**НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ**
**«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ**
**імені ІГОРЯ СІКОРСЬКОГО»**
**ФАКУЛЬТЕТ ІНФОРМАТИКИ ТА ОБЧИСЛЮВАЛЬНОЇ ТЕХНІКИ**
**КАФЕДРА АВТОМАТИКИ ТА УПРАВЛІННЯ В ТЕХНІЧНИХ СИСТЕМАХ**

«На правах рукопису»
УДК _____

«До захисту допущено»
Завідувач кафедри
_____ _____
(підпис) (ініціали, прізвище)
“___”_____20__р.

# Магістерська дисертація

зі спеціальності (спеціалізації) 121, інженерія програмного забезпечення
(код і назва спеціальності)

(інженерія програмного забезпечення комп’ютерних систем)

на тему: Автоматизована система керування процесом розпізнавання зображень

Виконав: студент 2 курсу, групи ІТ-83мп
(шифр групи)

Рябчун Андрій Володимирович _____
(прізвище, ім’я, по батькові) (підпис)

Науковий керівник доцент кафедри АУТС, к.т.н, доцент Катін П.Ю. _____
(посада, науковий ступінь, вчене звання, прізвище та ініціали) (підпис)

Консультант _____ _____ _____
(назва розділу) (науковий ступінь, вчене звання, , прізвище, ініціали) (підпис)

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

Засвідчую, що у цій магістерській дисертації 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 (інститут) Інформатики та обчилювальної техніки
(повна назва)

Кафедра Кафедра автоматики та управління в технічних системах
(повна назва)

Рівень вищої освіти – другий (магістерський) за освітньо-професійною (освітньо-науковою) програмою

Спеціальність (спеціалізація) 121, інженерія програмного забезпечення
(код і назва спеціальності)

(інженерія програмного забезпечення комп'ютерних систем)

ЗАТВЕРДЖУЮ
Завідувач кафедри

_____ _____
(підпис) (ініціали, прізвище)

«___»_____20__ р.

**ЗАВДАННЯ**
**на магістерську дисертацію студенту**
Рябчуну Андрію Володимировичу
(прізвище, ім'я, по батькові)

1. Тема дисертації: Автоматизована система керування процесом розпізнавання зображень

науковий керівник дисертації доцент кафедри АУТС, к.т.н, доцент Катін П.Ю.,
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

затверджені наказом по університету від «___»_____ 20__ р. №_____

2. Строк подання студентом дисертації _____

3. Об'єкт дослідження автоматизація процесу взаємодії з моделями машиного навчання

4. Предмет дослідження (вихідні дані для магістерської дисертації за освітньо-професійною програмою) моделі машинного навчання в області комп'ютерного зору, хмарні технології

5. Перелік завдань, які потрібно розробити ознайомитися з

технологією комп'ютерного зору, сформувати вимоги до системи та сценарії використання, реалізувати алгоритми керування за допомогою програмних кодів, інтегрувати модель машиного навчання у вебсервіс, використовуючи отримані дані розробити систему управління процесом розпізнавання зображень

6. Орієнтовний перелік ілюстративного (графічного) матеріалу діаграма послідовності процесу розпізнавання, структурна діаграма, діаграма прецедентів, діаграма розміщення, діаграма діяльності процесу розпізнавання

7. Орієнтовний перелік публікацій

8. Консультанти розділів дисертації*

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

9. Дата видачі завдання 5 вересня 2019 року

Календарний план

| № з/п | Назва етапів виконання магістерської дисертації | Строк виконання етапів магістерської дисертації | Примітка |
|---|---|---|---|
| 1 | Порівняльний аналіз існуючих рішень | 29.10.19 – 30.10.19 | |
| 2 | Визначення основних вимог до системи | 31.10.19 – 01.11.19 | |
| 3 | Розроблення сценаріїв використання системи | 01.11.19 – 04.11.19 | |
| 4 | Вибір технологій для розробки | 05.11.19 – 06.11.19 | |
| 5 | Розроблення структурної схеми | 07.11.19 – 09.11.19 | |
| 6 | Реалізація бізнес-логіки | 11.11.19 – 12.11.19 | |
| 7 | Розроблення стартап-проекту | 15.11.19 – 18.11.19 | |
| 8 | Оформлення текстового матеріалу | 21.11.19 – 24.11.19 | |
| 9 | Оформлення графічного матеріалу | 27.11.19 – 30.11.19 | |
| 10 | Подача дисертації на перевірку | 03.12.19 | |

Студент _____ Рябчун А.В.
(підпис) (ініціали, прізвище)

Науковий керівник дисертації _____ Катін П.Ю.
(підпис) (ініціали, прізвище)

* Консультантом не може бути зазначено наукового керівника

# РЕФЕРАТ

Магістерська дисертація складається з 103 сторінок, 22 зображень, 48 таблиць, 16 посилань на використані джерела та 8 додатків.

Актуальність даної дисертації полягає в тому, що наразі існують складнощі взаємодії моделей машиного навчання та великих корпоративних систем, та на момент розробки системи не існує аналогів, що дозволяють автоматизувати цю взаємодію.

Мета дисертації – розробка автоматизованої системи управління процесом розпізнавання зображень, яка дозволить просто інтегрувати корпоративні системи та модулі комп’ютерного зору.

Об'єктом дослідження дисертації є автоматизація процесу взаємодії з моделями машиного навчання.

Предметом дослідження є моделі машиного навчання в області комп’ютерного зору та хмарні технології.

Під час вирішення завдань було застосовано загальні методи наукового пізнання, такі як: вимірювання, порівняння та спостереження, індукція та дедукція. За допомогою вказаних наукових методів було досліджено існуючі рішення, та визначено підходи до оптимізації системи.

Результати роботи можуть бути використаними у якості алгоритмів або окремих модулів, що дозволяють впровадити логіку автоматизації взаємодії з моделями машиного навчання в хмарному середовищі.

Ключові слова: архітектура, машинне навчання, сервіс, хмарне середовище, комп’ютерний зір.

# SUMMARY

The master's thesis consists of 103 pages, 22 figures, 48 tables, 16 references to the sources used and 8 appendices.

The relevance of this thesis is that now there are some complexities of interaction between machine learning models and large corporate systems, and at the time of development of the system there are no analogues that allow to automate this interaction.

The purpose of the thesis is to develop an automated control system for the image recognition process, which will allow easy integration of corporate systems and computer vision modules.

The object of the thesis is the automation of the process of interaction with machine learning models.

The subject of the research is machine learning models in the field of computer vision and cloud technologies.

General methods of scientific cognition, such as measurements, comparisons and observations, induction and deduction were used to solve the problems. Existing solutions were investigated with the help of these scientific methods, and approaches to optimization of development were determined.

The results of the work can be used as algorithms or individual modules that allow to implement the logic of automation of interaction with machine learning models in the cloud environment.

Keywords: architecture, machine learning, service, cloud environment, computer vision.

ЗМІСТ

# ПЕРЕЛІК СКОРОЧЕНЬ І ТЕРМІНІВ

API – application programming interface – інтерфейс програмної взаємодії.

CV – computer vision – комп’ютерний зір.

ML – machine learning – машинне навчання.

SDK – software development kit – набір інструментів для розробки.

HTTP – hypertext transfer protocol – протокол передачі гіпертексту.

GPU – графічний прискорювач.

Daemon – фоновий процес в ОС Linux.

CLI – утиліта командного рядка.

# ВСТУП

Питання необхідності систем автоматизації робочих процесів за допомогою методів машиного навчання давно перейшло у практичну площину. Все більше підприємств впроваджують собі подібні системи, відчуваючи на власному досвіді переваги нових в практичному сенсі технологій, які підвищують ефективність робочих процесів.

Інтерес до комп'ютерного зору з’явився одним з перших в області штучного інтелекту і машинного навчання. Перша архітектура штучної нейронної мережі - персептрон – була запропонована Френком Розенблаттом, який був натхненний структурою сітківки ока [1]. Його дослідження показують, що прогрес комп'ютерного зору в розпізнаванні образів символів визначається двома факторами: розвитком теорії, методів і розвитком апаратних засобів. Довгий час теоретичні та академічні дослідження випереджали практичне застосування систем комп'ютерного зору.

Дійсно масового застосування методи комп’ютерного зору набули лише менше 10 років тому, з досягненнями відповідного рівня продуктивності процесорів у персональних і мобільних комп’ютерах. Таким чином, в плані практичного застосування системи комп’ютерного зору пройшли ряд етапів: етап індивідуального рішення (як в частині апаратного забезпечення, так і алгоритмів) конкретних завдань; етап застосування в професійних областях (особливо в промисловості і оборонній сфері) з використанням спецпроцесорів, спеціалізовані системи формування зображень і алгоритми, призначені для роботи в умовах низької апріорної невизначеності, однак ці рішення допускали масштабування; і етап масового застосування.

Однією з ключових проблем, що здатні вирішити методи комп'ютерного зору, є інформаційний пошук та навчання. Багато задач доповненої реальності тісно пов'язані з інформаційним пошуком (так що деякі системи, такі як Google Goggles, складно віднести до якоїсь конкретної області), що робить суттєвий самостійний інтерес. Завдання пошуку зображень за змістом також різноманітні. Вони включають зіставлення зображень при пошуку зображень унікальних об'єктів, наприклад

архітектурних споруд, скульптур, картин і т. д., Виявлення і розпізнавання на зображеннях об'єктів класів різного ступеня спільності (автомобілів, тварин, меблів, осіб людей і т. д., а також їх підкласів), категоризація сцен (місто, ліс, гори, узбережжя і т. д.). Ці завдання можуть зустрічатися в різних додатках – для сортування зображень в домашніх цифрових фотоальбомах, для пошуку товарів за їхніми зображеннями в інтернет-магазинах, для вилучення зображень в геоінформаційних системах, для систем біометричної ідентифікації, для спеціалізованого пошуку зображень в соціальних мережах (наприклад, пошуку осіб людей, привабливих для користувача) і т. д., аж до пошуку зображень в Інтернеті.

Враховуючи велику актуальність методів комп’ютерного зору, програмні засоби, що вирішують вищевкащані проблеми дуже цінуються користувачами. Однак практичний досвід показує, що недостатньо просто створити модель машинного навчання для вирішення певної задачі. Необхідно забезпечити її ефективну та продуктивну роботу, та забезпечити можливість інтеграції з іншими системами, такими як, наприклад, системи аналітики. Хоча такі моделі можна використовувати і без інтеграцій, на практиці людські ресурси лише змінюють тип роботи – якщо спочатку люди виконували функції моделі, після впровадження люди виконують роботу по взаємодії з моделлю. На практиці це може займати навіть більше часу, ніж до впровадження моделі. Тому можливість інтеграції методів машиного навчання є основою для ефективного використання людських ресурсів.

# 1 АНАЛІЗ ІСНУЮЧИХ РІШЕНЬ

В звязку з великою популярністю методів машинного навчання, машиного зору та руху open source, існує велика кількість як інструментів для розробки моделей машиного навчання, так і значна кількість готових до використання моделей. Однак, такі інструменти та моделі практично не дають можливості використання у виробничих умовах. Точність, ефективність, економічність, продуктивність та насамперед можливість до зручної інтеграції є ключовими характеристиками, що необхідні для впровадження методів машиного навчання у інформаційні системи великих підприємств. Основні програмні продукти, що задовільняють перераховані вище вимоги, пропонуються як SaaS (система-як-сервіс) і переважно розроблені корпораціями, що надають послуги хмарних обчислень.

## 1.1 Google Vision API

Google є однією з найвідоміших компаній у галузі штучного інтелекту та машинного навчання. Він надає безліч хмарних обчислювальних сервісів в якості API для комп'ютерного зору. Vision API допомагає додатку зрозуміти, що знаходиться в зображенні, класифікуючи вміст за відомими категоріями і надаючи мітки (див. рисунок 1.1).

Він також здатний виявляти орієнтири-наприклад, будівлі, пам'ятники, природні структури або логотипи, виконуючи розпізнавання символів, яка підтримує широкий спектр мов. Знаходження обличчя дозволяє виявити обличчя та емоції людини. На жаль, підтримка розпізнавання обличчя відсутня. Крім того, ви можете використовувати API для пошуку схожих зображень в інтернеті і фільтрації явного або насильницького контенту.

Google також забезпечує відео-інтелект для виконання аналізу відео, класифікації та маркування. Це дозволяє здійснювати пошук по відео на основі витягнутих метаданих. Також можна виявити зміну сцени і відфільтрувати явний вміст. Всі ці функції доступні через REST API для легкої інтеграції.

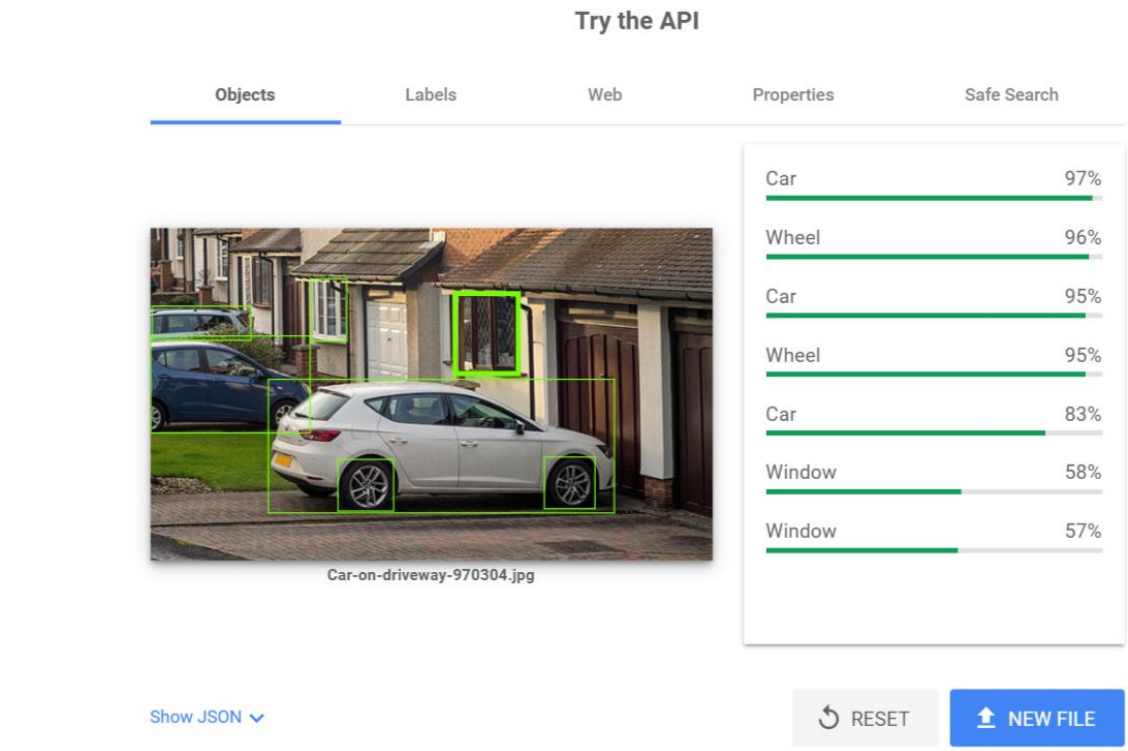


Рисунок 1.1 – Демонстрація роботи Google Vision API

## 1.2 Microsoft Azure Recognition Services

Microsoft Azure – це ще одна служба хмарних обчислень. Azure надає кілька служб комп'ютерного зору. Вони обгорнуті у вигляді різних API – API комп'ютерного зору для завдань загального призначення CV, API особи для виявлення і розпізнавання осіб, модератор контенту для фільтрації та деякі інші, які ще перебувають у стані попереднього перегляду.

API комп'ютерного зору дозволяє класифікувати вміст зображення, надаючи повний список тегів (див. рисунок 1.2) і намагаючись побудувати опис сцени на природній мові. Крім того, API здатний розпізнавати знаменитостей і пам'ятки.

Іншою особливістю є оптичне розпізнавання символів (OCR) друкованого тексту і в якості попереднього перегляду. OCR для рукописних текстів також доступна, але поки тільки для англійської мови.

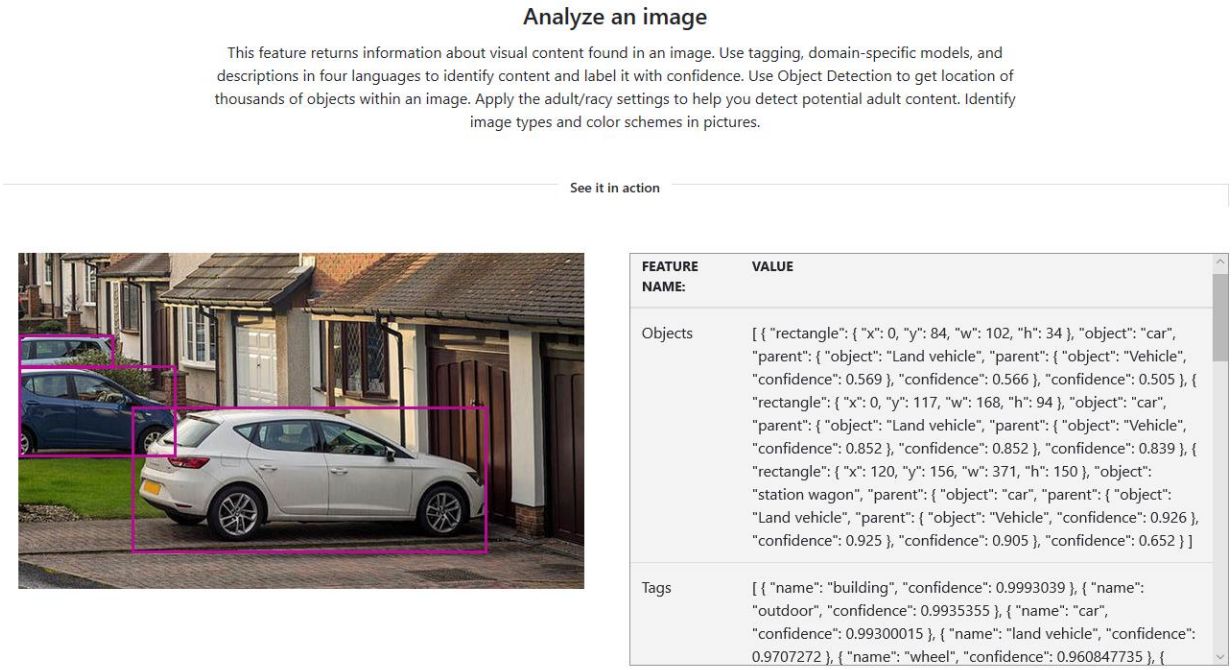


Рисунок 1.2 – Демонстрація роботи Azure Image Recognition

Face API використовується для виявлення облич на зображеннях і отримання границь і рис обличчя, таких як емоційний стан, стать, вік, волосся на обличчі, оцінка посмішки і лицьові орієнтири. Ще одна особливість – можливість внесення обличчя в базу даних для подальшого розпізнавання осіб.

Content Moderator може використовуватися для фільтрації відео і зображень. Небажаний вміст фільтрується за допомогою класифікаторів на основі машинного навчання і оптичного розпізнавання символів.

Video Indexer і Custom Vision Service поки доступні в якості попереднього перегляду. Індексатор відео використовується для вилучення інформації з відео. Він здатний аналізувати настрої, витягувати ключові слова і метадані, а також виявляти людей. Custom Vision Service дозволяє створювати точно налаштовані моделі комп'ютерного зору для конкретного випадку використання. Ця послуга здатна до інкрементального навчання – сторена модель буде поліпшуватися з плином часу з кожним наданим зображенням.

## 1.3 Amazon

Amazon Rekognition – це сервіс комп'ютерного зору, розроблений Amazon. В основі сервісу лежить механізм Deep Learning (глибоке навчання) і безшовна інтеграцію з іншими сервісами Amazon. Він надається в якості API для зображень і відео. Rekognition може виявляти, які об'єкти і люди знаходяться в сцені і що в цілому відбувається на зображенні. Він може працювати як фільтр контенту для контенту для дорослих. Крім того, сервіс може розпізнавати текст на зображенні.

Однією з можливостей Rekognition є здатність виявляти, розпізнавати і ідентифікувати людей. Він здатний точно ідентифікувати людину на фото і відео, використовуючи власний набір даних зображень особи. Сервіс також здатний аналізувати настрій, вік, наявність головних уборів, волосся на обличчі й інші особливості. Для відео можна відстежувати зміну цих показників з плином часу. Rekognition дозволяє відстежувати людей уздовж відео, навіть якщо вони відвертаються від камери або залишають сцену.

## 1.4 Clarifai

Clarifai – це ще одна порівняно молода компанія, що надає послуги комп'ютерного зору. Clarifai працює виключно з комп’ютерним зором, та має безліч різних функцій. Кожна конкретна задача вирішується за допомогою відповідної моделі. Деякі з моделей, тим не менш, знаходяться на етапі бета-тестування і постійно удосконалюються. Наприклад, існує модель, яка розпізнає обличчя на зображенні. Існує особлива модель для кожного з наступних параметрів: вік, стать, етнічна належність чи впізнаваність знаменитостей.

Загальна модель є найбільш універсальною. Вона здатна розпізнавати присутні об'єкти в зображенні (рисунок 1.3). Модель може бути використаний для будь-якого аналізу зображень. Також можлива побудова власної моделі для подальшого навчання на своїх зображеннях для досягнення найкращих результатів.

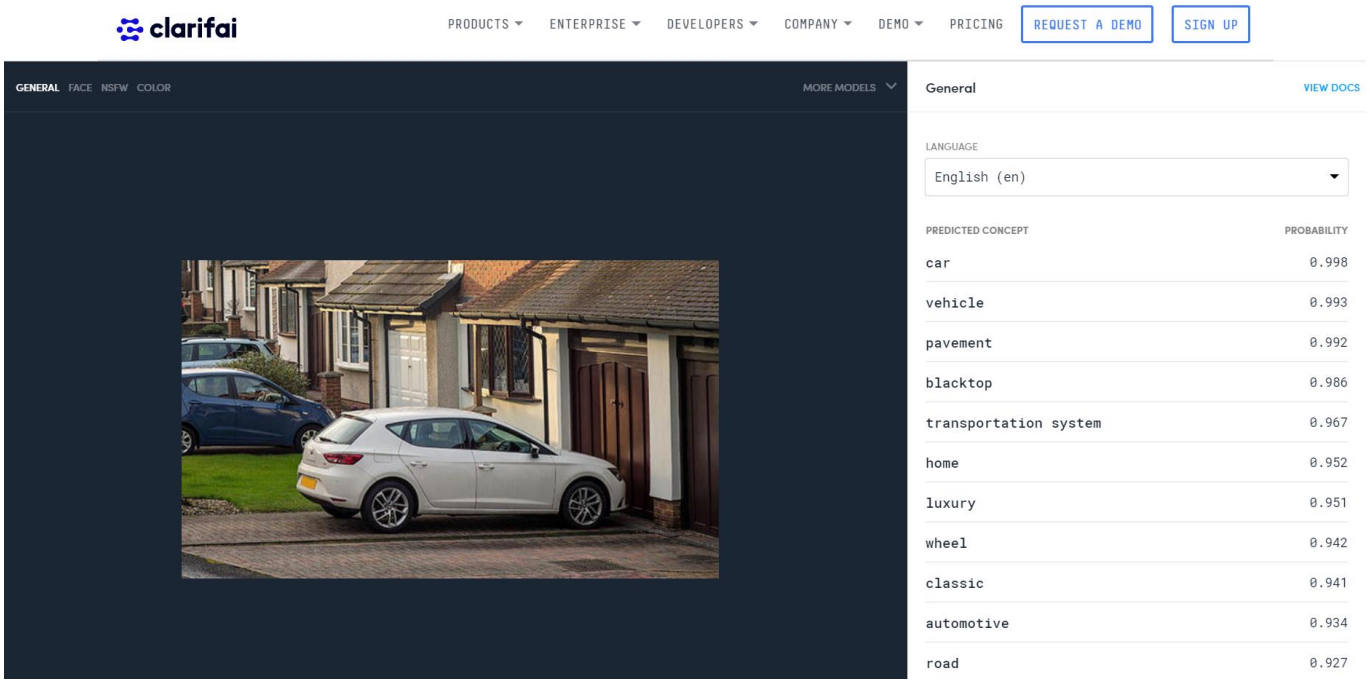


Рисунок 1.3 – Демонстрація роботи Clarifai

Clarifai також надає деякі вузькоспрямовані моделі, такі як ідентифікація візерунків та виявлення домінуючого кольору. Існує модель для ідентифікації одягу та аксесуарів, ще одна для ідентифікації продуктів харчування, а також для логотипу або фірмових найменувань. Існує дві моделі для вбудовування граней обличчя або загальних елементів. Вони засновані на розпізнаванні осіб і відповідних загальних моделях. Вбудовування дозволяє здійснювати низькорівневий контроль процесу машинного навчання. Існує також модель, для перевірки, чи містить зображення небезпечний контент.

## 1.5 IBM Watson Visual Recognition

IBM Watson Visual Recognition не має великої кількості готових моделей, але даний сервіс дозволяє створювати власні. За замовчуванням використовується загальна модель для розппізнавання об'єктів на зображенні або для визначення домінуючої палітри колорів. Інша модель використовується для виявлення обличчя (не розпізнавання), ще одна – для виявлення їжі. Також існує OCR модель, яка

знаходиться в приватній бета-версії на даний момент. API також дозволяє експортувати модель у форматі, сумісному з Core ML (Apple iOS).

## 1.6 Kairos

Напрямком роботи Kairos є розпізнавання осіб. За допомогою їх продуктів можна виявити особи на фотографіях або відео, ідентифікувати і валідувати людей. Kairos надає можливість визначити емоційний стан, вікову групу (наприклад, дитина, молодий, дорослий, старший), стать людини, риси особи, такі як очі, брови і т. д. Kairos доступний як Cloud API або SDK для інтеграції в автономному режимі.

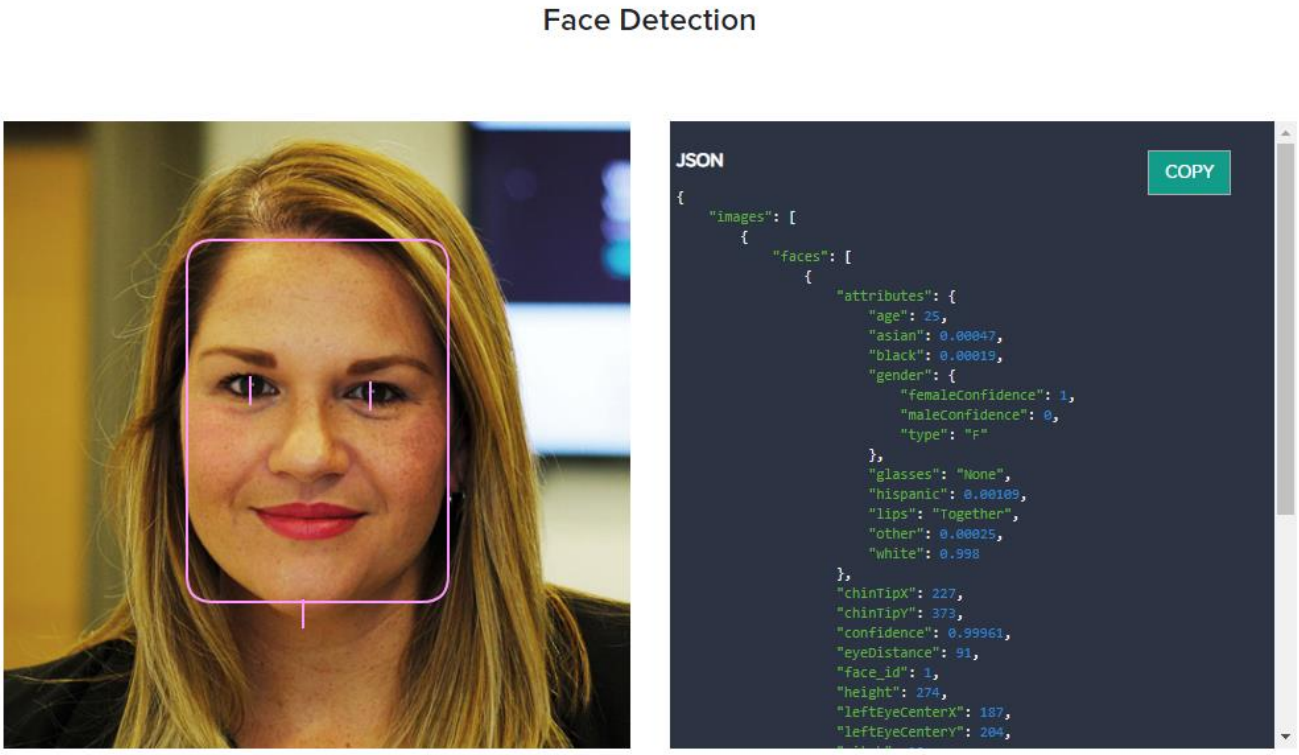


Рисунок 1.4 – Демонстрація роботи Kairos

## 1.7 Висновки до розділу

Існує велика кількість хмарних сервісів по вирішенню типових проблем за допомогою методів машиного навчання, а деякі з них (Azure Custom Vision, IBM Visual Recognition) навіть підтримують створення своїх моделей для вирішення

нестандартних завдань. Але практика показує, що можливість точного налаштування моделі в хмарних сервісах не дає бажаний рівень точності під час вирішення проблеми розпізнавання та класифікації однотипних об’єктів (наприклад, марки та моделі автомобілів, замість стандартного «автомобіль»). Інша проблема – відносно велика вартість роботи сервісу.

Таку точність можуть дати лише моделі, що створені під повним контролем фахівцями у сфері машиного навчання за допомогою спеціального інструментарію. Але за замовчуванням з такими моделями практично неможливо працювати користувачу. Постає потреба у програмному засобі, що може об’єднувати ці два підходів та мати ключові переваги – точність та простоту взаємодії.

# 2 ФОРМУВАННЯ ВИМОГ ДО СИСТЕМИ

Аналіз вимог, який також називають розробкою вимог, – це процес визначення очікувань користувачів щодо нового продукту. Ці очікування, що називаються вимогами, повинні бути кількісно оцінені та деталізовані. У розробці програмного забезпечення такі вимоги часто називають функціональними специфікаціями. Аналіз вимог є важливим аспектом управління проектами.

Саме цей етап розробки програмного продукту передбачає часте спілкування з користувачами системи для визначення конкретних очікувань на особливості системи, вирішення конфліктів чи неоднозначностей вимог, які вимагають різні користувачі чи групи користувачів, та документації всіх аспектів процесу розробки проекту від початку до кінця. Зусилля потрібно зосереджувати на забезпечення відповідності кінцевої системи чи продукту потребам клієнта, а не на спробу формувати очікування користувачів, щоб відповідати вимогам.

Вимоги поділяються на функціональні та нефункціональні. Аналіз вимог можна розділити на три етапи:

- збір вимог – включає в себе аналіз прикладної області, огляд існуючих рішень, спілкування з клієнтами та майбутніми користувачами;
- аналіз вимог – на цьому етапі визначаються основні особливості та характеристики системи: цілісність, суперечливість вимог між собою, відносини між ними та їх повнота;
- документування вимог - вимоги повинні бути формально задокументовані, якщо це необхідно.

Усі вимоги розділяються на функціональні та нефункціональні.

## 2.1 Функціональні вимоги

Функціональними вимогами називаються ті вимоги, що висвітлюють необхідний функціонал програмного продукту. В загальному випадку функціонал визначається поведінкою системи та вхідними і вихідними параметрами. Це може

бути взаємодія з даними, бізнес-процес, взаємодія з користувачем, або будь-який інший специфічний функціонал системи.

Основними функціональними вимогами є:

- система повинна приймати на вхід зображення разом з напівструктурованими метаданими для подальшої обробки і аналітики;
- система повинна зберігати вхідні зображення у спеціальному сховищі;
- система повинна оброблювати напівструктуровані метадані та зберігати їх у форматі, що підтримує базові операції з даними (фільтрація, сортування, групування, агрегація);
- система повинна мати можливість моніторингу свого стану по ключовим метрикам роботи;
- система повинна зберігати результати роботи механізму розпізнавання зображень у сховище;
- система повинна унеможливлювати випадки, коли механізм розпізнавання зображень працює без корисного навантаження – система приймає рішення по необхідності запуску та зупинки цього механізму;
- система повинна мати можливість інтеграції з зовнішніми сервісами з допомогою сучасних протоколів обміну даними;
- система повинна бути доступною в мережі Інтернет;
- система повинна мати можливість роботи з різними типами механізмів розпізнавання зображень.

## 2.2 Нефункціональні вимоги

Нефункціональні вимоги визначають якісні характеристики програмної системи. Вони являють собою набір стандартів, які використовуються для оцінки конкретної функції системи.

Нефункціональна вимога є важливою для забезпечення зручності використання та ефективності всієї програмної системи. Невиконання нефункціональних вимог

може призвести до створення систем, які не задовольняють потреби користувача. Опис нефункціональних вимог не менш важливий, ніж опис функціональних вимог.

Основними нефункціональними вимогами є:

- допустима швидкість розпізнавання одного зображення – не більше 5 секунд;
- допустима кількість вхідних зображень для обробки – не більше 17 тисяч одиниць на добу;
- точки інтеграції мають захищеність від неавторизованого втручання;
- точки інтеграції мають мати не менше 99.9% SLA (доступності);
- система має бути економічно вигідною у порівнянні з конкурентами;
- розробка системи має відбуватися на основі хмарної платформи Microsoft Azure;
- необхідно забезпечити можливість доробки і перерозгортання окремих компонентів без зупинки усієї системи;
- необхідно забезпечити можливість до швидкого розгортання (до 4 годин) усієї системи у новому середовищі Microsoft Azure;
- за замовчування повинна існувати інтеграція для імпорту даних через HTTP API;
- необхідно забезпечити журналювання наступних подій: зображення завантажене, зображення успішно опрацьоване, при опрацюванні зображення виникла помилка;

## 2.3 Висновки до розділу

В даному розділі були розроблені загальні функціональні та нефункціональні вимоги до системи. Вимоги до системи – це ключові характеристики, що визначають якість системи і її відповідність вимогам користувача. У всіх типах вимог пропуск вимоги може поставити під загрозу цілісність рішення. Функціональні і нефункціональні вимоги тісно пов'язані безліччю взаємозв'язків.

Кожне рішення забезпечує ефективність за рахунок вичерпного переліку вимог, зібраних в процесі запуску і впровадження. Вимоги можна розділити на дві категорії:

основні вимоги і базові вимоги. Основні вимоги випливають з їх аналогії з "функціональними вимогами", які безпосередньо пов'язані з рішенням. Однак так звані базові вимоги не можуть бути безпосередньо пов'язані з рішенням, вони необхідні для створення стійкого середовища, яке підтримує основні функціональні вимоги. Таким чином, ці "нефункціональні вимоги" створюють інфраструктуру для підтримки системних рішень.

При формуванні вимог до системи був обраний загальний шлях розвитку, який базується на таких принципах:

- гнучкість системи керуванням процесом розпізнавання зображення;
- економічна ефективність готового рішення у порівнянні з існуючими рішеннями;
- широкі можливості інтеграції системи з іншими системами.

# 3 СЦЕНАРІЇ ВИКОРИСТАННЯ СИСТЕМИ

Сценарії використання можуть описувати поведінку системи при взаємодії з кимось або чимось у зовнішньому середовищі і описувати вимоги таким чином, щоб користувачі та зацікавлені сторони могли їх зрозуміти.

Підходи на основі прецедентів дозволяють розробляти вимоги до поведінки системи, з якою взаємодіють користувачі, наприклад функціональні вимоги або вимоги користувачів.

Прецедент називає систему "чорним ящиком" і взаємодіє з нею з точки зору зовнішнього спостерігача. Цей підхід до побудови архітектури фокусується на тому, як повинна бути виконана система, і робить припущення про те, як реалізувати розглянуті функції.

Варіанти використання записуються на рівні абстракції та функціональності системи, визначаючи функції або послуги, які система надає користувачеві, дозволяючи суб'єктам взаємодіяти з системою.

Опис проводиться на рівні системної функції і включає в себе функції і послуги, які система надає користувачу, а також дії, які суб'єкт може виконувати при взаємодії з системою.

## 3.1 Ролі користувачів у системі

Користувачі системи діляться на 3 типи: користувач, що імпортує дані у систему (імпортер); користувач, що експортує дані з системи (експорер); користувач-адміністратор системи. У загальному випадку, користувачами є зовнішні системи, що інтегруються з розроблюваною системою, але за допомогою спеціального програмного забезпечення ці ролі можуть виконувати люди.

Ролі користувачів:

- імпортер — завантаження даних у систему (передача зображення та метаданих);
- експортер — вивантаження оброблених даних для подальшої їх обробки;

- адміністратор — отримання внутрішньої статистики роботи системи, налаштування режиму роботи підсистеми розпізнавання.

## 3.2 Опис сценаріїв використання

Детальний опис сценаріїв використання представлений на таблицях 3.1 – 3.7. Для опису дій використовуються наступні позначення: Для опису дій користувача використовується літера К та номер дії; для дій системи – літера С та номер дії. Кожному сценарію використання надано унікальний ідентифікатор.

Таблиця 3.1 – Опис сценарію використання «Імпорт даних у систему»

| | |
|---|---|
| Назва сценарію | Імпорт даних у систему |
| Ідентифікатор | UC1 |
| Спричиняє події | Підготовка зображень до розпізнавання, збереження метаданих у сховищі. |
| Короткий опис | Для початку роботи з системою необхідно внести файли-зображення та метадані у систему для подальшої обробки. |
| Частота | Можливе постійне використання |
| Актори | Імпортер |
| Передумови | Зображення має бути у форматі, що підтримується системою. |
| Вихідні умови | Зображення збережене у сховищі, вхідні метадані збережені у сховищі разом з посиланням на зображення. Зображення поміщене у чергу до механізму розпізнавання об’єктів. |
| Опис дій | К1 – користувач формує вхідне повідомлення у правильному форматі та передає його до системи.<br>Повідомлення має містити один/кілька зображень з відповідними напівструктурованими метаданими.<br>С1 – система приймає вхідне повідомлення.<br>С2 – система валідує вхідне повідомлення. |

Продовженні таблиці 3.1

| | |
|---|---|
| Назва сценарію | Імпорт даних у систему |
| Опис дій (продовження) | С3 – зображення зберігається в об’єктному сховищі.<br>С4 – система створює запис про зображення в табличному сховищі, що містить посилання на зображення, сервісні дані та вхідні метадані.<br>С5 – зображення поміщається у чергу до механізму розпізнавання системи. |
| Очікувані умови | Створення запису про зображення, збереження зображення до сховища, запис повідомлення про необхідність розпізнавання об’єктів на зображенні до коректні дані; отримання повідомлення про помилку, якщо користувач надіслав некоректні дані. |

Таблиця 3.2 – Опис сценарію використання «Запуск процесу розпізнавання об’єктів за розкладом»

| | |
|---|---|
| Назва сценарію | Запуск процесу розпізнавання об’єктів за розкладом |
| Ідентифікатор | UC2 |
| Спричиняє подію | Запуск процесу розпізнавання об’єктів |
| Короткий опис | Запуск системою за розкладом процесу розпізнавання об’єктів на зображеннях, які стоять в черзі на розпізнавання |
| Актори | Система |
| Частота | Задається об’ємом даних, які потрібно оброблювати |
| Передумови | Черга на розпізнавання містить повідомлення |
| Вихідні умови | Результати розпізнавання збережені у табличному сховищі та підв’язані до відповідних записів, що створилися під час завантаження даних у систему. Результат зберігається у форматі «як є», тобто система зберігає відповідь розпізнавання |

Продовження таблиці 3.2

| | |
|---|---|
| Назва сценарію | Запуск процесу розпізнавання об’єктів за розкладом |
| Опис дій | С1 – система періодично за розкладом спричинює запуск процесу розпізнавання.<br>С2 – система перевіряє наявність зображень у черзі на розпізнавання.<br>С3 – система вмикає та підготовлює механізм розпізнавання об’єктів.<br>С4 – система почергово передає зображення з черги на механізм розпізнавання, отримуючи результат.<br>С5 – отриманий результат зберігається у табличному сховищі, підв’язуючись до запису, що був створений на етапі імпорту.<br>С6 – після заверення обробки всіх зображень з черги, система вимикає механізм розпізнавання. |
| Очікувані умови | Збереження даних про розпізнані об’єкти в сховище. Передача результатів розпізнавання у точку інтеграції експорту. Механізм розпізнавання працює лише за умови наявності корисних даних для обробки. |

Таблиця 3.3 – Опис сценарію використання «Запуск процесу розпізнавання об’єктів за вимогою»

| | |
|---|---|
| Назва сценарію | Запуск процесу розпізнавання об’єктів за вимогою |
| Ідентифікатор | UC3 |
| Спричиняє подію | Запуск процесу розпізнавання об’єктів |
| Короткий опис | Адміністратор вручну запускає процес розпізнавання об’єктів на зображеннях, які стоять в черзі на розпізнавання. |
| Актори | Адміністратор |
| Передумови | Черга на розпізнавання містить повідомлення. |

Продовження таблиці 3.3

| | |
|---|---|
| Назва сценарію | Запуск процесу розпізнавання об’єктів за вимогою |
| Вихідні умови | Результати розпізнавання збережені у табличному сховищі та підв’язані до відповідних записів, що створилися під час завантаження даних у систему. За наяності додаткових інтеграцій, дані передаються до зовнішніх систем |
| Опис дій | К1 – користувач вручну спричинює запуск процесу розпізнавання.<br>С1 – система перевіряє наявність зображень у черзі на розпізнавання.<br>С2 – система вмикає та підготовлює механізм розпізнавання об’єктів.<br>С3 – система почергово передає зображення з черги на механізм розпізнавання, отримуючи результат.<br>С4 – отриманий результат зберігається у табличному сховищі, підв’язуючись до запису, що був створений на етапі імпорту.<br>С5 – після заверення обробки всіх зображень з черги, система вимикає механізм розпізнавання. |
| Очікувані умови | Збереження даних про розпізнані об’єкти в сховище |

Таблиця 3.4 – Опис сценарію використання «Експорт даних з системи за вимогою»

| | |
|---|---|
| Назва сценарію | Експорт даних з системи за вимогою |
| Ідентифікатор | UC4 |
| Короткий опис | На цьому етапу відбувається передача результатів роботи системи у зовнішнє середовище. |
| Актори | Експортер |
| Передумови | Дані існують у системі |

Продовження таблиці 3.4

| | |
|---|---|
| Назва сценарію | Експорт даних з системи за вимогою |
| Вихідні умови | Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів. |
| Опис дій | К1 – робить запит до сховища результатів механізму розпізнавання<br>С1 – система повертає дані користувачу |
| Очікувані умови | Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів. |

Таблиця 3.5 – Опис сценарію використання «Постійний експорт даних з системи»

| | |
|---|---|
| Назва сценарію | Постійний експорт даних з системи |
| Ідентифікатор | UC5 |
| Короткий опис | Відбувається постійна передача результатів роботи системи у зовнішнє середовище за допомогою інтеграційних модулей, що працюють завдяки передбаченим точкам інтеграції. |
| Актори | Експортер |
| Спричинюється подіями | Поява нових результатів розпізнавання об’єктів |
| Передумови | Дані існують у системі; є під’єднаний інтеграційний модуль експорту до відповідної точки інтеграції |
| Вихідні умови | Нові результати роботи механізму розпізнавання зображень постійно проходять точку інтеграції. Дані експортовані у необхідний формат та доступні для роботи в інших системах. |
| Опис дій | К1 – користувач має налаштовану точку інтеграції<br>С1 – система перехоплює результати розпізнавання.<br>С2 – окрім стандартного процесу збереження результатів у сховище, система надсилає дані в інші системи за допомогою |

Продовження таблиці 3.5

| | |
|---|---|
| Назва сценарію | Постійний експорт даних з системи |
| Опис дій (продовження) | допомогою модулю інтеграції. |
| Очікувані умови | Дані експортовані у необхідний формат та доступні для роботи в інших системах/для інших процесів. Нові результати роботи механізму розпізнавання зображень постійно проходять точку інтеграції |

Таблиця 3.6 – Опис сценарію використання «Отримання аналітики по використанню ресурсів та об’ємах даних»

| | |
|---|---|
| Назва сценарію | Отримання аналітики по використанню ресурсів та об’ємах даних |
| Ідентифікатор | UC6 |
| Короткий опис | В цьому сценарії використання користувач має змогу отримати аналітику по використанню системи. |
| Актори | Адміністратор |
| Передумови | - |
| Вихідні умови | Користувач отримує історичні метрики та/або метрики реального часу |
| Опис дій | К1 – користувач формує запит по типам та критеріям аналітики.<br>С1 – система повертає метрики у зручному для подальшої роботи форматі. |
| Очікувані умови | Користувач отримує історичні метрики та/або метрики реального часу:<br>- кількість оброблених зображень за період часу;<br>- актуальний стан механізму розпізнавання; |

Продовження таблиці 3.6

| | |
|---|---|
| Назва сценарію | Отримання аналітики по використанню ресурсів та об’ємах даних |
| Очікувані умови (продовження) | - середня швидість розпізнавання об'єктів на одному зображенні;<br>- використаний об’єм сховища зображень;<br>- помилки, які відбулися під час роботи системи. |

Таблиця 3.7 – Опис сценарію використання «Керування регламентом автоматичного запуску процесу розпізнавання»

| | |
|---|---|
| Назва сценарію | Керування регламентом автоматичного запуску процесу розпізнавання |
| Ідентифікатор | UC7 |
| Короткий опис | В цьому сценарії використання користувач має змогу налаштувати регламент (розклад) автоматичного запуску процесу розпізнавання зображень |
| Актори | Адміністратор |
| Передумови | - |
| Вихідні умови | Система сконфігурована для автоматичного запуску по розкладу |
| Опис дій | К1 – користувач задає розклад автоматичного запуску процесу розпізнавання зображень.<br>С1 – зберігає заданий розклад і в подальшому запускає механізм розпізнавання згідно до нього. |
| Очікувані умови | Система сконфігурована для автоматичного запуску по регламенту, що задав користувач. |

## 3.3 Висновки до розділу

Описані інтерактивні сценарії повністю відповідають вимогам програмного продукту. Аналіз вимог до системи показує, що з самою системою мають взаємодіяти три ролі користувачів, причому під час нормальної роботи системи людина буде виконувати лише одну роль – адміністратора. Взаємодію з функціоналом, що буде розроблений для інших ролей, будуть виконувати інші системи після інтеграції. Саме цей фактор спрощує розробку у плані, що зазвичай дії системи є детермінованими у більшій мірі, ніж дії людини, і це зменшує пріоритет на розробку валідування вхідних даних.

На основі цих сценаріїв можна розробити функціональні тести, щоб довести, що програмне забезпечення правильно виконує описані функції. Згідно до описаних сценаріїв була побудована діаграма прецедентів (див. додаток А).

# 4 СТРУКТУРНА СХЕМА СИСТЕМИ

Структурна схема відображає склад і взаємодію різних частин системи. Структурна схема системи, як правило, вказує на наявність підсистем та інших структурних компонентів, що дозволяє управляти взаємодією багаторівневих ієрархічних програмних модулей.

Розробка блок-схем є одним з найважливіших етапів проектування програмного забезпечення, сукупність усіх компонентів, їх взаємозв'язків, існуючих блок-схем об'єктів між ними, є картою, що полегшує модифікацію і розширення компонентів системи. Повна розроблена структурна схема відображена в додатку Б.

Система спроектована згідно до концепції багатошарової архітектури. Багаторівнена система – це система, що розроблена та розподілена між кількома рівнями. Суть полягає у тому, що прикладні рівні (шари) логічно розділяються по рівням абстракцій та типу функціонування. Кількість рівнів залежить від вимог бізнесу та типу проекту. Будь-який система, яке залежить від проміжного програмного забезпечення (middleware) або використовує його, називається багаторівневою.

Спроектована система складається з наступних шарів: рівень взаємодії з даними, рівень збереження даних, рівень управління механізмом розпізнавання зображень, рівень розпізнавання зображень. Кожен шар взаємодіє лише з сусіднім шаром, причому потік виконання основного процесу починається з найвищого рівня – рівня взаємодії з даними, поступово переходить до найнижчого – рівня розпізнавання, та повертається до найвищого (див. рис. 4.1). Використання такого підходу дозволяє спростити розуміння послідовності дій у системі та спрощує подальший розвиток продукту. Так як дані рухаються в одному порядку, зручно зрозуміти, на якому етапі обробки даних знаходиться певний об’єкт, і визначити, які дії будуть виконуватися надалі.


Рисунок 4.1 – Рівні архітектури системи

Рівень взаємодії з даними включає в себе точки інтеграції для процесів імпорту та експорту даних, так як цього потребують функціональні вимоги, та модулі збереження даних. Оскільки всі вхідні (зображення, метадані) та вихідні (результати) дані повинні зберігатися у сховищі, а окрім цього вхідні зображення повинні поміщатися у спеціальну чергу, модулі цього рівня напряму взаємодіють з рівнем збереження даних, що зображено на рисунку 4.2.


Рисунок 4.2 – Підсистема взаємодії з даними

Рівень збереження даних включає в себе сховища даних – об’єктні для зображень, табличні для метаданих та результатів розпізнавання, та черга для зображень, що поступають на механізм розпізнавання. В сховища дані поступають з рівня взаємодії з даними (насамперед з точок інтеграцій), а за допомогою черги зображення передаються далі в підсистему керування розпізнаванням. Схема компонентів та взаємодії рівня збереження даних зображена на рис. 4.3.

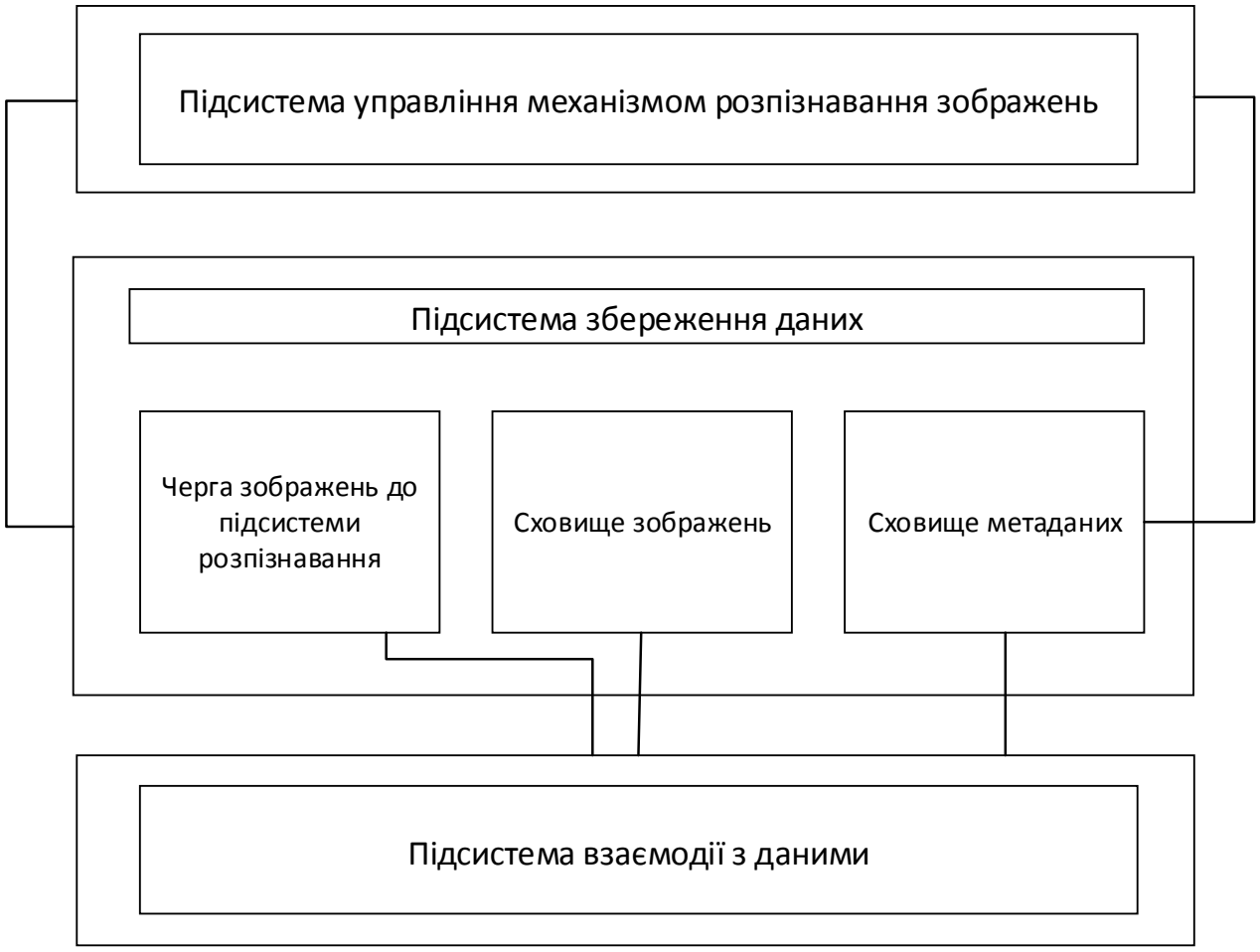


Рисунок 4.3 – Рівень збереження даних

Рівень управління механізмом розпізнавання зображень призначений для отримання поставлених в чергу зображень, запуску та підготовці механізму розпізнавання зображень, взаємодія з цим механізмом та подальшої його зупинки. Оскільки запуск механізму може бути автоматичним і мати свій регламент, існує модуль автоматичного запуску. Окрім цього, модуль взаємодії з механізмом розпізнавання генерує метрики, які необхідно збирати спеціальни модулем збору

метрик та аналітичних даних. Струтурна схема цього та суміжних рівнів зображена на рисунку 4.4.


Рисунок 4.4 – Структурна схема рівня управління механізмом розпізнавання зображень

Основною ціллю системи управління є підсистема розпізнавання зображень, яка складається з самого модулю розпізнавання та модулю управління середовищем розпізнавання. Наприклад модуль розпізнавання являє собою ML модель, а середовище його виконання – віртуальна машина або контейнер. Струтурна схема рівня зображена на рисунку 4.5.

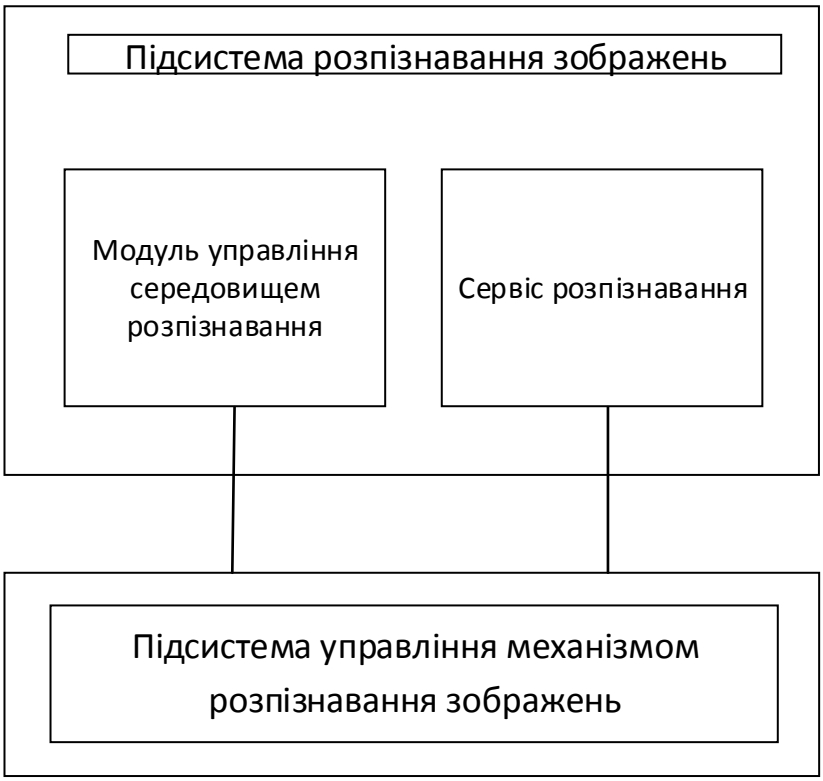


Рисунок 4.5 – Структурна схема шару розпізнавання зображень

Побудова багаторівневої архітектури спрощує супровід і масштабування готової системи та процес розробки системи на початкових етапах впровадження. Кожен рівень можливо замінити повністю, достатньо лише дотримуватися інтерфейсів, які визначаються на етапі створення архітектури проекту. Схожим чином можливо додавати нові функціональні рівні між уже існуючими.

Повна структурна діаграма системи зображена в додатку Б.

# 5 ВИБІР ТА ОБГРУНТУВАННЯ ЕЛЕМЕНТІВ ТА ТЕХНОЛОГІЙ

Вибір технологій є надзвичайно важливим етапом проектування системи. Хоча система і поділена на рівні, що дає змогу використовувати різні техногії за умовою дотримання програмних інтерфейсів взаємодії, на практиці зручніше використовувати однаковий стек технологій у всьому проекті і відхилятися лише за наявності явних переваг одної технології над стеком технологій уього проекту.

## 5.1 Вибір платформи розробки системи

Виходячи з вимог до системи та архітектури, основною платформою розробки була обрана платформа .NET Core, завдяки таким її перевагам:

- .NET Core являє собою відкриту платформи для розробки програмних засобів широкого призначення (див. рис. 5.1). Ключовою особливістю платформи є її незалежність щодо архітектури процесора та операційної системи – застосунки, розроблені за допомогою цієї платформи, можуть виконуватися на робочих станціях користувачів, на наземних серверах, на пристроях інтернету речей або в хмарному середовищі [2];

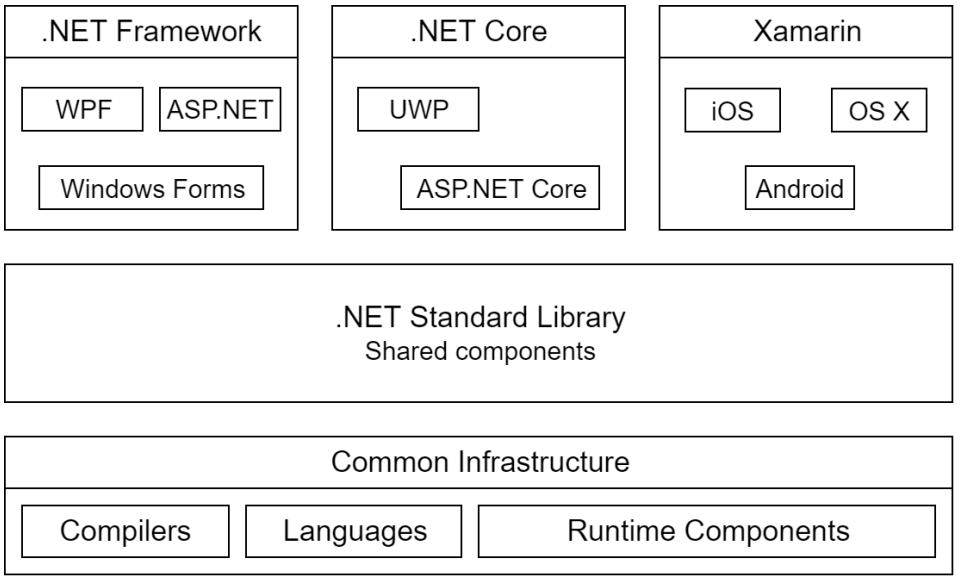


Рисунок 5.1 – Інфраструктура платформи .NET

- основною (але не єдиною) мовою програмування на цій платформі є C# - об’єктно-спрямована, статично-типізована, гнучка мова програмування високого рівня, що має обширні можливості лаконічного опису різнопланових програм;
- незважаючи на напівдинамічну середовище виконання (сирцевий код компілюється не в машинні коди, а в так зване «проміжне представлення), продуктивність застосунків, розроблених на цій платформі досягає продуктивності нативних рішень (див. таблицю 5.1);

Таблиця 5.1 – Порівняння продуктивності веб-фреймворків для задачі видавання неформатованого тексту (дані сайту www.techempower.com [3])

| Позиція | Фреймворк | Запитів у секунду | Порівняльний відсоток | Платформа |
|---|---|---|---|---|
| 1 | hyper | 7,006,181 | 100.0% | Rust |
| 2 | tokio-minihttp | 7,006,181 | 100.0% | Rust |
| 3 | ulib-plaintext_fit | 7,004,608 | 100.0% | C++ |
| 4 | actix | 7,000,911 | 99.9% | Rust |
| 5 | ulib | 6,998,172 | 99.9% | C++ |
| 6 | libreactor | 6,997,422 | 99.9% | C |
| 7 | actix-raw | 6,996,104 | 99.8% | Rust |
| 8 | atreugo-prefork | 6,995,436 | 99.8% | Go |
| 9 | firenio-http-lite | 6,994,344 | 99.8% | Java |
| 10 | aspcore | 6,993,704 | 99.8% | .NET |
| 11 | aspcore-rhtx | 6,990,400 | 99.8% | .NET |
| 12 | wizzardo-http | 6,987,612 | 99.7% | Java |
| 13 | fasthttp | 6,983,911 | 99.7% | Go |
| 14 | may-minihttp | 6,977,134 | 99.6% | Rust |
| 15 | fasthttp-prefork | 6,969,608 | 99.5% | Go |

- завдяки тому, що платформа розробки .NET та платформа хмарних обчислень Azure розвиваються з участю компанії Microsoft, інтеграція Azure з додатками на

платформі .NET є більш зручною, ніж з будь-якою іншою технологією – саме під платформу .NET існують SDK для практично усіх хмарних сервісів [4]. А фактор наявності SDK зменшує кількість ресурсів на інтеграцію, отже і на розробку усього проекту.

Таким чином використання платформи .NET є найбільш оптимальним для розробки основної частини системи.

## 5.2 Вибір хмарних сервісів для побудови системи

Хмарна платформа Microsoft Azure надає широкий спектр послуг (див. рисунок 5.1) для розробки програмних систем. Хмарні обчислення – технологія розподіленої обробки даних, у якій обчислювальні ресурси і потужності надаються користувачу у вигляді Інтернет-послуги. Перевагою хмарних рішень є доступність, відносно низька вартість, гнучкість і надійність.

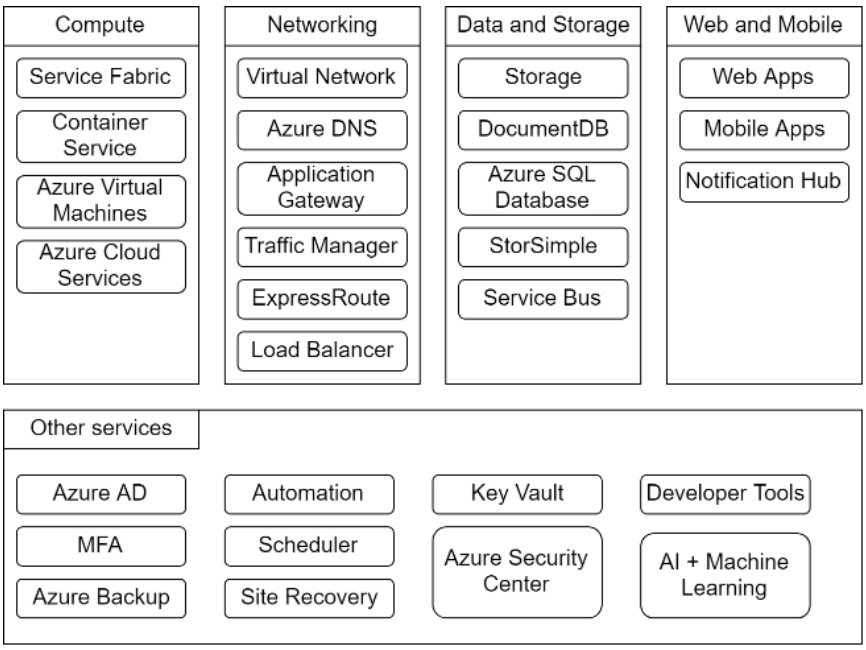


Рисунок 5.1 – Хмарні сервіси Microsoft Azure

### 5.2.1 Середовище виконання логіки системи

Azure надає можливість розміщення веб-застосунків декількома підходами. Характеристики, переваги та недоліки для проекту, визначені та описані в таблиці 5.2.

Таблиця 5.2 – Порівняння підходів розміщення веб-застосунків

| Назва | Характеристика | Переваги | Недоліки |
|---|---|---|---|
| Virtual Machine | Забезпечує повну емуляцію фізичної машини та операційної системи | Можливість розгортання усієї системи в одній машині. | Висока вартість. Складність розгортання компонентів. |
| App Service | Віртуальний сервер, що призначений переважно для розміщення веб-застосунків | Орієнтованість на веб-застосунки. | Складність розгортання окремих компонентів. |
| Cloud Functions | Невеликі блоки логіки, що обмінюються даними за допомогою інших систем | Простота моніторингу роботи додатку. Простота розгортання окремих компонентів. Вбудовані інтеграції з сервісами Azure | - |
| Container Service | Середовище виконання додатку, що поміщений у контейнер | Простота розгортання окремих компонентів. Можливість роботи будь-якої технології, яка підтримує Linux | Контейнеризація є надмірною |

Продовження таблиці 5.2

| Назва | Характеристика | Переваги | Недоліки |
|---|---|---|---|
| Service Fabric | Платформа для роботи великих корпоративних систем, що розділені на дрібні сервіси | Багатий функціонал по управлінню та розміщенню окремих сервісів | Контейнеризація є надмірною. Висока вартість |

В якості середовища виконання логіки системи була обрана технологія Azure Functions. Azure Functions – кероване подіями середовище обчислення, що надає змогу запускати певну логіку після певних подій, які можуть траплятися як у хмарному середовищі, так і в наземній інфраструктурі. Azure Functions дозволяють виконувати дії, підключаючись до джерел даних або рішень обміну повідомленнями, що спрощує обробку і реагування на події. Джерелом подій можуть також слугувати HTTP запити, шо дозволяє розробляти Web API – метод взаємодії програмних продуктів, який є доступним широкому колу додатків.

Azure Functions мають декілька режимів роботи. Першим режимом є так званий Consumption plan. При використанні Consumption плану екземляри функцій динамічно додаються і видаляються в залежності від кількості вхідних подій. Цей безсерверний (serverless) план автоматично масштабується і не має вартості коли не використовується. Але є суттєвий недолік, що унеможливлює використання цього плану для нашої системи – при використанні Consumption плану функції не можуть запускатися за розкладом, що конфліктує з функціональними вимогами до системи.

Тому був обраний альтернативний варіант – App Service Plan – під час якого функції працюють в рамках окремого віртуального серверу. Azure App Service Plan являє собою віртуальний сервер, що призначений переважно для розміщення веб-застосунків, що дозволяє абстрагуватися від керування інфраструктурою – фізичними серверами та віртуальними машинами. Даний тип ресурсу має змогу до швидкого і прозорого масштабування, як горизонтального (кількість таких віртуальних серверів), так і вертикального (зміна потужності обраховувальних ресурсів: швидкість процессору, кількість оперативної пам’яті, пропускної здатності мережі).

Саме цей ресурс має свою вартість – на одному App Service Plan можливо запустити декілька веб-сайтів (App Service) та/або Functions.

### 5.2.2 Сервіс для взаємодії компонентів

Хоча функції і підтримують синхронну модель взаємодії (за допомогої HTTP API), основний підхід до використання – це асинхронне реагування на події. Оскільки типів подій у системі декілька, зручно використовувати патерн (та відповідне програмне забезпечення) під назвою «Брокер повідомлень». Він використовується для валідації, перетворення та маршрутизації повідомлень. Брокер опосередковує комунікацію між додатками, зводячи до мінімуму взаємну обізнаність, яку додатки повинні мати один про одного, щоб мати можливість обмінюватися повідомленнями, ефективно реалізуючи розв'язку. В якості сервісу брокеру повідомлень була обрана система Azure Service Bus з наступних причин:

- власний сервіс Azure;
- прозора інтеграція з Azure Functions;
- простота розгортання і адміністрування екземпляром системи.

Для побудови системи будуть використані наступні типи каналів обміну повідомленнями (рис 5.2):

- queue – представляє собою впорядкований (FIFO) канал передачі повідомлень між двома точками;
- topic – абстрагує декілька черг в одну на стороні відправки повідомлень, причому кожна черга може мати свого споживача, використовується для сценарія publisher/consumer.

Саме механізм Topic/Subscription забезпечить простоту налаштування точок інтеграцій – дані будуть надходити на topic, а кожна активна інтеграція матиме відповідний subscription. Це дозволить під’єднувати нові інтеграції без зупинки усієї системи.

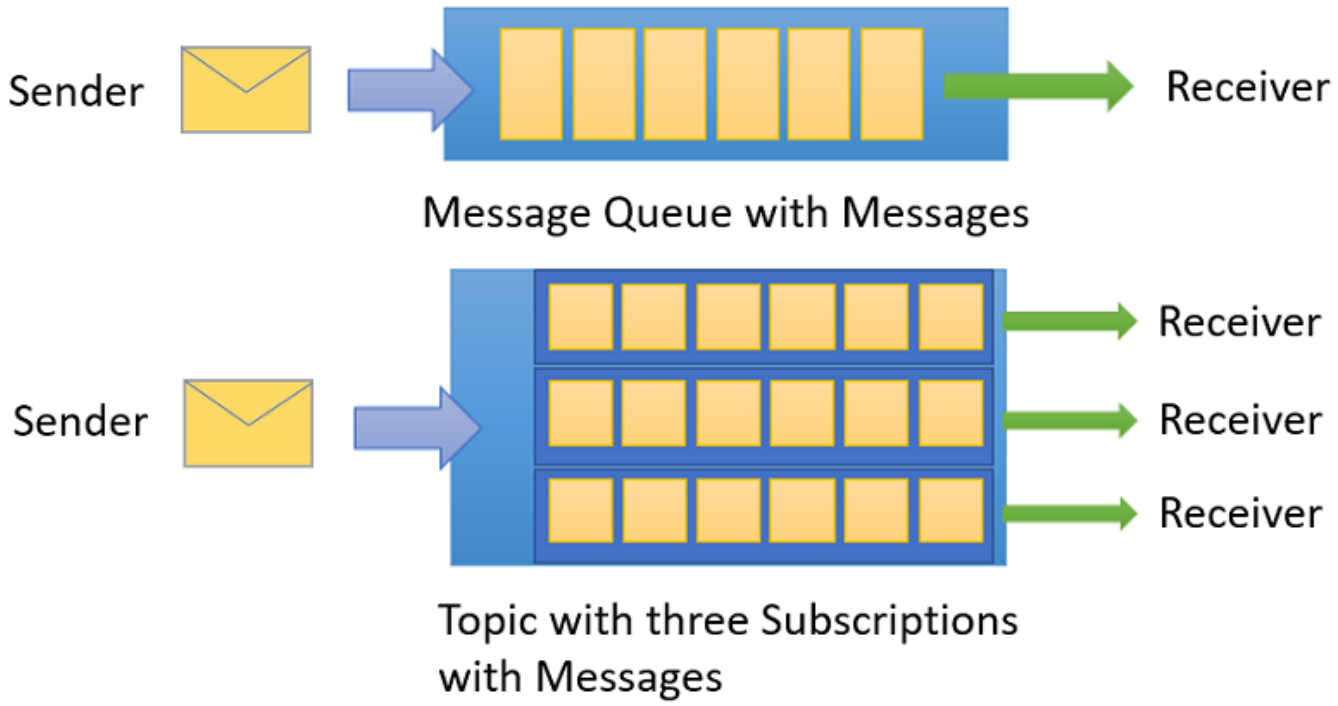


Рисунок 5.2 – Відмінність Queue від Topic

### 5.2.3 Зберігання даних

Azure надає багато сервісів для збереження системних даних. Окрім нативних для хмари сервісів Azure Storage, Cosmos DB, Azure SQL, існує безліч різнопланових баз даних від інших поставників, такі як: MongoDB, Redis, Elasticsearch, MySQL, Oracle Database, і т.д. Визначено, що для реалізації проекту необхідно 2 види сховищ – об’єктне та документне.

Об’єктне сховище призначене для зберігання великих об’ємів неструктурованих даних – даних, що не мають чіткої моделі або визначення. Такими даними є текстові та двійкові файли. Тому в якості об’єктного сховища систему був обраний сервіс Azure Blob Storage, так як якісних альтернатив йому в Azure немає.

Blob Storage оптимізований для наступних сценаріїв:

- зберігання статичних файлів-ресурсів веб-сайту з можливістю віддачі їх прямо до веб-браузера;
- зберігання файлів для розподіленного доступу;
- потокове відображення аудіо- та відеоматеріалів;

- запис до файлів-журналів;
- зберігання даних бекапів, архівів.

Інтеграція з сервісом відбувається через HTTP або через SDK.

Складніше визначитися з сховищем метадати. Оскільки структура даних є фактично нереляційною, є сенс розглядати лише нереляційні бази даних, які описуються загальним поняттям NoSQL – not only SQL. Розглянуті документні сховища з їх перевагами та недоліками описані в таблиці 5.3.

Таблиця 5.3 – Порівняння сервісів зберігання документоподібних даних

| Назва | Характеристика | Переваги | Недоліки |
|---|---|---|---|
| Azure Storage Table | Безсхемне табличне сховище для структурованих даних | Простота використання. Низька вартість. Підтримка Azure Functions | Мало можливостей обробки даних |
| Azure Cosmos DB | Високомасштабована документна база | Має різні інтерфейси взаємодії | Висока вартість. Надлишковість функціоналом |
| MongoDB | Відкрита документна база | Висока підтримка спільнотою | Надлишковість функціоналом |
| Elasticsearch | Документна база, спрямована на повнотекстовий пошук | - | Важкість розгортання. Надлишковість функціоналом |

В якості документного сховища був обраний сервіс Azure Storage Table. Один екземпляр сервісу являє собою таблицю, кожен рядок якого містить композитний ключ, що складається з атрибутів PartitionKey та RowKey. Таблиця не має жорсткої схеми (як в SQL базах даних) – тому може працювати з напівструктурованими даними. Сервіс добре рекомендує себе у сценаріях, коли необхідно зберігати дані, що не потребують складних об’єднань чи відношень.

### 5.2.4 Служба моніторингу та аналітики

Для моніторингу за станом роботи системи зручно використовувати системи Application Performance Management. Нативним для компонентів Azure є сервіс Application Insights, який дає можливість повного моніторингу системи. Application Insights – це агрегатор телеметрії. Використовується для перегляду, фільтрації, аналізу журналів і відстеження шляхів виконання додатків. Важливою особливістю є повідомлення власника програми у разі виникнення вийняткових ситуацій. Сервіс підтримує як інтерактивний, так і передбачальний аналіз даних за допомогою модулів машинного навчання.

Application Insights здатен оброблювати такі події:

- trace – трасування виконання функцій та процесів;
- request – запити користувача (в основному HTTP запити);
- page view – перегляд сторінок на веб сайті або екранів додатків у мобільних та настільних додатках;
- exception – помилки та вийняткові ситуації, які виникли в роботі системи;
- dependency – звернення до іншої частини системи або до залежності (HTTP запити системи, звернення до бази даних, тощо);
- availability – метрики, що показують чи є система доступною для зовнішніх користувачів.

Значною перевагою Application Insights є простота інтеграції з Azure Functions, що автоматизовує передачу метрик з бізнес-логіки системи.

### 5.2.5 Середовище роботи механізму розпізнавання зображень

Ядро розпізнавання зображення представлене у вигляді ML моделі, що побудована за допомогою бібліотеки машиного навчання Keras[5]. Ця високорівнева бібліотека призначена для побудови та виконання штучних нейронних мереж, які здатні виконуватися як на центральному процесорі (CPU), так і на графічному прискорювачі (GPU). Оскільки, згідно до вимог до системи, від механізму розпізнавання зображень потребується висока продуктивність, а її можливо досягти лише використовуючи графічний прискорювач. Відштовхуючись від цього, є два варіанти хмарного середовища обчислення з підтримкою GPU:

- GPU-enabled container instance – потребує запакування механізму в Docker контейнер. Перевагою є мінімалістичність середовища, але блокуючим недоліком є нестабільна робота та повільний запуск і налаштування механізму (в середньому на це витрачається до години);
- Virtual Machine з GPU – Azure пропонує широкий спектр віртуальних машин для різних призначень. Існує окремий клас віртуальних машин, що призначений для роботи з Data Science і пропонує окрім стандартного набору компонентів екземпляр GPU. Перевагами використання такого середовища є необхідність налаштовування лише один раз та відносно велика швидкість запуску та зупинки середовища.

На основі порівняння була обрана віртуальна машина як модель середовища роботи ML моделі.

## 5.3 Висновки до розділу

У цьому розділі роз'яснюються та визначаються основні технології системи та з’ясовуються питання, пов'язані з побудовою складної архітектури системи.

В якості основної платформи розробки була обрана платформа .NET Core завдяки великій продуктивності виконання коду та великій продуктивності розробки на цій платформі. Середовищем виконання додатку було обрано Azure Function, що

дозволяє розділяти логіку на дрібні модулі, робота яких представляє роботу усієї системи.

Для забезпечення комунікації між функціями був обраний сервіс Azure Service Bus, який реалізує патерн «брокер повідомлень» і забезпечує асинхронний обмін даними та подіями між компонентами системи.

Azure Storage, а саме такі його компоненти як Azure Storage Table та Blob Storage Container були обрані для забезпечення надійного, масштабованого зберігання даних – файлів зображень, вхідних метаданих та результатів розпізнавання.

В якості системи агрегації метрик був обраний сервіс Application Insights – потужна система типу Application Performance Management, що дозволяє ефективно аналізувати стан роботи системи. Використання цього сервісу допоможе швидко реагувати на помилки виконання, що можуть виникати під час роботи системи.

Для створення середовища виконання ML моделі була обрана віртуальна машина з графічним прискорювачем, що забезпечить високу швидкsсть обробки зображень.

Комплекс технологій і хмарних сервісів компанії Microsoft в повній мірі забезпечує усім необхідним для розробки сучасних, доступних і масштабованих програмних систем. Важливою перевагою є те, що в разі більш різнорідних компонентів (від різних постачальників), інтеграція та конфігурація компонентів і сервісів може бути більш трудомісткою.

# 6 РЕАЛІЗАЦІЯ ЛОГІКИ СИСТЕМИ

Після вибору технологій проектування системи та аналізу вимог до системи було визначено, що система повинна бути модульною та мати сервісну архітектуру. Для реалізації системи необхідно розробити бізнес-логіку і налаштувати хмарне середовище для ефективної та продуктивної роботи системи.

Процес реалізації системи був поділений на наступні етапи:

- налаштування хмарних сервісів;
- розробка модуля базової інтеграції для можливості імпорту даних через HTTP API;
- розробка модуля обміну даними з сховищем;
- розробка модуля керування механізму розпізнавання зображень;
- розробка модуля взаємодії з механізмом розпізнавання зображень;
- розміщення модулей з бізнес-логікою систему у хмарному середовищі.

Саме таких порядок забезпечує оптимальну швидкість розробки системи завдяки тому, що модулі розроблються в тому порядку, в якому буде направлятися потік даних в реальному використанні.

## 6.1 Розгортання і налаштування хмарних сервісів

Для початку роботи з Azure необхідно мати обліковий запис Microsoft (можливо використовувати як особисті, так і робочі облікові записи) та активну підписку на Azure. Для налаштування сервісів Azure можна користуватись або веб-порталом або утилітою командного рядка azure-cli.

Для початку зручно буде створити окрему ресурсну групу під проект. Ресурсна група дозволяє логічно групувати ресурси Azure. При створенні більшої частини ресурсів необхідно вказувати регіон – це атрибут, що вказує у якому із центрів обробки даних необхідно розгортати сервіси. Регіон також може впливати на перелік доступних сервісів та їх вартості. Тут і надалі для усих ресурсів. був обраний регіон

«Europe West», тому що саме цей датацентр знаходиться найближче до регіону просування проекту.

По-перше, в новостворенній ресурсній групі необхідно створити App Service Plan – віртуальний сервіс буде слугувати середовищем виконання модулів системи на основі Azure Functions. При створенні можна вказати операційну систему – Linux або Windows. Так як у App Service Plan на основі Windows більша функціональність, оберемо саме цю ОС. Також необхідно обрати рівень ресурсів – для забезпечення безперебійної роботи системи необхідно вибрати рівень не менше ніж Standard S1 – менші рівні не підтримують режим постійного включення, що необхідно для виконання певних дій за регламентом.

На основі створеного App Service Plan необхідно створити контейнери для виконання логіки модулей – Function App. Під час створення необхідно надати глобально унікальне ім’я для сервісу – йому надається ім’я хоста у вигляді {instance name}.azurewebsites.net, що використовується для HTTP тригерів функцій. Необхідно створити по одному Function App для кожного модулю з бізнес-логікою (рис. 6.1).

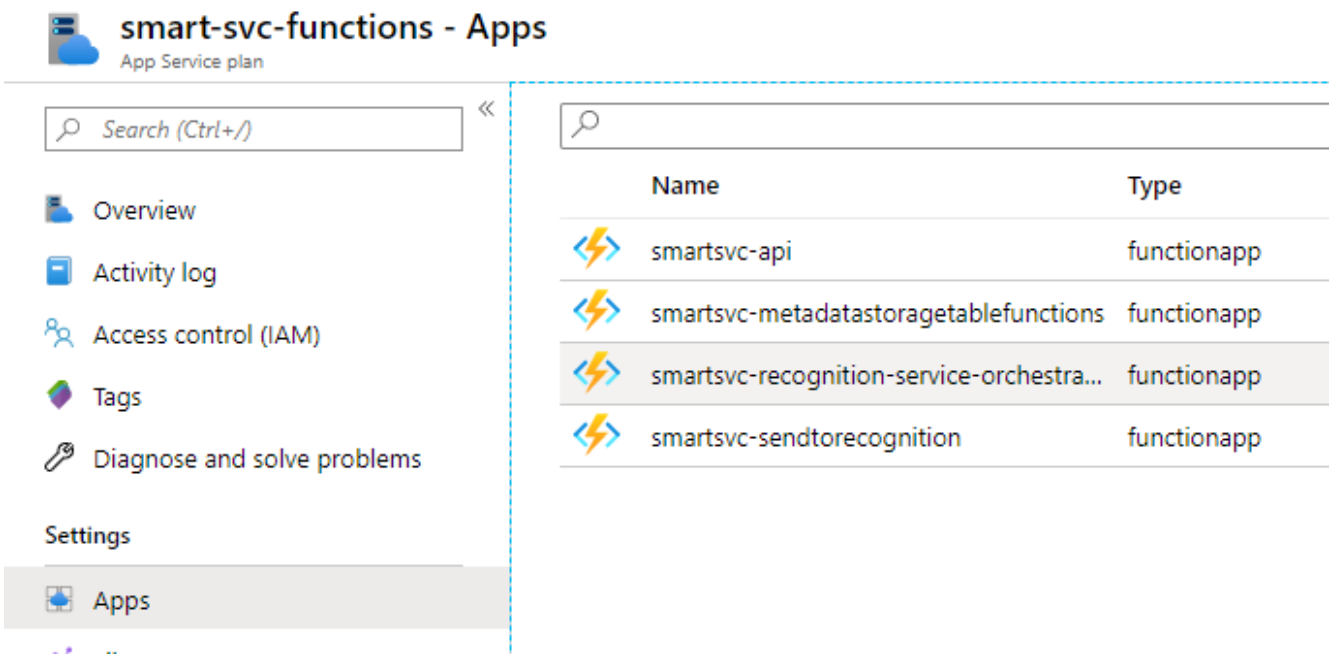


Рисунок 6.1 – Створені Function App в рамках App Service Plan

Далі потрібно створити Storage Account – контейнер для сервісів зберігання інформації. Йому також необхідно надати глобально унікальну назву та обрати такі налаштування: рівень продуктивності, тип реплікації та тип взаємодії. В створенному Storage Account створюємо Blob Container в якості об’єктного сховища та таблицю для вхідних метаданих і результатів розпізнавання.

Для забезпечення комунікації між функціями необхідно створити екземпляр брокеру повідомлень Azure Service Bus. Завдяки структурній схемі зрозуміло, які функції будуть взаємодіяти між собою. Важливим є правильне використання підходів queue та topic/subscription - необхідно проаналізувати систему та визначити, де взаємодія має відбуватися строго лише між двома модулями. Визначено, що таким логічним з’єднанням є лише взаємодія між механізмом розпізнавання зображень та модулем взаємодії з цим механізмом. Тому для цього з’єднання оберемо тип брокера – queue, для інших – topic. Для управління ресурсом Azure Service Bus окрім веб-порталу, зручно користуватися спеціальним додатком Service Bus Explorer [6]. Створений та налаштований екземпляр сервісу можна побачити на рисунку 6.2.

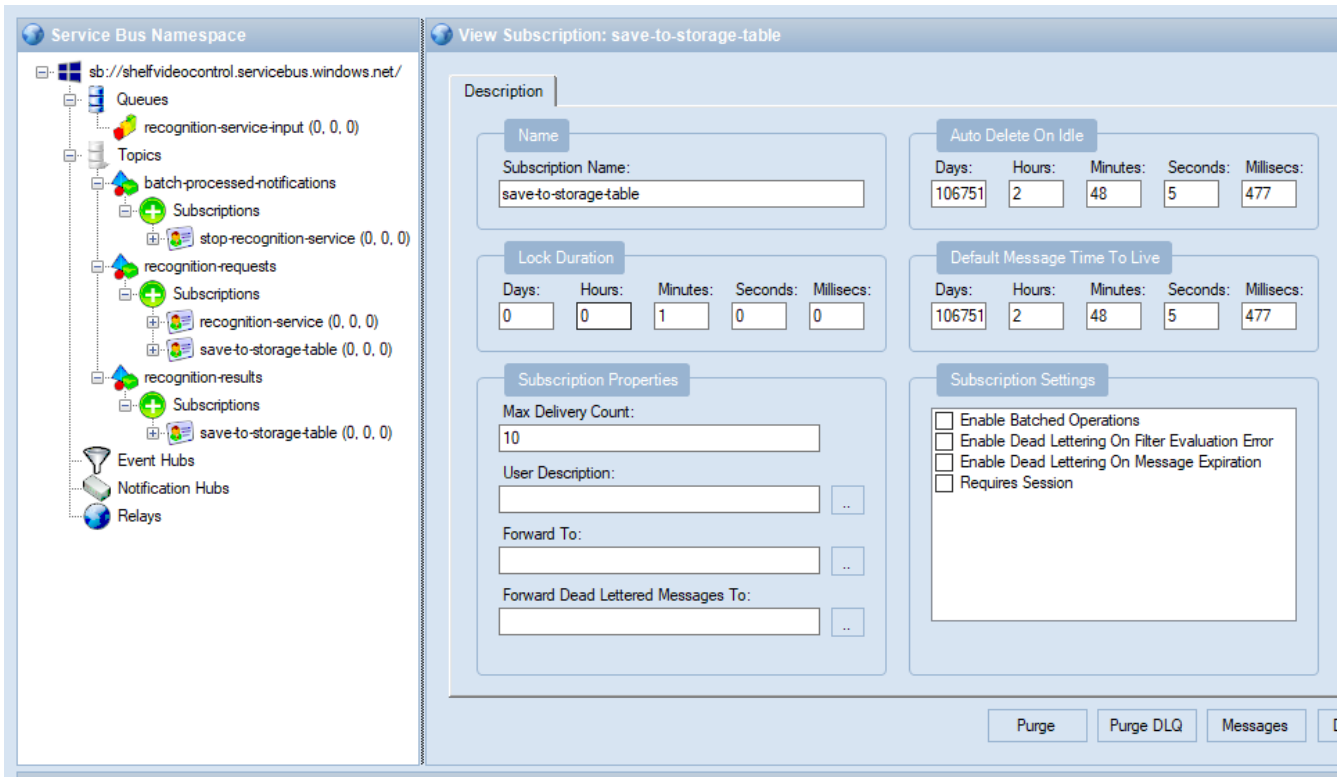


Рисунок 6.2 – Налаштований екземпляр сервісу у додатку Service Bus Explorer

Створення і налаштування ресурсу Application Insights є простішим у порівнянні з іншими ресурсами. Для підключення до створенного екземпляру використовується так званий InstrumentationKey – унікальний GUID (globally unique identifier), якого цілком достатньо для підключення додатків до APM. Підключення до функцій передбачає виставлення параметру APPINSIGHTS_INSTRUMENTATIONKEY в панелі Application Settings фукціональних додатків.

Створення віртуальної машини починається з вибору необхідної операційної системи. Оскільки необхідна підтримка графічного прискорювача, вибір обмежується платформою Linux, а саме операційною системою Ubuntu 16.04 LTS. Слід зауважити, що віртуальні машини з GPU є надзвичайно потужними в плані потужності центрального процесора та об’єму оперативної пам’яті, і, відповідно, відносно дорогими. Тому оберемо найдешевшу віртуальну машину з GPU – рівень NC6 [7]. На даній машині використовується графічний прискорювач NVIDIA Tesla K80. Під час створення Azure пропонує багато різнопланового функціоналу по управлінню машиною, але важливими для системи є наступні дії:

- налаштування фаєрволу – необхідно відкрити порти 22 (для підключення по протоколу віддаленого виклику команд ssh) та 443 (для взаємодії з HTTP API механізму розпізнавання) для вхідних з’єднань;
- встановлення розширення NVIDIA GPU Driver Extension для розширенної можливості взаємодії з графічним прискорювачем;
- налаштування режиму автовідключення на випадок відмови модулю керування віртуальною машиною.

Створення віртуальної машини за замовчуванням створює і необхідні ресурси, що забезпечують її роботу:

- мережевий інтерфейс;
- публічна IP адреса;
- віртуальний диск;
- віртуальна мережа;
- Network Security Group – фаєрвол.

Після розгортання усіх сервісів ресурсна група виглядає так, як зображено на рисунку 6.3.

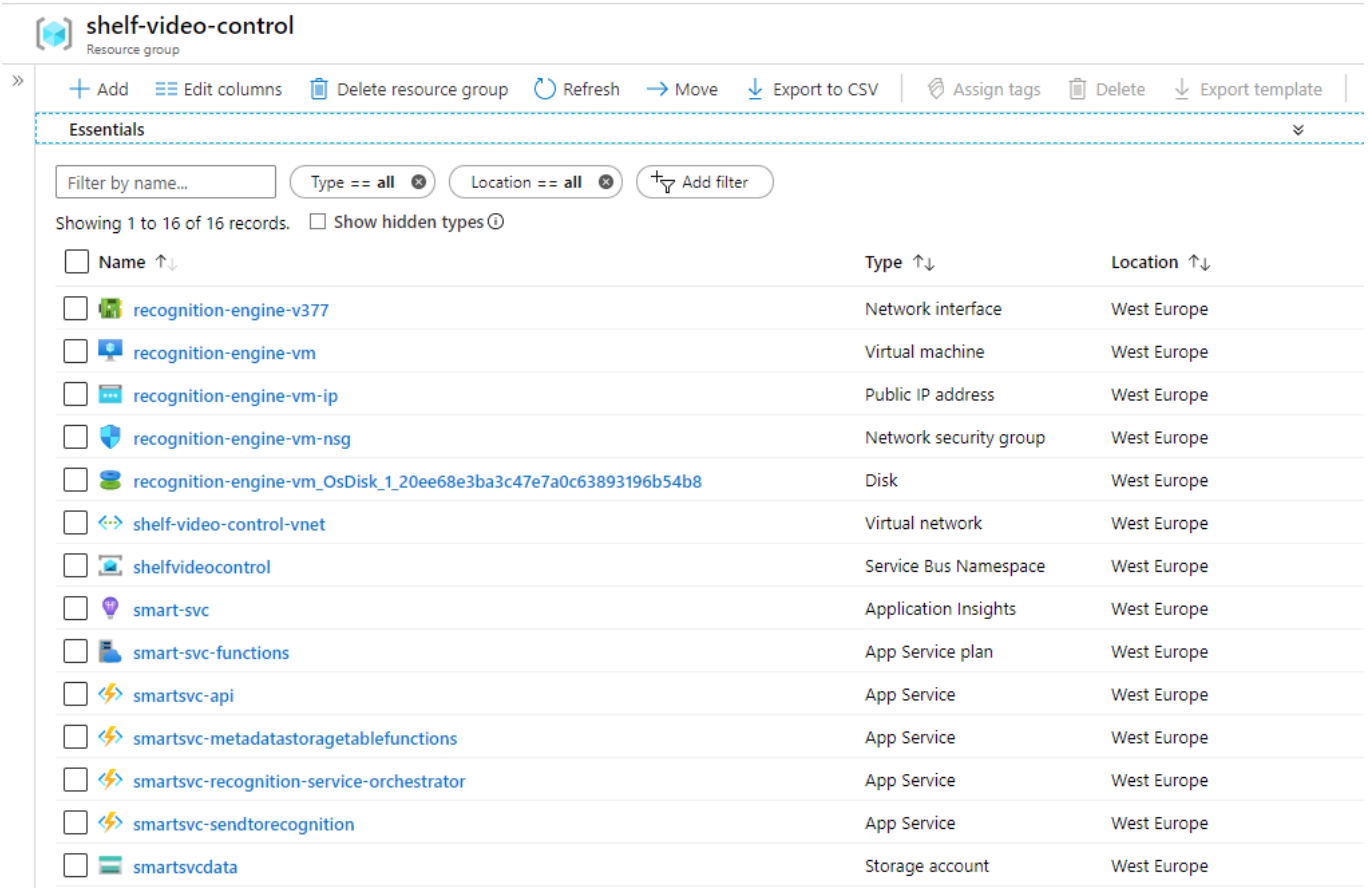


Рисунок 6.3 – Ресурсна група

Наявність сервісів (насамперед Azure Service Bus) забезпечує можливість розробки модулів бізнес-логіки системи.

## 6.2 Реалізація модуля базової інтеграції для імпорту даних

У вимогах до системи не передбачена необхідність розробки модуля інтеграції, а лише наявність точок інтеграції для інших систем, однак на практиці дуже зручно розробити примітивний модуль, що використовує інтерфейс точки інтеграції. Дуже зручно використовувати інтеграційни модуль, що є адаптером для точки інтеграції і дає можливість працювати з системою за допомогою розширеному переліку протоколів. Такий підхід має значні переваги:

- налагодження додатку прискорюється завдяки зручному для ручної взаємодії інтерфейсу;
- розробник має змогу оцінити інтерфейс інтеграції з точки зору споживача, що надає змогу спроектувати інтерфейс найкращим чином;
- на практиці такі інтеграції-адаптери часто стають єдиним способо взаємодії з іншими системами.

Тому було прийняте рішення побудови модуля-адаптеру на основі HTTP API. Такий модуль має єдину кінцеву точку (endpoint) та сприймає HTTP запити у форматі JSON (JavaScript Object Notation). Кожен запит має містити колекцію об’єктів, кожен з яких містить такі властивості:

- id – рядок, що містить унікальний ідентифікатор;
- metadata – об’єкт, що містить властивості з типами число або рядок;
- contentType – рядок з типом контенту згідно до стандарту RFC822 [8];
- data – рядок, що містить зображення, серіалізоване у формат Base64.

Структура функції описана у таблиці 6.1, а алгоритм обробки кожного вхідного об’єкту – на рисунку 6.4.

Таблиця 6.1 – Структура фунції UploadPictures

| Назва функції | UploadPictures |
|---|---|
| Тригер | HTTP запит |
| Залежності | Blob Container – pictures<br>Storage Table – pictureMetadata.<br>Service Bus Topic – recognition-pictures. |
| Результат | HTTP відповідь |

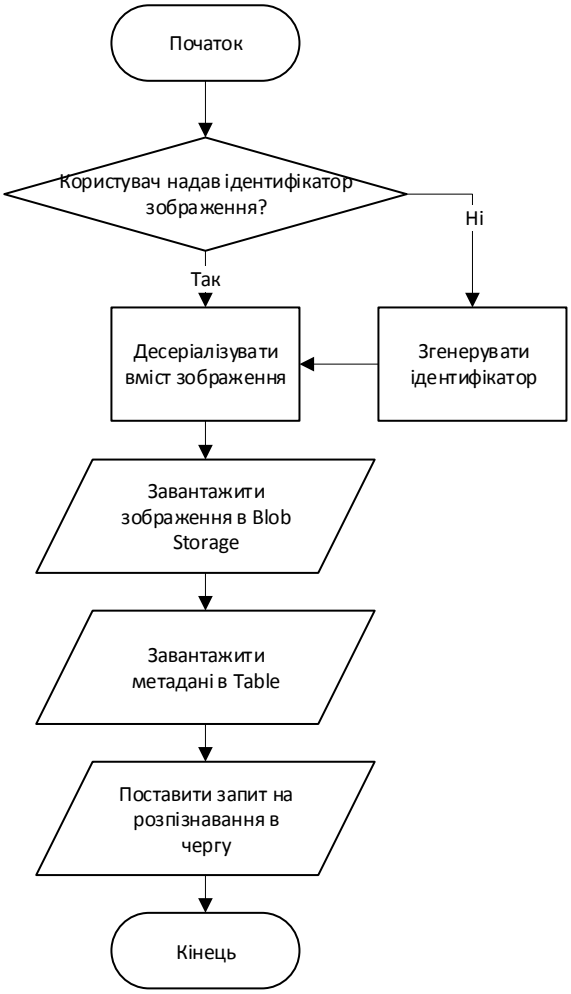


Рисунок 6.4 – Алгоритм обробки одного зображення

Спроектований модуль є зручним для інтеграцій з середовищами, які не підтримуються Azure Service Bus (такі як, наприклад, веб-браузер).

## 6.3 Реалізація модуля взаємодії зі сховищем

Модуль обміну взаємодії зі сховищем призначений тільки для маніпуляції з даними в таблиці. Він містить 3 функції.

Функція FlattenRequestMetadata призначена для представлення об'єкту метаданих у плоскому форматі в таблиці. Її структура описана в таблиці 6.2. Метадані зберігаються у форматі JSON в окремій колонці таблиці під час процесу імпорту. Вказана функція читає властивості JSON об’єкту та створює окрему колонку під кожну властивість (якщо колонки ще не існує), а в відповідному рядку відповідної

колонки проставляє значення з об’єкту. Концепт та принцип роботи «випрямлення» показаний на рис. 6.5.

Таблиця 6.2 – Структура функції FlattenRequestMetadata

| | |
|---|---|
| Назва функції | FlattenRequestMetadata |
| Тригер | Service Bus Subscription – recognition-requests/flatten-request-metadata-in-table |
| Залежності | Storage Table – pictureMetadata. |
| Результат | - |

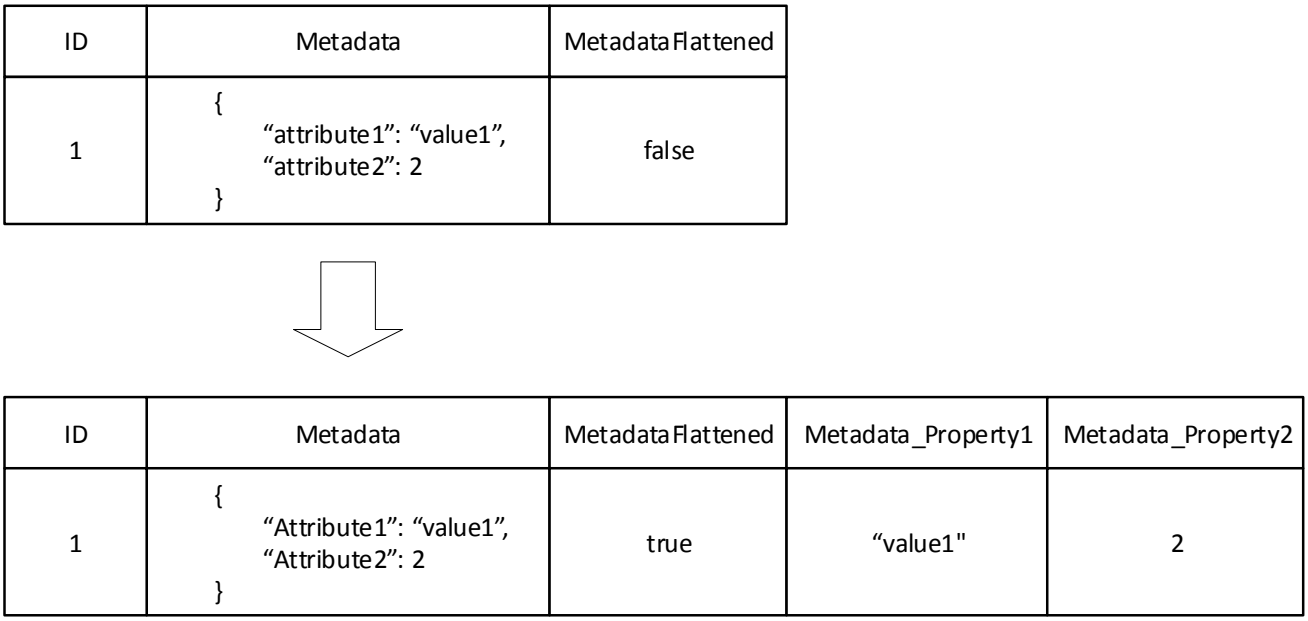


Рисунок 6.5 – Процес випрямлення метаданих

Назва функції SaveRecognitionResultToStorageTable є самоописовою. Вона призначена для обробки подій надходження корисних результатів розпізнавання. Структура функції описана в таблиці 6.3.

Таблиця 6.3 – Структура функції SaveRecognitionResultToStorageTable

| | |
|---|---|
| Назва функції | SaveRecognitionResultToStorageTable |
| Тригер | Service Bus Subscription – recognition-results/save-to-storage-table |

Продовження таблиці 6.3

| | |
|---|---|
| Назва функції | SaveRecognitionResultToStorageTable |
| Залежності | Storage Table – pictureMetadata. |
| Результат | - |

Розроблена аналогічна функція для обробки помилок, що виникли в механізмі розпізнавання під час своєї роботи – SavePictureRecognitionError. Її структура описана в таблиці 6.4.

Таблиця 6.4 – Структура фунції SavePictureRecognitionError

| | |
|---|---|
| Назва функції | SavePictureRecognitionError |
| Тригер | Service Bus Subscription – recognition-errors/save-to-storage-table |
| Залежності | Storage Table – pictureMetadata. |
| Результат | - |

Оскільки потоки важливих даних зосереджені в топіках recognition-requests, recognition-results, recognition-errors, саме вони є точками інтеграції та забезпечують гнучність системи і простоту інтеграції інших систем.

## 6.4 Реалізація модуля управління процесом розпізнавання зображень

Даний модуль призначений для керуванням середовищем, де розміщений механізм розпізнавання зображень. Модуль містить 2 функції.

StartRecognition – функція, що запускає та підготовлює механізм розпізнавання до експлуатації. Саме ця функція може запускатися за регламентом, що вказаний в панелі адміністрування Azure, а може запускатися вручну за вимогою адміністратора. Структура функції описана в таблиці 6.5.

Таблиця 6.5 – Структура функції StartRecognition

| | |
|---|---|
| Назва функції | StartRecognition |
| Тригер | Timer / on-demand |
| Залежності | Azure SDK - Virtual Machine Connection.<br>Service Bus Queue – recognition-service-input.<br>Service Bus Subscription - recognition-requests/recognition-service. |
| Параметри | Schedule – cron вираз регламенту запуску.<br>RecognitionBatchSize – максимальна кількість повідомлень, що може бути перенаправлена на механізм розпізнавання.<br>RecognitionServiceEndpointTemplate – шаблон шляху до кінцевої точки з розпізнаванням.<br>HealthCheckEndpointTemplate – шаблон шляху до кінцевої точки для перевірки статусу механізму.<br>Azure... – набір параметрів для підключення до Azure програмним шляхом та можливості керування станом віртуальної машини. |
| Результат | - |

Саме проект цієї функції забезпечує ефективність та економічність усієї системи. Логіка роботи полягає у тому, що за наявності запитів на розпізнавання у відповідній черзі функція запускає механізм розпізнавання та пересилає деяку кількість запитів в іншу чергу, призначену для взаємодії з сервісом розпізнавання. Останне переслане повідомлення містить спеціальну ознаку, яка повідомляє, що обробка пакету запитів закінчилась і механізм розпізнавання можно вимикати. Алгоритм роботи, що реалізовує функція, зображений на рисунку 6.5.

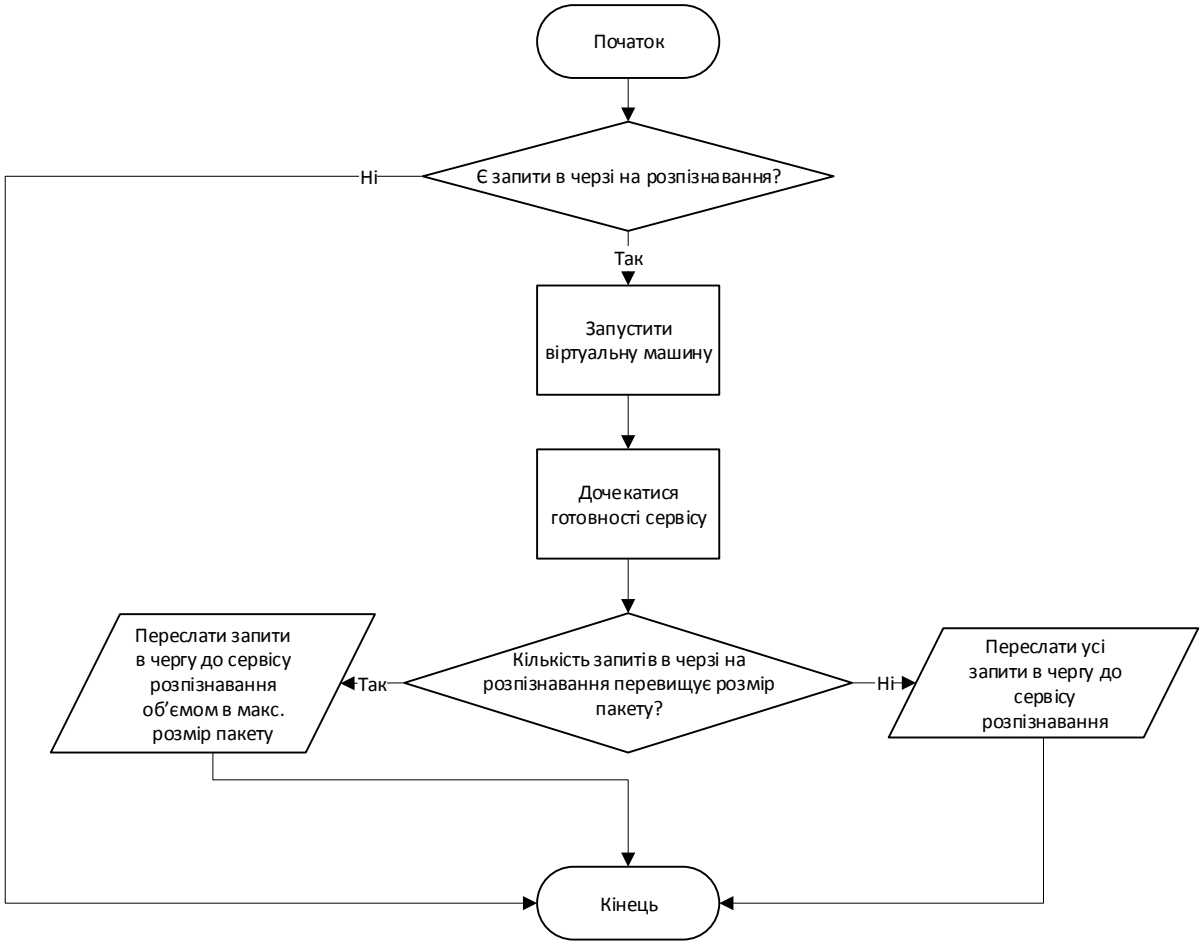


Рисунок 6.5 – Блок-схема алгоритму роботи функції StartRecognition

Зворотньою є функція StopRecognitionService – яка реагує на подію «оброблене останній запит в пачці» і зупиняє віртуальну машину. ЇЇ структура описана в таблиці 6.6.

Таблиця 6.6 – Структура фунції StartRecognition

| Назва функції | StopRecognitionService |
|---|---|
| Тригер | Service Bus Subscription - batch-processed-notifications/stop-recognition-service |
| Залежності | Azure SDK - Virtual Machine Connection. |
| Параметри | Azure... – набір параметрів для підключення до Azure програмним шляхом та можливості керування станом віртуальної машини. |
| Результат | - |

## 6.5 Реалізація модуля взаємодії з сервісом розпізнавання зображень

Після реалізації модулю, що керує середовищем роботи механізму розпізнавання зображень, необхідно забезпечити комунікацію між самим механізмом та системою. Від сервісу вимагається наступний функціонал:

- основна функціональна вимога – можливість приймання зображення для подальшої передачі на ML модель та повернення результатів її виконання на зображенні;
- необхідно забезпечити можливість отримання статусу роботи сервісу (health check), оскільки запуск сервісу бути відносно тривалим через те, що під час запуску ініціалізується графічний прискорювач та підготовлюється ML модель для роботи.

В якості протоколу обміну було обрано HTTP API завдяки своїй простоті. Для подальшої роботи необхідно, щоб сервіс розпізнавання зображень реалізовував інтерфейс, що описаний в таблиці 6.7.

Таблиця 6.7 – Вимоги до HTTP інтерфейсу сервіса розпізнавання

| Функціонал | Взаємодія з ML моделлю | Перевірка статусу сервісу |
|---|---|---|
| Відносний шлях | /recognize | / |
| HTTP метод | POST | GET |
| Формат тіла запиту | JSON<br>{<br>“img”: “base64 string”<br>} | - |
| Можливі статус коди | 200, 500 | 200, 0 (сервіс не відповідає) |
| Формат тіла відповіді | Довільний JSON | - |

Модуль взаємодії з сервісом розпізнавання зображень складається з одної функції SendToRecognition, її структура описана в таблиці 6.8, а алгоритм роботи – на рисунку 6.6.

Таблиця 6.8 – Структура функції SendToRecognition

| | |
|---|---|
| Назва функції | SendToRecognition |
| Тригер | Service Bus Queue – recognition-service-input |
| Залежності | Blob Storage – pictures<br>Service Bus Topic - recognition-errors<br>Service Bus Topic - batch-processed-notifications<br>RecognitionService |
| Результат | Service Bus Message – recognition-results |

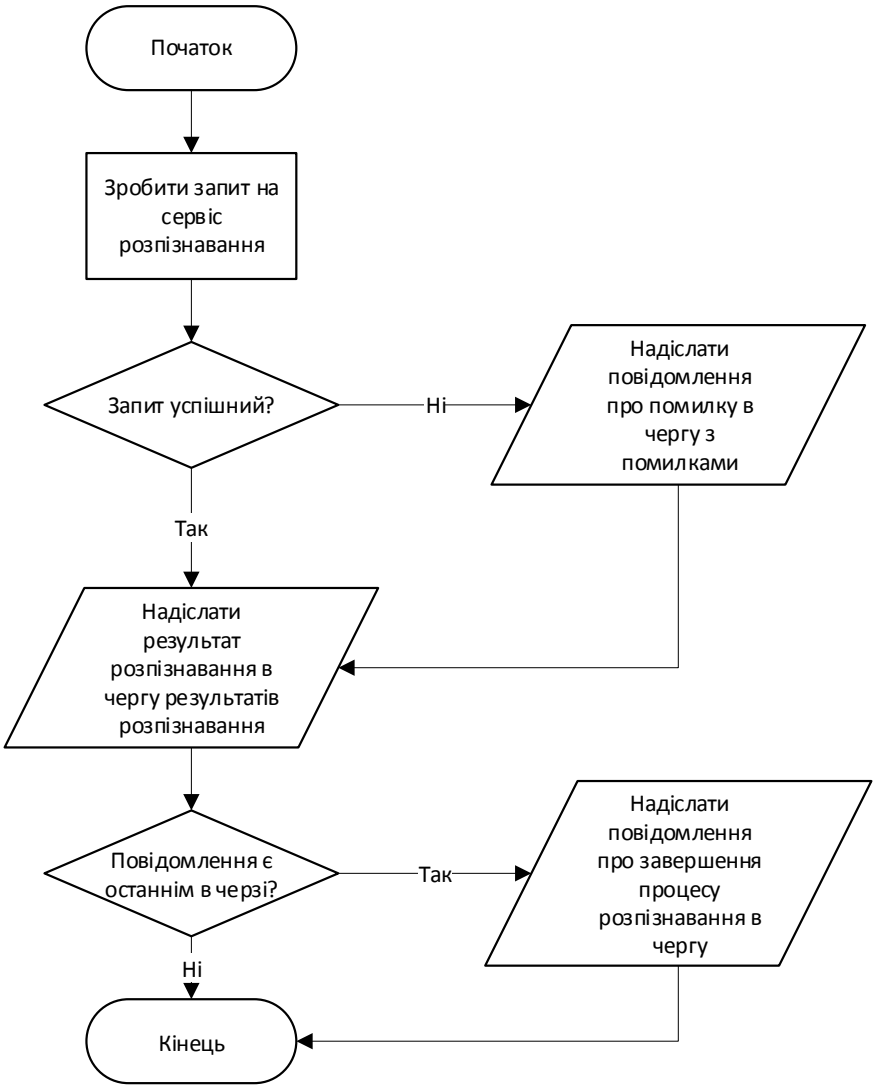


Рисунок 6.8 – Блок-схема алгоритму роботи функції SendToRecognition

## 6.6 Розміщення та налаштування розроблених компонентів у хмарному середовищі

Публікація розроблених функцій у функціональні додатки в хмарному середовищі проводиться в 2 етапи.

Перш за все необхідно розмістити артефакти збірки проектів у відповідних додатках. Для цього можна використати додаток командного рядку Azure Functions Core Tools [9], який підтримує роботу з функціями, що розроблені на усіх підтримуваних технологіх – .NET, JavaScript, Python, PowerShell, PHP, Bash. Але оскільки розробка функціональних модулів проводилася в IDE (інтегроване середовище розробки) Visual Studio 2019, зручно встановити розширення Azure Functions SDK. Кожен функціональний модуль, що логічно представлений .NET проектом, має розміститися на окремий функціональний додаток (рисунок 6.9).

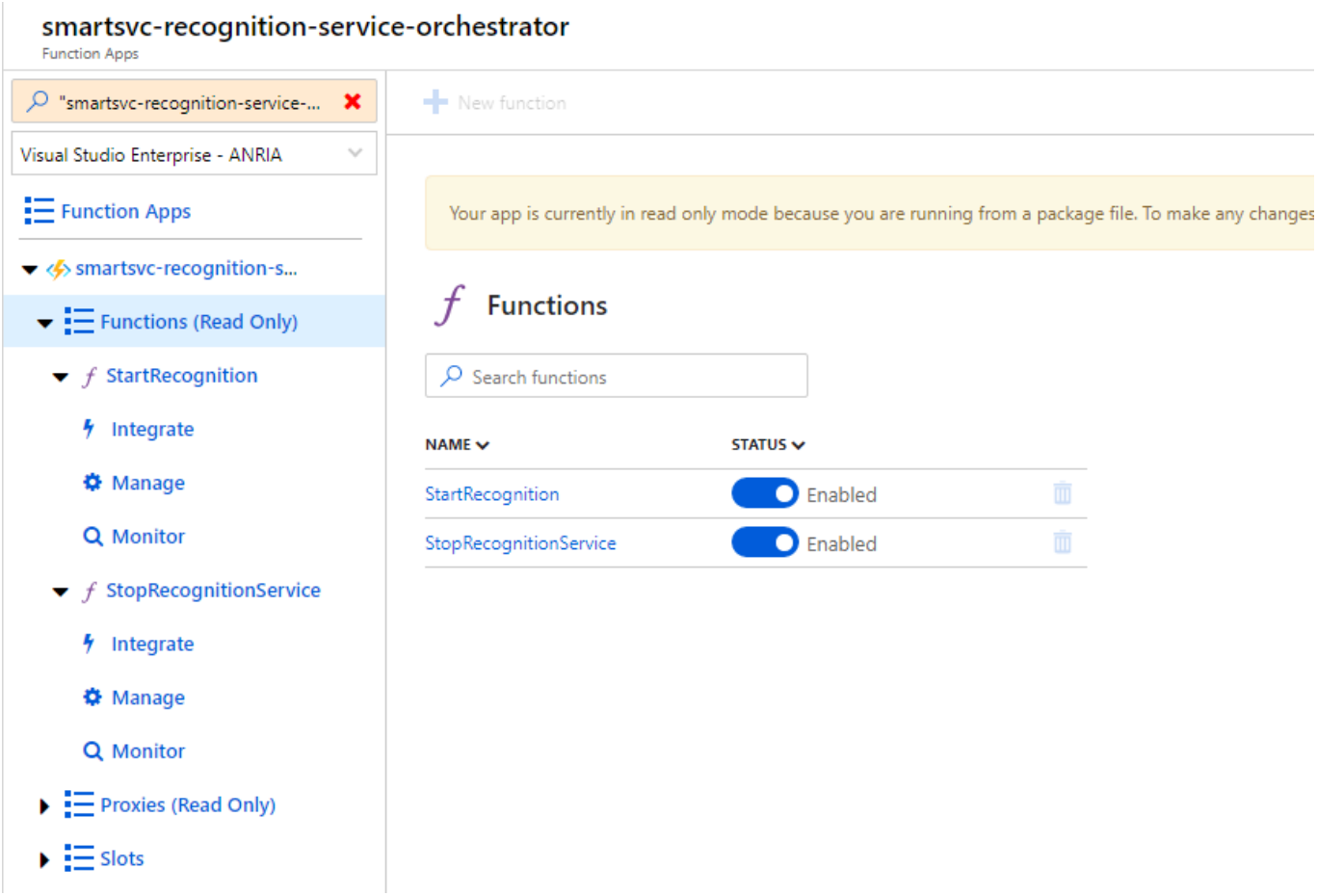


Рисунок 6.9 – Приклад опублікованого функціонального додатку

Але самої публікації не достатньо. Оскільки були використані гарні практики розробки програмного забезпечення, параметри, що можуть змінюватися в залежності від середовища, були винесені в так звану конфігурацію. Такими параметрами, наприклад, є строки підключення або ключі доступу. Механізм конфігурації відповідає за те, що деякі параметри функціональний модуль буде отримувати від середовища виконання. В разі локального запуску під час розробки ці значення цих параметрів задається в файлі local.settings.json, який не буде працювати в хмарному функціональному додатку. Натомість, середовище Azure передає параметри через змінні середовища – environment variables, які задаються в панелі управління функціональним додатком (рисунок 6.10).

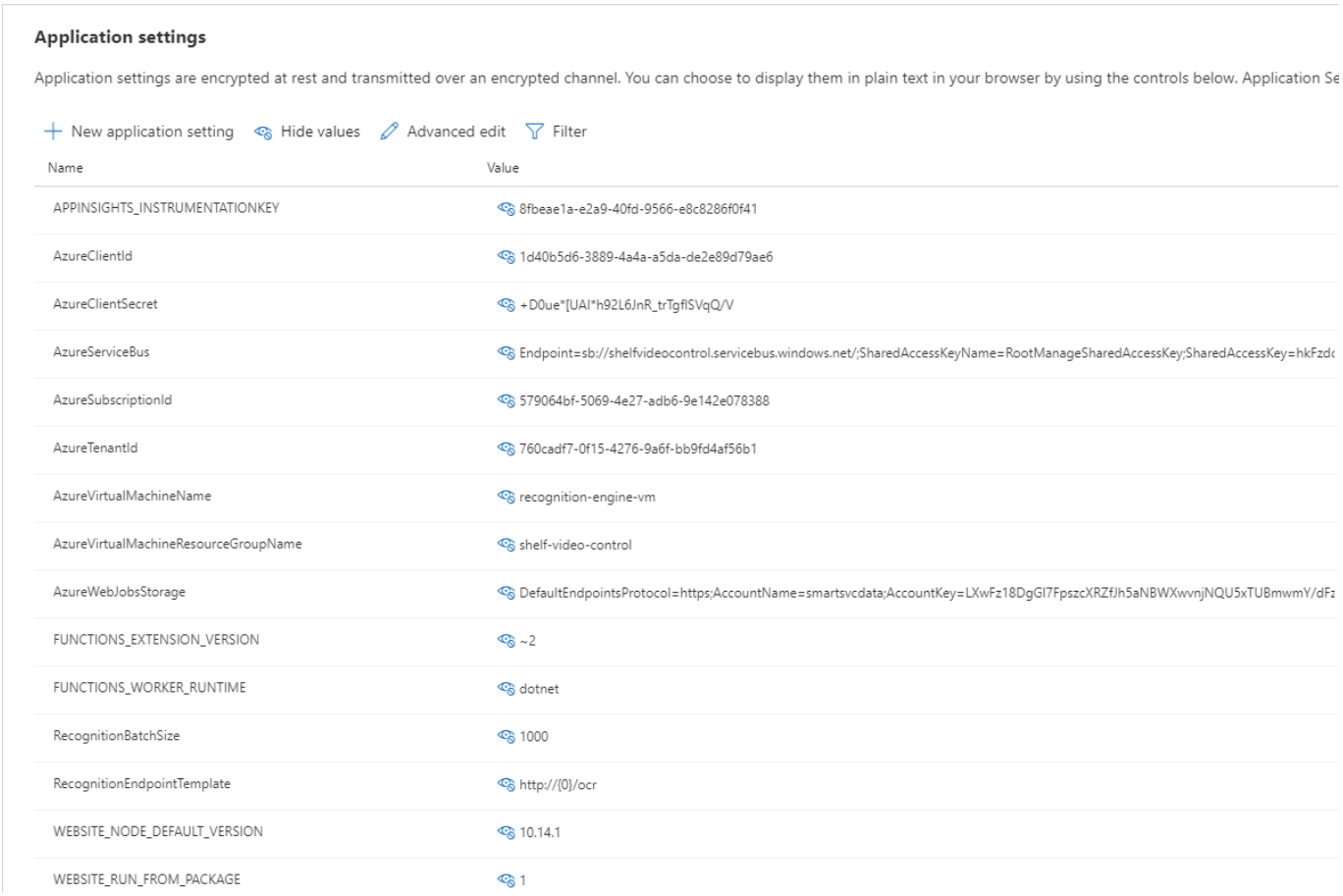


Рисунок 6.10 – Панель конфігурації Azure Function

Після налаштування необхідно пересвідчитись, чи функція коректно працює в новому середовищі. Для цього можна скористатись або розгорнутим екземпляром

Application Insights, або панеллю моніторингу фунціонального додатку (рисунок 6.11).

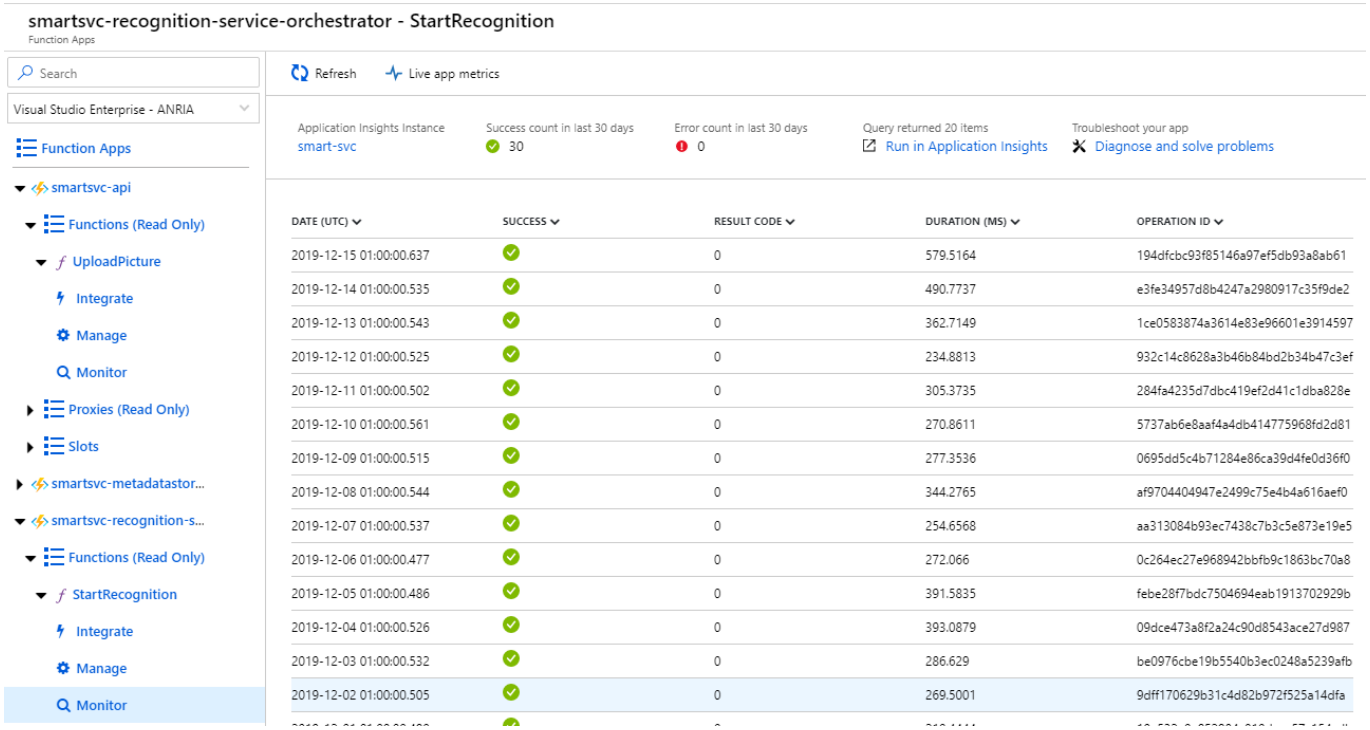


Рисунок 6.11 – Панель моніторингу окремої функції

Оскільки серед розроблених функцій є одна, що викликається за допомогою HTTP запиту, необхідо подбати про неможливість неавторизованої взаємодії з нею. Для цього використовується принцип ключів авторизації [10]. Є такі види авторизації:

- без авторизації – функція запускатися для запитів без наданого ключа;
- ключ функціонального додатку (host key) – спільний ключ для екземпляру функціонального додатку надає доступ до всіх розміщених там функцій;
- ключ функції (function key) – створюється для окремої функції і надає доступ лише до неї.

Для функцій, що мають інакший тип виклику, така авторизація не потрібна. Такі функції інтегровані з хмарними сервісами Azure, усі ключі до яких розміщенні у конфігурації функціонального додатку.

Важливо зазначити, що етап створення функціонального додатку за замовчуванням створює нові екземпляри Azure Storage та Application Insights. Але на практиці зручніше мати по одному екземпляру кожного сервісу, тому бажано видалити створені сервіси (вони мають напіввипадкову назву) та налаштувати функціональні додатки на окремі сервіси, змінивши ключ інструментації (для Application Insights) та строку підключення (для Azure Storage) в конфігурації.

## 6.7 Висновки до розділу

Завдяки високій модульності кодування усієї системи було поділене на етапи. Першим етапом розробки є створення необхідної інфраструктури для забезпечення роботи функціональних модулей навіть під час їх розробки. Ключовим ресурсом в такому плані є брокер повідомлень, так как переважна частина функцій мають тригер на появу нових повідомлень.

Після цього відбувається кодування самих функціональних модулей в порядку, що співпадає з логічним потоком даних у системі. Так, спочатку було розроблено модуль інтеграції з HTTP, функція якого записувала передані користувачем дані в окрему чергу. Далі розроблявся функціональний модуль, що отримує повідомлення з цієї черги, певним чином оброблювала їх та відправляла повідомлення в іншу чергу. Таким чином останнім розробленим функціональним модулем став модуль взаємодії з сервісом розпізнавання. На даному етапі розробки системи сервіс представляє собою чорний ящик, що реалізує окремий інтерфейс взаємодії. Взаємодія і структура функціональних модулей зображена на додатках Г та И відповідно.

Останнім етапом розробки логіки системи стає публікація та налаштування усіх функціональних модулів в хмарному середовищі. Публікація до завершення роботи над проектом, з урахуванням хмаро-орієнтованих технологій, допомагає рано виявити та вирішити інтеграційні помилки. Публікація системи детально описана у додатку В.

Основний процес системи – процес розпізнавання зображень з черги описаний у додатках Д, Е, Ж.

# 7 РОЗРОБЛЕННЯ ТА АДАПТАЦІЯ МОДЕЛІ МАШИНОГО НАВЧАННЯ

Для початку роботи над інтеграцією ML моделі у систему необхідно зрозуміти структуру та архітектуру моделі, щоб вибрати найбільш відповідні технології забезпечення ефективної взаємодії з зовнішнім середовищем.

## 7.1 Опис розробки типової моделі машиного навчання для розпізнавання зображень

Наразі існує декілька виокремлених та узагальнених задач комп’ютерного зору. Перша задача звучить як «класифікація зображень» і включає в себе присвоєння зображенню мітки, яка відповідає за опис усього зображення.

Наступними слідують задачі, що включають в себе більшу деталізацію зображення. Задача локалізації об’єктів полягає у тому, що окрім мітки, модель машиного навчання знаходить прямокутник, який якомога точніше містить об’єкт, пов’язаний з міткою.

Задача виявлення об’єктів полягає у тому, щоб знайти та виокремити прямокутники, що обмежують окремі об’єкти, що знаходяться на одному зображенні.

Задача сегментації полягає у максимально точному виокремленні окремих об’єктів на зображенні. Мета сегментації полягає в тому, щоб спростити та/або змінити відтворення зображення в щось більш значуще і більш легке для аналізу. Сегментація зображень зазвичай використовується для визначення місця розташування об'єктів і меж (ліній, кривих і т. д.) в образах. Більш точно, сегментація зображення-це процес присвоєння мітки кожному пікселю зображення таким чином, щоб пікселі з однією і тією ж міткою мали певні характеристики.

Наглядне порівняння результатів роботи моделей під різні задачі зображено на рисунку 7.1.

Для розробки моделі була обрана бібліотека машиного навчання Keras та архітектура мережі RetinaNet [11]. Задача, що вирішувалась – виявлення об'єктів на зображенні.

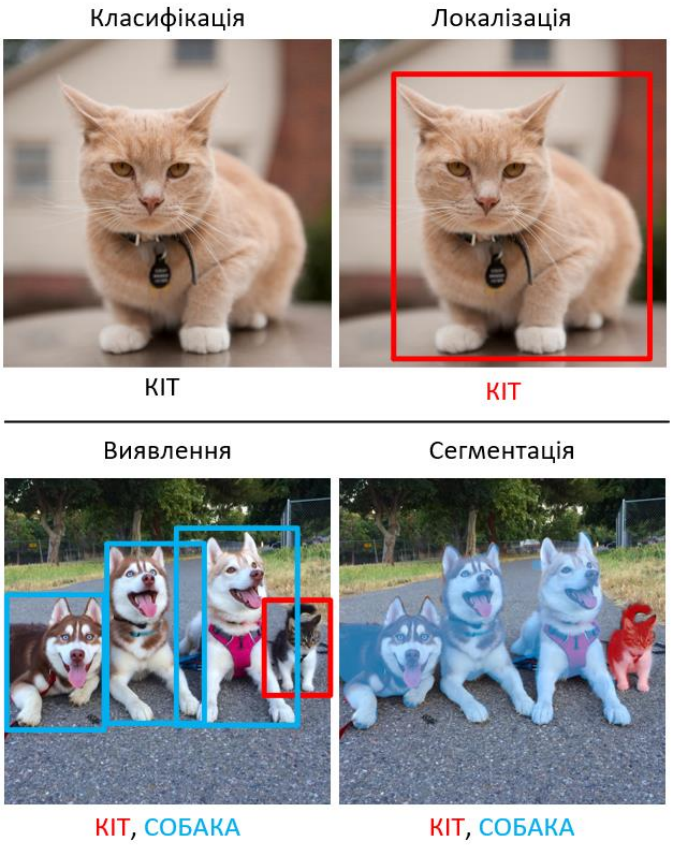


Рисунок 7.1 –Типові задачі комп’ютерного зору

Keras – відкрита бібліотека для роботи з нейронними мережами для мови Python. Вона надає високорівневе API для навчання і використання нейронних мереж. В якості бекенду можуть використовуватись різні середовища роботи машинного навчання: TensorFlow, Microsoft Cognitive Toolkot, R та Theano. Ця бібліотека спроектована для швидких експериментів з глибинними нейронними мережами та характеризується високою модульністю та розширюваністю. RetinaNet – архітектура мережі, ща являє собою реалізацію алгоритму Focal Loss for Dense Object Detection [12]. Визначається відносно високими показниками швидкості та точності.

Перший етап створення моделі полягає у навчанні моделі. Для цього необхідно підготувати спеціальний набір даних – зображення певного розміру та метадані, що містять координати обмежуючих полігонів. Такі метадані зручно створювати за допомогою спеціального інструменту для розмітки labelImg [13].

Після підготовки датасету потрібно переходити до самого процесу навчання. В цьому етапі допомагає так зване передаточне навчання, що полягає у донавчані

готової моделі замість повного навчання з нуля. Для цього існують вже готові мережі з попередньо проставленими вагами на ребрах.

Команда командного рядку, що запускає навчання має наступний вид: python3 keras_retinanet/bin/train.py –freeze-backbone –random-transform –weights weights/ResNet-50-model.keras.h5 –batch-size 16 –steps 1000 –epochs 10 csv csv/annotations.csv csv/classes.csv

Вона має такі параметри:

- batch-size – визначає кількість зображень, які будуть пропускатися через мережу;
- epochs – кількість епох навчання – одна епоха відповідає тому, що весь датасет пропускається через мережу вперед і назад;
- steps – кількість кроків (кількість ітерацій пропускання зображень) в кожній епосі.

Результатом навчання є файл моделі з розширення .m5, який можна використовувати в додатку. Також існує спеціальний інструментарій для трансформації keras моделей в інші формати, такі як TensorFlow та ONNX.

Така модель може виконуватися як на центральному процесорі, так і на графічному прискорювачі. Це залежить від типу бібліотеки TensorFlow, яка виконується в додатку: для виконання моделей на CPU необхідно підключити пакет tensorflow, для виконання на GPU – пакет tensorflow-gpu.

## 7.2 Інтеграція моделі машиного навчання у сервіс

Розроблювана система ставить перед сервісом розпізнавання зображень наступні вимоги:

- сервіс має працювати на віртуальній машині в Azure;
- сервіс повинен мати HTTP інтерфейс, що описаний в розділі 6.

Оскільки визначено, що переважна кількість моделей працюють у середовищі Python, буде зручно реалізувати HTTP інтерфейс за допомогою цієї технології. Для

цього необхідно обрати бібліотеку або фреймворк для забезпечення роботи у якості веб-сервера. Найпопулярнішими рішеннями є:

- Django – комплексний багатофункціональний веб-фреймворк;
- Flask – мікрофреймворк для побудови API та вебсайтів;

Оскільки сервіс не є просунутим веб-додатком, а лише використовує базовий функціонал обміну через HTTP, був обраний мікрофреймворк Flask, завдяки своїй простоті та підтримці спільнотою.

Сам додаток відповідає за логіку взаємодії по HTTP на основі необхідного інтерфейсу та за виконання роботи моделі.

Постає проблема автоматизованого запуску сервісу під час запуску віртуальної машини. Сам по собі Flask додаток не може запускатися, але це можуть робити сервіси ініціалізації системи. Оскільки віртуальна машина має платформу Linux, то для нас доступно багато можливостей тонкого налаштування запуску системи.

Першим менеджером запуску є init. Init – це deamon-процес, який починає свою роботу, як тільки запускається система і продовжує працювати до завершення роботи. Фактично init – це перший процес, який запускається при завантаженні операційної системи, що робить його батьком всіх інших запущених процесів прямо або побічно, і тому зазвичай йому присвоюється «pid=1». Якщо якимось чином init-deamon не зміг запуститися, процес не буде запущений, і система досягне стадії, що називається «панікою ядра» (kernel panic).

Але набільш актуальним менеджером запуску є система systemd. Подібно init, systemd є батьком всіх інших процесів, що прямо чи побічно і є першим процесом, який запускається при завантаженні, тому зазвичай призначається «pid=1». Він був розроблений для подолання недоліків init. Він сам є фоновим процесом, який призначений для паралельного запуску процесів, що дозволяє скоротити час завантаження і обчислювальні витрати. Він має багато інших функцій в порівнянні з init. Основними перевагами є:

- сучасний і ефективний інтерфейс;
- більш простий процес завантаження;
- конкурентний і паралельний запуск процесів при завантаженні;

- інструкція ініціалізації процесів написана у файлі конфігурації, а не в системному скрипті;
- планування завдань з використанням календарних таймерів systemd;
- ведення журналу подій за допомогою journald.

Враховуючи численні переваги було прийняте рішення щодо використання сервісу systemd.

Були визначені та формалізовані (у форматі bash скриптів) дії, які необхідні для налаштування середовища віртульної машини для автоматичного запуску сервісу розпізнавання (таблиця 7.1).

Таблиця 7.1 – Сценарій налаштування віртуальної машини

| Назва дії | Фрагмент bash сценарію |
|---|---|
| Обновлення стандартних пакетів | - sudo apt-get update -y<br>- sudo apt-get install -y software-properties-common unzip |
| Встановлення Python 3.6 | - sudo add-apt-repository -y ppa:deadsnakes/ppa<br>- sudo apt-get update -y<br>- sudo apt-get install -y python3.6 python3-pip python3.6-dev build-essential libsm6 libxext6 libxrender1<br>- sudo -H python3.6 -m pip install --upgrade pip |
| Встановлення середовища виконання NVIDIA CUDA | - sudo apt-key adv --fetch-keys http://developer.download.nvidia.com/compute /cuda/repos/ubuntu1604/x86_64/7fa2af80.pub<br>- wget http://developer.download.nvidia.com/compute /cuda/repos/ubuntu1604/x86_64/cuda-repo-ubuntu1604_9.1.85-1_amd64.deb<br>- sudo apt install -y ./cuda-repo-ubuntu1604_9.1.85-1_amd64.deb |

Продовження таблиці 7.1

| Назва дії | Фрагмент bash сценарію |
|---|---|
| Встановлення середовища виконання NVIDIA CUDA (продовження) | - wget http://developer.download.nvidia.com/compute/machine-learning/repos/ubuntu1604/x86_64/nvidia-machine-learning-repo-ubuntu1604_1.0.0-1_amd64.deb<br>- sudo apt install -y ./nvidia-machine-learning-repo-ubuntu1604_1.0.0-1_amd64.deb<br>- sudo apt update -y<br>- sudo apt install -y cuda9.0 cuda-cublas-9-0 cuda-cufft-9-0 cuda-curand-9-0 cuda-cusolver-9-0 cuda-cusparse-9-0 libcudnn7=7.2.1.38-1+cuda9.0 libnccl2=2.2.13-1+cuda9.0 cuda-command-line-tools-9-0 |
| Клонування коду проекту | - mkdir recognition<br>- cd recognition<br>- git clone [repository URL]<br>- cd SVC |
| Встановлення залежностей проекту | - python3.6 -m pip install setuptools==39.0.1 –user<br>- python3.6 -m pip install -r requirements.txt –user<br>- python3.6 -m pip install numpy --user |
| Встановлення бібліотеки Keras | - curl https://raw.githubusercontent.com/fizyr/keras-retinanet/c4c738d/setup.py -o setup.py<br>- python3.6 setup.py build_ext –inplace<br>- python3.6 setup.py install --user |
| Налаштування переадресації запитів | - sudo iptables -t nat -A PREROUTING -p tcp --dport 80 -j REDIRECT --to-port 8080 |

Продовження таблиці 7.1

| Назва дії | Фрагмент bash сценарію |
|---|---|
| Налаштування автозапуску | - sudo bash -c 'cat > /etc/systemd/system/recognition-server.service <<- EOF<br>[Unit]<br>Description=RecognitionWebServer<br>Wants=network.target<br>After=syslog.target network-online.target<br>[Service]<br>User=smartsvc<br>Type=simple<br>WorkingDirectory=$PWD/<br>ExecStart=/usr/bin/python3.6 app.py<br>Restart=on-failure<br>RestartSec=10<br>KillMode=process<br>[Install]<br>WantedBy=multi-user.target<br>EOF'<br><br>- sudo systemctl daemon-reload<br>- sudo systemctl enable recognition-server<br>- sudo systemctl start recognition-server |

Таким чином середовище стає відтворюваним можливо повністю відтворити середовище на новій віртуальній машині у випадку необхідності її перерозгортання.

Для можливості програмного управління віртуальною машиною, як і іншими ресурсами Azure, необхідно створити Service Principal [14](сервісного користувача) у відповідній Azure Active Directory та надати йому доступ до ресурсної групи системи. Service Principal – об’єкт Active Directory, який дозволяє робити певні дії в

середовищах Azure Active Directory та Microsoft 365. Для забезпечення можливості програмної взаємодії з порталом Azure необхідні наступні дані:

- TenantId – ідентификатор директорії Azure;
- ClientId – ідентифікатор створеного Service Principal;
- ClientSecret – ключ доступу до Service Principal, що потрібно згенерувати на порталі;
- SubscriptionId – ідентифікатор підписки Azure;
- VirtualMachineResourceGroupName – назва ресурсної групи, де розміщується система;
- VirtualMachineName – назва віртуальної машини.

Дані параметри використовуються функцією StartRecognition (див. таблицю 6.6 розділу 6).

Під час обернення логіки виконання моделі у вебсервіс можлива проблема розділення ресурсів – оскільки контекст виконання моделі є спільним на весь вебсервіс, паралельні запити до сервісу призводять до зависання сервісу та припинення прийому HTTP запитів. Це вирішується усуненням конкурентного доступу до сервісу через зменшення кількості паралельних обробок повідомлень функції SendToRecognition. За замовчуванням, одна функція може оброблювати 32 повідомлення паралельно, але це значення змінюється через налаштування функціонального додатку.

# 8 ТЕСТУВАННЯ СИСТЕМИ

Тестування програмного забезпечення – це процес випробовування та перевірки прозграмного продукту на якість. Основна мета процесу полягає у перевірках відповідності між фактичною поведінкою програми і очікуваною поведінкою за допомогою кінцевого набору тестів.

Модульне, або юніт-тестування – це процес тестування, який дає змогу перевірити правильність роботи окремих модулів вихідного коду програми. Ідея полягає в тому, щоб написати тест для кожної функції або методу. Це дозволяє виявляти і виправляти помилки на ранніх етапах розробки модулю, до початку роботи на інтеграцією окремого модуля з іншими. Також юніт-тести дозволяють швидко перевіряти, чи не спричинила чергова модифікація кодової бази програми регресію, тобто виникнення помилки у вже перевіреному місці програми.

В ході комплексного інтеграційного тестування були об'єднані окремі програмні модулі і протестована їх вхаємодія. Зазвичай це відбувається після модульних тестів. Інтеграційне тестування засвідчує або спростовує факт, що компоненти ПЗ, коли зібрані до групи, функціонують корректно.

Наскрізне тестування проводиться на повній, інтегрованій системі з метою перевірити відповідність системи вимогам. Метою наскрізного тестування є засвідчити виконання функціональниг вимог та сценаріїв взаємодії.

Сценарії тестування використовуються для документування опису послідовності дій у системі та очікуваної поведінки. На основі сценаріїв використання були розроблені високопріоритетні сценарії тестування (див. таблиці 8.1 – 8.5)

Таблиця 8.1 – Сценарій тестування «Імпорт даних у систему – правильний формат, кілька зображень»

| | |
|---|---|
| Назва тесту | Імпорт даних у систему – правильний формат, кілька зображень |
| Ідентифікатор | TC1 |

Продовження таблиці 8.1

| | |
|---|---|
| Назва тесту | Імпорт даних у систему – правильний формат, кілька зображень |
| Короткий опис | Користувач імпортує зображення в систему у правильному форматі |
| Передумови | Точка інтеграції імпорту налаштована |
| Тестові кроки | - користувач формує вхідне повідомлення і передає його до системи. |
| Тестові дані | Повідомлення має правильний формат та містить кілька зображень |
| Очікувані результати | Система на запит відповідає успіхом. Зображення збережені у сховищі, вхідні метадані збережені у сховищі разом з посиланням на зображення. Зображення поміщені у чергу до механізму розпізнавання об’єктів. |
| Отримані результати | Система на запит відповідає успіхом. Зображення збережені у сховищі, вхідні метадані збережені у сховищі разом з посиланням на зображення. Зображення поміщені у чергу до механізму розпізнавання об’єктів. |
| Статус сценарію | Успіх |

Таблиця 8.2 – Сценарій тестування «Імпорт даних у систему – правильний формат, без зображень»

| | |
|---|---|
| Назва тесту | Імпорт даних у систему – неправильний формат |
| Ідентифікатор | TC2 |
| Короткий опис | Користувач надсилає повідомлення для імпорту даних у правильному форматі, але не передає зображень |
| Передумови | Точка інтеграції імпорту налаштована |
| Тестові кроки | - користувач формує вхідне повідомлення і передає його до системи. |

Продовження таблиці 8.2

| | |
|---|---|
| Назва тесту | Імпорт даних у систему – неправильний формат |
| Тестові дані | Повідомлення має правильний формат, але не містить зображень |
| Очікувані результати | Система на запит відповідає успіхом. Внутрішній стан системи не змінюється |
| Отримані результати | Система на запит відповідає успіхом. Внутрішній стан системи не змінюється |
| Статус | Успіх |

Таблиця 8.3 – Сценарій тестування «Імпорт даних у систему – неправильний формат»

| | |
|---|---|
| Назва тесту | Імпорт даних у систему – неправильний формат |
| Ідентифікатор | ТС3 |
| Короткий опис | Користувач надсилає неправильно сформоване повідомлення для імпорту даних |
| Передумови | Точка інтеграції імпорту налаштована |
| Тестові кроки | - користувач формує вхідне повідомлення і передає його до системи. |
| Тестові дані | Повідомлення має неправильний формат |
| Очікувані результати | Система на запит відповідає помилкою. Внутрішній стан системи не змінюється |
| Отримані результати | Система на запит відповідає помилкою. Внутрішній стан системи не змінюється |
| Статус | Успіх |

Таблиця 8.4 – Сценарій тестування «Запуск процесу розпізнавання об’єктів за вимогою – відсутність зображень в черзі»

| | |
|---|---|
| Назва тесту | Запуск процесу розпізнавання об’єктів за вимогою – відсутність зображень в черзі |
| Ідентифікатор | TC4 |
| Короткий опис | Користувач запускає процес розпізнавання об’єктів, але у вхідній черзі відсутні зображення для розпізнавання. |
| Передумови | У вхідній черзі відсутні зображення для розпізнавання. |
| Тестові кроки | - користувач переходить до адмінпанелі функції керування зображень;<br>- користувач запускає функцію запуску процесу розпізнавання. |
| Тестові дані | - |
| Очікувані результати | Функція успішно запускається і відразу зупиняється.<br>Механізм розпізнавання зображень не запускається.<br>Внутрішній стан системи не змінюється. |
| Отримані результати | Функція успішно запускається і відразу зупиняється.<br>Механізм розпізнавання зображень не запускається.<br>Внутрішній стан системи не змінюється. |
| Статус | Успіх |

Таблиця 8.5 – Опис сценарію використання «Запуск процесу розпізнавання об’єктів за вимогою – відсутність зображень в черзі»

| | |
|---|---|
| Назва тесту | Запуск процесу розпізнавання об’єктів за вимогою – наявні зображень в черзі |
| Ідентифікатор | TC5 |
| Короткий опис | Користувач запускає процес розпізнавання об’єктів, і у вхідній черзі наявні зображення для розпізнавання. |
| Передумови | У вхідній черзі відсутні зображення для розпізнавання. |

Продовження таблиці 8.5

| | |
|---|---|
| Назва тесту | Запуск процесу розпізнавання об’єктів за вимогою – наявні зображень в черзі |
| Тестові кроки | - користувач переходить до адмінпанелі функції керування зображень;<br>- користувач запускає функцію запуску процесу розпізнавання. |
| Тестові дані | - |
| Очікувані результати | Функція успішно запускається та пересилає вхідні повідомлення в чергу до сервісу розпізнавання.<br>Механізм розпізнавання зображень є запущеним. |
| Отримані результати | Функція успішно запускається та пересилає вхідні повідомлення в чергу до сервісу розпізнавання.<br>Механізм розпізнавання зображень є запущеним. |
| Статус | Успіх |

На основі описаних тестових сценаріїв та сценаріїв використання розроблена матриця відповідності вимогам (Requirements Traceability Matrix). Матриця описана в таблиці 8.6.

Таблиця 8.6 – Матриця відповідності вимогам

| № сценарію використання | Опис сценарію використання | № сценаріїв тестування | Статус |
|---|---|---|---|
| | | TC1 | успіх |
| | | TC2 | успіх |
| | | TC3 | успіх |
| UC2 | Запуск процесу розпізнавання об’єктів за розкладом | - | |

Продовження таблиці 8.6

| № сценарію використання | Опис сценарію використання | № сценаріїв тестування | Статус |
|---|---|---|---|
| | | TC4 | успіх |
| | | TC5 | успіх |
| UC4 | Експорт даних з системи за вимогою | - | - |
| UC5 | Постійний експорт даних з системи | - | - |
| UC6 | Отримання аналітики по використанню ресурсів та об’ємах даних | - | - |
| UC7 | Керування регламентом автоматичного запуску процесу розпізнавання | - | - |

Сформована матриця відповідності вимогам показує, що не для усіх сценаріїв використання описано сценарії тестування. Сценарії використання UC1 та UC3 повністю покриті тест-кейсами, інші ж не мають жодного відповідного тестового сценарію.

# 9 РОЗРОБЛЕННЯ СТАРТАП-ПРОЕКТУ

Стартап – це тимчасова структура направлена на пошук і реалізацію масштабованої бізнес ідеї – ідеї, яка може бути використана для побудови нової компанії чи нового напрямку діяльності у вже працюючій компанії з метою виробництва бізнес продуктів.

Бізнес продукт - це товар чи послуга, які можна запропонувати на ринку для реалізації і які будуть задовольняти потреби споживачів. Бізнес ідея відповідає на питання «Як?», тобто описує або логічно описує як організація може створити бізнес продукт і представити даний бізнес продукт на ринку з метою отримання прибутку.

## 9.1 Опис ідеї проекту

Таблиця 9.1 – Опис ідеї стартап-проекту

| № | Зміст ідеї | Напрямки використання | Вигоди для користувача |
|---|---|---|---|
| 1. | Автоматизація керування процесом розпізнавання зображень | 1. Інтеграція моделей машинного навчання з іншими системами | Оптимізація процесу взаємодії користувачів з процесами, що виконуються в моделі машинного навчання, яка використовується для обробки візуальних даних |

Основними техніко-економічними характеристиками ідеї є:

- швидка та якісна інтеграція з іншими системами;
- автоматизація взаємодії різних архітектур ML моделей;
- використання системи як хмарного сервісу;
- робота з власними ML моделями користувачів;
- гнучкість та висока модульність системи;
- можливість адаптації системи під потреби конкретного користувача;
- економічна ефективність роботи ML моделей.

Згідно до розділу 1, основними найближчими за функціональністю конкурентами є сервіси Azure Custom Vision та IBM Visual Recognition. Для визначенню. Для визначення сильних, нейтральних та слабких характеристик системи, було здійснено збір та аналіз інформації щодо значень техніко-економічних показників для ідеї власного проекту та проектів-конкурентів відповідно до визначеного вище переліку. Таблиця 9.2 містить порівняння даної системи із вищевказаними конкурентами. В ній стовпчик відповідає W відповідає за слабку сторону, N – за нейтральну сторону та S – за сильна сторону.

Таблиця 9.2 – Визначення сильних, слабких та нейтральних характеристик ідеї проекту

| № | Техніко-економічні характеристики ідеї | Потенційні товари/концепції конкурентів | | | W | N | S |
|---|---|---|---|---|---|---|---|
| | | Мій проект | Azure Custom Vision | IBM Visual Recognition | | | |
| 1. | Швидка та якісна інтеграція з іншими системами | Має | Має | Має | - | + | - |
| 2. | Автоматизація взаємодії різних архітектур ML моделей; | Має | Немає | Немає | - | - | + |
| 3. | Використання системи як хмарного сервісу | Має | Має | Має | - | + | - |
| 4. | Робота з власними ML моделями користувачів | Має | Немає | Немає | - | - | + |
| 5. | Гнучкість та висока модульність системи | Має | Немає | Немає | - | - | + |
| 6. | Можливість адаптації | Має | Немає | Немає | - | - | + |

| | | | | | | | |
|---|---|---|---|---|---|---|---|
| | системи під потреби | | | | | | |

Продовження таблиці 9.2

| № | Техніко-економічні характеристики ідеї | Потенційні товари/концепції конкурентів | | | W | N | S |
|---|---|---|---|---|---|---|---|
| | | Мій проект | Azure Custom Vision | IBM Visual Recognition | | | |
| 6 | конкретного користувача | | | | | | |
| 7. | Економічна ефективність роботи ML моделей | Має | Немає | Немає | - | - | + |

Отже, проект може бути успішним для реалізації на ринку, оскільки він займає вузьку, але потребовану нішу систем автоматизації взаємодії з ML моделями.

## 9.2 Технологічний аудит ідеї проекту

Таблиця 9.3 – технологічна здійсненність ідеї проекту

| № | Ідея проекту | Технології її реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 1. | Швидка та якісна інтеграція з іншими системами | Azure Service Bus, Azure Functions | Наявна | Доступна |
| 2. | Автоматизація взаємодії різних архітектур ML моделей; | Azure Virtual Machine | Наявна | Доступна |

| | | | | |
|---|---|---|---|---|
| 3. | Використання системи як | Хмарне середовище | Наявна | Доступна |

Продовження таблиці 9.3

| № | Ідея проекту | Технології її реалізації | Наявність технологій | Доступність технологій |
|---|---|---|---|---|
| 3. | хмарного сервісу | Microsoft Azure | | |
| 4. | Робота з власними ML моделями користувачів | Azure Virtual Machine | Наявна | Доступна |
| 5. | Гнучкість та висока модульність системи | Azure Service Bus, Azure Functions | Наявна | Доступна |
| 6. | Можливість адаптації системи під потреби конкретного користувача | Azure Functions | Наявна | Доступна |
| Обрана технологія реалізації ідеї проекту: є можливою. | | | | |

Згідно з проведеним дослідженням перераховані технології є легкодоступними до використання та побудови системи, методи реалізації є доступними, більшість технологій є платними.

## 9.3 Аналіз ринкових можливостей запуску стартап-проекту

Визначення ринкових можливостей та загроз є наступним етапом запуску стартап-проекту. Ринкові можливості можна використовуват під час ринкового впровадження проекту, а ринкові ризики навпаки – можуть перешкодити реалізації проекту на ринку. Для запуску стартап-проекту у ринок, необхідно визначити стан ринку, визначити його загрози, спланувати напрями розвитку, потреби потенційних клієнтів та пропозицій конкурентів. Інноваційна діяльність, в порівнянні з іншими

видами діяльності, є у більшій міра пов'язана з ризиком, як повної гарантії позитивного результату практично немає. В результаті інноваційні проекти більшою мірою залежать від факторів невизначеності, які є причиною виникнення ризиків. Потенціальна характеристика ринку зображена в таблиці 9.4.

Таблиця 9.4 – Попередня характеристика потенційного ринку стартап-проекту

| № | Показники ринку (найменування) | Характеристика |
|---|---|---|
| 1. | Кількість головних гравців, од | 1 (розробники) |
| 2. | Загальний обсяг продаж, грн/ум. од. | Понад 5000 |
| 3. | Динаміка ринку (якісна ціна) | Зростає |
| 4. | Наявність обмежень для входу | Конкуренція |
| 5. | Специфічні вимоги до стандартизації та сертифікації | Відсутні |
| 6. | Середня норма рентабельності в галузі (або по ринку), % | 140% |

У результаті проведення дослідження було визначено що норма рентабельності є задовільною, динаміка ринку є позитивною в зв'язку з розвитком середнього та великого бізнесу, а ринок є привабливим для входження.

Наступним кроком необхідно характеристики потенційних клієнтів. Характеристика клієнтів описана у таблиці 9.5.

Таблиця 9.5 – Характеристика потенційних клієнтів стартап-проекту

| № | Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
|---|---|---|---|---|
| 1. | Потреба в економії | Середній та | Клієнтами | - можливість |

| | | | | |
|---|---|---|---|---|
| | ресурсів для взаємодії з | великий бізнес у сферах | відрізняються призначеннями | використовувати власну модель |

Продовження таблиці 9.5

| № | Потреба, що формує ринок | Цільова аудиторія (цільові сегменти ринку) | Відмінності у поведінці різних потенційних цільових груп клієнтів | Вимоги споживачів до товару |
|---|---|---|---|---|
| | моделями машиного навчання | роздрібної торгівлі | та архітектурами ML моделі | - простота інтеграції з іншими системами |

Аналіз потенційних груп клієнтів встановлює те, що основна частина клієнтів – це підприємства середнього та великого бізнесу у сферах роздрібної торгівлі. Тому зосередитись важливо на їх основних вимогах, можливість викоритовувати власну модель і мати можливість до простої інтеграції з іншими системами. Після визначення вимог клієнтів необхідно перевірити кон'юнктуру ринку і врахувати фактори, що сприяють реалізації проекту на ринку (таблиця 9.6), а також фактори, що перешкоджають цьому (таблиця 9.7).

Таблиця 9.6 – Фактори можливостей стартап-проекту

| № | Фактор | Зміст можливості | Можлива реакція компанії |
|---|---|---|---|
| 1. | Прийняття клієнтами нових технологій для оптимізації бізнесу процесів | Нові клієнти, розширення ринку | Масштабування та просування продукту на ринку |
| 2. | Висока вартість використання готових рішень | Нові клієнти, розширення ринку | Масштабування та просування продукту на ринку |

| | | | |
|---|---|---|---|
| | конкурентів | | |

Таблиця 9.7 – Фактори загроз стартап-проекту

| № | Фактор | Зміст загрози | Можлива реакція компанії |
|---|---|---|---|
| 1. | Розширення функціоналу систем конкурентів | Є сильна залежність від постачальників складових даної системи | Аналіз вимог користувачів та розширення функціоналу власної системи |
| 2 | Поява нової системи-конкурента на ринку | Поява конкурентів | Аналіз вимог користувачів та розширення функціоналу власної системи |
| 3. | Зміна цінової політики постачальника | Можлива нерентабельність у випадку підвищення цін за використання сервісів хмарного провайдеру Azure | Адаптація цінової політики для клієнтів проекту |
| 4. | Економічний спад | Відсутність нових клієнтів через те, що оптимізація бізнес процесів матиме низький пріоритет | Зсув пріоритетів на збереження клієнтської бази |
| 5. | Мала кількість продажів | Прогнозовані продажі не виправдались | Пошук нових можливостей покращення продукту, збирання та дослідження відгуків користувачів |

Основні встановлені фактори загроз пов’язані з розвитком та появою нових

конкурентів, на що потрібно реагувати аналізом вимог користувачів та адаптацією системи під ці вимоги. Встановлено, що інші зазначені загрози – не є критичними, та мають низькі шанси на їх виникнення. Сприятливі можливості до ринкового впровадження, призводять до зростання попиту і відповідно масштабування проекту. Можливості якісно значно переважають ризики та загрози, а отже проект має потенціал до розвитку в ринковому сегменті. Наступним кроком є аналіз існуючої конкуренції, шо описаний у таблиці 9.8.

Таблиця 9.8 – Ступеневий аналіз конкуренції на ринку

| Особливості конкурентного середовища | В чому проявляється дана характеристика | Вплив діяльності підприємства |
|---|---|---|
| Тип конкуренції: олігополія | Конкурентами є домінуючі в області компанії | Виготовлення продукту, що відрізняється нижчою ціною та більш конкретним призначенням |
| За рівнем конкурентної боротьби: світовий | Ринок охоплює всю планету | Забезпечення відповідності продукту стандартам різних країн |
| За галузевою ознакою: внутрішньо-галузева | Конкуренти знаходяться в одній галузі (хмарні сервіси штучного інтелекту) | Аналіз факторів, які впливають на успіх у даній галузі |
| Конкуренція за видами товарів: товарно-родова | Товари конкурентів виконують схожі функції, хоча основне призначання – інше. | Продукт має бути кращим у галузі |
| За характером конкурентних переваг: не цінова | Товар якісніше та більшій мірі задовольняє вимоги клієнтів у порівнянні з | Вкладання ресурсів у підтримання вищої якості адаптацію продукту |

| | | |
|---|---|---|
| | конкурентами. | |
| За інтенсивністю: не марочна | Товар тільки виходить на ринок | Забезпечити успішний старт продукту на ринку |

Завдяки ступеневому аналізу конкуренції на ринку, можна зробити висновок, ринок є перспективним, незважаючи на те що конкурентні товари утримують позиції на ринку. Товари конкурентів не задовольняють потреби користувачів у повній мірі, завдяки чому проект може укріплювати позиції на ринку. Більш детальний аналіз умов конкуренції в галузі описаний в таблиці 8.9.

Таблиця 9.9 – Аналіз конкуренції в галузі за М. Портером

| Складові аналізу | Прямі конкуренти в галузі | Потецнійні конкуренти | Постачальники | Клієнти | Товари-замінники |
|---|---|---|---|---|---|
| | Хмарні сервіси штучного інтелекту з можливістю адаптації під вимоги клієнтів | Конкурентом може стати будь-який продукт що буде задовольняти потребу автоматизації взаємодії з власними моделями кліентів | Єдиним постачальником є Microsoft Azure | Середній та великий бізнес у сферах роздрібної торгівлі | Наявність на ринку конкурентів, але прямих аналогів замінників на даний момент не існує |
| Висновки | Інтенсивність конкурентної боротьби | Потенційні конкуренти можуть виникнути згодом | Існує сильна залежність від постачальни | Основними вимогами клієнтів є простота інтеграції та | Обмежень через товари-замінники немає |

| | | | | | |
|---|---|---|---|---|---|
| | доволі висока | | ків, проте вони не диктують умови роботи ринку. | можливість використання власних моделей | |

Аналіз за М. Портером визначив, що на даний час існують лише прямі конкуренти та існує можливість виникнення нових. Товарів-замінників не існує. Конкурентна ситуація на ринку є сприятливою, проект може мати успіх. Наступним кроком необхідно встановити ключові сторони котрі повинен мати проект, для того аби бути конкурентоспроможним. Опис обґрунтування факторів конкурентоспроможності описано в таблиці 9.10.

Таблиця 9.10 – Обґрунтування факторів конкурентоспроможності

| № | Фактор конкурентоспроможності | Обґрунтування факторів |
|---|---|---|
| 1. | Лояльна цінова політика | Вартість товару є нижчою за конкурентів |
| 2. | Здатність до підвищення ефективності процесів клієнта | Система збільшує ефективність бізнес-процесів користувача |
| 3. | Гнучкість та адаптивність | Система може адаптуватися під додаткові вимоги окремих користувачів |
| 4. | Комплексний підхід | Поєднання усього функціоналу в одному місці покращує взаємодію користувача з системою |
| 5. | Можливість розширення продукту | Продукт передбачає можливість розширення, що полягає у створенні інтеграцій для найбільш типових систем у сфері |

Щоб мати успіх в умовах сучасної ринкової ситуації, були обрані фактори, на яких буде заснована його перевага проекту над конкурентами. Найважливішими серед факторів конкурентоспроможності є лояльна цінова політика, здатність до підвищення ефективності процесів клієнта та можливості розширення продукту шляхом створення інтеграцій для найбільш типових систем у сфері. Порівняння факторів конкурентоспроможності дає можливість визначити сильні та слабкі сторони стартап-проекту (таблиця 8.11). Скоротимо позначення конкурентів: Microsoft Azure Custom Vision – Azure, IBM Visual Recognition – IBM.

Таблиця 9.11 – Порівняльний аналіз сильних і слабких сторін стартап-проекту

| Фактор конкурентоспроможності | Бали 1-20 | Рейтинг товарів-конкурентів порівняно з проектом | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | | -3 | -2 | -1 | 0 | +1 | +2 | +3 |
| Лояльна цінова політика | 15 | IBM | | Azure | | | | |
| Здатність до підвищення ефективності процесів клієнта | 10 | | | | Azure IBM | | | |
| Гнучкість та адаптивність | 12 | | | Azure IBM | | | | |
| Комплексний підхід | 5 | | | | | IBM | Azure | |
| Можливість розширення продукту | 14 | | Azure IBM | | | | | |

Порівняльний аналіз демонструє, що за факторами конкурентоспроможності проект має значні переваги над конкурентами і тому може мати успіх на ринку. Для повноти аналізу, необхідно провести SWOT-аналіз на основі ринкових можливостей, загроз, сильних та слабких сторін (таблиця 9.12). Переваги SWOT-аналізу полягають

в тому, що він дозволяє просто, в потрібному контексті розглянути на положення компанії, продукту або послуги в галузі, і тому є найпопулярнішим інструментом в управлінні ризиками і прийнятті управлінських рішень. Результати SWOT аналізу можно використати для складення списку дій, як потрібно виконати для успішного просунення проекту. Основна ідеє полягає в в аналізі сильних сторонів проекту і з'ясування, як можна використовувати ці сильні сторони щоб скористатися визначеними можливостями та боротися з загрозами на ринку.

Таблиця 9.12 – SWOT-аналіз стартап-проекту

| | |
|---|---|
| Сильні сторони:<br>- функції, що відсутні у конкурентів;<br>- найбільш повне задоволення вимог клієнтів;<br>- висока ефективність впровадження проекту для конкретного клієнту<br>- актуальність проекту; | Слабкі сторони:<br>- не найсприятливіше ринкове середовище;<br>- складність масштабного впровадження;<br>- сильна залежність від постачальників;<br>- вузька ніша. |
| Можливості:<br>- поява крупних клієнтів – інвесторів;<br>- зростання попиту на продукцію;<br>- вихід на нові ринки або сегменти.<br>- розширення ринку в наслідок зацікавленості в продукті;<br>- розвиток технологій. | Загрози:<br>- проблеми з постачальниками;<br>- поява нової системи-конкурента на ринку;<br>- мала кількість продажів; |

SWOT-аналіз дав змогу комплексно оцінити і порівняти основні сторони проекту та ринку. На основі SWOT-аналізу були розроблені розроблені альтернативи ринкової поведінки для виведення стартап-проекту на ринок (таблиця 9.13).

Таблиця 9.13 – Альтернативи ринкового впровадження

| Альтернатива (орієнтовний | Ймовірність отримання | Строки реалізації |
|---|---|---|

| | | |
|---|---|---|
| комплекс заходів) ринкової поведінки | ресурсів | |
| Наступник | Значна | 1-2 роки |
| Загарбник | Можлива | Більше 2 років |
| Виклик лідеру | Мала | 1-2 роки |

Для стартап-проекту були обрані три альтернативи ринкової поведінки. Використання комплекса заходів «Наступник» дозволить спеціалізуватися на одній частині ринку, а потім поглиблювати свій вплив. «Виклик лідеру» має малу ймовірність отримання ресурсів. Іншою альтернативою є «Загарбник», який орієнтований на захоплення ринку від конкурентів. Але найкраще за строком реалізації та ймовірністю отримання ресурсів підходить стратегія «Наступник».

## 9.4 Розроблення ринкової стратегії проекту

Для реалізації проекту на ринку, необхідно виокремити положення, що коротко описують потенційних споживачів, цільовий ринок, спосіб позиціонування товару і величини обсягів продажу, яких планується досягнути за перші кілька років реалізації товару. Першим кроком визначення стратегії охоплення ринку є вибір цільових груп потенційних споживачів (таблиця 9.14).

Таблиця 9.14 – Вибір цільових груп потенційних споживачів

| Опис профілю цільової групи потенційних клієнтів | Готовність споживачів сприйняти продукт | Орієнтований попит в межах цільової групи (сегменту) | Інтенсивність конкуренції в сегменті | Простота входу в сегмент |
|---|---|---|---|---|
| Великий бізнес у сфері роздрібної торгівлі | Середня | Високий | Висока | Середня |

| | | | | |
|---|---|---|---|---|
| Середній бізнес у сфері роздрібної торгівлі | Висока | Середній | Середня | Мала |
| Які цільові групи обрано: середній та великий бізнес у сфері роздрібної торгівлі | | | | |

Оскільки потенційні клієнти належать до одного сегменту ринку, доцільним є обрання стратегії концентрованого маркетингу.

Наступний крок розробки ринкової стратегії полягає в визначенні базової стратегії розвитку (таблиця 9.15).

Таблиця 9.15 – Визначення базової стратегії розвитку

| Обрана альтернатива розвитку проекту | Стратегія охоплення ринку | Ключові конкурентоспроможні позиції відповідно до обраної альтернативи | Базова стратегія розвитку |
|---|---|---|---|
| Наступник | Концентрація на потребах великого сегменту ринку | Надання клієнтам програмного продукту з унікальним функціоналом | Стратегія спеціалізації |

Для першого періоду просування стартапу була обрана стратегія спеціалізації. Ця стратегія передбачає зосередження зусиль на потребах одного сегмента без охоплення всього ринку. Стартап-проект має на меті випускати продук кращої якості з унікальним функціоналом.

Наступним етапом процесу розробки ринкової стратегії є вибір стратегії конкурентної поведінки (таблиця 9.16).

Таблиця 9.16 – Визначення базової стратегії конкурентної поведінки

| Чи є проект | Чи буде компанія | Чи буде компанія | Стратегія |
|---|---|---|---|

| | | | |
|---|---|---|---|
| «першопрохідцем» на ринку? | шукати нових споживачів, або забирати існуючих у конкурентів? | копіювати основні характеристики товару конкурента, і які? | конкурентної поведінки |
| Продукт є першопрохідцем у вузькому сегменті задоволення потреб клієнтів | В першу чергу проект має зайняти порожню нішу у наданні послуг автоматизації взаємодії з моделями ML | Компанія буде копіювати лише таку характеристику, як комплексний підхід до надання послуг | Наступник |

Отже, як базову стратегію конкурентної поведінки було обрано стратегію Наступник. Вона полягає у виборі адаптивної поведінки, позгоджуючи свої рішення з рішеннями, прийнятими конкурентами. Наступним кроком є визначення стратегії позиціонування (таблиця 9.17).

Таблиця 9.17 – Визначення стратегії позиціонування

| Вимоги до товару цільової аудиторії | Базова стратегія розвитку | Ключові конкурентоспро можні позиції власного стартап-проекту | Вибір асоціацій, які мають сформувати комплексну позицію власного проекту (три ключових) |
|---|---|---|---|
| Зручність інтеграції з іншгими системами, можливість роботи з клієнтськими ML моделями | Стратегія спеціалізації | Наступник | Ефективність бізнес-процесів, власна ML модель, інтеграція з будь-якою системою |

Таким чином, позиціонування продукту визначається на основі SWOT-аналізу, вибору цільових груп потенційних споживачів, визначення стратегії захоплення ринку та конкурентоспроможних переваг продукту.

## 9.5 Розроблення маркетингової програми стартап-проекту

Для того щоб успішно реалізувати проект у встановлені терміни, необхідно розробити маркетингову програму – систему взаємопов'язаних заходів щодо визначення дій виробника з усіх питань маркетингової кампанії на заданий період часу. Формування маркетингового плану базується на даних комплексного маркетингового дослідження, що дозволяє виявити поточні та майбутні потреби і потреби потенційних споживачів з урахуванням обраної стратегії і маркетингової стратегії.

Для початку потрібно сформувати маркетингову концепцію товару, для цього було визначено ключові переваги концепції потенційного товару (таблиця 9.18).

Таблиця 9.18 – Визначення ключових переваг концепції потенційного товару

| Потреба | Вигода, яку пропонує товар | Ключові переваги перед конкурентами (існуючі або такі, що потрібно створити) |
|---|---|---|
| Ефективність бізнес-процесів користувача | Зменшується кількість ресурсів для взаємодії ML моделі з даними | Більша ефективність бізнес-процесів після впровадження нашої системи, ніж систем конкурентів |
| Можливість використання ML моделі користувача | Швидкість впровадження товару | Можливість автоматизовувати роботу існуючих моделей |
| Простота інтеграції з іншими | Швидкість впровадження товару | Багато готових інтеграцій під типові системи в сфері ринку |

| | | |
|---|---|---|
| системами | | |

Ключовими перевагами концепції даного проекту є можливість повищувати ефективність бізнес-проців у більшій мірі, ніж конкуренти, велика швидкість впровадження та гнучкість системи.

Подальшим кроком є розробка трирівневої маркетингової моделі товару, для уточнення ідеї продукту, особливостей процесу надання товару, та його фізичних складових (таблиця 9.19).

Таблиця 9.19 – Опис трьох рівнів моделі товару

| Рівні товару | Сутність та складові | | |
|---|---|---|---|
| | Опис базової потреби споживача, яку задовольняє товар (згідно концепції), її основної функціональної вигоди | | |
| | Автоматизація управління процесом розпізнавання зображень | | |
| | Властивості/ характеристики | М/Нм | Вр/Тх/Тл/Е/ Ор |
| | 1. Економічні: вартість експлуатації залежить від реального використання системи.<br>2. Технологічні: використання сучасних технологій.<br>3. Безпека: дотримання безпеки зберігання даних за допомогою технологій Microsoft Azure.<br>4. Надійності: система повинна справно працювати по проходженню декількох років, гарантована підтримка – 1 рік.<br>5. Екологічні: відповідність нормативам | –/+ | +/+/+/+/+ |

| | | | |
|---|---|---|---|
| | використання електронних пристроїв та програмного забезпечення | | |
| | Марка: Smart SVC | | |
| | До продажу: представлення клієнтам продукту | | |
| | Після продажу: розгортання системи, інтеграція моделі у систему (за наявності) | | |
| За рахунок чого потенційний товар буде захищено від копіювання: захищення інтелектуальної власності шляхом патентування | | | |

Наступним кроком є визначення цінової межі, на яку потрібно орієнтуватися при встановленні ціни потенційного продукту (таблиця 9.20).

Таблиця 9.20 – Визначення меж встановлення ціни

| Рівень цін на товари-замінники | Рівень цін на товари-аналоги | Рівень доходів цільової групи споживачів | Верхня та нижня межі встановлення ціни на товар/послугу |
|---|---|---|---|
| 300-400 ум. од. | 400-600 ум. од. | 1000+ ум. од. | 250-300 ум. од. |

Ціна була визначена у допустимих межах для споживачів і нижчою ніж у аналогів. Ці дані допоможуть надалі, під час розповсюдження товару

Сформована систему збуту описана у таблиці 8.21

Таблиця 9.21 – Формування системи збуту

| Специфіка закупівельної поведінки цільових клієнтів | Функції збуту, які має виконувати постачальник товару | Глибина каналу збуту | Оптимальна система збуту |
|---|---|---|---|
| Покупка ліцензії програми в магазині AppSource[15] та Azure Marketplace[16] | Розгортання системи, інтеграція ML моделі у систему (за наявності) | Глибока | Сторонні посередники |

Для оптимальності процесу збуту було вирішено використовувати майданчики для продажу систем, що побудовані на основі хмарного середовища Azure.

Останнім кроком маркетингової програми є розроблення маркетингових комунікацій (таблиця 9.22).

Таблиця 9.22 – Концепція маркетингових комунікацій

| Специфіка поведінки цільових клієнтів | Канали комунікацій, якими користуються цільові клієнти | Ключові позиції, обрані для позиціонування | Завдання рекламного повідомлення | Концепція рекламного звернення |
|---|---|---|---|---|

Продовження таблиці 9.22

| | | | | |
|---|---|---|---|---|
| Серед клієнтів багато людей, що користуються середовищем Azure для автоматизації бізнес-процесів | Професійні соціальні мережі, конференції | Оптимізація процесу взаємодії користувачів з процесами, що виконуються в моделі машинного навчання, яка використовується для обробки візуальних даних | Донести користувачам інформацію про якісний і сучасний продукт з перевагами | Оригінально сповістити споживачів про новий продукт |

Основними каналами комунікаціями для користувачів є професійні соціальні мережі та конференції, і використання таких каналів є необхідним для вдалого розповсюдження інформації про товар.

## 9.6 Висновки до розділу

У розділі було визначено та описано можливість успішного виходу на ринок розроблюваної системи. Проведено аналіз самої ідеї та визначені ключові характеристики, що дозволять вийти та закріпитися реалізації цієї ідеї. Проведено аналіз конкурентів та визначена конкурентноспроможність ідеї, проведено аналіз можливих стратегій розвитку та маркетингу продукту. Був проведений аудит технологій, використовуваних при реалізації проекту.

Також було проаналізовано техніко-економічні характеристики ідеї, на підставі яких було прийнято рішення, наскільки проект буде відрізнятися від існуючих аналогів. Основними конкурентними перевагами є можливість автоматизації моделі машиного навчання, що представить користувач, та простота автоматизації продукту з іншими системами.

Був проведений аналіз ринкових можливостей для запуску стартап-проекту. Розглянуто характеристики потенційного ринку, визначено актуальний стан конкуренції у сфері та ринкової реалізації стартап-проекту. За результатами проведеного аналізу було визначено, що проект не має прямих аналогів, отже, він має високу конкурентоспроможність. Для проекту була обрана стратегія просування на ринку.

Проведено аналіз групи потенційних замовників для розробки ринкової стратегії проекту. У відповідності з визначеним сегментом ринку була обрана стратегія розвитку – спеціалізації, і стратегія конкурентної дії - наступник. Для проекту була розроблена маркетингова програма, в рамках якої був сформований цінова політика товару і вибраний оптимальний канал збуту та поширення товару.

# ВИСНОВКИ

У дипломному проекті була розроблена автоматизована система керування процесом розпізнавання зображень. Система є гнучкою, економічно та функціонально конкурентоспроможною і має можливість успішно вийти на ринок.

В ході виконання роботи над проектом був проведений аналіз предметної області систем машиного зору та проаналізовано існуючі рішення хмарних систем вказаного типу. За результатами аналізу були визначені основні критерії розвитку та актуальності проблеми. Найбільш важливим недоліком існуючих рішень, який виправляє проект системи, є відсутність можливості надання зручної інтеграції до моделі машиного навчання, що надав користувач.

Здійснено визначення та дослідження функціональних і нефункціональних вимог до системи, на основі них визначені ролі користувачів і сформульовані типові сценарії використання системи. Процеси для основних ролей – імпортера та експортера – виконуються системами, що інтегруються з розробленою системою. Окрім цього, передбачена взаємодія з людиною в рамках ролі адміністратора системи.

Була розроблена багаторівнева структура системи, описано кожен рівень системи та їх взаємодію один з одним. Система складається з чотирьох рівнів: рівень взаємодії з даними та імпорту/експорту, рівень збереження даних, рівень управління процесом розпізнавання зображень та рівень розпізнавання зображень.

Обрана платформа розробки логічних компонентів системи та визначені хмарні сервіси та технології для забезпечення роботи системи. Основною платформою є .NET Core, проект у значному обсязі базується на хмарних рішеннях Microsoft Azure – Cloud Function, Service Bus, Storage та інших.

Були розроблені функціональні модулі з основною логікою системи та визначено, як повність розмістити і налаштувати систему в хмарному середовищі Microsoft Azure. Також було описаний процес розробки типової моделі комп’ютерного зору, її оркестрація та інтеграція з іншими частинами системи.

На основі розробленої системи було проведено дослідження потенційного виходу на ринок як стартап-проекту. Проаналізовано ідею, що покладена у систему,

та визначено основні техніко-економічні характеристики що надають проекту конкурентноспроможності. Також були проаналізовані ринкові можливості щодо запуску проекту та на основі цього розроблені ринкова та маркетингова стратегія. Згідно до проведеного дослідження, система є конкурентоспроможною та здатною до виходу на ринок.

Подальшим напрямком роботи є розвиток проекту в напрямку «системи-як-сервіс», тобто створення комплексної хмарної системи комп’ютерного зору що може мати наступний функціонал:

- можливість створення власних ML моделей;
- можливість роботи з декількома різними архітекурами моделей одночасно;
- розробка функціоналу задач, коли вхідні дані логічно об’єднуються в задачу, що містить дані про тип/клас/версію моделі, необхідну постобробку даних, та інші дії, що забезпечать отримання бажаного результату.

# ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Perceptrons [Електронний ресурс] : Режим доступу: https://www.britannica.com/technology/perceptrons#ref1009913
2. .NET Core [Електронний ресурс] : Режим доступу: https://dotnet.microsoft.com
3. Round 18 results – TechEmpower Framework Benchmark [Електронний ресурс] : Режим доступу: https://www.techempower.com/benchmarks/#section=data-r18&hw=ph&test=plaintext
4. Azure SDK for .NET [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/en-us/dotnet/azure/dotnet-tools?view=azure-dotnet&tabs=windows
5. Keras: The Python Deep Learning library [Електронний ресурс] : Режим доступу: https://keras.io/
6. Service Bus Explorer [Електронний ресурс] : Режим доступу: https://github.com/paolosalvatori/ServiceBusExplorer
7. Azure Virtual Machines N-series [Електронний ресурс] : Режим доступу: https://azure.microsoft.com/en-us/pricing/details/virtual-machines/windows/#n-series
8. RFC822: Standard for ARPA Internet Text Messages [Електронний ресурс] : Режим доступу: https://www.w3.org/Protocols/rfc822/
9. Azure Function Core Tools [Електронний ресурс] : Режим доступу: https://github.com/Azure/azure-functions-core-tools
10. Azure Functions HTTP Triggers and Bindings [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/en-us/azure/azure-functions/functions-bindings-http-webhook?tabs=csharp#authorization-keys
11. Keras RetinaNet [Електронний ресурс] : Режим доступу: https://github.com/fizyr/keras-retinanet
12. Focal Loss for Dense Object Detection [Електронний ресурс] : Режим доступу: https://arxiv.org/abs/1708.02002
13. labelImg [Електронний ресурс] : Режим доступу: https://github.com/tzutalin/labelImg

14. Application and service principal objects in Azure Active Directory [Електронний ресурс] : Режим доступу: https://docs.microsoft.com/en-us/azure/active-directory/develop/app-objects-and-service-principals

15. Microsoft AppSource [Електронний ресурс] : Режим доступу: https://appsource.microsoft.com/en-us/

16. Azure Marketplace [Електронний ресурс] : Режим доступу: https://azuremarketplace.microsoft.com/en-us/marketplace/

# ДОДАТОК А

# ДОДАТОК Б

# ДОДАТОК В

# ДОДАТОК Г

# ДОДАТОК Д

# ДОДАТОК Е

# ДОДАТОК Ж

# ДОДАТОК И