 598 | 255 (0%) | 4 (1%) |

Джерело плагіату

| ID | Опис | Наявність плагіату в документі | |
|---|---|---|---|
| | | Символи | Лексеми |

Завідувачу кафедри КІІС
д-р.техн.наук, проф. Говорущенко Т. О.

Рибанчука Дмитра Миколайовича

ПІБ здобувача вищої освіти

ФІТ, 4 курсу, групи КІ2-19-2

ЗАЯВА

З правилами чинного Положення «Про систему забезпечення академічної доброчесності у Хмельницькому національному університеті» від 01.07.2022, згідно з яким виявлення плагіату є підставою для відмови в допуску кваліфікаційної роботи до захисту та застосування заходів дисциплінарної та академічної відповідальності, ознайомлений (а). Про використання програмно-технічних засобів для перевірки кваліфікаційних робіт здобувачів вищої освіти на плагіат оповіщений(а) та надаю свою згоду на обробку та збереження університетом моєї роботи в інституційному репозитарії університету.

Також надаю університету право на передачу моєї роботи для обробки та збереження в базах даних програмно-технічних засобів (Unicheck та Anti-Plagiarism) та використання роботи для виявлення плагіату в інших роботах, які перевіряються програмно-технічними засобами та користувачами, що мають доступ до цих програмно-технічних засобів, виключно в обмежених цілях для виявлення плагіату в текстах робіт.

Робота для перевірки університетом надається в друкованому та електронному варіанті. Електронна версія моєї роботи збігається (ідентична) з друкованою.

22 травня 2023 року

РІШЕННЯ ЕКСПЕРТНОЇ КОМІСІЇ

КАФЕДРИ КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ ТА ІНФОРМАЦІЙНИХ СИСТЕМ

ПРО ДОПУСК КВАЛІФІКАЦІЙНОЇ РОБОТИ ДО ЗАХИСТУ

Підтверджуємо ознайомлення з результатом звіту подібності щодо роботи, генерованого системою виявлення текстових збігів/ідентичності/схожості:

Назва: Кіберфізична система відстеження транспортних засобів в реальному часі

Автор: Рибанчук Дмиро Миколайович

Спеціальність: 123 – Комп'ютерна інженерія

Освітня програма: освітньо-професійна

Науковий керівник: Кисіль Тетяна Миколаївна, к.ф.-м.н., доц.

Після аналізу звіту подібності зроблено такий висновок:

| № | Висновок | Позначка про відповідність |
|---|---|---|
| 1 | Запозичення, виявлені в роботі, є законними і не є плагіатом. Робота приймається до захисту. | відповідає |
| 2 | Виявлені запозичення не є плагіатом, розміщені в розділах, які не описують безпосередньо авторське дослідження, але кількість цитат перевищує обсяг, виправданий поставленою метою роботи. Робота приймається до захисту, але має бути відкоригована. Відкоригований варіант має бути поданий на кафедру за 2 дні до захисту, разом із заявою щодо самостійності виконання письмової роботи та ідентичності друкованої та електронної версії роботи | |
| 3 | Виявлені запозичення не є плагіатом, але частково розміщені в розділах, які описують безпосередньо авторське дослідження, а кількість цитат перевищує обсяг, виправданий поставленою метою роботи. В зв'язку з цим мета роботи та поставлені завдання не були досягнені. Робота може бути допущена до захисту (наступного року) після того як буде відкоригована та допрацьована і успішно пройде повторну перевірку на академічний плагіат. | |
| 4 | Робота містить навмисні текстові спотворення, передбачувані спроби укриття запозичень або інші прояви академічного плагіату. Робота містить фабрикацію або фальсифікацію даних. Робота не допускається до захисту. | |

Підтвердження:

Запозичення, виявлені в роботі, є законними і не є плагіатом, оскільки:

1) запозичення розміщені в розділах аналізу існуючих аналогів та прототипів, які не описують безпосередньо авторське дослідження і не стосуються результатів роботи;
2) усі запозичення фрагментарні, або мають належним чином оформленні посилання;
3) окремі виявлені збіги є загальновживаними фразами або виразами, про що свідчить посилання системи на збіг з багатьма джерелами на один фрагмент речення;
4) всі зафіксовані системою ознаки модифікації тексту відносяться до комбінування латинських символів зі україномовними скороченнями індексів в формулах, що не є модифікацією тексту

Сумарний обсяг всіх запозичень, визначений системою виявлення збігів/ ідентичності/схожості, складає 4.22% і адресується до 73 першоджерела, що, з урахуванням наведених обґрунтувань, відповідає характеру наукового дослідження і свідчить на користь кваліфікаційної роботи.

Керівник роботи _____ Т. М. Кисіль

Гарант ОП _____ С. М. Лисенко

Завідувач кафедри КІІС _____ Т. О. Говорущенко

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ХМЕЛЬНИЦЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

# РЕЦЕНЗІЯ НА КВАЛІФІКАЦІЙНУ РОБОТУ

Дипломник: Рибанчук Дмитро Миколайович

Тема: Кіберфізична система відстеження транспортних засобів в реальному часі

Спеціальність: 123 «Комп'ютерна інженерія»

Обсяг кваліфікаційної роботи:

Кількість листів креслень 3 Кількість сторінок записки 68

1. Короткий зміст роботи та прийнятих рішень: Метою кваліфікаційної роботи є розробка кіберфізична система відстеження транспортних засобів в реальному часі.

2. Висновок про відповідність роботи дипломному завданню: Робота повністю відповідає поставленому завданню.

3. Характеристика виконання кожного розділу, ступінь використання останніх досягнень науки і техніки і передових методів роботи: В першому розділі кваліфікаційної роботи проведено дослідження предметної області. Описано сучасний стан проблеми. Розглянуто різні методи та підходи, що використовуються для відстеження транспортних засобів в реальному часі. Обґрунтовано найважливіші аспекти відстеження: прорахування найефективніших маршрутів, зменшення кількості простоїв на дорогах та планування подорожі більш ефективно, відстеження споживання палива/заряду, що дозволяє оптимізувати витрати та продовжити термін експлуатації транспортного засобу. В другому розділі кваліфікаційної роботи проаналізовані існуючі системи для вирішення поставленої задачі. В третьому розділі кваліфікаційної роботи реалізовано простий інтерфейс задля спрощення взаємодії користувача і система.

4. Позитивні сторони роботи: висока практична цінність роботи.

5. Негативні сторони роботи: слід було більше приділити увагу архітектурі системи .

6. Оцінка графічного оформлення та пояснювальної записки роботи: Пояснювальна записка оформлена коректно, згідно діючих стандартів оформлення документації.

7. Відгук про роботу в цілому: Робота виконана на достатньому технічному рівні.

8. Інші зауваження: _____

9. Оцінка дипломної роботи: добре/С/4.00

Рецензент (прізвище, ім'я, по батькові, посада, місце роботи) _____

Рафальчук Галина Іванівна, доцент кафедри ТПЗ ХНУ

"07" 06 2023 р. _____(підпис)