ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
КАФЕДРА КОМП'ЮТЕРНОЇ ІНЖЕНЕРІЇ

КВАЛІФІКАЦІЙНА РОБОТА

на тему: «АРХІТЕКТУРА ІНТЕГРАЦІЇ ШІ КОМПОНЕНТІВ У РОЗПОДІЛЕНИХ СИСТЕМАХ»

на здобуття освітнього ступеня магістра
зі спеціальності 123 Комп'ютерної інженерії
(код, найменування спеціальності)
освітньо-професійної програми Комп'ютерні системи та мережі
(назва)

Кваліфікаційна робота містить результати власних досліджень. Використання ідей, результатів і текстів інших авторів мають посилання на відповідне джерело

_____ Денис ТУЖИЛІН
(підпис) *Ім'я, ПРІЗВИЩЕ здобувача*

Виконав: здобувач(ка) вищої освіти гр.
КСДМ-61 Денис ТУЖИЛІН
Ім'я, ПРІЗВИЩЕ
Керівник: Артем АНТОНЕНКО
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ
Рецензент: _____
науковий ступінь, вчене звання Ім'я, ПРІЗВИЩЕ

Київ 2025

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ

Навчально-науковий інститут Інформаційних технологій

Кафедра Комп'ютерної інженерії

Ступінь вищої освіти Магістр

Спеціальність 123 «Комп'ютерна інженерія»

Освітньо-професійна програма Комп'ютерні системи та мережі

ЗАТВЕРДЖУЮ
Завідувач кафедрою Наталія ЛАЩЕВСЬКА
_____Ім'я, ПРІЗВИЩЕ
«_____» _____ 2024р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

Тужиліну Денису Ігорьовичу
(прізвище, ім'я, по батькові здобувача)

1. Тема кваліфікаційної роботи: «Архітектура інтеграції ШІ компонентів у розподілених системах»
керівник кваліфікаційної роботи: Артем АНТОНЕНКО, канд.техн.наук, доцент,
(Ім'я, ПРІЗВИЩЕ, науковий ступінь, вчене звання)
затверджені наказом Державного університету інформаційно-комунікаційних технологій від «15» жовтня 2024р. №320
2. Строк подання кваліфікаційної роботи «29» грудня 2024р.
3. Вихідні дані до кваліфікаційної роботи: Науково-технічна література, технологічні рамки та протоколи, дизайн і архітектура системи, інтеграція та функціональність, оцінка.
4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
    1. Теоретичні основи ШІ
    2. Хмарні сервіси штучного інтелекту
    3. Розроблення та проєктування рішення API та інтеграції ШІ
    4. Розробка CI/CD рішень
5. Перелік ілюстративного матеріалу: *презентація*
    1. Мета, об'єкт, предмет дослідження
    2. Актуальність роботи
    3. Систематизація ШІ відгалужень
    4. Класифікація ШІ сервісів за видом діяльності
    5. Розробка API концепту
    6. Зіставлення хмарних сервісів
    7. Проектування концептуальної схеми міжсервісних з'єднань
    8. Проектування сервісної архітектури
    9. Висновки

10. Публікації та апробація роботи

6. Дата видачі завдання«01» вересня 2024р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1	Збір даних	10.10-12.10.2024	Виконано
2	Обробка матеріалу	12.10-15.10.2024	Виконано
3	Розробка теоретичної частини	15.10-18.10.2024	Виконано
4	Дослідження систем штучного інтелекту і машинного навчання широко застосування вузького і широкого спектора	18.10-22.10.2024	Виконано
5	Дослідження хмарних систем та архітектур штучного інтелекту	22.10-26.10.2024	Виконано
6	Проектування системи та архітектури модульного інтегратора штучного інтелекту	26.10-30.11.2024	Виконано
7	Висновки за результатами аналізу	30.11-02.12.2024	Виконано
8	Розробка презентації	02.12-08.12.2024	Виконано
9	Передзахист	09.12-11.12.2024	Виконано
10	Здача в деканат	20.12-29.12.2024	Виконано

Здобувач вищої освіти _____ Денис ТУЖИЛІН
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

Керівник кваліфікаційної роботи _____ Артем АНТОНЕНКО
(підпис) *(Ім'я, ПРІЗВИЩЕ)*

РЕФЕРАТ

Текстова частина кваліфікаційної роботи на здобуття освітнього ступеня магістр: с.70, Рис.14, Джерела 28.

Мета роботи – аналіз можливостей інтеграції AI-систем та моделей у сучасні системи CI/CD та бізнес-процеси, а також розробка концептуальної моделі інтеграції AI як модульних хмарних сервісів. Визначення основних архітектурних підходів до модульної інтеграції ШІ на базі хмарних платформ для забезпечення гнучкості, масштабованості та економічної ефективності рішень, орієнтованих на бізнес-процеси.

Об'єкт дослідження – взаємодії між системами штучного інтелекту та різними елементами інформаційних інфраструктур, зокрема, в контексті використання хмарних технологій для підтримки та розгортання ШІ як модулі.

Предмет дослідження – особливості проектування модульної архітектури інтеграції ШІ у хмарних середовищах, її вплив на ефективність бізнес-процесів та можливі технічні та економічні виклики, пов'язані з її впровадженням.

Короткий зміст роботи: У роботі проведено аналіз сучасних підходів до інтеграції ШІ у бізнес-процеси, визначено архітектурні принципи та підходи до проектування модульної інтеграції ШІ, орієнтованої на гнучкість та масштабованість у хмарних середовищах. Використовувалися методи аналізу наукових праць, порівняння сучасних підходів до проектування та впровадження ШІ-рішень, узагальнення результатів досліджень. Дослідження пропонує новий підхід до проектування модульної інтеграції ШІ, що базується на використанні хмарних платформ для забезпечення гнучкості та ефективності розгортання. Результати дослідження можуть використовуватися для розробки ефективних рішень щодо інтеграції ШІ в бізнес-процеси компаній, що дозволить підвищити

їхню конкурентоспроможність на ринку. Також матеріали роботи можуть бути корисними для подальших наукових досліджень у галузі модульної архітектури та інтеграції ШІ.

КЛЮЧОВІ СЛОВА: ШТУЧНИЙ ІНТЕЛЕКТ, МАШИННЕ НАВЧАННЯ, Хмарні послуги, БІЗНЕС ІНТЕГРАЦІЇ, AWS, GCP, AZURE.

ABSTRACT

The text part of the qualification work for obtaining a master's degree: 70 pages, 14 figures, 28 sources.

Purpose of the work – Analysing the possibilities of integrating AI systems and models into modern CI/CD systems and business processes, as well as developing a conceptual model of AI integration as modular cloud services. Identification of the main architectural approaches to modular AI integration based on cloud platforms to ensure flexibility, scalability, and cost-effectiveness of business process-oriented solutions.

Object of the research – peculiarities of designing a modular architecture for AI integration in cloud environments, its impact on the efficiency of business processes, and possible technical and economic challenges associated with its implementation.

Summary of the work: The paper analyses modern approaches to integrating AI into business processes, defines architectural principles and approaches to designing modular AI integration focused on flexibility and scalability in cloud environments. The methods used in the study were the analysis of scientific papers, comparison of modern approaches to the design and implementation of AI solutions, and generalisation of research results. The study proposes a new approach to the design of modular AI integration based on the use of cloud platforms to ensure flexibility and efficiency of deployment. The results of the study can be used to develop effective solutions for integrating AI into companies' business processes, which will increase their competitiveness in the market. Also, the materials of this paper may be useful for further research in the field of modular architecture and AI integration.

KEYWORDS: Artificial Intelligence, Machine Learning, Cloud Services, BUSINESS INTEGRATION, AWS, GCP, AZURE.

ЗМІСТ

Стор.

ВСТУП

Актуальність теми. Сучасний розвиток технологій, особливо штучного інтелекту, є важливим фактором автоматизації бізнес-процесів та оптимізації взаємодії людини з комп'ютером. ШІ вже активно використовується в областях, таких як обслуговування клієнтів, виробництво та логістика, демонструючи високу ефективність. Однак ефективна інтеграція ШІ в інфраструктуру потребує ретельного підходу до проектування архітектури та вибору відповідних технологій для забезпечення масштабованості та гнучкості. Це робить дослідження методів модульної інтеграції ШІ надзвичайно актуальним.

Ступінь опрацьованості проблеми. Штучний інтелект активно досліджується та впроваджується на багатьох підприємствах. Однак питання інтеграції ШІ-модулів у різні бізнес-системи потребує більш детального дослідження, зокрема щодо використання хмарних платформ для забезпечення адаптивності та економічної ефективності. Наявні дослідження здебільшого зосереджені на самих ШІ-алгоритмах, проте не вистачає уваги до архітектурних рішень, які можуть забезпечити їх оптимальне впровадження.

Цілі та завдання роботи. Метою даної роботи є розробка концептуальної моделі модульної інтеграції ШІ, орієнтованої на хмарні платформи, для ефективного та гнучкого впровадження ШІ у бізнес-процеси, шляхом дослідження, аналізу та оцінки сучасних архітектур та систем ШІ.

Об'єкт дослідження. Хмарні послуги та системи штучного інтелекту включаючи галузеві моделі.

Предмет дослідження. Архітектура систем штучного інтелекту та його похідних частин.

Методологія дослідження. Для досягнення мети дослідження використано методи аналізу наукових праць, схематичного моделювання, а також порівняння різних сервісів хмарних платформ та технологій мікросервісного з'єднання.

Джерела дослідження. Джерелами дослідження є наукові статті, навчальна література, наукові роботи та офіційна документація хмарних послуг.

Наукова новизна. У роботі представлено новий підхід до інтеграції ШІ-сервісів на основі хмарних платформ, що дозволяє створювати гнучку, масштабовану та економічно ефективну інфраструктуру для ШІ-додатків. Зокрема, запропонована архітектура враховує можливості автоматизованого розгортання, підтримку багатохмарної інтеграції та підвищену безпеку.

Теоретична та практична значущість роботи. Розроблена архітектура модульної інтеграції ШІ має високу практичну цінність, оскільки може бути впроваджена у багатьох сферах бізнесу для підвищення їхньої конкурентоспроможності. Теоретичні результати також можуть бути основою для подальших досліджень у галузі інтеграції ШІ на основі хмарних рішень.

Виносяться на захист становища. Представлено концептуальну архітектуру для модульної інтеграції ШІ-сервісів на хмарних платформах, що підвищує гнучкість та економічну ефективність. Оцінка економічних характеристик та розрахунок вигідності впровадження хмарних рішень для ШІ.

1 СТРУКТУРА І ПРОЦЕСИ ШТУЧНОГО ІНТЕЛЕКТУ

1.1 Визначення та класифікація компонентів штучного інтелекту

Штучний інтелект є широким терміном, що включає різні технології, що імітують когнітивні процеси людського мозку. Його основне завдання полягає у створенні систем, здатних навчатися, адаптуватися та вирішувати складні завдання без необхідності програмувати кожен крок. У контексті архітектури модульної інтеграції AI цей термін охоплює найпоширеніші рішення, що застосовуються у бізнес-процесах та проектній діяльності. (Рис.1.1)

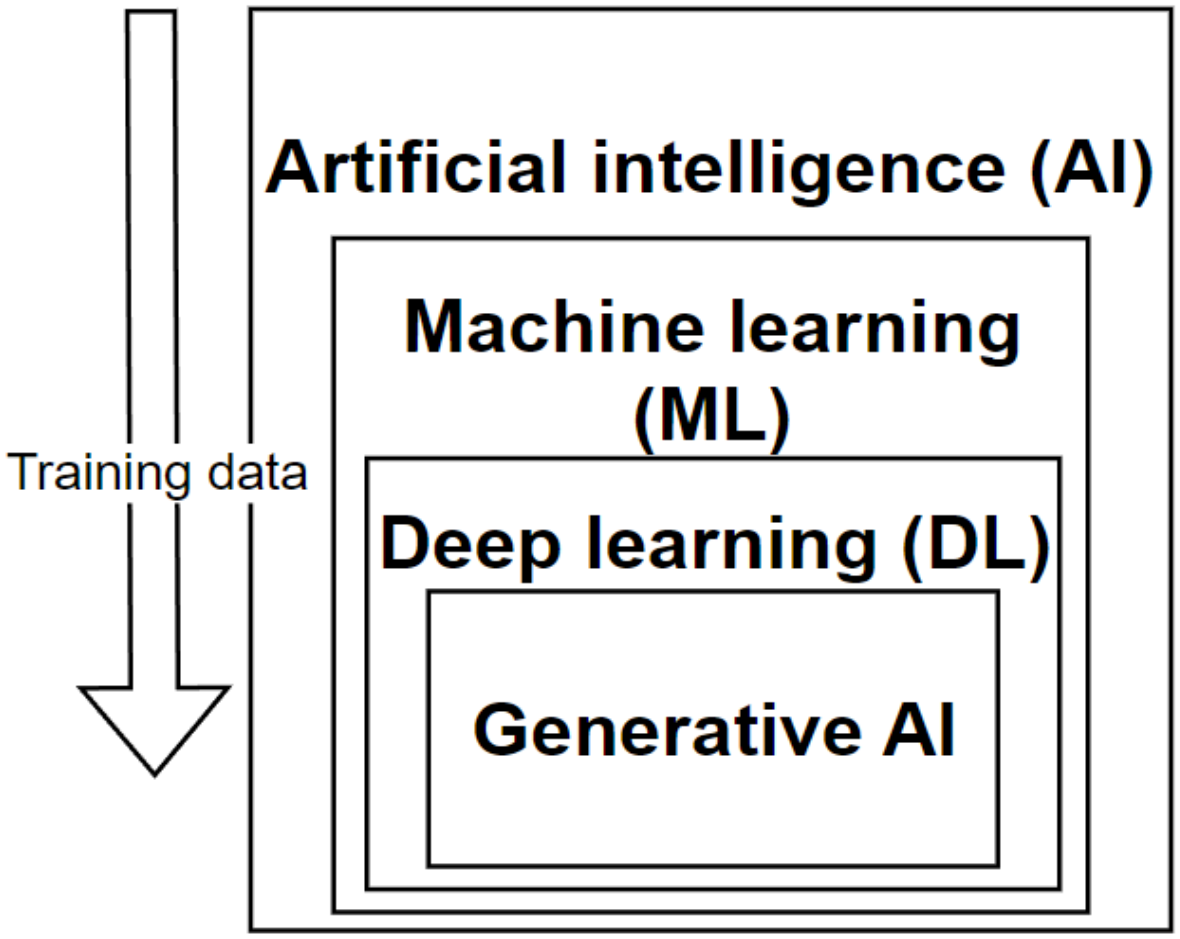


Рисунок 1.1 Структура обробки даних AI

Основою більшості систем AI є машинне навчання (Machine Learning, ML), що використовує методи та алгоритми для аналізу даних та підвищення продуктивності машин під час виконання завдань. Ці алгоритми навчаються на великих обсягах даних, що дозволяє системі покращувати свої результати із накопиченням досвіду. Важливо, коли говорять про моделі AI, найчастіше маються на увазі саме моделі, побудовані з використанням ML. Проте машинне навчання включає і більш складні системи, такі як глибоке навчання (Deep Learning, DL).

Глибоке навчання використовує багатошарові нейронні мережі, що імітують роботу людського мозку. Ці мережі складаються з кількох шарів нейронів, де кожен шар обробляє дані більш високому рівні абстракції. Завдяки цій структурі глибоке навчання здатне вирішувати складні завдання, такі як розпізнавання зображень та обробка природної мови.

Генеративний AI (Generative AI) є підмножиною глибокого навчання і спрямований на створення нових даних, що базуються на існуючих патернах. Ці системи можуть генерувати текст, зображення та інші форми даних, не вимагаючи значного переучування чи налаштування. Наприклад, генеративні моделі, такі як GPT (Generative Pre-trained Transformer), використовують вже навчені патерни для створення унікального контенту. Генеративні AI-системи знаходять широке застосування таких областях, як автоматизація творчості, генерація текстів і зображень, соціальній та інших завданнях, що з генерацією нових даних.

Для того, щоб ефективно використовувати AI в бізнесі, необхідно розуміти структуру обробки даних, що включає кілька ключових етапів. Спочатку дані збираються з різних джерел, включаючи бази даних, сенсорні пристрої або публічні API. використання в алгоритмах. Потім відбувається навчання моделей, де застосовуються алгоритми машинного або глибокого навчання для вирішення конкретних завдань. передбачення, класифікації чи генерації контенту. Сучасні системи AI здатні адаптуватися до змін та використовувати нові дані для коригування своїх моделей, що робить їх більш гнучкими та ефективними.

У контексті модульної інтеграції AI особлива увага приділяється тому, як і через що різні компоненти AI можна ефективно інтегрувати в існуючі бізнес-процеси. до інференції та адаптації моделей. Саме тому канонічна архітектура будується по взаємозв'язку трьох ключових елементів: машинного навчання (ML), глибокого навчання (DL) та генеративного AI. Така модель дозволяє представити AI як багаторівневу систему, де кожен рівень є основою більш складних алгоритмів.

1.2 Типізація тренувальних даних та процес машинного навчання

Першим і ключовим питанням, яке повинні ставити собі розробники, які планують інтеграцію штучного інтелекту до свого проекту, є: "Якими даними ми оперуємо?" Відповідь це питання визначає як підхід до роботи з даними, а й вибір AI/ML сервісу, здатного найкраще впоратися з поставленими завданнями. Різні сервіси пропонують специфічні інструменти для роботи з певними типами даних - від структурованих таблиць до неструктурованих аудіо- та відеозаписів. Якість, тип, обсяг і актуальність даних безпосередньо впливають ефективність обраного рішення, тому аналіз характеристик даних стає відправною точкою у прийнятті архітектурних рішень.

Процес створення моделі машинного навчання (ML) починається з етапу збирання та обробки даних. Якість моделі безпосередньо залежить від якості використовуваних даних, що відображає відомий вираз "garbage in, garbage out". Саме етап обробки та підготовки даних є основою успішного функціонування будь-якої ML-моделі. [7] (рис.1.2)

Рисунок 1.2 Формування ML-моделі

Дані для машинного навчання класифікуються за двома основними ознаками: наявності міток та структури.

1. Розмічені дані (Labeled data) – це дані, де кожен приклад супроводжується міткою або цільовою змінною, яка є правильним результатом чи класифікацією. Такі мітки можуть бути створені вручну експертами або автоматично за допомогою спеціальних методів розмітки.

2. Нерозмічені дані (Unlabeled data) – дані, що не містять міток або цільових змінних. Вони складаються лише з вхідних ознак, без зазначення бажаного результату, що робить їх складними для використання у завданнях класифікації та регресії.[8]

Також дані діляться за своєю структурою:

− структуровані дані (Structured data) – організовані за певною схемою, наприклад, як таблиці чи бази даних. Цей тип даних найбільше підходить для традиційних алгоритмів машинного навчання. Приклад: табличні дані (Excel, CSV-файли), дані тимчасових рядів (курси валют, погодні дані);

− Неструктуровані дані (Unstructured data) немає чіткої структури. Для їх обробки потрібні складніші методи аналізу. приклад: текстові документи, зображення, відео;

Для кращого розуміння необхідно розглянути, як взаємодіють типи даних та методи навчання:

− структуровані з розміченими даними: використовуються завдання контрольованого навчання, де мета – навчити модель даних з відомими мітками;

− структуровані з нерозміченими даними: застосовують у завданнях неконтрольованого навчання (unsupervised learning), де модель самостійно виявляє закономірності даних;

− неструктуровані з розміченими даними: також підходять для контрольованого навчання, де текст чи зображення розмічено вручну для класифікаційних завдань;

− неструктуровані з нерозміченими даними: дані застосовні у завданнях неконтрольованого навчання для пошуку прихованих закономірностей чи кластеризації;

Вибір відповідного типу даних та його коректна підготовка є критично важливими чинниками побудови успішної моделі машинного навчання. Метод організації та розмітки даних визначає вибір способів навчання та точність моделі.

Раніше вже згадувалося навчання у контексті структур та маркування тренувальних даних. Такі зібрані навчальні дані надходять до алгоритмів машинного навчання. де обробляються процесами навчання МЛ. Традиційно таке навчання поділяється на три великі категорії: контрольоване навчання, неконтрольоване навчання та навчання з підкріпленням.

При контрольованому навчанні (supervised learning) алгоритми тренуються за вказаними даними. Ціль полягає в тому, щоб вивчити функцію відображення, яка може передбачити вихід для нових, невідомих вхідних даних.

Непідконтрольне навчання (unsupervised learning) відноситься до алгоритмів, що навчаються на немаркованих даних. Ціль полягає в тому, щоб виявити властиві вхідним даним закономірності, структури або взаємозв'язку.

При навчанні з підкріпленням (reinforcement learning/RL) машина отримує лише оцінку роботи як керівництво, а при напівконтрольному навчанні маркується лише частина навчальних даних. , щоб поліпшити процес ухвалення рішень.

Після того, як модель навчена, час використовувати отриману інформацію для прогнозування та прийняття рішень. Це називається висновком (inferencing).

У машинному навчанні існує два основних типи висновків:

Пакетні висновки (Batch inferencing) – це коли комп'ютер отримує великий обсяг даних, наприклад зображень чи тексту, і аналізує їх відразу, щоб одержати набір результатів. Цей тип висновків часто використовують у таких завданнях, як аналіз даних, де швидкість процесу прийняття рішення менш важлива, як точність результатов.[9]

Висновки у реальному часі (Real-time inferencing) – це коли комп'ютер повинен приймати рішення швидко, реагуючи на нову інформацію у міру надходження. Це важливо для додатків, що вимагають негайного прийняття рішень, наприклад, у чат-ботах або самоврядних автомобілях. Комп'ютер повинен обробляти дані, що надходять, і приймати рішення практично миттєво, не витрачаючи часу на аналіз великого масиву даних.

У той же час RL може працювати як зі структурованими, так і неструктурованими даними, залежно від того, які вхідні дані надає середовище.

Однак RL фокусується на максимізації винагороди, а не на роботі безпосередньо з мітками, тому вона не пов'язана з різницею міченою/німецькою.

1.3 Поняття глибинного навчання

Нейронні мережі імітують роботу людського мозку, що складається з нейронів, які взаємодіють один з одним, щоб обробляти та передавати інформацію. У контексті машинного навчання нейронні мережі відіграють центральну роль, забезпечуючи здатність моделей навчатися і адаптуватися на основі великих обсягів даних. Вони є основою для побудови складних ML-моделей, які здатні вирішувати завдання, починаючи від класифікації та прогнозування до обробки зображень та тексту. Глибокі нейронні мережі, що містять безліч шарів, забезпечують високий рівень абстракції, що дозволяє моделям ефективно вирішувати завдання, які раніше були неможливими для

традиційних алгоритмів. У цьому контексті глибоке навчання стає невід'ємним інструментом для розвитку та вдосконалення штучного інтелекту.

Глибоке навчання, як результат досліджень роботи мозку, засноване на натхненні структурою та функціями людського мозку. мозку. Штучні нейронні мережі складаються з безлічі вузлів (або нейронів), організованих у шари: вхідний, прихований та вихідний. (Рис.1.3)

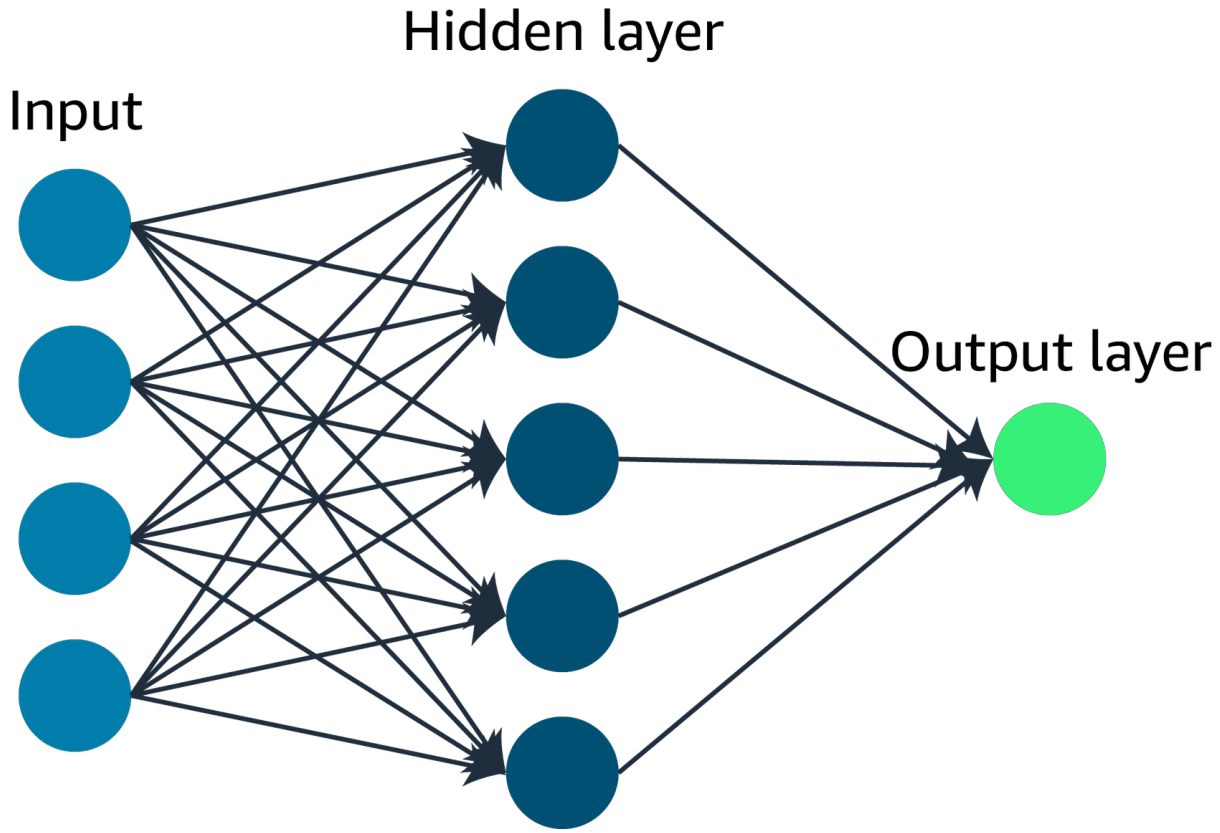


Рисунок 1.3 Шари глибинного навчання

Коли нейронна мережа навчається на великій кількості даних, наприклад, даних про клієнтів, які купують або використовують певні послуги, вона починає виявляти закономірності, регулюючи зв'язки між вузлами, що розрізняють типи клієнтів. нових клієнтів.

Глибоке навчання застосовується у різних галузях. Наприклад, у комп'ютерного зору (Computer Vision) він зробив революцію, надавши ефективні методи таких завдань, як класифікація зображень, виявлення об'єктів та сегментація. Ці технології дозволяють комп'ютерам інтерпретувати та розуміти візуальну інформацію з високою точністю.

Іншою важливою областю є обробка природної мови (Natural Language Processing, NLP), де глибоке навчання дозволило значно просунутися у вирішенні завдань, пов'язаних із взаємодією комп'ютерів та людської мови. За допомогою глибоких нейронних мереж стали можливі такі завдання, як класифікація текстів, аналіз настроїв, машинний переклад і навіть створення осмисленого тексту, що має широке застосування в сучасних цифрових сервісах і бізнесі.

Хоча глибинне навчання і вважається окремим від ML типом AI, воно все ж таки залишається галуззю машинного навчання.

1.4 Foundation Modules та Генеративний AI

Генеративний AI тісно пов'язаний із використанням моделей, які заздалегідь навчені великим обсягам даних, доступним в Інтернеті. Ці моделі називаються базовими моделями (Foundation Models, FM). На відміну від традиційних підходів машинного навчання, де для кожного завдання необхідно збирати окремі марковані дані та вивчати унікальні моделі, FM надають можливість адаптувати одну універсальну модель для вирішення безлічі завдань. Це може включати такі завдання, як генерація тексту, узагальнення даних, отримання інформації, створення зображень, взаємодія з чат-ботами та відповіді на запитання. FM стає відправною точкою розробки більш спеціалізованих моделей, що значно скорочує витрати часу та ресурсів створення нових рішень у сфері штучного інтелекту.[10]

Життєвий цикл FM включає кілька етапів, починаючи з вибірки даних, передтренування моделі і закінчуючи її оптимізацією, оцінкою і розгортанням у виробничих середовищах. Цей процес також включає постійне вдосконалення моделі через зворотний зв'язок. Важливою особливістю FM є навчання з використанням самоконтрольованого навчання (self-supervised learning), де модель самостійно витягує мітки зі структури даних без необхідності заздалегідь розмічених прикладів. Це робить процес навчання масштабованим та гнучким.

Етап передтренування включає роботу з немаркованими даними, що спрощує збирання даних. В результаті передтренування модель вчиться розпізнавати різні структури та зв'язки всередині даних, такі як зміст, контекст та відносини між словами. Наприклад, модель може розуміти, що слово "drink" може використовуватися як іменник ("напій") або як дієслово ("пити"). Надалі така модель може бути доучена для вирішення спеціалізованих завдань, що робить її універсальною основою різних програм.

Попередньо навчені моделі можуть бути покращені через різні методи оптимізації, такі як проектування підказок (prompt engineering), генерація з розширенням пошуку (retrieval-augmented generation, RAG) та тонке налаштування (fine-tuning) на спеціалізованих даних. Ці методи дозволяють досягти кращої продуктивності моделі для конкретних завдань. При цьому оцінка ефективності моделі проводиться через різні метрики, щоб переконатися у відповідності до результату необхідних бізнес-цілей чи дослідницьких задач.[11]

Одним із ключових етапів є оптимізація результатів моделі. У цьому процесі може використовуватися ряд методів, від відносно простих і дешевих, таких як проектування підказок, до складніших, як тонке налаштування. Проектування підказок передбачає розробку та оптимізацію запитів до моделі, щоб направити її роботу до отримання потрібних результатів. Форма підказок залежить від завдання, поставленого перед моделлю, і може містити інструкцію, контекст, вхідні дані та очікуваний формат відповіді. Цей метод особливо корисний на ранніх етапах впровадження моделі, оскільки потребує мінімальних витрат часу та ресурсів.

Тонка настройка є складнішим процесом, у якому модель, навчена загальним даним, доучается на вузькоспеціалізованих наборах даних, що покращує її продуктивність для конкретних завдань. Процес fine-tuning включає додавання специфічних даних, що дозволяє моделі точніше впоратися з поставленими завданнями. Це особливо корисно в тих випадках, коли потрібна висока точність у вузькоспеціалізованих областях, таких як медицина чи юридична сфера. Тонка настройка може бути реалізована в декількох формах,

включаючи навчання моделі на прикладах інструкцій або використання навчання з підкріпленням на основі зворотного зв'язку з людиною (Reinforcement Learning from Human Feedback, RLHF). У разі зворотний зв'язок, надана користувачами, допомагає моделі краще відповідати їх перевагам.[12]

Якщо розглядати оптимізацію на прикладі конкретного завдання, наприклад, у медичних дослідженнях, можна взяти вже попередньо навчену модель і довчити її на медичних статтях, щоб вона видавала більш релевантні та контекстні результати. Такий підхід дозволяє значно підвищити продуктивність моделі у вузькоспеціалізованих галузях.

У підході RAG використовується можливість отримання релевантних документів або даних, які використовуються для створення контексту і допомагають моделі генерувати більш точні відповіді. На відміну від тонкої установки, що змінює вагові коефіцієнти моделі, RAG додає додаткову інформацію, не змінюючи базову структуру моделі. Це робить RAG гнучким методом покращення продуктивності без необхідності глибокої зміни самої моделі.

Генеративний AI, як було зазначено, базується на методах глибокого навчання, що забезпечують архітектурні основи та алгоритми, здатні обробляти величезні обсяги даних. Одними з найпомітніших типів моделей, що використовують FM, є великі мовні моделі (Large Language Models, LLMs), дифузійні моделі (Diffusion Models), мультимодальні моделі, генеративні мережі змагань (Generative Adversarial Networks, GANs) та варіаційні автоенкодери (VAN). . Ці моделі відіграють ключову роль у розвитку та застосуванні генеративного AI. (Рис.1.4)

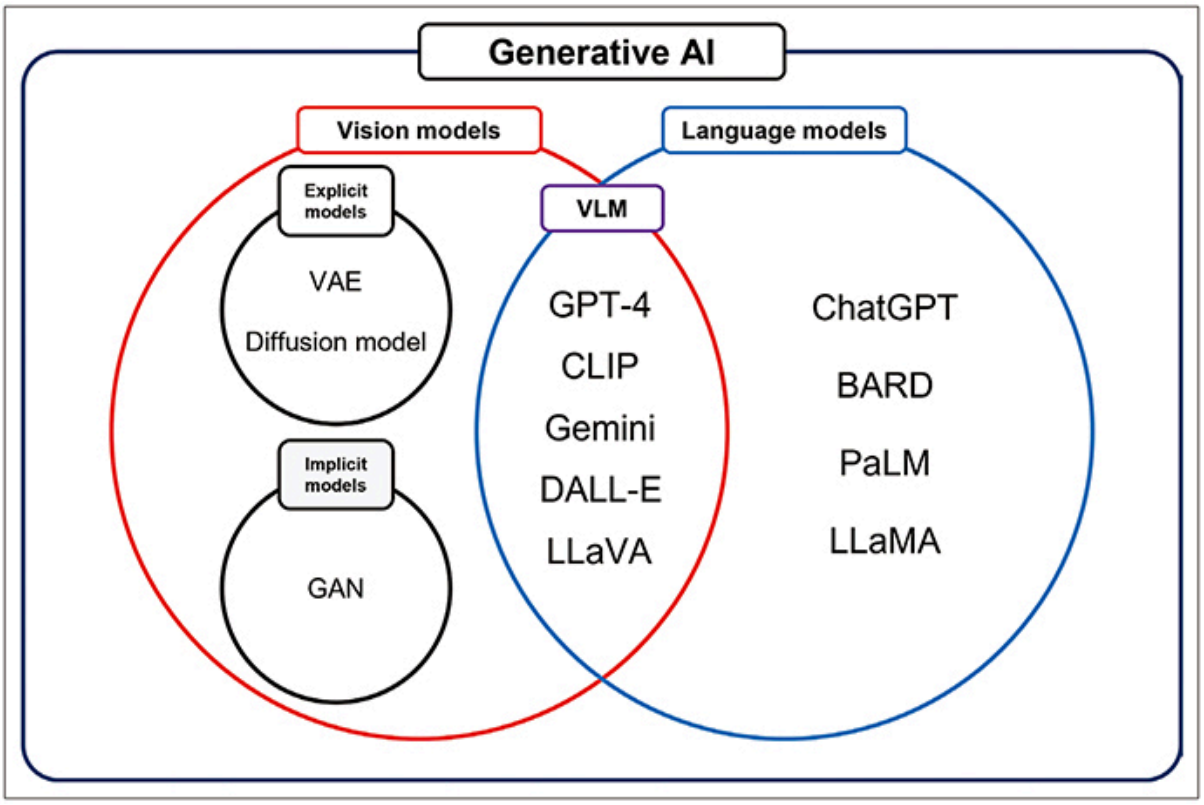


Рисунок 1.4 Шари генеративного AI

Великі мовні моделі (LLM), побудовані на архітектурі трансформерів, здатні генерувати людиноподібний текст і навчаються великим обсягом текстових даних. Їх основними компонентами є токени та векторні вкладення, які допомагають моделі розпізнавати контекст слів та взаємозв'язку. Наприклад, слово "cat" матиме векторне уявлення, близьке до таких слів, як "kitten" і "feline", що відображає їх семантичний зв'язок. Це дозволяє моделі генерувати текст, усвідомлюючи лексичні та контекстуальні взаємозв'язки, що робить її здатною до більш складних завдань, таких як узагальнення тексту, відповіді на запитання та творчий лист. Крім цього, саме LLM використовується як основа для побудови обробки природної мови (NLP). [2][4]

Дифузійні моделі працюють з урахуванням принципу перетворення шуму структуровані дані, поступово додаючи значну інформацію. Такі моделі використовуються для створення зображень та інших типів даних. Навчання таких моделей включає етапи прямої та зворотної дифузії, що дозволяє моделі ефективно відновлювати впорядковані структури з вихідного шуму. Це робить

дифузні моделі корисними як для генерації зображень, але й інших складних завдань у сфері штучного інтелекту.

Мультимодальні моделі можуть обробляти та об'єднувати декілька типів даних, таких як текст та зображення. Це робить їх особливо корисними для створення складних даних, наприклад, графіки на основі текстових описів або автоматичної генерації підписів до зображень.

Генеративні змагальні мережі (GANs) працюють на основі взаємодії двох нейромереж: генератора та дискримінатора. Генератор створює синтетичні дані, а дискримінатор оцінює їхню справжність. Цей процес призводить до поступового поліпшення якості даних, що генерується, що робить GANs корисними для створення реалістичних зображень і відео.

Варіаційні автоенкодери (VAEs) виконують стиск вхідних даних в низькорозмірний латентний простір і відновлюють їх з цього простору, що дозволяє генерувати нові дані на основі розподілу ймовірності. Це робить VAEs особливо ефективними для генерації нових даних, які за своєю структурою та змістом близькі до вихідних даних.[6]

Таким чином, всі ці моделі демонструють модульний підхід, при якому кожна з них може застосовуватися для вирішення специфічних завдань, а Foundation Models забезпечують гнучкість і масштабування адаптації.

1.5 Сценарії використання елементів штучного інтелекту

Для отримання більшої картини структури штучного інтелекту найпростішим методом залишається побачити методи застосування його елементів.

AI. Комп'ютерний зір дозволяє системам розпізнавати зображення та відео, які знаходять застосування в автономному керуванні, медичній візуалізації та безпеці.

Обробка природної мови працює з людськими мовами для класифікації тексту, аналізу настроїв, перекладу та генерації тексту. Наприклад, у страхуванні NLP дозволяє витягувати дані з документів і захищати конфіденційну информацию.[2]

Інтелектуальна обробка документів (IDP) отримує дані з неструктурованих документів, прискорюючи обробку заявок та документів, наприклад, у фінансовій сфері при кредитуванні.

Виявлення шахрайства допомагає виявляти підозрілі дії, ідентифікуючи шахрайські операції у фінансових транзакціях та продажах, що особливо важливо у роздрібній торгівлі та телекомунікаціях.

ML. Використання ML стає актуальним, коли просте програмування із чітко заданими правилами не може ефективно вирішити завдання. Наприклад, для фільтрації спаму в електронній пошті недостатньо покладатися на статичні правила, оскільки повідомлення можуть сильно відрізнятися за структурою та змістом. Машинне навчання дозволяє розробити алгоритми, що аналізують дані, що виділяють ключові ознаки та класифікуючі повідомлення з високою точністю, навіть якщо структура повідомлень змінюється з часом.

Таким чином, машинне навчання вирішує такі проблеми як: [1]

1. Регресія.
2. Класифікація.
3. Сегментація.
4. Аналіз мереж

Особливо ML корисний у випадках, де обсяг даних настільки великий, що людина чи навіть традиційний програмний код не впораються із завданням у розумні терміни. У таких випадках ML пропонує можливість аналізу та обробки

даних у масштабах, необхідних для бізнесу чи складних дослідницьких проектів. Моделі машинного навчання не лише автоматизують цей процес, а й підвищують якість та швидкість обробки.

Однак використання ML не завжди є виправданим. Якщо завдання можна вирішити за допомогою набору простих правил, використання ML може виявитися зайвим. вибрати підходящий підхід.

Генеративний AI. Генеративний AI знаходить застосування у багатьох сценаріях завдяки своїй гнучкості та можливості виконувати завдання на основі навчання на великих обсягах даних. Основні напрямки використання включають генерацію текстів, зображень, а також підтримку взаємодії природною мовою.

У сценаріях створення текстів, таких як створення текстів, резюме, перефразування, обробка природної мови, дозволяє моделям, наприклад, автоматизувати роботу з документацією, відповідати на запитання і навіть допомагати в аналізі настроїв. Наприклад, текстові моделі активно використовуються у службах підтримки клієнтів для створення автоматичних відповідей, що знижує навантаження на співробітників та підвищує якість обслуговування.

Генеративні AI моделі для створення зображень та візуальних елементів особливо корисні у маркетингу та креативних індустріях, де потрібне швидке створення візуальних матеріалів. орієнтовані створення зображень, і тому вони знаходять застосування у графічному дизайні, рекламних кампаніях та візуальному брендингу.

Також генеративний AI активно застосовується для створення коду та підтримки розробників. Такі моделі можуть генерувати кодові фрагменти, перевіряти та виправляти помилки, допомагати у тестуванні та автоматизувати частину роботи, що особливо корисно для прискорення процесів розробки та підвищення якості коду.

При виборі генеративної моделі для конкретного сценарію слід враховувати багато факторів, таких як продуктивність, обмеження ресурсів, можливості моделі та вимоги до відповідності нормативним вимогам. Наприклад, у сценаріях з

високими вимогами до надійності чи точності потрібне використання моделей з високою точністю та адаптивністю, а у додатках, де потрібна відповідність стандартам конфіденційності, важливо оцінити можливість керування даними та коректність моделі.

Таким чином, генеративний AI, чи то для тексту, зображень чи коду, дозволяє автоматизувати та оптимізувати процеси в різних галузях, від підтримки клієнтів та маркетингу до розробки та креативних проектів, що допомагає покращити ефективність бізнесу та знизити витрати на виконання повторюваних завдань.

1.6 AI для вирішення основних глобальних проблем

AI/ML-сервіси значно впливають на вирішення глобальних проблем, таких як продовольча безпека, зміна клімату, втрата біорізноманіття, покращення якості життя, підвищення рівня кібербезпеки та прискорення наукового прогресу. Ці технології дозволяють аналізувати величезні масиви даних, виявляти закономірності, розробляти стійкі рішення та пропонувати інноваційні підходи, що допомагають покращити умови життя та зберегти природні ресурси.[13]

Штучний інтелект та машинне навчання активно використовуються в аграрній промисловості, де вони допомагають оптимізувати процеси виробництва, підвищувати врожайність та знижувати витрати. Завдяки цим технологіям агропідприємства можуть ефективно керувати посівними площами та ресурсами, прогнозувати врожайність. Використання AI у сільському господарстві включає аналіз стану ґрунту та кліматичних даних, що дозволяють визначати оптимальні умови для росту рослин та вибирати найкращі терміни посадки. Це допомагає знизити вплив несприятливих погодних умов та покращити ефективність використання ресурсів. AI також застосовується для ранньої діагностики захворювань рослин. На основі аналізу зображень та даних про погоду моделі

машинного навчання дозволяють виявляти ознаки захворювань рослин на ранніх стадіях, що допомагає аграріям оперативно вживати заходів та мінімізувати втрати врожаю. Важливу роль відіграє і управління водними ресурсами: розумні системи поливу, оснащені алгоритмами AI, можуть регулювати інтенсивність поливу в залежності від стану ґрунту та прогнозу погоди, тим самим заощаджуючи воду та енергію. Ці можливості роблять аграрну промисловість більш стійкою і готовою до кліматичних умов, що змінюються, сприяючи підвищенню продовольчої безпеки.

AI та ML знаходять широке застосування у вирішенні екологічних завдань, пов'язаних із зміною клімату. Ці технології використовуються для аналізу кліматичних даних, які допомагають урядам та організаціям розробляти стратегії боротьби з глобальним потеплінням, зниження викидів вуглекислого газу та переходу до відновлюваних джерел енергії. Машинне навчання дозволяє моделювати зміни клімату та прогнозувати екстремальні явища погоди, такі як урагани, посухи та повені. Це забезпечує державним та приватним структурам дані для оперативного реагування та захисту населення від природних катастроф. Оптимізація енергоспоживання також стає можливою завдяки AI: алгоритми машинного навчання підвищують ефективність виробництва та управління енергетичними мережами, що дозволяє скорочувати втрати енергії та оптимізувати її розподіл. Важливим напрямком є підтримка переходу до відновлюваних джерел енергії, де AI допомагає оптимізувати розміщення сонячних панелей та вітряних турбін з урахуванням кліматичних умов, що сприяє максимальному використанню природних ресурсів. Ці технологічні рішення спрямовані на створення стійкої та екологічно чистої інфраструктури, що важливо для зниження вуглецевого сліду та переходу до низьковуглецевої економіки.

AI та ML відіграють ключову роль у збереженні біорізноманіття, забезпечуючи можливості для точного моніторингу та аналізу стану різних екосистем. За допомогою технологій AI можна відстежувати популяції тварин, фіксувати їх міграцію та аналізувати зміни у чисельності та поведінці видів. Ці технології дозволяють своєчасно виявляти загрози, такі як браконьєрство, втрата

місць проживання та зміна клімату, що допомагає розробляти адаптивні заходи щодо охорони та збереження видів. Машинне навчання також аналізує супутникові знімки та дані з дронів, що спрощує моніторинг лісових масивів, коралових рифів та інших вразливих екосистем. Це дозволяє швидко виявляти випадки вирубування лісів, руйнування екосистем та інші антропогенні впливи, що дозволяє вживати заходів щодо їх збереження. Генеративний AI також знаходить застосування в галузі біорізноманіття, надаючи можливості для створення штучних екосистем та розробки генетичних моделей для відновлення видів, що вимирають. Це особливо актуально в умовах клімату, де такі рішення можуть підтримувати рідкісні та вразливі види, а також моделювати відновлення пошкоджених екосистем.

Сфери охорони здоров'я та підвищення якості життя також виграють від використання AI та ML. Ці технології дозволяють аналізувати медичні дані, виявляти індивідуальні ризики здоров'я та розробляти персоналізовані методи лікування. AI-системи обробляють медичні карти, геномні дослідження та лабораторні дані, які допомагають лікарям передбачати розвиток захворювань та підбирати оптимальні методи лікування з урахуванням індивідуальних особливостей пацієнта. Машинне навчання відіграє важливу роль в оптимізації діагностики та терапії: алгоритми можуть навчатися на мільйонах медичних записів, виявляючи патерни, які допомагають знаходити ефективні методи лікування для пацієнтів із різними генетичними, віковими та іншими особливостями. Це робить медицину більш точною та персоналізованою, особливо в лікуванні хронічних та рідкісних захворювань. Генеративний AI у медицині сприяє створенню нових ліків та терапії, моделюючи взаємодію лікарських речовин з клітинами та тканинами, що прискорює розробку безпечних та високоефективних препаратів. Ці технології також використовуються для створення рекомендацій щодо здорового способу життя, що допомагає уповільнити процеси старіння та зміцнити імунітет.

AI, ML та Generative AI мають важливий вплив на кібербезпеку, допомагаючи виявляти та запобігати кіберзагроз. Ці технології дозволяють

аналізувати мережевий трафік та виявляти аномалії, які можуть вказувати на погрози. Системи, що використовують AI, можуть навчатися на даних про попередні атаки та адаптуватися, щоб розпізнавати нові типи загроз та своєчасно реагувати на них. AI алгоритми аналізують мережевий трафік та визначають підозрілі шаблони та поведінки, мінімізуючи вплив людського фактора та дозволяючи запобігати атакам до їх реалізації. Крім того, машинне навчання може використовуватися для аналізу поведінки користувачів та виявлення загроз зсередини компанії, що допомагає запобігти атакам, які здійснюються при використанні цих співробітників. Генеративний AI використовується для створення симуляцій та пасток (honeypots), які допомагають виявити зловмисників та відстежувати їхню поведінку, що дає цінні дані для розробки більш ефективних заходів кіберзахисту. Таким чином, AI і ML забезпечують проактивний підхід до забезпечення безпеки, що стає все більш актуальним в умовах зростання кількості кібератак і складності загроз, що підвищується.

Науковий прогрес також прискорюється завдяки можливостям AI та ML, які надають дослідникам нові інструменти для аналізу, моделювання та генерації даних. Ці технології дозволяють автоматизувати складні обчислення та аналізувати великі масиви даних, що спрощує та прискорює дослідницькі процеси. Машинне навчання може знаходити приховані закономірності в геномних даних, аналізувати результати тисяч досліджень та моделювати поведінку матеріалів, що допомагає дослідникам швидше формулювати та перевіряти гіпотези. Generative AI відкриває можливості для моделювання складних процесів та розробки нових молекул, що особливо актуально у фармацевтиці та матеріалознавстві. Завдяки цим технологіям вчені можуть швидше переходити від теорії до практики, а також розробляти та тестувати гіпотези у віртуальних лабораторіях, що суттєво знижує витрати на проведення фізичних експериментів. AI також допомагає оптимізувати робочі процеси в лабораторіях, звільняючи дослідників від рутинних завдань та дозволяючи їм фокусуватися на поглибленому аналізі та теоретичній роботі. Ці технології підтримують прагнення суспільства до наукового прогресу, надаючи потужні

аналітичні інструменти, які роблять дослідницький процес більш продуктивним та стійким.

Таким чином, застосування AI/ML-сервісів для вирішення глобальних проблем відкриває перед суспільством унікальні можливості створення більш стійкого, безпечного та інклюзивного майбутнього. Ці технології не лише дозволяють покращити існуючі процеси, а й відкривають нові напрями наукових досліджень та економічного розвитку. Штучний інтелект та машинне навчання вже сьогодні допомагають керувати ресурсами, прогнозувати природні та соціальні зміни, забезпечувати продовольчу безпеку та захищати біорізноманіття, сприяючи переходу до сталого розвитку.

Майбутнє AI/ML-технологій обіцяє ще більш глибоку інтеграцію в різні сфери життя, де вони виступатимуть як потужний інструмент для вирішення як локальних, так і глобальних завдань. їх безпечне та моральне використання. Розвиток AI та ML має супроводжуватися увагою до прав людини, питань зокрема та безпеки, щоб створені рішення справді приносили користь суспільству та сприяли покращенню якості життя кожної людини.

В результаті, AI/ML-сервіси допомагають вирішувати глобальні проблеми завдяки унікальній здатності аналізувати величезні масиви даних, виявляти закономірності та пропонувати оптимальні рішення для найскладніших викликів сучасності. Ці технології впроваджують принцип передиктивного аналізу, дозволяючи прогнозувати та запобігати ризикам до їх настання, як це робиться в кліматичному прогнозуванні або у сфері кібербезпеки. AI та ML здатні автоматизувати рутинні процеси та знижувати людський фактор, що особливо важливо в таких галузях як аграрне виробництво та управління ресурсами. Їхнє застосування прискорює реакцію на проблеми, як це відбувається в охороні здоров'я, коли моделі допомагають ранній діагностиці та створенню персоналізованих методів лікування.

Ці технології також підтримують комплексні системи моніторингу, надаючи точні дані у реальному часі, що особливо важливо для збереження біорізноманіття та стійкості екосистем. Таким чином, AI/ML-сервіси дозволяють поєднувати

аналіз даних, автоматизацію та моніторинг у єдиний керований процес, що призводить до більш швидкого та точного прийняття рішень на всіх рівнях. Комплексний підхід, що поєднує AI, ML та методи генеративного моделювання, стає основою стійких змін, спрямованих на покращення якості життя, охорону навколишнього середовища та розвиток безпечного цифрового майбутнього.

В результаті AI/ML-сервіси стають не просто інструментами, а необхідними компонентами у стратегіях подолання світових викликів та створення майбутнього, в якому технологічний прогрес гармонійно поєднується із потребами суспільства та природи.

2 СИСТЕМАТИЗАЦІЯ AI ТЕХНОЛОГІЙ З ВИДУ ДІЯЛЬНОСТІ І ДОСЛІДЖЕННЯ ЇХ ІНТЕГРАЦІЙНИХ СПОСІБ

2.1 Визначення сервісних напрямів штучного інтелекту та його класифікація

Хмарні сервіси – це не тільки залізні сервери, які можна отримати в оренду без необхідності їхнього безпосереднього розміщення у користувачів. Основа їхньої появи в систематизації знань IT-технологій, а саме сервісів. Сервіси є акумуляторами готових рішень, які не вимагають написання нового сервісу. Однак переважно вимагають створення коду, що вміє взаємодіяти із цими сервісами. Систематизація сервісних напрямків штучного інтелекту, запропонованих найхмарнішими провайдерами, такими як AWS, Azure та GCP, дозволяє виділити кілька ключових категорій, що відображають багате розмаїття областей застосування.(Рис.2.1)

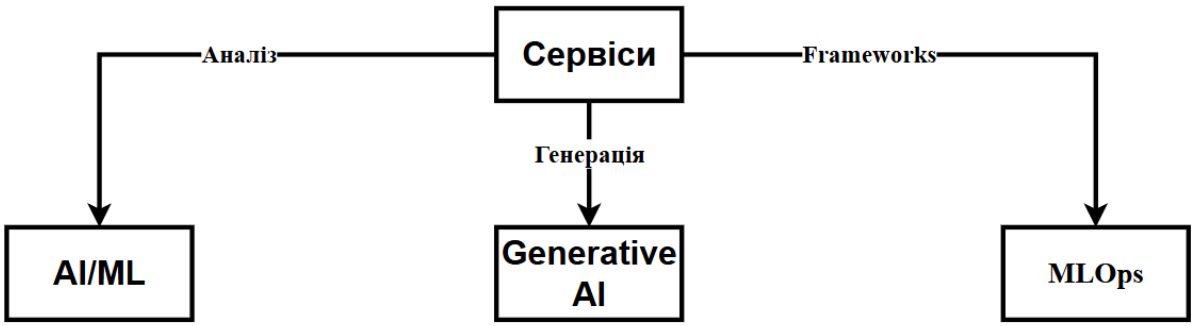


Рисунок 2.1 Класифікація ШІ сервісів

Однією з таких категорій є AI/ML сервіси, що надають всі необхідні інструменти для виконання машинного навчання і штучного інтелекту. Ці послуги включають широкий спектр функцій: від обробки даних і побудови моделей до їх навчання та подальшого застосування для вирішення конкретних завдань. На практиці AI/ML-сервіси широко використовують для обробки природної мови, де

вони допомагають аналізувати текст, робити автоматичний переклад, виконувати розпізнавання мови та обробляти текстові дані. Візуальний аналіз, що також входить до цього класу, надає можливості для обробки зображень та відео, дозволяючи ідентифікувати об'єкти, аналізувати особи, класифікувати та структурувати дані з візуальних джерел. Також одним із важливих напрямків у рамках AI/ML-сервісів є створення рекомендаційних систем, що дозволяють компаніям надавати користувачам персональні пропозиції, контент, продукти чи послуги. На додаток до цього AI/ML-сервіси використовують методи аналізу даних, створюють прогнози та розподіляють дані за класами, забезпечуючи таким чином комплексний підхід до управління процесом навчання моделей.

Особливе місце серед сервісів штучного інтелекту займають generative AI рішення, що дозволяють створювати контент на основі навчених моделей, тих же foundation models. Насправді Generative AI, на відміну класичних AI/ML-сервисов, спрямоване створення нових даних – тексту, зображень, коду та іншого контенту. Такі можливості знаходять все більш широке застосування, наприклад для створення синтетичних даних, створення описів і унікальних текстів, проектування дизайнів і навіть композування музики. Сервіси Generative AI можуть включати моделі для створення осмисленого тексту і діалогів, інструментів для генерації зображень на основі текстових запитів, а також спеціалізованих інструментів для автоматизації програмування, таких як генерація коду за запитами розробників. Ця категорія AI-сервісів сприяє не тільки збільшенню продуктивності в креативних галузях, але й дозволяє суттєво прискорити розробку програмного забезпечення.

Важливу роль в екосистемі штучного інтелекту відіграють ML Frameworks, вони MLOps, що пропонують усіляку підтримку для розробки, тренування та розгортання моделей машинного навчання. ML Frameworks є набором інструментів, що створюють фундамент для ефективної роботи з великими обсягами даних і багатошаровими нейронними мережами. Фреймворки та сервіси в цій категорії включають потужні інструменти для підготовки даних і створення моделей, що дозволяє розробникам зосередитися на побудові точних

передбачуваних систем. Вони також підтримують управління життєвим циклом моделей, включаючи їхнє відстеження, розгортання, оновлення та масштабування, що суттєво спрощує процес підтримки моделей після їх введення в експлуатацію. Більш того, хмарні середовища, що пропонуються в рамках ML Frameworks, надають безпечні та масштабовані рішення для розробки та тренування моделей, забезпечуючи повний контроль доступу та захисту даних.[22]

Можна ясно простежити, що структура організації AI як повторюється у напрямах сервісів. Це зумовлено особливістю кожного компонента AI. Саме їх можливістю застосування незалежно один від одного або залежно лише побічно.

2.2 Інтеграційні методи для міжсервісних зв'язків

Для інтеграції модулів штучного інтелекту з іншими системами та програмами застосовуються різні підходи API, що забезпечують ефективну передачу даних, керування та взаємодію між компонентами. Найбільш популярні методи міжсервісних зв'язків включають RESTful API, GraphQL, SOAP, gRPC кожен з яких має унікальні характеристики і застосовується для специфічних завдань.

Інтерфейси API відіграють ключову роль у забезпеченні взаємодії між різними модулями та системами, надаючи стандартизований спосіб обміну даними. Вони дозволяють передавати, вимагати та обробляти дані незалежно від архітектури або платформи, на якій працюють компоненти. Через API модулі можуть взаємодіяти з базами даних, зовнішніми сервісами або один з одним, забезпечуючи синхронізацію даних та виконання запитів у реальному часі.

Такі інтерфейси забезпечують структуровану передачу даних у форматі, зрозумілому всім системам, що взаємодіють. Наприклад, дані можуть передаватися у вигляді JSON, XML або інших форматів, а специфікація API

гарантує їхню правильну обробку. Це дозволяє будувати гнучкі і масштабовані системи, де кожна частина може взаємодіяти з іншими через чітко визначені протоколи, незалежно від своїх внутрішніх особенностей.[18]

RESTful API (Representational State Transfer) – це архітектурний стиль, що широко використовується для розробки API, побудований на базі HTTP. RESTful API надає просту та легковажну структуру, що робить його ідеальним для інтеграції мікросервісів та AI-модулів, забезпечуючи зручну взаємодію між компонентами через стандартні HTTP-запити (GET, POST, PUT, DELETE). RESTful API масштабується та легко адаптується до потреб високонавантажених систем, підтримує інтеграцію з контейнерами та CI/CD-пайплайнами для автоматизації. Його популярність також пов'язана з можливістю передачі даних у форматі JSON або XML, що спрощує обробку даних та покращує продуктивність.

GraphQL – це мова запитів та серверне середовище для роботи з API, розроблене Facebook. Основна перевага GraphQL полягає у гнучкості запитів: дозволяє клієнтам отримувати лише необхідні дані, що знижує навантаження на мережу та підвищує швидкість виконання запитів. GraphQL особливо корисний для інтеграції складних ієрархій даних, які часто зустрічаються в системах з великими AI-модулями, де потрібні точкові запити до великих обсягів даних. Цей метод інтеграції забезпечує зручну роботу з моделями AI, дозволяючи клієнтам вимагати вкладених структур даних, уникаючи надмірної передачі інформації, характерної для RESTful API. [19]

SOAP (Simple Object Access Protocol) – це протокол обміну повідомленнями, розроблений для забезпечення високого ступеня надійності та безпеки. Він застосовується у корпоративних системах, де необхідні суворі стандарти безпеки та формати обміну, такі як банківська та медична сфери. SOAP використовує XML передачі даних і може працювати поверх різних протоколів передачі (HTTP, SMTP). Хоча SOAP менш гнучкий і складніше у налаштуванні проти REST і GraphQL, він залишається затребуваним у випадках, коли потрібно додатковий рівень безпеки і узгодженість форматов.[20]

RESTful API, GraphQL та SOAP доповнюють один одного, забезпечуючи баланс між гнучкістю, продуктивністю та безпекою. RESTful API є стандартом архітектури, де потрібна швидка інтеграція та взаємодія між компонентами. GraphQL стає корисним, коли необхідні більш точні запити на дані та мінімізація трафіку, тоді як SOAP застосовується у тих випадках, коли безпека та стандарти обміну даними мають пріоритетне значення.

Крім RESTful API, GraphQL та SOAP для інтеграції компонентів штучного інтелекту також застосовується протокол gRPC (Google Remote Procedure Call). Цей високопродуктивний фреймворк повідомлень використовує HTTP/2 та серіалізацію даних у форматі Protocol Buffers, що робить його придатним для швидкої та ефективної взаємодії у високонавантажених розподілених системах. gRPC забезпечує низьку затримку і може обробляти багатопоточність, що корисно для підтримки AI інтенсивних обчислень та високої пропускної спроможності. [21]

gRPC особливо необхідний для міжмодульної комунікації у мікросервісних архітектурах та підтримує розробку різними мовами програмування, таких як Python, Go, Java та C++, що полегшує інтеграцію модулів AI, написаних на різних платформах.

Інтеграція цих API дозволяє модульній архітектурі AI ефективно працювати в багатокомпонентних системах, де різні модулі взаємодіють, обмінюються даними та надають високорівневі можливості автоматизації та управління, забезпечуючи стійкість та гнучкість системи.

2.3 Вибір API для бізнес-проєкту

Вибір правильного API для бізнес-проєкту та інтеграції з хмарними сервісами AI залежить від кількох ключових факторів. Перш за все, слід враховувати особливості проєкту, цілі інтеграції, вимоги до продуктивності,

безпеки та гнучкості. Також важливо звернути увагу на специфіку архітектури та взаємодії між компонентами системи.

RESTful API стане оптимальним вибором для швидкої інтеграції з хмарними сервісами та іншими компонентами системи. Він підтримує передачу даних у форматах JSON і XML, забезпечує масштабованість і легкість у реалізації. Це рішення особливо ефективне для мікросервісної архітектури та CI/CD-пайплайнів, де важливими є швидка розробка та автоматизація.

GraphQL підходить для проєктів, які вимагають гнучкого управління запитами до складних ієрархій даних. Ця технологія дозволяє оптимізувати трафік, надаючи клієнтам тільки необхідні дані, що особливо актуально при роботі з великими AI-модулями. Якщо ваш проєкт пов’язаний із великими обсягами даних або складною структурою, GraphQL буде ідеальним рішенням.

SOAP, незважаючи на його складність у налаштуванні, є відмінним вибором для проєктів, де безпека та узгодженість даних мають пріоритетне значення, наприклад, у банківській сфері або охороні здоров’я. Він забезпечує високу надійність і працює на базі строгих стандартів передачі даних через XML.

gRPC рекомендується для проєктів із високими вимогами до продуктивності, низькою затримкою та інтенсивними обчисленнями. Завдяки використанню HTTP/2 і серіалізації Protocol Buffers, gRPC забезпечує високу швидкість передачі даних і підтримує багатопоточність. Він ідеально підходить для мікросервісних архітектур та AI-додатків, які працюють із високонавантаженими розподіленими системами.

Також варто враховувати сумісність API із мовами програмування та платформами, які використовуються у вашому проєкті. Наприклад, RESTful API і GraphQL мають універсальну підтримку, тоді як gRPC надає

розширені можливості для різних мов, таких як Python, Java чи Go. Для роботи з хмарними платформами (AWS, Azure, Google Cloud) слід перевірити, чи API інтегрується із сервісами цих провайдерів, і врахувати особливості авторизації та безпеки, наприклад, підтримку токенів, секретів чи OAuth.

Таким чином, вибір API визначається специфікою вашого бізнес-проєкту та вимогами до інтеграції. RESTful API підійде для простих і масштабованих рішень, GraphQL – для складних структур даних, SOAP – для високої безпеки, а gRPC – для високонавантажених систем із низькою затримкою.

2.4 Сервіси штучного інтелекту та їх інтеграційні властивості

Не менш важливим залишається аспект того, що послуги так само діляться за специфікою застосування.

Провайдери хмарних сервісів, як говорилося раніше, націлені на покриття максимальної кількості сценарій користування. Звідси породжується безліч сервісів та підсервісів, що займаються вирішенням тих чи інших завдань.

ML-фреймворки в MLOps відіграють ключову роль у забезпеченні повного циклу роботи з моделями машинного навчання (ML) та спрощують завдання, пов'язані з їх розробкою, навчанням, розгортанням та моніторингом. фахівців за даними та інженерам, сприяючи впровадженню машинного навчання у реальні програми та бізнес-процеси.[25]

Так, наприклад, Amazon SageMaker від AWS – це комплексний інструмент для автоматизації всіх етапів життєвого циклу ML-моделей, включаючи збирання та підготовку даних, навчання, розгортання та моніторинг. SageMaker пропонує гнучку інтеграцію через API та SDK, надаючи можливість взаємодії з

ML-модулями через RESTful API, а також підтримує ефективніші протоколи gRPC та GraphQL для модульної інтеграції та високопродуктивного обміну даними. Такий підхід є особливо корисним для модульного інтегратора, оскільки дозволяє використовувати гнучкі протоколи передачі даних, зберігаючи незалежність окремих компонентів. Крім того, SageMaker підтримує AWS Step Functions, які допомагають створювати складні автоматизовані пайплайни. Ці пайплайни пов'язують процеси підготовки даних, тренування, тестування, розгортання та моніторингу, забезпечуючи безшовну інтеграцію різних ML-модулів в одному середовищі. Такий рівень автоматизації робить систему не тільки адаптованішою, а й стійкішою, оскільки знижує вплив людського фактора і мінімізує ручні помилки. SageMaker Pipelines включають функції версіонування та відстеження стану моделей, що особливо важливо для систем, що вимагають постійного оновлення моделей і ретельного управління їх життєвим циклом. Використання вбудованих інструментів для оптимізації гіперпараметрів та підтримки розподіленого навчання дозволяє значно скоротити час тренування, а також підвищити точність моделей, що є суттєвою перевагою для систем із високим навантаженням на AI-компоненти.[14]

Microsoft Azure Machine Learning пропонує безліч інструментів для інтеграції AI та ML-моделей, орієнтованих на гнучкість та масштабованість. В рамках Azure Machine Learning надається API, а також підтримка Docker-контейнерів, що дозволяє розробляти та розгортати моделі в різних інфраструктурах. Це особливо важливо для архітектури модульного інтегратора, оскільки контейнери забезпечують стандартизацію та узгодженість розгортання незалежно від цільового середовища. Azure Machine Learning підтримує інтеграцію з відкритими фреймворками, такими як TensorFlow та PyTorch, а також іншими сервісами Azure, такими як Azure DevOps та Azure Kubernetes Service (AKS), що розширює можливості автоматизації та масштабування. Однією з ключових переваг Azure Machine Learning є підтримка взаємодії через API RESTful, а також через gRPC та GraphQL, що робить комунікацію між модулями гнучкою та легко масштабованою. Це також дозволяє інтегрувати з CI/CD

пайплайни, такі як GitHub Actions та GitLab CI/CD, що дозволяє налаштувати безперервний цикл тестування, оновлення та розгортання моделей. Такі пайплайни забезпечують підтримку життєвого циклу моделей та дозволяють легко відстежувати зміни та версії. Azure також надає інструменти для забезпечення прозорості та зрозумілості моделей, що є важливою вимогою для багатьох індустрій, в яких використовуються алгоритми AI, що вимагають нагляду та дотримання нормативних вимог. Наприклад, вбудовані функції для пояснення та інтерпретації моделей допомагають фахівцям отримувати уявлення про те, як і чому модель дійшла певних результатів, що покращує контроль та довіру до системи.[14]

На платформі Google Cloud Platform аналогічне завдання виконує сервіс Vertex AI, який надає зручні та потужні інструменти для керування всіма етапами MLOps. Vertex AI інтегрує дані та моделі в єдиному робочому просторі, де користувачі можуть керувати пайплайном без необхідності перемикатися між різними сервісами. Завдяки AutoML, вбудованому в Vertex AI, користувачі з мінімальними знаннями в машинному навчанні можуть налаштовувати та тренувати моделі, що робить систему доступнішою для широкого кола бізнес-користувачів. Це дозволяє значно скоротити час на створення та налаштування моделей, що критично важливо для бізнес-вимог, що швидко змінюються. Vertex AI також підтримує TPU (Tensor Processing Unit), що забезпечує високу обчислювальну потужність для виконання завдань глибокого навчання та зменшує час тренування. Інтеграція Vertex AI з пайплайнами CI/CD спрощує процес оновлення та розгортання моделей, а вбудовані засоби моніторингу якості даних та продуктивності моделей допомагають підтримувати стабільність системи. Для інтеграції, що масштабується, доступні як стандартні RESTful API, так і gRPC, що полегшує високопродуктивний обмін даними. Це особливо важливо для систем, що працюють з великою кількістю даних та потребують постійного контролю якості та продуктивності моделей. Інтеграція з Terraform та Kubernetes робить Vertex AI зручним для управління

інфраструктурою, забезпечуючи високий рівень автоматизації та контролю за розподілом ресурсів.[15]

Generative AI сервіси. Generative AI інтегруються через набір інструментів, що дозволяють керувати генеративними моделями для створення контенту, синтезу даних та інтерактивних програм. Ці сервіси пропонують API, CI/CD пайплайни та DevOps-інструменти для легкої інтеграції моделей до додатків та інфраструктури.

AWS включає Generative AI в Amazon SageMaker і Bedrock. SageMaker JumpStart прискорює налаштування моделей, підтримуючи взаємодію з ними через RESTful API та за потреби через високопродуктивні протоколи gRPC та GraphQL. JumpStart пропонує більше 150 готових моделей з відкритим кодом, які можна розгорнути для створення тексту, зображень чи даних. Amazon Bedrock, інтегрований з Foundation Models (FM), надає доступ до моделей від Amazon та партнерів через RESTful API, що спрощує налаштування та розгортання Generative AI для бізнес-завдань без необхідності у власних потужностях. Amazon Q використовує бази даних для відповіді на запитання, а Amazon Q Developer інтегрується з IDE для створення коду. Інтеграція з CI/CD пайплайнами, такими як GitHub Actions та Jenkins, автоматизує оновлення моделей. DevOps-інструменти, наприклад AWS CloudFormation і Terraform, згадуваний раніше, допомагають керувати інфраструктурою, оптимізуючи розгортання AWS.[14]

Microsoft Azure пропонує Azure OpenAI Service з підтримкою OpenAI моделей, таких як GPT-4, інтегруючи їх через RESTful API для чат-ботів та генераторів контенту. Azure OpenAI Service, що тісно пов'язаний з Azure Cognitive Services, додає можливості обробки тексту та зображень в одному робочому процесі. API підтримує автоматичне масштабування та налаштування під конкретні завдання. Azure DevOps та GitHub Actions автоматизують розгортання, а підтримка Docker та Kubernetes спрощує контейнеризацію моделей. Azure Kubernetes Service (AKS) надає гнучкість керувати контейнеризированными моделями.[14]

Google Cloud Platform пропонує Vertex AI Generative AI Studio з доступом до BERT та T5. Studio забезпечує API-інтерфейси для налаштування та розгортання генеративних моделей через HTTP(S)-запити для обміну даними та управління моделями в режимі реального часу. GCP підтримує інструменти CI/CD, такі як Cloud Build та GitLab CI/CD, для автоматизації оновлень та контролю версій моделей, а також Terraform та Kubernetes для керування інфраструктурою. Підтримка Google Kubernetes Engine (GKE) та TPU прискорює навчання та виведення глибоких моделей, особливо корисних для завдань з інтенсивними обчисленнями.

AI/ML послуги. AI/ML сервіси Amazon Web Services (AWS) пропонують широкий спектр інструментів для розробки інтелектуальних додатків. Amazon Comprehend виконує обробку природної мови (NLP), дозволяючи аналізувати тексти та отримувати ключові поняття, імена та події, що автоматизують аналіз великих обсягів даних, наприклад клієнтських відгуків. Amazon Translate забезпечує машинний переклад тексту для глобальної аудиторії, а Textract витягує дані зі сканованих документів. Amazon Lex створює чат-ботів з підтримкою розпізнавання мови, а Polly перетворює текст на мовлення. Amazon Transcribe автоматизує перетворення аудіо в текст, а Rekognition аналізує зображення та відео, розпізнаючи об'єкти та обличчя. Kendra допомагає ефективно шукати інформацію у неструктурованих даних, а Personalize створює персональні рекомендації. AWS DeepRacer є навчальною платформою вивчення методів посиленого обучения.[14]

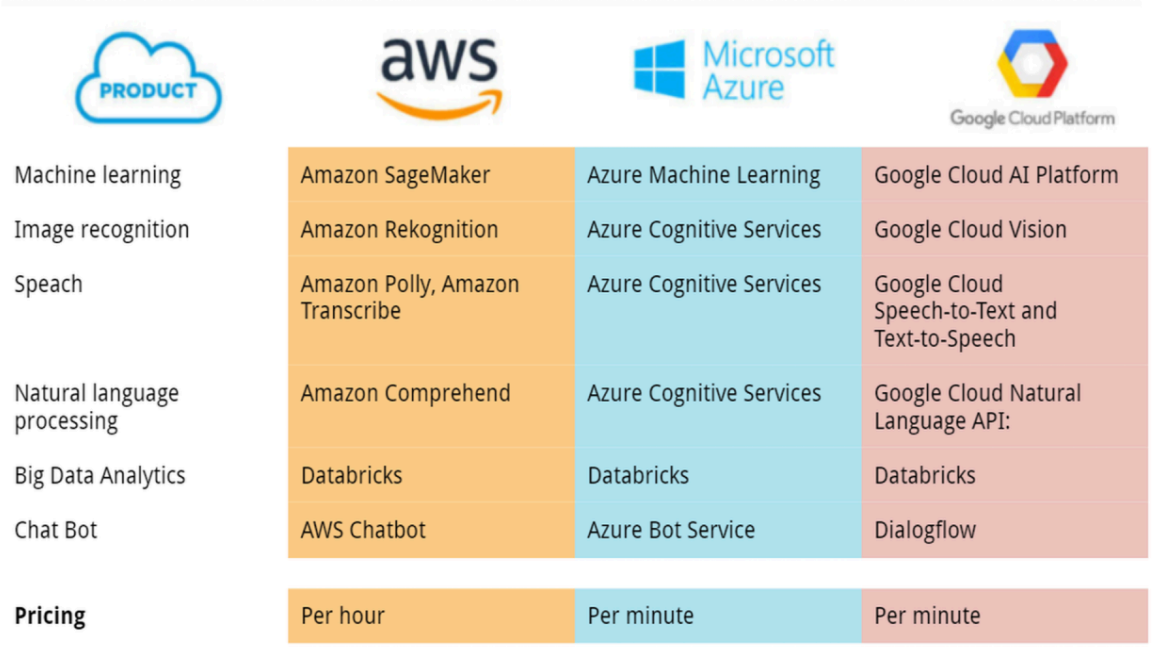
Microsoft Azure пропонує широкий вибір AI/ML сервісів для створення модульних архітектур. Azure Cognitive Services включає API для аналізу зображень, мовлення і тексту, в той час як Azure Text Analytics дозволяє отримувати ключові фрази і аналізувати тональність текстів. Azure Translator забезпечує переклад, а Form Recognizer автоматизує вилучення даних із документів. Azure Bot Services та Speech Services дозволяють створювати чат-боти та голосові інтерфейси, а Video Indexer аналізує відео. Azure Custom Vision створює моделі для розпізнавання зображень, а Face API дозволяє аналізувати

обличчя. Azure Personalizer використовує посилене навчання для персоналізації рекомендацій.[14]

Google Cloud Platform (GCP) надає рішення щодо побудови систем AI/ML із модульним підходом. Google Cloud Natural Language виконує текстовий аналіз та Cloud Translation пропонує інструменти для машинного перекладу. Document AI розпізнає текст та дані у документах, а Dialogflow створює чат-боти та голосові інтерфейси. Сервіси Text-to-Speech та Speech-to-Text Google Cloud забезпечують автоматизацію обробки голосових даних, а Vision AI розпізнає об'єкти та обличчя на зображеннях. Recommendations AI допомагає створювати персональні рекомендації. GCP підтримує DevOps-інтеграцію та CI/CD процеси, що спрощує розгортання та управління моделями AI з використанням Cloud Build та Terraform.[15]

2.5 Порівняння цільових ШІ сервісів між AWS, Azure, GCP

AI сервіси можуть бути класифіковані за цільовим призначенням залежно від завдань, які вони вирішують, та сфери застосування. Такі послуги можуть охоплювати різні області, включаючи обробку природної мови (NLP), розпізнавання мови та зображень, машинне навчання, аналіз великих даних та створення чат-ботів. Кожна з цих категорій включає спеціалізовані інструменти, які допомагають автоматизувати процеси, покращувати взаємодію з користувачами та отримувати цінні інсайти з даних. Залежно від потреб бізнесу або конкретного завдання, AI сервіси можуть бути адаптовані для роботи з текстами, зображеннями, відео, голосовими інтерфейсами та багатьма іншими. (Рис.2.2)



PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Machine learning	Amazon SageMaker	Azure Machine Learning	Google Cloud AI Platform
Image recognition	Amazon Rekognition	Azure Cognitive Services	Google Cloud Vision
Speach	Amazon Polly, Amazon Transcribe	Azure Cognitive Services	Google Cloud Speech-to-Text and Text-to-Speech
Natural language processing	Amazon Comprehend	Azure Cognitive Services	Google Cloud Natural Language API:
Big Data Analytics	Databricks	Databricks	Databricks
Chat Bot	AWS Chatbot	Azure Bot Service	Dialogflow
Pricing	Per hour	Per minute	Per minute

Рисунок 2.2 Класифікація ШІ сервісів

*ML.*Amazon SageMaker (AWS) пропонує повний цикл розробки моделей машинного навчання. Це включає етапи тренування, тестування і розгортання. Сервіс надає вбудовані алгоритми та підтримує інтеграцію з Jupyter Notebooks, що спрощує процес розробки для користувачів. SageMaker також орієнтований на масштабованість та автоматизацію, дозволяючи розробляти та впроваджувати рішення з мінімальними витратами часу.

Azure Machine Learning від Microsoft відрізняється глибокою інтеграцією з іншими сервісами Azure та підтримує підхід MLOps, що робить його потужним інструментом для автоматизації робочих процесів у машинному навчанні. Вбудована підтримка AutoML дозволяє створювати моделі із мінімальним ручним втручанням. Azure Machine Learning також є гнучким у виборі фреймворків: користувачі можуть працювати з TensorFlow, PyTorch та іншими популярними бібліотеками, що робить сервіс універсальним для різних завдань.

Google Cloud AI Platform є потужним інструментом для створення масштабованих моделей машинного навчання. Завдяки інтеграції з TensorFlow та підтримці AutoML користувачі можуть автоматизувати та прискорювати розробку моделей. Платформа тісно пов'язана з BigQuery ML, що дозволяє обробляти та

аналізувати великі обсяги даних безпосередньо всередині екосистеми Google. Це робить AI Platform оптимальним вибором для завдань, які потребують високої продуктивності та роботи з великими даними.

*Image Recognition.*Amazon Rekognition надає потужні інструменти для розпізнавання об'єктів, облич та тексту на зображеннях та відео. Це рішення дозволяє не тільки ідентифікувати об'єкти та витягувати текст із зображень, а й виконувати аналіз відео в реальному часі. Такий підхід корисний для завдань безпеки (наприклад, відеоспостереження) та аналізу контенту. Rekognition також підтримує розпізнавання осіб з високою точністю, включаючи вилучення демографічної інформації (стаття, вік, емоції).

Azure Cognitive Services (Vision API) включає різні функції для роботи з зображеннями і відео. Цей сервіс дозволяє проводити розпізнавання об'єктів, текстів з використанням OCR (оптичне розпізнавання символів), а також аналізувати зображення та відео для отримання характеристик, таких як колір, текстури або форма об'єктів. Також підтримується детектування осіб, що може бути корисним для додатків у галузі безпеки та медицини. На відміну від Rekognition, Azure надає більші можливості для інтеграції з іншими хмарними сервісами Microsoft.

Google Cloud Vision у свою чергу пропонує потужні можливості для аналізу зображень. Включаючи розширені можливості OCR, класифікацію об'єктів та їх характеристик, а також аналіз емоцій, що дає змогу глибше розуміти контекст зображення. Google Cloud Vision також надає точні алгоритми для розпізнавання та класифікації зображень, що робить його відмінним вибором для програм, які потребують високої точності в ідентифікації об'єктів, таких як аналітика продуктів або модерація контенту.

Speech. AWS надає два ключові сервіси для роботи з промовою: Amazon Polly та Amazon Transcribe. Polly займається генерацією мови з тексту та підтримує безліч мов, що дозволяє створювати додатки з голосовими інтерфейсами, що підтримують користувачів по всьому світу. Transcribe використовується для

транскрибації аудіо до тексту, що робить сервіс корисним для автоматизації обробки аудіо та відеоматеріалів. AWS забезпечує високу точність розпізнавання мовлення та має інструменти для адаптації моделей під специфічні завдання.

Azure Cognitive Services (Speech) пропонує комплексне рішення для роботи з мовою, включаючи розпізнавання мови (Speech-to-Text), синтез мови (Text-to-Speech), а також можливість перекладу мовлення в реальному часі. На відміну від інших сервісів, Azure підтримує створення власних моделей, які можна навчати для специфічних областей або особливостей. Це робить сервіс ідеальним для побудови високоякісних голосових інтерфейсів та багатозадачних додатків, де важлива точність та адаптація під різні акценти та умови використання.

Google Speech-to-Text та Text-to-Speech забезпечують високу точність розпізнавання мови та синтезу голосу. Google пропонує потужні алгоритми для розпізнавання мови за допомогою нестандартних словників, що корисно для нішевих додатків, де важливо враховувати специфічну лексику. Також є кастомізація голосів, що дозволяє створювати унікальні голосові інтерфейси. Ці можливості роблять сервіси Google ідеальними для програм, де необхідне точне та адаптивне розпізнавання мовлення з можливістю створення персоналізованих голосів.

Natural Language Processing. Amazon Comprehend - це сервіс, що спеціалізується на обробці та аналізі тексту. Він включає інструменти для розпізнавання емоцій, що допомагає визначати настрої і почуття в тексті. Також Comprehend виконує вилучення ключових слів та категоризацію тексту, що дозволяє ефективно організовувати та сортувати великі обсяги даних. За допомогою функції отримання сутностей сервіс виділяє ключові елементи тексту, такі як імена людей, організації та інші важливі дані, що корисно для автоматизованої обробки інформації та аналізу контенту.

Azure Cognitive Services (Text Analytics) пропонує кілька корисних інструментів для роботи з текстами. Він підтримує розпізнавання тональності тексту, що дозволяє аналізувати, чи позитивний, негативний чи нейтральний

налаштування міститься у повідомленні. Також Azure дозволяє отримувати ключові фрази та іменовані сутності, такі як компанії, місця та інші важливі терміни. Важливою особливістю є функція перекладу тексту, що дозволяє інтегрувати багатомовні можливості у додаток. Ці інструменти можуть бути корисними у завданнях, пов'язаних з аналізом відгуків, соціальних мереж або інших текстових даних.

Google Cloud Natural Language API пропонує потужні інструменти для аналізу та обробки тексту. Він включає аналіз синтаксису, що допомагає виявляти граматичні структури і відносини між словами в реченнях. Також сервіс виконує добування сутностей, що дозволяє визначати ключові об'єкти та терміни в тексті, а також їх класифікацію. На відміну від інших сервісів, Google надає підтримку більшої кількості мов, що робить її придатною для міжнародних проектів, де необхідно працювати з текстами різними мовами.

Big Data Analytics. Всі три великі хмарні провайдери - AWS, Azure і GCP - використовують Databricks як ключовий інструмент для роботи з великими даними. Databricks надає платформу для аналізу, обробки та моделювання великих обсягів даних за допомогою Apache Spark. Однак кожен із провайдерів має свої особливості інтеграції з іншими сервісами для оптимізації роботи з великими даними.

AWS надає глибоку інтеграцію Databricks з такими сервісами, як Amazon EMR (Elastic MapReduce) та S3. EMR використовується для обробки та аналізу великих даних з використанням Hadoop та Spark, а S3 служить для зберігання даних у хмарі, що дозволяє ефективно зберігати та витягувати дані для аналізу. У зв'язку з Databricks це забезпечує потужні можливості для масштабованої обробки даних та побудови аналітичних рішень на основі великих даних.

Azure інтегрує Databricks з Azure Data Lake, що дозволяє ефективно керувати і зберігати великі обсяги даних у сховищі, що високо масштабується. Azure Data Lake є платформою для зберігання даних у їх сирому вигляді, що дозволяє працювати з неструктурованими даними, такими як журнали, зображення або текстові файли. У поєднанні з Databricks це надає потужну

платформу для аналітики та машинного навчання, здатну обробляти дані на високих швидкостях.

GCP також використовує Databricks для обробки великих даних, але з особливим акцентом на інтеграцію з BigQuery. BigQuery - це повністю керована аналітична база даних, призначена для виконання запитів щодо величезних обсягів даних. Спільне використання Databricks з BigQuery дозволяє оптимізувати обробку даних та їх аналіз, використовуючи можливості масштабованого зберігання та аналізу даних у реальному часі. Це рішення особливо корисне для обробки великих обсягів структурованих даних та аналітики у хмарі.

Таким чином кожен провайдер пропонує унікальні підходи та інструменти для роботи з великими даними, пропонуючи інтеграцію з різними сервісами для підвищення ефективності обробки та аналізу даних.

Chat Bot. AWS Chatbot надає інтеграцію з Amazon Lex, що є потужним інструментом для створення чат-ботів, що використовують обробку природної мови. Цей сервіс дозволяє розробляти розмовні інтерфейси для програм із можливістю налаштування різних повідомлень та взаємодій через месенджери та інші канали. Завдяки тісній інтеграції з іншими сервісами AWS, такими як Lambda, S3 та CloudWatch, AWS Chatbot дозволяє легко розгортати та керувати чат-ботами, а також налаштовувати автоматичні повідомлення та дії на основі запитів користувачів.

Azure Bot Service пропонує повний набір інструментів для створення, тестування та керування чат-ботами. ключовими особливостями є інтеграція з LUIS (Language Understanding Intelligent Service), який дозволяє розробляти боти, здатні розуміти та інтерпретувати природну мову, виявляючи наміри користувачів та витягуючи сутності із запитів. Це робить Azure Bot Service ідеальним для створення складних, інтелектуальних чат-ботів для бізнесу та спілкування з клієнтами.

Dialogflow від Google (частина Google Cloud) пропонує гнучкі інтерфейси для створення чат-ботів, що підтримують Natural Language Understanding (NLU) для більш точного розуміння запитів користувачів. Dialogflow дозволяє будувати

високоякісні чат-боти, які можуть бути інтегровані з різними платформами та програмами, включаючи Google Assistant, Facebook Messenger та інші. Цей сервіс також підтримує створення інтелектуальних розмовних інтерфейсів з можливістю додавання контексту та персоналізації взаємодій, що робить його придатним для різних сценаріїв використання від простих ботів до складних віртуальних помічників.

2.6 Сервісне ціноутворення

Як відомо, найчастіше сервіси підключаються до основного коду за допомогою запитів API. Проте, її менш важливим пунктом є ціноутворення хмарних сервісів.

Платформа Azure пропонує Azure Machine Learning, де вартість залежить від використання віртуальних машин та специфічних інструментів для навчання та деплой моделей. Services розраховуються за обсягом повідомлень, що робить сервіс зручним для систем, що масштабуються в сфери обслуговування клієнтів та корпоративних додатків.

AWS також має великий набір AI-сервісів, таких як Amazon SageMaker для машинного навчання, де вартість залежить від потужностей для тренування та передбачень. , що відповідають за обробку природної мови та перекладу, надають гнучкість у ціноутворенні, оскільки оплата також йде за кількість викликів API, що вигідно для компаній із нерегулярним використанням цих сервісів. Вартість Amazon Rekognition залежить від кількості проаналізованих зображень та відео, що підходять для програм безпеки та аналітики.

Google Cloud Platform (GCP) пропонує Google Cloud Machine Learning та Cloud AutoML, за вартістю схожі з аналогічними сервісами AWS та Azure, оплата йде за потужність та обсяг даних. Такі послуги, як Google Natural Language API та Vision AI, оплачуються за використання API, що дозволяє контролювати витрати в залежності від частоти дзвінків. Google пропонує великі можливості для

інтеграції з іншими сервісами Google, у тому числі BigQuery для аналізу великих даних.

Таким чином, ціни на AI-сервіси в цих хмарних платформах в основному залежать від потужностей та кількості викликів API, що дає гнучкість у витратах і підходить як для стартапів, так і для великих організацій.

3 ПРОЕКТУВАННЯ АРХІТЕКТУРИ МОДУЛЬНОГО ІНТЕГРАТОРА AI

3.1 Проектування принципової архітектури AI/ML конектора

Проектування принципової модульної архітектури системи інтеграції штучного інтелекту (AI) починається з визначення ключових компонентів, необхідних для ефективного виконання завдань та досягнення бізнес-цілей. Основними елементами такої архітектури є сервіси AI/ML та контролер. AI/ML сервіси забезпечують функціонал для обробки даних та застосування алгоритмів машинного навчання, тоді як конектор служить сполучною ланкою, що управляє логікою взаємодії між різними сервісами.

Конектор не обмежується лише AI/ML сервісами; він також інтегрує інші мікросервіси, забезпечуючи цілісне рішення обробки запитів та управління потоками даних. Це створює основу для комплексної системи, де модулі можуть динамічно взаємодіяти один з одним, забезпечуючи гнучкість і адаптивність до вимог, що змінюються.

Модульна архітектура дозволяє організувати розгортання компонентів на різних інфраструктурах, чи це локальні сервери чи хмарні рішення. Це забезпечує високий рівень масштабованості та гнучкості, дозволяючи бізнесу швидко адаптуватися до змін у середовищі. створення продуманої архітектури, здатної задовольняти поточні та майбутні потреби организации.(рис.1.3)

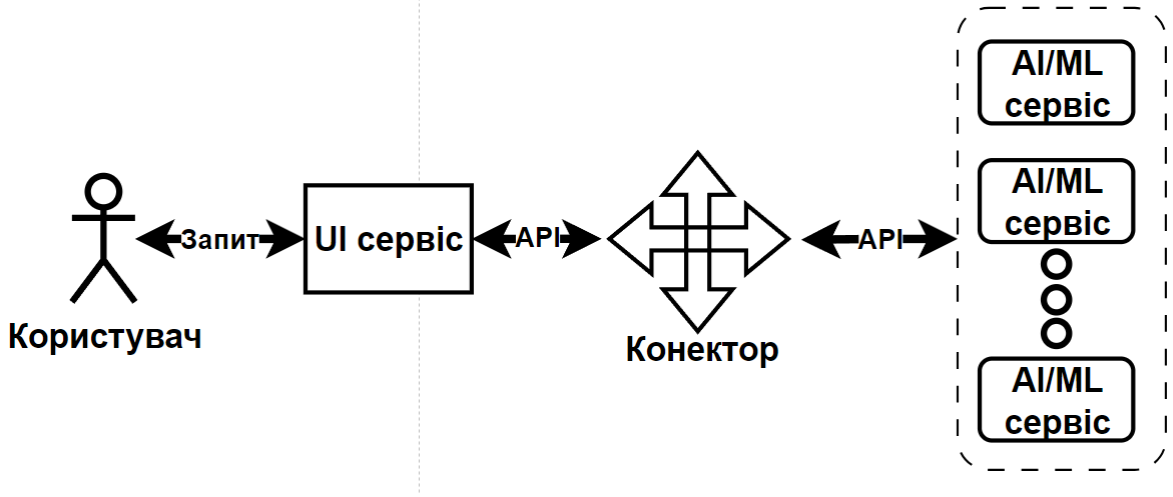


Рисунок 3.1 Принципова схема модульного інтегратора

Також важливим аспектом проектування є розширення системи. Модульна архітектура повинна передбачати можливість додавання нових сервісів та компонентів без їх переробки. Це важливо для забезпечення стійкості до змін, що робить систему більш стійкою та життєздатною у довгостроковій перспективі.

Важливим аспектом проектування архітектури є те, що має бути між користувачем та конектором. Якщо конектор виконує роль перемикача вибору AI/ML сервісів, необхідно передбачити механізми оцінки запитів. Цією оцінкою може бути обслуговування машинного навчання, якому передається тіло запиту через коннектор.

1. Запит користувача перенаправляється конектором у сервіс машинного навчання, навчений виявлення патернів.
2. Сервіс машинного навчання аналізує дані, визначає закономірності та формує інтерпретований наказ для подальшого сервісу через конектор.
3. Конектор приймає запит від ML-сервісу та передає завдання відповідному AI/ML сервісу.
4. Сервіс AI/ML обробляє запит і повертає користувачеві відповідь через конектор. (Рис.3.2)

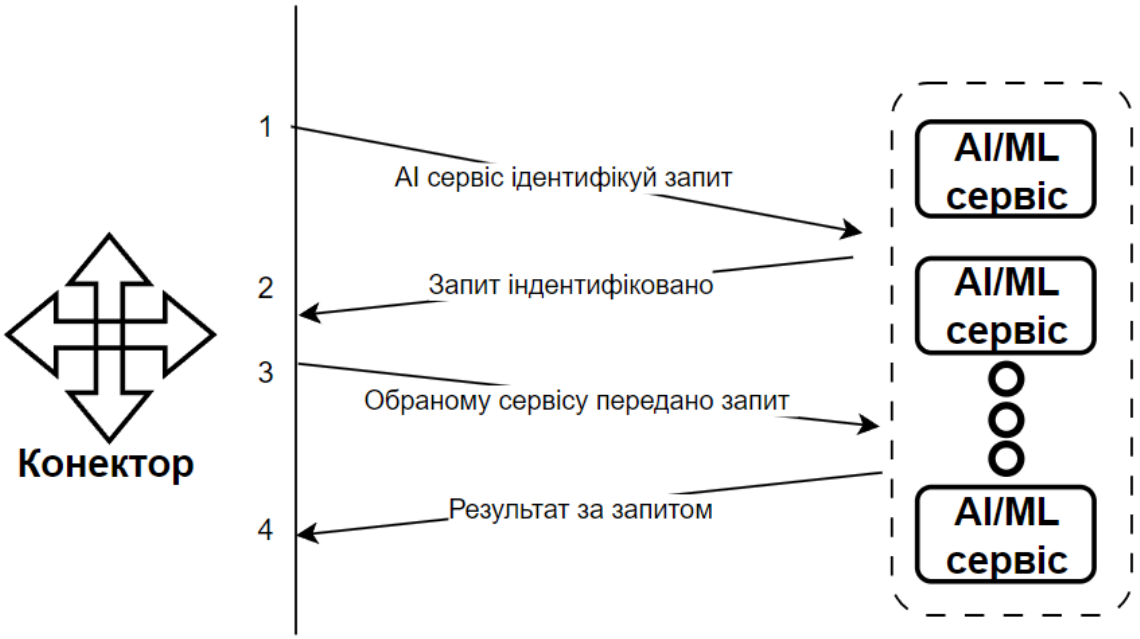


Рисунок 3.2 API з'єднання ШІ конектора

Проектування конектора - це процес, що вимагає врахування багатьох факторів. Насамперед важливо визначити цільове призначення інтеграції, виходячи з сервісів AI, які потрібно підключити. Але також не менш важливим є аспект безпеки.

Мови програмування конектора. При виборі мови програмування для конектора слід звернути увагу на кілька популярних варіантів, таких як Python, Java, JavaScript, C# та Go. Кожна з цих мов має свої унікальні особливості, які можуть вплинути на кінцевий вибір в залежності від специфіки сервісів, що використовуються.

Python виділяється як одна з найкращих мов для роботи з AI/ML сервісами завдяки своїй простоті, читаності та багатій екосистемі бібліотек. Це дозволяє розробникам швидко створювати прототипи та впроваджувати рішення побудови моделей, що робить його ідеальним вибором для завдань, пов'язаних з видаленням інформації, таких як автоматизація аналізу текстів і розпізнавання зображень за допомогою Amazon Rekognition або Google Vision AI. високонавантажених системах, і в таких випадках можливе використання оптимізованих бібліотек або написання критичних компонентів більш продуктивними мовами.

Java також є чудовим вибором для розробки AI-конекторів, особливо для сервісів, що потребують високої продуктивності та надійності, таких як Azure Machine Learning та Amazon Kendra. Портування Java дозволяє розробляти програми, які працюватимуть на різних платформах, а підтримка багатопоточності робить його особливо придатним для роботи з великими обсягами даних. Однак, як і у випадку з Python, розробка Java може зайняти більше часу через складніший синтаксис і структуру.

JavaScript, особливо в контексті Node.js, є потужним інструментом створення серверних програм, які можуть ефективно взаємодіяти з API AI/ML сервісами, такими як Microsoft Azure OpenAI Service та Google Cloud Translation. JavaScript дозволяє створювати асинхронні програми, що ідеально підходять для обробки запитів у сервіси та інтеграції їх у веб-додатки. Тим не менш, слід враховувати, що JavaScript може мати обмеження щодо продуктивності в порівнянні з іншими мовами.

C# є ще одним сильним кандидатом, особливо для інтеграції із сервісами Microsoft, такими як Azure Bot Services та Azure Machine Learning. C# пропонує надійну платформу для створення продуктивних програм, а його тісна інтеграція з Azure дозволяє зручно розробляти та розгортати рішення, використовуючи переваги екосистеми Microsoft. Однак слід зазначити, що екосистема для C# може бути менш гнучкою порівняно з Python.

Go, відомий своєю високою продуктивністю та простотою, також може використовуватися для розробки AI-конекторів. Він відмінно підходить для створення масштабованих додатків, які можуть ефективно обробляти безліч запитів одночасно. розвинена, що може ускладнити реалізацію деяких функцій.

Безпека. Обговорюючи безпеку, слід зазначити, що конфіденційність даних при взаємодії з API є критично важливою, особливо в контексті інтеграції AI-сервісів. Аутентифікація користувачів та програм забезпечує перевірку ідентичності перед доступом до ресурсів. Найбільш поширені методи автентифікації включають OAuth 2.0 та API-ключі, які допомагають контролювати доступ до API та мінімізувати ризик несанкціонованого використання.

Шифрування даних як на етапі передачі (наприклад, з використанням HTTPS), так і при зберіганні захищає інформацію від перехоплення та доступу зловмисників. Симетричне та асиметричне шифрування може використовуватися для захисту конфіденційних даних, забезпечуючи їх цілісність та конфіденційність. Важливо регулярно оновлювати та керувати ключами шифрування, щоб уникнути потенційних уразливостей. Також слід впроваджувати моніторинг та аудит доступу до API, що допоможе виявити аномалії та своєчасно реагувати на можливі загрози.

Моніторинг та дебаг. Моніторинг є невід'ємною частиною процесу розробки та експлуатації конекторів, особливо у контексті інтеграції AI-сервісів. Він дає можливість не тільки відстежувати стан додатків, а й значно спрощує процес налагодження, допомагаючи швидко виявляти та усувати помилки. Ефективний моніторинг може суттєво прискорити процеси CI/CD, надаючи командам своєчасні повідомлення про критичні інциденти. А також дозволяє розробникам відстежувати метрики продуктивності, такі як час відгуку API, кількість помилок та рівень навантаження на систему. Це сприяє більш швидкому та точному дебаггінгу, оскільки надає важливі дані про стан системи та її взаємодію з іншими компонентами. І що не менш важливо, багато сучасних моніторингових рішень інтегруються з месенджерами, такими як Slack, Microsoft Teams, Telegram і Discord, що дозволяє розробникам та операційним командам миттєво реагувати на проблеми у своїх каналах зв'язку без надання безпосередніх доступів до інфраструктурного середовища.

Існує кілька варіантів для моніторингу, які можуть бути використані залежно від конкретних потреб проекту:

1. Кодово-інтеграційний.
2. Хмарно сервісний.
3. Сторонні послуги.

Інтеграційний моніторинг має на увазі необхідність додавання певного коду для підключення до сервісів моніторингу. Прикладом такого рішення є Azure

Application Insights, який вимагає від розробників інтеграції спеціального коду для передачі даних про продуктивність програми та її стан. Це може включати реєстрацію ключових подій, таких як помилки, затримки в обробці запитів або зависання системи. Отримані дані дозволяють глибше аналізувати поведінку програми, виявляти вузькі місця та оптимізувати продуктивність. Однак такий підхід може додати складності у розробку, оскільки потребує додаткового часу для реалізації та тестування інтеграції.

Хмарні моніторингові сервіси, такі як AWS CloudWatch, Google Cloud Monitoring та Azure Monitor пропонують потужні інструменти для відстеження продуктивності та стану програм. Ці сервіси дозволяють збирати метрики та логи, а також надають можливість рендерингу даних у реальному часі. Наприклад, AWS CloudWatch може відстежувати стан API, збираючи дані про затримки та помилки. Це дозволяє швидко виявляти та усувати проблеми, мінімізуючи час простою. Хмарні рішення також забезпечують автоматичне масштабування, що є важливим фактором для AI-сервісів, що потребують динамічного настроювання ресурсів залежно від навантаження.

Крім того, ці сервіси дозволяють налаштовувати сповіщення про критичні події. Це означає, що команда розробки може миттєво отримувати сповіщення про проблеми, що дозволяє швидше реагувати на інциденти та підтримувати високу якість обслуговування.

Сторонні рішення, такі як ELK Stack, Prometheus та Grafana, надають гнучкі інструменти для моніторингу та аналізу продуктивності систем. Ці інструменти можна розгорнути як на серверах, так і в контейнерах Docker, що робить їх придатними для різних архітектур. Наприклад, ELK Stack дозволяє збирати, аналізувати та візуалізувати логи, що є важливим аспектом діагностики проблем. За допомогою Prometheus можна відстежувати метрики та отримувати сповіщення про події, Grafana надає потужні засоби візуалізації даних.

Інтеграція сторонніх сервісів моніторингу дозволяє командам точніше ідентифікувати помилки та вузькі місця у продуктивності систем. Вони пропонують можливість надсилання повідомлень через месенджери, що дозволяє

розробникам оперативно реагувати на інциденти та мінімізувати час простою. Крім того, дані, зібрані сторонніми інструментами, можуть використовуватись для аналізу продуктивності та оптимізації процесів CI/CD, що значно покращує загальну ефективність розробки.

На закінчення, моніторинг відіграє ключову роль у розробці та експлуатації конекторів, забезпечуючи контроль за станом додатків та спрощуючи процес налагодження. Інтеграційні рішення, хмарні послуги та сторонні інструменти моніторингу надають розробникам широкий спектр можливостей для ефективного управління програмами. Це дозволяє не тільки швидко реагувати на проблеми, але й покращувати якість розробки та обслуговування, забезпечуючи надійну роботу вбудованих AI-сервісів. За допомогою моніторингу можна прискорити процеси CI/CD, підвищити продуктивність та забезпечити безпеку даних, що в результаті призводить до більш успішних проектів та задоволення потреб користувачів.

Вибір обчислювальної бази. Конектор може бути розміщений різними способами, кожен з яких має унікальні переваги та недоліки.

Перший варіант – розміщення на локальних серверах компанії. Це дозволяє забезпечити повний контроль над даними та інфраструктурою, мінімізуючи ризики, пов'язані з безпекою та конфіденційністю інформації чи VirtualBox, може спростити розгортання та управління конектором, дозволяючи ефективно використовувати доступні ресурси. завдяки близькості до даних, а також можуть інтегруватися з існуючою інфраструктурою підприємства, що сприяє поліпшення сумісності.

Іншим варіантом є використання хмарних послуг. Конектор можна розмістити на віртуальних машинах, що надаються такими хмарними платформами як AWS EC2, Azure VM або GCP Compute Cloud. Це рішення дозволяє використовувати потужність хмарних ресурсів та гнучкість для масштабування, що робить конектор більш ефективним при динамічній роботі. Крім того, він може легко адаптуватися до змінних вимог і збільшення

навантаження. Однак важливо враховувати можливі витрати на хмарні ресурси та питання безпеки, пов'язані з передачею даних у хмару.[23]

Контейнерні технології, такі як Kubernetes (K8s) та Docker, також надають можливість розміщення конектора на віртуальних. Кубернет дозволяє розгортати, керувати та масштабувати контейнеризовані програми у хмарному середовищі. Він забезпечує автоматичний розподіл навантаження, керування станом та самовідновлення, що робить його ідеальним для роботи з мікросервісами. Розміщення конектора в Kubernetes забезпечує гнучкість та адаптивність інфраструктури, що дозволяє швидко реагувати на зміни навантаження та забезпечувати високу доступність додатків.

І останнім третім варіантом є використання serverless архітектури, як-от AWS Lambda, Azure Functions або Google Cloud Functions. значно спрощує розгортання та управління програмами, дозволяючи розробникам зосередитись на бізнес-логіці, а не на інфраструктурі. Проте слід зазначити, що сервери однаково використовуються, але вони керуються хмарним провайдером.[24]

Таким чином, вибір способу розміщення конектора залежить від конкретних вимог бізнесу, доступного бюджету, рівня необхідної безпеки та гнучкості в управлінні ресурсами.

3.2 Вибір та оцінка архітектурних патернів для ШІ

При проектуванні сервісних зв'язків важливо враховувати цілі та бюджет проекту. Це дозволить вибрати найбільш підходящий підхід, що відповідає вимогам обчислювальної потужності конектора, так і доступності програмного забезпечення для користувачів. У результаті виникли дві архітектури, що замінили монолітні програми, які прагнули поєднати всі функції в одному.

Так наприклад, архітектура запит-відповідь передбачає взаємодію, при якій один компонент, що діє як клієнт, відправляє запит іншому компоненту – серверу – і чекає на відповідь. Цей синхронний підхід вимагає від клієнта припинення

роботи до завершення обробки запиту сервером. Передбачуваність та послідовність обміну даними, що спрощує налагодження та тестування. Однак, такі системи можуть зіткнутися з проблемами масштабування, особливо при високому навантаженні та стають чутливими до затримок, оскільки кожен запит вимагає негайної відповіді. При інтенсивному обміні даними продуктивність може значно знижуватися, оскільки клієнт очікує завершення кожної операції.

На відміну від архітектури запит-відповідь подієво-орієнтована архітектура ґрунтується на асинхронному обміні повідомленнями між компонентами. У цьому підході події вирушають та обробляються незалежно від інших процесів. Компоненти обробляють події без очікування негайної відповіді, що забезпечує більшу гнучкість та масштабованість системи. Такий підхід полегшує управління навантаженнями, оскільки елементи можуть обробляти події у своєму темпі. Ця архітектура чудово підходить для складних систем, таких як IoT або системи моніторингу, що потребують високої адаптивності. Однак вона має і недоліки, такі як складність налагодження, оскільки події можуть оброблятися у будь-який час, а також необхідність керування станом та послідовністю подій.

Архітектура запиту-відповіді відрізняється тим, що вона не вимагає допоміжних сервісів для обробки запитів, оскільки вся взаємодія відбувається у синхронному режимі між клієнтом та сервером. Це спрощує розробку та тестування, але може обмежувати масштабованість. Події-орієнтована архітектура, навпаки, може задіяти додаткові послуги для паралельної обробки подій, що дозволяє ефективно управляти навантаженням і збільшує гнучкість системи. Вона чудово підходить для складних сценаріїв, де необхідно обробляти велику кількість подій у реальному часі. Архітектура запит-відповідь підходить для сценаріїв, де важлива передбачуваність та простота у налагодженні. Однак за високого навантаження вона може стати вузьким місцем. У той час як архітектурно-орієнтована архітектура надає можливість більш складних систем, підтримуючи асинхронну обробку і гнучкість, що ідеально підходить для розподілених додатків і систем з високим навантаженням. Вибір архітектури

залежить від конкретних вимог проекту, таких як очікуване навантаження, складність обробки даних та необхідність масштабування.

Таким чином, подієво-орієнтована архітектура, що виникла у відповідь на недоліки архітектури запит-відповідь, є найкращим вибором для AI-орієнтованого конектора. Вона не тільки знижує затримку при обробці запитів, але й забезпечує кращу масштабованість та гнучкість системи. Це особливо важливо в умовах високих навантажень і вимог, що швидко змінюються, характерних для сучасних AI-додатків. Нарешті, вибір архітектури має ґрунтуватися на конкретних потребах проекту, щоб досягти оптимального балансу між продуктивністю, гнучкістю та простотою розробки.

3.3 Побудова уніфікованої мікросервісної архітектури

Побудова уніфікованої мікросервісної архітектури включає кілька ключових компонентів:

1. Маршрутизація відповідає за з'єднання та балансування трафіку між користувачем та програмою, забезпечуючи ефективний розподіл навантаження.
2. Обчислювальні послуги надають інфраструктуру для розміщення основної програми, забезпечуючи необхідні ресурси для роботи та масштабування.
3. Сервіси безпеки керують доступом до інфраструктури та функціоналу програми, захищаючи дані та запобігаючи несанкціонованому доступу.
4. Бази даних служать для зберігання інформації, включаючи кеш для прискорення доступу до даних, що часто запитуються.
5. Сторонні сервіси додають додаткові функції, розширюючи можливості програми та покращуючи її функціональність.

Якщо відомі цільові AI/ML послуги та обчислювальні засоби, решта сегментів архітектури потребує більш детального вивчення. Для конектора, що

надсилає запити до необхідних послуг, масштабування має критичне значення. Конектор повинен мати репліки свого коду, на які балансуватиметься трафік від інтерфейсу користувача. Вирішення цієї задачі включає використання сторонніх засобів моніторингу, які обробляють метрики та керують масштабуванням обчислювальних потужностей та реплік коду конектора.

Розміщення коду в архітектурі подій оптимально провадиться в Docker-контейнерах. Контейнеризація спрощує оновлення та розміщення на різних серверах, а також дозволяє реплікувати образи, що знижують завантаження. Масштабування серверів можливе завдяки хмарним сервісам, що пропонують функції автоматичного збільшення ресурсів зі зростанням навантаження, реагуючи на моніторинг.

Балансування трафіку здійснюється за допомогою проксі-серверів, які мають вбудоване програмне забезпечення для відстеження кількості реплік цільового та перенаправлення трафіку для його розвантаження. , Active Directory від Azure та Google oAuth від GCP, що управляють авторизацією та автентифікацією користувачів, що підключаються до інтерфейсу взаємодії.

У такій архітектурі бази даних відіграють критичну роль. По-перше, можуть значно прискорити процеси обробки запитів, використовуючи брокери повідомлень, такі як Kafka та RabbitMQ. Крім того, бази даних можуть бути використані для зберігання статистичних даних та інших допоміжних процесів, що робить їх невід'ємною частиною загальної архітектури системи (рис.3.3).

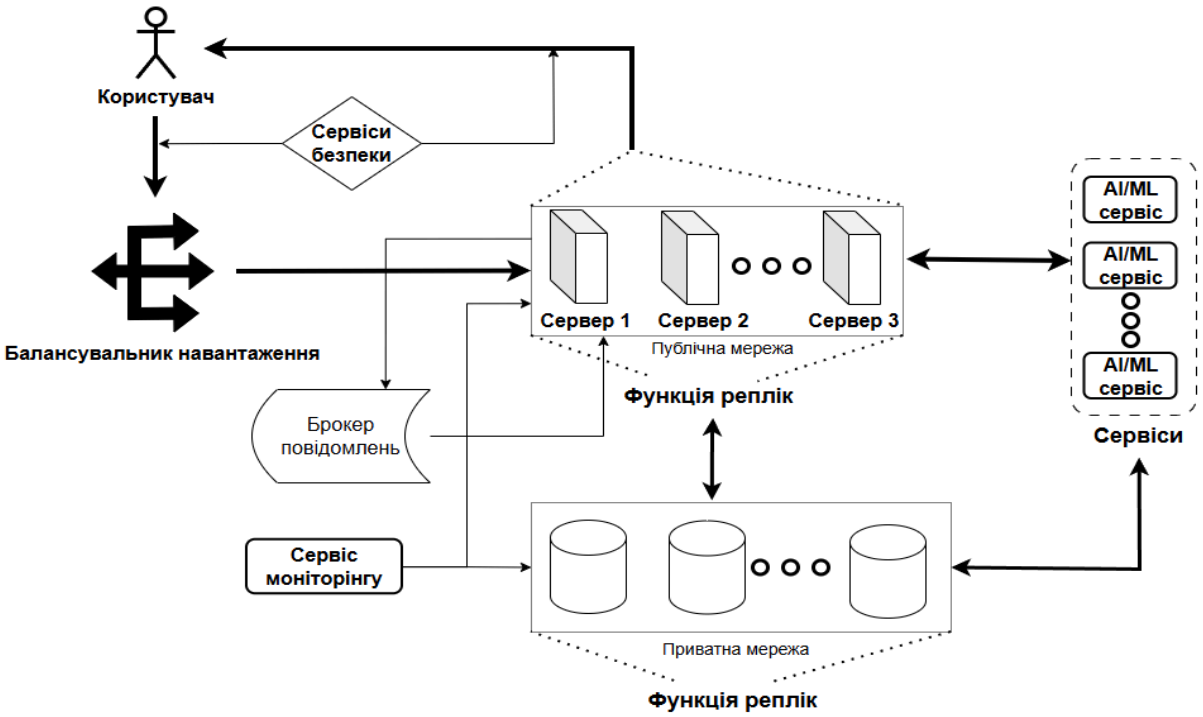


Рисунок 3.3 Принципова схема мікросервісної взаємодії конектора

На закінчення розгляду архітектури хотілося б приділити увагу внутрішній складовій – архітектурі організації Docker-контейнерів конектора і не тільки. Система серверів має кілька рівнів віртуалізації та управління ресурсами сервера.

Першим рівнем є залізний сервер та його операційна система. Цей рівень характерний наявністю лише однієї оболонки, операційної, яка здійснює управління процесами системи.

Другим рівнем можна назвати віртуалізацію. Віртуалізація серверів за своєю суттю – імітація системи. Це програмна частина, що виконує запозичення ресурсів, виділених операційною системою від сервера. Саме цей рівень є віртуальними машинами, що надаються хмарними провайдерами.

Заключним рівнем на сьогодні залишаються контейнери. По суті, лише програмна сегментація віртуальних машин та серверів. та організації Docker-контейнерів. Однак вони мають розширений функціонал, розроблений завдяки різним варіаціям управління контейнерами та їх мережами. Для

модульності інтегратора та розширення його функціоналу рекомендується використовувати саме Docker-технологію. (Рис.3.4)

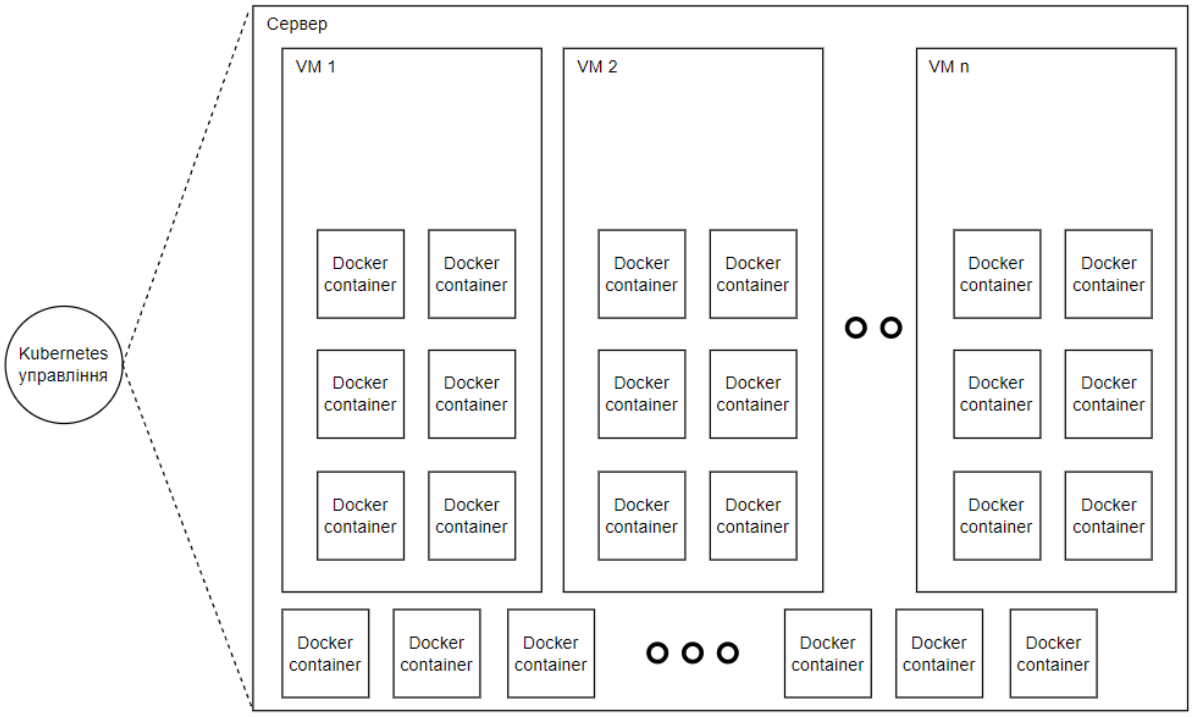


Рисунок 3.4 Організація рівнів віртуалізації

Таким чином, запропонована система надає багаторівневу архітектуру для обробки запитів з високим ступенем надійності та масштабованості. трафік, динамічно керуючи кількістю реплік віртуальних серверів, що дозволяє системі масштабуватися під навантаження, що росте, і підтримувати стабільну роботу.

Віртуальні сервери використовують другий рівень зуму для гнучкого керування запитами, який реалізований через контейнери-конектори. Вони перенаправляють запити на ML-сервіси, де визначається подальший маршрут обробки даних. Тимчасове збереження тіла запиту в NonSQL базі даних полегшує управління чергами на наступних етапах обробки, де підключено брокер повідомлень. Це рішення забезпечує паралельну обробку запитів, мінімізуючи затримки та підвищуючи швидкість відгуку системи.

У фінальній стадії контролер отримує готову відповідь від сервісів та відправляє клієнту, завершуючи цикл обробки. Така модульна архітектура не

тільки оптимізує управління ресурсами та обробку запитів, але й підвищує гнучкість та стійкість системи в умовах інтенсивного навантаження, що робить її ідеальним рішенням для сучасних розподілених обчислювальних середовищ.

3.4 Організація інтеграційного конектора

Програмна частина конектора, що інтегрує модулі для різних функціональних завдань, включаючи AI, реалізована досить просто та ефективно. Після визначення необхідних сервісів для інтеграції вибирається відповідний API взаємодії із нею. Наступним етапом є написання програмного коду, сумісного з вибраним сервісом, що забезпечує коректну передачу даних та отримання відповідей.

Особливу увагу слід приділяти організації та структурі коду. Вибір оптимальної архітектури та схеми взаємодії спрощує розробку та подальшу підтримку системи, а також підвищує її продуктивність. системах. У процесі роботи над конектором для маршрутизації трафіку структура програми повинна будуватися на основі потоків запитів та відповідей, розглянутих раніше, з урахуванням особливостей черговості та пріоритетів обробки запитів.

Таким чином, завдяки впорядкованому підходу та чіткому дотриманню схем взаємодії програмна частина конектора стає більш гнучкою та адаптованою. Це дозволяє ефективно інтегрувати AI та інші модулі, підтримуючи їх узгоджену роботу та забезпечуючи надійність усієї системи. (Рис.3.5)

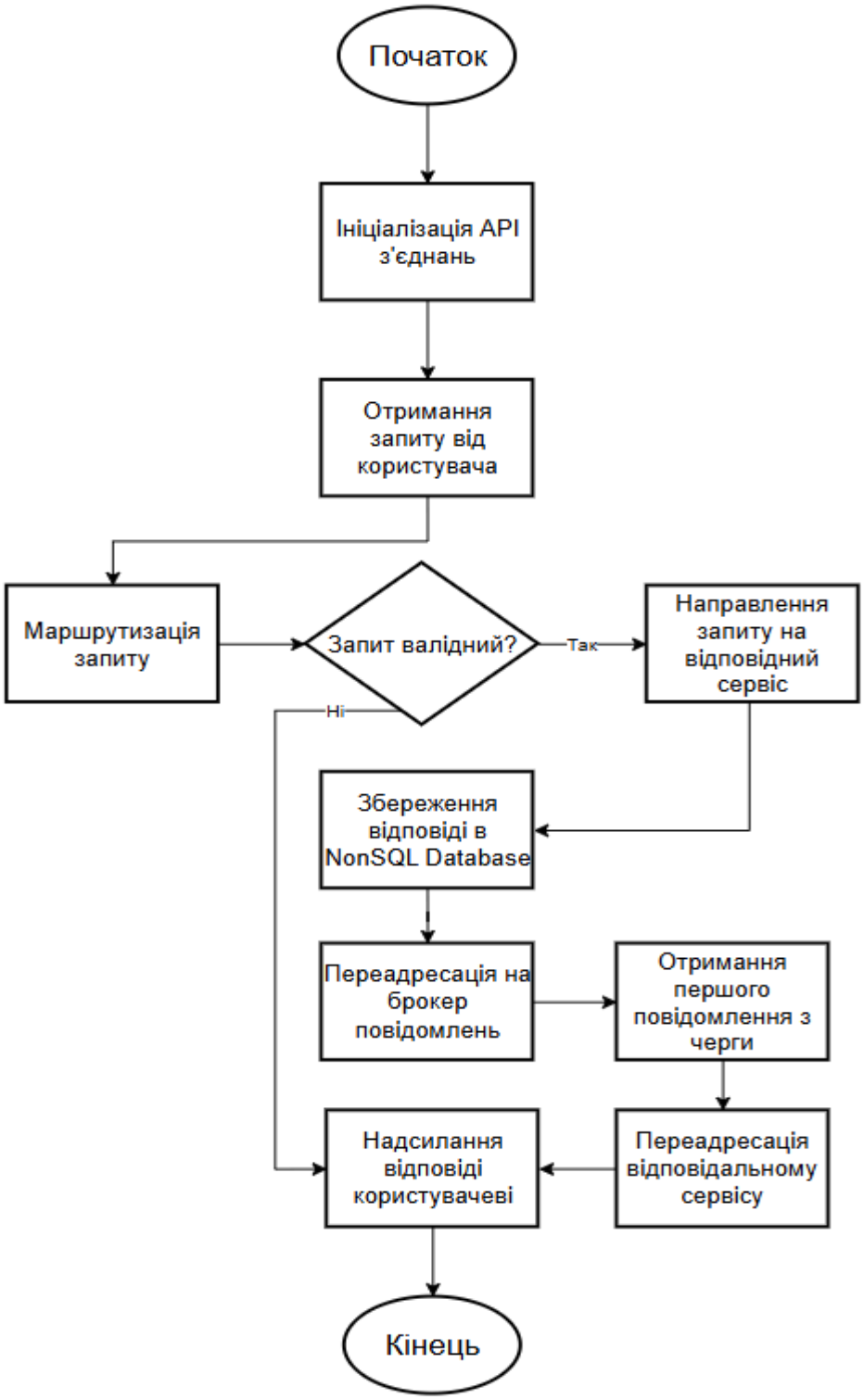


Рисунок 3.5 Блок-схема роботи конектора

Конектор починає свою роботу з ініціалізації запуску та встановлення API з'єднання із сервісами. Потім він приймає запит від користувача, який проходить процес валідації. При успішній валідації запит передається сервісу визначення його значення. Отримана відповідь зберігається в базі даних і направляється в

брокер повідомлень, який формує черги для обробки запитів конекторами. Конектори отримують нові запити з черги, позначаючи їх, щоб уникнути колізій. Останній етап – переадресація відповіді користувачеві.

Ідеальна структура програми забезпечує швидкодію та високу швидкість реакції, що знижує навантаження у періоди простою. Валідація включає виявлення невідповідностей певним стандартам повідомлення, а також може включати автентифікацію та авторизацію користувачів для забезпечення безпеки системи та контролю доступу.

4 ПРОЕКТУВАННЯ АРХІТЕКТУРИ МОДУЛЬНОГО ІНТЕГРАТОРА AI

4.1 Управління життєвим циклом системи інтегратора та даних

В управлінні життєвим циклом системи інтегратора та даних важливе місце займає впровадження практики безперервної інтеграції (CI) та безперервної доставки (CD). Ці підходи дозволяють командам розробників ефективно керувати оновленнями та змінами, мінімізуючи ризики, пов'язані з розгортанням та забезпечуючи високу якість коду.

За допомогою CI/CD команди можуть автоматизувати процеси збирання, тестування та розгортання програм.

GitHub Actions надає автоматизовані робочі процеси, які запускаються у відповідь зміни в репозиторії. GitLab CI, інтегрований у GitLab, дозволяє налаштувати пайплайни для тестування та розгортання, використовуючи конфігураційний файл. Bitbucket Pipelines, у свою чергу, пропонує простий спосіб налаштування процесів CI/CD у рамках платформи Atlassian.

Процес CI/CD зазвичай включає кілька ключових етапів, починаючи з кодування, коли розробники відправляють зміни репозиторій. Потім відбувається складання та тестування коду. Після успішного завершення тестів зміни можуть бути автоматично розгорнуті у тестовому чи продакшн-середовищі. Нарешті, важливо проводити моніторинг роботи системи, щоб оперативно реагувати на можливі проблеми.

Впровадження CI/CD у систему інтегратора даних значно підвищує швидкість та якість розробки, а також спрощує керування життєвим циклом додатків. Автоматизація рутинних завдань дозволяє командам зосередитись на складніших аспектах розробки, тим самим покращуючи ефективність роботи.

API і вебхуки відіграють ключову роль передачі даних у режимі реального часу. В API-інтеграціях дані виймаються та передаються на запит, забезпечуючи

точкове оновлення. Наприклад, дані, необхідні для навчання моделі можуть бути завантажені з різних джерел на основі фільтрів або параметрів, заданих через API. Вебхуки, навпаки, дозволяють автоматично передавати дані, як тільки відбувається їх зміна, що особливо важливо для обробки подій та роботи з системами, де потрібна миттєва реакція. Наприклад, при оновленні параметрів користувача у вихідній базі даних система автоматично відправляє дані в інтегратор, а далі в послуги, що формують ML datasets.

Другий підхід – поточна передача даних. Потокові платформи, такі як Apache Kafka, RabbitMQ або Amazon Kinesis дозволяють обробляти та передавати великі обсяги даних у реальному часі. Дані передаються пакетами або окремими подіями залежно від налаштувань та потреб. Потокова передача даних є особливо корисною для аналітики та моделей машинного навчання, які обробляють дані в реальному часі. Наприклад, сервіс для обробки логів або даних із сенсорів IoT передаватиме інформацію в ML-моделі через чергу Kafka, що дозволяє підтримувати постійний потік даних та забезпечує швидку обробку інформації в реальному часі.

ETL-пайплайни (Extract, Transform, Load) є методом передачі даних. Ці пайплайни витягують дані з різних джерел, перетворять їх у потрібний формат і завантажують у кінцеві сховища для подальшого використання. Наприклад, дані CRM та ERP-систем інтегруються в єдиний формат, очищаються від дублів, непотрібних полів та завантажуються у ML-сервіси для створення моделей. ETL-пайплайни широко застосовуються при роботі з великими даними та агрегованими даними, де потрібна передробка та очищення даних перед подачею моделі машинного навчання.

Сховища даних та бази даних відіграють роль централізованих систем зберігання інформації, до яких AI/ML сервіси можуть отримувати доступ безпосередньо. Наприклад, Amazon RDS або Google BigQuery підтримують запити на вибірку і дозволяють передавати дані безпосередньо в ML-сервіси, за винятком необхідності додаткових етапів копіювання та доставки. Для прискорення роботи часто використовуються кеш-системи, такі як Redis або

Memcached, які тимчасово зберігають дані, що часто запитуються, знижуючи навантаження на основне сховище і оптимізуючи доступ до даних.

Файлові сховища використовуються для передачі великих обсягів даних, таких як мультимедіа, відео та зображення, що потребують значних ресурсів. У цьому випадку дані завантажуються в Amazon S3 або Google Cloud Storage і організуються як посилання або метадані. ML-сервіси можуть отримувати доступ до даних у сховищах через посилання або через попередньо налаштовані пайплайни. Такий підхід дозволяє працювати з великими даними та забезпечує масштабованість та гнучкість управління ресурсами та зберіганням.

Ці методи забезпечують надійну передачу даних у ML datasets, покращуючи продуктивність систем та мінімізуючи затримки в обробці інформації. Оновлення Ось огляд методів, які застосовуються для забезпечення надійної та швидкої передачі даних в AI/ML сервіси.

4.2 Автоматизація хмарної архітектури

Для ефективного управління хмарною архітектурою, особливо для масштабованих AI/ML проектів, Terraform відіграє ключову роль, дозволяючи описувати та керувати ресурсами інфраструктури у вигляді коду. в AWS можна розгорнути інфраструктуру для Amazon SageMaker за допомогою конфігурацій Terraform, створивши потрібні віртуальні машини та сховища.

Крім того, Terraform спрощує управління цільовими сервісами, такими як Amazon SageMaker, Azure Machine Learning або Google AI Platform. швидше додавати потужності для навчання моделей у разі зростання навантаження.

Terraform також підтримує багатоклаудну інфраструктуру, що є особливо корисним для великих проектів, що потребують розподілу ресурсів між провайдерами. Це мінімізує ризики, залежно від однієї платформи. У конфігурації

можна одночасно керувати сервісами, наприклад, налаштувати GCP AutoML для аналізу даних та інтегрувати AKS для контейнеризованих додатків в Azure.

Автоматизація CI/CD через системи типу GitHub Actions або GitLab CI дозволяє оперативно впроваджувати зміни та тестувати оновлення моделей без ручного налаштування. Інфраструктурні оновлення запускаються автоматично, що допомагає підтримувати актуальність та стабільність усієї системи. Наприклад, можна використовувати для швидкого розгортання нових версій моделей оптимізуючи робочі процеси.

Terraform також підтримує комплексне управління безпекою та доступом через такі інструменти, як IAM для AWS або Cloud Identity у GCP. Це дозволяє централізовано налаштовувати ролі та політики доступу, захищаючи конфіденційні дані. Наприклад, можна автоматизувати налаштування прав доступу до даних навчання та управління API-ключами, створюючи умови для безпечної обробки даних.

Використання Terraform для AI/ML проектів забезпечує надійне управління інфраструктурою, оптимізацію ресурсів, покращення безпеки та підтримку багатоклаудових рішень, роблячи його цінним інструментом для автоматизації сучасних хмарних архітектур.

На додаток до CI/CD конектора Ansible дозволяє автоматизувати розгортання та налаштування архітектури конектора на серверах, забезпечуючи однаковість та надійність конфігурації. Спочатку за допомогою Ansible можна розгорнути на серверах необхідні базові компоненти: від операційної системи та бібліотек до Docker та інших залежностей. Це гарантує, що інфраструктура під конектор готова до роботи та відповідає вимогам.

Для запуску коннектора Ansible підтримує керування контейнерами Docker. У цьому випадку playbooks можуть описувати запуск та конфігурацію контейнерів, що особливо корисно для створення стабільного середовища. Ansible може гарантувати, що контейнери конектора завжди мають необхідні налаштування, такі як мережеві підключення, необхідні змінні середовища та правильний порядок запуску. При масштабуванні це допомагає використовувати

додаткові екземпляри контейнерів, що особливо актуально підвищення продуктивності при високих навантаженнях.

Крім того, Ansible дозволяє інтегрувати CI/CD-процеси, прискорюючи оновлення інфраструктури та програми. Системи CI/CD, такі як Jenkins, GitLab CI або GitHub Actions можуть запускати Ansible playbooks, щоб оновити конфігурації, перезапустити сервіси або застосувати зміни до інфраструктури, пов'язані з новими версіями. Це автоматизує розгортання оновлень, роблячи їх плавними та зменшуючи час простою системи.

Для забезпечення безпеки Ansible може керувати конфігурацією доступу: налаштуванням фаєрволів, правилами SSH, правами користувачів і розподілом ключів. Це запобігає несанкціонованому доступу до конектора та ресурсів, з якими він взаємодіє, створюючи безпечне середовище для обміну даними.

Ansible також забезпечує моніторинг та резервне копіювання. За допомогою playbooks можна встановити та налаштувати інструменти, такі як Prometheus та Grafana для відстеження стану сервера та стану контейнерів конектора. Це допомагає відстежувати метрики та своєчасно реагувати на будь-які відхилення у роботі. Важливою частиною управління інфраструктурою є регулярне резервне копіювання даних та конфігурацій, які можна автоматизувати через Ansible, створюючи стратегію швидкого відновлення системи при збоях.

Таким чином, Ansible стає критично важливим компонентом для підтримки, оновлення та безпеки архітектури конектора, полегшуючи управління інфраструктурою, підвищуючи надійність і дозволяючи легко адаптуватися до умов та вимог системи, що змінюються.

4.3 Організація розгортання та створення інфраструктури

В основі CI/CD архітектури для розгортання та доставки оновлень лежать ключові інтеграційні процеси, що забезпечують гнучкість налаштування та

автоматизації хмарної архітектури. Організація CI/CD надає безліч методів налаштування CI та додавання конфігурацій автоматизації, що дозволяє покращити процеси доставки та підтримки цільових сервісів.

Основні атрибути CI/CD архітектури:

1. Конфігурація автоматизації та CI/CD pipeline. Pipeline служить важливим компонентом CI/CD, являючи собою набір інструкцій, що визначають порядок розгортання та доставки додатків, даних та моделей. З його допомогою можна автоматично керувати процесами деплой та оновлень на різних платформах, таких як GitHub, GitLab, Bitbucket, Azure DevOps та інші.

2. Послуги безпеки. Вони відіграють ключову роль у забезпеченні ізольованого доступу CI/CD pipeline до цільових сервісів, що особливо важливо для забезпечення безпеки даних та додатків. Сервіси безпеки можуть включати базові IAM конфігурації для AWS, Azure Managed Identity, а також SSH-доступ, якщо використовуються on-premise сервери. Ці заходи безпеки обмежують доступ до чутливих ресурсів, забезпечуючи безпечну та захищену передачу даних.

3. Цільовий сервіс. Під цим маються на увазі кінцеві точки, куди направляються дані та образи додатків для їхнього розгортання. Цільовими сервісами можуть бути бази даних, сервери додатків, а також AI/ML сервіси, які обслуговують програми та дані, що використовуються у виробничих середовищах. Різні методи інтеграції дозволяють автоматично керувати оновленнями та забезпечують безперебійну передачу даних та моделей у потрібне місце. (Рис.4.1)

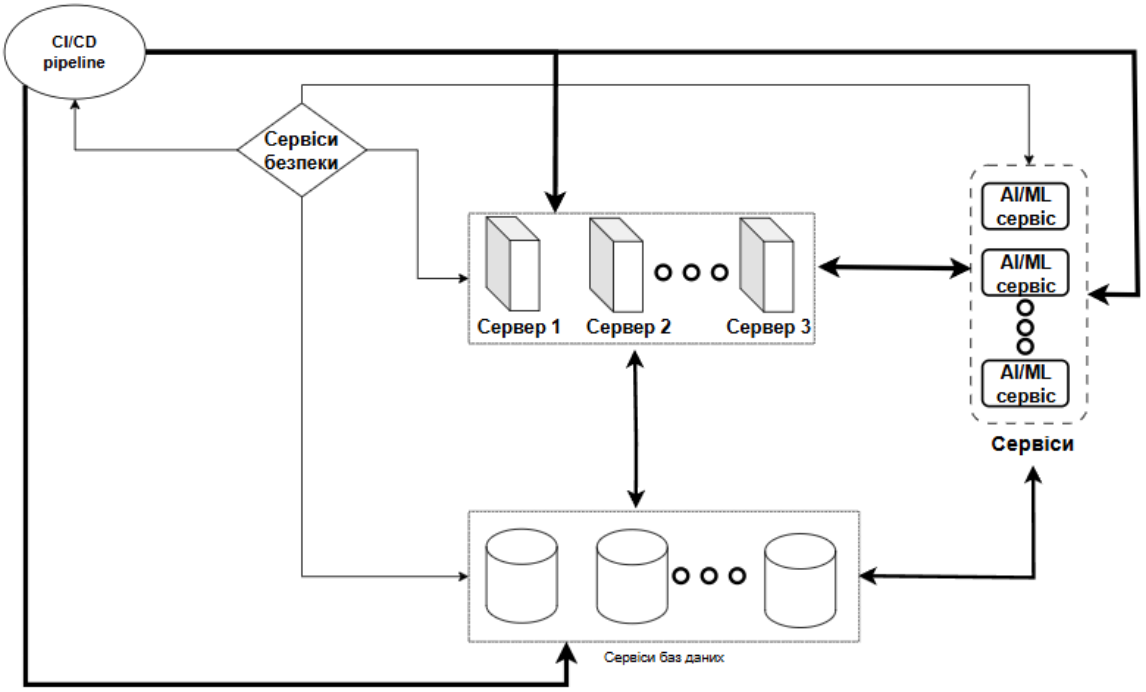


Рисунок 4.1 Організація доступу між CI/CD pipeline та цільовими сервісами

Структура CI/CD архітектури забезпечує високий рівень автоматизації та безпеки, інтегруючи інструкції та конфігурації для безперервної доставки та оновлення хмарних сервісів та додатків.

Підхід до створення хмарної інфраструктури можна ефективно організувати за допомогою інструментів інфраструктури як коду (Infrastructure as Code, IaaC), таких як Terraform та Ansible, що забезпечують стабільність, передбачуваність та відтворюваність при налаштуванні ресурсів. Завдяки інтеграції з CI/CD конвеєрами змінні оточення, включаючи конфіденційні дані, можуть передаватися до цих утиліт безпосередньо через CI/CD сервіси, що додатково підвищує безпеку і спрощує управління інфраструктурою.

Terraform дозволяє легко керувати змінними оточення. Наприклад, передачі не конфіденційної інформації Terraform використовують файли .tfvars, які містять значення всім потрібних змінних. /CD, такі як GitHub Actions, GitLab CI/CD та інші. dev, staging, production та UAT. Кожна ситуація використовує свої параметри хмарних сервісів та конфігурації серверів, що дозволяє оптимально налаштувати інфраструктуру під конкретні завдання та потреби кожного оточення.

Ansible також добре інтегрується у процеси CI/CD та підтримує гнучку роботу зі змінними. Ansible змінні можуть бути оголошені в будь-якому файлі, доступному для плейбуків, і можуть бути передані за допомогою прапора --extra-var, що дозволяє керувати конфігураціями на різних рівнях з високим ступенем контролю і зручністю. За допомогою Ansible можна детальніше контролювати конфігурації серверів і сервісів, використовуючи той же підхід до організації оточень, що і Terraform.

Важливим аспектом організації CI/CD є структура пайплайнів (pipeline), яка може бути незалежною, залежною чи гібридною. Ці типи дозволяють адаптувати процес розгортання під різні вимоги програми, зберігаючи ефективність та забезпечуючи гнучкість.

Незалежні пайплайни є workflow, які можуть виконуватися автономно, незалежно від інших процесів. Такий підхід характерний для розгортання монолітних додатків або додатків, яким не потрібні інші оновлення. Ці пайплайни, як правило, прості в реалізації, мають високу автономність і можуть працювати паралельно, що зменшує час розгортання.

Залежні пайплайни, також відомі як reusable workflows, вимагають суворої послідовності виконання, так як вони залежать від стану інших компонентів. даних додатків критично важливо, щоб оновлення виконувались у потрібному порядку, інакше можуть виникнути проблеми сумісності.

Гібридні пайплайни поєднують переваги незалежних та залежних моделей, забезпечуючи як послідовне, так і паралельне виконання. Цей тип структури дозволяє об'єднати всі процеси CI/CD в єдину систему, де частина операцій може виконуватися одночасно, інші запускаються після успішного завершення залежних кроків. Такий підхід є оптимальним для великих проектів, де необхідна гнучкість в управлінні процесами оновлення та взаємодії між компонентами. (Рис.4.2)

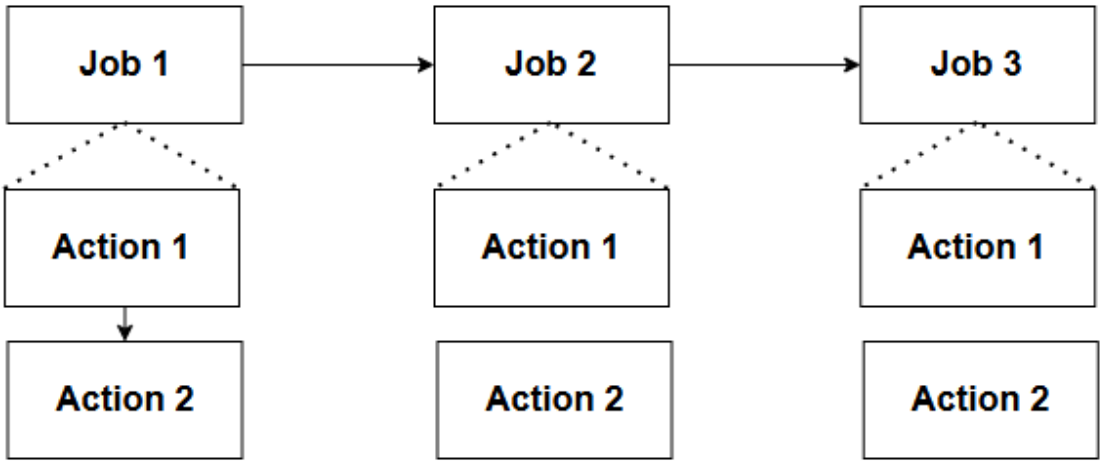


Рисунок 4.2 Гібридні workflows

Ці три типи пайплайнів допомагають точно налаштувати процес розгортання, враховуючи специфіку програми та інфраструктури, що забезпечує надійність та стабільність оновлень.

4.4 МЛ операції

MLOps є ключовою концепцією в розробці та впровадженні моделей машинного навчання, яка тісно пов’язана з CI/CD підходами для ефективного управління життєвим циклом моделей. Використання CI/CD дозволяє автоматизувати процеси підготовки даних, навчання, тестування, розгортання, оновлення та моніторингу моделей, що забезпечує високу швидкість і гнучкість розробки. Це особливо важливо в умовах постійно змінних даних та вимог, де швидке впровадження нових моделей є конкурентною перевагою. Автоматизація скорочує час між розробкою і впровадженням, підвищує точність прогнозів та стабільність роботи систем, а також знижує ризик використання застарілих чи неефективних моделей.

Організація CI/CD у MLOps починається з налаштування пайплайнів, що забезпечують послідовність виконання завдань, таких як підготовка даних, навчання, валідація і тестування моделей. Для цього використовуються платформи на кшталт GitHub Actions, GitLab CI/CD або Jenkins, які пропонують гнучкі інструменти для автоматизації процесів. Особлива увага приділяється інтеграції даних та керуванню їхніми версіями, що досягається за допомогою інструментів, таких як DVC чи MLflow. Це забезпечує прозорість змін даних і моделей, критично важливу для відтворюваності результатів.

Процеси автоматизованого розгортання моделей передбачають використання контейнерів або скомпільованих артефактів для інтеграції у цільові середовища, такі як Kubernetes, Azure Machine Learning або Amazon SageMaker. Це дозволяє масштабувати моделі та забезпечувати їхню стабільну роботу в продакшен-середовищах. Важливим аспектом є безпека та контроль доступу, які реалізуються через секрети CI/CD систем або хмарні сервіси аутентифікації. (Рис.4.3)

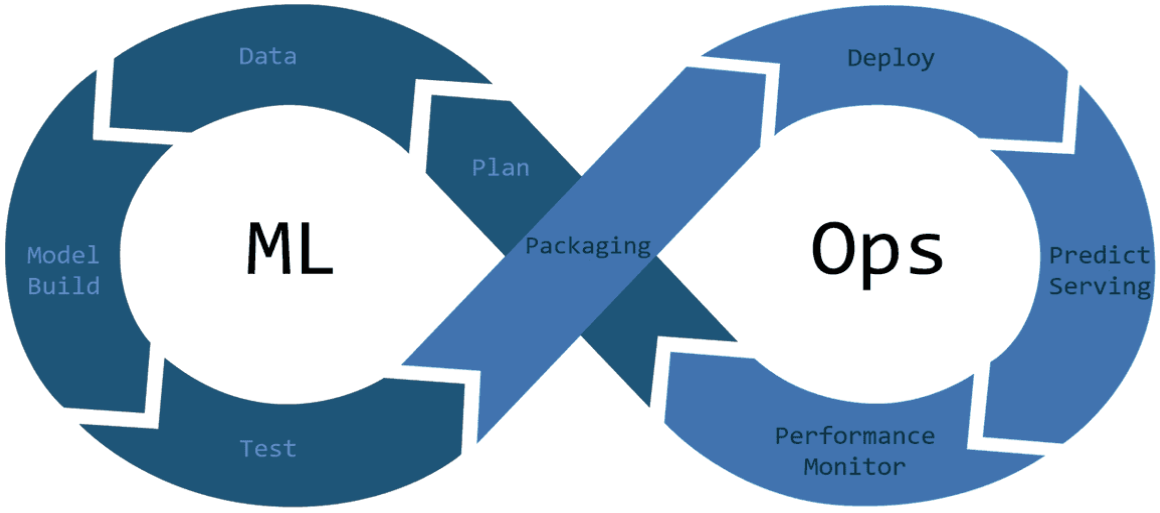


Рисунок 4.3 MLops pipeline

MLOps pipeline охоплює всі етапи життєвого циклу моделей. Спершу здійснюється підготовка даних шляхом очищення, нормалізації та структурування для створення якісних навчальних і тестових наборів. Потім визначаються цілі

проєкту, вимоги до моделі, і розробляється стратегія роботи. У процесі навчання відбувається тестування різних алгоритмів і гіперпараметрів, а успішно навчені моделі переходять до тестування для перевірки відповідності цільовим метрикам. Після тестування модель упаковується, наприклад, у формат контейнера Docker, для інтеграції у різні середовища. На етапі розгортання модель стає доступною для використання реальними користувачами, інтегруючись у веб-додатки або хмарні сервіси.

Моніторинг продуктивності забезпечує відстеження ключових метрик моделі, дозволяючи виявляти деградацію та зміни у вхідних даних. Це забезпечує основу для постійного вдосконалення моделей через донавчання та адаптацію до змін середовища. Таким чином, використання MLOps pipeline та CI/CD підходів дозволяє автоматизувати, стандартизувати і оптимізувати процеси розробки та підтримки моделей, що сприяє економії ресурсів, підвищенню якості та ефективності систем.

ВИСНОВКИ

Результати дослідження систематизували області штучного інтелекту, які можна застосувати у звичайних сервісах, а також довели, що для ефективної інтеграції штучного інтелекту в інформаційні системи важливу роль відіграють тренувальні дані. А також класифікували сервіси, що надають необхідний функціонал залежно від проєктних вимог.

Так само було досліджено інтеграційний потенціал хмарних AI/ML сервісів. І доведено простоту й ефективність їхнього впровадження без істотних змін коду за допомогою API запитів, таких як GraphQL, RESTful API, gRPC і SOAP.

Як центральний елемент, було розроблено конектор, що забезпечує зв'язок між різними AI/ML сервісами і бізнес-системами, керуючи передачею даних, маршрутизацією API запитів і взаємодією між компонентами. Його ефективність підтверджується спрощеним способом впровадження самих сервісів порівняно з витратами на розробку власних ШІ сервісів.

Модульна архітектура дає змогу конектору гнучко адаптуватися до мінливих вимог, завдяки незалежній від основного коду логіці з'єднання. А автоматизоване розгортання через CI/CD спрощує процеси оновлення та масштабування, на відміну від ручних засобів розгортання.

На доповнення, завдяки використанню віртуалізованих засобів на базі технології Docker для впровадження додатків мікросервісної архітектури дослідження доводить зниження навантаження на ресурси.

Не менш важливою знахідкою є те, що використання інструментів автоматизації інфраструктури в коді, як-от Terraform і Ansible, дає змогу автоматизувати процеси розгортання й управління інфраструктурою, знижуючи витрати та мінімізуючи людські помилки під час ручної конфіграції. Це підкреслює важливість хмарних платформ AWS, Azure і GCP як систем, що підтримують IaaC, які дають змогу ефективно впроваджувати AI/ML сервіси, а також розміщувати програмне забезпечення.

У підсумку, результати роботи також показали, що побудова CI/CD процесів для AI-моделей значно спрощує їхню інтеграцію в наявні бізнес-процеси, знижуючи час від розробки до впровадження. Це особливо помітно в порівняння з ручним способом розробки і доставки коду в продуктивне середовище.

Таким чином, було розроблено концептуальну систему інтеграції сервісів штучного інтелекту засобом модульної інтеграції через хмарних провайдерів. А так само доведено ефективність її застосування для забезпечення гнучкості, масштабованості та звести до мінімуму технічні та економічні виклики.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Explorations in Artificial Intelligence and Machine Learning / Prof. Dott-Ing. Roberto V. Zicar, r, 2006 - P. 3-18.
2. Artificial Intelligence A Modern Approach 4th Edition / Peter Norvig, Stuart Russell, 2017 - P. 11-12, 28, 651-654, 823-825, 860-862.
3. MATLAB Deep Learning: 3 Machine Learning, Neural Networks and Artificial Intelligence / Phil Kim, r, 2017 - P. 20.
4. LLM [Електронний ресурс] / Онлайн-енциклопедія Вікіпедія. - Режим доступу: https://en.wikipedia.org/wiki/Large_language_model
5. Глибинне навчання [Електронний ресурс] / Онлайн-енциклопедія Вікіпедія. - Режим доступу: https://en.wikipedia.org/wiki/Deep_learning
6. VAEs [Електронний ресурс] / Онлайн-енциклопедія Вікіпедія. - Режим доступу: https://en.wikipedia.org/wiki/Variational_autoencoder
7. Artificial Intelligence and potential for Garbage In, Garbage Out / Leslie Citrome - 29 Nov 2023. - Vol. 40
8. Influence of Structured, Semi-Structured, Unstructured data on various data models / Shagufta Praveen, Umesh Chandra - Dec 2017 - Vol. 8
9. BATCH: Machine Learning Inference Serving on Serverless Platforms with Adaptive Batching / Ahsan Ali, Riccardo Pinciroli, Evgenia Smirni, Feng Yan - Nov 2020
10. Generative Artificial Intelligence – Foundations, Use Cases and Economic Potential / Volker Brühl - Feb 2024
11. Retrieval-Augmented Generation for AI-Generated Content: A Survey / Penghao Zhao, Hailin Zhang, Qinhan Yu, Zhengren Wang, Yunteng Geng, Fangcheng Fu, Ling Yang, Wentao Zhang, Bin Cui - 29 Feb 2024
12. 7 Reinforcement Learning from Human Feedback (RLHF) / Dawn Sivan, K. Satheesh Kumar, Veena Raj, Rajan Jose - 2024 - Generative AI and LLMs pp.135-154

13. До проблеми типізації глобальних проблем людства / Філософсько-психологічні аспекти духовності сталого розвитку людства: зб. міжнар. наук.-практ. конф. – Львів: ЛНУ ім. І. Франка - 2022. С. 101-105.

14. AWS and Azure AI ML Services [Електронний ресурс] / Документація. - Режим доступу: https://learn.microsoft.com/en-gb/azure/architecture/aws-professional/services#ai-and-machine-learning

15. GCP and Azure AI ML Services [Електронний ресурс] / Документація. - Режим доступу: https://learn.microsoft.com/en-gb/azure/architecture/gcp-professional/services#ai-and-machine-learning

16. V. Iglovikov, S. Seferbekov, A. Buslaev, A. Shvets, "TernausNetV2: Fully Convolutional Network for Instance Segmentation", in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, Salt Lake City, UT, USA, 18–22 June 2018; pp. 233–237. URL: https://arxiv.org/pdf/1806.00844.pdf

17. Mark Girolami, Simon Rogers, A First Course in Machine Learning, Chapman & Hall/CRC, 2011, 305c.

18. Evaluating GraphQL and REST API Services Performance in a Massive and Intensive Accessible Information System / Benny Leonard Enrico Panggabean, Armin Lawi, Takaichi Yoshida - 2024

19. The GraphQL Handbook / Hasura company - 2024

URL: https://graphql-engine-cdn.hasura.io/assets/main-site/case-studies/hasura-graphql-handbook-24.pdf

20. Recipe and Formulation Sheet of Soap / Muhammad Shoukat Hussain - September 2022

21. Hook-in Privacy Techniques for gRPC-based Microservice Communication / Louis Loechel, Siar-Remzi Akbayin, Elias Grünewald, Jannis Kiesel - April 2024

22. Machine Learning Operations (MLOps): Overview, Definition, and Architecture / Dominik Kreuzberger, Niklas Kühl, Sebastian Hirschl - January 2023

23. Cloud Computing: IaaS general purpose VM performance comparison between Microsoft Azure, Amazon AWS, and Google Cloud GCP / Mert Cakir - January 2023

24. Comparative security and compliance analysis of serverless computing platforms: AWS lambda, azure functions, and google cloud functions / Darshan Khanal Dibya, Maharjan Sushil - January 2024

25. DevOps vs MLOps: хто це такі і чим відрізняються їхні задачі / Vlad Melnyk - 17 May 2025 URL: https://dou.ua/forums/topic/48792/

26. PROSPEKTIVE GLOBALE WISSENSCHAFTLICHE TRENDS / OBJECT RECOGNITION SYSTEMS USING INTELLIGENT TECHNOLOGIES IN UAV / CHAPTER 4 / Koval O., Goots V., Borysova E.O., Fefelov D.V., Kravchenko A.A., Kuzmenko R.H., Kuzmin O.V., Maksymiuk A.I., Osadcha V.A., Vozniuk S.R., Omelchenko M.S., Antonenko A.V., Golubenko O.I., Tkachenko O.M., Popereshnyak S.V., Tonkykh O.G., Tverdokhlib A.O., Korotin D.S., Balvak A.A., Vostrikov S.O., Burachynskyi A.Y., Huminiuk V.Y., Mishkur Y.V., Haleta V.S., Skudnyi D.Y., Tuzhilin D.I., Horbiychuk M.I., Yednak I.S., Mysak I.V., Omelchenko M.P., Kovalenko L.I., Femiak V., Cherniuk V., Vovk L., Vasylenko O.B., Stashenko M.S., Chvrova O.E., Serheta I.V., Mostova O.P., Dabybida N., Popovych D., Korin M., Kuziv O., Derpak Y.Y., Maikut-Zabrodska I., Honcharova O.V., Shevchenko V.Y., Melnychenko S.H.

27. Антоненко А. В., Тужилін Д.І. CODE THAT THINKED A LOT. «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ» Науково-техн. конф., м.Київ, 24 квітня 2024 р. С. 445-446 URL: https://duikt.edu.ua/uploads/p_2661_45497999.pdf

28. Антоненко А. В., Тужилін Д.І. INTERIOR VIEW. DYNAMICS OF VIRTUAL WORLDS. «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ» Науково-техн. конф.,

м.Київ, 24 квітня 2024 р. С. 385-386 URL: https://duikt.edu.ua/uploads/p_2661_45497999.pdf

Додаток А

ДЕМОНСТРАЦІЙНІ МАТЕРІАЛИ

ДЕРЖАВНИЙ УНІВЕРСИТЕТ
ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЙ
НАВЧАЛЬНО-НАУКОВИЙ ІНСТИТУТ
ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ
Кафедра Комп'ютерної інженерії

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня магістра

НА ТЕМУ: «АРХІТЕКТУРА ІНТЕГРАЦІЇ ШІ КОМПОНЕНТІВ У РОЗПОДІЛЕНИХ СИСТЕМАХ»

Виконав студент групи КСДМ-61

Тужилін Денис Ігорович

Керівник дипломної роботи:
Антоненко Артем Васильович,
кандидат технічних наук, доцент

Київ – 2024

Мета роботи – метою цієї роботи є розробка концептуальної моделі модульної інтеграції ШІ, орієнтованої на хмарні платформи, для ефективного та гнучкого впровадження ШІ в бізнес-процеси, шляхом дослідження, аналізу та оцінки сучасних архітектур і систем ШІ.

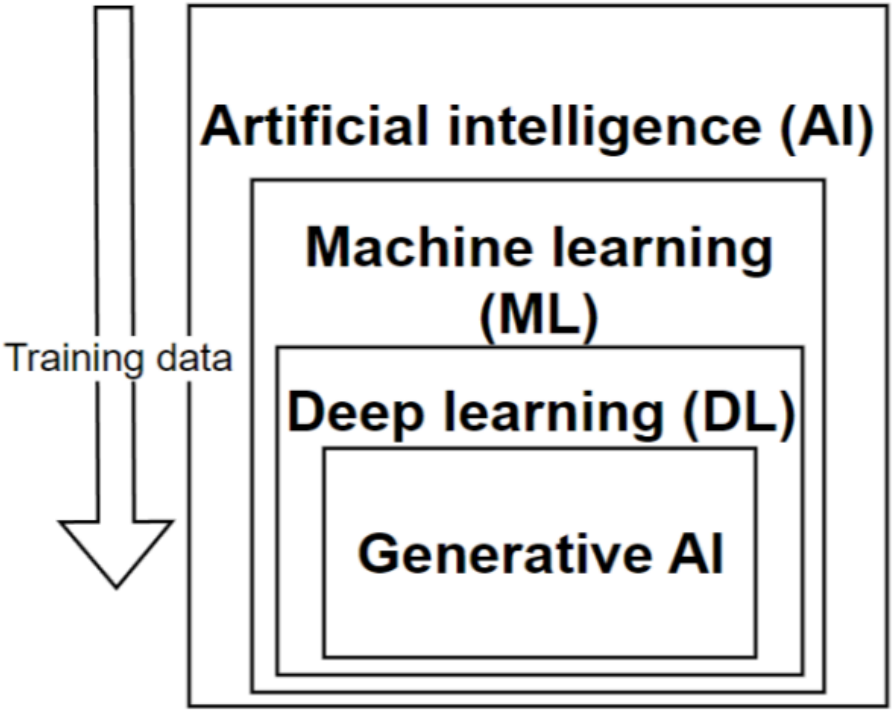
Об'єкт дослідження – хмарні сервіси та системи штучного інтелекту включно з похідними моделями.

Предмет дослідження – архітектура систем штучного інтелекту та його похідних частин.

АКТУАЛЬНІСТЬ ОБРАНОЇ ТЕМИ

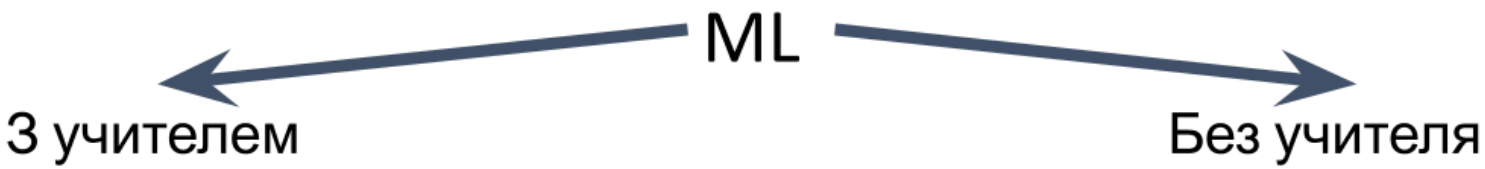
Сучасний розвиток технологій, особливо штучного інтелекту, є важливим фактором автоматизації бізнес-процесів та оптимізації взаємодії людини з комп'ютером. ШІ вже активно використовується у сферах, таких як обслуговування клієнтів, виробництво та логістика, демонструючи високу ефективність. Проте ефективна інтеграція ШІ в інфраструктуру потребує ретельного підходу до проектування архітектури й вибору відповідних технологій для забезпечення масштабованості й гнучкості. Це робить дослідження методів модульної інтеграції ШІ надзвичайно актуальним.

ВІРТУАЛІЗАЦІЯ ШІ ОБОЛОНОК



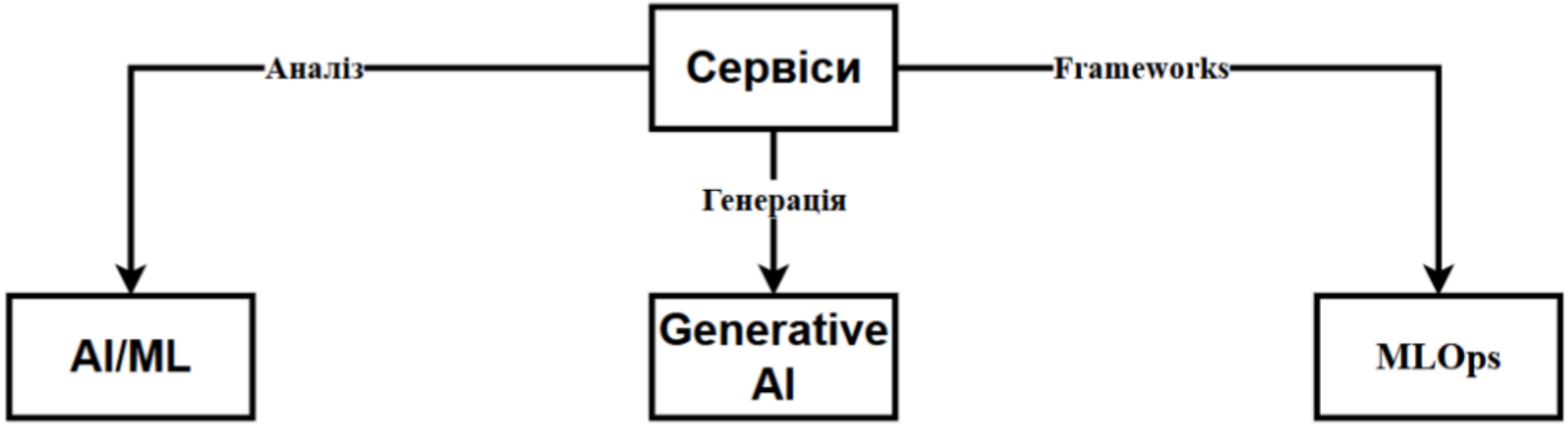
- AI - всеосяжне поняття.
- ML - сукупність нейронних систем.
- DL - нейронні системи.
- Генеративний AI - система аналізу, що базується на готових ML моделях.

ПРИНЦИПИ НАВЧАННЯ МОДЕЛЕЙ

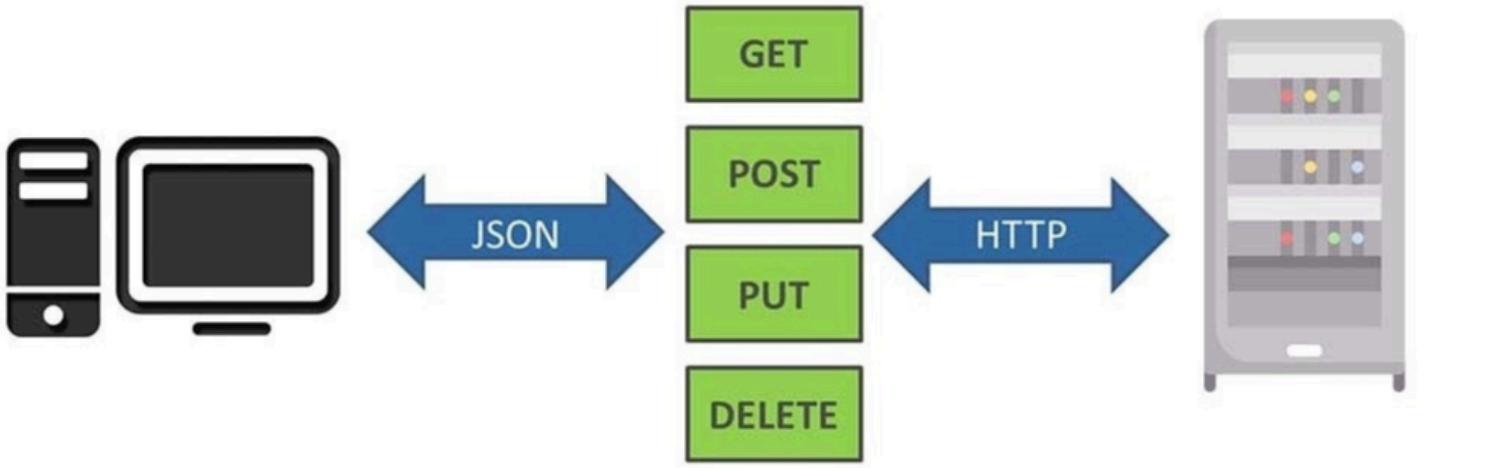


- Структуровані дані
- Неструктуровані дані
- Помічені дані
- Непомічені дані

КЛАСИФІКАЦІЯ ШІ СЕРВІСІВ



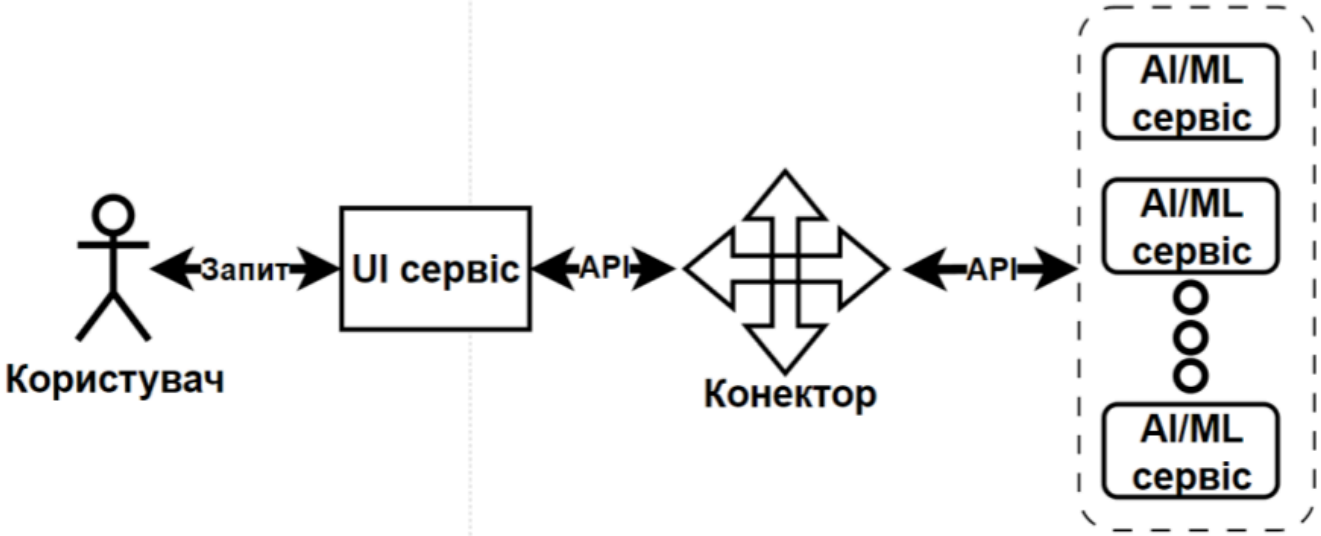
API


GET
POST
PUT
DELETE
JSON
HTTP
Client Request
HTTP Methods
Server Response

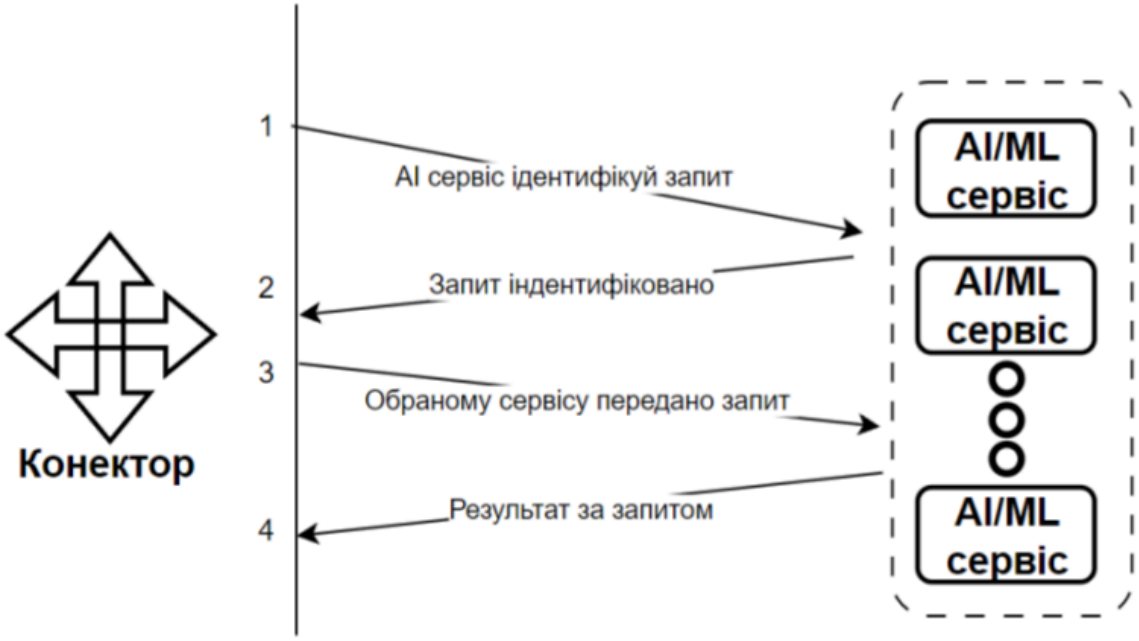
ХМАРНІ СЕРВІСИ ШІ

PRODUCT	aws	Microsoft Azure	Google Cloud Platform
Machine learning	Amazon SageMaker	Azure Machine Learning	Google Cloud AI Platform
Image recognition	Amazon Rekognition	Azure Cognitive Services	Google Cloud Vision
Speach	Amazon Polly, Amazon Transcribe	Azure Cognitive Services	Google Cloud Speech-to-Text and Text-to-Speech
Natural language processing	Amazon Comprehend	Azure Cognitive Services	Google Cloud Natural Language API:
Big Data Analytics	Databricks	Databricks	Databricks
Chat Bot	AWS Chatbot	Azure Bot Service	Dialogflow
Pricing	Per hour	Per minute	Per minute

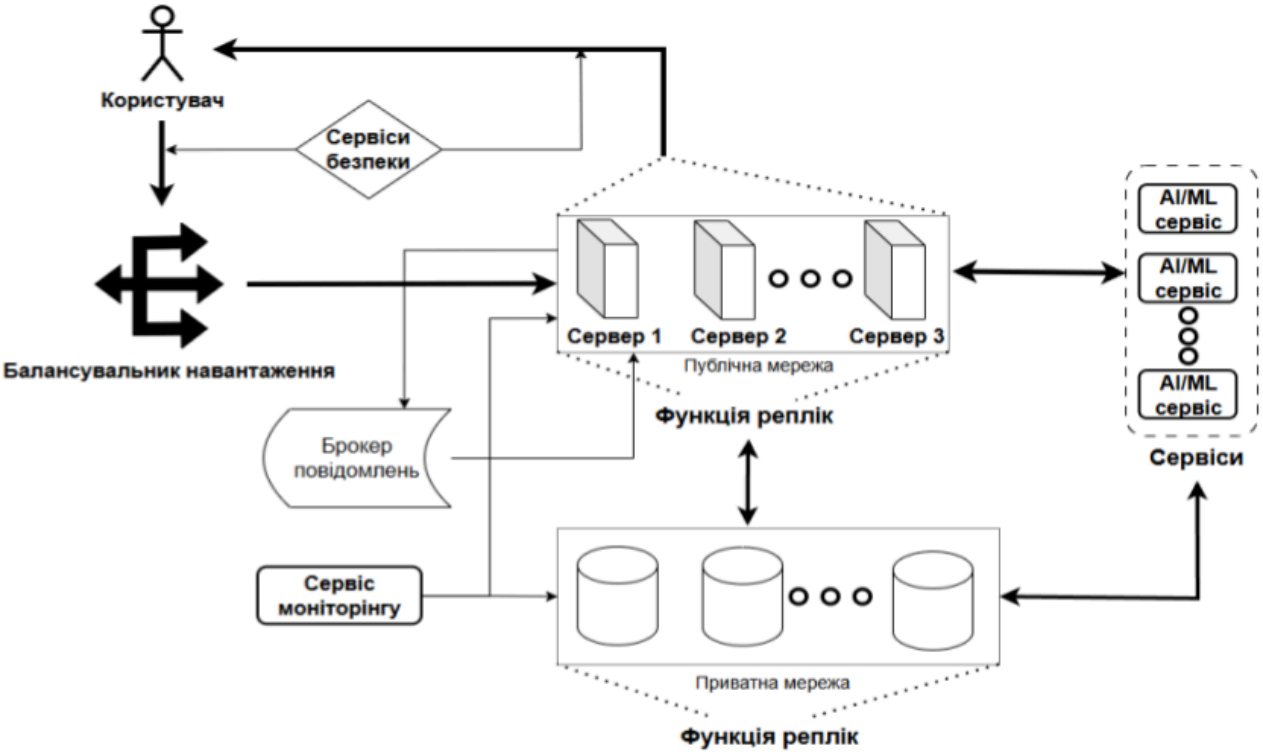
ПРИНЦИП МІЖСЕРВІСНОГО З'ЄДНАННЯ З ШІ



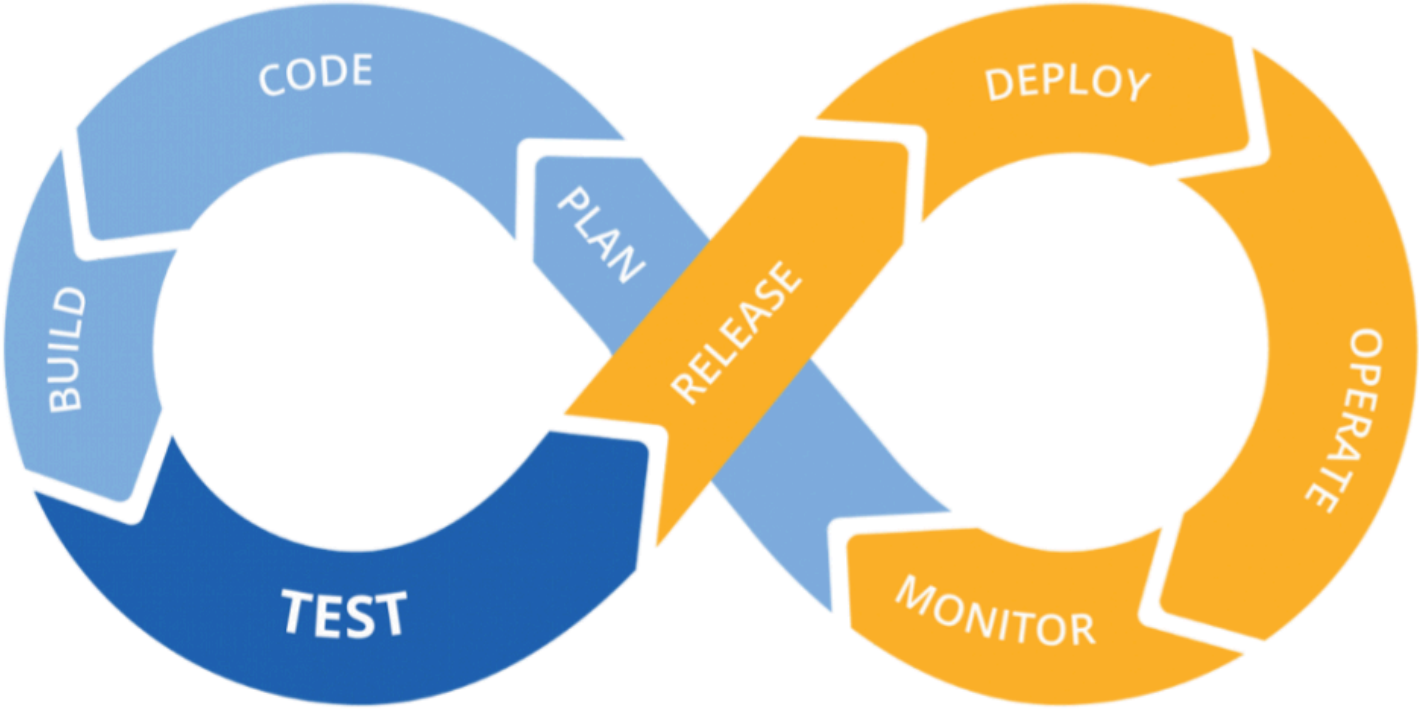
API З'ЄДНАННЯ КОНЕКТОРА



СЕРВІСНА АРХІТЕКТУРА



CI/CD



ВИСНОВКИ

Результати дослідження систематизували області штучного інтелекту, які можна застосувати у звичайних сервісах, а також довели, що для ефективної інтеграції штучного інтелекту в інформаційні системи важливу роль відіграють тренувальні дані. А також класифікували сервіси, що надають необхідний функціонал залежно від проєктних вимог.

Так само було досліджено інтеграційний потенціал хмарних AI/ML сервісів. І доведено простоту й ефективність їхнього впровадження без істотних змін коду за допомогою API запитів, таких як GraphQL, RESTful API, gRPC і SOAP.

Як центральний елемент, було розроблено конектор, що забезпечує зв'язок між різними AI/ML сервісами і бізнес-системами, керуючи передачею даних, маршрутизацією API запитів і взаємодією між компонентами. Його ефективність підтверджується спрощеним способом впровадження самих сервісів порівняно з витратами на розробку власних ШІ сервісів.

Модульна архітектура дає змогу конектору гнучко адаптуватися до мінливих вимог, завдяки незалежній від основного коду логіці з'єднання. А автоматизоване розгортання через CI/CD спрощує процеси оновлення та масштабування, на відміну від ручних засобів розгортання.

ПРОДОВЖЕННЯ ВИСНОВКІВ

На доповнення, завдяки використанню віртуалізованих засобів на базі технології Docker для впровадження додатків мікросервісної архітектури дослідження доводить зниження навантаження на ресурси.

Не менш важливою знахідкою є те, що використання інструментів автоматизації інфраструктури в коді, як-от Terraform і Ansible, дає змогу автоматизувати процеси розгортання й управління інфраструктурою, знижуючи витрати та мінімізуючи людські помилки під час ручної конфіграції. Це підкреслює важливість хмарних платформ AWS, Azure і GCP як систем, що підтримують IaaC, які дають змогу ефективно впроваджувати AI/ML сервіси, а також розміщувати програмне забезпечення.

У підсумку, результати роботи також показали, що побудова CI/CD процесів для AI-моделей значно спрощує їхню інтеграцію в наявні бізнес-процеси, знижуючи час від розробки до впровадження. Це особливо помітно в порівняння з ручним способом розробки і доставки коду в продуктивне середовище.

Таким чином, було розроблено концептуальну систему інтеграції сервісів штучного інтелекту засобом модульної інтеграції через хмарних провайдерів. А так само доведено ефективність її застосування для забезпечення гнучкості, масштабованості та звести до мінімуму технічні та економічні виклики.

РЕЗУЛЬТАТИ ВПРОВАДЖЕННЯ

1. Антоненко А. В., Тужилін Д.І. CODE THAT THINKED A LOT. «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ» Науково-техн. конф., м.Київ, 24 квітня 2024 р. С. 445-446 URL: https://duikt.edu.ua/uploads/p_2661_45497999.pdf

2. Антоненко А. В., Тужилін Д.І. INTERIOR VIEW. DYNAMICS OF VIRTUAL WORLDS. «ЗАСТОСУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ В ІНФОРМАЦІЙНО-КОМУНІКАЦІЙНИХ ТЕХНОЛОГІЯХ» Науково-техн. конф., м.Київ, 24 квітня 2024 р. С. 385-386 URL: https://duikt.edu.ua/uploads/p_2661_45497999.pdf