Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти другий (магістерський)

Проєктування та дослідження IT-інфраструктури на основі AWS Serverless для зберігання та обробки персональних даних студентів
(тема)

Виконав:

студент II курсу, групи СПм-22-5

Сафанков Д.В.
(прізвище, ініціали)

Спеціальність

123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма

Системне програмування
(повна назва освітньої програми)

Керівник: доц. Шматко О.В.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри ЕОМ _____ Коваленко А.А.
(підпис) (прізвище, ініціали)

2024 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти другий (магістерський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)

Освітня програма Системне програмування
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20___ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

студенту Сафанкову Данилу Валерійовичу
(прізвище, ім’я, по батькові)

1. Тема роботи Проєктування та дослідження IT-інфраструктури на основі AWS Serverless для зберігання та обробки персональних даних студентів

затверджена наказом по університету від “ 01 ” квітня 2024 р. № 257 Ст

2. Термін подання студентом роботи до екзаменаційної комісії 15 червня 2024 р.

3. Вхідні дані до роботи Текстові файли формату JSON

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблеми та огляд існуючих рішень;
2) вибір технології розробки та інструментальних засобів;
3) розробка алгоритмічного забезпечення;
4) розробка програмних модулів;
5) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (слайдів)

Слайд-презентація – 11 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	02.04.24-08.04.24	
2	Вибір технології розробки та	09.04.24-16.04.24	
	інструментальних засобів		
3	Розробка алгоритмічного забезпечення	17.04.24-22.04.24	
4	Розробка програмних модулів	23.04.24-06.05.24	
5	Відлагодження програмних модулів	07.05.24-23.05.24	
6	Оформлення матеріалів кваліфікаційної	24.05.24-03.06.24	
	роботи		
7	Подання кваліфікаційної роботи керівникові	04.06.24-07.06.24	
	та її попередній захист		
8	Подання кваліфікаційної роботи	08.06.24-12.06.24	
	на рецензування		

Дата видачі завдання 01 квітня 2024 р.

Студент _____
(підпис)

Керівник роботи _____ доц. Шматко О.В.
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 67 с., 19 рис., 1 дод., 18 джерел.

ХМАРА, ТЕКСТОВИЙ ФАЙЛ, АРХІТЕКТУРА, ВЕБ-СЕРВІС, РОЗРОБКА, ПРОГРАМА, ТЕРРАФОРМ, ПРОЕКТУВАННЯ, ВІДТВОРЕННЯ, БЕЗСЕРВЕРНІ ТЕХНОЛОГІЇ.

Метою кваліфікаційної роботи є проектування та дослідження IT-інфрастурктури на основі AWS Serverless для зберігання та обробки персональних даних студентів.

У ході виконання кваліфікаційної роботи було проаналізовано існуючі на даний час хмарні провайдери, їх переваги та недоліки. Написано терраформ файли, які дозволяють створити компоненти в AWS хмарі, а також Python код який є основою тестування проекту.

ABSTRACT

Master’s thesis: 67 pages, 19 figures, 1 appendices, 18 sources.

AWS CLOUD, AZURE, GCP, ORACLE, IBM, SAP, ALIBABA, VMWARE, SALEFORCE, DIGITALOCEAN, TERRAFORM, SERVERLESS, MEDIAFILE, DEVELOPMENT, PROGRAM, MEDIA.

The major goal of this thesis is designing and promoting IT infrastructure based on AWS Serverless for storing and modifying students data.

During the qualification work, the existing Cloud providers, and their advantages and disadvantages were analyzed. Terraform files have been designed that can create components in AWS cloud, also Python code was written to test the workload.

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ, СКОРОЧЕНЬ І ТЕРМІНІВ

AWS – амазон веб сервіси (англ., Amazon Web Services)

GCP –хмарна платформа гугл (англ., Google Cloud Platform)

API – програмний інтерфейс додатку (англ., Application Programming Interface)

SaaS – система як сервіс (англ., Software as a Service)

PaaS – платформа як серіс (англ., Platform as a Service)

IaaS – інфраструктура як сервіс (англ., Infrastructure as a Service)

ВСТУП

Хмарні обчислення є надзвичайно актуальною темою, оскільки вони користуються великим попитом у сучасному світі. Практично кожна людина сьогодні так чи інакше взаємодіє з хмарними технологіями.

Хмарні обчислення – це модель, яка забезпечує зручний і повсюдний мережевий доступ на вимогу до загального пулу конфігуруємих обчислювальних ресурсів (наприклад, серверів, мереж передачі даних, пристроїв зберігання даних, додатків і сервісів – як разом, так і окремо).

Важливо зазначити, що хмарні обчислення є ключовим напрямком у розвитку IT-технологій. Вони забезпечують ефективну підтримку обчислювальної інфраструктури для багатьох користувачів. Крім того, для державних установ та корпоративних клієнтів хмарні технології надають можливість керування даними без потреби у повному адмініструванні апаратного та програмного забезпечення.

Керованість є особливо важливою у хмарних середовищах. Слід підкреслити, що в порівнянні з традиційними системами, досягнення високого рівня керованості у хмарних середовищах ускладнюється через три основні фактори: обмежене людське втручання, значний діапазон робочих навантажень та різноманітність спільно використовуваних інфраструктур. У більшості випадків відсутні адміністратори баз даних або систем, які могли б допомагати розробникам у створенні додатків на основі хмарних сервісів; адміністрування платформ переважно здійснюється автоматично.

Хмарний постачальник (cloud provider) – це компанія, яка надає послуги та рішення на основі хмарних обчислень, використовуючи віртуальні обчислювальні потужності: сервери, мережі, системи зберігання даних тощо. Хмарний постачальник пропонує клієнтам інструменти у вигляді сервісу самообслуговування, що дозволяють створювати, коригувати, видаляти

віртуальні машини, а також налаштовувати надані обчислювальні потужності: обсяг сховища, мережеві ресурси, розмір пам'яті та інше.

Актуальність теми обумовлена швидким розвитком хмарних технологій і їх впливом на сучасне інформаційне середовище. Хмарні обчислення дозволяють значно знизити витрати на інформаційну інфраструктуру, підвищують гнучкість і масштабованість IT-ресурсів, що є важливим для бізнесу та державних установ.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність розробки

На сьогодні кількість користувачів, які користуються хмарними технологіями, постійно зростає. Завдяки спеціалізованим платформам користувачі мають доступ до великої кількості обчислювальних ресурсів.

Розробники таких хмарних сервісів прагнуть зробити їх максимально зручними для всіх користувачів. Кожна з існуючих хмарних платформ має свої унікальні особливості та функції, які задовольняють потреби різних користувачів.

Хмарні технології надають користувачам безліч можливостей для значного підвищення швидкості та ефективності процесів розробки програмного забезпечення. Зокрема, вони дозволяють:

- скоротити час розробки та випуску нових версій. Прискорити процеси Dev&Test завдяки інтеграції популярних інструментів розробки та найпоширеніших мов програмування (JavaScript, Java, Python, PHP, Go) з хмарним середовищем;
- створити динамічне середовище для розробки у вигляді ізольованих "пісочниць" з повним контролем ресурсів. Це включає окремі політики контролю доступу для різних команд, встановлення лімітів на використання ресурсів та видів сервісів, а також інструменти комплексного моніторингу та управління сервісами хмари та процесами розробки;
- використовувати Dev&Test інфраструктуру з оплатою за фактичне споживання ресурсів, включаючи Spot-інстанси. Застосовувати підхід Infrastructure as Code для створення інфраструктури в хмарі, автоматизувати процеси розгортання та контролю витрат на рівні кожного сервісу.

Тому розробка та впровадження компонентів у хмарному середовищі є вкрай актуальним завданням. Крім того, хмарні технології сприяють

підвищенню безпеки даних, забезпечуючи резервне копіювання та відновлення, що є критично важливим для сучасних бізнесів. Інтеграція з хмарними сервісами також дозволяє компаніям швидше адаптуватися до змін ринкових умов, збільшуючи свою гнучкість та конкурентоспроможність.

1.2 Аналіз існуючих рішень

1.2.1 Хмара AWS

Amazon Web Services (AWS) – це найпоширеніша у світі хмарна платформа з найширшими можливостями, яка пропонує понад 200 повнофункціональних сервісів для центрів обробки даних по всьому світу. Мільйони клієнтів, включаючи стартапи, які швидко зростають, великі корпорації та провідні державні установи, використовують AWS для зниження витрат, підвищення гнучкості та прискорення впровадження інновацій.

AWS забезпечує більшу кількість сервісів та їх функціональних можливостей, ніж будь-який інший постачальник хмарних послуг. Це включає інфраструктурні технології, такі як інструменти для обчислень, сховища даних та бази даних, а також інновації у сферах машинного навчання та штучного інтелекту, озер даних та аналітики, а також Інтернету речей. Завдяки цим сервісам клієнти можуть швидше, легше та дешевше переносити поточні додатки у хмару та реалізовувати будь-які можливі проекти.

Однією з ключових переваг AWS є її глобальна інфраструктура, яка дозволяє користувачам розгортати свої додатки в різних регіонах, забезпечуючи високу доступність та стійкість до збоїв. Клієнти можуть вибирати з 33 географічних регіонів і 105 зон доступності, що дозволяє легко масштабувати ресурси відповідно до потреб бізнесу.

Крім того, AWS активно підтримує екосистему партнерів, надаючи можливості для інтеграції з різноманітними сторонніми інструментами та сервісами. Це дає змогу користувачам розширювати функціональність своїх

додатків та використовувати найновіші технологічні досягнення. AWS також пропонує широкий спектр навчальних ресурсів та сертифікаційних програм, що допомагають IT-фахівцям підвищувати свою кваліфікацію та залишатися в курсі останніх тенденцій у сфері хмарних обчислень.

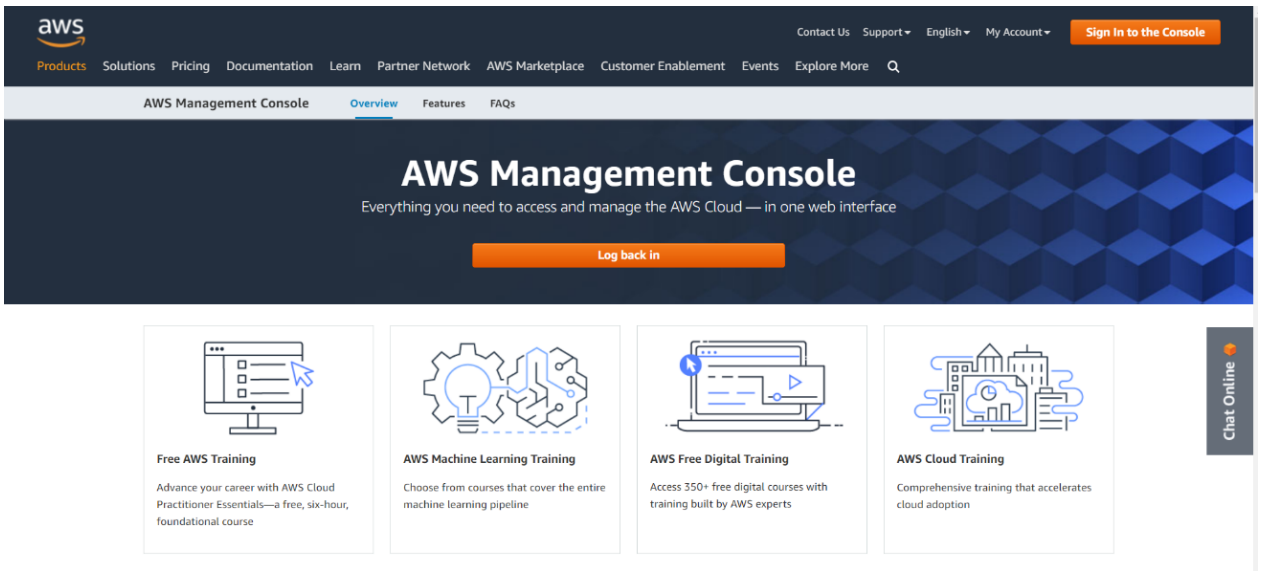

Рисунок 1.1 – Консоль управління AWS

Переваги AWS хмари:

- простота використання;
- гнучкість;
- надійність;
- масштабованість та висока потужність;
- безсерверні обчислення (serverless computing);
- saas;
- paas;
- iaas.

1.2.2 Хмара Azure

Azure – це хмарна платформа з постійно зростаючим набором служб, яка допомагає вирішувати різноманітні бізнес-завдання. Служби Azure можуть варіюватися від простих веб-сервісів для розміщення вашого бізнесу в хмарі до повністю віртуалізованих комп'ютерів для виконання складних програмних рішень. Azure надає широкий спектр хмарних послуг, таких як віддалене сховище даних, хостинг баз даних та централізоване управління обліковими записами користувачів. Крім того, Azure пропонує інноваційні можливості, включаючи штучний інтелект (AI) та Інтернет речей (IoT).

Однією з головних переваг Azure є її інтеграція з іншими продуктами та сервісами Microsoft, такими як Office 365 та Dynamics 365, що дозволяє компаніям створювати злагоджену екосистему для своїх бізнес-процесів. Ця інтеграція спрощує впровадження та управління різними бізнес-застосунками, забезпечуючи безперервність робочих процесів і покращуючи продуктивність.

Крім того, Azure надає розширені можливості для забезпечення безпеки та відповідності нормативним вимогам. Платформа включає в себе широкий набір інструментів для моніторингу, керування і захисту даних, що дозволяє організаціям дотримуватися вимог безпеки та стандартів конфіденційності. Azure також підтримує гібридні хмарні рішення, що дозволяє компаніям зберігати критично важливі дані на власних серверах, одночасно використовуючи переваги хмарних сервісів.

Azure активно підтримує розвиток та впровадження сучасних технологій через свої сервіси для розробників. Платформа надає інструменти для безперервної інтеграції та доставки (CI/CD), дозволяючи автоматизувати процеси розгортання додатків і оновлень. Azure DevOps, GitHub та інші сервіси допомагають розробникам створювати, тестувати та впроваджувати програмне забезпечення з високою швидкістю та надійністю. Завдяки цьому, компанії можуть швидше реагувати на зміни ринкових умов і впроваджувати нові функції, забезпечуючи своїм клієнтам найкращий досвід.

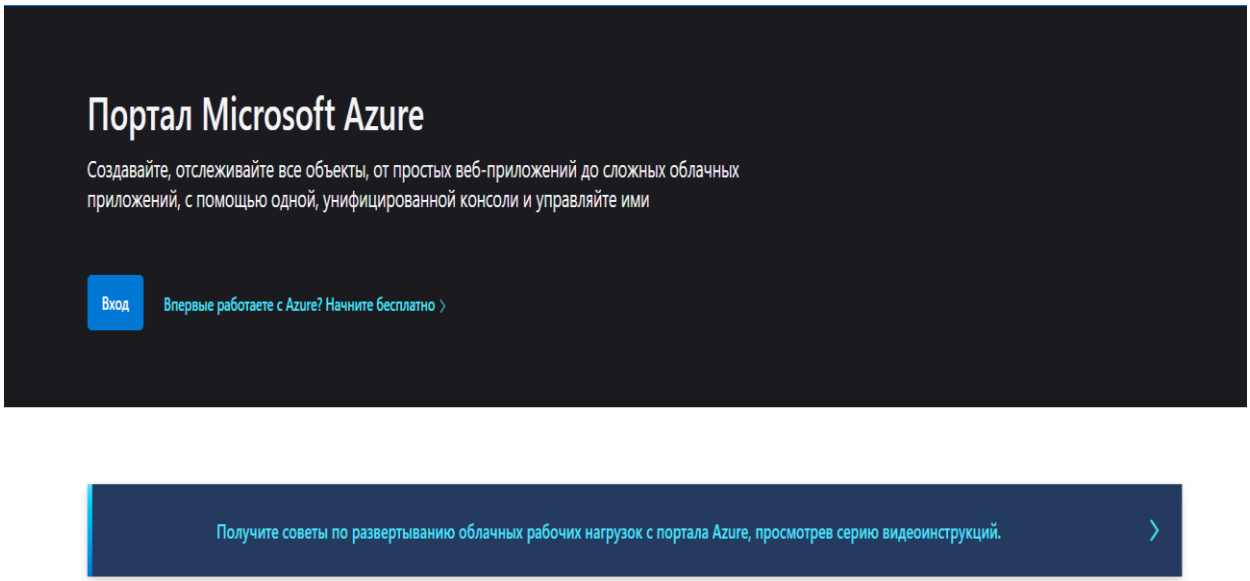

Рисунок 2.2 – Консоль управління Azure

Переваги Azure хмари:

- надійна хмара;
- проста робота з гібридною хмарою;
- надійність;
- висока масштабованість;
- інтеграція з іншими продуктами Microsoft;
- потужні інструменти для аналітики та обробки великих даних;
- вбудовані функції безпеки та відповідності нормативним вимогам;
- підтримка різноманітних мов програмування та фреймворків;
- можливість використання штучного інтелекту (AI) та машинного навчання (ML);
- широка мережа центрів обробки даних по всьому світу;
- гнучка система ціноутворення з оплатою за використані ресурси.

1.2.3 Хмара GCP

Google Cloud Platform (GCP) – це набір сервісів хмарних обчислень, який пропонує широкий спектр модульних хмарних рішень, включаючи обчислювальні потужності, розробку додатків, машинне навчання, аналіз даних та зберігання даних. Розробники, адміністратори хмарних сервісів та інші IT-спеціалісти можуть отримати доступ до GCP через загальнодоступні або виділені мережі.

Однією з ключових переваг GCP є його потужні можливості для аналізу та обробки даних. Використовуючи сервіси, такі як BigQuery та Dataflow, компанії можуть обробляти та аналізувати великі обсяги даних у реальному часі, що допомагає отримувати цінну аналітичну інформацію для прийняття обґрунтованих бізнес-рішень. Крім того, GCP активно підтримує використання штучного інтелекту та машинного навчання, надаючи інструменти, такі як TensorFlow та AutoML, які дозволяють розробникам створювати та впроваджувати інтелектуальні рішення.

GCP також відзначається своєю масштабованістю та гнучкістю. Завдяки глобальній інфраструктурі Google, користувачі можуть легко масштабувати свої ресурси відповідно до потреб бізнесу, забезпечуючи високу доступність та стійкість до збоїв. Крім того, GCP пропонує широкий спектр інструментів для розробників, таких як Cloud Functions та Kubernetes Engine, що спрощують процес створення, розгортання та управління додатками. Ще однією важливою перевагою GCP є високий рівень безпеки та відповідності нормативним вимогам.

Платформа забезпечує багаторівневий захист даних, використовуючи передові методи шифрування та засоби управління доступом. Це дозволяє організаціям забезпечувати безпеку своїх даних та дотримуватися міжнародних стандартів і регуляторних вимог. GCP також підтримує гібридні та мультихмарні стратегії, що дозволяє компаніям інтегрувати свої локальні

ресурси з хмарними, забезпечуючи безшовний перехід та максимальну ефективність.

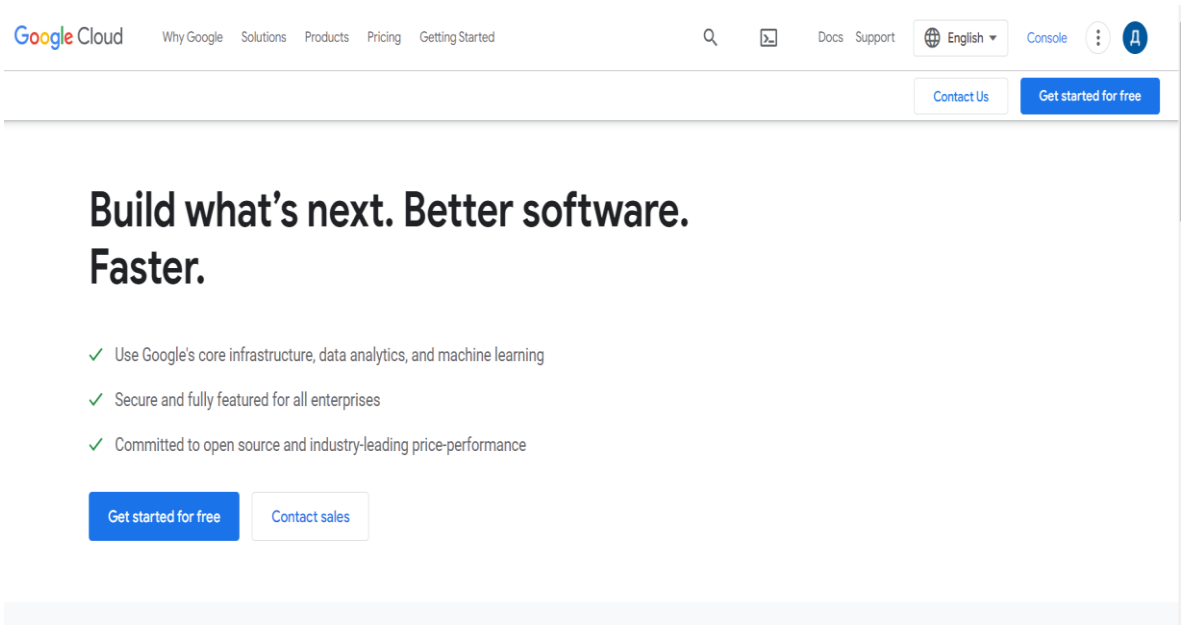

Рисунок 2.3 – Консоль управління GCP

Переваги GCP хмари:

- масштабованість;
- швидка спільна робота;
- безпека інфраструктури;
- висока продуктивність обчислень Інтеграція зі штучним інтелектом та машинним навчанням;
- потужні інструменти для аналізу та обробки даних;
- глобальна мережа центрів обробки даних;
- гнучкість у виборі інструментів та мов програмування;
- підтримка гібридних та мультихмарних рішень;
- надійне резервне копіювання та відновлення даних;
- прозоре ціноутворення та оплата за використані ресурси.

1.2.4 Хмара Alibaba

Хмара Alibaba Cloud, заснована у 2009 році, є світовим лідером у сфері хмарних обчислень та штучного інтелекту, надаючи послуги тисячам підприємств, розробників та урядових організацій у більш ніж 200 країнах та регіонах. На базі цієї хмари реалізовані послуги AliExpress, Taobao, міжнародна платіжна система AliPay.

Однією з головних переваг Alibaba Cloud є її здатність забезпечувати високий рівень безпеки та надійності. Хмара пропонує широкий спектр рішень для захисту даних, включаючи передове шифрування, багаторівневі засоби аутентифікації та управління доступом. Ці заходи гарантують, що конфіденційна інформація залишається захищеною від несанкціонованого доступу та кіберзагроз. Крім того, Alibaba Cloud активно працює над забезпеченням відповідності міжнародним стандартам безпеки та регуляторним вимогам, що є критично важливим для багатьох підприємств і урядових організацій.

Іншою важливою перевагою Alibaba Cloud є її масштабованість та гнучкість. Платформа дозволяє підприємствам легко збільшувати або зменшувати обчислювальні ресурси відповідно до поточних потреб, що допомагає оптимізувати витрати та підвищити ефективність. Це особливо корисно для бізнесів, які мають сезонні або коливальні навантаження, оскільки вони можуть швидко адаптуватися до змін ринкових умов без значних додаткових витрат.

Крім того, Alibaba Cloud надає потужні інструменти для розробників, які допомагають створювати, тестувати та розгортати додатки з високою швидкістю та надійністю. Платформа підтримує широкий спектр мов програмування та фреймворків, що дозволяє розробникам використовувати ті інструменти, які найкраще відповідають їхнім потребам. Також Alibaba Cloud пропонує інноваційні рішення в галузі штучного інтелекту та машинного навчання, що допомагає підприємствам автоматизувати рутинні завдання,

аналізувати великі обсяги даних і приймати більш обґрунтовані бізнес-рішення.

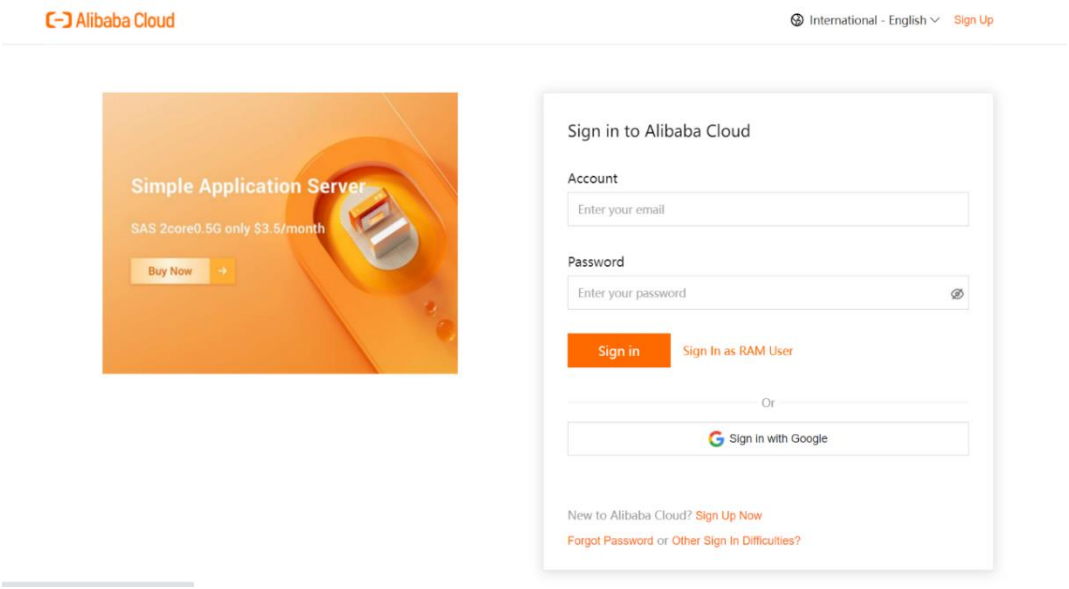

Рисунок 2.4 – Консоль управління Alibaba

Переваги Alibaba хмари:

- надійна присутність в Азії з прицілом на міжнародний ринок;
- шлюз доступу в Китай (China Gateway);
- комплексні рішення в галузі безпеки та відповідність вимогам;
- широкий вибір сервісів для розробників та підприємств;
- гнучкі ціноутворення та оплата за фактом використання;
- продуктивність та ефективність роботи завдяки високопродуктивним серверам та мережевим технологіям;
- інноваційні рішення в області штучного інтелекту та аналізу даних;
- підтримка гібридних та мультихмарних стратегій розгортання;
- технічна підтримка та консультування від експертів з області хмарних обчислень;
- великий масштаб можливостей для масштабування бізнесу будь-якого розміру;
- висока доступність та надійність інфраструктури для бізнес-процесів.

1.2.5 Хмара IBM

IBM Cloud – це комплекс технологій та послуг у сфері хмарних обчислень, який включає в себе хостинг на основі технологій SoftLayer, що займають провідні позиції у світі. Ця інфраструктура надає потужності для розміщення веб-сайтів та інших сервісів для широкого кола користувачів. Компанії з великими обсягами даних добре знають, що завантажені середовища можуть впливати на продуктивність.

Проте, завдяки використанню окремих серверів SoftLayer, підприємства можуть запускати важкі додатки без обмежень, які можуть виникнути в умовах загального використання ресурсів. Це дозволяє максимально ефективно використовувати потужності хмарної інфраструктури та уникати проблем, пов'язаних зі сплесками навантаження.

Зокрема, IBM Cloud надає різноманітні інструменти для моніторингу та оптимізації ресурсів, що дозволяє підтримувати стабільну продуктивність додатків навіть у найінтенсивніших умовах використання.

Крім того, IBM Cloud активно працює над розробкою інноваційних рішень у галузі штучного інтелекту та аналізу даних, що допомагає підприємствам автоматизувати процеси та приймати обґрунтовані рішення на основі даних. Пропонує не лише потужність та надійність для розміщення великих обсягів даних, але й інструменти для оптимізації ресурсів та підтримки стабільної продуктивності навіть у найінтенсивніших умовах використання. Їхні технології дозволяють підприємствам масштабувати свої додатки та ефективно використовувати ресурси, забезпечуючи високу доступність та продуктивність. Крім цього, постійні інновації у галузі штучного інтелекту та аналізу даних роблять IBM Cloud привабливим вибором для підприємств, які бажають автоматизувати свої процеси та робити обґрунтовані рішення на основі даних.

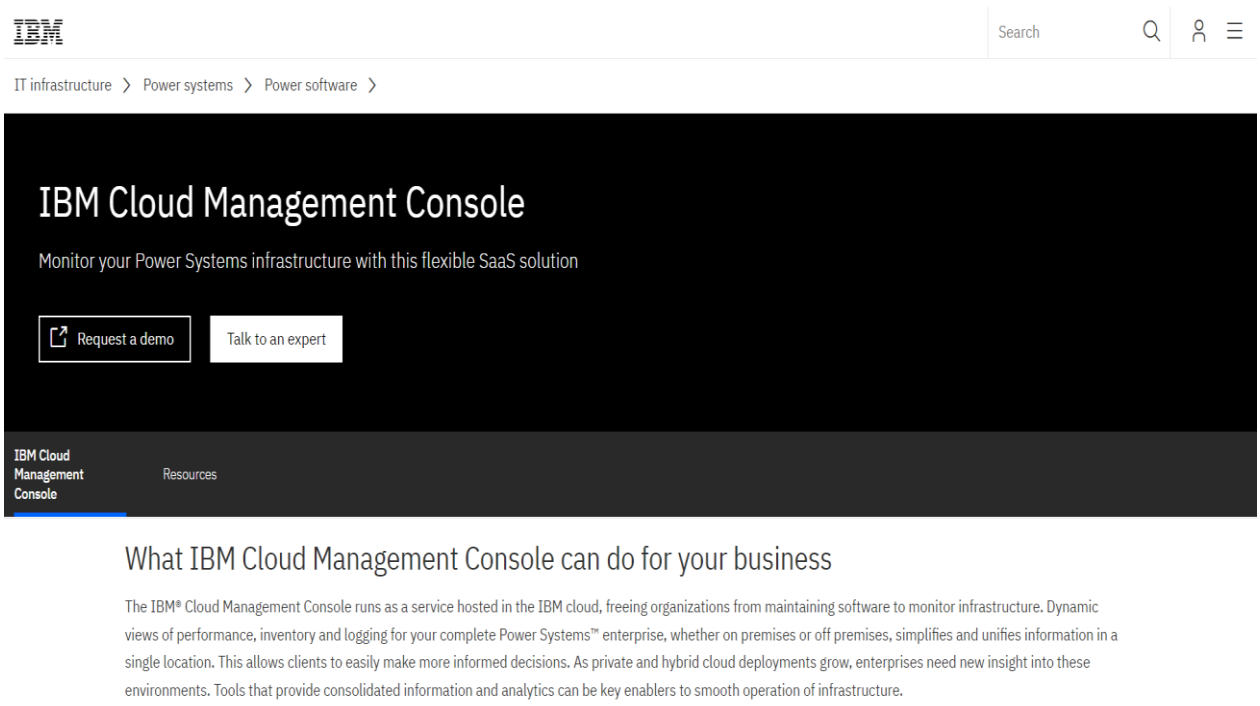

Рисунок 2.5 – Консоль управління IBM

Переваги IBM хмари:

- контроль витрат;
- розгортання програмного забезпечення як послуги звільняє персонал від встановлення та обслуговування програмного забезпечення;
- захищені хмарні послуги, доступні з мобільних пристроїв, планшетів і ПК.

1.2.6 Хмара Oracle

Oracle Cloud – це загальнодоступне облако нового покоління, яке допоможе вам швидко перетворити свій бізнес і модернізувати IT за рахунок інтегрованих, послуг на всіх рівнях технологічного стека. Ви отримуєте взаємозалежні хмарні служби, якими легко користуватися та управляти, маневренність, надійність, масштабованість та безпеку, необхідні сучасному підприємству. Інфраструктура Oracle Cloud дозволяє створювати та запускати

додатки та сервіси в середовищі Oracle. Новим користувачам може бути складно орієнтуватися на цій платформі, і потрібен час, щоб звиктуни до неї.

Oracle Cloud є привабливим вибором для підприємств, оскільки надає взаємозалежні хмарні послуги на всіх рівнях технологічного стека. Вона дозволяє швидко перетворити бізнес та модернізувати IT завдяки інтегрованим рішенням, що забезпечують маневренність, надійність, масштабованість та безпеку, необхідні для сучасних підприємств.

Інфраструктура Oracle Cloud створена для того, щоб підтримувати створення та запуск додатків та сервісів в середовищі Oracle, що дозволяє користувачам ефективно використовувати її потужності. Це стає особливо важливим у сучасному цифровому середовищі, де швидкість та ефективність мають вирішальне значення для конкурентоспроможності.

Незважаючи на великий потенціал та можливості Oracle Cloud, новим користувачам може знадобитися трохи часу для того, щоб орієнтуватися на платформі та вивчити всі її можливості. Проте, з перебігом часу та налагодженням процесів використання, вони зможуть повністю використовувати переваги цього облікового середовища для свого бізнесу.

Бізнеси, що вибирають Oracle Cloud, можуть бути впевнені у тому, що вони отримують не лише доступ до передових технологій, а й підтримку та експертні знання від команди професіоналів Oracle. Це дозволяє підприємствам швидко адаптуватися до змінних потреб ринку та впроваджувати інновації з впевненістю у їхній успішності. В результаті, Oracle Cloud стає не лише платформою для розвитку та розширення бізнесу, а й стратегічним партнером у досягненні цілей та успіху компанії.

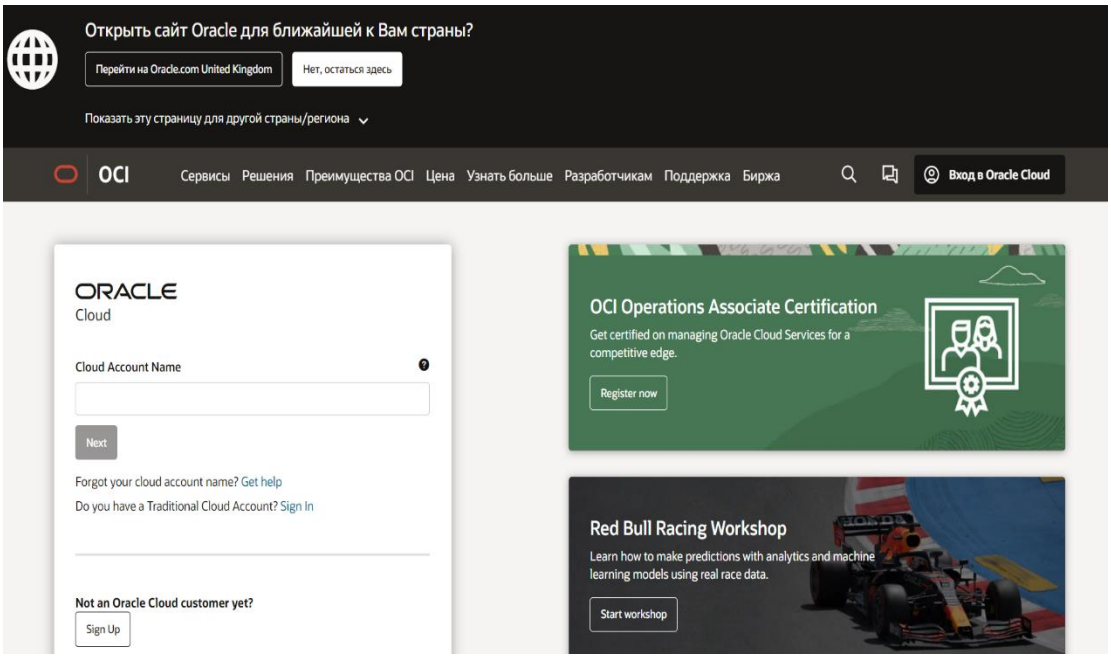

Рисунок 2.6 – Консоль управління Oracle

Переваги Oracle хмари:

- гнучкість;
- масштабованість;
- операційні витрати замість капітальних витрат;
- швидке впровадження інновацій;
- зниження витрат на розробку та тестування;
- оплата лише за використаний час.

1.2.7 Хмара Salesforce

Salesforce Cloud – це інноваційна онлайн-платформа, яка революціонізує спосіб управління бізнесом. Заснована у 1999 році, Salesforce швидко стала однією з провідних компаній у галузі відносин з клієнтами та хмарних технологій. Її продукти та послуги відомі своєю ефективністю, надійністю та інноваційністю, і вони використовуються компаніями будь-якого масштабу, від стартапів до великих корпорацій.

Однією з ключових переваг Salesforce Cloud є його всебічність. Платформа надає широкий спектр інструментів та сервісів для управління бізнес-процесами, від відносин з клієнтами до аналітики, маркетингу та продажів. Це означає, що компанії можуть консолідувати свою діяльність на одній платформі, спрощуючи процеси та забезпечуючи їхню злагодженість.

Крім того, Salesforce Cloud відомий своєю гнучкістю та розширюваністю. Він пропонує різні рішення та конфігураційні опції, які можна адаптувати до потреб конкретного бізнесу. Незалежно від того, чи потрібно підприємству простий інструмент для відстеження продажів, або комплексна система для управління всією діяльністю, Salesforce має рішення, яке відповідає вимогам та бюджету.

Завдяки постійним інноваціям та оновленням, Salesforce Cloud залишається лідером у світі бізнес-технологій. Він постійно розширюється та вдосконалюється, щоб забезпечувати своїм користувачам найсучасніші інструменти та можливості для досягнення успіху у сучасному бізнес-середовищі.

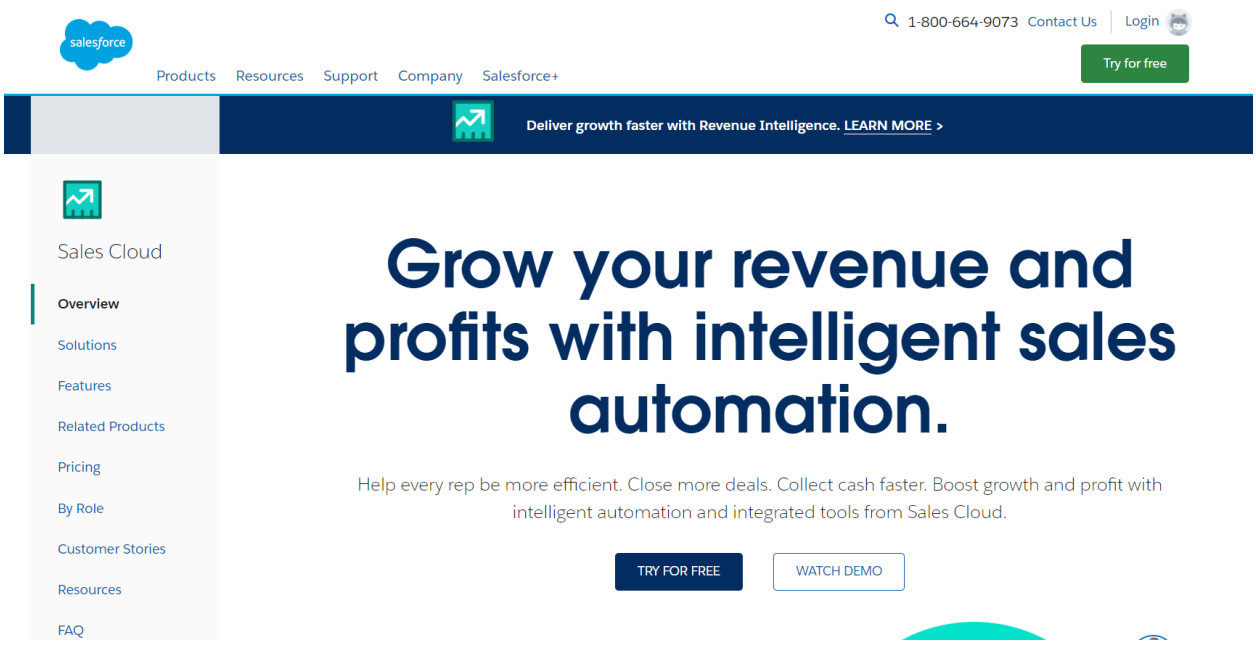

Рисунок 2.7 – Консоль управління Salesforce

Переваги Salesforce хмари:

- управління обліковими записами та контактами;
- звіти та інформаційні панелі;
- слідкування за станом бізнесу в режимі реального часу;
- прогноз продажів.

1.2.8 Хмара SAP

SAP Cloud – це інноваційна хмарна платформа, що пропонується за моделлю платформи як послуга (PaaS). Вона надає широкі можливості для обчислень в оперативній пам'яті, базової платформи та унікальних мікросервісів, спрямованих на створення та розширення інтелектуальних хмарних рішень з підтримкою мобільних пристроїв. Орієнтована на прискорення цифрової трансформації, ця платформа дозволяє легко, швидко та економічно розробляти потрібні програмні рішення без необхідності інвестування у власну локальну інфраструктуру.

Однією з ключових переваг SAP Cloud є її гнучкість та масштабованість. Платформа може адаптуватися до різноманітних потреб підприємств будь-якого розміру та галузі, надаючи інструменти та сервіси, необхідні для успішного функціонування та розвитку бізнесу.

Крім того, SAP Cloud постійно оновлюється та вдосконалюється, враховуючи нові технологічні та бізнесові вимоги. Вона надає користувачам доступ до останніх інновацій та технологій, що дозволяє їм залишатися попереду у конкурентному середовищі.

Платформа SAP Cloud відома також своєю високою рівню безпеки та захищеності даних. Вона використовує передові методи шифрування та механізми захисту, щоб забезпечити конфіденційність та цілісність даних користувачів. Завдяки цьому підприємства можуть бути впевнені у захищеності своєї інформації та даних клієнтів.

Крім того, SAP Cloud пропонує високий рівень підтримки та обслуговування користувачів. Команда експертів здійснює постійний моніторинг та надає оперативну допомогу у вирішенні будь-яких технічних питань чи проблем, що виникають під час використання платформи. Такий підхід дозволяє користувачам максимально ефективно використовувати всі можливості та переваги SAP Cloud для свого бізнесу.

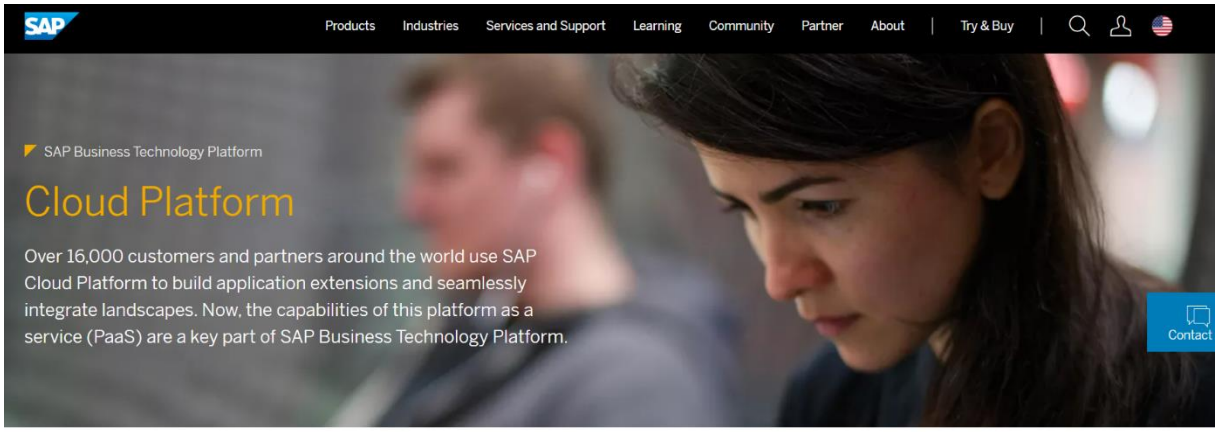

Рисунок 2.8 – Консоль управління SAP

Переваги SAP хмари:

- швидке розгортання попередньо налаштованих програм і служб;
- проста розробка бізнес-додатків;
- цілісне управління життєвим циклом усіх інтерфейсів.

1.2.9 Хмара Rackspace

Rackspace Cloud – це комплексне рішення у сфері хмарних обчислень, яке надає широкий спектр послуг для створення, розгортання та управління хмарними інфраструктурами підприємств. Крім цього, вона забезпечує доступ

до різноманітних інструментів, включаючи необмежену базу даних, хостинг веб-сайтів, поштові рішення та можливості торговельного обліку для виставлення рахунків та підтримки клієнтів.

Однією з ключових переваг хмари Rackspace є її велика гнучкість та адаптивність до потреб різних типів організацій. Вона пропонує індивідуальний підхід до кожного клієнта, надаючи можливість вибору необхідних сервісів та конфігурацій для оптимального задоволення їхніх потреб.

Крім базових сервісів хмарних обчислень, Rackspace Cloud також надає додаткові послуги для забезпечення безпеки хмарного середовища, розробки API, створення та управління базами даних та іншими компонентами. Це дозволяє підприємствам отримати повний спектр необхідних інструментів та ресурсів для ефективного функціонування їхніх хмарних інфраструктур.

Окрім того, Rackspace відома своєю високоякісною технічною підтримкою та обслуговуванням клієнтів. Команда експертів завжди готова надати консультації та допомогу щодо вирішення будь-яких технічних питань та проблем, що виникають у процесі використання хмарної інфраструктури.

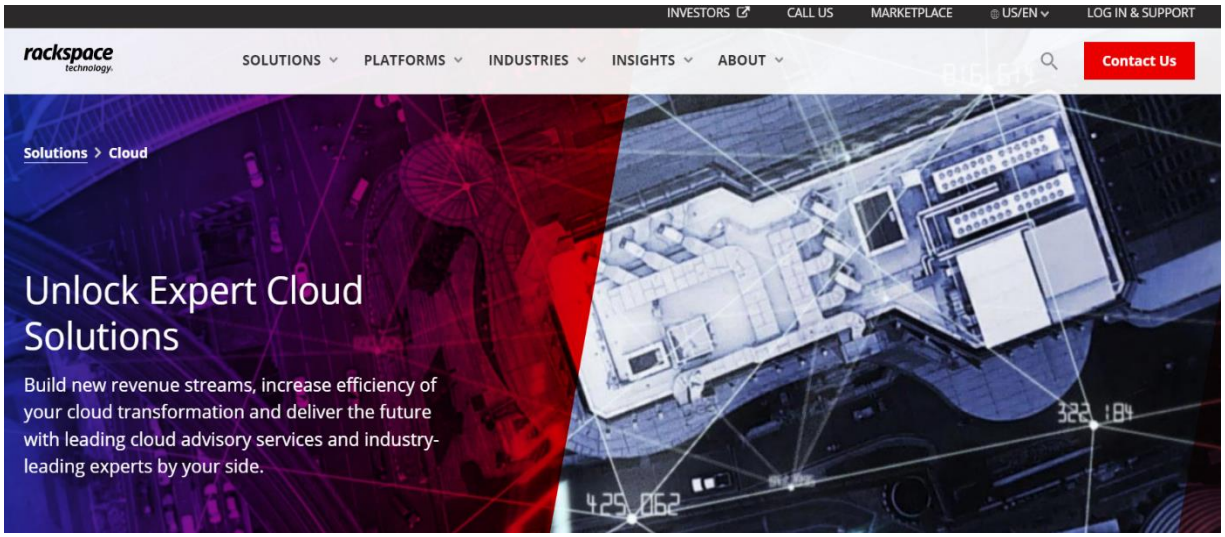

Рисунок 2.9 – Консоль управління Rackspace

Переваги SAP хмари:

- підтримка приватної хмари;
- управління веб-контентом;
- продуктивність і співпраця.

1.2.10 Хмара VMware

VMware Cloud – відома платформа, яка стоїть на чолі у сфері надання хмарних послуг. Вона надає безпечні, ефективні та гнучкі ресурси для хмарних обчислень тисячам компаній та відділів IT по всьому світу. VMware Cloud Director допомагає організувати та керувати хмарними сервісами компаній, забезпечуючи їм необхідну підтримку та контроль. Благодаря рішенням VMware, ви можете зберігати ваші існуючі пакети операційних систем та додатків, а також розгортати їх на внутрішніх або зовнішніх ресурсах будь-якого постачальника послуг vCloud.

Однією з важливих переваг VMware є гнучкість рішень, яка дозволяє зберегти функціональність існуючих систем, адаптуючи їх до нових умов. Крім того, ви отримуєте можливість використовувати рішення VMware для підтримки традиційних систем, звільнившись від багатьох пов'язаних з ними незручностей, таких як перенесення системи або встановлення виправлень безпеки.

Завдяки рішенням VMware, ви отримуєте більш передбачуваний рівень продуктивності та надійності, що є важливими факторами для успішної роботи вашої компанії в хмарному середовищі. Ви можете ефективно керувати ресурсами, забезпечуючи максимальну продуктивність та зменшуючи можливість виникнення непередбачених проблем.

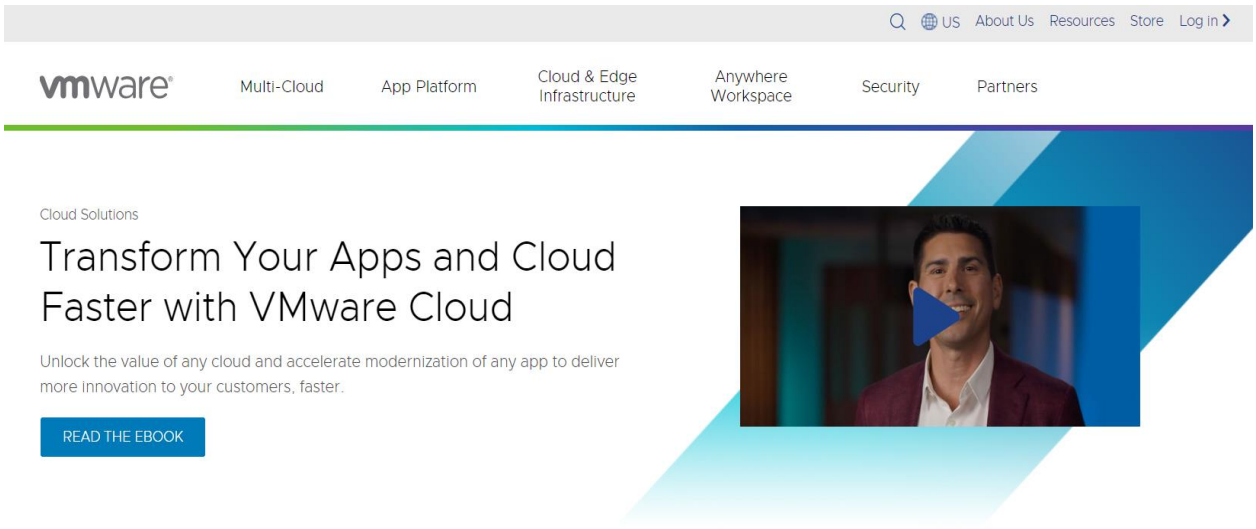

Рисунок 2.10 – Консоль управління VMware

Переваги VMware хмари:

- безпека та конфіденційність даних;
- управління веб-контентом;
- інтеграція з існуючими системами.

2 ПОСТАНОВКА ЗАДАЧІ

Метою цього проекту є розробка програмних компонентів на основі AWS Serverless. На підставі отриманої інформації будується сервіс, який дозволяє зберігати та обробляти персональні дані студентів.

Слід наголосити, що однією з особливостей є розгортання інфраструктури за допомогою Infrastructure as Code (IaC) методу, в основі якого open-source програмне забезпечення Terraform. Щоб протестувати роботу сервісу, треба встановити на ваш локальний пристрій Postman Api Platform.

Програма повинна містити компоненти AWS Serverless архітектури, а саме:

- AWS Api Gateway;
- AWS Lambda;
- AWS DynamoDB.

А також інші сервіси AWS провайдера:

- AWS IAM Role;
- AWS IAM Policy.

Вхідними даними для сервіса є JSON body яке відсилається на AWS Api Gateway ендпоінт за допомогою Postman.

3 ОПИС ТЕХНОЛОГІЙ ТА МЕТОДІВ, ЩО ВИКОРИСТОВУЮТЬСЯ

3.1 Операційна система Windows

Windows – це родина комерційних операційних систем, розроблених корпорацією Microsoft, призначених для зручного управління за допомогою графічного інтерфейсу. Початково Windows була лише графічною оболонкою для широко використовуваної операційної системи MS-DOS у 1980-х та 1990-х роках. За даними ресурсу Net Applications, на серпень 2014 року близько 88% ПК використовували операційні системи сімейства Windows. Вона підтримується на платформах x86, x86-64 та ARM. Найновішою версією на сьогоднішній день є Windows 11, яка була представлена у липні 2021 року.

На червень 2019 року операційна система Windows встановлена не менше, ніж на 88,5% ПК та робочих станцій. За даними компанії Net Applications, на той момент ринкова частка Windows склала 88,33%. Зниження цієї частки перш за все пов'язане з тенденцією зменшення продажів ПК у світі, а також зі збільшенням частки операційних систем конкурентів - macOS та Linux. Найбільш популярною версією Windows за даними W3Schools з липня 2017 року є Windows 10 (приблизно 37%).

Windows є однією з найбільш впливових операційних систем у світі, забезпечуючи широкий спектр функцій та можливостей для користувачів у всіх сферах життя. Вона використовується як в домашніх умовах, так і в бізнес-середовищі, де надійність, безпека та продуктивність мають критичне значення. Windows надає зручний графічний інтерфейс, що дозволяє навіть неспеціалістам легко взаємодіяти з комп'ютером.

Однією з ключових переваг Windows є її широкий екосистема програмного забезпечення. Через велику кількість розробників та підтримку від корпорації Microsoft, на цій операційній системі доступний величезний

асортимент програм та додатків для всіх потреб, від розваг до продуктивності та бізнесу. Це робить Windows привабливою для широкого кола користувачів.

Незважаючи на конкуренцію зі сторони інших операційних систем, таких як macOS та Linux, Windows залишається лідером на ринку ПК. Компанія Microsoft продовжує активно розвивати та вдосконалювати свою операційну систему, випускаючи регулярні оновлення та нові версії, які відповідають потребам сучасного користувача.

3.2 Postman Api platform

Postman – це програмне забезпечення, призначене для взаємодії з API. Це відомий клієнт API, який спрощує процес розробки, тестування та документування API. Використовуючи Postman, користувачі можуть надсилати HTTP/s запити до різних сервісів і отримувати від них відповіді. Цей підхід дозволяє перевіряти роботу бекенд-систем та забезпечувати їх правильне функціонування. Postman доступний як веб-додаток за адресою https://www.postman.com/, а також є нативним настільним застосунком для операційних систем Mac (Intel і M1), Windows (32-бітні / 64-бітні) та Linux (64-бітні).

Postman – це не лише інструмент для тестування API, але й потужний засіб для автоматизації процесів. Він дозволяє створювати колекції запитів, зберігати та керувати ними, а також створювати автоматичні тести API. Благодаря своїм функціональним можливостям, Postman став необхідним інструментом для розробників програмного забезпечення та тестувальників.

3.3 Мова програмування Python

Python – це скриптова мова, якиа використовується для вирішення великого обсягу самих різнопланових проблем і задач. Python корисний у створенні комп’ютерних і мобільних додатків, його застосовують у роботі з

великим обсягом інформації, при розробці веб-сайтів та інших різноманітних проектів, які використовуються в машинному навчанні. Дану мову програмування використовують великі відомі корпорації, такі як Spotify та Amazon (наприклад, для аналізу створення даних та алгоритмізації), YouTube і навіть Walt Disney. Таким чином, Python знайшов своє місце в різних областях – з його допомогою можна вирішити безліч завдань різної складності.

Програми для iOS і Android створюються на сотнях різних мов, і Python серед них. Найчастіше, звичайно, це стосується їх серверної складової, тому що за інтерфейс відповідають найпоширеніші фреймворки або якісь вузькоспеціалізовані технології. Тим не менш, за бекенд (тобто внутрішню логіку програми) часто відповідає саме Python.

Однією з основних переваг Python є його висока переносимість. Це означає, що програми, написані на Python, можуть працювати на будь-якій операційній системі без змін. Це робить Python ідеальним вибором для розробки крос-платформенного програмного забезпечення, яке може працювати як на Windows, так і на macOS або Linux.

Python також відомий своєю великою активною спільнотою розробників. Ця спільнота підтримує безліч відкритих проектів та бібліотек, які допомагають вирішувати різноманітні завдання. Завдяки цьому, розробники можуть швидко знаходити рішення на будь-яку проблему та вдосконалювати свої навички.

Окрім того, Python постійно розвивається та оновлюється. Нові версії мови регулярно випускаються з новими функціями та покращеннями, що дозволяє розробникам залишатися в актуальному стані та використовувати останні досягнення у своїх проектах.

3.4 Програмне забезпечення Terraform

Terraform кодифікує хмарні API у декларативні файли конфігурації. Він управляє зовнішніми ресурсами (такими, як загальнодоступна хмарна

інфраструктура, інфраструктура приватної хмари, мережеві пристрої, програмне забезпечення як послуга і платформа як послуга) з провайдерами. HashiCorp підтримує загальний список офіційних провайдерів, а також може інтегруватися з провайдерами, розробленими спільнотою.

Програмне забезпечення Terraform вирізняється своєю здатністю до інфраструктурного програмування, що дозволяє створювати, змінювати та управляти інфраструктурою як код. Це означає, що всі параметри і компоненти інфраструктури описуються у вигляді конфігураційних файлів, що дозволяє забезпечити повторюваність та відтворюваність усієї інфраструктури. Будь-які зміни в інфраструктурі можна легко застосувати, а також відстежувати за допомогою систем контролю версій, що робить процес управління інфраструктурою більш прозорим та контрольованим.

Однією з ключових переваг Terraform є його велика екосистема модулів та плагінів. Terraform має широкий вибір офіційних провайдерів, які охоплюють практично всі відомі хмарні платформи та послуги. Крім того, відкритість та гнучкість платформи дозволяють спільноті розробників створювати власні модулі та плагіни для інтеграції з різноманітними сервісами та технологіями. Це робить Terraform універсальним інструментом для автоматизації будь-якого типу інфраструктури, незалежно від специфіки та потреб проекту.

3.5 Мова програмування HashiCorp Configuration Language (HCL)

Мова програмування HashiCorp Configuration Language (HCL), яка використовується в Terraform, визначається як низькорівневий синтаксис, що дозволяє описувати конфігураційні файли. Цей синтаксис також застосовується в інших продуктах компанії HashiCorp. Окрім цього, конструкції, що описуються у Terraform, можуть бути виражені у синтаксисі JSON. Хоча JSON легше генерувати та аналізувати програмним шляхом, його важче читати та редагувати людям. Таким чином, використання HCL сприяє

зручнішому та більш інтуїтивному розробленню та конфігуруванню інфраструктури. Варто зазначити, що HCL спрощує роботу з конфігураційними файлами, роблячи їх більш зрозумілими та легкими для використання.

3.6 Базові сервіси AWS, які використовуються в даній роботі

3.6.1 Identity and Access Management (IAM)

Система IAM забезпечує точний контроль над доступом до різних сервісів AWS, дозволяючи вам точно визначати, хто та за яких умов може отримати доступ до певних ресурсів. Завдяки цій системі, ви можете керувати дозволами для користувачів та систем, встановлюючи їм лише необхідні привілеї для виконання їхніх завдань. Політики IAM дозволяють здійснювати гнучкий та безпечний контроль над доступом, забезпечуючи захист ваших ресурсів в хмарному середовищі AWS.

3.7 Безсерверне обчислення

AWS надає інструменти для запуску коду, управління даними та інтеграції програм без необхідності управління серверами. Безсерверні технології, які надає AWS, автоматично масштабуються, забезпечують високу доступність і прозору систему оплати за фактичне використання, сприяючи гнучкості та ефективному управлінню витратами. Використання цих технологій дозволяє уникнути рутинних завдань з управління інфраструктурою, таких як надання ресурсів та вирішення проблем, дозволяючи розробникам зосередитися на написанні коду, який відповідає потребам клієнтів.

Центральним елементом безсерверних програм на AWS є AWS Lambda – обчислювальна платформа, яка працює на основі подій. Вона інтегрована з

більш ніж 200 сервісами AWS та додатками, які пропонуються за моделлю програмного забезпечення як послуга (SaaS). Це дає можливість автоматично викликати функції при виникненні певних подій, що дозволяє створювати реактивні, еластичні та швидкодіючі додатки.

Крім AWS Lambda, AWS пропонує інші безсерверні сервіси, такі як Amazon API Gateway для створення та управління API, Amazon DynamoDB для керування базами даних, та Amazon S3 для зберігання об'єктів. Ці сервіси дозволяють розробникам будувати розподілені та масштабовані додатки, обмежуючи необхідність управління серверами та інфраструктурою. Вони надаються у вигляді повноцінних послуг, які можна швидко впровадити та інтегрувати у ваші додатки.

3.7.1 Api Gateway

Amazon API Gateway – це повністю керований сервіс, створений для розробників, який спрощує процеси створення, публікації, обслуговування, моніторингу та забезпечення безпеки API будь-якого розміру. Цей сервіс дає можливість програмам отримувати доступ до даних, бізнес-логіки та функціональних можливостей ваших серверних сервісів через API. За допомогою API Gateway можна легко створювати API RESTful та WebSocket, які є важливою складовою реалізації двостороннього зв'язку в реальному часі в програмах.

API Gateway бере на себе всі необхідні завдання, пов'язані з прийомом та обробкою великої кількості одночасних викликів API, включаючи управління трафіком, підтримку CORS, авторизацію та контроль доступу, обмеження кількості запитів, моніторинг та керування версіями API. Робота з API Gateway не вимагає мінімальних абонплат або стартових внесків. Ви платите лише за отримані виклики API та переданий обсяг даних, а завдяки багаторівневій системі ціноутворення можна ефективно керувати витратами

відповідно до масштабу використання API. Користувачі можуть відправляти до 10000 запитів на секунду без проблем.

3.7.2 AWS Lambda

AWS Lambda – це бессерверний, керований визначний сервіс, який дозволяє вам виконувати код практично для неограниченого типу додатків або сервісів без надання серверів та їх обслуговування. Ви можете включити Lambda з понад 200 сервісів і додатків, наданих за моделями ПО як послуга (SaaS), оплачувана тільки ресурсами, які використовуєте.

Під час першого виклику функції AWS Lambda створює інстанс функції та запускає її метод-обробник для обробки події. Коли функція повертає відповідь, вона залишається активною та чекає на обробку додаткових подій. Якщо ви знову викликаєте функцію під час обробки першої події, Lambda ініціалізує інший інстанс, і функція обробляє обидві події одночасно. У міру надходження нових подій Lambda направляє їх до доступних інстансів і створює нові інстанси за потреби. Коли кількість запитів зменшується, Lambda зупиняє невикористані інстанси, щоб звільнити можливості масштабування для інших функцій. За замовчуванням регіональна квота паралельності починається з 1000 інстансів. High availability – Lambda запускає вашу функцію в кількох зонах доступності, щоб гарантувати, що вона доступна для обробки подій у разі припинення обслуговування в одній зоні. Якщо ви налаштуєте свою функцію на підключення до віртуальної приватної хмари (VPC) у своєму обліковому записі, вкажіть у яких зонах доступності вона буде працювати, щоб забезпечити високу доступність.

3.7.3 AWS DynamoDB

AWS DynamoDB – це повністю керований сервіс бази даних, що надає високопродуктивне та надійне зберігання та обробку даних в масштабах від

мінімальних до дуже великих. Цей сервіс відрізняється відмінною масштабованістю, гнучкістю та надійністю, що робить його ідеальним вибором для будь-яких додатків, які потребують високошвидкісного доступу до даних.

AWS DynamoDB впроваджує модель "плати за використання", що дозволяє користувачам платити лише за те, що вони використовують, без потреби в передоплаті або мінімальних витрат. Користувачі мають можливість регулювати потужність обчислення та масштабування в/в незалежно для кожної таблиці, що дозволяє оптимізувати витрати відповідно до потреб конкретного додатка.

Однією з ключових особливостей DynamoDB є його висока доступність та надійність. Сервіс автоматично розподіляє дані через різні області та реплікує їх для забезпечення неперервності роботи. Такий підхід дозволяє зменшити можливість виникнення збоїв та забезпечити швидкий доступ до даних для користувачів навіть у випадку виникнення проблем в одній з областей.

4 РОЗРОБКА КОМПОНЕНТІВ У TERRAFORM

Структура Terraform проекту може бути описана лише в одному файлі: main.tf, а також поділена на декілька, а саме: main.tf, variables.tf, outputs.tf, та файли, де описуються компоненти.

4.1 Файл main.tf

Основный файл, у якому описуються конфігурація проекту, за якою будуть відтворені компоненти у хмарі. У файлі знаходяться блоки коду, які відповідають за роботу з окремими об’єктами – ресурсами.

4.2 Блок коду terraform

Цей блок має основну конфігурацию для визначення загальної версії terraform – required_version та опис провайдера з яким пропонується робота та його версія – required_providers. Реалізація блоку наведена у лістингу 4.1;

Лістинг 4.1 – Блок коду terraform

```
terraform {
  required_version = ">=1.0.1"

  required_providers {
    aws = {
      source  = "hashicorp/aws"
      version = "~> 3.0"
    }
  }
}
```

4.3 Блок коду provider

Даний блок коду визначає який провайдер буде використовуватися, регіон для створення компонентів, а також логін – access_key і пароль – secret_key для з’єднання з хмарним постачальником. Реалізація блоку наведена у лістингу 4.2;

Лістинг 4.2 – Блок коду provider

```
provider "aws" {
 region     = var.region
 access_key = ""
 secret_key = ""
}
```

4.4 Файл api-gateway.tf

У цьому файлі написана конфігурація Api Gateway. Він приймає та оброблює виклики API та за логікою загружає зображення до s3 бакету. Реалізація блоків наведена у лістингу 4.3;

Лістинг 4.3 – Блоки коду для створення Api Gateway конфігурації

```
resource "aws_api_gateway_rest_api" "students_api" {
  name = "students"
}

resource "aws_api_gateway_resource" "delete_resource" {
  parent_id =
aws_api_gateway_rest_api.students_api.root_resource_id
  path_part = "delete"
  rest_api_id = aws_api_gateway_rest_api.students_api.id
}

resource "aws_api_gateway_resource" "get_resource" {
  parent_id =
aws_api_gateway_rest_api.students_api.root_resource_id
  path_part = "get"
  rest_api_id = aws_api_gateway_rest_api.students_api.id
```

```
}

resource "aws_api_gateway_resource" "sort_by_group_resource" {
  parent_id =
aws_api_gateway_rest_api.students_api.root_resource_id
  path_part = "sort-by-group"
  rest_api_id = aws_api_gateway_rest_api.students_api.id
}

resource "aws_api_gateway_resource" "store_resource" {
  parent_id =
aws_api_gateway_rest_api.students_api.root_resource_id
  path_part = "store"
  rest_api_id = aws_api_gateway_rest_api.students_api.id
}

resource "aws_api_gateway_resource" "update_resource" {
  parent_id =
aws_api_gateway_rest_api.students_api.root_resource_id
  path_part = "update"
  rest_api_id = aws_api_gateway_rest_api.students_api.id
}

resource "aws_api_gateway_method" "delete_method" {
  rest_api_id   = aws_api_gateway_rest_api.students_api.id
  resource_id   = aws_api_gateway_resource.delete_resource.id
  http_method   = "POST"
  authorization = "NONE"
}

resource "aws_api_gateway_method" "get_method" {
  rest_api_id   = aws_api_gateway_rest_api.students_api.id
  resource_id   = aws_api_gateway_resource.get_resource.id
  http_method   = "GET"
  authorization = "NONE"
}

resource "aws_api_gateway_method" "sort_by_group_method" {
  rest_api_id   = aws_api_gateway_rest_api.students_api.id
  resource_id   =
aws_api_gateway_resource.sort_by_group_resource.id
  http_method   = "GET"
  authorization = "NONE"
}

resource "aws_api_gateway_method" "store_method" {
  rest_api_id   = aws_api_gateway_rest_api.students_api.id
  resource_id   = aws_api_gateway_resource.store_resource.id
  http_method   = "POST"
  authorization = "NONE"
}

resource "aws_api_gateway_method" "update_method" {
```

```
  rest_api_id   = aws_api_gateway_rest_api.students_api.id
  resource_id   = aws_api_gateway_resource.update_resource.id
  http_method   = "POST"
  authorization = "NONE"
}

resource "aws_api_gateway_integration" "delete_integration" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id = aws_api_gateway_resource.delete_resource.id
  http_method = aws_api_gateway_method.delete_method.http_method

  type                    = "AWS_PROXY"
  integration_http_method = "POST"
  uri                     = aws_lambda_function.delete.arn
}

resource "aws_api_gateway_integration" "get_integration" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id = aws_api_gateway_resource.get_resource.id
  http_method = aws_api_gateway_method.get_method.http_method

  type                    = "AWS_PROXY"
  integration_http_method = "GET"
  uri                     = aws_lambda_function.get.arn
}

resource "aws_api_gateway_integration"
"sort_by_group_integration" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id =
aws_api_gateway_resource.sort_by_group_resource.id
  http_method =
aws_api_gateway_method.sort_by_group_method.http_method

  type                    = "AWS_PROXY"
  integration_http_method = "GET"
  uri                     = aws_lambda_function.sort-by-
group.arn
}

resource "aws_api_gateway_integration" "store_integration" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id = aws_api_gateway_resource.store_resource.id
  http_method = aws_api_gateway_method.store_method.http_method

  type                    = "AWS_PROXY"
  integration_http_method = "POST"
  uri                     = aws_lambda_function.store.arn
}

resource "aws_api_gateway_integration" "update_integration" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id = aws_api_gateway_resource.update_resource.id
```

```
  http_method = aws_api_gateway_method.update_method.http_method

  type                    = "AWS_PROXY"
  integration_http_method = "POST"
  uri                     = aws_lambda_function.update.arn
}

resource "aws_api_gateway_method_settings" "method_settings" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  stage_name  = "prod"

  settings {
    method_path     = "/*/*"
    metrics_enabled = true
    logging_level   = "INFO"
  }
}

resource "aws_api_gateway_integration_response"
"integration_response" {
  rest_api_id = aws_api_gateway_rest_api.students_api.id
  resource_id = aws_api_gateway_resource.delete_resource.id
  http_method = aws_api_gateway_method.delete_method.http_method
  status_code = aws_api_gateway_method.delete_method.status_code

  response_templates = {
    "application/json" = ""
  }
}
```

4.5 Файл dynamodb.tf

Створення бази даних для інформації про студентів. Основне призначення зберігати дані для подальшої обробки за допомогою lambda сервісу. Реалізація блоків наведена у лістингу 4.4;

Лістинг 4.4 – Блоки коду для створення dynamodb

```
provider "aws" {
  region = "us-east-1" # Change to your desired AWS region
}

resource "aws_dynamodb_table" "students" {
  name           = "students"
  billing_mode   = "PAY_PER_REQUEST"
  hash_key       = "surname"
```

```
  range_key      = "name"

  attribute {
    name = "surname"
    type = "S"
  }

  attribute {
    name = "name"
    type = "S"
  }

  global_secondary_index {
    name               = "group_index"
    hash_key           = "group"
    projection_type    = "ALL"
    read_capacity      = 5
    write_capacity     = 5
  }
}
```

4.6 Файл lambda.tf

Основний ресурс, логікою якого є взаємодія з базою даних. Реалізація блоків наведена у лістингу 4.5;

Лістинг 4.5 – Блоки коду для створення lambda функції

```
resource "aws_lambda_function" "delete" {
  function_name    = "delete"
  handler          = "lambda_function.handler"
  runtime          = "python3.8"
  memory_size      = 128
  timeout          = 10
  filename         = "home/user/lambda_delete.zip"
}

resource "aws_lambda_function" "get" {
  function_name    = "get"
  handler          = "lambda_function.handler"
  runtime          = "python3.8"
  memory_size      = 128
  timeout          = 10
  filename         = "home/user/lambda_get.zip"
}
```

```
resource "aws_lambda_function" "sort_by_group" {
  function_name    = "sort-by-group"
  handler          = "lambda_function.handler"
  runtime          = "python3.8"
  memory_size      = 128
  timeout          = 10
  filename         = "home/user/lambda_sort-by-group.zip"
}

resource "aws_lambda_function" "store" {
  function_name    = "store"
  handler          = "lambda_function.handler"
  runtime          = "python3.8"
  memory_size      = 128
  timeout          = 10
  filename         = "home/user/lambda_store.zip"
}

resource "aws_lambda_function" "update" {
  function_name    = "update"
  handler          = "lambda_function.handler"
  runtime          = "python3.8"
  memory_size      = 128
  timeout          = 10
  filename         = "home/user/lambda_update.zip"
}
```

4.7 Файл iam-configuration.tf

Опис ресурсів для визначення дозволу на виконання якоїсь дії. Наприклад: дозвіл lambda сервісу взаємодіяти з dynamodb. Реалізація блоків наведена у лістингу 4.6;

Лістинг 4.6 – Блоки коду для створення iam конфігурації

```
// Lambda execution role
resource "aws_iam_role" "lambda-execution-role" {
  name = var.lambda_role_name
  assume_role_policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = "sts:AssumeRole"
        Effect = "Allow"
        Sid    = ""
```

```
        Principal = {
          Service = "lambda.amazonaws.com"
        }
      },
    ]
  })
}

// Lambda execution role policy
resource "aws_iam_policy" "lambda-policy" {
  name = var.lambda_policy_name
  policy = jsonencode({
    Version = "2012-10-17"
    Statement = [
      {
        Action = [
          "dynamodb:Query",
          "dynamodb:GetItem",
          "dynamodb:PutItem",
          "dynamodb:UpdateItem",
          "dynamodb:DeleteItem"
        ]
        Effect   = "Allow"
        Resource = "*"
      },
    ]
  })
}
// Attach policy to Lambda execution role
resource "aws_iam_policy_attachment" "attach-lambda-policy" {
  name       = "lambda-dynamodb-access"
  roles      = [aws_iam_role.lambda-execution-role.name]
  policy_arn = aws_iam_policy.lambda-policy.arn
}
```

4.8 Файл variables.tf

Файл для опису змінних, які використовуються у файлах з конфігурацією ресурсів. Реалізація блоків наведена у лістингу 4.7

Лістинг 4.7 – Блоки коду для опису змінних

```
variable "dynamodb_name" {
  default     = "image-filestore-bucket"
  description = "Name of s3 bucket"
}
variable "region" {
```

```
  default     = "us-east-1"
  description = "Region where resources are created"
}

variable "lambda_role_name" {
  default     = "lambda-execution-role"
  description = "Lambda execution role"
}

variable "lambda_name" {
  default     = "image-rekognition-lambda"
  description = "Lambda name"
}

variable "lambda_policy_name" {
  default     = "lambda-permissions"
  description = "lambda permissions"
}

variable "output_file" {
  default     = "D:/graduation/rekognition-lambda.zip"
  description = "place where to put zip"
}

variable "api_name" {
  default     = "api"
  description = "who will send email"
}
```

4.9 Файл outputs.tf

Файл для опису змінних, які доступні користувачу terraform для подальшого використання або ознайомлення. Змінні доступні при першому запуску відтворення конфігурації та доступні у консолі. Реалізація блоків наведена у лістингу 4.9

Лістинг 4.9 – Блоки коду для опису змінніх , які доступні у консолі

```
output "invokeUrl" {
  value =
"${aws_api_gateway_deployment.prod.invoke_url}${aws_api_gateway_
stage.stage.stage_name}/${aws_api_gateway_resource.
resource.path_part}/${aws_api_gateway_resource.itemResource.path
_part}"
}
```

5 РОЗРОБКА LAMBDA ФУНКЦІЙ

Основну логіку роботи сервісу представляє lambda, написаний на мові Python код, який взаємодіє з dynamodb: зберігає, модифікує, сортує, отримує та видаляє дані з бази даних.

Лістинг 5.1 – Python код для зберігання даних студента

```
import json
import boto3
import os
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table_name = os.environ['TABLE_NAME']
table = dynamodb.Table(table_name)

def lambda_handler(event, context):
    try:
        # Parse the request body
        body = json.loads(event['body'])

        # Extract data
        name = body['name']
        surname = body['surname']
        group = body['group']
        subjects = body['subjects']
        course_number = body['course_number']

        # Create the item to store
        item = {
            'name': name,
            'surname': surname,
            'group': group,
            'subjects': subjects,
            'course_number': course_number
        }

        # Store the item in the DynamoDB table
        table.put_item(Item=item)

        return {
            'statusCode': 200,
            'body': json.dumps({'message': 'Student data stored
successfully'})
```

```
        }
    except ClientError as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'body': json.dumps({'error': str(e)})
        }
```

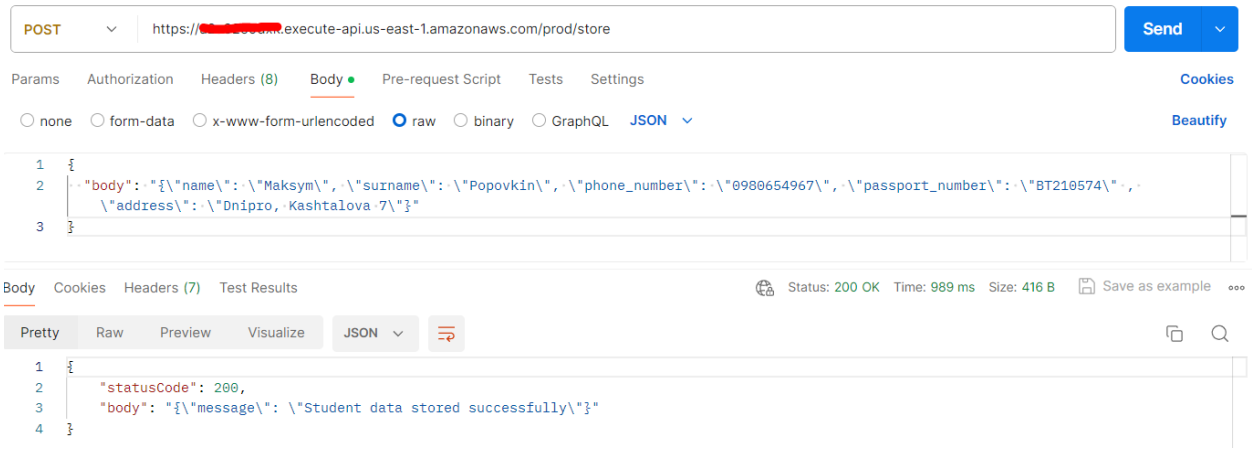

Рисунок 5.1 – Зберігання інформації про студента

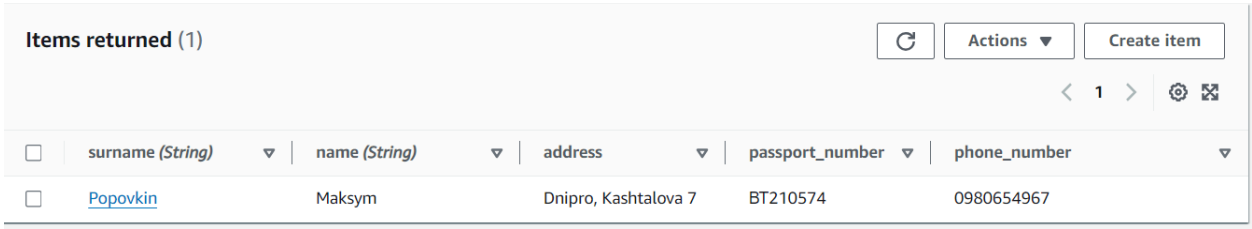

Рисунок 5.2 – Перевірка збереження даних в DynamoDB

Лістинг 5.2 – Python код для отримання даних про студента

```
import json
import boto3
from botocore.exceptions import ClientError
from decimal import Decimal

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('students')
```

```
# Helper function to convert Decimal to float/int
def decimal_default(obj):
    if isinstance(obj, Decimal):
        return float(obj) if obj % 1 != 0 else int(obj)
    raise TypeError

def lambda_handler(event, context):
    try:
        # Ensure 'surnames' is in the event
        if 'surnames' not in event:
            raise KeyError('surnames')

        # Directly parse the surnames from the event
        surnames = event['surnames']

        # Initialize an empty list to store the students' data
        students = []

        # Scan the table to get all items
        response = table.scan()

        # Check if 'Items' is in the response
        if 'Items' in response:
            # Filter the items by the specified surnames
            for item in response['Items']:
                if item['surname'] in surnames:
                    # Use the helper function to convert the
response
                    item = json.loads(json.dumps(item,
default=decimal_default))
                    students.append(item)

            # Format the response
            formatted_students = []
            for student in students:
                formatted_response = f"surname:
{student['surname']}, name: {student['name']}"
                remaining_fields = ', '.join(
                    f"{key}: {', '.join(value) if
isinstance(value, list) else value}"
                    for key, value in student.items()
                    if key not in ['surname', 'name']
                )
                if remaining_fields:
                    formatted_response += f",
{remaining_fields}"
                formatted_students.append(formatted_response)

            # Join the formatted students with newlines
            formatted_response_text =
'\n'.join(formatted_students)

            return {
```

```
                'statusCode': 200,
                'body': formatted_response_text
            }
        else:
            return {
                'statusCode': 404,
                'body': 'No students found'
            }
    except KeyError as e:
        return {
            'statusCode': 400,
            'body': f"Error: Missing key {str(e)}"
        }
    except ClientError as e:
        return {
            'statusCode': 500,
            'body': f"Error: {str(e)}"
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'body': f"Error: {str(e)}"
        }
```

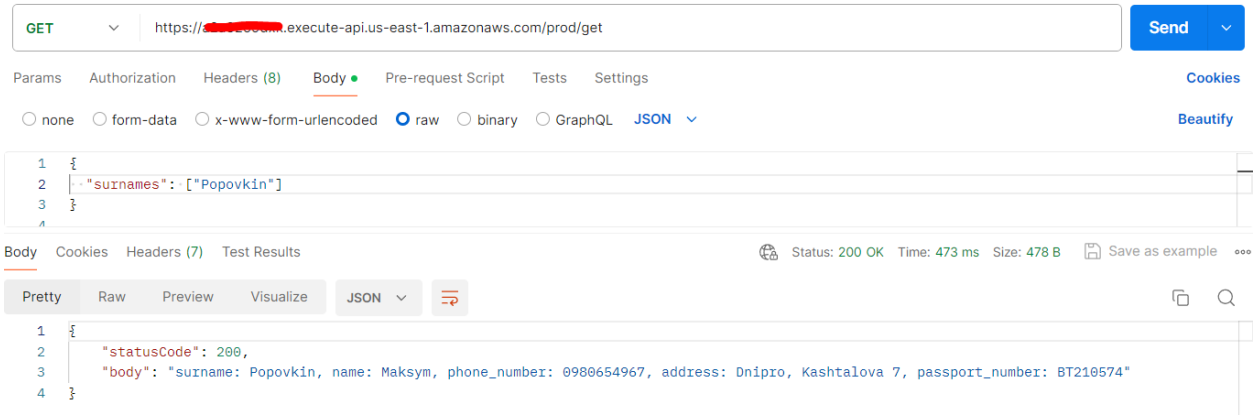

Рисунок 5.3 – Отримання інформації про студента

Лістинг 5.3 – Python код для модифікації даних студента

```
import json
import boto3
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('students')
```

```
def lambda_handler(event, context):
    try:
        # Parse the request body
        body = json.loads(event['body'])

        # Extract data
        name = body['name']
        surname = body['surname']
        update_expression = "SET "
        expression_attribute_values = {}
        expression_attribute_names = {}  # Define expression
attribute names

        # Determine what needs to be updated
        if 'group' in body:
            update_expression += "#grp = :group, "
            expression_attribute_values[':group'] =
body['group']
            expression_attribute_names['#grp'] = 'group'  #
Define expression attribute name
        if 'subjects' in body:
            update_expression += "subjects = :subjects, "
            expression_attribute_values[':subjects'] =
body['subjects']
        if 'course_number' in body:
            update_expression += "course_number =
:course_number, "
            expression_attribute_values[':course_number'] =
body['course_number']

        # Remove trailing comma and space
        update_expression = update_expression.rstrip(', ')

        # Update the item in the DynamoDB table
        response = table.update_item(
            Key={
                'surname': surname,
                'name': name
            },
            UpdateExpression=update_expression,

ExpressionAttributeValues=expression_attribute_values,
            ExpressionAttributeNames=expression_attribute_names
# Pass expression attribute names
        )

        return {
            'statusCode': 200,
            'body': json.dumps({'message': 'Student data updated
successfully'})
        }
    except ClientError as e:
        return {
```

```
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'body': json.dumps({'error': str(e)})
        }
```

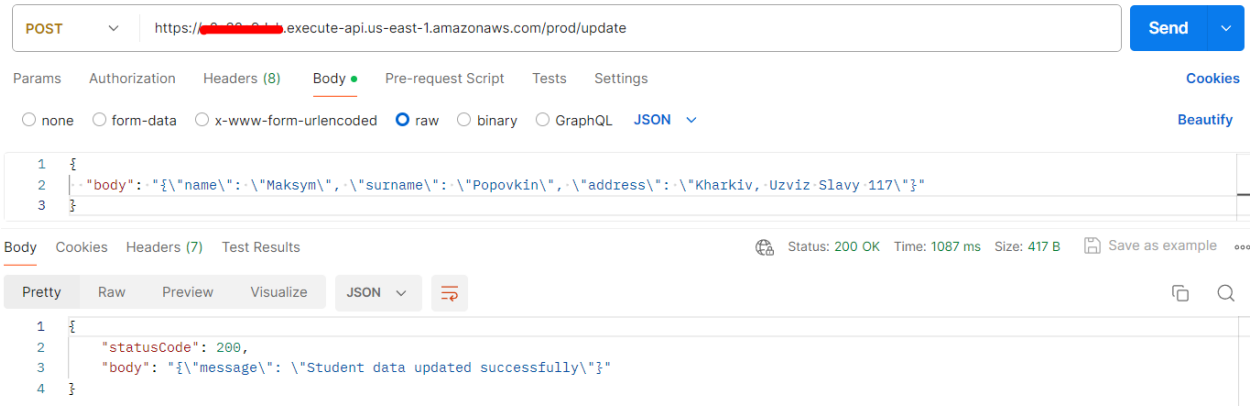

Рисунок 5.4 – Модифікація інформації про студента

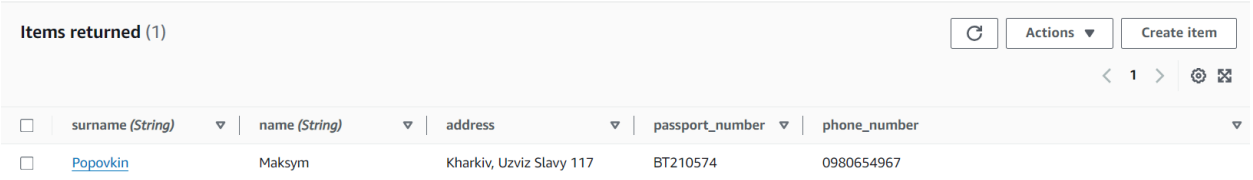

Рисунок 5.5 – Перевірка модифікації даних студента

Лістинг 5.4 – Python код для видалення даних студента

```
import json
import boto3
from botocore.exceptions import ClientError

dynamodb = boto3.resource('dynamodb')
table = dynamodb.Table('students')

def lambda_handler(event, context):
    try:
        # Parse the request body
        body = json.loads(event['body'])
```

```
        # Extract data
        student_name = body.get('name')
        student_surname = body.get('surname')

        if not student_surname:
            return {
                'statusCode': 400,
                'body': json.dumps({'error': 'Surname attribute
is required'})
            }

        # Delete the item from the DynamoDB table
        table.delete_item(
            Key={'surname': student_surname}
        )

        return {
            'statusCode': 200,
            'body': json.dumps({'message': 'Student data deleted
successfully'})
        }
    except ClientError as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': str(e)})
        }
    except Exception as e:
        return {
            'statusCode': 400,
            'body': json.dumps({'error': str(e)})
        }
```

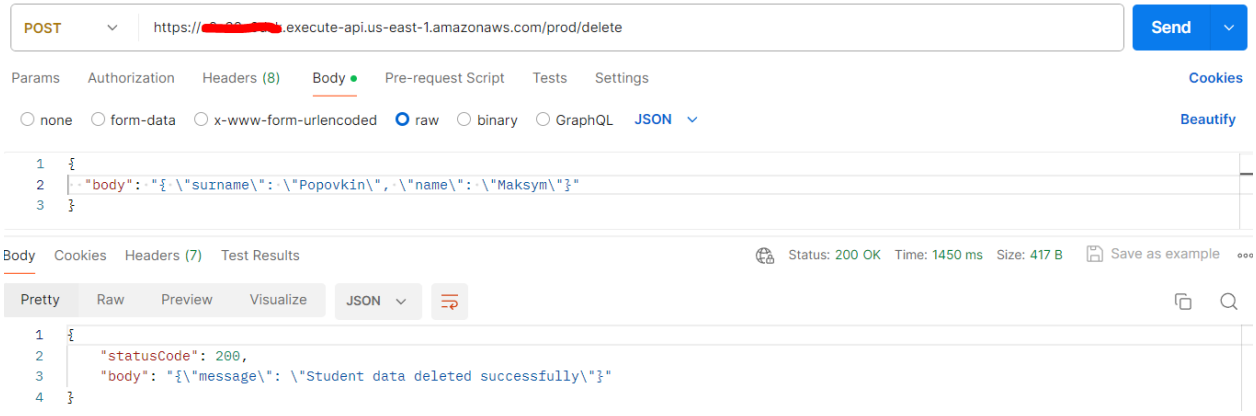

Рисунок 5.4 – Видалення інформації про студента

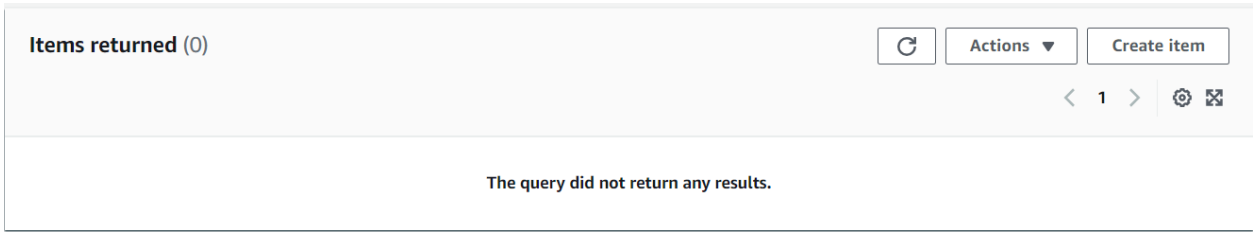

Рисунок 5.5 – Підтвердження видалення інформації про студента

Лістинг 5.5 – Python код для сортування даних студента за групою

```
import json
import boto3

# Initialize DynamoDB client
dynamodb = boto3.client('dynamodb')

def lambda_handler(event, context):
    try:
        # Parse the request body
        body = json.loads(event['body'])

        # Extract group information
        group = body.get('group')

        if not group:
            return {
                'statusCode': 400,
                'body': json.dumps({'error': 'Group attribute is
required'})
            }

        # Query students based on group using GSI
        response = dynamodb.query(
            TableName='students',
            IndexName='group',
            KeyConditionExpression='#grp = :group',  # Using
expression attribute name
            ProjectionExpression='surname, #nm',  # Projection
expression to include only surname and name
            ExpressionAttributeNames={'#grp': 'group', '#nm':
'name'},  # Mapping reserved keyword to expression attribute
name
            ExpressionAttributeValues={':group': {'S': group}}
# Assuming 'group' is a string attribute
        )

        # Extracting only the required attributes from the
response
```

```
        students = [{'surname': item['surname']['S'], 'name':
item['name']['S']} for item in response.get('Items', [])]

        return {
            'statusCode': 200,
            'body': json.dumps({'students': students})
        }
    except Exception as e:
        return {
            'statusCode': 500,
            'body': json.dumps({'error': f"Internal Server
Error: {str(e)}"})
        }
```

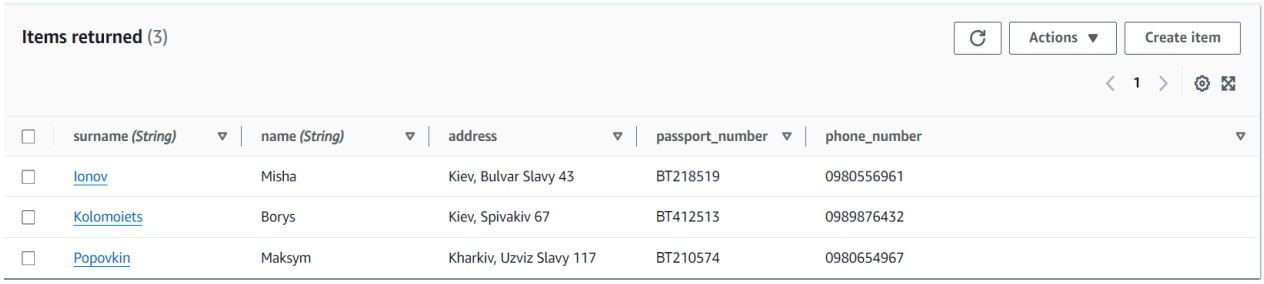

Рисунок 5.6 – Інформація в базі даних DynamoDB

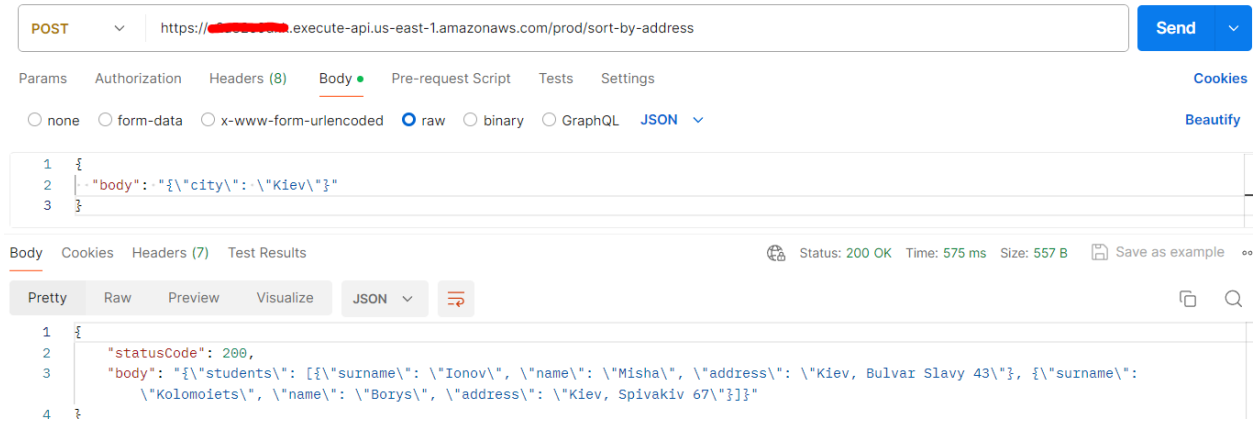

Рисунок 5.7 – Відсортовані дані студентів за адресою

На підставі цих рисунків, ми можемо вважати виконану мою кваліфікаційну роботу в повному обсязі.

6 ІНСТРУКЦІЯ КОРИСТУВАЧА

Для розгортання компонентів інфраструктури та їх тестування потрібно виконати декілька дій, а саме:

Крок 1. Мати акаунт в AWS, або створити його. Для цього скористайтеся посиланням [14].

Крок 2. Встановити на ваш локальний пристрій інструмент командного рядка AWS (CLI), посилання [15].

Крок 3. Завантажити Terraform та Python code з Github, посилання [16].

Крок 4. Завантажити Postman на локальний пристрій [17].

Крок 5. У консолі відкрити клоновану директорію та виконати команди Terraform для будування компонентів:

- aws configure, для підключення до aws;
- terraform init, для завантаження основних компонентів terraform;
- terraform apply, для початку розгортання компонентів.
- Через декілька секунд, користувач побачить в консолі інструкцію як користуватися компонентами

ВИСНОВКИ

У процесі виконання кваліфікаційної роботи були розглянуті та вивчені існуючі інструменти та рішення, які забезпечують можливість розробки та використання програмних компонентів в області обробки та зберігання персональних даних. Основною платформою для дослідження вибрана AWS Serverless архітектура, яка надає ефективність та гнучкість у впровадженні та масштабуванні рішень.

Особливий акцент у роботі було зроблено на створенні та використанні Lambda функцій написаних на мові програмування Python. Ці Lambda функції виявилися незамінними у випробуванні та впровадженні компонентів системи, забезпечуючи швидкість та ефективність розробки.

Одним із ключових аспектів дослідження став вибір Terraform як інструменту для швидкого розгортання та керування інфраструктурою. Використання Terraform дозволило забезпечити простоту та швидкість зміни конфігурації інфраструктури, що є критичним в умовах постійно змінюючогося середовища.

У результаті дослідження було підтверджено, що використання AWS Serverless архітектури разом з Lambda функціями та Terraform дозволяє створювати ефективні та гнучкі рішення для обробки та зберігання персональних даних студентів, забезпечуючи високу швидкість розробки та масштабованість системи.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Хмарні технології. Переваги і недоліки (Документація від Вал Тек) [Електронний ресурс] – URL: https://valtek.com.ua/ua/system-integration/it-infrastructure/clouds/cloud-technologies.

2. Сервіси для розробників (Документація від softline) [Електронний ресурс] – URL: https://aws.softline.com/solutions/servisy-aws-dlya-razrabotki-i-testirovaniya.

3. Що таке Microsoft Azure (Документація від Simpla) [Електронний ресурс] – URL: https://simpla.com.ua/ru/blog/chto-takoe-microsoft-azure.

4. Що таке хмарна платформа Google [Електронний ресурс] – URL: https://education-wiki.com/.

5. Всесвітнє визнання за технологічні інновації [Електронний ресурс] – URL: https://www.alibabacloud.com/about/about-new

6. Хмарні обчислення IBM Cloud, платформа SoftLayer [Електронний ресурс] – URL: https://www.itparadigma.ru/blog/oblachnyie-vyichisleniya-ibm-cloud-platforma-softlayer/.

7. Програми. платформа. Інфраструктури. [Електронний ресурс від Oracle] – URL: https://www.oracle.com/cis/cloud/cloud-summary.html.

8. Salesforce хмара [Електронний ресурс] – URL: https://roi4cio.com/catalog/product/salesforce-sales-cloud.

9. Що таке хмарна платформа SAP [Електронний ресурс] – URL: https://dataseed.ru/produkty/sap-cloud-platform-chto-takoe-oblachnaya-platforma-sap/.

10. Rackspace Cloud Management [Електронний ресурс від spaculus software] – URL: https://www.spaculus.org/rackspace-cloud-management-services.

11. VMware хмара [Електронний ресурс від vmware] – URL: https://www.vmware.com/cloud-solutions.html.

12. Автоматизуйте інфраструктуру в будь-якій хмарі [Електронний ресурс від Terraform] – URL: https://www.terraform.io/

13. AWS документація [Електронний ресурс від AWS] – URL: https://docs.aws.amazon.com/

14. Create AWS account [Електронний ресурс] – URL: https://aws.amazon.com/ru/account/

15. Install AWS CLI [Електронний ресурс від AWS] https://docs.aws.amazon.com/cli/latest/userguide/getting-started-install.html

16. Clone code from Github [Електронний ресурс] - URL: https://github.com/Safankov/graduation

17. Install Postman [Електронний ресурс] - URL: https://www.postman.com/downloads/

18. Шматко О.В, Сафанков Д.В, Розробка та впровадження хмарної платформи для токенізації персональних даних студентів // Системи управляння, навігації та зв’язку, № 2, 2024, С 87-94.