Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет кібербезпеки
(повна назва)

Кафедра інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти другий (магістерський)

Безпечна платформа на базі AWS для збору даних та співпраці дослідників
(тема)

Виконав:
здобувач 2 року навчання,
групи ІМІм-24-2
Григор'єв Д.А.
(прізвище, ініціали)
Спеціальність
172 Електронні комунікації та радіотехніка
(код і повна назва спеціальності)
Тип програми освітньо-наукова
(освітньо-професійна або освітньо-наукова)
Освітня програма
«Інформаційно-мережна інженерія»
(повна назва освітньої програми)
Керівник доц. Кривенко С.А.
(посада, прізвище, ініціали)

Допускається до захисту

Зав. кафедри _____ Москалець М.В.
(підпис) (прізвище, ініціали)

2026 р.

Не містить відомостей, заборонених до відкритого публікування

Студент _____

Керівник_____

Харківський національний університет радіоелектроніки

Факультет кібербезпеки
(повна назва)

Кафедра інформаційно-мережної інженерії
(повна назва)

Рівень вищої освіти другий (магістерський)

Спеціальність 172 «Електронні комунікації та радіотехніка»
(код і повна назва)

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

Освітня програма «Інформаційно-мережна інженерія»
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___»_____20__ р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Григор'єв Данило Андрійович
(прізвище, ім'я, по батькові)

1. Тема роботи Безпечна платформа на базі AWS для збору даних та співпраці дослідників

затверджена наказом університету від 23 березня 2026р. № 232 Ст_

2. Термін подання студентом роботи до екзаменаційної комісії 26 травня 2026р.

3. Вихідні дані до роботи: _____

У цій роботі Вам необхідно використовувати знайомі сервіси AWS для створення рішення. Певні розділи завдання мають на меті перевірити ваші навички, які ви отримали під час курсу «Глобальна інформаційна інфраструктура». Необхідно застосовувати принципи архітектурного проєктування для виконання наступного: створити екземпляр бази даних, до якого PHP-додаток може надсилати запити; створити та розгорнути високо доступний PHP-додаток з розподілом навантаження між кількома веб-серверами та зонами доступності; використовувати AWS Secrets Manager; Імпортувати дані в базу даних MySQL з файлу дампа SQL; захистити додаток, щоб запобігти публічному доступу до серверів додатків та бекенд-систем.

4. Перелік питань, що потрібно опрацювати в роботі: _____

аналіз сучасної архітектури; модель високо доступного середовища; результати тестування платформи та перевірка імпортованих даних.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) 21 слайд у форматі PowerPoint _____

6. Консультанти розділів роботи (п.6 включається до завдання 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
Основна частина	доц. Кривенко С.А.		19.05.2026

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Терміни виконання етапів роботи	Примітка
1	*Ознайомлення із завданням. Уточнення ТЗ.*	*28.03-01.04*	вик
2	*Підбір літератури за темою роботи.*	*01.04-15.04*	вик
3	*Виконання розділу 1*	*16.04-30.04*	вик
4	*Виконання розділу 2*	*01.05-11.05*	вик
5	*Виконання розділу 3*	*12.05-17.05*	вик
7	*Оформлення презентаційного матеріалу, підготовка до захисту у ЕК*	*17.05-19.05*	вик

Дата видачі завдання 28 березня 2026 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Кривенко С.А
(підпис) (посада, прізвище, ініціали)

РЕФЕРАТ

Пояснювальна записка випускної магістерської кваліфікаційної роботи містить: 67 стор., 36 рис., 2 табл., 20 джерел.

AWS, AMAZON CLOUD9, AMAZON DYNAMO DB, AMAZON SIMPLE STORAGE SERVICE, AMAZON SIMPLE QUEUE SERVICE, AMAZON Simple NOTIFICATION SERVICE.

Об’єкт дослідження – безпечна платформа на базі AWS для збору даних.

Предмет дослідження – приклад організації соціальних досліджень, некомерційної організації, яка надає веб-сайт дослідникам у галузі соціальних наук для отримання статистики глобального розвитку. Наприклад, відвідувачі сайту можуть шукати різні дані, такі як тривалість життя в будь-якій країні світу за останні 10 років.

Мета кваліфікаційної роботи — завершити впровадження, переконатися в безпеці веб-сайту та підтвердити, що веб-сайт повертає дані зі сторінки запиту.

Методи досліджень – сервіси Amazon Web Service.

ABSTRACT

The explanatory note of the final master's qualification work contains: 67 pages, 36 figures, 2 tables, 20 sources.

AWS, AMAZON CLOUD9, AMAZON DYNAMO DB, AMAZON SIMPLE STORAGE SERVICE, AMAZON SIMPLE QUERY SERVICE, AMAZON Simple NOTIFICATION SERVICE.

The object of the study is a secure AWS-based platform for data collection.

The research subject is an example of a social research organization, a non-profit organization that provides a website for social science researchers to obtain statistics on global development. For example, visitors to the site can search for various data, such as life expectancy in any country in the world over the past 10 years.

The goal of the qualification work is to complete the implementation, ensure the security of the website, and confirm that the website returns data from the request page.

Research methods - Amazon Web Service services.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, СИМВОЛІВ, ОДИНИЦЬ І ТЕРМІНІВ

ASG – Auto Scaling Group – група автоматичного масштабування

AWS – Amazon Web Services – дочірня компанія Amazon.com, що надає платформу хмарних обчислень в оренду приватним особам, компаніям та урядам на основі платної підписки

DynamoDB – Amazon DynamoDB – безсерверна база даних NoSQL.

EC2 – Amazon Elastic Compute Cloud – веб сервіс, котрий надає обчислювальні потужності в хмарі

ENI – Elastic Network Interface – віртуальний мережевий інтерфейс для мережевих ресурсів.

EventBridge – сервіс для управління подіями та інтеграції додатків.

HTML – HyperText Markup Language – мова розмітки для створення веб-сторінок.

HVAC – Heating, Ventilation, & Air Conditioning = це комплекс інженерних систем, призначених для опалення, вентиляції та кондиціонування повітря.

IDE – integrated development environment – Інтегроване середовище розробки.

IAM – Identity and Access Management – сервіс для управління правами доступу та ідентифікацією.

IMDS – EC2 Instance Metadata Service – Служба метаданих екземплярів EC2

Lambda – AWS Lambda – сервіс для виконання коду без управління серверами.

LAMP – абревіатура набору вільного програмного забезпечення з відкритим кодом, в який входять Linux, веб сервер Apache, MySQL та інтерпретатор Perl/PHP/Python — основні компоненти для побудови життєздатного багатоцільового веб сервера

RDS – Amazon Relational Database Service – сервіс для управління реляційними базами даних.

S3 – Amazon Simple Storage Service – сервіс для зберігання даних у хмарі.

URL – Uniform Resource Locator – єдиний вказівник на ресурс

VPC – Virtual Private Cloud – віртуальна приватна мережа для безпечного розгортання ресурсів.

ВСТУП

Кваліфікаційна робота пов’язана з розв’язанням наступної бізнес проблєми [1]. Приклад організації соціальних досліджень – це (фіктивна) некомерційна організація, яка надає веб-сайт дослідникам у галузі соціальних наук для отримання статистики глобального розвитку. Наприклад, відвідувачі сайту можуть шукати різні дані, такі як тривалість життя в будь-якій країні світу за останні 10 років.

Ширлі Родрігес, дослідниця некомерційної організації, розробила веб-сайт. Вона вважала, що було б корисно поділитися зібраними нею даними з іншими дослідниками. Ширлі зберігає дані в базі даних MySQL, і дані доступні через веб-сайт PHP, який вона створила. Спочатку вона опублікувала сайт через комерційну хостингову компанію, яка надає обмежену підтримку з технічних питань та безпеки.

За останній рік веб-сайт Ширлі зріс у популярності. В результаті збільшення трафіку вона почала отримувати скарги на те, що сайт не такий адаптивний, як раніше. Вона також зазнала спроби порушення безпеки за допомогою програми-вимагача. Порушення безпеки було невдалим, але її керівник, Матео Джексон, запропонував Ширлі дослідити нові способи розміщення веб-сайту.

Ширлі почула про AWS і спочатку перенесла свій веб-сайт і базу даних до екземпляра EC2, який працює у публічній підмережі. Вона також запускає екземпляр MySQL на тому ж екземплярі EC2.

Ширлі звернулася до ХНУРЕ, щоб переконатися, що її поточний дизайн відповідає найкращим архітектурним практикам. Вона хоче переконатися, що має надійний і безпечний веб-сайт. Один з колег розпочав процес перенесення сайту на більш безпечну реалізацію, але його перевели на інший проєкт.

Мета кваліфікаційної роботи — завершити впровадження, переконатися в безпеці веб-сайту та підтвердити, що веб-сайт повертає дані зі сторінки запиту.

Для досягнення мети необхідно розв’язати наступні завдання.

Забезпечити безпечне розміщення бази даних MySQL.

Забезпечити безпечний доступ до бази даних.

Забезпечити анонімний доступ для веб-користувачів.

Запустити веб-сайт на екземплярі t2.micro EC2, що працює в приватних підмережах, та надати адміністраторам доступ через Secure Shell (SSH).

Забезпечити високу доступність веб-сайту за допомогою балансування навантаження.

Забезпечити автоматичне масштабування, що використовує шаблон запуску.

В першому розділі цієї роботи наведено кілька ключових міркувань, які повинні враховуватися у процесі прийняття рішень, описаний процес створення бази даних Amazon RDS MySQL. Визначено вимоги щодо групи безпеки бази даних, доступу до бази даних веб-застосунку, використанню Secrets Manager для зберігання імені користувача та пароля. З’ясовуються наслідки увімкнення додаткових функцій під час налаштування бази даних та їх вплив на оцінку вартості.

Другий розділ пояснювальної записки присвячений побудові моделі високо доступного середовища на основі керованого сервісу балансування навантаження програм у доступних публічних підмережах, створенню сервера застосунків у групі автоматичного масштабування. Досліджується використання групи автоматичного масштабування для запуску серверів застосунків на екземплярах EC2, на яких розміщено веб-застосунок. Аналізується використання політики відстеження цілей. Описується перенос даних з вихідної бази даних, яка знаходиться на екземплярі EC2, до нової бази даних Amazon RDS MySQL. Розробляється використання доступного файлу дампа даних, застосування диспетчера секретів, щоб отримати ім'я користувача та пароль.

Результати тестування отримання статистики глобального розвитку з веб-сайту наведені в третьому розділі. Щоб протестувати платформу, розглянуто відкриття веб-програми з браузера, використовуючи URL-адресу сервісу

балансування навантаження. Щоб переглянути та запитати імпортовані дані, розглянуто функцію запитів у програмі.

1 АНАЛІЗ СУЧАСНОЇ АРХІТЕКТУРИ

У цьому розділі описаний процес створення бази даних Amazon RDS MySQL. Група безпеки бази даних повинна дозволяти доступ до бази даних лише веб-застосунку. Використано Secrets Manager для зберігання імені користувача та пароля. Початкову базу даних збережено як countries. Використано групу підмережі бази даних, яка вже налаштована в лабораторному режимі. Враховано що увімкнення додаткових функцій під час налаштування бази даних може призвести до стягнення плати, яка вплине на оцінку вартості. Увімкнена лише функція, яка необхідна для даної роботи.

1.1 Міркування щодо рівня бази даних

У цьому підрозділі наведено огляд того, що слід враховувати під час вибору бази даних [2].

1.1.1 Ключові міркування

Архітектору мережі, доводиться вибирати між різними типами баз даних, коли розглядається, який тип найкраще впорається з певним робочим навантаженням. Перш ніж вибрати базу даних, є кілька ключових міркувань, які повинні враховуватися у процесі прийняття рішень [3].

Міркування 1 – масштабованість.

Важливо мати правильно масштабовану базу даних. У випадку традиційних локальних баз даних вплив на продуктивність бази даних під час її масштабування може бути непередбачуваним і може вимагати простою. Якщо недостатньо розширити свою базу даних, програми можуть перестати працювати. Якщо надмірно розширити свою базу даних, збільшуються свої початкові витрати, на

закупівлю ресурсів, які не потрібні, що порушує принцип оптимізації витрат AWS Well-Architected Framework. В ідеалі слід вибрати рішення для бази даних, яке має ресурси для обробки необхідної пропускної здатності під час запуску, а також яке можна масштабувати пізніше, якщо пропускну здатність потрібно збільшити.

Міркування 2 – вимоги до сховища.

Необхідно враховувати вимоги до сховища робочого навантаження. Наскільки великою має бути база даних, щоб обробляти вимоги до даних? Чи потрібно зберігати гігабайти, терабайти чи петабайти? Різні архітектури баз даних підтримують різні максимальні обсяги даних. Деякі проєкти баз даних ідеально підходять для традиційних програм, а інші – для кешування або керування сеансами.

Міркування 3 – характеристики даних.

Необхідно враховувати, що потрібно зберігати, вибираючи базу даних. Основою будь-якого вибору бази даних є характеристики даних, які потрібно зберігати, отримувати, аналізувати та з якими потрібно працювати. Ці характеристики включають модель даних (чи є вона реляційною, структурованою чи напівструктурованою, використовує високо зв’язаний набір даних чи часові ряди?), доступ до даних (як потрібно отримати доступ до своїх даних?), ступінь, до якого потрібні дані з низькою затримкою, та чи є певний розмір запису даних, який мається на увазі.

Міркування 4 – довговічність.

Довговічність даних стосується гарантії того, що дані не будуть втрачені, а доступність даних – можливості отримати доступ до своїх даних. Який рівень довговічності та доступності даних потрібен? Якщо дані, які зберігатимуться, є абсолютно критично важливими для бізнесу, слід обрати рішення для бази даних, яке зберігає кілька надлишкових копій ваших даних у кількох географічно розділених фізичних місцях. Таке рішення зазвичай призводить до збільшення витрат, тому важливо збалансувати потреби бізнесу з міркуваннями вартості. Ще одним важливим фактором є те, чи застосовуються до даних вимоги щодо місця

зберігання даних або нормативні вимоги. Наприклад, чи існують регіональні закони про конфіденційність даних, яких необхідно дотримуватися? Якщо так, необхідно вибрати рішення для бази даних, яке може забезпечити відповідність вимогам.

1.1.2 Реляційні та нереляційні бази даних

Світ змінився [4], і універсальний підхід використання реляційних баз даних для кожної програми більше не працює. Реляційні бази даних все ще відіграють важливу роль у дизайні та функціональності програм, але низка спеціально створених баз даних набирає популярності. Сучасні програми повинні враховувати соціальний, мобільний, Інтернет речей (IoT) та глобальний доступ. Спеціально створені моделі баз даних розробляються з нуля для швидкого та ефективного виконання конкретних функцій, які вимагають ці програми.

У системі керування реляційними базами даних дані зберігаються в табличній формі стовпців та рядків, а запити до даних виконуються за допомогою SQL. Якщо випадок використання добре підходить для суворих правил схеми, де структура схеми добре визначена і не потребує частих змін, реляційна база даних буде гарним вибором. Під час міграції на локальне реляційне робоче навантаження або якщо робоче навантаження включає онлайн-обробку транзакцій, цей тип бази даних підходить. Однак, якщо програмі потрібна надзвичайна ємність для читання/запису, реляційна база даних може бути не найкращим вибором. Реляційна база даних може бути найкращим рішенням з мінімальними зусиллями для багатьох випадків використання.

Транзакції реляційних баз даних відповідають стандарту ACID. Це означає, що вони допомагають забезпечити цілісність даних, надаючи транзакції, які є атомарними, узгодженими, ізольованими та довговічними. Атомарні транзакції означають, що їх слід розглядати як єдину логічну операцію з даними. Узгоджені транзакції завжди діють або поводяться однаково. Ізольовані транзакції означають,

що виконуються одночасно, але стан системи має бути таким самим, як якби транзакції виконувалися послідовно. Довговічна транзакція означає, що після підтвердження транзакції її вплив має бути постійним навіть у разі збою системи.

Нереляційні бази даних іноді називають NoSQL-базами даних. Це спеціально створені бази даних для різних моделей даних, включаючи ключ-значення, графи, документи, дані в пам'яті та пошук. Вони можуть зберігати структуровані дані, такі як записи реляційних баз даних; напівструктуровані дані, такі як документи JSON; та неструктуровані дані, такі як файли фотографій або повідомлення електронної пошти. Цей тип бази даних має гнучкі схеми, де кожен об'єкт може мати різну структуру. Коли потрібен рівень кешування для покращення продуктивності читання, під час зберігання документів JSON або коли потрібне отримання даних за одну мілісекунду, нереляційна база даних є підходящим варіантом. Нереляційні бази даних оптимізовані для певних моделей даних та шаблонів доступу, що допомагає забезпечити вищу продуктивність, а не намагається досягти функціональності реляційних баз даних.

Як реляційні, так і нереляційні бази даних можуть масштабуватися горизонтально та вертикально. Наприклад, Amazon Relational Database Service (Amazon RDS) може мати репліки читання разом з реляційним масштабуванням горизонтально. Redis може масштабуватися вертикально з більшими типами екземплярів.

1.1.3 Параметри бази даних Amazon

Основним варіантом реляційної служби баз даних від AWS є Amazon RDS. Це керована служба [5], яку можна використовувати для налаштування, експлуатації та масштабування реляційної бази даних у хмарі. З Amazon RDS можна вибрати Amazon Aurora з сумісністю з MySQL, Amazon Aurora з сумісністю з PostgreSQL, Amazon RDS для MySQL, Amazon RDS для MariaDB, Amazon RDS для PostgreSQL, Amazon RDS для Oracle та Amazon RDS для SQL Server. Amazon

RDS використовується для транзакційних програм, таких як планування ресурсів підприємства (ERP), управління взаємовідносинами з клієнтами (CRM) та програми електронної комерції для зберігання структурованих даних.

Існує кілька варіантів нереляційної служби баз даних. Amazon DynamoDB для баз даних типу «ключ-значення», Amazon Neptune для графових баз даних та Amazon ElastiCache для баз даних у пам'яті – це лише деякі з них. Це спеціально створені бази даних, призначені для швидкого та ефективного виконання конкретних функцій, необхідних цим програмам.

У табл. 1.1 наведено завдання, за які ви відповідаєте, залежно від місця розміщення бази даних.

Таблиця 1.1 - Менше відповідальності з керованими службами баз даних AWS

Завдання	Розміщення бази даних локально	База даних хостів в Amazon EC2	База даних хоста в керованій службі баз даних AWS
Електроенергія, опалення, вентиляція та кондиціонування повітря та мережа	x		
Стійка та стек	x		
Обслуговування сервера	x		
Встановлення ОС	x		
Патчі для ОС	x		
Встановлення бази даних	x	x	
Патчі бази даних	x	x	
Резервні копії баз даних	x	x	
Висока доступність	x	x	
Масштабування	x	x	
Оптимізація застосунків	x	x	x

Коли ваша база даних знаходиться локально, адміністратор бази даних відповідає за все: від оптимізації програм і запитів до налаштування обладнання. Адміністратор також займається встановленням виправлень для обладнання та

налаштуванням мережі, живлення, опалення, вентиляції та кондиціонування повітря HVAC.

Якщо ви переходите до бази даних, яка працює на екземплярі Amazon EC2, ви більше не керуєте базовим обладнанням або операціями центру обробки даних. AWS займається обслуговуванням фізичного середовища центру обробки даних, а операційна система (ОС) попередньо встановлена на екземплярі EC2, який ви запускаєте. Однак ви все ще відповідаєте за масштабування, встановлення виправлень для ОС та керування всім програмним забезпеченням і операціями резервного копіювання.

Якщо база даних розміщена в керованій базі даних AWS, такій як Amazon RDS або Aurora, AWS відповідає за обробку поширених і повторюваних завдань адміністрування бази даних. Ви відповідаєте лише за оптимізацію запитів і забезпечення ефективної роботи рівня бази даних для вашої програми. Ці рішення забезпечують високу доступність, масштабованість та резервне копіювання бази даних як вбудовані опції, які ви можете налаштувати.

Ваша відповідальність зменшується, коли ви переходите від розміщення бази даних локально до її розміщення в керованих службах баз даних AWS.

1.1.4 Планування ємності бази даних

Планування потужності бази даних – це процес, у якому враховується поточна та майбутня потужність під час вибору та оновлення ресурсів бази даних. Мета полягає в коригуванні та оптимізації ресурсів бази даних на основі моделей використання та прогнозів. Необхідно постійно контролювати, чи може існуюча інфраструктура витримати очікуване навантаження [6]. Якщо ні, потрібно буде оцінити динаміку витрат на масштабування інфраструктури.

Існує два способи масштабування баз даних: вертикальний та горизонтальний. Вертикальне масштабування передбачає розширення ресурсів, які використовує існуючий сервер, для збільшення його потужності. Цей складний та

трудомісткий процес включає оновлення пам'яті, сховища або обчислювальної потужності. Зазвичай це означає, що база даних буде недоступна протягом певного періоду часу. Горизонтальне масштабування передбачає збільшення кількості серверів, на яких працює база даних, що зменшує навантаження на сервер. Обчислювальна потужність додається під час роботи бази даних, що зазвичай означає, що це масштабування відбувається без простою.

1.2 Створення примірника RDS

Перше завдання в цій роботі — створити екземпляр RDS.

Екземпляр RDS було створено в консолі Amazon RDS з наступними специфікаціями.

У розділі параметрів рушія бази даних **Engine options** для типу рушія було вибрано MariaDB. Версія рушія: 8.4.8.

Для шаблонів обрано «Розробка/Тестування».

У розділі **Settings** налаштовано такі параметри: ідентифікатор **DB instance identifier** екземпляра бази даних **countries**; ім'я **Master username** головного користувача **admin**; пароль головного користувача та підтвердження пароля головного користувача **WhMn?amlypgZFdU1:I7kN#i6$770**.

У розділі конфігурації екземпляра **Instance configuration** для класу екземпляра бази даних **DB instance class** виберано **Burstable classes (includes t classes)**, а потім вибрано **db.t3.micro**.

У розділі сховищ налаштовано такі параметри: тип сховища **General Purpose SSD(gp2)**; виділене сховище 20 ГіБ.

Для параметра доступності і довговічності **Availability & durability** вибрано «Не створювати резервний екземпляр».

У розділі підключень **Connectivit**y налаштовано такі параметри: віртуальна приватна хмара VPC **Project VPC**; група підмережі баз даних **Example-DB-subnet-group**; публічний доступ обрано «Ні»; існуюча група безпеки VPC вибрано

ExampleDB-SG та очищена група безпеки за замовчуванням; зона доступності **us-east-1a**; для порту бази даних залишено порт TCP за замовчуванням 3306.

У розділі моніторингу очищено параметр **Enable Enhanced Monitoring**.

В завершення обрано **Create database**, результат створення бази даних Amazon RDS наведено на рис. 1.1.

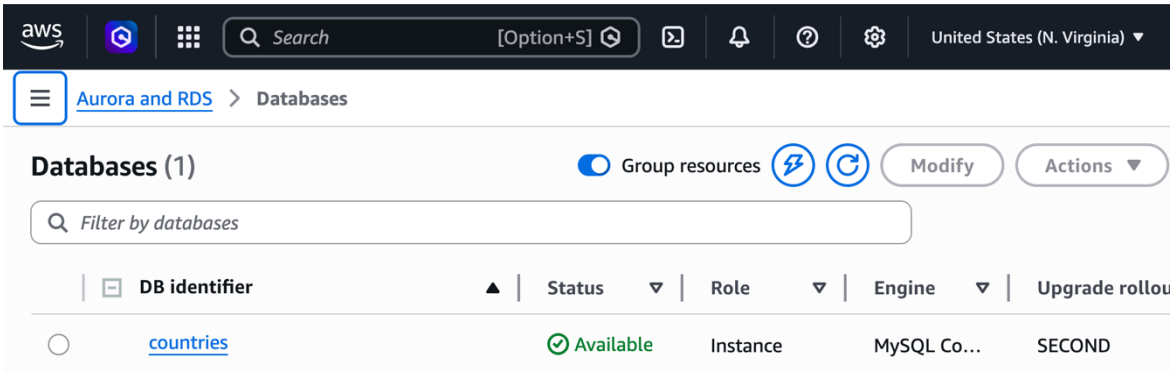

Рисунок 1.1 – База даних **countries**.

1.3 Висновки з аналізу відомої архітектури

Вибираючи базу даних, необхідно враховувати масштабованість, вимоги до сховища, характеристики даних, вартість та вимоги до довговічності. Планування ємності бази даних також є важливим фактором [7].

Реляційні бази даних мають суворі правила схеми, масштабуються вертикально, забезпечують цілісність даних і добре працюють зі структурованими даними, що зберігаються в таблицях.

Нереляційні бази даних мають гнучкі схеми, масштабуються горизонтально, забезпечують вищу масштабованість і гнучкість і добре працюють з напівструктурованими та неструктурованими даними.

Обрано керовану службу бази даних AWS, оптимізації підлягає лише платформа для збору даних та співпраці дослідників.

2 МОДЕЛЬ ВИСОКО ДОСТУПНОГО СЕРЕДОВИЩА

Модель високо доступного середовища побудовано на основі керованого сервісу балансування навантаження програм у доступних публічних підмережах. Ім'я DNS сервісу балансування навантаження використовується для доступу до веб-програми. Використано дві зони доступності.

Створено сервер застосунків у групі автоматичного масштабування. Група автоматичного масштабування використовується для запуску екземплярів EC2 (серверів застосунків), на яких розміщено веб-застосунок. Код PHP (додаток Б) використано щоб встановити потрібний веб-застосунок на віртуальній машині. Шаблон запуску EC2 з назвою Example-LT містить як файл дампа SQL, так і розпаковані ресурси PHP-застосунку, вбудовані в нього. Використовується політика відстеження цілей.

Дані з вихідної бази даних, яка знаходиться на екземплярі EC2, перенесено до нової бази даних Amazon RDS MySQL. Використано доступний файл дампа даних. Щоб отримати ім'я користувача та пароль застосовано диспетчер секретів.

2.1 Високо доступне середовище

Щоб створити екземпляр сервісу балансування навантаження HTTP-застосунків, були налаштовані такі параметри: назва **Load balancer name** екземпляра **Inventory-LB;** віртуальна приватна хмара VPC **Project VPC**; використані дві публічні підмережі; параметри конфігурації безпеки HTTPS:за замовчуванням; група безпеки **ALBSG**; цільова група **Target group** створено **Inventory-App.**

Після активація екземпляра сервісу балансування навантаження, можна отримати його DNS ім'я (рис. 2.1).

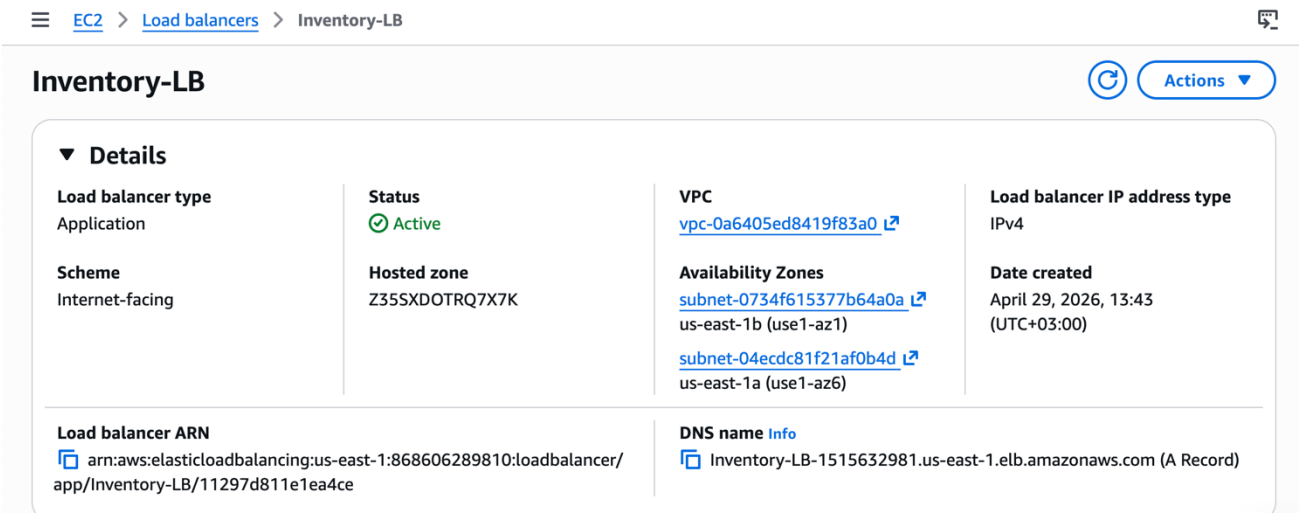

Рисунок 2.1 – DNS сервісу балансування навантаження

2.2 Сервер застосунків у групі автоматичного масштабування

Створено групу автоматичного масштабування.

Для кроку 1. Вибрано шаблон запуску та конфігурацію, налаштовано такі параметри: назва групи автоматичного масштабування – **Inventory-ASG**; шаблон запуску – Inventory-LT, створення якого описано вище.

Для кроку 2. Вибрано параметри запуску екземпляра, налаштовано такі параметри: віртуальна приватна хмара – **Project VPC**; зони доступності та підмережі – приватна підмережа 1 та приватна підмережа 2 (ці параметри запускають екземпляри EC2 у приватних підмережах в обох зонах доступності).

Для крок 3. Налаштовано додаткові параметри.

У розділі балансування навантаження вибрано підключитися до існуючого сервісу балансування навантаження.

У розкривному списку існуючих цільових груп балансування навантаження вибрано Inventory-App.

У розділі перевірки справності налаштовано такі параметри: вибрано увімкнути перевірки справності еластичного балансування навантаження; для періоду пільги перевірки справності **Health check grace period** введено 90 секунд;

У розділі додаткових налаштувань вибрано **Enable group metrics collection within CloudWatch** (цей параметр фіксує метрики з інтервалом у 1 хвилину, що дозволяє автоматичному масштабуванню швидко реагувати на зміну моделей використання).

Для кроку 4. Налаштовано розмір групи та політики масштабування.

Для розміру групи вибрано 2 як бажана ємність.

Для максимальної бажаної ємності налаштовано такі параметри: **Min desired capacity** 2**; Max desired capacity** 6 (ці параметри дозволяють автоматичному масштабуванню автоматично додавати або видаляти екземпляри, завжди залишаючи запущеними 2–6 екземплярів).

Для параметра **Automatic scaling – optional** використано політику відстеження цілей.

Для кроку 5. Додавання сповіщень не налаштовано жодних параметрів.

Для кроку 6: Додано тег та налаштовано такі параметри: ключ **Name**; значення **Inventory-App**.

Ці параметри позначають групу автоматичного масштабування назвою, яка також відображається в екземплярах EC2, запущених групою автоматичного масштабування. Можна використовувати теги, щоб визначити, які екземпляри EC2 пов’язані з якою програмою. Також можна додати теги, такі як «Центр витрат», щоб призначити витрати програми у файлах виставлення рахунків.

Для кроку 7: Переглянуто деталі групи автоматичного масштабування та вибрано **Create Auto Scaling group**.

Група автоматичного масштабування **Inventory-ASG** відображається в консолі (рис. 2.2).

Група автоматичного масштабування спочатку показує кількість екземплярів, що дорівнює нулю, але нові екземпляри запускаються, щоб досягти бажаної кількості екземплярів,.

У стовпцях відображається така інформація:

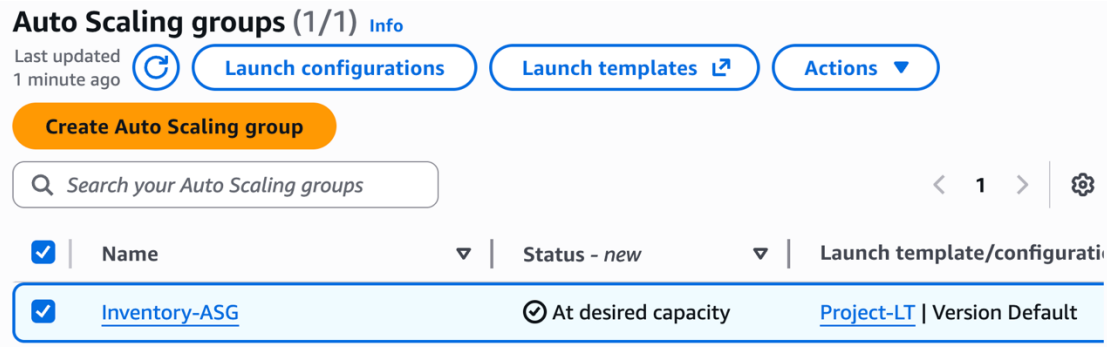

Рисунок 2.2 – Група автоматичного масштабування.

Коли група не має екземплярів, стовпець «Стан» **Status** вказує на «Оновлення ємності».

Бажана ємність становить 2 екземпляри. Amazon EC2 Auto Scaling намагається запустити два екземпляри, щоб досягти бажаної кількості.

Мінімальна та максимальна кількості встановлені на 2 та 6 екземплярів. Amazon EC2 Auto Scaling намагається завжди надавати два екземпляри, навіть якщо трапляється збій.

Програма працює у двох зонах доступності. Amazon EC2 Auto Scaling підтримує цю конфігурацію, навіть якщо екземпляр або зона доступності виходять з ладу [8].

2.3 Оновлення груп безпеки

Трирівнева архітектура використовується в розгорнутій моделі. Групи безпеки налаштовані для забезпечення дотримання цих рівнів (рис. 2.3).

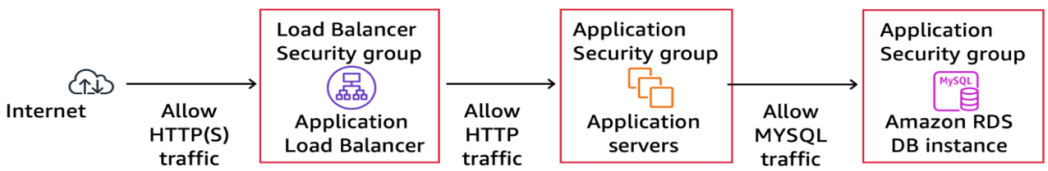

Рисунок 2.3 – Групи безпеки.

Групу безпеки **ALBSG** балансування навантаження налаштовано під час його створення. Він приймає весь вхідний HTTP- та HTTPS-трафік. Сервіс балансування навантаження налаштовано на пересилання вхідних запитів до цільової групи. Коли Auto Scaling запускає нові екземпляри, він автоматично додає ці екземпляри до цільової групи.

Група безпеки програм була надана як частина налаштування проєкту. Потім вона була налаштована на прийом лише вхідного трафіку від сервісу балансування навантаження. У лівій панелі навігації вибрано **Security Groups**, **Inventory-App**, вкладку вхідних правил **Inbound rules**. В порожню група безпеки додано правило для прийому вхідного HTTP-трафіку від сервісу балансування навантаження, не потрібно було налаштовувати HTTPS-трафік, оскільки сервіс балансування навантаження було налаштовано на пересилання HTTPS-запитів через HTTP. Ця практика перенесла безпеку на сервіс балансування навантаження, зменшуючи обсяг роботи, необхідної для окремих серверів програм. На сторінці редагування правил вхідного зв'язку вибрано додавання правил та налаштовано такі параметри: **Type – HTTP**; **Port** – **80**; **Source** –**ALBSG**; **Description** = **Traffic from load balancer.** Сервери додатків тепер можуть отримувати трафік від сервісу балансування навантаження. Це включає перевірки справності, які сервіс балансування навантаження виконує автоматично.

Групу безпеки бази даних налаштовано на прийом лише вхідного трафіку від серверів додатків. У списку груп безпеки вибрано **ExampleDB-SG** і не вибрано жодних інших груп безпеки. Існуюче правило дозволяє трафік на порту 3306 (використовується MySQL) з будь-якої IP-адреси в межах VPC. Це гарне правило, але безпеку можна додатково обмежити. На вкладці правил вхідного зв'язку виберано редагування правил вхідного зв'язку та налаштовано такі параметри: для існуючого правила вибрано **Delet**e; далі вибрано **Add rule**; **Type** – **MYSQL/Aurora**; **Source** – **Inventory-App**.

Таким чином налаштована трирівнева безпеку. Кожен елемент на рівні приймає трафік лише з рівня вище. Крім того, використання приватних підмереж

означає, що є два бар’єри безпеки між Інтернетом та ресурсами програми. Ця архітектура відповідає найкращій практиці застосування кількох рівнів безпеки.

2.4 Імпорт даних до бази даних RDS MySQL

Наразі сайт використовує один екземпляр Amazon EC2 для розміщення свого веб-сервера, бази даних та коду програми. Тим часом данні, що зберігається в базі даних, надають цінну бізнес-інформацію, яку персонал не хоче втрачати. Крім того є додаткові занепокоєння. Базу даних потрібно постійно оновлювати та виправляти, і в персоналу не завжди є час на виконання цих завдань. Крім того, адміністрування бази даних – це спеціалізована навичка [9]. Навчання інших адмініструванню бази даних – це не те, на що персонал хоче витрачати час [10]. Тим часом також непокоїть те, що персонал не робить резервних копій даних так часто, як мало б [11]. Нарешті, власники сайту також хочуть зменшити витрати на робочу силу, пов'язані з інвестиціями в технічне навчання, необхідними для управління базою даних [12]. У цьому підрозділі описана моделі переносу даних з бази даних на екземплярі EC2 до Amazon RDS [13]. Зокрема, виконано перенос бази даних MariaDB, яка працює на екземплярі EC2, до бази даних MariaDB, яка працює на Amazon RDS.

2.4.1 Робота з базою даних на екземплярі EC2

У цьому пункті наведені спостереження деталей бази даних MariaDB, яка працює на екземплярі EC2 та експорту існуючих даних з бази даних за допомогою утиліти mysqldump.

У консолі Amazon EC2, у лівій навігаційній панелі, вибрано **Instances**, а потім вибрано екземпляр EC2 **Inventory-App**.

Щоб відкрити меню «Підключитися до екземпляра», вибрано **Connect**.

Вибрано диспетчер сеансів **SSM Session Manager**. Вибрано **Connect**.

Відкрита нова вкладка браузера з термінальним сеансом, підключеним до екземпляра EC2.

У командному рядку введено такі команди (рис. 2.4).

```
bash
sudo su
su ec2-user
whoami
cd /home/ec2-user/
```

Рисунок 2.4 – Виклик оболонки bash.

Перша команда відкриває оболонку bash. Друга команда перемикає сеанс на використання облікового запису користувача root на екземплярі EC2. Третя команда перемикає на використання облікового запису ec2-user. Четверта команда має повернути вивід, який підтверджує, що ви підключені як ec2-user. Остання команда перемикає термінал на домашній каталог ec2-user (рис. 2.5).

```
sh-5.2$ bash
[ssm-user@ip-10-0-3-231 bin]$
[ssm-user@ip-10-0-3-231 bin]$ sudo su
[root@ip-10-0-3-231 bin]# su ec2-user
[ec2-user@ip-10-0-3-231 bin]$ whoami
ec2-user
[ec2-user@ip-10-0-3-231 bin]$ cd /home/ec2-user/
[ec2-user@ip-10-0-3-231 ~]$
```

Рисунок 2.5 – Домашній каталог ec2-user.

Агент **Systems Manager** (ssm agent) інстальовано за замовчуванням на всіх екземплярах Amazon Linux 2 (і деяких інших типах операційних систем). Під час запуску роботи та створення екземпляру EC2, дані користувача вказували на те, що на екземплярі має бути запущено службу **агента ssm**. Також до екземпляра EC2 було додано роль AWS Identity and Access Management (IAM), яка включає

політику IAM під назвою AmazonSSMManagedInstanceCore. Ці дві дії зробили екземпляр доступним через Systems Manager Session Manager.

Щоб переглянути деталі бази даних, яка працює на екземплярі EC2, у терміналі виконані такі команди (рис. 2.6).

```
service mariadb status
mysql --version
```

Рисунок 2.6 – Версія бази даних.

Вивід показав, що локально встановлена база даних MariaDB на цьому екземплярі EC2 працює. Також відображається номер версії бази даних.

У консолі керування AWS у полі пошуку було введено **Secrets Manager**, щоб відкрити консоль менеджера секретів AWS.

У лівій навігаційній панелі вибрано **Secrets**. Тут зберігаються секрети. PHP-код програми посилається на ці значення для отримання інформації про підключення до бази даних.

У вкладці **Overview** у розділі **Secret value** виберіть **Retrieve secret value** та скопійовано значення секрету **Secret value** для паролю *password* в буфер обміну.

Щоб підключитися до бази даних, яка працює на екземплярі EC2, у вкладці браузера за допомогою терміналу bash виконано таку команду (рис. 2.7)

```
mysql -u root -p
```

Рисунок 2.7 – Підключення до бази даних на екземплярі EC2.

Коли було запропоновано ввести пароль бази даних, було вставлене скопійоване значення параметра *password*.

Запит **MariaDB>** вказує на підключення до бази даних MariaDB, яка працює на цьому екземплярі EC2 (рис. 2.8).

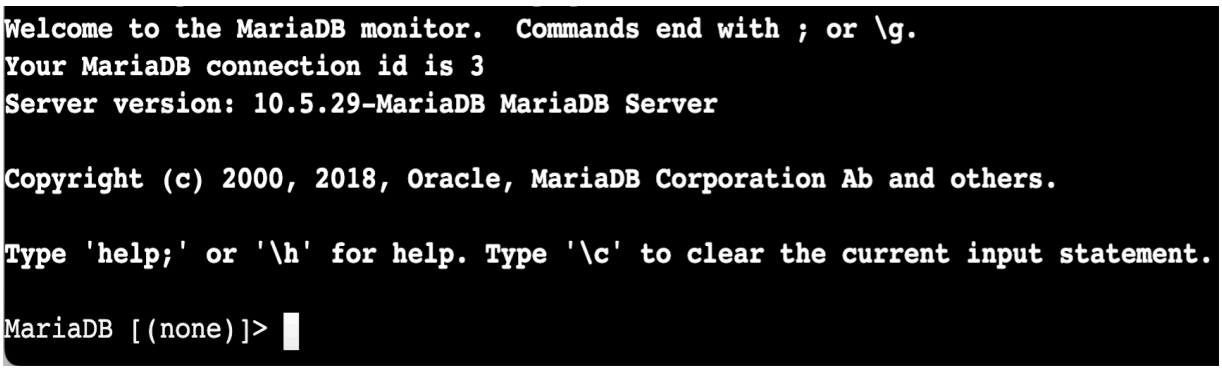

Рисунок 2.8 – Сервер бази даних на екземплярі EC2.

Щоб переглянути дані в існуючій базі даних, виконано такі команди (рис. 2.9).

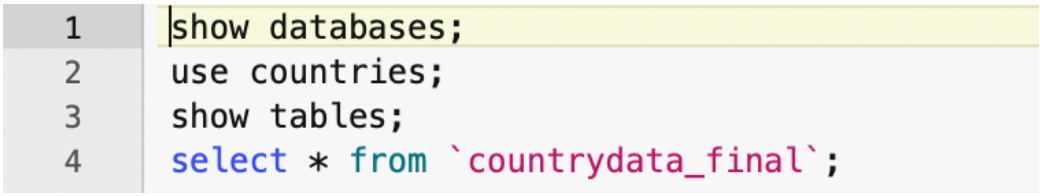

Рисунок 2.9 – Перегляд даних таблиці countrydata_final .

Зокрема, переглянуто таблиці, що підтримують веб-застосунок.

Щоб вийти з клієнта SQL, було введено таку команду: **exit**. Щоб записати існуючі дані у файл за допомогою утиліти **mysqldump**, виконано таку команду: **mysqldump --databases countries -u root -p > Countrydatadump.sql.**

Коли буде запропоновано ввести пароль бази даних, вставте значення *password* із секретів менеджера секретів [14]. Щоб підтвердити успішне виконання **mysqldump**, виконано команду **ls** у терміналі. Вивід показав, що файл **> Countrydatadump.sql** створено. Щоб переглянути вміст файлу, виконано команду **cat Countrydatadump.sql**.

У наступному пункті описаний імпорт цих даних до нової бази даних RDS.

2.4.2 Робота з базою даних RDS

У консолі керування AWS обрано консоль Aurora and RDS і де створений раніше екземпляр RDS **countries** є доступний [15].

Екземпляр RDS працює на рівні зони доступності **us-east-1a.** Екземпляру RDS не призначено публічну IPv4-адресу. Значення **Private DB Subnet 1** тегу імені застосовується до підмережі, в якій працює екземпляр RDS. Його можна знайти у вкладці браузера в консолі VPC. На панелі підключень та безпеки консолі RDS відображається ідентифікатор цієї підмережі **subnet-0613d36b077909a8d**.

Далі було встановлене мережеве з'єднання з терміналу, що працює на екземплярі EC2, до нового екземпляра RDS.

Ось синтаксис, який використано для підключення: **mysql -u admin -p --host <rds-endpoint**>.

Значення <rds-endpoint> було замінено фактичною кінцевою точкою RDS для екземпляра RDS - **countries.cdqopvwt82mc.us-east-1.rds.amazonaws.com.**

Після виконання команди було введено пароль для екземпляра RDS. Цей пароль було визначено під час створення екземпляра RDS.

Правила вхідного зв'язку групи безпеки, в якій працює екземпляр RDS, були оновлені, як це описано в попередньому підрозділі. Клієнтське програмне забезпечення MySQL намагається підключитися до бази даних через TCP-порт 3306.

Порт 3306 не відкрито для всіх вихідних IP-адрес. Це було б небезпечно [16]. Натомість його відкрито лише для серверів у групі безпеки, яка використовується екземпляром EC2, з якого виконується підключення. Введення **sg-** у поле джерела, використовувалось, щоб переглянути варіанти.

Підтверджено, що налаштування групи безпеки дозволяють трафік через TCP-порт 3306 від екземпляра EC2 до бази даних [17]. Ці команди виконано в терміналі **Systems Manager Session Manager**. У наступній команді замінено <rds-endpoint> на фактичну кінцеву точку Amazon RDS: **nmap -Pn <rds-endpoint>**.

Вивід команди показує, що порт 3306 відкритий для служби MySQL, це підтверджує, що налаштування групи безпеки дозволяють трафік.

Команда **nmap** показала, що порт відкритий, тому наступна команда mysql -u admin -p --host <rds-endpoint> також успішно відпрацювала. Що

дозволило виконати команду **show databases;**! Відобразився такий вивід (рис. 2.10).

```
Type 'help;' or '\h' for help. Type '\c' to clear the current input statement.

MariaDB [(none)]> show databases;
+--------------------+
| Database           |
+--------------------+
| information_schema |
| innodb             |
| mysql              |
| performance_schema |
+--------------------+
4 rows in set (0.00 sec)

MariaDB [(none)]>
```

Рисунок 2.10 – Результат виконання команди **show databases**.

Зверніть увагу, що бази даних **countries** ще немає у списку [18]. Це очікувана ситуація, оскільки ще не імпортовано жодних даних. Щоб відключитися, виконано команду **exit;**.

Щоб імпортувати дані, експорт яких описано в попередньому пункті, до екземпляра бази даних RDS, у наступній команді замінено фактичну кінцеву точку: **mysql -u admin -p --host <rds-endpoint> < Countrydatadump.sql.**

У запиті пароля введено *password* для екземпляра RDS.

Щоб підтвердити імпорт даних та підключитися до бази даних RDS, виконано таку команду: **mysql -u admin -p --host <rds-endpoint>**.

У запиті пароля введено *password* для екземпляра RDS.

Щоб підтвердити імпорт даних, виконано такі команди (рис. 2.11).

```
1  show databases;
2  use countries;
3  show tables;
4  select * from `countrydata_final`;
```

Рисунок 2.11 – Перегляд даних таблиці в базі даних RDS.

Вивід оператора **select** відображає таблицю, що підтримує веб-застосунок. Щоб вийти з клієнта SQL, виконано таку команду: **exit;**.

3 РЕЗУЛЬТАТИ ТЕСТУВАННЯ ПЛАТФОРМИ ТА ПЕРЕВІРКА ІМПОРТОВАНИХ ДАНИХ

Щоб протестувати платформу, веб-програма відкрита з браузера, використовуючи URL-адресу сервісу балансування навантаження [19]. Щоб переглянути та запитати імпортовані дані, використовувалась функція запитів у програмі [20].

3.1 Вхідні данні

Початкова архітектура проєкту наведена на рис. 3.1.

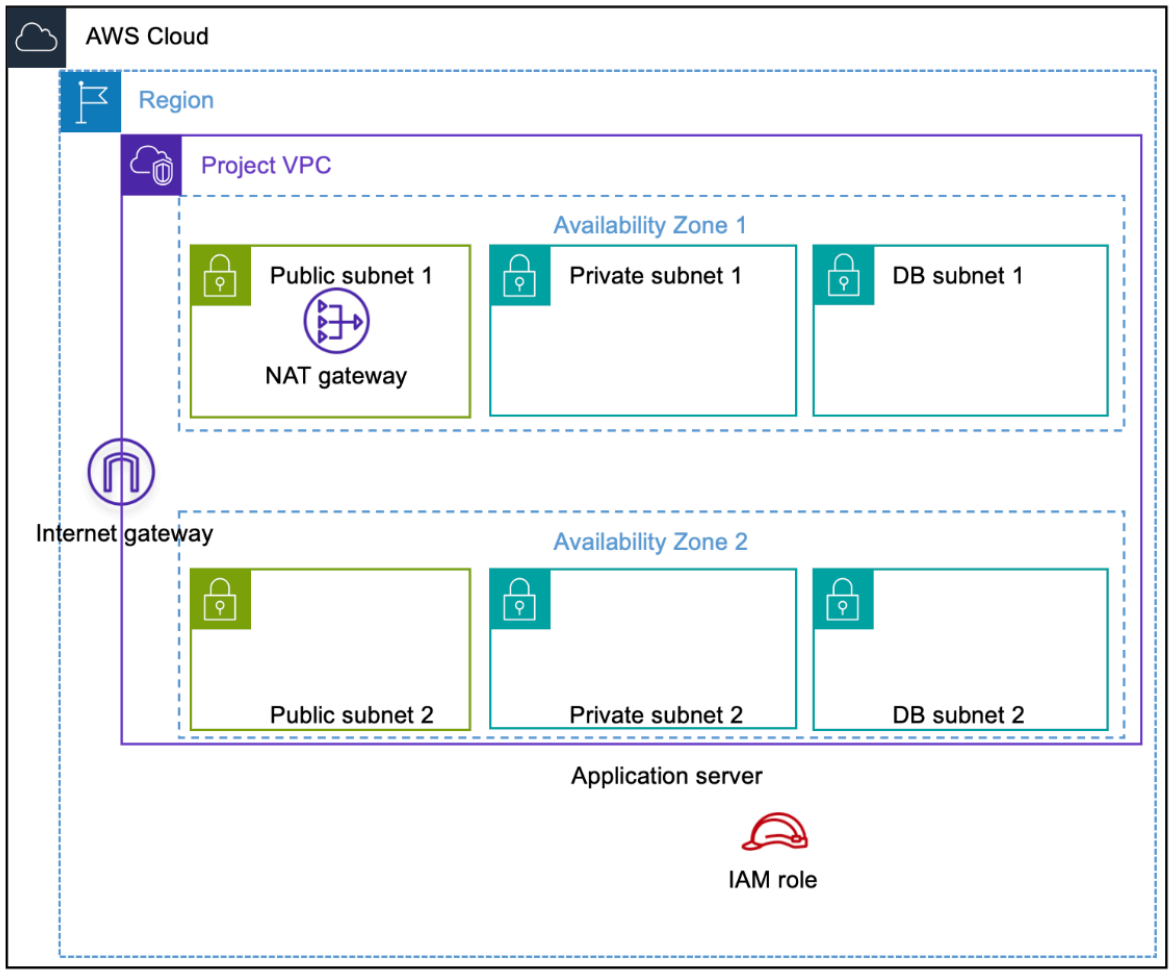

Рисунок 3.1 - Початкова архітектура проєкту

Проєкт побудовано на базі архітектури (рис. 3.2), яка має декілька зон доступності, між якими розподілені декілька підмереж, в одній знаходиться база даних, а в іншій група підмережи, яка пов’язана з цією базою даних.

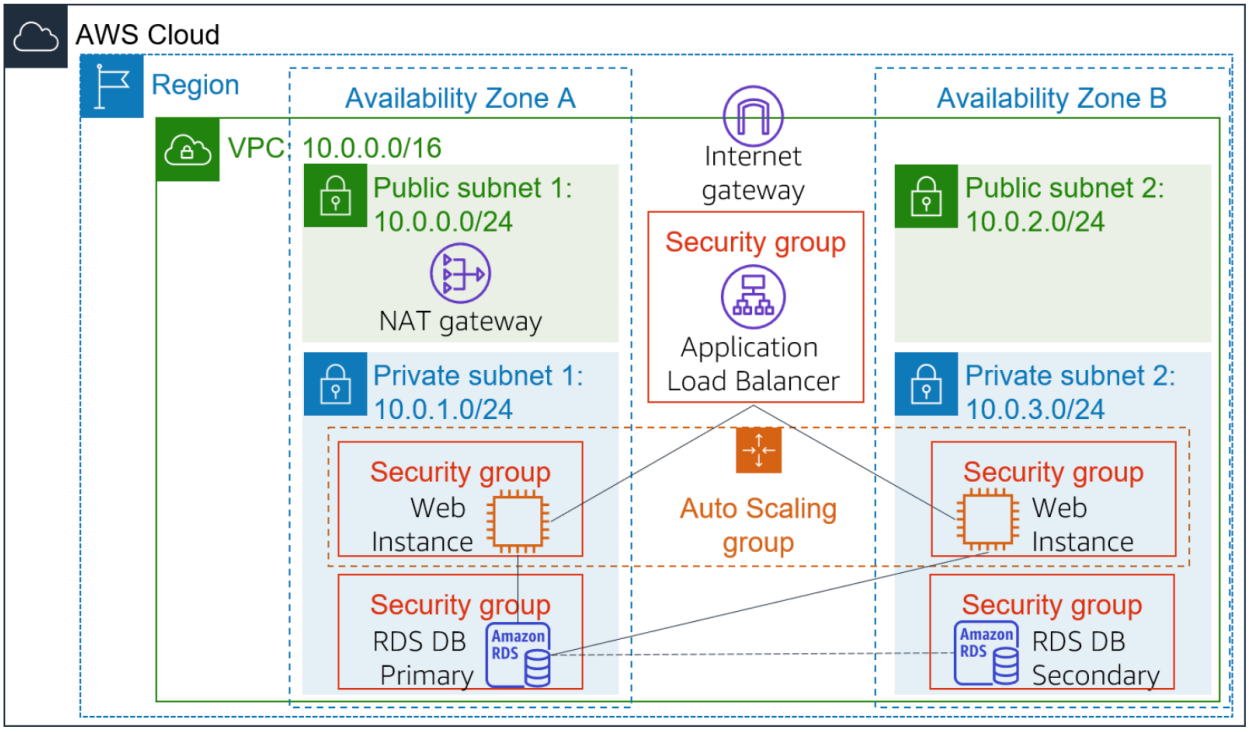

Рисунок 3.2 - Фінальна архітектура проєкту

Підсумовуючи, у цій роботі досліджувалося, як: вказати, як розподіляти трафік між екземплярами Amazon EC2 за допомогою Elastic Load Balancing; визначити, як Amazon CloudWatch дозволяє контролювати ресурси та програми AWS у режимі реального часу; пояснити, як Amazon EC2 Auto Scaling запускає та випускає сервери у відповідь на зміни робочого навантаження; виконувати завдання масштабування та балансування навантаження для покращення архітектури.

3.2 Сторінка запиту даних країни

Веб-сторінка (файл query.php додаток Б) дозволяє користувачам вибирати та виконувати різні запити даних про країни.

3.2.1 Структура розмітки

Розділ заголовку HTML наведено на рис. 3.3.

```
<head>
<meta charset="UTF-8">
<title>Query Page</title>
<link rel="stylesheet" href="css/styles.css">
</head>
```

Рисунок 3.3 – Заголовок

Він встановлює кодування символів UTF-8 (підтримує міжнародні символи), встановлює заголовок вкладки браузера на "Сторінка запиту" та посилається на зовнішній файл CSS для стилізації.

Розділ тіла HTML має наступні елементи.

Перший наведено на рис. 3.4.

```
<div id="header" class="mainHeader">
    <hr>
    <div class="center"><H1>Example Social Research Organization</H1></div>
</div>
```

Рисунок 3.4 – Header

Він створює заголовок з горизонтальною лінією (`<hr>`) та відображає назву організації як заголовок по центру.

Другий наведено на рис. 3.5.

```
<div class="center"><H2>Country Data Query Page</H2></div>
```

Рисунок 3.5 – Заголовок сторінки

Показує, для чого призначена ця сторінка

Третій наведено на рис. 3.6.

```
1   <H2><a href="index.php">Home</a></H2>
```

Рисунок 3.6 – Навігація

Надає посилання на головну сторінку.

Четвертий наведено на рис. 3.7.

```
1  <form id="form1" method="post" action='query2.php'>
2      <select name="selection">
3          <option value="">Select...</option>
4          <option value='Q1'>Mobile phones</option>
5          <option value='Q2'>Population</option>
6          <option value='Q3'>Life Expectancy</option>
7          <option value='Q4'>GDP</option>
8          <option value='Q5'>Childhood Mortality</option>
9      </select>
10     <input type="submit" value="Submit">
11 </form>
```

Рисунок 3.7 – Форма запиту (головна функція)

Головна функція працює так:

- **Форма.** Надсилає дані до `query2.php` за допомогою методу POST;

- **Спадаюче меню**. Користувач вибирає один із 5 варіантів запиту;

- Кнопка **Submit**. Надсилає вибране значення до query2.php для обробки.

П’ять варіантів запитів:

- Q1 = Дані про мобільні телефони;

- Q2 = Дані про населення;

- Q3 = Дані про тривалість життя;

- Q4 = Дані про ВВП (економічні);
- Q5 = Дані про дитячу смертність.

П’ятий наведено на рис. 3.8.

```
<div id="Copyright" class="center">
    <h5>&copy; 2026, Amazon Web Services, Inc. or its Affiliates. All rights reserved.</h5>
</div>
```

Рисунок 3.8 – Нижній колонтитул.

В ньому повідомлення про авторські права.

Підсумок. Це простий інтерфейс запитів, де користувачі: вибирають тип даних про країну, які вони хочуть побачити; натискають "Надіслати"; перенаправляються на `query2.php`, який обробляє їхній вибір і відображає результати

3.3 Динамічна веб-сторінка

Код оператору PHP Switch з динамічними включеннями (файл query2.php додаток Б) створює динамічну веб-сторінку, яка відображає різний вміст на основі вибору користувача.

3.3.1 Огляд структури

Структура файлу наведено на рис. 3.9.

```
<html>
<body>
    <?php ... ?>  <!-- PHP logic -->
    <div>...</div> <!-- Copyright footer -->
</body>
</html>
```

Рисунок 3.9 – Структура файлу.

Цей код — простий PHP-скрипт, вбудований у HTML-документ. Він використовує оператор switch для включення різних PHP-файлів на основі POST-параметра з назвою selection.

3.3.2 Покроковий аналіз

Крок 1 - включення файлу конфігурації на рис. 3.10.

```
include 'get-parameters.php';
```

Рисунок 3.10 – Включення файлу.

Завантажує зовнішній файл PHP (містить підключення до бази даних та налаштування конфігурації).

Крок 2 - запис вибору користувача на рис. 3.11.

```
$_pick = $_POST['selection'];
```

Рисунок 3.11 – Запис вибору користувача.

Отримує дані з HTML-форми, надісланої методом POST, ім'я поля форми - `'**selection**'` та зберігає значення у змінній `$_pick`.

Крок 3 - оператор Switch (логіка прийняття рішень) на рис. 3.12.

```
switch ($_pick) {
    case "Q1":
        include 'mobile.php';
        break;
    // ... more cases
}
```

Рисунок 3.12 – Запис вибору користувача.

Оператор працює так:

- порівнює `$_pick` з кількома можливими значеннями;
- коли знайдено збіг, він включає відповідний PHP-файл;
- `break;` запобігає переходу до наступного випадку.

Крок 4 - п'ять варіантів в табл. 3.1.

Таблиця 3.1 - П'ять варіантів

Вибір користувача	Завантажений файл	Ймовірний вміст
Q1	mobile.php	Статистика мобільних телефонів
Q2	population.php	Дані про населення
Q3	lifeexpectancy.php	Інформація про тривалість життя
Q4	gdp.php	Статистика ВВП
Q5	mortality.php	Рівень смертності

Крок 4 - нижній колонтитул щодо авторських прав на рис. 3.13.

```
<div id="Copyright" class="center">
    <h5>&copy; 2026, Amazon Web Services...</h5>
</div>
```

Рисунок 3.13 – Нижній колонтитул.

Відображає повідомлення про авторські права: `&copy;` = © symbol; `class="center"` пропонує стилізацію CSS (не показано).

Приклад послідовності дій на практиці:

Дія 1. Користувач заповнює форму за допомогою радіокнопок/випадаючих списків (рис. 3.14):

```
<form method="POST">
    <select name="selection">
        <option value="Q1">Mobile Stats</option>
        <option value="Q2">Population</option>
    </select>
    <button type="submit">Submit</button>
</form>
```

Рисунок 3.14 – Нижній колонтитул.

Дія 2. Користувач вибирає "Q2" та надсилає.

Дія 3. Цей код отримує `$_POST['selection'] = "Q2"`.

Дія 4. Оператор switch відповідає `case "Q2"`.

Дія 5. Файл `population.php` включається та виконується.

Результати тестування отримання статистики глобального розвитку з веб-сайту наведені на рис. 3.15.

Pick another query

Country Name	Population	Urban Population
Afghanistan	26697430	5771984
Albania	3071856	1280964
Algeria	30533827	18259229
American Samoa	57625	51171
Andorra	64634	59722
Angola	13926373	6823923
Antigua and Barbuda	77656	24928
Argentina	36930709	33274569
Armenia	3076098	2002540
Aruba	90271	42157
Australia	19153000	16701416
Austria	8011566	5271610
Azerbaijan	8048600	4120883
Bahamas, The	297651	244074

Рисунок 3.15 – Інформація щодо населення країн на літеру А.

Дія 6. Нижній колонтитул з інформацією про авторські права відображається під контентом.

Ключові поняття:

- `**include**` вставляє та виконує інший PHP-файл;

- `**$_POST**`- суперглобальний масив, що містить дані форми;

- `**switch**`-ефективний спосіб обробки кількох умов%

- **Динамічний контент**. Одна й та сама сторінка відображає різні дані залежно від вибору користувача.

Це поширений шаблон для створення простих систем керування контентом або додатків для панелей інструментів!

3.4 Отримання налаштувань із сховища параметрів

Керування метаданими та секретами AWS (файл get-parameters.php додаток Б) реалізовано в PHP-коді, який отримує метадані AWS EC2 та облікові дані бази даних з AWS Secrets Manager. Цей код виконує три основні завдання, а саме отримує: токен автентифікації з сервісу метаданих EC2; зону доступності поточного екземпляра EC2; облікові дані бази даних RDS з AWS Secrets Manager.

3.4.1 Покрокове пояснення

Крок 1. Початкове налаштування (рис. 3.16)

```php
<?php
        # Retrieve settings from Parameter Store
        error_log('Retrieving settings');
        require 'aws-autoloader.php';
```

Рисунок 3.16 – Початкове налаштування.

Записує початок процесу, завантажує AWS SDK для PHP.

Крок 2. Отримання токена метаданих IMDSv2 (рис. 3.17)

```php
        $ch = curl_init();

        // get a valid TOKEN
        $headers = array (
                'X-aws-ec2-metadata-token-ttl-seconds: 21600'
        $url = "http://169.254.169.254/latest/api/token";
        #echo "URL ==> " .  $url;
        curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers );
        curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
        curl_setopt( $ch, CURLOPT_CUSTOMREQUEST, "PUT" );
        curl_setopt( $ch, CURLOPT_URL, $url );
        $token = curl_exec( $ch );
```

Рисунок 3.17 – Токен метаданих.

Мета – отримання токена сеансу для безпечного доступу до метаданих.

Метод – запит PUT до служби метаданих.

Час життя токена – 21600 секунд (6 годин).

Крок 3. Отримання зони доступності (рис. 3.18)

```
22          // then get metadata of the current instance
23          $headers = array (
24                  'X-aws-ec2-metadata-token: '.$token );
25
26          $url = "http://169.254.169.254/latest/meta-data/placement/availability-zone";
27
28          curl_setopt( $ch, CURLOPT_URL, $url );
29          curl_setopt( $ch, CURLOPT_HTTPHEADER, $headers );
30          curl_setopt( $ch, CURLOPT_RETURNTRANSFER, true );
31          curl_setopt( $ch, CURLOPT_CUSTOMREQUEST, "GET" );
32          $result = curl_exec( $ch );
33          $az = curl_exec( $ch );
```

Рисунок 3.18 – Зона доступності.

Використовує токен для автентифікації, отримує зону доступності "us-east-1a".

Крок 4. Отримання регіону (рис. 3.19)

```
37          $region = substr($az, 0, -1);
```

Рисунок 3.19 – Регіон.

Видаляє останній символ з AZ для отримання регіону "us-east-1a" → "us-east-1".

Крок 5. Ініціалізація клієнтів AWS (рис. 3.20)

```
40          $secrets_client = new Aws\SecretsManager\SecretsManagerClient([
41            'version' => 'latest',
42            'region'  => $region
43          ]);
44          #fetch the endpoint ep
45          $rds_client = new  Aws\Rds\RdsClient([
46            'version' => 'latest',
47            'region'  => $region
48          ]);
```

Рисунок 3.20 – Клієнт AWS.

Створює клієнтів для менеджера секретів та служб RDS.

Крок 6. Отримати кінцеву точку RDS (рис. 3.21)

```
49          $dbresult = $rds_client->describeDBInstances();
50          $dbresult = $dbresult['DBInstances'][0]['Endpoint']['Address'];
51          $ep = $dbresult;
```

Рисунок 3.21 – Кінцева точка RDS.

Отримує інформацію про екземпляри RDS, витягує URL-адресу кінцевої точки бази даних.

Крок 7. Отримати облікові дані бази даних (рис. 3.22)

```
54          #fetch secrets for the endpoint
55          $secretresults = $secrets_client->listSecrets(array(
56            ['Key'=>['name'],
57            'Values'=>['rds!']
58            ])
59            );
60            $result = $secrets_client->getSecretValue([
61              'SecretId' => $secretresults['SecretList'][0]['Name'],
62          ]);
63          $result = $result['SecretString'];
64          $result = json_decode($result, true);
65      #       echo $result;
66
67          #$result = $result['SecretString'];
68       #      print($result);
69          #$result = json_decode($result, true);
70          $un = $result['username'];
71          $pw = $result['password'];
72          $db = 'countries';
```

Рисунок 3.22 – Облікові дані бази даних.

Виводить список секретів, що містять "rds!" в назві, отримує значення секрету, парсить JSON для вилучення імені користувача та пароля.

Крок 8. Обробка помилок (рис. 3.23)

```
75          catch (Exception $e) {
76            $ep = '';
77            $db = '';
78            $un = '';
79            $pw = '';
80          }
```

Рисунок 3.23 – Обробка помилок.

Встановлює порожні значення, якщо виникає помилка.

3.4.2 Ключові поняття

Служба метаданих екземплярів EC2 IMDS: спеціальна IP-адреса: `169.254.169.254`; надає інформацію про екземпляр; версія 2 (IMDSv2) вимагає автентифікації за допомогою токена.

Менеджер секретів AWS: безпечно зберігає облікові дані бази даних; автоматично обертає секрети; краще, ніж жорстко закодовані паролі.

Використані параметри **cURL**:

- `CURLOPT_RETURNTRANSFER`: Повертає відповідь у вигляді рядка;
- `CURLOPT_CUSTOMREQUEST`: Встановлює метод HTTP (PUT/GET);
- `CURLOPT_HTTPHEADER`: Додає власні заголовки.

Демонстрація найкращих практик безпеки:

- використовує imdsv2 (автентифікація на основі токена);
- отримує облікові дані з Менеджера секретів;
- жорстко закодовані облікові дані відсутні;
- обробка помилок за допомогою try-catch.

3.5 Вартість використання ресурсів

Ось приклад (рис. 3.24), який показує вартість використання ресурсів у регіоні us-east-1 протягом місяця та цілого року.

Estimate summary				
Upfront cost	Monthly cost	Total 12 months cost	Currency	
0	87.662	1051.944	USD	
Region	US East (N. Virginia)			
Detailed Estimate				
Group	Description	Service	Monthly	Configuration summary
My Estimate	database	Amazon RDS for MySQL	29.42	Storage amount (20 GB), Storage for each RDS instance (General Purpose SSD (gp2)), Nodes (1), Instance type (db.t3.micro), Utilization (On-Demand only) (100 %Utilized/Month), Deployment option (Multi-AZ), Pricing strategy (OnDemand)
My Estimate	Application Load Balancer	Application Load Balancer	16.93	Number of Application Load Balancers (1)
My Estimate	Application Server	Amazon EC2	7.592	Tenancy (Shared Instances), Operating system (Linux), Workload (Consistent, Number of instances: 2), Advance EC2 instance (t3.micro), 50% utilization
My Estimate	Nat Gateway	Network Address Translation (NAT) (	33.3	Number of NAT Gateways (1)
My Estimate	Secrets Manager	AWS Secrets Manager	0.42	Number of secrets (1), Average duration of each secret (30 days), Number of API calls (100 per day)

Рисунок 3.24 - Вартість використання ресурсів.

ВИСНОВКИ

У ході виконання кваліфікаційної роботи магістра було реалізовано безпечну платформу на базі AWS для збору даних та співпраці дослідників.

Цей проєкт створено в контрольованому лабораторному середовищі з обмеженнями щодо послуг, функцій та бюджету. Розглянуто такі припущення для проєкту.

Додаток розгорнуто в одному регіоні AWS (рішення не обов'язково має бути багаторегіональним).

Веб-сайт не обов'язково має бути доступним через HTTPS або користувацький домен.

Рішення розгорнуто на машинах Amazon Linux 2023 за допомогою наданого PHP-коду. Шаблон для запуску екземплярів EC2 вже доступний.

Рішення використовує наданий PHP-код.

Рішення використовує служби та функції в межах обмежень доступного середовища.

Рішення використовує три пари підмереж.

Рішення використовує доступну роль AWS IAM.

База даних розміщена в двох зонах доступності.

Інформацію про підключення до бази даних збережено у Secrets Manager.

Веб-сайт є загальнодоступним без автентифікації.

Найкращою практикою безпеки є дозвіл на доступ до веб-сайту через автентифікацію. Однак, ці функції виходять за рамки цього проєкту.

ПЕРЕЛІК ДЖЕРЕЛ

[1] 30-й Міжнародний молодіжний форум «Радіоелектроніка та молодь у XXI столітті». Зб. Матеріалів форуму. Т.4. / [Електронний ресурс] – Харків: ХНУРЕ. 2026. – 200 с. – pdf 10,1 Mb.

[2] Amazon Web Services. Amazon Simple Queue Service Developer Guide.[Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/AWSSimpleQueueService/latest/SQSDeveloperGuide/welcome.html.

[3] Wicherski, M. "Building Event-Driven Microservices: Leveraging Organizational Data at Scale." – O'Reilly Media, 2021. – 300 с.

[4] Fowler, M. "Patterns of Enterprise Application Architecture." – Addison-Wesley Professional, 2002. – 560 с.

[5] Villamizar, M., et al. "Evaluating the Monolithic and the Microservice Architecture Pattern to Deploy Web Applications in the Cloud." – 2015 IEEE 8th International Conference on Cloud Computing. – pp. 931-938.

[6] Adzic, G., Chatley, R. "Serverless Computing: Economic and Architectural Impact." – ACM Digital Library, 2017. – 5(2): 409-415.

[7] Sharma, S. "Serverless Computing: A Comprehensive Guide to Architecting Serverless Applications." – Springer, 2020. – 230 с.

[8] Amazon Web Services. AWS Lambda Developer Guide. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/lambda/latest/dg/welcome.html.

[9] Amazon Web Services. Amazon DynamoDB Developer Guide. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/amazondynamodb/latest/developerguide/Introduction.html.

[10] Eyk, E. van, et al. "A Serverless Ecosystem for Edge Computing." – IEEE Internet Computing, 2018. – 22(4): 16-22.

[11] Roberts, M., Chapin, J. "Programming AWS Lambda." – O'Reilly Media, 2020. – 250 с.

[12] Amazon Web Services. AWS Well-Architected Framework. [Електронний ресурс]. – Режим доступу: https://aws.amazon.com/architecture/well-architected/.

[13] Amazon Web Services. Amazon S3 User Guide. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/AmazonS3/latest/userguide/Welcome.html.

[14] Amazon Web Services. Amazon SNS Developer Guide. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/sns/latest/dg/welcome.html.

[15] Glikson, A., Nellis, S. "AWS for Solutions Architects: Design and implement AWS solutions for enterprise applications." – Packt Publishing, 2021. – 400 с.

[16] Behrendt, M., et al. "AWS Lambda in Action: Event-Driven Serverless Applications." – Manning Publications, 2019. – 320 с.

[17] Bass, L., Weber, I., Zhu, L. "DevOps: A Software Architect's Perspective." – Addison-Wesley Professional, 2015. – 240 с.

[18] Amazon Web Services. Amazon RDS User Guide. [Електронний ресурс]. – Режим доступу: https://docs.aws.amazon.com/AmazonRDS/latest/UserGuide/Welcome.html.

[19] Kumar, P. "Mastering AWS Lambda: Learn how to build and deploy serverless applications." – Packt Publishing, 2020. – 320 с.

[20] Parikh, J. "Implementing Serverless Architectures with AWS: Harnessing AWS for Scalable Cloud Applications."– Packt Publishing, 2021. – 350 с.