Міністерство освіти і науки України
Національний технічний університет
«Дніпровська політехніка»

Інститут електроенергетики
(інститут)

Факультет інформаційних технологій
(факультет)

Кафедра Програмного забезпечення комп'ютерних систем
(повна назва)

ПОЯСНЮВАЛЬНА ЗАПИСКА
кваліфікаційної роботи ступеня

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *Чемериса Євгена Дмитровича*
(ПІБ)

академічної групи *122-20-3*
(шифр)

освітньої програми *Комп'ютерні науки*
(код і назва напряму підготовки)

на тему: *Розробка автоматизованої інформаційної системи для обробки зображень з використанням Serverless архітектури у контексті Amazon Web Services*

Керівники	**Прізвище, ініціали**	**Оцінка за шкалою**		**Підпис**
		рейтинговою	**інституційною**	
кваліфікаційної роботи	*доц. Кабак Л.В*			
розділів:				
спеціальний	*доц. Кабак Л.В*			
економічний	*доц. Касьяненко Л.В.*			
Рецензент	*доц. Каштан В.Ю.*			
Нормоконтролер	*доц. Гуліна І. Г.*			

Дніпро
2024

Міністерство освіти і науки України
НТУ «Дніпровська політехніка»

ЗАТВЕРДЖЕНО:
завідувач кафедри
програмного забезпечення комп'ютерних систем
(повна назва)

М.О. Алексєєв
(підпис) (прізвище, ініціали)

« » 2024 року

ЗАВДАННЯ
на кваліфікаційну роботу

бакалавра
(назва освітньо-кваліфікаційного рівня)

студента *122-20-3* *Чемериса Є.Д.*
(група) (прізвище та ініціали)

тема кваліфікаційної роботи *Розробка автоматизованої інформаційної системи для обробки зображень з використанням Serverless архітектури у контексті Amazon Web Services*

затверджена наказом ректора НТУ «ДП» від *23 . 05 .2024 р.* № *470 -с*

Розділ	Зміст виконання	Термін виконання
Спеціальний	***На основі матеріалів виробничої практики та інших науково-технічних джерел провести аналіз стану рішення проблеми та постановку задачі.*** ***Обґрунтувати вибір та здійснити реалізацію методів вирішення проблеми***	***13.05.2024 р.***
Економічний	***Провести розрахунок трудомісткості розробки програмного забезпечення, витрат на створення ПЗ й тривалості його розробки***	***27.05.2024 р.***

Завдання видав _____ *доц.Кабак Л.В.*
(підпис) (посада, прізвище, ініціали)

Завдання прийняв до виконання _____ *Чемерис Є.Д.*
(підпис) (прізвище, ініціали)

Дата видачі завдання: *14.01.2024 р.*

Термін подання кваліфікаційної роботи до ЕК: *10.06.2024 р.*

ВСТУП

У динамічному світі технологій кожна онлайн-служба має власну унікальну архітектуру, що включає інтерфейс взаємодії із користувачем, набір фреймворків, інструменти управління базами даних, тип розміщення серверів, склад команди підтримки сервісу та багато іншого. Від вибору типу розміщення обчислювальних ресурсів, на власних серверах чи використовуючи хмарні сервіси, залежить полегшення підтримки інфраструктурних рішень, їх масштабованість, ефективність та гнучкість.

Зазвичай, хмарні сервіси пропонують вищий рівень абстракції над взаємодією із сервером або ПЗ. Користувачу таких сервісів не потрібно взаємодіяти із процесами ПЗ самостійно, оновлення подібних сервісів відбувається в декілька дій у користувальницькому інтерфейсі.

Хмарне середовище має набагато більші обчислювальні можливості, ніж власно створена інфраструктура серверів, та, відповідно, така кількість ресурсів має багато користувачів. Це дозволяє кожному з них ефективно масштабувати власну інфраструктуру і не усвідомлювати обмежень підконтрольних обчислювальних можливостей.

Оскільки, використовуючи хмарні сервіси, користувач має можливість не обмежувати себе у створенні або видаленні ресурсів, така інфраструктура має високу ефективність, завдяки можливості використовувати сервіси виключно за необхідності.

Хмарні платформи реалізують нові сервіси, стежать за оновленнями ПЗ, розробляють високоефективні процесори і надають ці переваги через власні інтерфейси взаємодії – користувальницькі інтерфейси, консолі, API, SDK, тощо.

Окреме місце хмарних обчислень займають Serverless сервіси. Використання архітектури, побудованої на таких сервісах, повністю звільняє розробників від необхідності керувати серверним обладнанням та дозволяє

зосередитися на бізнес-логіці [1]. Такі рішення автоматично масштабуються відповідно до поточного навантаження, забезпечуючи високу ефективність і гнучкість. Враховуючи ці переваги, Serveless архітектура є конкурентоспроможною опцією проектування веб-сервісів у сучасному цифровому просторі.

В рамках цієї дипломної роботи я розроблю інформаційну систему для редагування зображень на основі Serverless архітектури. Ця платформа використовуватиме такі сервіси, як AWS Lambda для запуску коду, Amazon S3 для масштабованого сховища та Amazon API Gateway для створення API для обробки запитів та інтеграції різних сервісів. Така інфраструктура підкреслює економічну ефективність, масштабованість і швидкість реагування, що є вирішальними для сервісу редагування зображень на вимогу користувача.

Беручи це все до уваги було сформовано тему дипломної роботи: «Розробка автоматизованої інформаційної системи для обробки зображень з використанням Serverless архітектури у контексті Amazon Web Services».

РОЗДІЛ 1
АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ ТА ПОСТАНОВКА ЗАВДАННЯ

1.1. Загальні відомості з предметної галузі

В ситуації швидкого розвитку ПЗ вибір правильної архітектури може мати вирішальне значення для успіху будь-якої програми. Традиційні, монолітні архітектури проклали шлях до більш динамічних рішень, що легше масштабуються.

Монолітна архітектура – це спосіб побудови інформаційної системи, при якому всі компоненти програми тісно пов’язані між собою в межах одного сервісу. Такий підхід спрощує процес розробки і підтримки системи на початковому етапі, але може призвести до проблем по мірі зростання навантаження [2]. Головними недоліками такої архітектури є невигідне масштабування (рис.1.1), оскільки покращення швидкодії певної частини системи потребує масштабування усіх інших частин, і складне розгортання, оскільки для оновлення будь-якої частини програми необхідно перерозгортати усю систему.

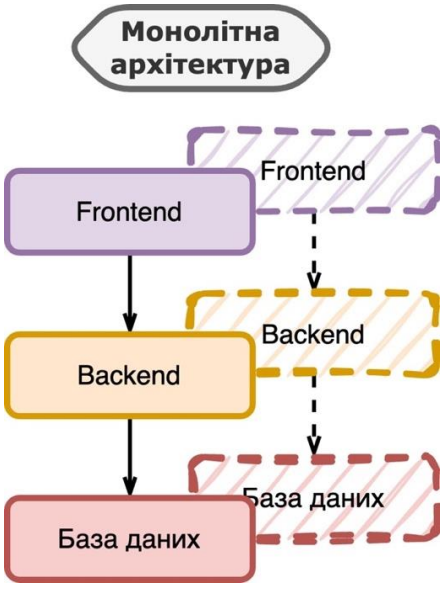

Рис. 1.1. Масштабування монолітної архітектури

Архітектура мікросервісів – це підхід, при якому система розбивається на менші сервіси, що взаємодіють між собою через мережу [3]. Кожен мікросервіс відповідає за окрему бізнес-функцію і може розгортатися незалежно від інших. Це забезпечує більш гнучку розробку, оскільки команди можуть оновлювати окремі компоненти, не впливаючи на решту системи.

Мікросервіси також покращують масштабованість (рис.1.2), оскільки кожен сервіс можна масштабувати окремо, залежно від попиту. Однак архітектура мікросервісів вимагає складної координації, масштабного управління інфраструктурою та надійного налаштування мережевого зв'язку, що може ускладнити підтримку та збільшити витрати.

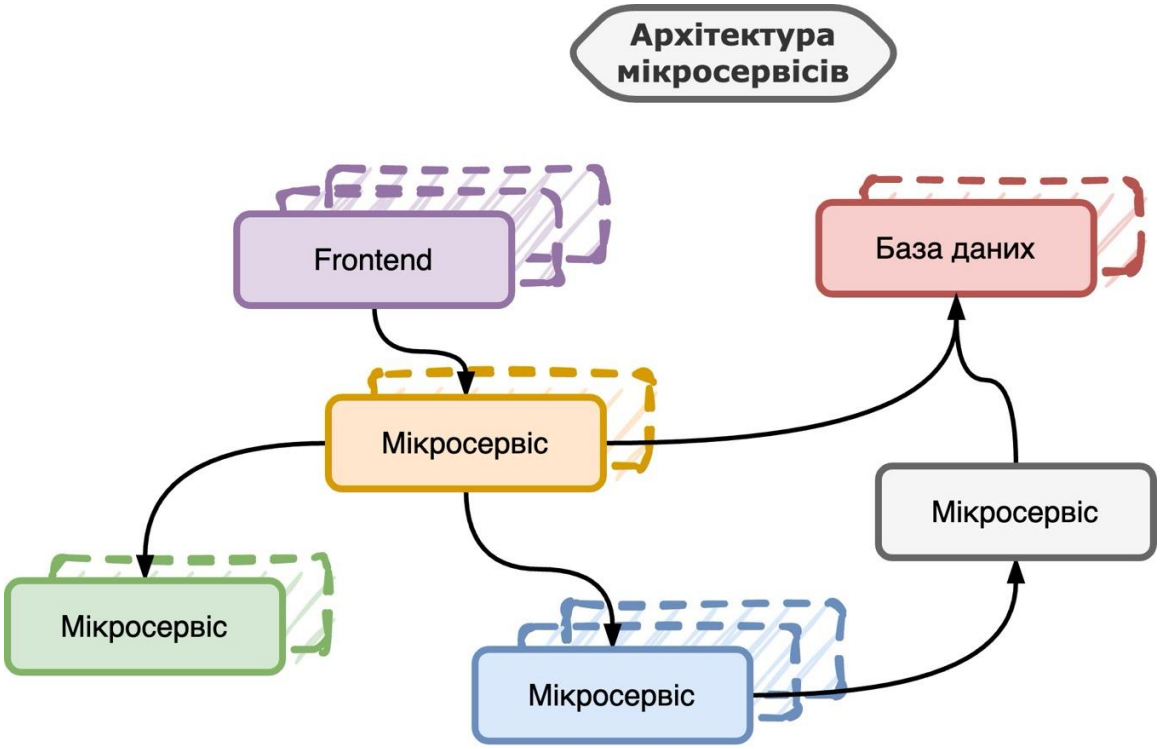

Рис. 1.2. Масштабування архітектури мікросервісів

Serverless архітектура – це модель виконання обчислень, в якій хмарна платформа бере на себе майже повну відповідальність за серверну інфраструктуру використовуваного сервісу.

Наприклад, хмарна платформа AWS явно відмежовує власну відповідальність за безпеку Serverless сервісу AWS Lambda, від відповідальності розробника [4] (рис.1.3).

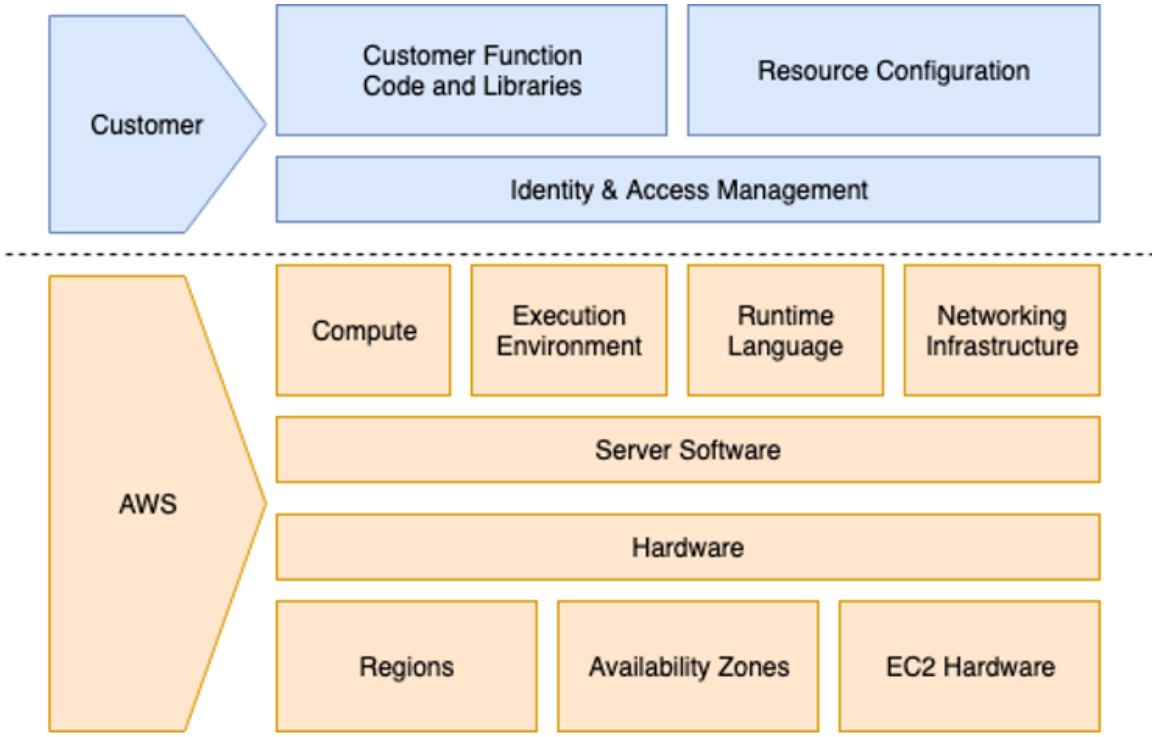

Рис. 1.3. Модель розподіленої відповідальності для AWS Lambda

У такій моделі AWS бере на себе відповідальність за обслуговування інфраструктури та послуг, що лежать в основі сервісу AWS Lambda, керування операційною системою і оточенням для виконання коду. В той час як розробник відповідає за написаний код і управління ідентифікацією та доступом. Така модель дозволяє розробникам сконцентруватись виключно на написанні коду, а не обслуговуванні ПЗ.

Термін “Serverless” може дещо вводити в оману, оскільки для проведення обчислень сервери все ще задіяні, але управління ними повністю абстраговане від розробника.

У звичних серверних системах розробники зазвичай беруть участь у багатьох процесах пов’язаних із обслуговуванням серверу: розгортання,

оновлення ПЗ та масштабування. Такі регулярні заходи можуть відволікати увагу від основних задач кодування.

У Serverless архітектурі всі ці операційні обов'язки абстраговані від розробника. Це не означає, що сервери зникають, вони просто приховані від очей розробника. Хмарні провайдери, такі як AWS, Google Cloud та Microsoft Azure, керують серверами, беручи на себе всі пов'язані з інфраструктурою завдання, такі як надання серверів, масштабування (рис.1.4), обслуговування, оновлення ПЗ та безпека.

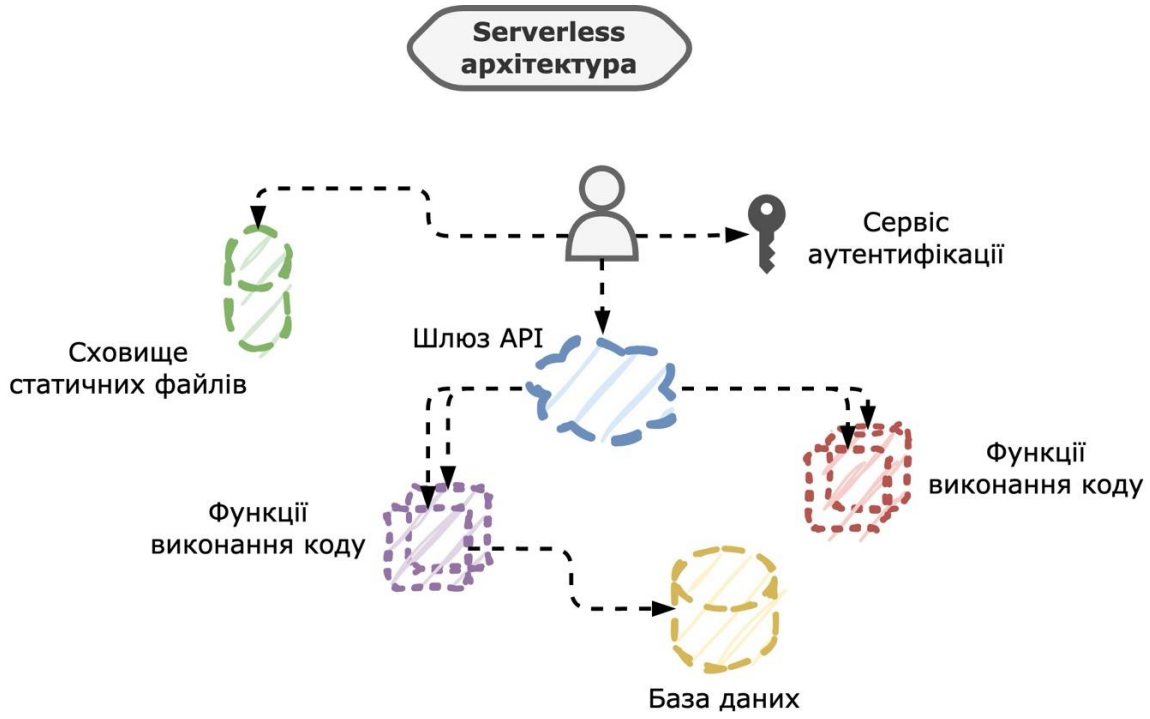

Рис. 1.4. Масштабування Serverless архітектури

До того ж Serverless архітектури забезпечують ефективну модель ціноутворення в хмарних обчисленнях, що базується на оплаті за використання. Ця модель розраховує витрати на основі фактичного використання, усуваючи потребу в попередньо придбаних ресурсах, що у традиційних системах часто призводить до надмірного недовикористання. У Serverless системах витрати розраховуються відповідно до кількості запитів, тривалості виконання коду та

ресурсів, виділених на ці запуски. Це означає, що система несе витрати лише тоді, коли її функції активно працюють.

У порівнянні з монолітною та мікросервісною архітектурами, Serverless архітектура пропонує поєднання масштабованості, ефективної моделі ціноутворення та меншої операційної складності. Це чудове рішення для систем, де важливо максимізувати ефективність та гнучкість у розробці ПЗ, що робить її все більш популярним вибором на цифровому ринку.

Amazon Web Services – це хмарна платформа, що надає широкий спектр сервісів, які можуть забезпечити обчислювальні ресурси, місце для зберігання даних, доставку контенту та інші функціональні можливості. З моменту свого заснування в 2006 році AWS став одним з найважливіших гравців в індустрії хмарних обчислень, пропонуючи понад 200 сервісів із центрами обробки даних по всьому світу.

AWS відомий своїм набором хмарних рішень, пропонуючи як традиційні, так і Serverless сервіси. Це робить AWS універсальним вибором для організацій, які прагнуть використовувати хмару для виконання різноманітних технологічних та бізнес-вимог.

Традиційні послуги AWS включають віртуальні сервери, сервіси для зберігання даних та інші. Такі сервіси, як Amazon EC2 (Elastic Compute Cloud), дозволяють створювати та керувати віртуальними машинами із можливістю налаштування параметрів, що стосуються процесора, пам'яті, сховища та мережі. Amazon RDS (Relational Database Service) та Amazon EBS (Elastic Block Storage) забезпечують роботу розробника з базами даних та взаємодію із віртуальними дисками відповідно. Ці сервіси надають детальний контроль над своїми хмарними ресурсами, подібно до управління фізичним обладнанням, але з додатковою масштабованістю та гнучкістю.

До того ж AWS також пропонує Serverless сервіси (рис.1.4), де за управління серверним обладнанням та програмним забезпеченням відповідає

сам хмарний провайдер, що дозволяє розробникам зосередитися виключно на коді.

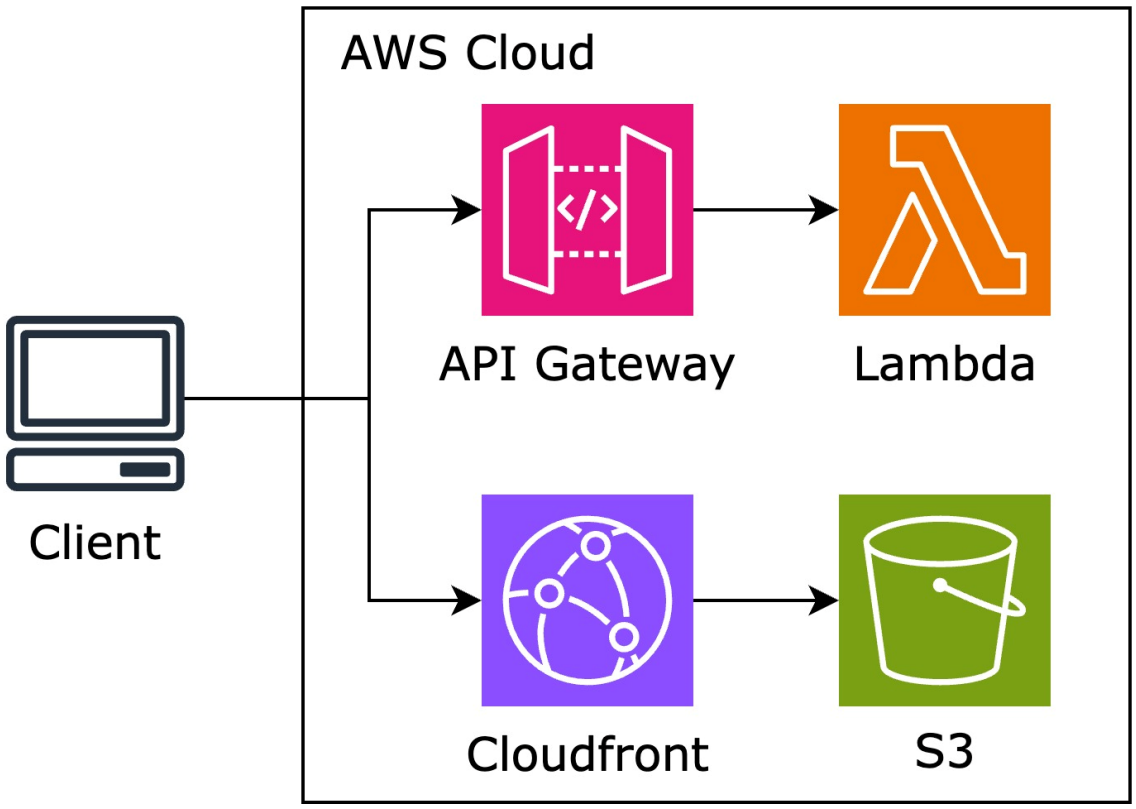

Рис. 1.5. Тривіальна Serverless інфраструктура

AWS Lambda - це сервіс Serverless обчислень, що виконує код у відповідь на події, не вимагаючи жодного резервування або управління серверами. Для запуску AWS Lambda-функції необхідно завантажити свій код на виконання, після цього сервіс створить усі необхідні ресурси для запуску та масштабування цієї функції [5]. Автоматичний запуск коду може бути налаштований у відповідь на події з інших сервісів AWS, також є можливість викликати його безпосередньо з будь-якого веб або мобільного додатку.

Amazon Simple Storage Service (Amazon S3) - це сервіс зберігання об'єктів, відомий своєю довговічністю, доступністю та продуктивністю. Він дозволяє розробникам зберігати необмежену кількість даних та отримувати до них доступ з будь-якої точки світу [6]. Такий підхід ідеально підходить для зберігання

статичних файлів різноманітних додатків, включаючи веб-сайти, мобільні додатки, системи резервного копіювання та відновлення.

Amazon API Gateway - це повністю керований AWS сервіс, який полегшує розробникам процес створення та підтримку API шлюзу. Він виступає інтерфейсом запуску бізнес-логіки системи, дозволяючи розробнику запускати AWS Lambda-функції або взаємодіяти із іншими сервісами AWS. Amazon API Gateway виконує завдання, пов'язані з прийомом і обробкою до сотень тисяч одночасних викликів API, управління доступом, моніторинг та підтримка різних версій API.

Ці сервіси автоматично масштабуються відповідно до попиту, а ціна нараховується лише за спожиті ресурси, прибираючи потребу постійного обслуговування серверів та планування ресурсів.

Ми живемо у світі де в комунікації та самовираженні домінує саме візуальний контент. Творчі люди стикаються із викликами у поширенні та управлінні своїми зображеннями. Існуючі платформи або занадто вузько зосереджені на простому обміні фотографіями, або орієнтовані переважно на професійних фотографів зі складними і часто дорогими функціями. Крім того, більшість платформ не пропонують надійних API, що обмежує творчу свободу користувачів, які бажають програмно взаємодіяти зі своїми зображеннями або інтегрувати ці функції у власні додатки.

Ця прогалина на ринку створює потребу в універсальній, зручній платформі, яка допоможе як професійним, так і аматорським творцям, пропонуючи їм інструменти для легкого завантаження, модифікації та обміну своїми роботами в межах спільноти, що їх підтримує.

Головними функціями такої платформи є:

– завантаження зображень високої якості та їх безпечне хмарне зберігання.
– зручні способи публікації на веб-сайтах.
– гнучкий API, що підтримує трансформацію зображень та зміну формату.

– інтуїтивно зрозумілий дизайн для легкої навігації та взаємодії із платформою.

– перевірка фото на відсутність чутливої інформації та генерація тегів, що описують вміст зображення.

Розглянемо деякі існуючі рішення для поширення та програмного редагування зображень, та оцінимо їх переваги та недоліки.

Flickr – це платформа для обміну фотографіями, орієнтована на любителів та професіоналів фотографії. Це веб-сайт, де кожен може завантажити власні фотографії (рис.1.6) і поділитись ними із сім'єю, друзями або спільнотою фотографів.

Рис. 1.6. Завантаження фото на Flickr

Основними перевагами Flickr є:

– присутність у спільноті фотографів, що сприяє на його популярність та розповсюдженість.

– підтримка завантаження та відображення зображень у високій роздільній здатності.

– можливість управляти правами на перегляд фотографій, забезпечуючи контроль над використанням і поширенням зображень.

– розширений API, який надає розробникам доступ до бібліотеки фотографій і тегів, створених користувачами.

Недоліки Flickr:

– безкоштовний тариф поставляється з обмеженнями щодо зберігання та функціоналу.

– для публікації зображень користувачу необхідно зареєструватись у Flickr.

– платформа призначена виключно для фотографій.

– після редагування зображення платформа модифікує контент за основним URL зображення, що унеможливлює посилання на версії до редагування.

Imgur – це популярний сервіс для розміщення та обміну зображеннями (рис.1.7). Він відомий своєю простотою та активною аудиторією.

Рис. 1.7. Завантаження фото на Imgur

Imgur має широке коло користувачів, в тому числі відвідувачів Інтернет-форумів, фотографів та викладачів. Сервіс надає функціонал швидкого обміну зображеннями та пошуку вірусного контенту.

Основними перевагами Imgur є:

– дозволяє легко завантажувати та ділитися зображеннями.

– функції спільноти, включаючи коментування та голосування, сприяють поширенню вірусного контенту та активній участі користувачів.

– мобільна версія платформи додає зручності використання системи навіть без доступу до персонального комп’ютеру.

Недоліки Imgur:

– відсутність розширених функцій, які можуть знадобитися фотографам чи дизайнерам.

– Imgur часто стискає зображення під час завантаження, що може знизити їх якість.

1.2. Призначення розробки та галузь застосування

Темою бакалаврської дипломної роботи виступає: «Розробка автоматизованої інформаційної системи для обробки зображень з використанням Serverless архітектури у контексті Amazon Web Services». Головною метою кваліфікаційної роботи є створення інформаційної системи, що представляє

собою інструмент для поширення і модифікації зображень, використовуючи Serverless сервіси AWS.

Головними критеріями інформаційної системи є:

– Зручність у використанні.

– Надійне зберігання зображень користувачів.

– Висока доступність системи.

– Швидкий доступ до користувальницьких зображень із будь-якої точки світу.

Система призначена для:

– завантаження зображень та їх хмарне зберігання.

– взаємодії по API, що підтримує трансформацію зображень та зміну формату.

– генерації тегів, що описують вміст зображення.

Система позиціонується як веб-додаток, який дає можливість завантажувати зображення для зберігання, перегляду, поширення та модифікації із можливістю програмної взаємодії використовуючи API.

1.3. Підстави для розробки

Відповідно до ОКХ та ОПП, згідно навчального плану та графіків навчального процесу, в кінці навчання студент виконує кваліфікаційну роботу (проект). Тема роботи узгоджується з керівником проекту, випускаючою кафедрою, та затверджується наказом ректора.

Отже, підставами для розробки (виконання кваліфікаційної роботи) є:

– ОКХ та ОПП за напрямом підготовки 6.050101 «Комп’ютерні науки»;

– графік навчального процесу та навчальний план;

– наказ ректора Національного технічного університету «Дніпровська політехніка» № _____ від __.__.____ р;

– завдання на дипломний проект на тему «Розробка автоматизованої інформаційної системи для обробки зображень з використанням Serverless архітектури у контексті Amazon Web Services».

1.4. Постановка завдання

Метою дипломного проекту є розробка інформаційної системи яка дає можливість завантажувати зображення для зберігання, перегляду, поширення та модифікації із можливістю програмної взаємодії використовуючи API.

Веб-додаток повинен реалізувати такі дії як:

– зберігання завантажених користувачами зображень.

– трансформація зображення за допомогою модифікації URL.

– можливість вбудовувати зображення у зовнішні веб-додатки.

– зберігання усіх версій зображення до та після модифікацій.

– поширення зображень за допомогою Content Delivery Network для швидкого доступу по всьому світу.

– перевірка фото на відсутність чутливої інформації та генерація тегів, що описують вміст зображення.

Для виконання проекту необхідно:

– проаналізувати існуючі рішення для завантаження і поширення зображень.

– спроектувати архітектуру системи.

– втілити архітектуру за допомогою Serverless сервісів AWS.

– реалізувати зручний інтерфейс додатку.

1.5. Вимоги до програми або програмного виробу

1.5.1. Вимоги до функціональних характеристик

Для досягнення поставлених цілей, розроблене програмне забезпечення повинно підтримувати виконання таких дій:

– реагування на дії користувача.

– коректна візуалізація елементів веб-додатку.

– прийняття файлу зображення від користувача та його зберігання у системі.

– модифікація файлу користувача використовуючи Serverless сервіси AWS.

Для підтримки вище перерахованих функцій у інформаційній системі має бути реалізовано:

– доступ до системи за допомогою веб-браузеру.

– програмна та апаратна сумісності.

– підтримка інфраструктури використовуючи підхід IaC.

1.5.2. Вимоги до інформаційної безпеки

Для коректної роботи інформаційної системи необхідно реалізувати:

– метод створення нових версій зображення, при неможливості їх індивідуальної модифікації.

– роботу ресурсів системи виключно по запиту користувача.

– контроль над AWS Identity Access Management. Використання методу найменших привілеїв.

– забезпечення цілісності даних у випадку збою системи.

Також система поширення зображень повинна мати наступні характеристики:

– захист від атак несанкціонованого доступу.

– захист від атак відмови сервісу.

– перевірка зображень на наявність чутливого контенту. Моментальне видалення такого зображення із системи.

1.5.3. Вимоги до складу та параметрів технічних засобів

Для забезпечення функціонування ПЗ необхідно, щоб обчислювальна машина, на якій буде експлуатуватися веб-додаток, мала такі характеристики:

– маніпулятор “миша”.

– клавіатура.

– доступ до Інтернету.

– процесор Intel Core i3-3220 з тактовою частотою 3.3 ГГц.

– не менше ніж 4 Гб оперативної пам’яті.

– монітор з діагоналлю 13".

Вище наведені характеристики є рекомендованими. Це означає, що при наявності характеристик не нижче зазначених, розроблений додаток буде функціонувати відповідно до вимог щодо надійності, безпеки та швидкості обробки даних.

1.5.4. Вимоги до інформаційної та програмної сумісності

Для коректного функціонування програми необхідно, щоб програмне забезпечення обчислювальної машини, на якій буде експлуатуватися веб-додаток, відповідало наступним вимогам:

– веб-браузер Firefox або Google Chrome.

– операційна система Windows 7/10/11.

РОЗДІЛ 2
ПРОЕКТУВАННЯ ТА РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ

2.1. Функціональне призначення системи

Під час виконання дипломної роботи було розроблено інфраструктуру на основі Serverless сервісів AWS для інформаційної системи із функціоналом редагування та поширення зображень.

Призначення розробленої інформаційної системи:

– ефективне поширення завантажених користувачами зображень за допомогою Content Delivery Network для швидкого доступу по всьому світу.

– трансформація зображення за допомогою модифікації URL.

– можливість вбудовувати зображення у зовнішні веб-додатки.

– зберігання усіх версій зображення до та після модифікацій.

– перевірка фото на відсутність чутливої інформації та генерація тегів, що описують вміст зображення.

Для досягнення вище перерахованих функціональних можливостей розроблена програма повинна:

– мати сумісність з стандартною апаратною конфігурацією обчислювальної машини.

– мати підтримку програмного забезпечення, а саме операційних систем Windows 11/10.

– мати підтримку сучасних браузерів Google Chrome та Firefox.

2.2. Опис застосованих математичних методів

Так як особливості предметної області розв'язуваної задачі не передбачають застосування математичних методів, при розробці інформаційної системи математичні методи не використовувалися.

2.3. Опис використаних технологій та мов програмування

Для реалізації інфраструктури проекту було використано хмарний провайдер Amazon Web Services (AWS). AWS - це одна із найбільш розповсюджених хмарних платформ, що пропонує понад 200 сервісів для реалізації інфраструктурних рішень різної складності. AWS має центри обробки даних по всьому світу, що дозволяє будувати системи із низьким рівнем затримки доставки контенту до усіх користувачів [7]. AWS надає різноманітні рішення, які можна використовувати для створення та розгортання будь-якого типу додатків у хмарі. Ці послуги включають в себе обчислювальні потужності, сховища даних, мережеві сервіси, бази даних, машинне навчання, Internet of Things (IoT), та сервіси керування доступом.

AWS пропонує ряд сервісів, що надають компаніям масштабовані обчислювальні потужності. Традиційні, монолітні системи можуть використовувати віртуальні сервери, що надає сервіс Amazon Elastic Compute Cloud (Amazon EC2). Інфраструктури побудові на основі контейнерів можуть основуватись на таких сервісах, як Amazon Elastic Container Service (Amazon ECS) або Amazon Elastic Kubernetes Service (Amazon EKS), що легко керують та масштабують системи такого типу. Для реалізації Serverless рішень AWS пропонує сервіс AWS Lambda, що усуває витрати на управління серверами. Така гнучкість дозволяє розробникам будувати середовища для своїх конкретних потреб.

Сервіси Amazon Simple Storage Service (Amazon S3) для масштабованого зберігання об'єктів, Amazon Elastic Block Store (Amazon EBS), що реалізує блокове сховище даних і Amazon Elastic File System (Amazon EFS) для підтримки гнучкого сховища файлів, формують систему зберігання даних в хмарному середовищі під кожен запит користувача. У цих системах користувачу надається можливість обирати тип сховища в залежності необхідної пропускної здатності носія або, наприклад, частоти доступу до інформації. Це робить перелічені вище

сервіси універсальними для вирішення більшості задач пов’язаних зі зберіганням даних у хмарі.

AWS пропонує сервіси керованих баз даних різного типу. Наприклад, сервіс Amazon Relational Database Service (Amazon RDS) для створення та підтримки реляційних баз даних, Amazon DynamoDB для баз даних NoSQL і Amazon Elasticache для кешів.

Amazon Virtual Private Cloud (Amazon VPC) дозволяє розробникам створювати мережу у хмарі AWS, що є ізольованим місцем для запуску обчислювальних ресурсів AWS.

AWS пропонує такі інструменти, як AWS CloudTrail, AWS Config та AWS CloudWatch для відстеження та керування усіма операціями, що відбуваються у хмарі. Такі сервіси зберігають історію дій усіх користувачів та сервісів і можуть запускати сповіщення, генерувати діаграми історичних даних та відповідати на події у хмарі іншими, заданими користувачем, подіями.

Такі сервіси, як AWS Identity and Access Management (AWS IAM) та Amazon Cognito забезпечують функції ідентифікації та безпеку об’єктів всередині та поза хмарою. Ці сервіси допомагають захистити та контролювати доступ до ресурсів AWS.

AWS підтримує різні архітектурні стилі для розгортання додатків. Кожен розробник має змогу обрати необхідні сервіси для розгортання монолітної, мікросервісної або Serverless архітектури.

Оскільки, інфраструктура розробляємої інформаційної системи базується на Serverless архітектурі, як моделі виконання обчислень, для її реалізації було обрано групу сервісів AWS, що підпадають під цю категорію. Серед них такі сервіси, як AWS Lambda, Amazon S3, Amazon DynamoDB та Amazon API Gateway.

Також, для підтримки інформаційної системи використовуються наступні сервіси AWS: Amazon CloudFront, AWS IAM та Amazon Rekognition.

AWS Lambda - це Serverless сервіс, який дозволяє запускати код без створення або керуванням серверами. AWS Lambda дозволяє розробникам відійти від взаємодії із базовою інфраструктурою та зосередитися на написанні коду та вирішенні бізнес-задач [8]. Розробнику достатньо створити AWS Lambda-функцію та завантажити у неї написаний код.

Ключові особливості AWS Lambda:

– AWS Lambda-функції можуть запускатися зовнішніми HTTP-запитами або сервісами AWS, такими як Amazon S3, Amazon DynamoDB і Amazon API Gateway. Це робить AWS Lambda гарним сервісом для створення масштабованих систем, які реагують на дії користувача в реальному часі.

– сервіс AWS Lambda автоматично масштабує виконання коду відповідно до кількості подій, які він обробляє, гарантуючи, що функція може впоратися з будь-яким рівнем робочого навантаження. AWS Lambda підлаштовується під кількість поточних запитів до функції, незалежно від того, чи це кілька запитів в день або в секунду.

– використовуючи AWS Lambda плата нараховується лише за той обчислювальний час, який фактично було використано. Ціна розраховується на основі кількості запитів і тривалості виконання коду, що вимірюється в мілісекундах, що робить сервіс економічно ефективним рішенням для непередбачуваних робочих навантажень.

– сервіс AWS Lambda інтегрується з AWS Identity and Access Management (IAM) для керування доступом у хмарному середовищі. Крім того, AWS Lambda виконує код у безпечному та ізольованому середовищі, яке керується виключно зі сторони AWS.

– розробники можуть налаштувати обсяг оперативної пам'яті, що виділяється для кожної AWS Lambda-функції окремо. Обсяг пам’яті також визначає обсяг доступної потужності процесора. Така гнучкість дозволяє оптимізувати продуктивність і вартість обчислень відповідно до конкретних вимог кожної функції.

Amazon Simple Storage Service (Amazon S3) - це масштабований сервіс хмарного зберігання даних, призначений для зберігання, резервного копіювання та архівування даних. Amazon S3 є основним компонентом хмарних обчислень, забезпечуючи надійну масштабованість, доступність даних, безпеку та продуктивність.

Ключові особливості Amazon S3:

– Amazon S3 забезпечує високий рівень відмовостійкості і доступності на рівні 99,99%. Дані автоматично копіюються між кількома сховищами в межах обраного регіону AWS.

– Amazon S3 може зберігати необмежену кількість даних. Сервіс автоматично масштабується для задоволення високого попиту, що робить його гарним вибором для зберігання даних будь-якого розміру.

– сервіс зашифровує дані як під час передачі, так і під час простою. Також для налаштування доступні права доступу як на рівні облікових записів, так і на рівні окремих об'єктів.

– завдяки можливості налаштувати клас сховища та правила життєвого циклу об’єктів, Amazon S3 допомагає оптимізувати витрати на основі моделей доступу та зберігання даних.

– Amazon S3 підтримує статичний хостинг веб-сайтів, що дозволяє розробникам розміщувати статичний контент, наприклад, файли HTML, CSS і JavaScript. Ця функція спрощує процес налаштування веб-сайту без необхідності використання сервера, роблячи його економічно ефективним і масштабованим.

Amazon DynamoDB - це повністю керований сервіс підтримки NoSQL баз даних. Сервіс пропонує високу продуктивність з легкою масштабованістю. Amazon DynamoDB призначений для обробки великих обсягів структурованих даних, що вимагають швидкого доступу з низькою затримкою, що робить його популярним вибором для веб, мобільних, ігрових, рекламних технологій, IoT та багатьох інших додатків.

Ключові особливості Amazon DynamoDB:

– Amazon DynamoDB може обробляти велику кількість запитів одночасно. Сервіс забезпечує автоматичне масштабування пропускної здатності, дозволяючи базам даних поступово зростати разом з потребами додатків.

– як повністю керований сервіс, Amazon DynamoDB усуває необхідність адміністративного втручання розробників, пов'язаних з експлуатацією та масштабуванням розподіленої бази даних. Не потрібно керувати серверами, встановлювати програмне забезпечення та оновлювати його.

– Amazon DynamoDB автоматично реплікує дані між трьома географічно розподіленими сховищами AWS в межах регіону, щоб забезпечити високу доступність і довговічність даних.

– сервіс надає вбудовані засоби безпеки, включаючи шифрування в режимі простою. Контроль доступу можна налаштувати за допомогою AWS IAM, гарантуючи, що дані захищені та доступні лише авторизованим користувачам.

Amazon API Gateway – керований AWS сервіс, що виконує всі завдання, пов'язані з прийомом і обробкою великої кількості паралельних викликів API, включаючи управління трафіком, авторизацію і контроль доступу, моніторинг і управління версіями API.

Ключові особливості Amazon API Gateway:

– Amazon API Gateway підтримує RESTful API та WebSocket API, що забезпечує зв'язок між клієнтами та серверами в режимі реального часу.

– сервіс призначений для надійної обробки тисяч паралельних викликів API, автоматично масштабуючись залежно від трафіку.

– Amazon API Gateway пропонує кілька рівнів безпеки. Він інтегрується з AWS IAM для контролю доступу. Сервіс також підтримує використання OAuth токенів.

– розробники можуть запускати кілька версій одного API одночасно, що дозволяє їм тестувати нові версії, не впливаючи на існуючий API.

– інтеграція з AWS CloudWatch дозволяє відстежувати та реєструвати виклики API. Це допомагає зрозуміти та оптимізувати продуктивність API.

Amazon CloudFront - це сервіс, що надає можливість реалізувати Content Delivery Network (CDN) для власних інформаційних систем [9]. Він призначений для доставки текстових даних, зображень, відео та API користувачам по всьому світу з низькою затримкою та високою швидкістю передачі.

Ключові особливості Amazon CloudFront:

– CloudFront має мережу регіональних кешів, які зберігають копії контенту ближче до користувачів. Ця мережа оптимізує доставку контенту з мінімальною затримкою та високою швидкістю передачі даних.

– CloudFront інтегрується з сервісами AWS, такими як Amazon S3, Elastic Load Balancing, Amazon EC2 та Amazon Route53. Це дозволяє спростити налаштування CDN-сервісів і пришвидшити доставку контенту.

– CloudFront використовує модель оплати по факту використання, тобто ціна використання включає лише плату за контент, який фактично було доставлено через CDN. Ціна залежить від регіону, з якого обслуговуються дані, та загального обсягу переданих даних.

Amazon Rekognition - це сервіс візуального аналізу на основі машинного навчання. Він дозволяє розробникам реалізувати аналіз зображень і відео у власних системах. Amazon Rekognition має функціонал ідентифікації об'єктів, людей, тексту та оточення на зображеннях і відео, а також виявляти будь-який неприйнятний контент.

– Amazon Rekognition може виявляти потенційно небезпечний або неприйнятний вміст як у зображеннях, так і у відеофайлах [10]. Ця функція допомагає вчасно виявити та відфільтрувати файли із чутливим вмістом.

– функція розпізнавання може ідентифікувати тисячі об'єктів (наприклад, транспортні засоби, домашніх тварин або меблі) і оточення (наприклад, пляж, місто або захід сонця). Ця функція допомагає автоматично каталогізувати зображення.

Для реалізації клієнтської частини інформаційної системи було використано HTML, CSS та JavaScript.

HTML – це стандартна мова розмітки, яка використовується для створення та структурування веб-сторінок. Елементи HTML є будівельними блоками всіх веб-сайтів, забезпечуючи структуру, розмежовуючи різні частини веб-сторінки, такі як заголовки, абзаци, посилання та інший вміст.

CSS – це мова таблиць стилів, яка використовується для опису представлення документа, написаного на HTML. CSS покращує зовнішній вигляд веб-контенту, дозволяючи розробникам створювати візуально привабливі веб-сторінки. Вона керує макетом, кольорами, шрифтами та всіма іншими аспектами зовнішнього вигляду веб-сайту.

JavaScript – це мова програмування, яка дозволяє створювати інтерактивні веб-сторінки. Вона може взаємодіяти та змінювати HTML і CSS, керувати мультимедіа та робити зовнішні запити. JavaScript є інструментом для взаємодії з бізнес-логікою інформаційної системи, полегшуючи обмін даними між користувачем і процесами на стороні сервера.

Для написання бізнес-частини інформаційної системи було використано Python. Наявність бібліотеки boto3 робить його гарним вибором для взаємодії з сервісами AWS.

Python - це високорівнева мова програмування, що підтримує декілька парадигм програмування та має велику кількість бібліотек і фреймворків, що дозволяє використовувати її в системах різного типу.

boto3 - це SDK для мови Python, що дозволяє розробникам створювати програмне забезпечення, яке взаємодіє із такими сервіси, як Amazon S3, Amazon EC2, Amazon DynamoDB та інші [11].

Важливою складовою створення та підтримки інфраструктури є метод управління ресурсами, які її утворюють, оскільки ефективне управління інфраструктурою суттєво впливає на масштабованість та надійність бізнесу в цілому. Метод задання ресурсів інфраструктури “Infrastructure as Code” (IaC)

наразі стала стандартом ефективного управління наявними ресурсами. IaC забезпечує системний підхід до управління інфраструктурою за допомогою набору файлів визначень наявних ресурсів та їх характеристик, замість того, щоб фізично конфігурувати систему [12].

IaC дозволяє розробникам автоматично створювати, контролювати і видаляти ресурси, замість того, щоб займатися налаштуванням і конфігурацією вручну. Такий підхід є більш зручним, надійним та ефективним способом управління інфраструктурою. Завдяки автоматизації, IaC економить час, забезпечує уніфіковане застосування конфігурацій і створює “єдину точку правди” описуючи існуючі ресурси та їх параметри у вигляді конфігураційних файлів.

У сфері IaC інструмент Terraform є гарним рішенням для управління інфраструктурою різного розміру та складності. Він відзначається декларативною структурою коду, створюючи “єдину точку правди” для існуючих ресурсів та дозволяє підтримувати ресурси різних хмарних провайдерів [13].

Terraform детально відстежує внутрішні залежності між ресурсами інфраструктури. Певні ресурси, такі як, наприклад, правила безпеки доступу до віртуальних машин, повинні бути створені до того, як можуть бути створені самі віртуальні машини. Terraform автоматично виявляє ці залежності, гарантуючи правильний порядок дій під час створення ресурсів. Для складних сценаріїв можна використовувати аргумент depends_on, щоб явно визначити послідовність створення, оновлення або видалення ресурсів [14]. Такий детальний контроль до залежностей сприяє плавному і передбачуваному розгортанню інфраструктури.

Однією із ключових функцій Terraform є система управління поточним станом інфраструктури. Terraform використовує центральний файл стану (рис.2.1), який відстежує конфігурацію всіх керованих ресурсів, діючи як загальний план існуючої інфраструктури. Файл стану відіграє важливу роль у підтримці узгодженості між існуючими ресурсами. Порівнюючи поточний стан з бажаним станом інфраструктури, визначеним у конфігураційних файлах,

Terraform визначає необхідні дії для оновлення інфраструктури, забезпечуючи надійне та ефективне управління ресурсами [15].

Рис. 2.1. Принцип роботи файлу стану Terraform

Terraform надає два різних типи блоків для побудови інфраструктури: блоки даних і блоки ресурсів. Блоки даних діють як агенти пошуку інформації про існуючі об’єкти, отримуючи дані із зовнішніх джерел, таких як API хмарних провайдерів, та інтегруючи їх в поточну конфігурацію. Використання блоків даних сприяє написанню чистого і лаконічного коду, без необхідності жорсткого кодування значень. Блоки ресурсів визначають конкретні елементи інфраструктури. Тут вказується тип ресурсу, наприклад, віртуальна машина, і його параметри. Terraform взаємодіє з хмарними платформами, на основі написаних блоків ресурсів для створення, оновлення або видалення відповідних ресурсів. Таке розмежування між пошуком даних і визначенням ресурсів забезпечує організований і структурований підхід до управління інфраструктурою.

Змінні є важливим функціоналом Terraform, що підвищує гнучкість і можливість багаторазового використання конфігурацій. Вони дозволяють застосування однієї і тієї ж конфігурації в різних умовах, змінюючи значення змінних замість основної логіки коду. Змінні також підвищують безпеку, дозволяючи передавати конфіденційні дані, такі як паролі або ключі API, без їх жорсткого кодування у файлах конфігурації. Змінні Terraform відіграють важливу роль у підтримці масштабованих інфраструктур, що робить їх важливим інструментом у сфері IaC.

Ще однією особливістю Terraform є підтримка публічних і приватних модулів. Модулі - це контейнери, які містять декілька ресурсів, що використовуються разом, дозволяючи позбавитися перевикористання коду інфраструктури [16].

Публічні модулі доступні через Terraform Registry та містять шаблони створені спільнотою Terraform. Ці шаблони можна використовувати якими вони є або адаптувати до персональних потреб, заощаджуючи час на розробку. Приватні модулі призначені для групування ресурсів локально.

2.4. Опис структури системи та алгоритмів її функціонування

Результатом реалізації автоматизованої інформаційної системи редагування зображень з використанням Serverless архітектури у контексті AWS стала інфраструктура поєднаних між собою сервісів (рис.2.2), що працюють лише за необхідності та за наявності запиту користувача. Створена інфраструктура ефективно використовує хмарні технології для створення надійного, масштабованого та безпечного середовища для розгортання інформаційних систем. Ця інфраструктура використовує можливості AWS використовуючи такі сервіси, як Amazon S3 для статичного хостингу веб-сайту, AWS Lambda для Serverless обчислень та Amazon API Gateway для управління трафіком API. Ця комбінація дозволяє створити систему, що може

масштабуватися відповідно до вимог програми, забезпечуючи ефективну роботу і зниження операційних витрат.

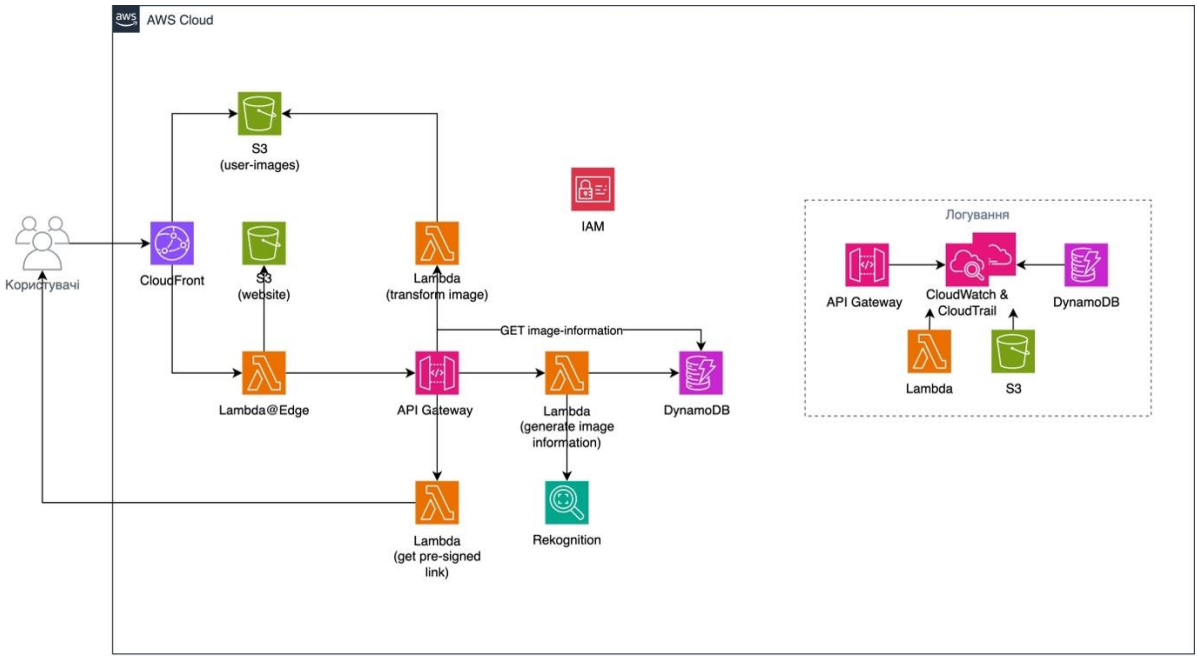

Рис. 2.2. Інфраструктура розробленої інформаційної системи

Дизайн інфраструктури підтримує високу доступність і відповідає найкращим практикам у сфері управління даними. Завдяки використанню керованих сервісів AWS, такі завдання як оновлення, масштабування та безпека, виносяться на другий план, дозволяючи зосередитися на розробці основних функціональних можливостей програми.

Використовуючи зазначену інфраструктуру створено сервіс редагування та поширення зображень, із метою надати користувачам універсальне рішення для завантаження зображень, їх редагування та ефективного обміну із користувачами. Інформаційна система розроблена для оптимізації роботи користувачів шляхом розповсюдження зображень по всьому світу, що забезпечує доступ до контенту з мінімальними затримками та високою швидкістю завантаження. Реалізований із використанням хмарних технологій та технологій штучного інтелекту, сервіс підтримує динамічні маніпуляції із зображеннями,

сканування на наявність чутливого контенту та генерацію метаданих ідентифікації об'єктів.

Під час розробки окрема увага приділялася інтеграції технологій розпізнавання зображень, які сканують зображення на наявність чутливого контенту. Ця функціональність не лише автоматизує модерацію контенту, але й гарантує, що завантаження не порушують безпечне середовище для обміну контентом.

Точкою входу в інфраструктуру є сервіс Amazon CloudFront, що виступає основним компонентом в оптимізації доставки контенту користувачам по всьому світу. Власний CloudFront дистрибутив дозволяє налаштувати перелік першоджерел, такі як Amazon S3 сховища для статичного контенту, екземпляри Amazon EC2 для динамічного контенту або будь-який HTTP-сервер для розподілення трафіку. Таке налаштування дозволяє централізовано керувати різноманітними джерелами контенту, гарантуючи, що всі запити користувачів будуть маршрутизовані відповідно до визначених правил.

Amazon CloudFront реалізує функцію з підвищення безпеки доставки контенту при взаємодії з Amazon S3 сховищами: використання HTTPS як основного протоколу взаємодії із інформаційною системою. Використовуючи сертифікати SSL/TLS, якими можна керувати через AWS Certificate Manager, CloudFront гарантує, що дані, які передаються між клієнтом і CDN, зашифровані [17]. Це особливо важливо при обслуговуванні статичного контенту, що зберігається в S3-бакетах, оскільки це запобігає перехопленню даних і забезпечує цілісність і конфіденційність інформації, що надається користувачам. Під час налаштування дистрибутива CloudFront було встановлено можливість прийому лише HTTPS-з'єднань Ця можливість необхідна для дотримання найкращих практик безпеки та підтримки довіри користувачів до платформи.

В архітектурі інформаційної системи першоджерелами дистрибутиву Amazon CloudFront виступають Amazon S3 сховища. Ця інтеграція дозволяє ефективно зберігати та доставляти статичні ресурси.

Оскільки метою дипломної роботи є побудова Serverless інфраструктури однією із характеристик якої є ефективна цінова модель, важливо притримуватись цієї ідеї на всіх стадіях розробки системи. Тож, як місце зберігання файлів веб-сторінки було обрано сховище Amazon S3.

Amazon S3 пропонує рішення для хостингу статичних веб-сайтів, що дозволяє розгортати веб-сторінки, які містять HTML, CSS, JavaScript та інші статичні ресурси , безпосередньо із Amazon S3 сховища (рис.2.3).

Рис 2.3. Вміст Amazon S3 сховища

Amazon S3 дозволяє налаштувати веб-хостинг без необхідності використання традиційного сервера [18]. Необхідно створити сховище, завантажити файли та налаштувати властивості сховища для обслуговування веб-сторінок (рис.2.4). Однією з ключових переваг використання S3 для статичного хостингу веб-сайтів є його здатність обробляти великі обсяги трафіку та висока надійність даних, що гарантує, що дані такого веб-сайту завжди будуть доступними для кінцевих користувачів.

Рис. 2.4. Налаштування хостингу веб-сайту

Будь-який запит, що надходить на веб-сайт розміщений із використанням Amazon S3 обслуговується безпосередньо Amazon S3. Оскільки Amazon S3 розроблений для роботи в першу чергу із об’єктами, а не трафіком, навантаження розподіляється без використання традиційних серверів, які можуть потребувати масштабування. Інфраструктура Amazon S3 може ефективно обробляти великі обсяги запитів завдяки своїй побудові та підтримці мережевої інфраструктури Amazon.

Amazon S3 також підтримує налаштування правил перенаправлення та статичних сторінок у відповідь на помилки. Правила перенаправлення можуть бути корисними для перенаправлення відвідувачів зі старих URL-адрес на нові або для маршрутизації користувачів у межах сайту, покращуючи його навігацію. Що стосується обробки помилок, то можна вказати спеціальний документ у відповідь на помилку, щоб покращити взаємодію з користувачем, надаючи більш дружні повідомлення, коли щось йде не так.

В той час як одне сховище Amazon S3 використовується для розміщення файлів веб-сайту, також було створено інше сховище для зберігання зображень, що користувачі завантажують на платформу. Таке інфраструктурне рішення було прийняте виходячи із ідеї спрощення підтримки та розмежування різного типу файлів у двох різних за своїм призначенням логічних сховищах. Обидва Amazon S3 сховища використовують Amazon CloudFront для ефективної та швидкої доставки контенту до користувачів по всьому світу. Таке рішення позитивно впливає як на користувальницький досвід, так і на економічну частину інфраструктури, оскільки використання кешованого контенту зменшує навантаження на оригінальний контент у місці його зберігання.

На діаграмі інфраструктури можна побачити що між сервісами Amazon CloudFront та сховищем Amazon S3, що використовується для розміщення файлів веб-сайту, знаходиться сервіс AWS Lambda@Edge.

AWS Lambda@Edge - це розширений функціонал сервісу AWS Lambda, який дозволяє обробляти дані користувачів ще на стадії їх кешування (рис.2.5), не доходячи до оригінального місця зберігання, завдяки можливості виконання коду в місцях розташування AWS, що є ближчими для кожного користувача. Такий підхід зменшує затримки та підвищує продуктивність при обслуговуванні контенту в Amazon CloudFront [19].

AWS Lambda@Edge дозволяє виконувати функції у відповідь на події CloudFront, такі як запит від глядача або відповідь глядачу, без перенаправлення трафіку напряму до джерела.

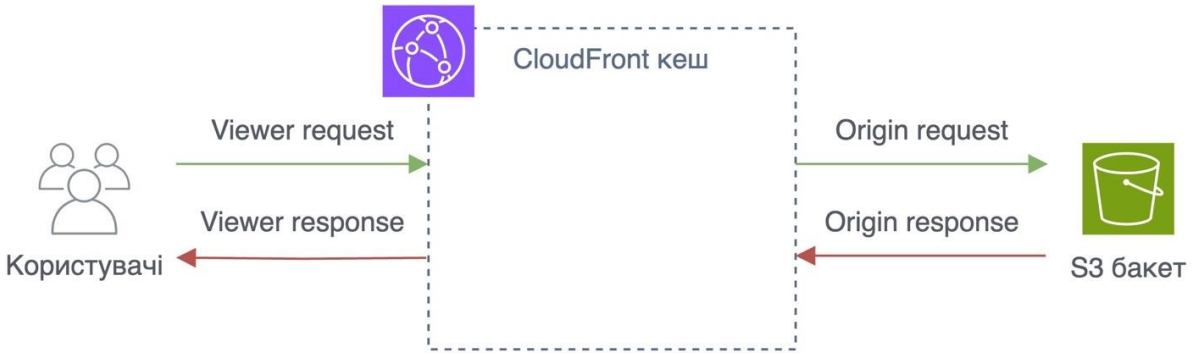

Рис. 2.5. Принцип роботи AWS Lambda@Edge

Такі функції можуть генерувати відповідь на основі місцезнаходження глядача, типу пристрою, заголовків, файлів cookie або шляхів URI. У випадку інформаційної системи редагування зображень цей функціонал було використано для розмежування взаємодії користувача із веб-інтерфейсом та API платформи в залежності від присутнього у URI запиту відповідного префіксу.

Якщо шлях URI починається з /i/, за яким слідує ідентифікатор зображення, функція створює шлях для доступу до сторінки редагування зображення, що зберігається в сховищі користувальницьких зображень Amazon S3 та перенаправляє користувача на цю сторінку.

Для шляхів, що починаються з /api/, функція змінює запит на перенаправлення до кінцевої точки API-шлюзу, де обробляється динамічний вміст або логіка програми.

Якщо шлях URI не містить зазначених вище префіксів, запит направляється на сховище Amazon S3, де зберігаються ресурси веб-сайту, такі як файли HTML, CSS і JavaScript, тобто сторінки веб-сайту.

У випадку взаємодії із API AWS Lambda@Edge-функція перенаправляє трафік на сервіс Amazon API Gateway, який є вхідною точкою до взаємодії із бізнес-логікою інформаційної системи.

В архітектурі інформаційної системи Amazon API Gateway відіграє роль центрального каналу, через який відбувається вся взаємодія з API. Цей сервіс

виступає в ролі точки управління, обробки та маршрутизації викликів API до різних сервісів, включаючи AWS Lambda-функції та Amazon DynamoDB. Використовуючи API Gateway, інформаційна система безпечно і масштабовано обробляє трафік API, що робить платформу більш динамічною.

AWS API Gateway налаштований на маршрутизацію трафіку до певних AWS Lambda-функцій на основі шляху та методу запиту. Таке налаштування дозволяє чітко розділити завдання в архітектурі програми, коли різні лямбда-функції виконують різні завдання, такі як обробка зображень або пошук метаданих. Після того, як AWS Lambda-функції обробляють дані, вони можуть взаємодіяти з Amazon DynamoDB для отримання або зберігання даних за потреби. Ця інтеграція є прикладом Serverless архітектури, яка підвищує продуктивність за рахунок зменшення відстані, яку дані повинні проходити між сервісами.

Таблиця 2.1

Логіка маршрутизації трафіку сервісом Amazon API Gateway

Кінцева точка	Характеристика кінцевої точки	
	Метод	Опис дії у хмарному середовищі
/api/	Root	Надає інформацію про API. Базовий шлях для всіх запитів API
/api/get-pre-signed-url	GET	Перенаправляє на AWS Lambda-функцію для отримання pre-signed URL-адреси для операцій із об’єктами сховища Amazon S3
/ base64/{id}	GET	Перенаправляє на AWS Lambda-функцію для отримання зображення зі сховища Amazon S3 у форматі base64 за його ідентифікатором
/base64/	POST	Перенаправляє на AWS Lambda-функцію що конвертує зображення з формату base64 і

		завантажує його до сховища Amazon S3 та повертає його ідентифікатор
/labels/{id}	GET	Перенаправляє на базу даних DocumentDB та повертає інформацію про вміст зображення користувачу

Функція редагування зображень реалізована за допомогою параметрів запиту в поєднанні зі Amazon API Gateway. Такий підхід дозволяє користувачам вказувати свої команди редагування безпосередньо в URL-адресі запиту. Параметри можуть комбінуватися в залежності від потреб користувача.

Таблиця 2.2

Параметри запиту редагування зображення

Параметр	Дія
rotate	Вказує на бажану дію повертання зображення
angle	Вказує кут повертання зображення
flip	Вказує на бажану дію відображення зображення
direction	Вказує напрямок відображення зображення
black_and_white	Вказує на бажану дію зміни кольору зображення на чорно-білий
format	Вказує формат файлу зображення

Коли користувач бажає відредагувати зображення, він може додати до URL-адреси пошуку зображення різні параметри запиту, які визначають конкретні трансформації або редагування. Наприклад, параметри можуть визначати такі дії,

як зміна розміру, відображення або зміна формату файлу. Кожен параметр представляє різні перетворення, які можна застосувати до зображення.

Процес завантаження зображень є безпечним і простим для реалізації завдяки функціоналу AWS Lambda-функції, яка створює Amazon S3 pre-signed URL-адрес для прямого завантаження файлів у сховище Amazon S3.

Amazon S3 pre-signed URL-адреса - це інструмент, призначений для надання тимчасового доступу до об'єктів, що зберігаються в сховищі Amazon S3. По суті, це URL-адреса, яку генерує серверна частина інформаційної системи, надаючи користувачам обмежений дозвіл на виконання певної дії - наприклад, завантаження або вивантаження файлу. Ця URL-адреса містить усю необхідну інформацію для автентифікації, тож користувачеві не потрібні облікові дані AWS або дозволи для прямого доступу до об'єкта.

Генерація Amazon S3 pre-signed URL-адреси включає в себе посилання на сховище Amazon S3 і об'єкт, до якого користувач отримає доступ, тип запиту і час закінчення терміну дії, після якого URL-адреса стане недійсною. Це важливо для підвищення безпеки платформи, оскільки знижує ризик несанкціонованого доступу. Amazon S3 pre-signed URL-адреси є універсальними і можуть бути налаштовані з різними умовами для посилення безпеки і контролю над операціями у сховищі. Наприклад, при створенні URL-адреси для завантаження зображень у інформаційну систему, було встановлено обмеження на розмір і тип файлу, файл обов’язково має бути зображенням, не більшим ніж 10 мегабайт за розміром.

Коли користувач звертається до шляху /api/get-pre-signed-url, Amazon API Gateway спрямовує цей запит до AWS Lambda-функції, спеціально розробленої для створення Amazon S3 pre-signed URL-адреси. Після створення URL-адреса надсилається назад користувачеві через шлюз API, що дозволяє йому завантажити зображення безпосередньо на S3, тим самим спрощуючи процес, усуваючи необхідність обробки передачі файлів платформою.

Користувачі мають можливість завантажувати свої зображення безпосередньо через API, кодуючи їх у формат base64. Цей метод реалізовано завдяки кінцевій точці /api/base64, куди користувачі можуть надсилати свої зображення, закодовані в base64, як частину тіла запиту. Після отримання AWS Lambda-функція обробляє ці закодовані рядки, перетворюючи їх назад у файли зображень і завантажуючи їх до загального сховища.

Використання кодування base64 для завантаження зображень є зручним для користувачів, оскільки формат base64 підтримується всіма популярними мовами програмування. Отже, розробники можуть реалізувати цю функціональність незалежно від середовища розробки. Функції для кодування і декодування base64 доступні в таких мовах, як Python, JavaScript, Java і багатьох інших, що робить цей метод доступним і простим для інтеграції в різні додатки. Цей підхід спрощує процес обробки зображень, дозволяючи розглядати їх як звичайний текст, який можна легко передавати за стандартними протоколами HTTP без необхідності використання форм або інших методів завантаження файлів.

Окрім завантаження зображень, інформаційна система також дозволяє користувачам отримувати зображення у зручному форматі для використання на власних платформах.

Ця функціональність надається через кінцеву точку /api/base64/{id}, де користувачі можуть запитувати зображення за його унікальним ідентифікатором. При зверненні до цієї кінцевої точки зображення завантажується зі сховища, конвертується у формат base64 та повертається користувачу у вигляді рядка, закодованого в base64.

Можливість отримувати зображення у форматі base64 особливо корисна для користувачів, яким потрібно інтегрувати ці зображення безпосередньо у веб-сторінки, мобільні додатки або інші платформи, що підтримують кодування base64. Оскільки зображення закодоване у base64 можна вбудовувати безпосередньо в HTML файли без необхідності виконання додаткових дій, цей метод спрощує управління контентом і прискорює завантаження веб-ресурсів,

усуваючи окремі HTTP-запити для завантаження файлів зображень. Це також покращує перенесення даних і простоту використання в різних системах і середовищах програмування, оскільки робота з текстовими даними часто менш складна, ніж з двійковими файлами.

Ще одна функція, яку надає Amazon API Gateway – це можливість отримувати метадані ідентифікації об'єктів та оточення на зображеннях за допомогою кінцевої точки /api/labels/{id}.

Ця функція дозволяє користувачам отримувати доступ до метаданих про зображення, таких як описи вмісту та теги, за допомогою запиту з унікальним ідентифікатором зображення. Коли робиться запит до цієї кінцевої точки, Amazon API Gateway забезпечує прямий запит до Amazon DynamoDB, де зберігаються ці метадані. Процес реалізовано наступним чином: як тільки API-шлюз отримує запит на кінцеву точку /api/labels/{id}, він запускає запит до Amazon DynamoDB, щоб отримати метадані, пов'язані з даним ідентифікатором зображення. Ці метадані зазвичай містять описову інформацію, яка класифікує або описує вміст зображення, наприклад, "ліс", "захід сонця" або "натовп". Ці метадані можуть застосовуватись по-різному, наприклад, для організації великих бібліотек зображень, покращення пошукової функціональності шляхом увімкнення пошуку на основі вмісту зображень або для реалізації систем рекомендацій на основі вмісту.

Безпосереднє отримання цих записів з DynamoDB є дуже ефективним рішенням, оскільки використовує здатність DynamoDB обробляти велику кількість запитів і відповідати на них з низькою затримкою. Така інтеграція спрощує архітектуру, зменшуючи кількість компонентів, задіяних у процесі отримання даних і гарантує швидке отримання метаданих, підвищуючи швидкість відповіді платформи.

В інформаційній системі процес генерації метаданих зображень повністю автоматизовано завдяки архітектурі, керованій подіями, з використанням Amazon Rekognition для складного аналізу зображень.

При застосуванні методики, коли діями в інфраструктурі керують події в хмарному середовищі, можна значно заощадити час на реалізацію тригерів AWS Lambda-функцій. Наприклад, завантаження зображення у сховище Amazon S3 може автоматично запустити AWS Lambda-функцію для обробки або аналізу зображення, оновлення баз даних і відображення змін у користувацькому інтерфейсі без ручного втручання.

Процес починається, коли нове зображення завантажується користувачем у сховище Amazon S3. Ця дія автоматично запускає подію (рис.2.6), яка викликає AWS Lambda-функцію, спеціально розроблену для цього завдання. AWS Lambda-функція отримує зображення з Amazon S3 і надсилає його до Amazon Rekognition, що аналізує зображення за допомогою алгоритмів машинного навчання. Amazon Rekognition здатний виявляти та ідентифікувати різні об'єкти та сцени на зображенні і на основі цього аналізу генерує набір описових метаданих.

Рис. 2.6. Налаштування тригеру AWS Lambda-функції

Після генерації метаданих AWS Lambda-функція зберігає їх у базі даних Amazon DynamoDB, пов'язуючи кожен набір метаданих з відповідним ідентифікатором зображення. Такий спосіб зберігання забезпечує швидкий і легкий пошук метаданих за потреби. Наприклад, коли користувач запитує метадані для певного зображення через кінцеву точку /api/labels/{id}, надану Amazon API Gateway, система отримує ці метадані з Amazon DynamoDB. Користувач отримує відповідь, що містить усі відповідні описові метадані, що дозволяє одразу зрозуміти і класифікувати вміст зображення.

Платформа дозволяє користувачам редагувати зображення використовуючи параметри в своїх API-запитах.

Окрема AWS Lambda-функція дозволяє користувачам вказати бажані параметри редагування, такі як зміна кольору, перевертання або відображення, просто додавши ці специфікації до URL-адреси виклику API. Amazon API Gateway відіграє ключову роль в управлінні виконанням цих модифікацій зображень оскільки, коли робиться такий запит з параметрами, саме цей сервіс отримує вхідні параметри та передає їх до відповідної AWS Lambda-функції.

AWS Lambda-функція використовує бібліотеку редагування зображень Python PIL, яка підтримує відкриття, маніпулювання та збереження різних форматів файлів зображень AWS Lambda-функція аналізує параметри запиту, щоб визначити модифікації, яких вимагає користувач. Використовуючи можливості бібліотеки PIL, вона виконує зміни над зображенням [20]. Якщо в параметрах запиту вказано напрямок відображення зображення, лямбда-функція використовує PIL для відповідного відображення зображення.

В інформаційній системі усі AWS Lambda-функції налаштовані на використання архітектури ARM (рис.2.7), зокрема процесорів Graviton від AWS. Це рішення зумовлене ефективністю та економічністю, які пропонують процесори на базі ARM, особливо в хмарних обчисленнях.

Рис. 2.7. Конфігурація архітектури AWS Lambda

Процесори AWS Graviton оптимізовані для хмарних робочих навантажень і призначені для забезпечення балансу обчислювальних та мережевих ресурсів для хмарних систем. Процесори Graviton забезпечують значне покращення показників співвідношення ціни та продуктивності у порівнянні з процесорами x86 попереднього покоління для різних робочих навантажень.

Використовуючи архітектуру ARM та процесори Graviton для лямбда-функцій, інформаційна система гарантує ефективне виконання завдань обробки

зображень як з точки зору продуктивності, так і з точки зору вартості [21]. Цей архітектурний вибір не лише підвищує швидкість реагування сервісу та зменшує загальне енергоспоживання хмарної інфраструктури.

Для реалізації інфраструктури AWS та ефективного управління її ресурсами було обрано Terraform як інструмент Infrastructure as Code (IaC).

Terraform дозволяє керувати хмарною інфраструктурою за допомогою простих, декларативних конструкцій програмування. Це надійне рішення для створення, модифікації та управління складною хмарною архітектурою протягом її життєвого циклу.

При створенні інфраструктури було впроваджено модульну систему з використанням Terraform. Такий підхід передбачає організацію інфраструктури в окремі модулі, де кожен модуль описує певну групу ресурсів. Використання модулів покращує керованість, можливість повторного використання та зрозумілість кодової бази інфраструктури. Наприклад, для різних компонентів архітектури, таких як dynamodb/, lambda/ та інших, були створені окремі модулі. Кожен з цих модулів відповідає за створення та конфігурацію ресурсів, пов'язаних з конкретною частиною інфраструктури.

Управління файлом стану Terraform з урахуванням безпеки та контролю версій є важливою практикою. Файл стану, який Terraform використовує для зіставлення реальних ресурсів з існуючою конфігурацією зберігається віддалено у сховищі Amazon S3, а не локально. Такий підхід підвищує безпеку файлу стану і додає можливість контролю версій.

2.5. Обґрунтування та організація вхідних та вихідних даних програми

Вхідними даними розробленої інформаційної системи є надані користувачем зображення, а також запити до API інтерфейсу.

Ці дані включають зображення, що користувачі надають через веб-інтерфейс або програмно через API. Інформаційна система приймає зображення

у різних форматах, таких як, наприклад, JPEG або PNG. У випадку програмної взаємодії користувачі можуть використовувати зображення конвертовані у текстовий формат base64.

Користувачі також можуть подавати запити на редагування зображень, що зберігаються в системі. Ці запити здійснюються через кінцеві точки API, які отримують вхідні дані у вигляді параметрів, що визначають необхідні редагування. Ці параметри обробляються AWS Lambda-функціями, які виконують потрібні перетворення.

Вихідними даними інформаційної системи є публічно доступні відредаговані зображення та метадані ідентифікації об'єктів та оточення на цих зображеннях.

Результатом редагування зображення є новий файл зображення, який зберігається у сховищі Amazon S3. Система надає вихідне посилання на відредаговане зображення, що дозволяє легко ділитися ним та інтегрувати його у власні платформи.

Інформаційна система надає метадані ідентифікації об’єктів, пов'язані із зображеннями. Ці метадані служать інструментом для розуміння користувачами змісту зображень та надання контекстної інформації в пошуку для користувальницьких платформ.

Для кожного API запиту генерується відповідь у форматі JSON. Ці відповіді включають коди стану, які інформують користувача про результат його запиту, наприклад, повідомлення про успіх або невдачу, ідентифікатори завантажених зображень та будь-які запитувані метадані.

2.6. Опис розробленої системи

2.6.1. Використані технічні засоби

Для свого функціонування інфраструктура платформи використовує обладнання хмарного провайдеру AWS. Система розроблена таким чином, щоб бути доступною на більшості комп'ютерів. Такий підхід гарантує, що широке

коло користувачів може отримати доступ до функцій платформи і скористатися ними без потреби в спеціалізованому чи потужному обладнанні.

Мінімальними вимогами до користувальницьких систем є:

– наявність маніпулятору “миша”.
– наявність клавіатури.
– доступ до Інтернету.
– процесор Intel Core i3-3220 з тактовою частотою 3.3 ГГц.
– не менше ніж 4 Гб оперативної пам’яті.
– монітор з діагоналлю 13".

2.6.2. Використані програмні засоби

Для коректного функціонування програми необхідно, щоб програмне забезпечення користувальницької системи відповідало наступним вимогам:

– встановлений веб-браузер Firefox або Google Chrome.
– операційна система Windows 7/10/11.

2.6.3. Виклик та завантаження програми

Використання платформи вимагає мінімальних налаштувань або технічних знань. Щоб отримати доступ до платформи, користувачу достатньо відкрити веб-сторінку в браузері.

Наразі інформаційна система доступна виключно в режимі розробки, розгорнути його копію можливо виключно шляхом створення власного сховища Amazon S3 із функцією обслуговування веб-сторінок і завантаження файлів веб-сайту у нього. Частину інфраструктури, що відповідає за бізнес-логіку можна створити за допомогою Terraform-коду, що описує усі необхідні компоненти системи.

2.6.4. Опис інтерфейсу користувача

Щоб отримати доступ до інформаційної системи, користувачу потрібно відкрити відповідний веб-сайт у своєму браузері, де він побачить стартову сторінку (рис.2.8). Ця сторінка є простою і орієнтованою на залучення користувачів, зокрема, завдяки заклику до дії "перетягування" зображення. Цей інтуїтивно зрозумілий дизайн заохочує користувачів одразу почати взаємодіяти з платформою, перетягнувши зображення на будь-яку частину сторінки веб-сайту.

Рис. 2.8. Стартова сторінка веб-сайту

Після перетягування зображення на стартову сторінку процес завантаження розпочинається автоматично. Платформа швидко обробляє завантажене зображення, надаючи користувачам доступ до ряду інструментів для редагування та обміну зображеннями.

Після переходу на сторінку редагування зображення (рис.2.9) користувачу відкривається завантажене зображення, яке відображається в лівій частині екрана. Таке налаштування гарантує, що користувач може швидко і легко розпочати роботу зі своїм зображенням, розуміючи із чим він працює.

Рис. 2.9. Сторінка редагування зображення

Під зображенням користувачі можуть знайти пряме посилання на нього. Ця функція корисна для розробників і творців контенту, яким може знадобитися вбудувати зображення безпосередньо на свої веб-сайти. Також під зображенням відображається частина метаданих ідентифікації об'єктів та оточення зображення.

На сторінці також відображаються позначки, які вказують на те, що зображення поширюється через мережу доставки контенту, що забезпечує оптимальний час завантаження та продуктивність для користувачів по всьому світу. Ще одна позначка підтверджує, що зображення не містить чутливого контенту, запевняючи користувачів у його придатності для широкої аудиторії.

У правій частині сторінки доступні різні варіанти редагування зображення. Користувачі можуть вибрати такі дії, як перетворення формату зображення між JPEG, PNG і WEBP, стиснення зображення для зменшення розміру файлу, трансформації зображення або застосування чорно-білого фільтра. Усі опції зроблено інтуїтивно зрозумілими, із відповідними кнопками, які користувачі можуть натискати, щоб застосувати бажані зміни.

Після застосування будь-яких редагувань користувачі можуть скопіювати посилання на змінену версію зображення. Ця функція зручна для користувачів, яким потрібно одразу використати або поділитися відредагованою версією. По відкриттю сторінки зміненого зображення (рис.2.10) можна побачити його відредаговану версію.

Рис. 2.10. Сторінка відредагованого зображення

. Інтерфейс інтуїтивно зрозумілий і простий, що дозволяє користувачам починати взаємодіяти зі своїми зображеннями одразу після завантаження сторінки. Така доступність має вирішальне значення для підтримки ефективного користувацького досвіду.