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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

«До захисту допущено»

Завідувач кафедри

_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)

“___” червня 2019 р.

# Дипломний проект

**на здобуття ступеня бакалавра**

з напряму підготовки **6.050102 «Комп'ютерна інженерія»**

на тему: Масштабовані програмні засоби. Засоби підвищення продуктивності ідентифікації обличчя людини

Виконала: студент IV курсу, групи КВ-51

Симотюк Микола Вікторович

(підпис)

Керівник доц.каф.СПСКС, к.т.н. Петрашенко А.В.

(підпис)

Консультант з нормоконтролю, доц.каф.СПСКС, к.т.н. Клятченко Я.М.

(підпис)

Рецензент_____
(посада, науковий ступінь, вчене звання, науковий ступінь, прізвище та ініціали) (підпис)

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

Київ – 2019 року

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

ФАКУЛЬТЕТ ПРИКЛАДНОЇ МАТЕМАТИКИ

Кафедра системного програмування і спеціалізованих комп‘ютерних систем

Рівень вищої освіти – перший (бакалаврський)

Напрям підготовки 6.050102 «Комп’ютерна інженерія»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Тарасенко В.П.
(підпис) (ініціали, прізвище)
«___» червня 2019 р.

**ЗАВДАННЯ**
**на дипломний проект студента**
Симотюка Миколи Вікторовича

1. Тема проекту «Масштаюовані програмні засоби моніторингу об’єктів охорони. Засоби підвищення продуктивності ідентифікації обличчя людини», керівник проекту доц.каф.СПСКС, к.т.н. Петрашенко А.В.,
затверджені наказом по університету від «22» травня 2019 р. №1330-С

2. Термін подання студентом проекту: дивись технічне завдання.

3. Вихідні дані до проекту: титульна сторінка, програма, технічне завдання (4 сторінки), пояснювальна записка (50 сторінок), п’ять додатків.

4. Зміст пояснювальної записки: перелік скорочень, умовних позначень та термінів; вступ; аналіз існуючих масштабованих програмних засобів моніторингу та об**ґрунтування теми дипломного проекту**; аналіз технологій для розробки масштабованих систем моніторингу; структура системи та опис роботи модулів; тестування системи та аналіз результатів; висновок; список використаної літератури.

5. Перелік графічного матеріалу (із зазначенням обов'язкових креслеників, плакатів, презентацій тощо): презентація; структурні схеми: розташування модулів програми, загальна схема проекту; схеми алгоритмів: основний маршрут користувача, система моніторингу.

6. Консультанти розділів проекту*

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

7. Дата видачі завдання

Календарний план

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів проекту | Примітка |
|---|---|---|---|
| 1. | Вивчення літератури за тематикою проекту | 15.04.2019 | |
| 2. | Розроблення та узгодження технічного завдання | 30.04.2019 | |
| 3. | Аналіз існуючих рішень | 05.05.2019 | |
| 4. | Підготовка матеріалів першого розділу дипломного проекту | 10.05.2019 | |
| 5. | Підготовка матеріалів другого розділу дипломного проекту | 18.05.2019 | |
| 6. | Підготовка графічної частини дипломного проекту | 20.05.2019 | |
| 7. | Оформлення документації дипломного проекту | 25.05.2019 | |
| 8. | Попередній огляд матеріалів диплому на кафедрі | 30.05.2019 | |

Студент _____ _____
(підпис) (ініціали, прізвище)

Керівник проекту _____ _____
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено керівника дипломного проекту.

# АНОТАЦІЯ

Об’єкт розробки – створення масштабованої комп’ютерної системи моніторингу об’єктів охорони з ідентифікацією обличчя людини.

Комп’ютерна система дозволяє: створювати модель обличчя користувача для нейромережі; оброблювати автоматично надіслані фотознімки та виявляти на них користувачів; у разі виявлення надсилати в реальному часі сповіщення з координатами знаходження. В процесі розробки були використані протокол WebSocket, бібліотека OpenCV, технологія Node.js та NoSQL база даних MongoDB.

В ході розробки:

- проведено аналіз різних методів виявлення обличчя на фотознімках;
- сформульовані вимоги масштабованої комп’ютерної системи;
- розроблена система моніторингу об’єктів охорони на основі мікросервісної архітектури;
- залучені методи розподіленої обробки запитів;
- проведено навантажувальне тестування;

Система надає єдиний програмний інтерфейс, котрий дозволяє незалежно розробляти та впроваджувати різні продукти для спостереження об’єктів.

Ключові слова:

МАСШТАБОВАНА КОМП’ЮТЕРНА СИСТЕМА МОНІТОРИНГУ ОБ’ЄКТІВ ОХОРОНИ З ІДЕНТИФІКАЦІЄЮ ОБЛИЧЧЯ ЛЮДИНИ, WebSocket, MongoDB, Node.js, Microservices, Face recognition.

# ABSTRACT

The object of development - the creation of a scalable computer monitoring system of objects of protection with identification of the person.

The computer system allows: to create a model of the person's face for the neural network; handle and automatically detect photos sent to users; in case of detection, to send in real time a notification with the location of the location. In the process of development, the WebSocket protocol, the OpenCV library, the Node.js technology and the NoSQL MongoDB database were used.

During the development:

• analyzed various methods of face detection in photos;

• the requirements of a scalable computer system are formulated;

• a system of monitoring of objects of protection on the basis of micro-service architecture was developed;

• Involved methods of distributed query processing;

• Load testing is carried out;

The system provides a single software interface that allows you to independently design and implement various products for monitoring objects.


Keywords:

SCALABLE COMPUTER SYSTEM OF MONITORING OF OBJECTS OF PROTECTION WITH IDENTIFICATION OF THE PERSON, WebSocket, MongoDB, Node.js, Microservices, Face recognition.

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | | | Документація загальна | | | |
| | | | Новорозроблена | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.002 ТЗ | Масштабовані програмні | 4 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Технічне завдання | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.003 ТП | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Відомість технічного | | | |
| | | | проекту | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.004 ПЗ | Масштабовані програмні | 50 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |

| Зм | Лист | № докум. | Підп | Дата | ІАЛЦ.467200.001 ОА | | |
|---|---|---|---|---|---|---|---|
| Розроб. | | Симотюк М.В. | | | Масштабовані програмні засоби. Засоби підвищення продуктивності ідентифікації обличчя людини. *Опис альбому* | Літ. | Лист 1 / Листів 3 |
| Перев. | | Петрашенко А.В. | | | | | |
| | | | | | | КПІ ім. Ігоря Сікорського, ФПМ, КВ-51 | |
| Н. контр. | | Клятченко Я.М. | | | | | |
| Затв. | | Тарасенко В.П. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | А4 | ІАЛЦ.467200.005 Д1 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Розташування модулів | | | |
| | | | програми | | | |
| | | | Схема структурна | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.006 Д2 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Загальна схема проекту | | | |
| | | | Схема структурна | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.007 Д3 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Основний маршрут | | | |
| | | | користувача | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.008 Д4 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Система моніторингу | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | | Диск CD-ROM | Текст ПЗ. Тексти програм. | 1 | | |
| | | | Графічний матеріал. | | | |

**ЗМІСТ**

## 1. НАЙМЕНУВАННЯ ТА ГАЛУЗЬ РОЗРОБКИ

Назва розробки: «Масштабовані програмні засоби моніторингу об’єктів охорони. Засоби підвищення продуктивності ідентифікації обличчя людини».

Галузь застосування: моніторинг об’єктів охорони.

## 2. ПІДСТАВА ДЛЯ РОЗРОБКИ

Підставою для розробки є завдання на виконання роботи першого (бакалаврського) рівня вищої освіти, затверджене кафедрою системного програмування і спеціалізованих комп’ютерних систем Національного технічного університету України «Київський Політехнічний Інститут імені Ігоря Сікорського».

## 3. МЕТА І ПРИЗНАЧЕННЯ РОБОТИ

Метою даного проекту є створення масштабованої комп’ютерної системи на основі якої створюються різні підсистеми для моніторингу, обілку об’єктів охорони та їх керування.

## 4. ДЖЕРЕЛА РОЗРОБКИ

Джерелом інформації є технічна та науково-технічна література, технічна документація, публікації у періодичних виданнях та електронні статті у мережі Інтернет.

## 5. ТЕХНІЧНІ ВИМОГИ

### 5.1. Вимоги до програмного продукту, що розробляється

- Обробка фотознімків користувача для подальшого створення його моделі як сутності в базі даних;
- Обробка фотографій місцевості для виявлення користувачів або неопізнаних осіб;
- Сповіщення про виняткові ситуації;
- Масштабована архітектура;
- Відмовостійкість та здатність витримувати високі навантаження;

### 5.2. Вимоги до апаратного забезпечення

- Процесор: 4-ядерний процесор;
- Оперативна пам’ять: 32 Гб;
- Наявність доступу до мережі Internet (Ethernet);

### 5.3. Вимоги до програмного та апаратного забезпечення користувача

- Будь-який пристрій зі здатністю відправляти http-запити зі знімками та технологією WebSocket;

## 6. ЕТАПИ РОЗРОБКИ

| № з/п | Назва етапів виконання дипломного проекту | Термін виконання етапів |
|---|---|---|
| 1. | Вивчення літератури за тематикою проекту | 15.04.2019 |
| 2. | Розроблення та узгодження технічного завдання | 30.04.2019 |
| 3. | Аналіз існуючих рішень | 05.05.2019 |
| 4. | Підготовка матеріалів першого розділу дипломного проекту | 10.05.2019 |
| 5. | Підготовка матеріалів другого розділу дипломного проекту | 18.05.2019 |
| 6. | Підготовка графічної частини дипломного проекту | 20.05.2019 |
| 7. | Оформлення документації дипломного проекту | 25.05.2019 |
| 8. | Попередній огляд матеріалів диплому на кафедрі | 30.05.2019 |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | | | Документація загальна | | | |
| | | | Новорозроблена | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.004 ПЗ | Масштабовані програмні | 50 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Пояснювальна записка | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.005 Д1 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Розташування модулів | | | |
| | | | програми | | | |
| | | | Схема структурна | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.006 Д2 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Загальна схема проекту | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467200.003 ТП | | |
|---|---|---|---|---|---|---|---|
| Розробив | | Симотюк М.В. | | | Масштабовані програмні засоби. Засоби підвищення продуктивності ідентифікації обличчя людини<br>Відомість технічного проекту | Літ. | Аркуш 1 / Аркушів 2 |
| Перевірив | | Петрашенко | | | | | |
| Консульт. | | | | | | **КПІ ім. Ігоря Сікорського, ФПМ, КВ-51** | |
| Н. контроль | | Клятченко Я.М. | | | | | |
| Зав. каф. | | Тарасенко В.П. | | | | | |

| Поз. | Формат | ПОЗНАЧЕННЯ | НАЙМЕНУВАННЯ | Кількість аркушів | № прим. | Примітки |
|---|---|---|---|---|---|---|
| | | | Схема структурна | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.007 Д3 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Основний маршрут | | | |
| | | | користувача | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | А4 | ІАЛЦ.467200.008 Д4 | Масштабовані програмні | 1 | | |
| | | | засоби. Засоби підвищення | | | |
| | | | продуктивності | | | |
| | | | ідентифікації обличчя | | | |
| | | | людини | | | |
| | | | Система моніторингу | | | |
| | | | Схема алгоритму | | | |
| | | | | | | |
| | | Диск CD-ROM | Текст ПЗ. Тексти програм. | 1 | | |
| | | | Графічний матеріал. | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |
| | | | | | | |

| Змін. | Арк. | № докум. | Підпис | Дата | ІАЛЦ.467200.003 ТП | Арк. 2 |
|---|---|---|---|---|---|---|

**ЗМІСТ**

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

JSON (JavaScript Object Notation) – текстовий формат обміну даних, оснований на JavaScript.

BSON (Binary JSON) – комп'ютерний формат обміну даними.

ПЗ – програмне забезпечення.

TCP (Transport Control Protocol) – протокол транспортного рівня передачі даних в моделі OSI.

HTTP (Hyper Text Transfer Protocol) – протокол прикладного рівня передачі даних в моделі OSI.

TLS (Transport Layer Security) – криптографічний інтернет-протокол.

RFC (Request for Comments) – документ із серії пронумерованих інформаційних документів Інтернету, що містить технічні специфікації та стандарти.

# ВСТУП

Проблеми злагодженої роботи розподіленої системи існують на всіх рівнях програмного забезпечення (ПЗ) та обладнання. Окремо, веб-системи додають труднощі з мережами. Тому існує безліч підходів для подолання цих проблем і немає жодного універсального.

Для такого направлення як “охорона” також критичним стає питання миттєвості, але існуюча класична клієнт-серверна архітектура не вирішує його. Взагалі початково Всесвітня мережа була призначена тільки для обміну документами, тому наразі для збільшення інтерактивності та функціональності застосунків складність зростає рік за роком, а кількість нових підходів для розробки кожен місяць.

Також при довгоочікуваних операціях (як-от аналіз зображень) найросповсюдженіші технології реалізації серверів працюють неефективно, даремно витрачаючи ресурси процесора.

Найбільші інтернет-корпорації світу вже давно зіткнулись з цими та багатьма іншими проблеми і вирішували їх протягом багатьох років. Це була тільки їх проблема, але з часом з’явилися нові учасники, тож зараз вже існує безліч підходів, технологій, стандартів та навіть мов програмування для вирішення цих задач. Важливо тільки правильно їх використовувати.

# 1. АНАЛІЗ ІСНУЮЧИХ МАСШТАБОВАНИХ ПРОГРАМНИХ ЗАСОБІВ МОНІТОРИНГУ ТА ОБҐРУНТУВАННЯ ТЕМИ ДИПЛОМНОГО ПРОЕКТУ

## 1.1. Причини використання охоронних систем

Охоронні системи призначені для виявлення несанкціонованого доступу на об’єкт охорони та застосовуються в цілях захисту від злому, псування майна та крадіжок [1]. Також, встановлення таких систем впливає на зниження кількості крадіжок у цілому [2]. Ці причини є основними, але не єдиними. Адже, наприклад, в’язниці також використовують системи безпеки для контролю над ув’язненими.

До того ж, існують комбіновані системи, які додатково забезпечують захист від пожеж та залиття водою. Комбіновані системи можуть поєднуватись з багатьма підсистемами, тому є дуже гнучкими та дозволяють використовувати одночасно різні типи сенсорів, зв’язку, сповіщення та інше. Тому загалом охоронна система є дуже потужним інструментом, котрий можна модифікувати, удосконалювати та розвивати у різних напрямках та різними способами, в залежності від вимог та складності самого об’єкту охорони.

Спосіб, розроблений у ході роботи над даним дипломним проектом, використовує зображення відеокамер для ідентифікації обличчя людини. Це дозволяє, наприклад, впроваджувати рівні доступу, базуючись виключно на попередньому аналізі фотознімків співробітників (або інших учасників об’єкту).

## 1.2. Огляд принципів роботи охоронних систем

З розвитком можливостей бездротових систем, а саме їх мініфікація, збільшення можливостей, пропускної здатності та надійності, ринок

охоронних систем стрімко почав збільшуватися. Тому зараз вони активно вбудовуються на складах, промислових, офісних будівлях, квартирах та ін. Відповідно до цього способи та підходи до їх впровадження різняться між собою.

Система охоронної сигналізації має три загальні компоненти: пристрої виявлення, блок керування та пристрій звітності. Пристрої виявлення встановлюються на захищеному приміщенні для виявлення охоронної системи. Вони зазвичай з'єднуються електричним контуром з блоком управління. Блок управління забезпечує живлення датчиків і приймає і оцінює сигнали від них. Після надходження сигналу тривоги блок управління повідомляє за допомогою пристрою подання сигналу, який також називається пристроєм індикації тривоги, про сигнал тривоги. Цей сигнал тривоги може бути або звуковим/візуальним пристроєм, встановленим у захищеному приміщенні, та/або сигналом для блоку прийому сигналізації у віддаленому місці. Кожен з цих пристроїв інтегрований і розроблений у різноманітні прикладні пристрої та системи, залежно від потреб безпеки та міркувань витрат.

Розглянемо класифікацію охоронних систем [3]. Вони діляться:

- За взаємодією з небезпекою**:**
  - Пасивні - направлені на швидке сповіщення власника або охоронних служб. Саме цей тип використовується в цьому дипломному проекті;
  - Активні - направлені на запобігання проникненню. При цьому може наноситись серйозна шкода здоров’ю зловмиснику, а в залежності від країни це може зумовити кримінальну відповідальність;
- За способом передачі інформації:
  - Провідні
    1) Аналогові - при такому способі передачі ППК (пристрій прийомо-контрольний) слідкує за станом шлейфу

сигналізації, який підключений безпосередньо до датчиків. При спрацьовуванні одного з датчиків, струм у шлейфі змінюється та ППК фіксує це. Недоліком є те, що неможливо визначити, котрий саме датчик спрацював, адже вони всі однаково впливають на струм;

2) Адресні - за допомогою адресів вирішується проблема, описана у попередньому способі;

- Бездротові - При такому способі охоронні датчики передають інформацію на приймаючу станцію за допомогою радіосигналів.
    1) Без зворотного зв'язку - недоліком є велика кількість способів перешкодити передачі інформації;
    2) Зі зворотнім зв'язком - дозволяє підтримувати неперервний моніторинг;
- Через GSM-мережі - використовується як для інформування власника шляхом SMS-повідомлення, так і передачею сигналу на пульт охоронної служби;
- Через протоколи Wi-Fi - подібно до попереднього пункту, але інформування кінцевого користувача відбувається за допомогою спеціального ресурсу в мережі Internet або мобільного додатка. В даному дипломному проекті використовується цей варіант в комбіновану вигляді з використанням як ресурсу в Internet, так і додатка, а реалізовано за допомогою технології PWA (Progressive Web Application);

● За способом сповіщення про тривогу:

- Центральна станція. В системі охоронної сигналізації центральної станції вторгнення автоматично передається до комерційного агентства, яке називається центральною станцією. Там підготовлені оператори та слідчі тривоги завжди присутні для нагляду, запису та

реагування на сигнал. Оператор може негайно відправити слідчих тривог і телефонує в поліцію;

- Охоронна фірма. В даному випадку пристрої виявлення та схеми підключені до постійно контрольованого приймального обладнання на центральній контрольній станції, яка знаходиться на захищеній власності і управляється персоналом, відповідальним перед власником нерухомості;
- Поліцейська станція. Дана станція підключається до охоронної сигналізації та складається з захисних пристроїв і ланцюгів, підключених через блок управління до постійно обладнаного поліцейського відділу. Вторгнення викликає звуковий/візуальний пристрій, що приводиться в дію, і сигналізує поліцейський відділ. Але її експлуатація здійснюється виключно під контролем власника нерухомості. Немає зовнішнього агентства, яке б гарантувало, що сигналізація була включена. Але ця система зазвичай недоступна
- Локальна сигналізація. Локальна охоронна сигналізація керує звуковим/візуальним пристроєм на захищеному об'єкті і у випадку несанкціонованого вторгнення або навіть спроби, пристрій відлякує зловмисників до привернення уваги сусіда або перехожого. Але функціонування цієї системи знаходиться під контролем власника майна. Як наслідок, немає гарантії, що система може бути включена, коли необхідний захист;
- Житлова охоронна сигналізація. Складається з пристроїв і схем, підключених через блок управління до зондового/візуального пристрою, який може мати віддалене підключення до центральної станції, поліцейської станції, власної станції або станції моніторингу. Станція моніторингу житлових приміщень - це установа, яка має чергового персоналу для підготовки сигналів, отриманих від систем охоронної сигналізації;

## 1.3. Аналіз існуючих охоронних систем

З розвитком технологій та суттєвого вдосконалення алгоритмів машинного навчання, у сфері охоронних систем виникають все більше нові технологічні компанії. Більшість з них використовують комбіновані системи, адже вони є найбільш ефективними та функціональними.

Так, українська компанія AJAX Systems виробляє системи, що дозволяють під’єднати одночасно до 150-ти пристроїв [4]. Щоб отримати такий ефект, їм знадобилось розробляти власну операційну систему реального часу для хабів зі здатністю надсилати смс та робити телефонні виклики. Також знадобилось створити власний радіопротокол передачі даних, котрий би міг підтримувати одразу 150 з’єзднань із пристроями (рисунок 1.1), не втрачаючи при цьому у швидкодії та затримці обробки. Окремої уваги заслуговують алгоритми обробки сигналів від сенсорів, адже саме за допомогою них вирішуються багато проблем правильної ідентифікації. Наприклад, як відрізнити людину від птаха або будь-кого іншого? Аби ще зменшити кількість хибних тривог, алгоритми аналізують інформацію відразу з кількох сенсорів, включаючи спектральний аналіз. Загалом, кількість вхідних параметрів надзвичайно велика, тому їх аналіз не є тривіальною задачею.


Рисунок 1.1

Але також існує учасник ринку з іншим підходом - Ring [5]. Ця компанія спеціалізується головним чином на дверних дзвінках (рисунок 1.2), всередину яких вбудована камера з великим кутом огляду та високою якістю запису. Також користувачу надається доступ до керування вхідними дверима, і все це у вигляді додатка на телефон із обміном даних через всесвітню мережу. Таким чином, можливо слідкувати за дверима та відчиняти їх, знаходячись на іншому материку.


Рисунок 1.2

Також у цієї компанії є власні розробки у сфері машинного навчання, розпізнавання обличчя людини. І саме на їх прикладі можна переконатись, що створити повністю робочий алгоритм, який би працював у всіх випадках, - надзвичайно складна задача. І навіть лідери ринку її не вирішили повністю. Так, якщо алгоритм не зміг обробити інформацію та коректно розпізнати обличчя, уривок відео аналізується вручну, тобто співробітниками компанії. А це додає ще окремого питання щодо конфіденційності даних і так далі. Це не є якимось окремим випадком, адже порушення конфіденційності інформації трапляється постійно навіть у компаніях не те що лідерах, а творцях ринку (Google, Facebook, Apple). І ця проблема є однією з найактуальніших - безпека. Частково причиною усіх неполадок та складностей роботи систем є основа взагалі усієї галузі - протоколи передачі даних та стандарти, котрі були спроектовані та впровадженні майже півстоліття тому. В той час було просто неможливо передбачити весь спектр послуг, який почали надавати у Всесвітній мережі - від

онлайн-кінотеатрів до банків. Тому зараз всі ці системи побудовані поверх звичайної передачі текстових документів та можливості посилатись один на одного. Тоді не було сенсу підміняти пакети у мережі, відслідковувати трафік та красти доступ до систем. Але зараз ситуація вкрай інакша, адже кожну хвилину у мережі здійснюються покупки та транзакції на мільйони доларів США. Тоді виходить, що інструменти застосовують для речей, для яких вони не були створені. Тому й вразливість цих систем є дуже високою. У будь-якому випадку ця тема є дуже обширною і у цьому дипломному проекті буде розглядатись лиш дотично.

Останнім часом провідні охоронні комплекси також працюють із залученням хмарний технологій (рисунок 1.3). Це дозволяє будувати більш гнучкі системи із високою швидкістю розгортання інфраструктури, застосовуючи готові рішення для моніторингу і оминаючи провайдерів та інших посередників. Також це суттєво зменшує витрати, адже покупець оплачує рівно стільки, скільки використав. Тож вже немає сенсу утримувати свої власні сервери (центри обробки даних), оновлювати та підтримувати їх. І якщо раніше відповідальність за відмовостійкість лежала на самій команді розробників, зараз відповідальними є провайдери. Так, хмарні технології дають можливість повністю абстрагуватись від інфраструктурного шару. І якщо немає ніяких обмежень з боку законодавства (наприклад, вимога до зберігання та обробки даних тільки в межах країни), цей спосіб набагато спростив усю розробку систем.

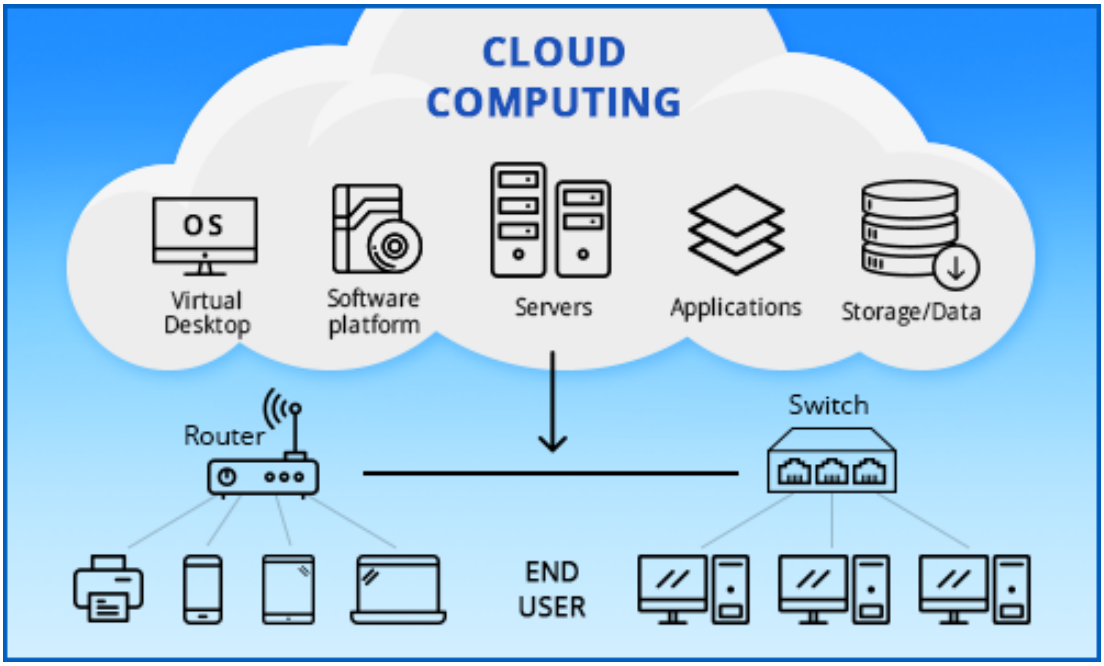



Рисунок 1.3

В даному дипломному проекті для цих цілей використовуються віртуальний виділений сервер (ВВС) від Digital Ocean. Взагалі, ВВС запускає власну копію операційної системи (ОС), тож клієнти можуть мати доступ на рівні суперкористувача до даного екземпляра операційної системи та мають можливість встановити практично будь-яке програмне забезпечення, яке працює на цій ОС. Для багатьох цілей вони функціонально еквівалентні виділеному фізичному серверу і, будучи визначеними програмним забезпеченням, можуть набагато легше створюватися і конфігуруватися. Вони набагато дешевші, ніж еквівалентний фізичний сервер. Однак, оскільки вони розділяють основне фізичне обладнання з іншими VPS, продуктивність може бути нижчою, залежно від навантаження будь-яких інших виконуваних віртуальних машин.

Звичайно, також у AJAX Systems і в інших продуктів є свої власні мобільні додатки, адже це найкращий спосіб просувати, підтримувати продукт. І його наявність не є додатковим бонусом, а необхідністю, зумовленою ринком.

## 1.4. Обґрунтування теми дипломного проекту

Окремо розглянемо недоліки охоронних систем, описаних вище:

- Мінімальна можливість кастомізації**.**

  Зазвичай такі системи є монолітами з обмеженими можливостями змінити якийсь конкретний модуль. Рішення, застосоване при розробці системи для даного дипломного проекту, надає можливість використовувати будь-яку камеру від будь-якого виробника (на мобільному телефоні, планшеті, ноутбуці, веб-камеру) з встановленим додатком та доступом до Internet;

- Складність монтажу та налаштування.

  Зазвичай встановлення програмного забезпечення та правильні дрібні налаштування системи не обходяться без професійного спеціаліста, котрий знає всі нюанси системи та може правильно все зібрати. З додатком натомість потрібно виконати прості кроки. І при будь-якій зміні місцеположення або функціональності якогось модуля все можливо виконати самостійно, не затрачаючи великої кількості часу. Це надає більш високу гнучкість та зменшує витрати на підтримку;

- Зручність експлуатації.

  Досить часто виникають ситуації, в яких користувач забуває вимкнути сигналізацію, тож спрацьовує хибна тривога може приїхати охоронна служба. В нашому ж випадку, система має модель користувача, що надає їй можливість розпізнати, хто саме знаходиться на території. Тому такі ситуації уникаються;

Також в даному дипломному проекті для сповіщення використовується інтерактивна веб-мапа, котра надає наглядність та зручність для операторів з охоронних фірм.

# 2. АНАЛІЗ ТЕХНОЛОГІЙ ДЛЯ РОЗРОБКИ МАСШТАБОВАНИХ СИСТЕМ МОНІТОРИНГУ

## 2.1. Порівняльний аналіз програмних платформ для розробки серверних додатків.

Найбільш популярним підходом у розробці веб-орієнтованих сервісів на сьогоднішній день є підхід з використанням таких програмних засобів (LAMP): Linux, Apache, MySQL та PHP. Із цього списку ми розглянемо обмеження тільки веб-серверу - Apache HTTP Server.

Ядро Apache HTTP Server повністю розроблено з використанням мови програмування C та реалізує одну з підсистем по контролю ресурсів та пам’яті - так званий пул (pool). Ця підсистема контролює в тому числі й створення дочірніх процесів. При кожному новому http запиті, з пулу дістається окремий процес (в іншому випадку його довелось би створювати, а це додатковий час обробки запиту) для ізольованого контексту, це дозволяє виконувати усі запити паралельно та незалежно.

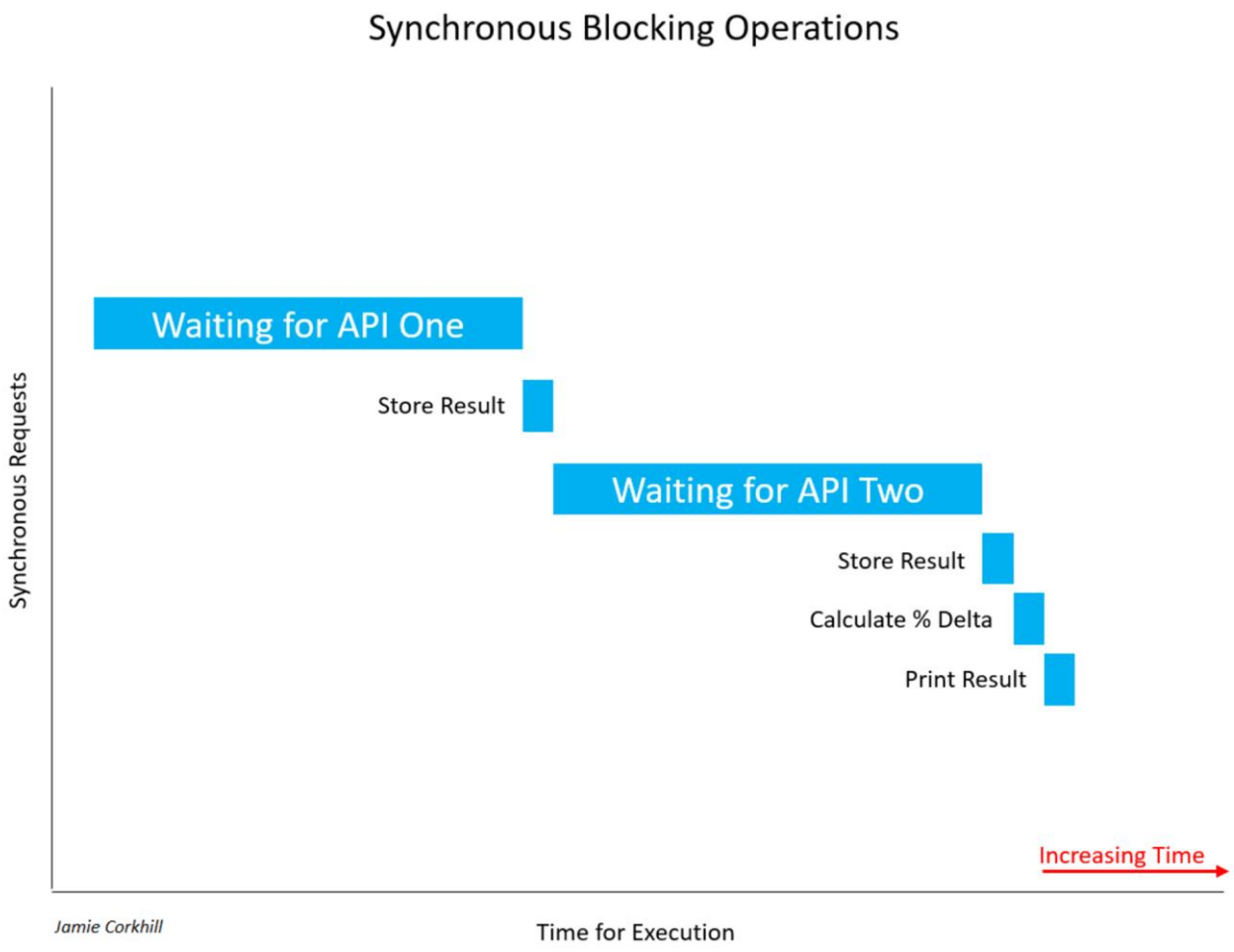



Рисунок 2.1

Проблема в тому, що кожен процес - це використані ресурси процесору та пам’яті, а більша частина клієнтського запиту - це очікування відповіді бази даних та інших зовнішніх програмних інтерфейсів. Виходить, кожен процес витрачає більшу частину ресурсів просто на очікування, що є надзвичайно неефективно. Цей підхід називається “синхронний ввід/вивід”, тобто система вводу/виводу просто блокується до моменту виконання операції (рисунок 2.1).

Насправді, такий підхід реалізований і в більшості інших програмних платформ. І в протилежність йому існує підхід “асинхронного вводу/виводу”, котрий очікувано дозволяє уникнути таких проблем.

Наприклад, найпопулярнішою на теперішній час платформою для веб-серверів з асинхронним вводом/виводом є NodeJS. Вона побудована навколо бібліотеки libuv, котра повністю фокусується на асинхронному вводі/виводі. Як

видно з назви, мовою програмування для NodeJS є JavaScript, що робить цю мову мовою загального значення, адже раніше вона використовувалась тільки для взаємодії з документами в браузерах. Для цього використовується рушій від браузера Google Chrome - V8, котрий також вбудований у цю платформу (рисунок 2.2). Усе це дало шалений розвиток цій мові програмування, вона швидко змінюється, і вже зараз з її допомогою розробляються десктопні та вбудовані додатки.

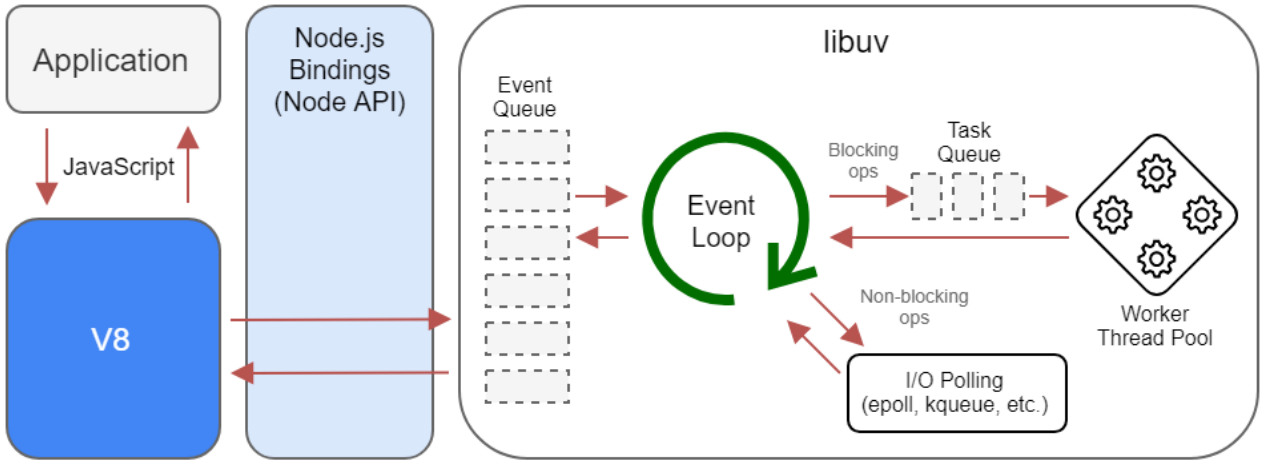



Рисунок 2.2

З асинхронним підходом клієнтський код JavaScript виконується в одному потоці. Здавалось би, як сервер, розрахований на тисячі запитів в секунду, може працювати в одному потоці. Але як видно з рисунку 2.2, уся робота з мережею, диском та іншими зовнішніми сервісами виконується саме за допомогою libuv, котра написана на мові програмування C та реалізує паралельність виконання на нижньому рівні.

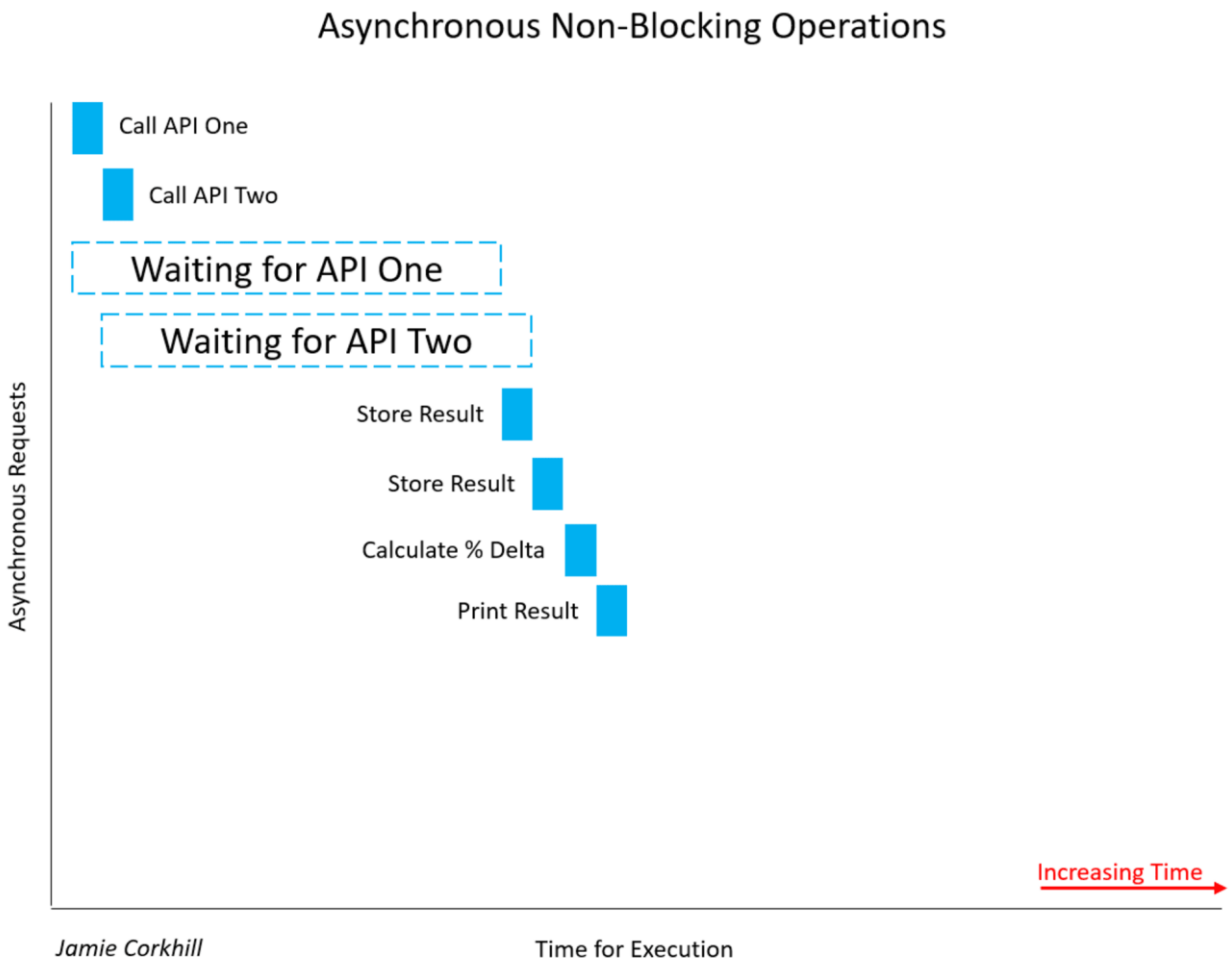



Рисунок 2.3

А вся комунікація між клієнтською програмою та системою вводу/виводу відбувається через так званий цикл подій (Event Loop), котрий й надає можливість писати клієнтський код в асинхронному стилі. Тож більше немає потребності у створенні тисяч дочірніх процесів, котрі ще й неефективно використовують ресурси процесора. Таким чином, усі запити обробляються в одному потоці, адже процесорний час, котрий раніше використовувався на очікування операції вводу/виводу (пунктирні блоки на рисунку 2.3), може використовуватись для обробки наступних запитів (рисунок 2.3)

Тепер створення дочірніх процесів можна використовувати для інших цілей, а саме для горизонтального масштабування. Взагалі, є два види масштабування - вертикальне та горизонтальне. При вертикальному масштабуванні здійснюється модернізація та покращення обладнання - більше

оперативної пам’яті та вдосконалення процесору (рисунок 2.4). Але таке масштабування є надзвичайно обмежене, адже вдосконалення обладнання має свою межу.

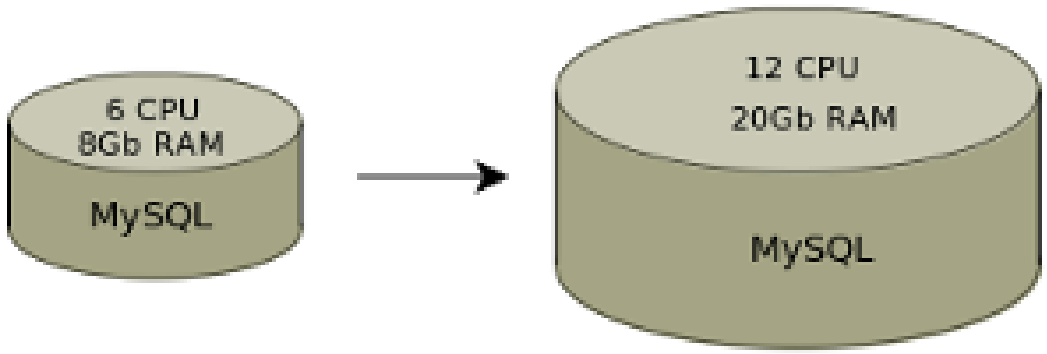



Рисунок 2.4

З іншою сторони, при горизонтальному масштабуванні у функціонуючу систему додаються нові обчислювальні потужності шляхом додавання нових серверів або інших окремих вузлів (рисунок 2.5). Так, система працює у виділеному кластері, у якому одночасно може знаходитись сотні обчислювальних вузлів. Важливо, аби це число було можливо змінювати в залежності від навантаження.

У даному дипломному проекті використовується саме горизонтальне масштабування, яке реалізується на кількох рівнях роботи системи. Найнижчий рівень - це саме створення дочірніх процесів. І ця можливість є другою причиною, чому в даному дипломному проекті для розробки використовується програмна платформа NodeJS, адже вона з самого початку проектувалась з вимогою до легкого горизонтального масштабування.

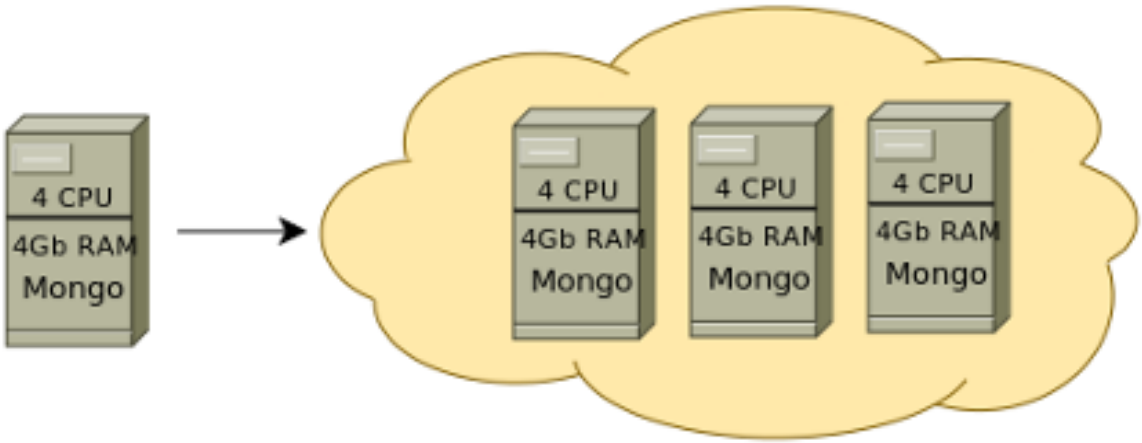



Рисунок 2.5

## 2.2. Порівняльний аналіз архітектурних стилів для проектування масштабованих систем.

Розглянемо три найбільш популярні способи комунікації вузлів розподіленої системи між собою:

- SOAP (Simple Object Access Protocol);
- REST (Representational State Transfer);
- Message Queue.

SOAP є протоколом, який зазвичай працює поверх HTTP(S), хоча підтримує й безліч інших протоколів. Його недоліками є неможливість роботи з іншими форматами даних, окрім XML, котрий сам по собі є надлишковим та має малу якість сприйняття. До цього, SOAP додає власні дані, від чого швидкість передачі повідомлень стрімко падає.

REST не є ні протоколом, ні стандартом, адже це просто архітектурний стиль. Але системи, реалізовані з залученням REST (котрі ще називаються RESTful), зазвичай використовують такі стандарти: HTTPS(S), JSON, URL. Комунікація RESTful систем виглядає просто та лінійно, але такі системи мають відповідати таким вимогам: модель клієнт-сервер, відсутність стану додатка,

кешування. Також, при використанні мікросервісної архітектури даний архітектурний стиль викликає додаткові труднощі при зростанні навантаження на всю систему в цілому. В даному дипломному проекті в цілях економії та швидкості розробки використовується даний варіант.

Message Queue - це загальний підхід для передачі даних, у якому за основу береться черга повідомлень. Існує безліч стандартів та реалізацій цього підходу. З найпопулярніших: RabbitMQ та Simple Queue Service (SQS) від Amazon Web Services (AWS). Даний варіант найкраще підходить для комунікації мікросервісів між собою та вирішує проблеми вищеописаних підходів, але його впровадження в систему є найбільш складним, адже він потребує існування в системі брокеру повідомлень. У випадку з RabbitMQ це додатковий сервіс, котрий потрібно підтримувати. Також кожна черга потребує тонкого налаштування в залежності від мікросервісів, які нею користуються, та вибраної стратегії обробки. Так, зазвичай до кожної черги додається ще одна “мертва” черга для можливості зберігання неопрацьованих по якихось причинах повідомлень та полегшення відлагоджування всієї системи.

Також проаналізуємо між собою монолітну та мікросервісну архітектуру (рисунок 2.6).

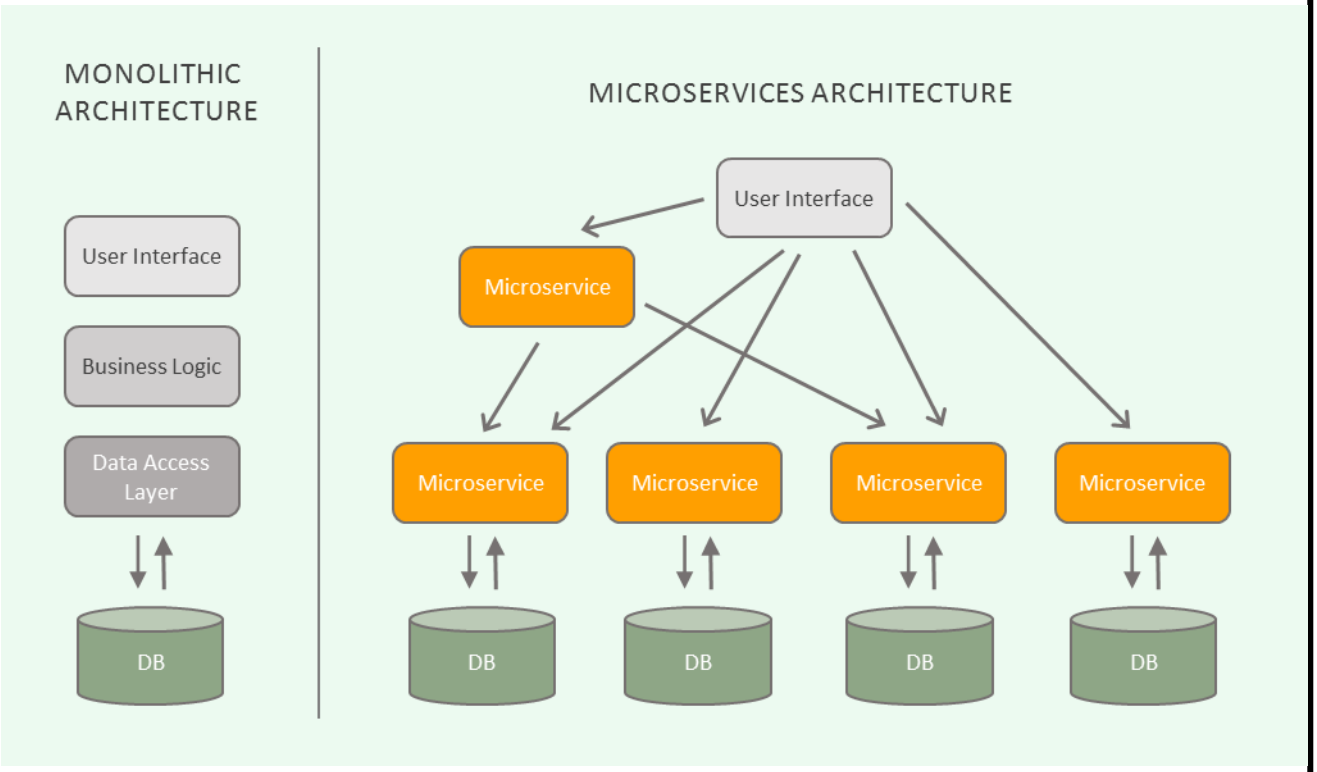



Рисунок 2.6

Монолітна архітектура передбачає реалізацію усієї функціональності продукту в рамках одного серверу. Це викликає кілька проблем.

Перша - це складність розробки при використанні концепції неперервної інтеграції та доставки продукту. Кожна нова функціональність потребує проходження багатьох етапів перед тим, як бути доставленою користувачу. Наприклад, для проходження автотестів потрібно розгорнути додаток у тестовому середовищі, а для цього його потрібно скомпілювати та зібрати, і у випадку моноліту, коли вся функціональність знаходиться в одній кодовій базі, це відбувається досить довго. І так на кожному етапі. Це надзвичайно уповільнює швидкість розробки. При мікросервісному підході, функціональність поділена між безліччю сервісів, котрі розгортаються, тестуються й доставляються незалежно один від одного, що є на порядок швидше.

Другою проблемою є проблема відповідальності команд за частини продукту. При монолітній архітектурі межі відповідальності розмиваються, адже весь проект знаходиться в єдиній кодовій базі. В мікросервісній

архітектурі такий поділ є легшим, адже кожна команда відповідає за конкретний мікросервіс, тому відповідальність стає чітко вираженою.

Третя проблема - проблема масштабування. Окремий мікросервіс є передбачуваним в плані споживання ресурсів, що дає можливість наперед визначити приблизні необхідні апаратні характеристики. Також такий мікросервіс легше масштабувати динамічно в залежності від навантаження.

## 2.3. Порівняльний аналіз систем керування базами даних (СКБД)

В даному дипломному проекті роль ефективності баз даних залежить від можливості шардування (рисунок 2.7).

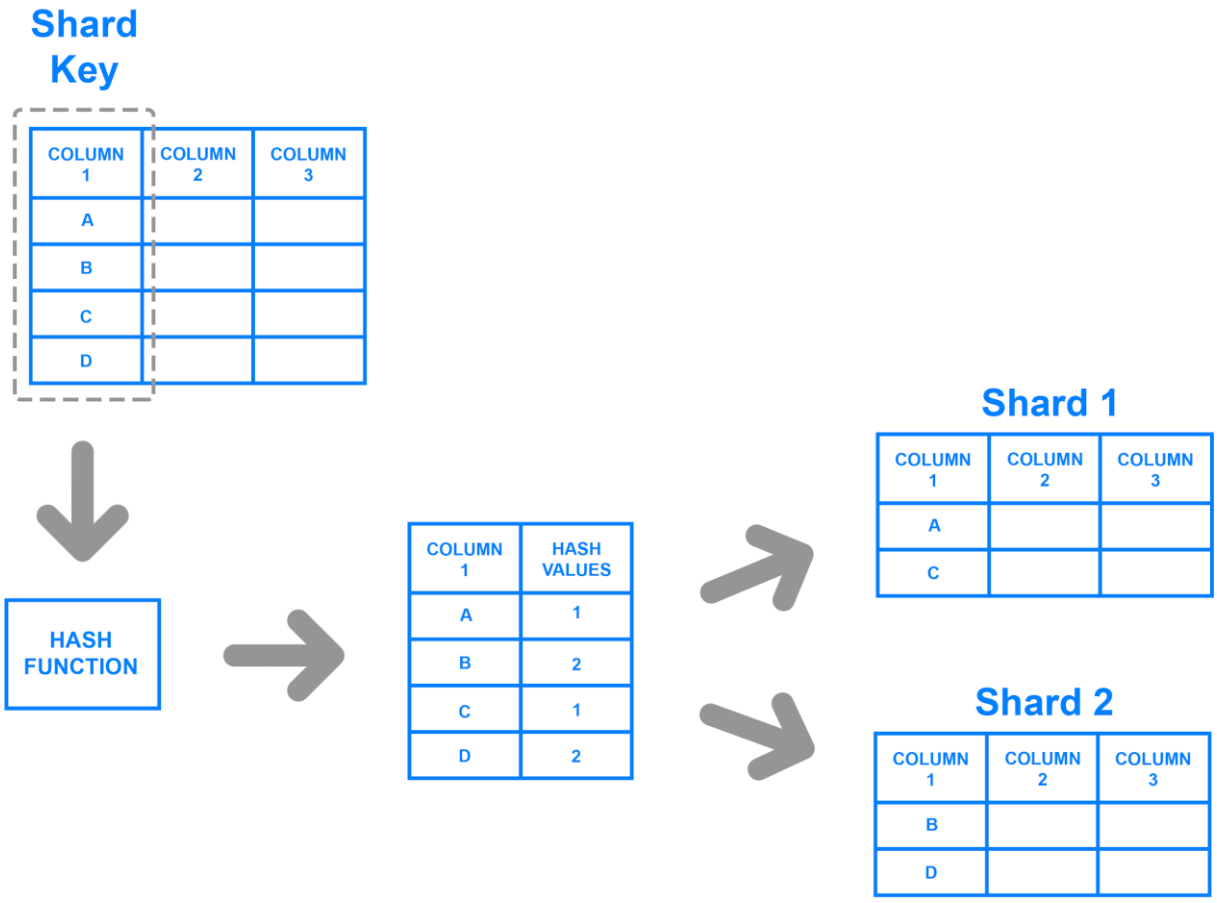



Рисунок 2.7

Тож було вирішено використовувати систему керування базами даних (СКБД) MongoDB. Це NoSQL СКБД, котра дозволяє налаштувати шардування та реплікацію (рисунок 2.8) в декілька простих кроків, адже з самого початку вона була спроектована для цих цілей. До того ж, в даній СКБД оперувати записами (або ж документами в термінології NoSQL СКБД, котрі всі в основному є документно-орієнтованими) зі складною структурою легше, адже вони представлені у JSON-подібному форматі, що дозволяє використовувати вкладеність. Також немає обмеження відповідності документів схемі, бо NoSQL СКБД є безсхемними. Звичайно, це може додавати деяких проблем із помилковими документами, записаними в базі даних, але при великих об'ємах даних та складною структурою з безліччю умов, цей підхід надає перевагу над класичними SQL СКБД.

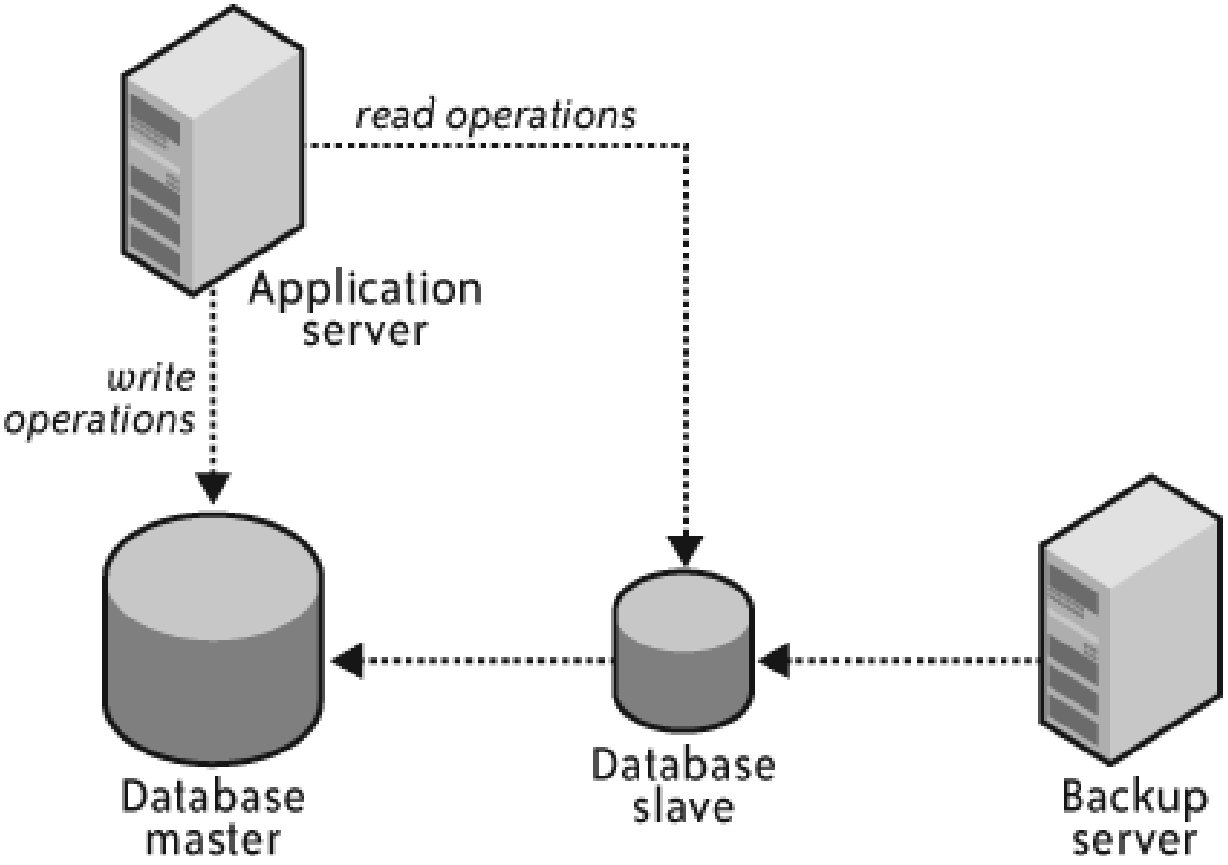



Рисунок 2.8

## 2.4. Інші способи масштабування

Також для більшої ефективності системи використовується кешування. Одним із способів є content delivery network (CDN) - мережа розповсюдження контенту - це географічно розподілена мережа проксі-серверів і їх центрів обробки даних. Мета полягає в тому, щоб забезпечити високу доступність і високу продуктивність, поширюючи сервіс просторово відносно кінцевих користувачів. CDN сьогодні обслуговують більшу частину інтернет-контенту, включаючи веб-об'єкти (текст, графіку та сценарії), завантажувані об'єкти (мультимедійні файли, програмне забезпечення, документи), програми (електронна комерція, портали), потокове мультимедіа, потокове відео медіа та сайти соціальних медіа.

CDN - це шар в екосистемі Інтернету. Власники контенту, такі як медіа-компанії та постачальники електронної комерції, платять операторам CDN за доставку їхнього контенту своїм кінцевим користувачам. У свою чергу, CDN платить провайдерам, операторам і операторам мереж для розміщення своїх серверів у їх центрах обробки даних.

CDN - це охоплюючий термін, що включає різні види послуг доставки контенту: потокове відео, завантаження програмного забезпечення, прискорення веб- та мобільного контенту, ліцензований та керований CDN, прозоре кешування та послуги для вимірювання продуктивності CDN, балансування навантаження, комутації та аналітики. Постачальники CDN можуть переходити на інші галузі, такі як безпека, захист від DDoS та брандмауери веб-додатків (WAF), а також оптимізація WAN.

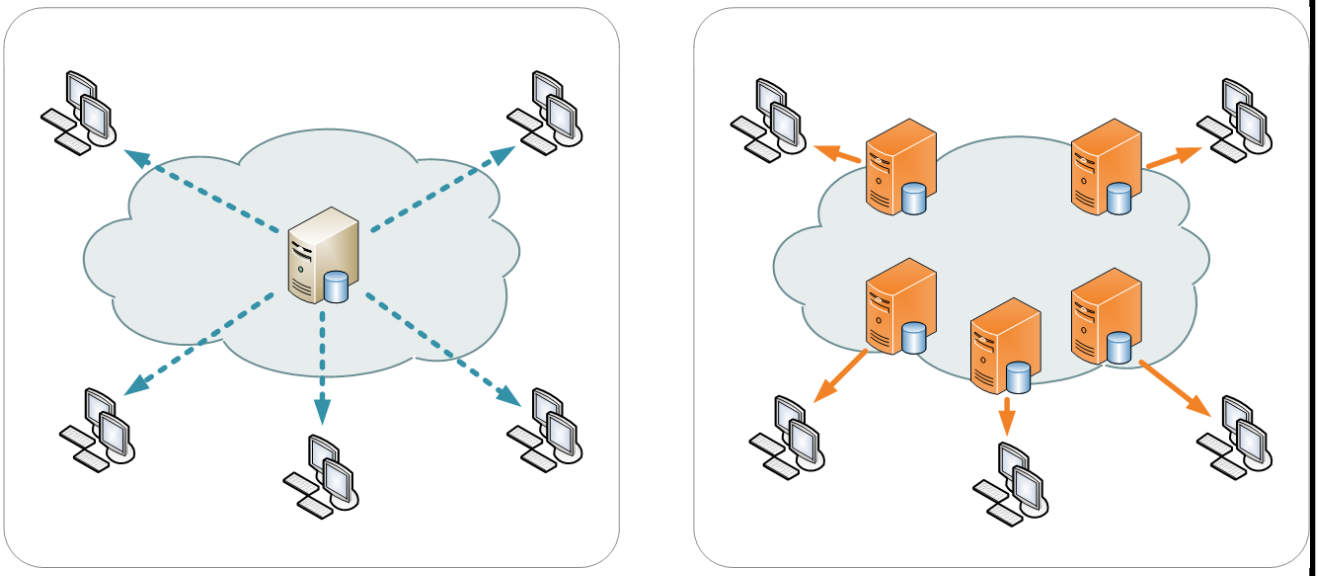


Рисунок 2.9

Вузли CDN зазвичай розгортаються в декількох місцях, часто на декількох магістралях (рисунок 2.9). Переваги включають скорочення витрат на пропускну спроможність, підвищення часу завантаження сторінки або збільшення доступності вмісту в усьому світі. Кількість вузлів і серверів, що складають CDN, змінюється залежно від архітектури, деякі досягають тисяч вузлів з десятками тисяч серверів на багатьох віддалених точках присутності.

Запити на вміст зазвичай алгоритмічно спрямовані на вузли, які певним чином є оптимальними. Під час оптимізації продуктивності можна вибрати місця, які найкраще обслуговують користувача. Це можна виміряти, вибравши місця, які є доступні користувачу з найменшою кількістю “хопів”, найменшу кількість секунд у мережі від клієнта, що запитує, або найвищу доступність з точки зору продуктивності сервера (поточної та історичної), щоб оптимізувати доставку через локальні мережі. При оптимізації вартості замість цього можна вибрати місця, які є найдешевшими.

Також для розгортання декількох екземплярів (а саме стільки, скільки доступно ядер на процесорі), використовується інструмент під назвою PM2 Runtime - це менеджер production процесів для програм Node.js із вбудованим балансуванням навантаження. Він дозволяє зберігати програми назавжди, перезавантажувати їх без простоїв і сприяти загальним завданням Devops.

В обчислювальних системах балансування навантаження покращує розподіл робочих навантажень на декількох обчислювальних ресурсах, таких як комп'ютери, комп'ютерний кластер, мережеві посилання або центральні процесори (рисунок 2.10). Балансування навантаження спрямоване на оптимізацію використання ресурсів, максимальну пропускну здатність, мінімізацію часу відгуку та уникнення перевантаження будь-якого ресурсу. Використання декількох компонентів з балансуванням навантаження замість одного компонента може підвищити надійність і доступність за допомогою надмірності. Балансування навантаження зазвичай передбачає виділене програмне забезпечення або апаратне забезпечення, наприклад багатошаровий комутатор або серверний процес системи доменних імен.

Балансування навантаження відрізняється від зв'язування каналів тим, що балансування навантаження розділяє трафік між мережевими інтерфейсами на основі мережевого сокету (четвертий рівень моделі OSI), тоді як зв'язок каналів передбачає розподіл трафіку між фізичними інтерфейсами на нижньому рівні.

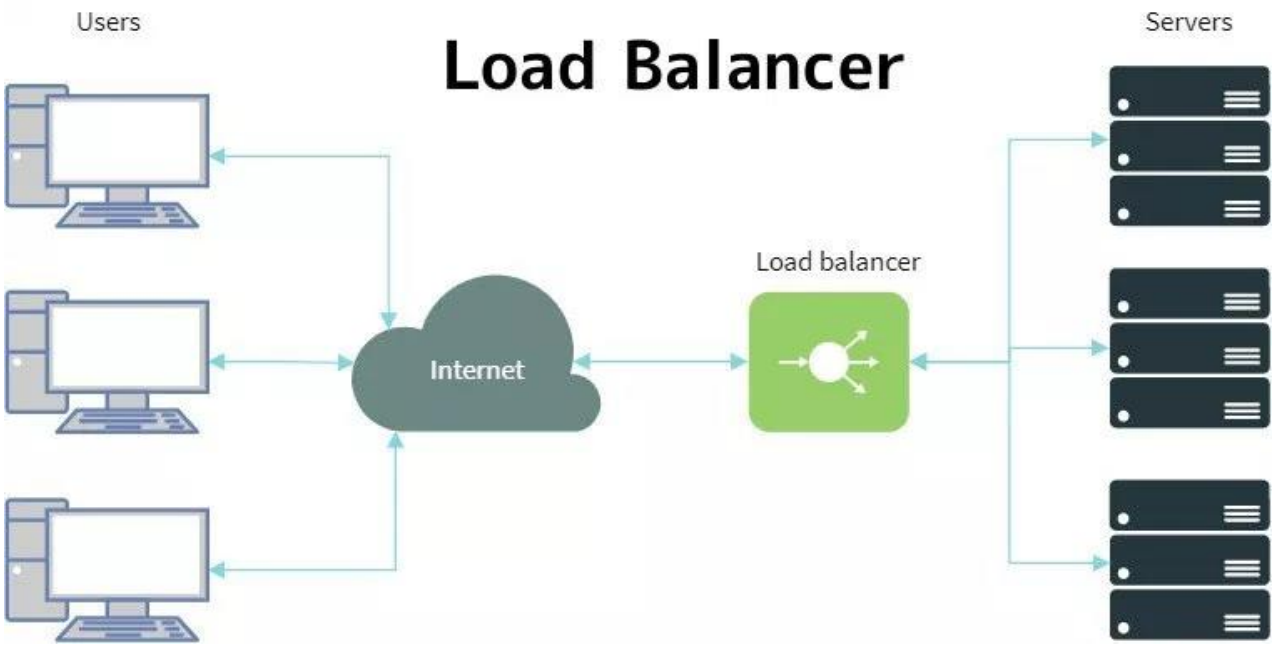



Рисунок 2.10

PM2 може масштабувати додаток по всіх доступних процесорах, створюючи кілька дочірніх процесів, які використовують один і той же порт

сервера. Він чудово працює для HTTP, TCP та UDP протоколів і може збільшити продуктивність на порядок на 16 основних машинах. Він також зменшує простої на оновленнях.

Це потужний інструмент для розгортання систем вже з вбудованим масштабуванням та балансувальником навантаження.

# 3. Структура системи та опис роботи модулів

## 3.1. Система керування базою даних (СКБД)

База даних складається з двох колекцій: models та users. В колекції users зберігається вся інформація про користувача, його унікальне ім’я та фотографії у форматі base64 (рисунок 3.1).

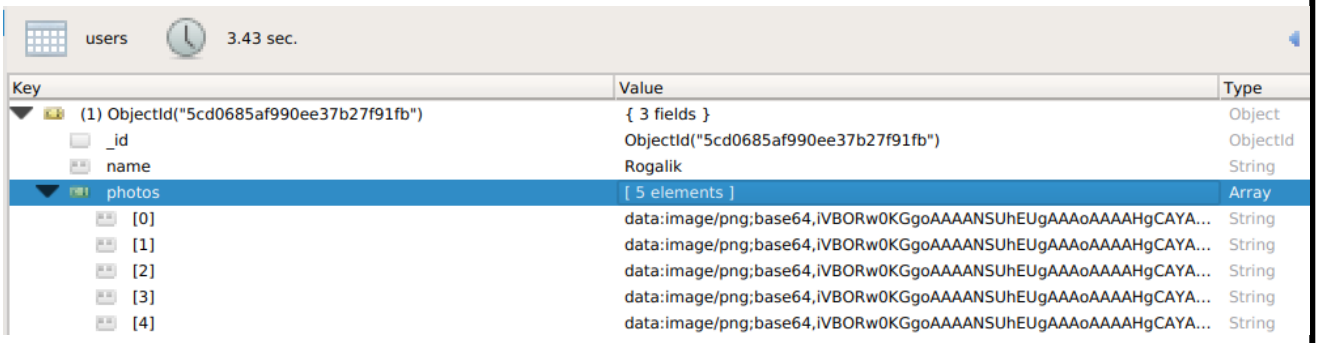



Рисунок 3.1

Моделі користувачів для аналізування та розпізнавання обличчя знаходяться в іншій колекції - models (рисунок 3.2). Для правильної роботи підсистеми face recognition їй потрібна одразу вся інформація про користувачів після їх класифікації, тому всі ці дані зберігаються в одному документі. Очевидно, розмір цього документу стрімко збільшується, адже модель кожного користувача займає приблизно 10 мегабайт. За замовчуванням обмеження на розмір одного документу в СКБД MongoDB встановлене в 16 мегабайт. Аби обійти це обмеження, MongoDB надає інструмент під назвою GridFS.

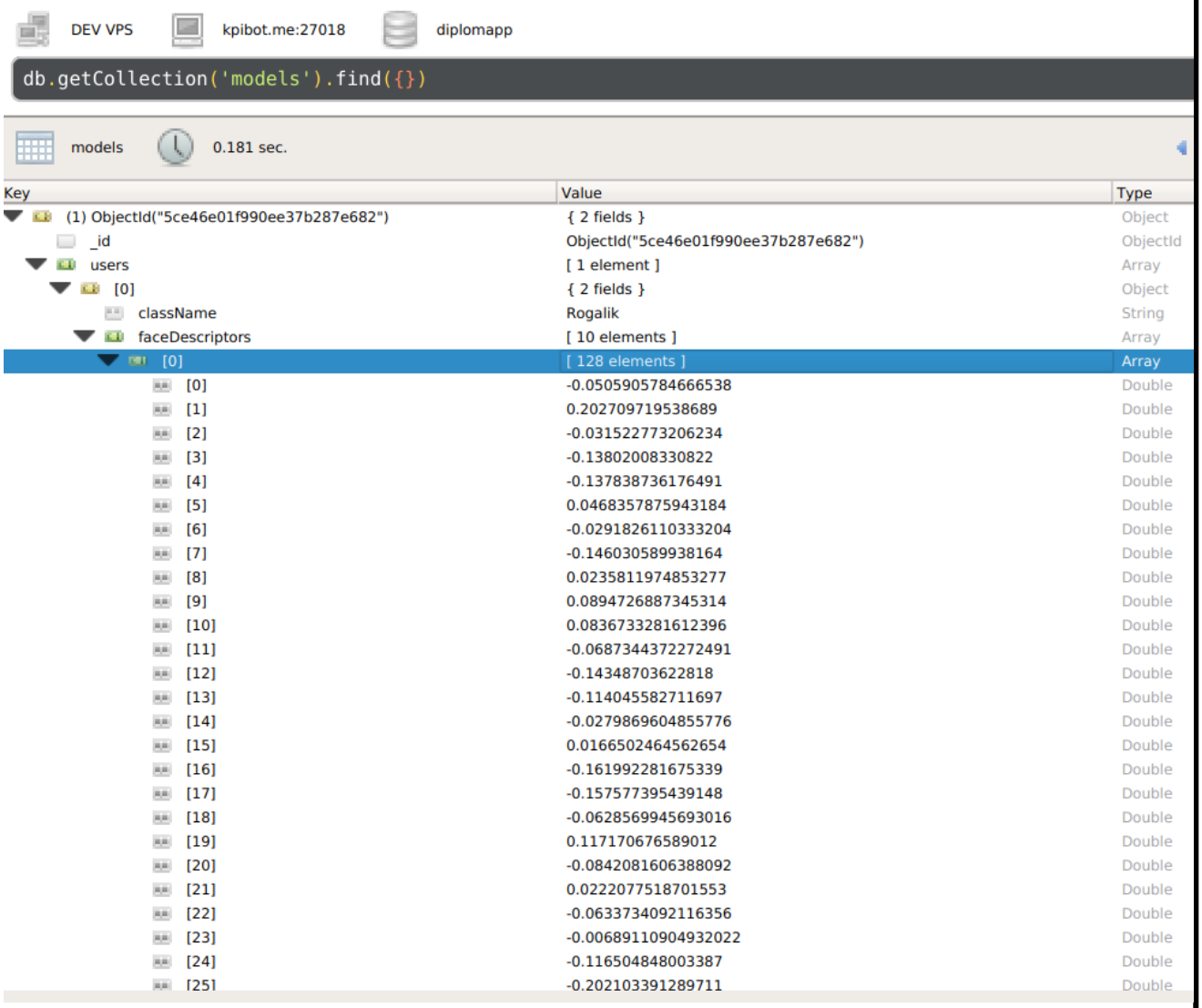



Рисунок 3.2

Замість того, щоб зберігати файл в одному документі, GridFS ділить файл на частини або фрагменти і зберігає кожен фрагмент як окремий документ. За замовчуванням GridFS використовує розмір фрагменту за замовчуванням 255 Кб; GridFS поділяє файл на блоки розміром 255 КБ, за винятком останнього фрагмента. Останній фрагмент має рівно необхідній розмір. Аналогічно, файли, які не перевищують розмір фрагмента, використовують лише стільки місця, скільки потрібно, плюс деякі додаткові метадані.

GridFS використовує дві колекції для зберігання файлів. Одна з них зберігає файлові шматки, а інші файли зберігають метадані. У розділі Колекції GridFS детально описується кожна колекція.

Коли ви надсилаєте запит до GridFS, щоб дістати файл, драйвер автоматично збирає цей файл з фрагментів, якщо необхідно. Також можливо виконувати діапазонні запити на файли, що зберігаються з використанням GridFS. Ви також можете отримати доступ до інформації з довільних розділів файлів, наприклад, щоб "пропустити" до середини відео- або аудіофайлу.

GridFS корисний не тільки для зберігання файлів, що перевищують 16 МБ, але й для зберігання будь-яких файлів, для потрібно отримати доступ, без необхідності завантажувати весь файл у пам'ять.

У деяких ситуаціях зберігання великих файлів може бути більш ефективним у базі даних MongoDB, ніж у файловій системі на рівні системи.

1. Якщо файлова система обмежує кількість файлів у каталозі, краще використовувати GridFS для зберігання необхідної кількості файлів.
2. Якщо є необхідність отримати доступ до інформації з частин великих файлів без завантаження цілих файлів в пам'ять, краще використовувати GridFS для відтворення розділів файлів, не читаючи весь файл у пам'ять.
3. Якщо є необхідність у тому, щоб файли та метадані автоматично синхронізувалися і розгорталися в ряді систем і засобів, можна використовувати GridFS. Використовуючи географічно розподілені набори реплік, MongoDB може автоматично поширювати файли та їх метадані на декілька екземплярів mongod.

Також краще не використовувати GridFS, якщо потрібно оновити вміст всього файлу атомічно. В якості альтернативи можна зберігати кілька версій кожного файлу і вказувати поточну версію файлу в метаданих. Після завантаження нової версії файлу можна оновити поле метаданих, яке вказує на "останній" статус, а потім видаляти попередні версії.

Крім того, якщо файли менші, ніж обмеження розміру документа BSON 16 МБ, є можливість зберігання кожного файлу в одному документі замість використання GridFS. Також можна використовувати тип даних BinData для зберігання двійкових даних.

GridFS зберігає файли в двох колекціях (рисунок 3.3):

1. Колекція chunks зберігає двійкові фрагменти даних. Кожен документ у цій колекції є окремим фрагментом файлу, представленим у GridFS;
2. Колекція files зберігає метадані файлів. Кожен документ у цій колекції є файлом у GridFS;

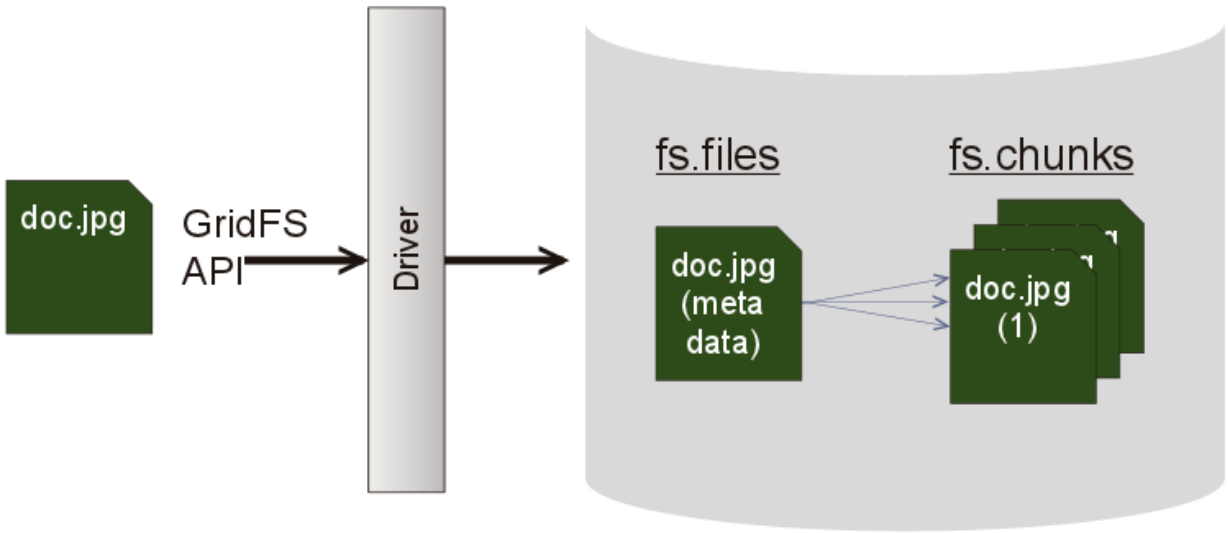



Рисунок 3.3

## 3.2. Модуль обробки фотознімків

Модуль приймає запити на такі ресурси:

1. GET /info;
2. POST /user;
3. POST /photo.

Важливо, що всі операції в системі мають бути ідемпотентними, адже досить часто через проблеми мережі запити повторюються, тому цей підхід є дуже важливим в проектуванні та розробці програмного забезпечення, особливо з мікросервісною архітектурою у випадку, коли мікросервіси комунікують один з одним використовуючи HTTP(S). Один з прикладів

проблем, котрі можуть виникнути, показаний на рисунку 3.4. Так, запит може бути прийнятим та обробленим системою, але відправник не отримає підтвердження, тому запит буде надіслано повторно.

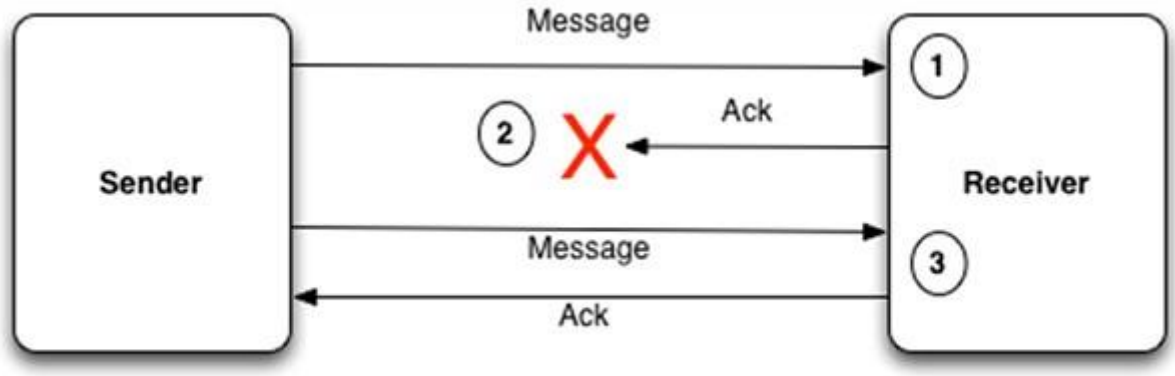



Рисунок 3.4

Взагалі, метод HTTP є ідемпотентним, якщо повторний ідентичний запит, зроблений один або кілька разів підряд, має один і той же ефект, не змінюючи стан системи. Іншими словами, ідемпотентний метод не повинен мати ніяких побічних ефектів, крім збору статистики або подібних операцій.

Запит GET /info повертає інформацію на даний момент про систему: її кодову назву, версію та ін. Версія системи використовується для визначення, яка нова функціональність вже наявна і якої поведінки від системи чекати, адже з кожною версією вона змінюється. Також для кожного окремого мікросервісу використовується своє власне версіонування. Для цього використовується методика під назвою Semver.

В розробці програмного зебезпечення існує важливе місце, яке називається "пекло залежностей". Чим більше система зростає і чим більше пакетів інтегрується в програмне забезпечення, тим більше шансів попасти в це місце

У системах з багатьма залежностями, випуск нових версій пакетів може швидко стати кошмаром. Якщо специфікації залежностей надто тісні, ви знаходитесь в небезпеці блокування версії (неможливість оновити пакет без випуску нових версій кожного залежного пакета).

Як вирішення цієї проблеми пропонується простий набір правил і вимог, які диктують, як номери версій призначаються і збільшуються. Ці правила ґрунтуються (але не обов'язково) на попередньо існуючих поширених практиках використання як в закритому, так і в відкритому програмному забезпеченні. Щоб ця система працювала, спочатку потрібно оголосити відкритий API. Важливо, щоб цей API був чітким і точним. Після того, як загальнодоступний API проідентифікований, потрібно повідомити про зміни до нього з певними збільшеннями до номера нової версії. Розглянемо формат версії X.Y.Z (Major.Minor.Patch). Виправлення помилок, які не впливають на роботу API, збільшують Patch версю, зворотно сумісні зміни або збільшення функціональності API збільшують Minor версію, а зворотні несумісні зміни API збільшують Major версію.

Це не нова або революційна ідея. Але без відповідності певній формальній специфікації номери версій, по суті, марні для управління залежностями. Надаючи ім'я та чітке визначення вищенаведеним ідеям, стає легким передати свої наміри користувачам вашого програмного забезпечення. Після того, як ці наміри стануть зрозумілими, нарешті можуть бути зроблені гнучкі (але не занадто гнучкі) специфікації залежності.

До того ж, мікросервісам важливо використовувати актуальну особисто для них версію інших мікросервісів. Щоб цей процес проходив під контролем, без нумерації версій не обійтись.

Запит POST /user створює нового користувача на основі його імені та списку фотографій. При цьому відбувається аналіз фотографій для побудови унікальної моделі опорно-векторної машини (ОВМ) користувача - дев’ять 128-вимірних векторів на основі 5-ти фотографій. 9 векторів замість одного значно підвищують відсоток правильних розпізнавань. Для побудови такої моделі використовується згорткова нейронна мережа (ЗНМ).

Метою алгоритму опорно-векторної машини є пошук гіперплощини в N-мірному просторі (N - кількість ознак), що чітко класифікує точки даних[6].

Щоб відокремити два класи точок даних, можна вибрати багато можливих гіперплощин. Метою є знайти площину, яка має максимальний запас, тобто максимальну відстань між точками даних обох класів (рисунок 3.5). Максимізація максимальної відстані забезпечує деяке підсилення, так що майбутні дані можуть бути класифіковані з більшою впевненістю.

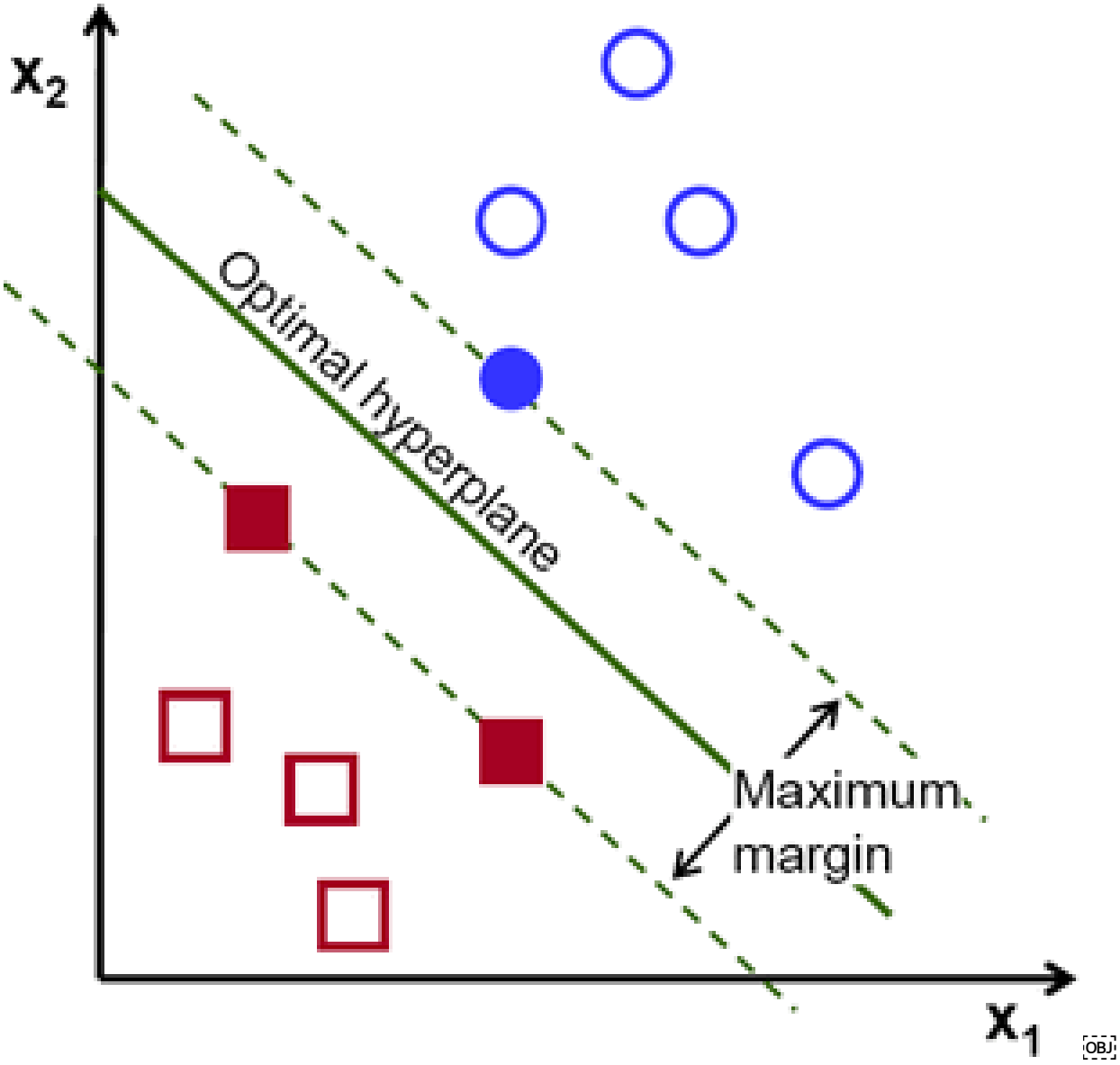



Рисунок 3.5

Гіперплощини - це межі рішень, які допомагають класифікувати точки даних. Точки даних, що падають на будь-яку сторону гіперплощини, можна віднести до різних класів. Крім того, розмір гіперплощини залежить від кількості функцій. Якщо кількість вхідних функцій дорівнює 2, то

гіперплощина - це лише лінія. Якщо число вхідних ознак дорівнює 3, то гіперплощина стає двовимірною площиною. Важко уявити, коли кількість ознак перевищує 3 (рисунок 3.6).

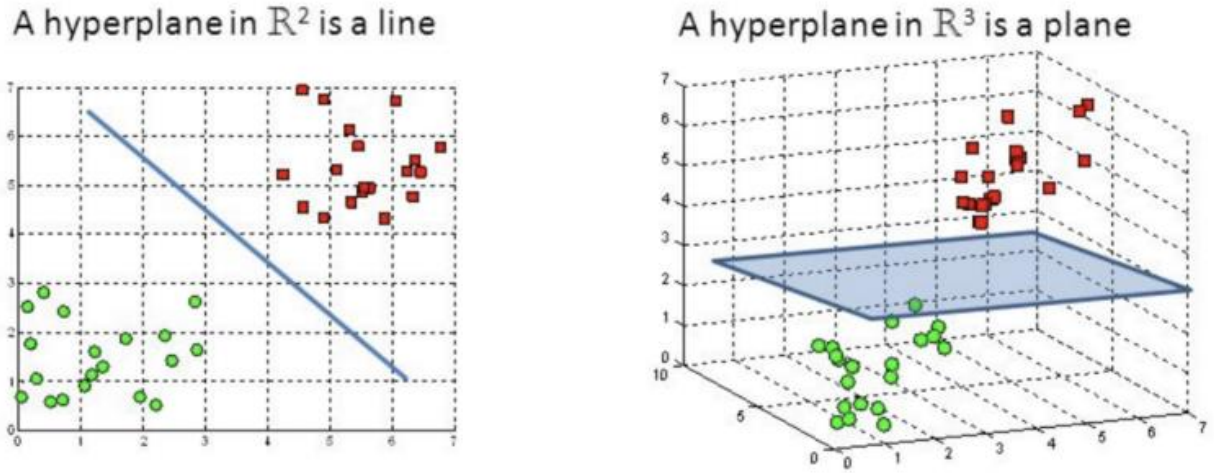



Рисунок 3.6

Вектори підтримки є точками даних, які знаходяться ближче до гіперплощини і впливають на положення і орієнтацію гіперплощини (рисунок 3.7). Використовуючи ці вектори підтримки, максимізується відстань для класифікатора. Видалення векторів підтримки змінить положення гіперплощини. Це ті пункти, які допомагають побудувати ОВМ.

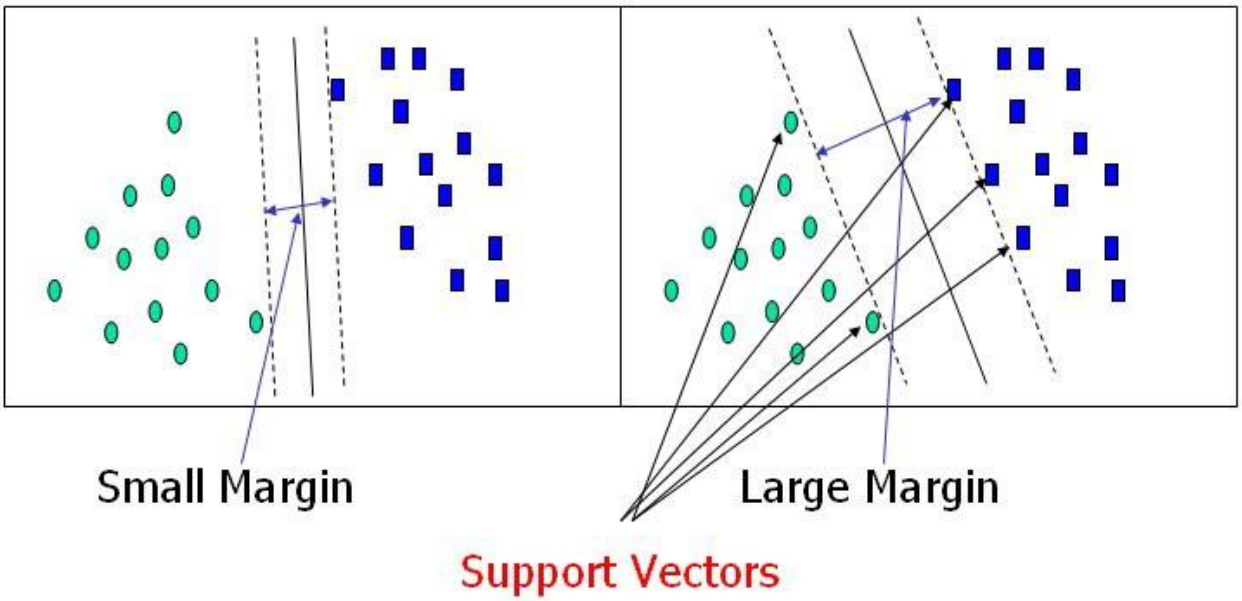



Рисунок 3.7

У алгоритмі ОВМ основною метою є максимізувати відстань між точками даних і гіперплощиною. Функція втрат, яка допомагає максимізувати цю відстань, називається завісні втрати (рисунок 3.8).

$$c(x, y, f(x)) = \begin{cases} 0, & \text{if } y * f(x) \geq 1 \\ 1 - y * f(x), & \text{else} \end{cases}$$

Рисунок 3.8

Вартість становить 0, якщо передбачене значення і фактичне значення мають один і той же знак. Якщо ця умова не відбувається, тоді обчислюється значення втрат. Також додається параметр регуляризації функції втрат. Метою параметра регуляризації є збалансування максимальної відстані та втрати. Після додавання параметра регуляризації, функції витрат виглядають, як показано на рисунку 3.0.

$$min_w \lambda \parallel w \parallel^2 + \sum_{i=1}^{n} (1 - y_i \langle x_i, w \rangle)_+$$

Рисунок 3.9

Тепер, коли є функція втрат, використовуються часткові похідні відносно ваг, щоб знайти градієнти. За допомогою градієнтів, можна оновити ваги (рисунок 3.10).

$$\frac{\delta}{\delta w_k} \lambda \parallel w \parallel^2 = 2\lambda w_k$$

$$\frac{\delta}{\delta w_k} \left(1 - y_i \langle x_i, w \rangle\right)_+ = \begin{cases} 0, & \text{if } y_i \langle x_i, w \rangle \geq 1 \\ -y_i x_{ik}, & \text{else} \end{cases}$$

Рисунок 3.10

Коли немає неправильної класифікації, тобто модель правильно передбачає клас точки даних, потрібно лише оновити градієнт від параметра регуляризації (рисунок 3.11).

$$w = w - \alpha \cdot (2\lambda w)$$

Рисунок 3.11

Коли існує помилка класифікації, тобто модель помиляється на прогнозуванні класу точки даних, включаються втрати разом з параметром регуляризації для виконання оновлення градієнта (рисунок 3.12).

$$w = w + \alpha \cdot (y_i \cdot x_i - 2\lambda w)$$

Рисунок 3.12

ЗНМ - це алгоритм глибокого навчання, який може приймати вхідне зображення, призначати важливість (навчальні ваги і упередження) різним аспектам та об'єктам на зображенні і вміти диференціювати один з іншого (рисунок 3.13). Попередня обробка, необхідна в ЗНМ, значно менша в порівнянні з іншими алгоритмами класифікації. Хоча в примітивних методах фільтри сконструйовані вручну, з достатньою кількістю тренувань, ЗНМ мають можливість оброблювати ці фільтри та характеристики.

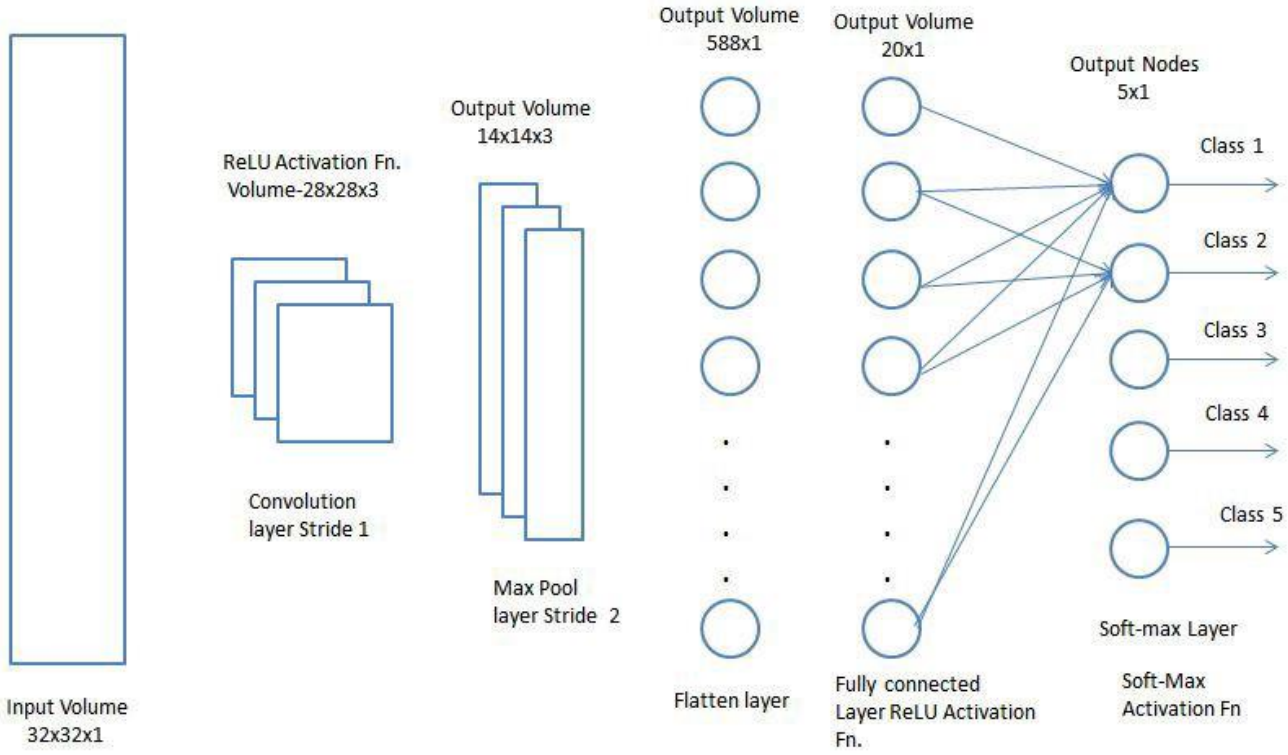



Рисунок 3.13

Архітектура ЗНМ аналогічна архітектурі зв'язності нейронів в людському мозку. Окремі нейрони реагують на подразники тільки в обмеженому регіоні поля зору, відомому як Рецептивне поле. Колекція таких полів перекривається, щоб охопити всю зону зору.

ЗНМ здатна успішно захоплювати просторові та тимчасові залежності у зображенні за допомогою застосування відповідних фільтрів. Архітектура краще підходить до набору зображень за рахунок зменшення кількості задіяних параметрів і повторного використання ваг. Іншими словами, мережу можна навчити краще розуміти складність зображення.

Тож в даному дипломному проекті модель користувача (або векторне вбудовування) - це відображення зображень його обличчя в числові векторні представлення шляхом проходження їх через згорткові нейронні мережі.

Оскільки ці векторні вбудовування представлені в спільному векторному просторі, векторну відстань можна використовувати для обчислення подібності між двома векторами. У контексті розпізнавання облич, ця векторна відстань може бути застосована для розрахунку того, наскільки подібні два обличчя. Крім того, ці вбудовування можуть бути використані як вхідні елементи у класифікації, кластеризації або регресії.

Для розробки використовувалась бібліотека OpenCV. Сама бібліотека реалізована за допомогою C++, але завдяки аддонам в Node.js, будь-який код на C чи C++ може бути підготовлений для використання в Node.js, хоча мовою програмування цієї програмної платформи є JavaScript. Аддона Node.js - це динамічно пов'язані спільні об'єкти, написані на мові C ++, які можна завантажити в Node.js, використовуючи функцію require(), і використовувати їх так само, як якщо б вони були звичайним модулем Node.js.

В свою чергу, запити на POST /photo обробляються, аналізуються фотознімки безпосередньо об’єкту охорони, намагаючись розпізнати обличчя людини та визначити хто саме зображений на знімку. Також крім знімків, обробляється інформація про місцерозташування камери та її власника. Якщо такий знімок був оброблений та було визначено обличчя людини, ця інформація передається в інший мікросервіс, який відповідає за сповіщення.

## 3.3. Модуль сповіщення

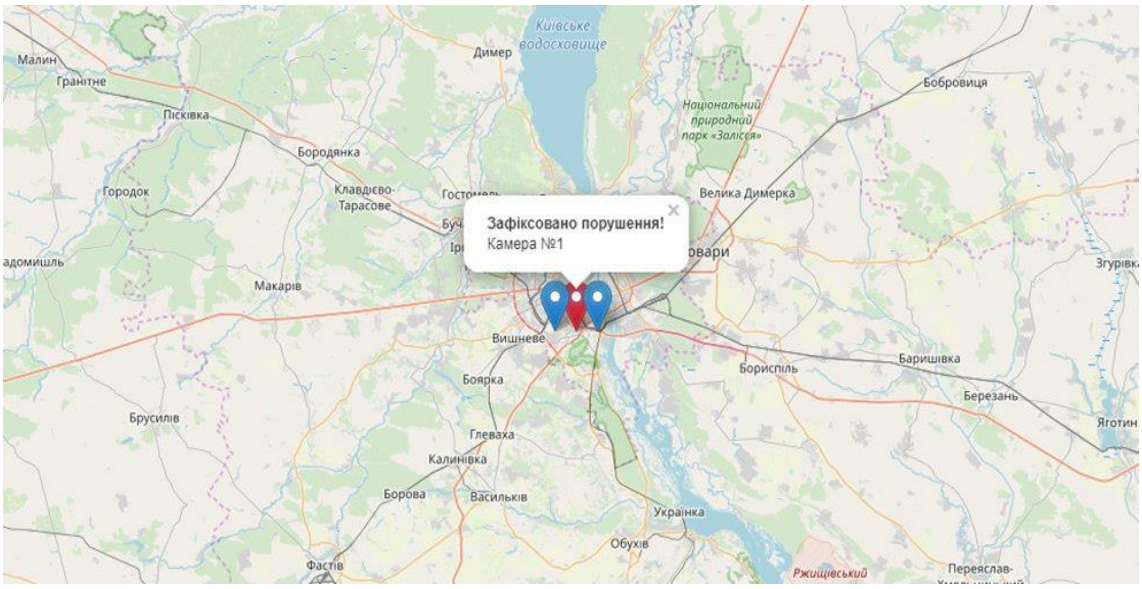



Рисунок 3.14

Модуль сповіщення є окремим мікросервісом, котрий приймає запити від модуля обробки фотографій на ресурс POST /alarm з усією потрібною інформацією. Усі клієнти системи підключаються саме до цього модуля за допомогою протоколу WebSocket. Тож при прийнятті такого запиту, інформація про виявлення обличчя людини відразу відправляється усім клієнтам системи (рисунок 3.14).

WebSocket - це комп'ютерний комунікаційний протокол, який забезпечує повний дуплексний канал зв'язку через одне з'єднання TCP. WebSocket відрізняється від HTTP. Обидва протоколи знаходяться на рівні 7 в моделі OSI і залежать від TCP на рівні 4. Хоча вони відрізняються, RFC (Request for Comments) 6455 стверджує, що WebSocket призначений для роботи над HTTP портами 80 і 443, а також для підтримки HTTP проксі і посередників. Це робить його сумісним з протоколом HTTP. Для досягнення сумісності WebSocket використовує заголовок HTTP Upgrade для переходу від протоколу HTTP до протоколу WebSocket.

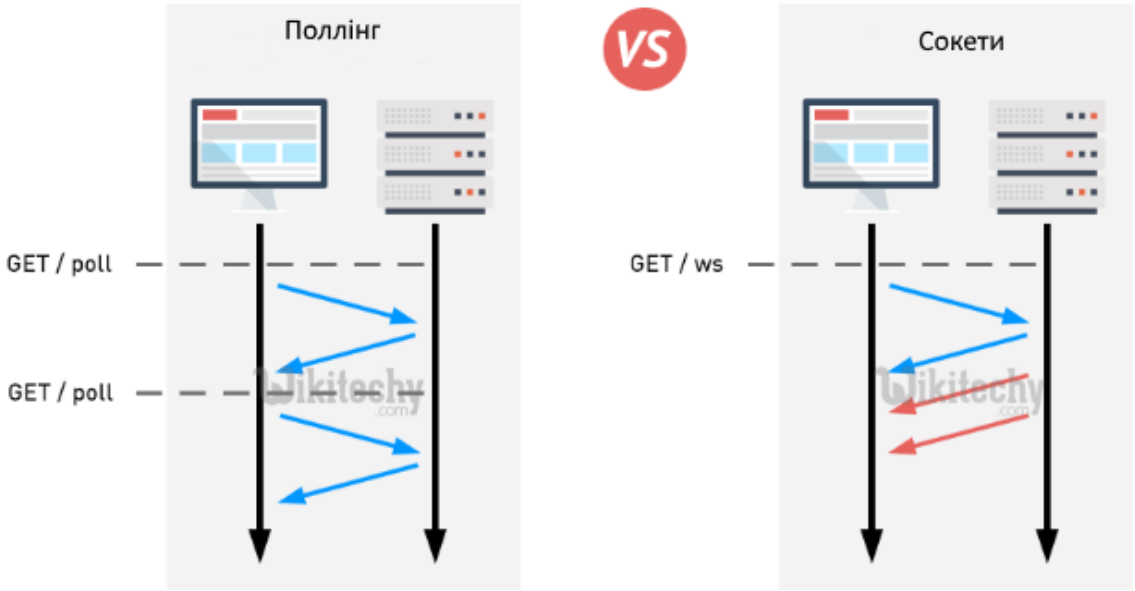
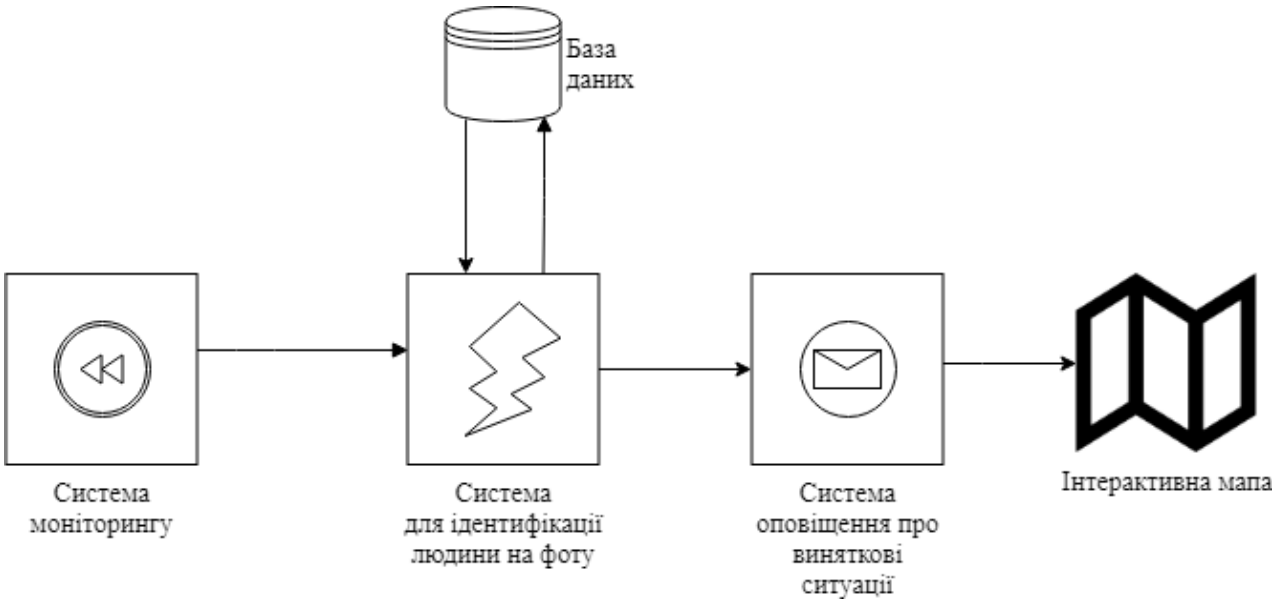



Рисунок 3.15

Протокол WebSocket дозволяє взаємодіяти між веб-браузером (або іншим клієнтським додатком) і веб-сервером з меншими накладними витратами, ніж напівдуплексні альтернативи, такі як HTTP-опитування, що полегшує передачу даних у реальному часі від і до сервера (рисунок 3.15). Це стало можливим завдяки стандартизованому способу передачі кентенту сервером до клієнта без попереднього запиту клієнтом, а також дозволяючи повідомленням передаватися вперед і назад, зберігаючи відкрите з'єднання. Таким чином, між клієнтом і сервером може відбуватися двостороння комунікація. Комунікації здійснюються через TCP-порт № 80 (або 443 у випадку TLS-зашифрованих з'єднань), що є корисним для тих середовищ, які блокують не-веб-підключення до Інтернету за допомогою брандмауера.




Загалом усі модулі системи та їх зв’язок зображений на рисунку 3.16. Важливо зазначити, що вся комунікація в системи відбувається за допомогою JSON Web Token (JWT) - це відкритий стандарт на основі JSON (RFC 7519) для створення маркерів доступу, які підтверджують деяку кількість вимог. Наприклад, сервер може генерувати маркер, який має значення "увійшов до системи як адміністратор" і надає його клієнту. Потім клієнт може використовувати цей маркер, щоб довести, що він увійшов до системи як адміністратор. Токени підписуються приватним ключем однієї сторони (як правило, сервера), так що обидві сторони можуть перевірити, що маркер є правильним. Маркери розроблені так, щоб бути компактними, URL-безпечними, і придатними для використання в контексті веб-браузера. JWT зазвичай можуть використовуватися для передачі ідентичності користувачів, що пройшли аутентифікацію, між постачальником ідентифікаційних даних і постачальником послуг (рисунок 3.17).

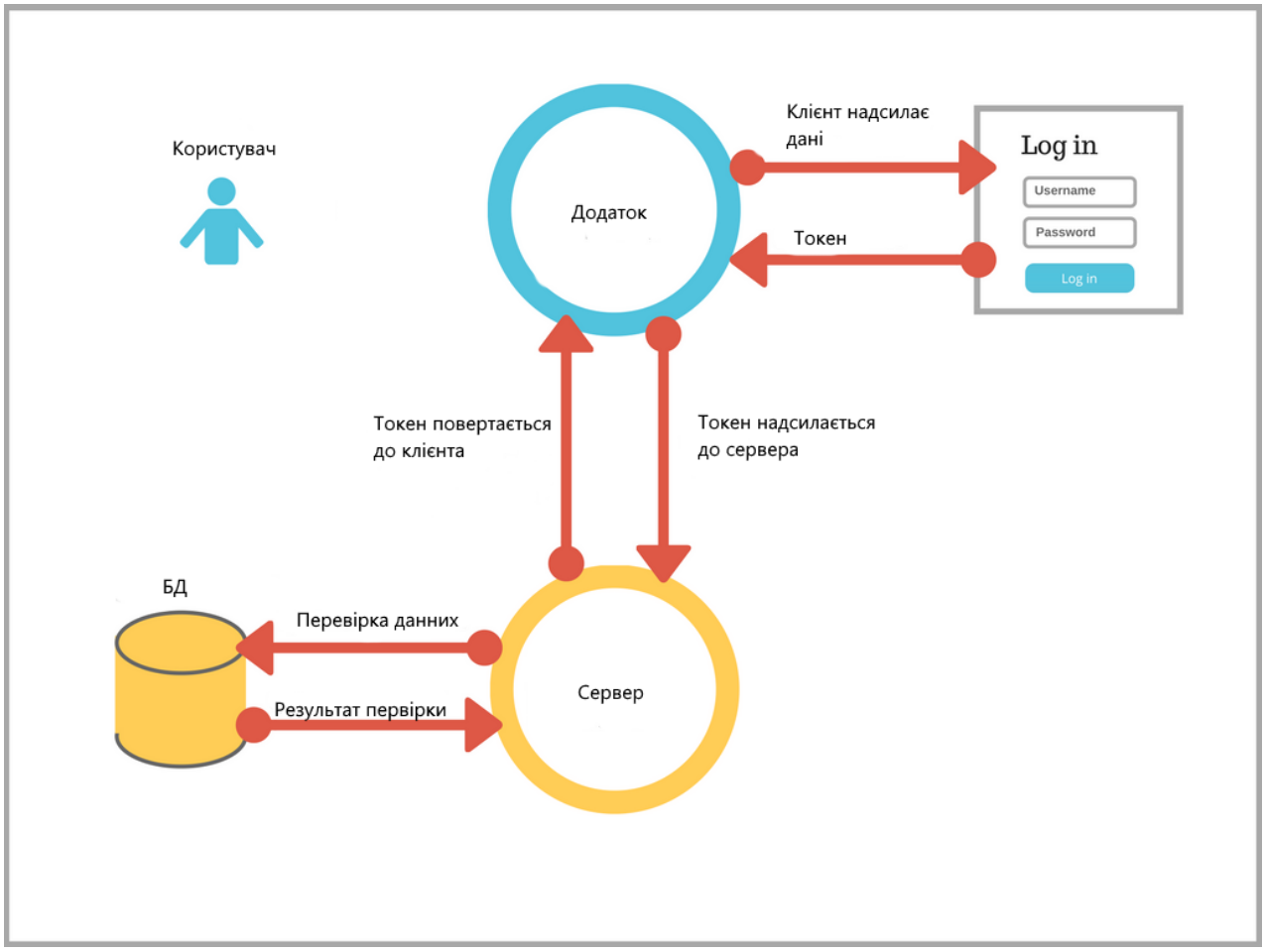



Рисунок 3.17

Під час аутентифікації, коли користувач успішно входить до системи з використанням своїх облікових даних, веб-маркер JSON повертається і повинен бути збережений локально (зазвичай у локальному або сеансовому сховищі, але також можна використовувати файли cookie) замість традиційного підходу створення сеансу на сервері та повернення cookie.

# 4. ТЕСТУВАННЯ СИСТЕМИ ТА АНАЛІЗ РЕЗУЛЬТАТІВ

## 4.1. Тестування роботи

Під час тестування були перевірені основні модулі системи. Приклад тіла запиту для створення нового користувача системи зображений на рисунку 4.1.


Рисунок 4.1

В результаті цього запиту була створена ОВМ модель користувача та збережена в базу даних (рисунок 4.2).

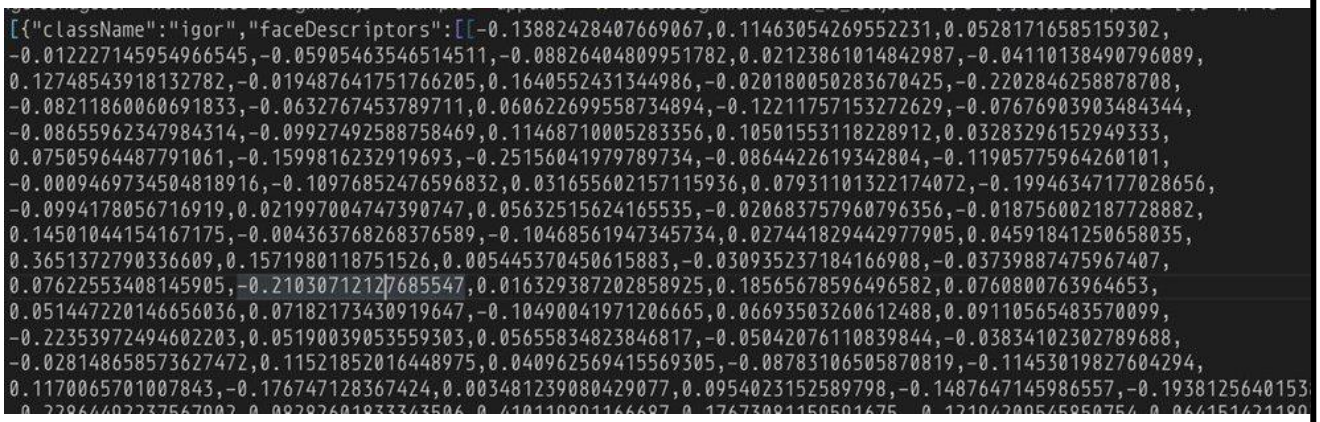



Рисунок 4.2

При тестуванні модуля сповіщення було перевірено роботу коректної передачі інформації про виявлення або невиявлення обличчя людини на фотознімках (рисунок 4.3)

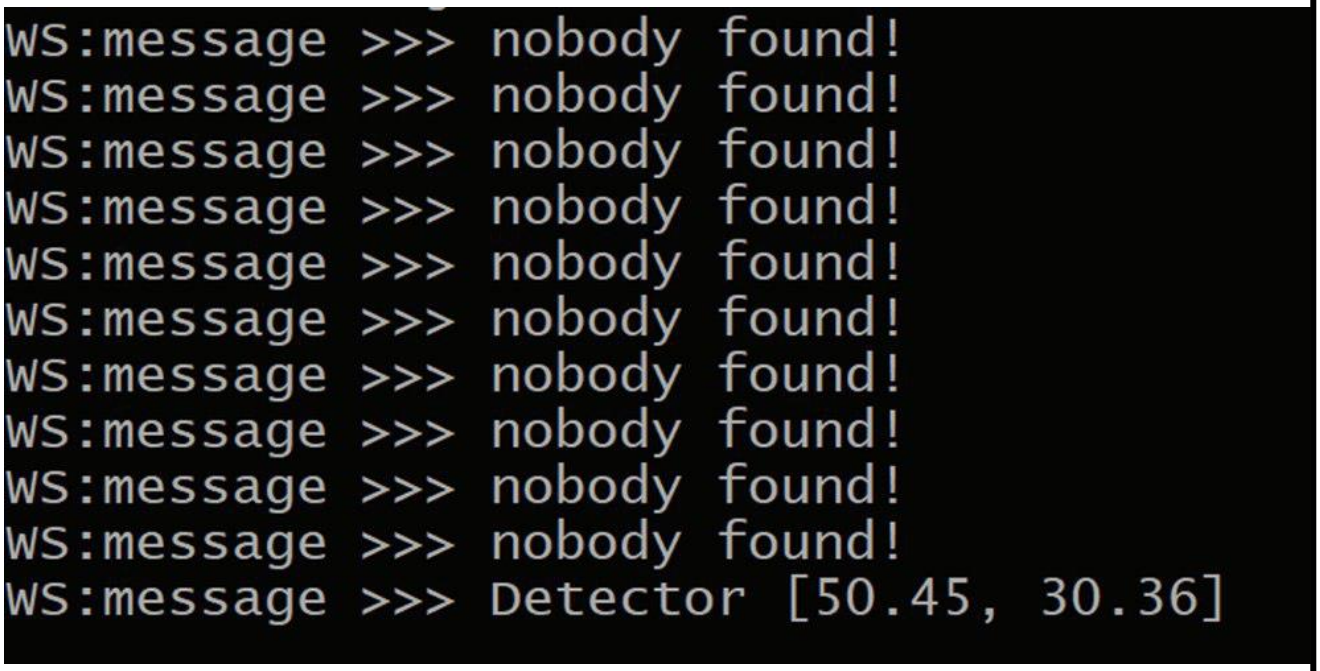



Рисунок 4.3

Також за допомогою linux-утиліти для навантажувального тестування siege були виміряні результати роботи системи при обробці фотографій, емулуючи 25 одночасних користувачів, котрі в середньому відсилали 17 запитів в секунду (рисунок 4.4).

```
✔ 05:21 ~/repos/diplomapp [master|…1] $ node test.js
Ben: $ siege -u kpibot.me:3003/photo -d1 -r10 -c25
..Siege 2.65 2006/05/11 23:42:16
..Preparing 25 concurrent users for battle.
The server is now under siege…done
Transactions: 250 hits
Elapsed time: 14.67 secs
Data transferred: 448000 MBytes
Response time: 1.43 secs
Transaction rate: 17.04 trans/sec
Throughput: 30538.51 bytes/sec
Concurrency: 7.38
Status code 200: 250
Successful transactions: 250
Failed transactions: 0
```

Рисунок 4.4

При таких умовах система обробляє запити в середньому за 1.43 секунди, що є непоганим результатом. Важливо зауважити, що

тестування відбувались не локально, а використовувався віддалений сервер.

# ВИСНОВКИ

Головною метою даного дипломного проекту було створити легко масштабовану систему моніторингу об’єктів охорони з високою швидкістю сповіщення. Система розроблялась з допомогою сучасних технологій веб-розробки, були проаналізовані та розглянуті їх переваги над класичними методами і підходами. Таким чином вдалось подолати розповсюджені проблеми в існуючих програмних рішеннях.

Також в ході розробки було реалізовано певну базу для побудови геоінформаційної системи, котра може бути досить ефективною в галузі охоронних систем. Сучасні продукти в цій галузі ще не почали впроваджувати дану технологію, але її потенціал важко переоцінити.

Одним з варіантів подальшого розвитку проекту є розширення можливих датчиків для підключення в розроблену систему, такі як: датчики звуку, руху, світла тощо. Адже дані пристрої також доступні на більшості мобільних платформах. Також важливо додавати нові шляхи сповіщення та їх дублювання для більшої гарантії реагування від профільних служб чи власника. Наприклад: СМС-повідомлення, електронна пошта, телефонні дзвінки тощо.

Загалом розроблена система продемонструвала свою надійність та здатність до існування.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ


1. Веб-ресурс охоронної системи на основі AI wave2cloud [Електронний ресурс]. – 2015. – Режим доступу: https://wave2cloud.com/. – Дата доступу: квітень 2019.
2. Lee, Seungmug; Wilson, Harry. "Spatial impact of burglar alarms on the decline of residential burglary" [Текст] - Palgrave Macmillan, a division of Macmillan Publishers Ltd, 2012. – стор. 85-93.
3. Американська компанія по стандартизації та сертифікації в галузі техніки безпеки [Електронний ресурс]. – 2014. – Режим доступу: https://www.ul.com/resources/alarm-system-description-forms. – Дата доступу: травень 2019.
4. Українська компанія по розробці охоронних систем [Електронний ресурс]. – 2012. – Режим доступу: https://ajax.systems/how-it-works/. – Дата доступу: березень 2019.
5. Компанія по розробці домашніх охоронних систем [Електронний ресурс]. – 2015. – Режим доступу: https://support.ring.com/. – Дата доступу: квітень 2019.
6. Стаття про метод опорних векторів [Електронний ресурс]. – 2017. – Режим доступу: https://towardsdatascience.com/support-vector-machine-introduction-to-machine-learning-algorithms-934a444fca47. – Дата доступу: лютий 2019.
7. Посилання на RPC стандарт WebSocket [Електронний ресурс]. – 2014. - Режим доступу: https://tools.ietf.org/html/rfc6455. – Дата доступу: березень 2019.
8. Establishing a Websocket PUBSUB server with Redis and Asyncio [Електронний ресурс]. – 2019. – Режим доступу: https://yeti.co/blog/establishing-a-websocket-pubsub-server-with-redis-and-asyncio-for-the-light-sensor/. – Дата доступу: травень 2019.

9. Medium RESTful-api [Електронний ресурс]. – 2018. – Режим доступу: https://medium.com/@lazlojuly/what-is-a-restful-api-fabb8dc2afeb. – Дата доступу: квітень 2019.