

Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
(повна назва)

Кафедра Інформаційно-мережної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Аналіз управління використанням ресурсів на базі хмарної платформи Amazon
Web Services
(тема)

Виконав:

Здобувач 4 року навчання,
групи ТРИМІЗ-22-1
Софія Таран
(власне ім'я, прізвище)

Спеціальність 172 Телекомунікації та
радіотехніка
(код і повна назва спеціальності)

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

Освітня програма Інформаційно-
мережна інженерія
(повна назва освітньої програми)

Керівник проф. Валерій Безрук
(посада, власне ім'я, прізвище)

Допускається до захисту

В.о. завідувача кафедри _____

(підпис)

Дарія Чеботарьова
(власне ім'я, прізвище)

2026

Харківський національний університет радіоелектроніки

Навчально-науковий центр заочної форми навчання
Кафедра Інформаційно-мережної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 172 Телекомунікації та радіотехніка
(код і повна назва)
Тип програми освітньо-професійна
(освітньо-професійна або освітньо-наукова)
Освітня програма Інформаційно-мережна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:

Зав. кафедри _____
(підпис)
«18» 05 2026 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Таран Софії Вікторівні
(прізвище, ім'я, по батькові)

1. Тема роботи Аналіз управління використанням ресурсів на базі хмарної платформи Amazon Web Services

затверджена наказом університету від 18 05 2026 р. № 73Стз

2. Термін подання здобувачем роботи до екзаменаційної комісії 13 07 2026 р.

3. Вихідні дані до роботи Дослідити utilization management, провести аналіз медичних інформаційних систем. Розробити інфраструктуру за допомогою IaC підходу у хмарній платформі AWS.

4. Зміст пояснювальної записки (перелік питань, які потрібно розробити)

Вступ

1. Теоретико-методологічні основи Utilization Management в культурі хмарних обчислень

2. Архітектурний аналіз інформацийних систем як об'єкта управління хмарними ресурсами

3. Порівняльний аналіз архітектурних вимог та моделей розгортання національних інформаційних платформ

4. Аналіз інфраструктурних можливостей хмарної платформи AWS та автоматизація управління ресурсами за допомогою TERRAFORM

Висновки

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій (п.5 включається до завдання за рішенням випускової кафедри) _____
Слайди у форматі Power Point (назва, вступ і актуальність кваліфікаційної роботи, Концепція Utilization Management, Архітектурні особливості комплексних систем рівня eHealth, Порівняльний аналіз медичної інформаційної системи (МІС) в Україні, Порівняння моделей розгортання: Cloud-native (SaaS) vs On-Premise, Хмарна платформа AWS як середовище для УМ, Профільний аналіз сервісів AWS, Автоматизація інфраструктури: Підхід Infrastructure as Code, Інструмент моніторингу та деплою Terraform, Практична реалізація та результати розгортання МІС, Розробка інфраструктури Terraform код, висновки та результати роботи).

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	проф. Безрук В.М.		12.07.2026

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / термін виконання етапів роботи	Примітка
1	Ознайомлення із завданням. Уточнення ТЗ	18.05-20.05.2026	виконано
2	Підбір літератури за темою роботи	21.05-03.06.2026	виконано
3	Виконання розділу 1	04.06-11.06.2026	виконано
4	Виконання розділу 2	12.06-15.06.2026	виконано
5	Виконання розділу 3	16.06-20.06.2026	виконано
6	Виконання розділу 4	21.06-05.07.2026	виконано
7	Оформлення презентаційного матеріалу	06.07-09.07.2026	виконано
8	Підготовка до захисту у ЕК	10.07-12.07.2026	виконано

Дата видачі завдання 18 05 2026 р.

Здобувач _____
 (підпис)

Керівник роботи _____ проф. Валерій Безрук _____
 (підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка: 58 стор. формату А4, 10 рис., 2 табл., 11 посилань.

Об'єкт дослідження – процеси управління використанням (Utilization Management / FinOps) обчислювальних ресурсів у хмарних середовищах.

Мета роботи – теоретичне обґрунтування, аналіз та практична реалізація відмовостійкої хмарної інфраструктури на базі платформи AWS з автоматизованим управлінням використанням ресурсів за допомогою підходу Infrastructure as Code (IaC) для забезпечення ефективного функціонування медичної інформаційної системи (MIC).

Проведено детальний порівняльний аналіз архітектурних профілів та інфраструктурних навантажень реальних українських медичних інформаційних систем, інтегрованих із системою eHealth. Доведено економічну та інженерну доцільність переходу від локальних серверів (On-Premise) до хмарної моделі обслуговування (SaaS) на базі екосистеми Amazon Web Services.

UTILIZATION MANAGEMENT, AWS, CLOUD COMPUTING, FINOPS, INFRASTRUCTURE AS CODE, TERRAFORM, МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА (MIC), EHEALTH, ОПТИМІЗАЦІЯ РЕСУРСІВ.

ABSTRACT

Explanatory note: 58 pp. format A4, 10 Fig., 11 reference, 2 tab.

Object of research – the processes of utilization management (FinOps) for computing resources in cloud environments.

Purpose of the work – to provide a theoretical foundation, analysis, and practical implementation of a fault-tolerant cloud infrastructure based on the AWS platform, featuring automated resource utilization management using the Infrastructure as Code (IaC) approach to ensure the effective operation of a medical information system (MIS).

A detailed comparative analysis was conducted of the architectural profiles and infrastructure workloads of real Ukrainian medical information systems integrated with the eHealth system. The economic and engineering feasibility of transitioning from on-premises servers to a cloud-based service model (SaaS) based on the Amazon Web Services ecosystem was demonstrated.

UTILIZATION MANAGEMENT, AWS, CLOUD COMPUTING, FINOPS, INFRASTRUCTURE AS CODE, TERRAFORM, HEALTHCARE INFORMATION SYSTEM (HIS), EHEALTH, RESOURCE OPTIMIZATION.

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ.....	8
ВСТУП.....	11
1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ UTILIZATION MANAGEMENT В КУЛЬТУРІ ХМАРНИХ ОБЧИСЛЕНЬ.....	13
1.1 Цілі управління використанням ресурсів у хмарі.....	14
1.2 Проблематика управління використанням ресурсів у хмарі.....	15
1.3 Компоненти управління використанням (Utilization Management).....	17
1.4 Програми управління використанням ресурсів.....	18
ВИСНОВКИ ДО РОЗДІЛУ 1.....	20
2 АРХІТЕКТУРНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ ЯК ОБ'ЄКТА УПРАВЛІННЯ ХМАРНИМИ РЕСУРСАМИ.....	21
2.1 Інфраструктурні типи розгортання інформаційних систем (MIS).....	22
2.2 Функціональні завдання та оптимізація процесів в ІС.....	23
2.3 Модульна структура та функціональні компоненти ІС.....	24
2.4 Ефективність та переваги використання цифровізованих систем.....	25
ВИСНОВКИ ДО РОЗДІЛУ 2.....	27
3 ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУРНИХ ВИМОГ ТА МОДЕЛЕЙ РОЗГОРТАННЯ НАЦІОНАЛЬНИХ ІНФОРМАЦІЙНИХ ПЛАТФОРМ.....	28
3.1 Архітектурний профіль ІС «Doctor Eleks».....	28
3.2 Модульна специфікація ІС «EMCiMED».....	29
3.3 Високонавантажений B2C-профіль ІС «Helsi».....	30
3.4 Порівняльний аналіз моделей розгортання: Cloud-native (SaaS) проти On-Premise (Server).....	31
ВИСНОВКИ ДО РОЗДІЛУ 3.....	33

4 АНАЛІЗ ІНФРАСТРУКТУРНИХ МОЖЛИВОСТЕЙ ХМАРНОЇ ПЛАТФОРМИ AWS ТА АВТОМАТИЗАЦІЯ УПРАВЛІННЯ РЕСУРСАМИ ЗА ДОПОМОГОЮ TERRAFORM.....	34
4.1 Моделі хмарних сервісів у розрізі оптимізації утилізації.....	34
4.2 Профільний аналіз сервісів AWS крізь призму Utilization Management...	36
4.3 Концепція Infrastructure as Code (IaC) як інструмент фінансового контролю.....	37
4.4 Інструмент декларативного управління інфраструктурою Terraform.....	38
4.5 Практична реалізація та деплой інфраструктури MISC в AWS.....	39
ВИСНОВКИ ДО РОЗДІЛУ 4.....	42
ВИСНОВКИ.....	43
ПЕРЕЛІК ПОСИЛАНЬ.....	45
ДОДАТОК А. АРХІТЕКТУРА ІНФРАСТРУКТУРИ MISC.....	47
ДОДАТОК Б. ФРАГМЕНТ TERRAFORM-КОДУ ІНФРАСТРУКТУРИ.....	48
ДОДАТОК В. СЛАЙДИ ПРЕЗЕНТАЦІЇ.....	51

ПЕРЕЛІК СКОРОЧЕНЬ

ALB (Application Load Balancer) – балансувальник навантаження рівня додатків;

API (Application Programming Interface) – інтерфейс прикладного програмування;

AWS (Amazon Web Services) – хмарна платформа компанії Amazon;

AZ (Availability Zone) – зона доступності (фізичний дата-центр у хмарі);

B2C (Business-to-Consumer) – модель взаємодії бізнесу з кінцевим споживачем;

B2B (Business-to-Business) – модель взаємодії між комерційними компаніями;

CapEx (Capital Expenditures) – капітальні витрати;

CDN (Content Delivery Network) – мережа доставки контенту;

CI/CD (Continuous Integration / Continuous Deployment) – безперервна інтеграція та безперервне розгортання;

CPU (Central Processing Unit) – центральний процесор;

CRM (Customer Relationship Management) – система управління відносинами з клієнтами;

DICOM (Digital Imaging and Communications in Medicine) – міжнародний стандарт для медичних зображень;

EC2 (Elastic Compute Cloud) – сервіс оренди віртуальних серверів у хмарі AWS;

ECS (Elastic Container Service) – служба оркестровки контейнерів від AWS;

EKS (Elastic Kubernetes Service) – служба керування кластерами Kubernetes в AWS;

ELB (Elastic Load Balancing) – служба автоматичного балансування навантаження від AWS;

ERP (Enterprise Resource Planning) – система планування ресурсів підприємства;

HCL (HashiCorp Configuration Language) – декларативна мова конфігурації Terraform;

HL7 (Health Level Seven) – міжнародний стандарт обміну медичною інформацією;

IAM (Identity and Access Management) – система керування ідентифікацією та доступом;

IaC (Infrastructure as Code) – інфраструктура як код;

IaaS (Infrastructure as a Service) – інфраструктура як послуга;

IOPS (Input/Output Operations Per Second) – кількість операцій введення-виведення за секунду;

OLTP (Online Transaction Processing) – оперативна обробка транзакцій;

OpEx (Operational Expenditures) – операційні витрати;

PaaS (Platform as a Service) – платформа як послуга;

PACS (Picture Archiving and Communication System) – система архівації та передачі медичних зображень;

RDS (Relational Database Service) – служба реляційних баз даних від AWS;

S3 (Simple Storage Service) – хмарне об'єктне сховище від AWS;

SaaS (Software as a Service) – програмне забезпечення як послуга;

UM (Utilization Management) – управління використанням (утилізацією) ресурсів;

VPC (Virtual Private Cloud) – віртуальна приватна хмара;

ЦБД – Центральна база даних;

ЕСОЗ – Електронна система охорони здоров'я;

ІС – Інформаційна система;

ІТ – Інформаційні технології;

КСЗІ – Комплексна система захисту інформації;

МІС – Медична інформаційна система;

МОЗ – Міністерство охорони здоров'я України;

НСЗУ – Національна служба здоров'я України.

ВСТУП

Стрімка цифровізація сучасної інфраструктури, зокрема в критично важливих секторах, таких як охорона здоров'я (рівня eHealth та масштабних медичних інформаційних систем), висуває жорсткі вимоги до обчислювальних потужностей. Перехід від традиційних локальних серверів (On-Premise) до хмарних архітектур став загальносвітовим стандартом. Проте, разом із гнучкістю та масштабованістю, компанії стикаються з критичною проблемою неефективного споживання ресурсів, відомою як «хмарне марнотратство» (Cloud Waste). Згідно з дослідженнями FinOps Foundation, до 30% інфраструктурних витрат у хмарі витрачаються на оплату потужностей, які фактично простують або використовуються не на повну силу.

У зв'язку з цим виникає гостра потреба в розробці та впровадженні методологій управління використанням ресурсів (Utilization Management – UM). Традиційний ручний моніторинг та конфігурування середовищ через графічні консолі провайдерів призводять до виникнення людського фактора, уповільнення життєвого циклу розробки (CI/CD) та появи «осиротілих» інфраструктурних компонентів, що таємно споживають бюджет. Найбільш перспективним розв'язанням цієї проблеми є автоматизація процесів UM за допомогою концепції «Інфраструктура як код» (Infrastructure as Code – IaC). Використання сучасних декларативних інструментів, таких як Terraform, у поєднанні з провідною хмарною платформою Amazon Web Services (AWS), дозволяє жорстко регламентувати квоти споживання у вигляді коду, динамічно адаптувати систему під змінні навантаження (Workloads) та гарантувати максимальний коефіцієнт корисного використання заліза.

Таким чином, дослідження методів аналізу та автоматизації управління утилізацією хмарних ресурсів для комплексних інформаційних систем є

надзвичайно актуальним науково-практичним завданням, що безпосередньо впливає на фінансову стійкість, технологічну гнучкість та безпеку сучасних цифрових продуктів.

1 ТЕОРЕТИКО-МЕТОДОЛОГІЧНІ ОСНОВИ UTILIZATION MANAGEMENT В КУЛЬТУРІ ХМАРНИХ ОБЧИСЛЕНЬ

Дослідження концепції управління використанням (Utilization Management, UM) демонструє, що цей термін історично еволюціонував і не має єдиного ізольованого визначення. Подібно до стратегій «стримування витрат» (cost containment) у традиційному менеджменті, у контексті інформаційних технологій та хмарних платформ Utilization Management розглядається як комплекс методів і підходів для моніторингу, оцінки та оптимізації споживання обчислювальних ресурсів з метою максимізації їх ефективності та мінімізації фінансових витрат.

Якщо адаптувати класичні підходи (наприклад, визначення Асоціації фінансового менеджменту HFMA) до інфраструктури хмарних послуг, то управління використанням у хмарі – це інтеграція аналізу утилізації потужностей, управління ризиками (простоями або перевантаженнями) та забезпечення якості сервісу (QoS) в єдину систему управління для забезпечення розумного виділення ресурсів та високої відмовостійкості архітектури.

У традиційних сферах UM запобігає фінансовим втратам від некомпетентного планування. У розрізі хмарної платформи AWS комплексний аналіз використання є критично важливим для запобігання так званому «хмарному марнотратству» (Cloud Waste) – ситуаціям, коли компанії платять за запущені, але фактично не задіяні або надлишкові віртуальні машини, сховища та мережні шлюзи. Аудит з боку провайдера або внутрішніх FinOps-команд (Financial Operations) дозволяє виявити неправомірні витрати, дублюючі сервіси або неправильно підібрані типи інстансів (наприклад, використання дорогого інстансу з оптимізацією обчислень там, где вистачило б базового) [1].

Впровадження потужної програми управління утилізацією гарантує, що цифрові продукти отримують належні ресурси (обчислювальну потужність, пам'ять) у потрібний час (наприклад, під час пікових навантажень), забезпечуючи економічно ефективну роботу бізнесу без втрати продуктивності.

1.1 Цілі управління використанням ресурсів у хмарі

Управління утилізацією в середовищі AWS має потрібну мету:

- Зниження та оптимізація витрат (Cost Optimization): це першочергова мета через постійне масштабування хмарних інфраструктур. Процедури УМ дозволяють оцінювати кожен виділений інфраструктурний компонент до і після його розгортання, перевіряючи його реальну необхідність та ефективність. Це дозволяє компаніям платити виключно за фактично спожиті потужності (модель Pay-as-you-go).

- Забезпечення продуктивності та надійності (Performance Efficiency): друга мета полягає у покращенні користувацького досвіду (User Experience). Аналіз утилізації допомагає чітко визначити, як зміна конфігурації ресурсів (наприклад, збільшення IOPS для дискової підсистеми) впливає на швидкість роботи системи. Отримані метрики використовуються для автоматизації архітектурних рішень у майбутньому.

- Запобігання інфраструктурним ризикам та перевитратам (Risk & Overspending Mitigation): огляди УМ збирають телеметрію з працюючих систем, надаючи інженерам та фінансовим аналітикам обґрунтовані дані для захисту бюджетів та планування майбутніх ІТ-ініціатив [1, 2].

У контексті аналізу управління ресурсами на базі AWS варто підкреслити три ключові аспекти, на які спрямована увага дослідження:

- Поелементна оцінка – досліджуються методи, які базуються на оцінці використання кожної окремої одиниці інфраструктури (конкретного сервера EC2, бази даних RDS або сховища S3) у кожному конкретному випадку.

- Проактивний контроль (Перед-верифікація) – основна увага приділяється перевірці та моделюванню архітектури до надання та активації послуг (використання інструментів на кшталт AWS Pricing Calculator), щоб уникнути архітектурних помилок.

- Економічна дія споживача – наголошується на діях, що вживаються безпосередньо покупцем хмарних послуг (архітекторами та розробниками компанії) для оптимізації свого рахунку за хмару, а не на стандартних базових обмеженнях самого провайдера AWS.

Перша характеристика відрізняє Utilization Management від методів, які аналізують лише загальні абстрактні витрати компанії без прив'язки до реальних технічних метрик. Друга характеристика відрізняє проактивне управління утилізацією від ретроспективного розбору величезних рахунків наприкінці місяця, коли гроші вже списано. Третя привертає увагу саме до усвідомленого керування архітектурою з боку бізнесу, що ділить фінансову відповідальність за ефективність хмари з постачальником послуг [2].

1.2 Проблематика управління використанням ресурсів у хмарі

Впровадження процесів Utilization Management у хмарних середовищах не позбавлене технологічних та організаційних викликів. Один із головних ризиків УМ полягає в тому, що він може спровокувати внутрішній конфлікт між командами розробки (DevOps/Engineering), які безпосередньо розгортають інфраструктуру, та фінансовим менеджментом (FinOps), що контролює витрати платформи [3].

Наприклад, якщо фінансовий департамент блокує виділення додаткових або потужніших віртуальних машин (інстансів), тягар оптимізації та ручного доопрацювання коду лягає на розробників. Якщо це відбувається постфактум, компанія отримує неочікувано високі рахунки від AWS за перевищення лімітів (так званий Cloud Bill Shock), які бізнес змушений покривати з незапланованих бюджетів.

Управління використанням завжди надає перевагу найбільш безпечним, стабільним та економічно ефективним типам інфраструктури. Якщо розробникам для тестування архітектури потрібен експериментальний або надпотужний сервіс (наприклад, інстанси з графічними прискорювачами для штучного інтелекту), процедура погодження такого бюджету всередині компанії вимагає багато часу. У гіршому випадку інженери змушені будувати обхідні архітектурні рішення, що шкодить продуктивності системи.

Рекомендації щодо економії хмарних ресурсів також можуть ігноруватися інженерами під час критичних релізів або аварій (Incidents). У таких ситуаціях системний архітектор завжди віддаватиме пріоритет працездатності та доступності сервісу (SLA), а не економії коштів, що знову ж таки викликає напругу пізніше, коли керівництво аналізує перевитрати за місяць.

Окрім цього, процеси UM створюють значне навантаження на персонал. Замість написання коду та створення продукту, Cloud-інженери змушені виконувати рутинні завдання з аналізу та аудиту метрик. Це може призвести до затримок у розгортанні критично важливих оновлень системи. Часто персонал чинить опір процедурам управління використанням, розглядаючи їх як зайву бюрократію, нав'язану менеджментом [2, 3].

Розробка та запуск повноцінної програми UM в AWS – це тривалий і складний процес, який вимагає повної перебудови IT-процесів (перехід до культури FinOps). Для успіху програми необхідно врахувати кілька факторів:

- Баланс інтересів: чітке узгодження цілей бізнесу (мінімізація витрат) та розробки (масштабованість і швидкість).
- Розподіл ролей: визначення зон відповідальності між DevOps-інженерами, власниками продуктів (Product Owners) та фінансовими аналітиками.
- Стандартизація: врахування специфіки різних ІТ-сервісів (наприклад, робота з базами даних вимагає іншого підходу до утилізації, ніж робота з безсерверними обчисленнями AWS Lambda).
- Compliance (Відповідність вимогам): інтеграція хмарних лімітів із державними та міжнародними стандартами безпеки даних (наприклад, GDPR, PCI-DSS), що регулюють зберігання інформації в хмарі [2].

1.3 Компоненти управління використанням (Utilization Management)

Ефективна стратегія управління використанням ресурсів у хмарі базується на трьох фундаментальних технічних компонентах описаних в рис. 1.1.

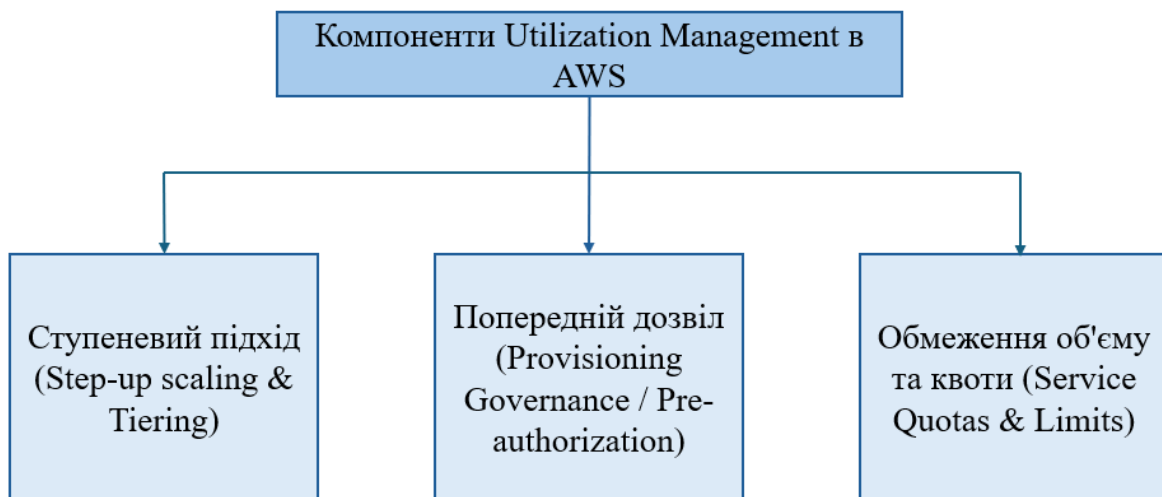


Рисунок 1.1 – Фундаментальні технічні компоненти на яких базується ефективна стратегія управління використанням ресурсів у хмарі

– Ступеневий підхід (Tiering / Step-up Approach): концепція полягає в тому, що для вирішення завдань спочатку мають використовуватися найдешевші, але достатні за ефективністю ресурси (наприклад, спочатку використовуються спотові інстанси AWS Spot Instances або базові типи обчислень). Тільки якщо базовий рівень не дає очікуваного результату чи стабільності, система переводиться на вищий, дорожчий рівень інфраструктури (наприклад, виділені сервери On-Demand). Це гарантує економічну доцільність для кожного окремого процесу.

– Попередній дозвіл (Provisioning Governance / Pre-authorization): цей компонент вимагає обов'язкової автоматичної або ручної перевірки та схвалення конфігурації інфраструктури перед її фактичним розгортанням в AWS. Це допомагає зробити роботу хмари безпечнішою, запобігає перевитратам, помилкам конфігурації та використанню надлишкових сервісів, які не відповідають архітектурним потребам проєкту.

– Обмеження об'єму та квоти (Service Quotas & Limits): встановлення жорстких лімітів на кількість ресурсів, які певна команда чи сервіс можуть запустити протягом певного періоду (наприклад, ліміт на кількість одночасних запусків AWS Lambda або максимальний об'єм сховища EBS). Це критично як з міркувань безпеки (захист від DDoS або помилок у коді, що нескінченно плодить сервери), так і для зменшення хмарних відходів (Cloud Waste) [1].

1.4 Програми управління використанням ресурсів

Успішні хмарні UM-програми базуються на забезпеченні інфраструктури саме тими потужностями, які необхідні бізнесу в конкретний момент часу. Вони складаються з п'яти ключових елементів:

– Управління ризиками: запобігання простоям критичних систем (Downtime) та уникнення фінансових втрат через неконтрольоване масштабування сервісів.

- Раннє втручання: автоматичне виявлення аномалій у споживанні ресурсів AWS (наприклад, раптовий стрибок витрат) для негайної стабілізації системи.
- Профілактичний моніторинг: постійний аналіз архітектурного «здоров'я» хмари за допомогою стандартів (наприклад, AWS Well-Architected Framework) для оптимізації навантаження ще до того, як виникнуть проблеми з продуктивністю.
- Навчання інженерів (Cloud Literacy): надання технічним командам знань про кращі практики управління витратами в AWS, управління лімітами та використання інструментів економії (Savings Plans, Reserved Instances).
- Аналіз використання (Utilization Analysis): безперервне визначення відповідності запусненої інфраструктури реальним бізнес-завданням. Він поділяється на три класичні фази огляду:
 - Проспективний (Попередній) аналіз – оцінка та моделювання архітектури до запуску інфраструктури (наприклад, за допомогою AWS Pricing Calculator або шаблонів Infrastructure as Code), щоб оцінити майбутні потреби та вартість систем.
 - Одночасний (Поточний) аналіз – моніторинг у реальному часі через заздалегідь визначені проміжки часу (через AWS CloudWatch). Фокусується на відповідності поточного завантаження систем (CPU, RAM) до виділених лімітів і налаштуванні тривалості процесів.
 - Аудит, що виконується після завершення певного робочого циклу або наприкінці місяця. Аналізує фактичну утилізацію потужностей за минулий період (через AWS Cost Explorer та AWS Trusted Advisor), щоб згенерувати правильні платіжні коди (Tags) для фінансової звітності та скоригувати архітектуру на майбутнє [1-3].

ВИСНОВКИ ДО РОЗДІЛУ 1

У першому розділі досліджено теоретико-методологічні основи управління використанням (Utilization Management) у контексті IT-інфраструктури та хмарних обчислень. Встановлено, що в умовах стрімкого масштабування цифрових продуктів UM є ключовим інструментом FinOps для боротьби з неефективним споживанням ресурсів (Cloud Waste). Визначено потрібну мету управління утилізацією: оптимізація фінансових витрат, забезпечення високої продуктивності систем та мінімізація інфраструктурних ризиків. Доведено, що ефективна програма UM базується на трьох компонентах: ступеневому виділенні потужностей, обов'язковій попередній авторизації архітектурних змін та жорсткому контролю квот. Головним викликом при впровадженні таких процесів є досягнення балансу між технічними потребами інженерів (гнучкість, швидкість розгортання) та бізнес-вимогами (мінімізація витрат).

2 АРХІТЕКТУРНИЙ АНАЛІЗ ІНФОРМАЦІЙНИХ СИСТЕМ ЯК ОБ'ЄКТА УПРАВЛІННЯ ХМАРНИМИ РЕСУРСАМИ

Сучасний етап цифровізації суспільства призвів до створення масштабних національних та корпоративних платформ. Прикладом такої високонавантаженої системи в Україні є Електронна система охорони здоров'я (ЕСОЗ), яка функціонує як дворівнева інформаційно-телекомунікаційна архітектура. Вона забезпечує автоматизацію обліку великих масивів даних, інтегруючи роботу кінцевих користувачів із Центральною базою даних (ЦБД).

Для взаємодії з ЦБД використовуються спеціалізовані сервіси – Медичні інформаційні системи (МІС). З технічної точки зору, МІС – це розподілене програмне забезпечення (Enterprise Software), архітектура якого розроблена для збору, обробки, передачі та безпечного зберігання терабайтів критично важливих даних [4, 5].

Оскільки функціонування таких систем потребує безперервної синхронізації з державними реєстрами та суворого дотримання стандартів кібербезпеки (наприклад, Комплексної системи захисту інформації – КСЗІ), розробники та оператори МІС масово відмовляються від традиційної локальної інфраструктури на користь віртуальних дата-центрів та міжнародних хмарних платформ, серед яких лідером є Amazon Web Services (AWS).

Архітектура та інфраструктурний базис системи E-Health показана на рис. 2.1.

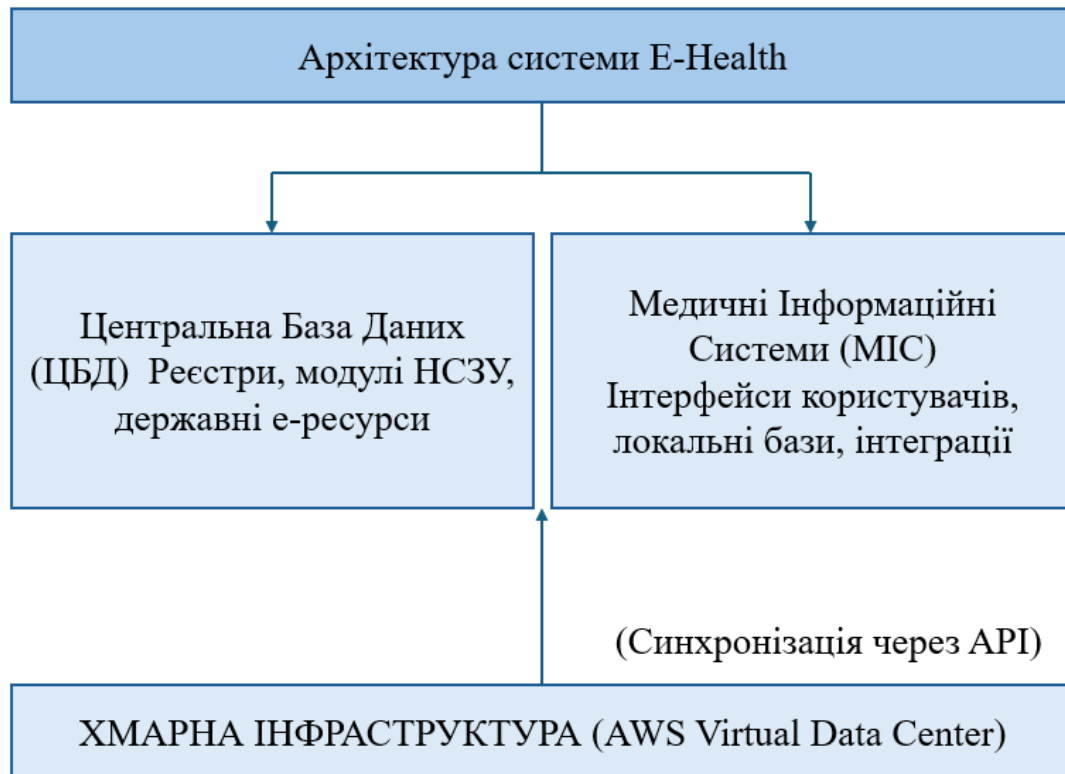


Рисунок 2.1 – Архітектура та інфраструктурний базис системи E-Health

2.1 Інфраструктурні типи розгортання інформаційних систем (МІС)

Залежно від архітектурного підходу та моделі споживання обчислювальних ресурсів, інформаційні системи поділяють на два ключові типи:

– Хмарні системи (Cloud-native / SaaS): оптимальне рішення для більшості організацій. Уся логіка системи, бази даних та інтерфейси розміщуються у хмарі провайдера. Перевага: система доступна з будь-якого пристрою через Інтернет за моделлю SaaS (Software as a Service). Клієнту не потрібно купувати та обслуговувати дороге серверне обладнання, а завдання з масштабування, оновлення та утилізації ресурсів лягають на плечі хмарного провайдера та розробника.

– Локальні / Серверні системи (On-Premise): використовуються переважно великими корпораціями або специфічними установами з надскладною

внутрішньою інфраструктурою. Програмне забезпечення розгортається безпосередньо на фізичних серверах (Bare Metal) всередині організації. Особливість: адмініструванням, безпекою та балансуванням навантаження організація займається самостійно, що вимагає великого штату ІТ-спеціалістів та значних капітальних витрат (CapEx), проте дозволяє будувати максимально ізольовані кастомні модулі [6].

2.2 Функціональні завдання та оптимізація процесів в ІС

Будь-яка сучасна інформаційна система (зокрема МІС) складається з функціональних блоків, які автоматизують операційну діяльність. До основних завдань, що створюють безпосереднє навантаження на обчислювальні ресурси, належать:

- Управління великими масивами даних (Big Data Management): Системи оперують терабайтами інформації (картки користувачів, цифрові профілі, результати досліджень, медіафайли). Усі дані мають бути доступні 24/7 з будь-якої точки входу, що потребує використання високоефективних хмарних сховищ та інструментів індексації.

- Генерація аналітики та звітності (Reporting Engines): Формування гнучких звітів, агрегація даних за періоди та аудит транзакцій створюють короткочасні, але надзвичайно потужні пікові навантаження на процесори (CPU) та бази даних.

- Контроль доступності та розподіл ресурсів: Забезпечення роботи внутрішніх порталів для персоналу та зовнішніх кабінетів для клієнтів (трансляція даних у реальному часі, онлайн-записи, черги), що вимагає динамічного балансування мережного трафіку (рис. 2.2) [1, 6].



Рисунок 2.2 – Основні завдання МІС

2.3 Модульна структура та функціональні компоненти ІС

Сучасні комплексні системи проектуються за мікросервісною або модульною архітектурою. Це дозволяє гнучко налаштовувати конфігурацію системи під конкретні вимоги та ізолювати обчислювальні ресурси для різних завдань. Структуру компонентів МІС можна класифікувати за такими групами:

- Аналітичні та управлінські компоненти: інструменти для ведення внутрішнього обліку, аналізу ефективності бізнес-процесів та аудиту якості послуг. На рівні обчислень вони потребують складних пошукових запитів до баз даних та візуалізації результатів у вигляді графіків та таблиць.

- Профільні операційні компоненти: модулі реєстрації користувачів, ведення електронних карток, збереження історії транзакцій (чи операцій) та ведення статистики.

- Фінансово-економічні компоненти: модулі розрахунку собівартості, тарифів, обліку матеріальних запасів (складів) та нарахування виплат. Створюють транзакційне навантаження на реляційні бази даних (OLTP-навантаження).

– Компоненти інтеграції та обміну даними (API Gateways): сервіси для ведення уніфікованих довідників та каталогів, які забезпечують обмін даними між різними державними та комерційними реєстрами через захищені API-інтерфейси.

– Загальнотехнічні та безпекові компоненти (Security & IAM): системи контролю доступу (Identity and Access Management), шифрування даних під час передачі та зберігання, а також захисту від зовнішніх кібератак [6].

2.4 Ефективність та переваги використання цифровізованих систем

Перехід від паперово-рутинних процесів до автоматизованих інформаційних систем (МІС) дає фундаментальні переваги як для операторів послуг, так і для кінцевих споживачів (табл. 2.1) [5, 6].

Таблиця 2.1 – Переваги для провайдерів і споживачів

Переваги для організації (провайдера послуг)	Переваги для кінцевого споживача (користувача)
Оптимізація документообігу: повна ліквідація паперової рутини, виключення дублювання інформації, миттєвий доступ до єдиного джерела істини (Single Source of Truth) для всього авторизованого персоналу.	Доступність та прозорість даних: Оперативний онлайн-доступ до власних даних, результатів аналізів чи транзакцій, можливість відстеження історії взаємодії в реальному часі.
Мінімізація людського фактора: автоматизація процесів різко знижує ризик втрати чи некоректного прикріплення документів (наприклад, плутанини з картками), що підвищує загальну якість сервісу.	Зручність та економія часу: можливість попереднього дистанційного запису через онлайн-реєстратури, повна ліквідація фізичних черг та швидкий зворотний зв'язок.

Продовження таблиці 2.1.

Впровадження віддалених технологій (Телесервіси): можливість проведення дистанційних консультацій у реальному часі (телемедицина), віддалений моніторинг станів, проведення консилиумів між фахівцями з різних географічних точок.	Безпека та верифікація: високий рівень захисту від підробки, модифікації або втрати персональних даних завдяки постійному контролю з боку користувача та шифруванню.
Системна узгодженість: ефективне управління внутрішнім розкладом, динамічний розподіл навантаження між філіями, точне прогнозування попиту на послуги та гнучке ціноутворення.	

ВИСНОВКИ ДО РОЗДІЛУ 2

У другому розділі проведено архітектурний аналіз комплексних інформаційних систем (на прикладі масштабних платформ рівня eHealth) як об'єктів для розгортання у хмарі. З'ясовано, що сучасні цифрові платформи мають розподілену модульну архітектуру і генерують різнотипні навантаження (обробка Big Data, генерація аналітики, транзакційні процеси). Зроблено висновок, що забезпечення стабільної роботи таких систем вимагає переходу від традиційних локальних серверів до віртуальних дата-центрів. Хмарна інфраструктура дозволяє гнучко розподіляти обчислювальні ресурси між окремими системними компонентами (аналітичними, фінансово-економічними, загальнотехнічними), автоматизувати документообіг та гарантувати високий рівень доступності і захисту даних відповідно до державних стандартів безпеки.

3 ПОРІВНЯЛЬНИЙ АНАЛІЗ АРХІТЕКТУРНИХ ВИМОГ ТА МОДЕЛЕЙ РОЗГОРТАННЯ НАЦІОНАЛЬНИХ ІНФОРМАЦІЙНИХ ПЛАТФОРМ

Для розуміння специфіки управління ресурсами в хмарі AWS необхідно проаналізувати технічні особливості та архітектурну структуру провідних інформаційних систем (МІС), що функціонують на ринку України та інтегровані з екосистемою eHealth. Кожна з цих систем формує унікальний тип інфраструктурного навантаження (Workload), що безпосередньо впливає на стратегію утилізації обчислювальних потужностей.

3.1 Архітектурний профіль ІС «Doctor Eleks»

Doctor Eleks є комплексним високонавантаженим ERP-рішенням для медичних установ будь-якого масштабу. З інфраструктурної точки зору, ця система характеризується високими вимогами до обробки неструктурованих даних та збереження медіаконтенту.

- Особливості навантаження: окрім стандартних реляційних баз даних для ведення карток та звітності, система містить підсистему PACS (архів медичних зображень) та вбудовані редактори для обробки відео та важких графічних файлів.

- Вимоги до ресурсів AWS: такий профіль системи вимагає використання високопродуктивних обчислювальних інстансів (наприклад, сімейства AWS EC2 Compute Optimized або Memory Optimized) для рендерингу та аналізу зображень, а також гнучких рівневих сховищ типу Amazon S3 та S3 Glacier для довгострокового та економічно вигідного зберігання великих архівів даних (PACS) з оптимізацією витрат на утилізацію дисків [5, 6].

На рис. 3.1 зображено медичну інформаційну систему Doctor Eleks.

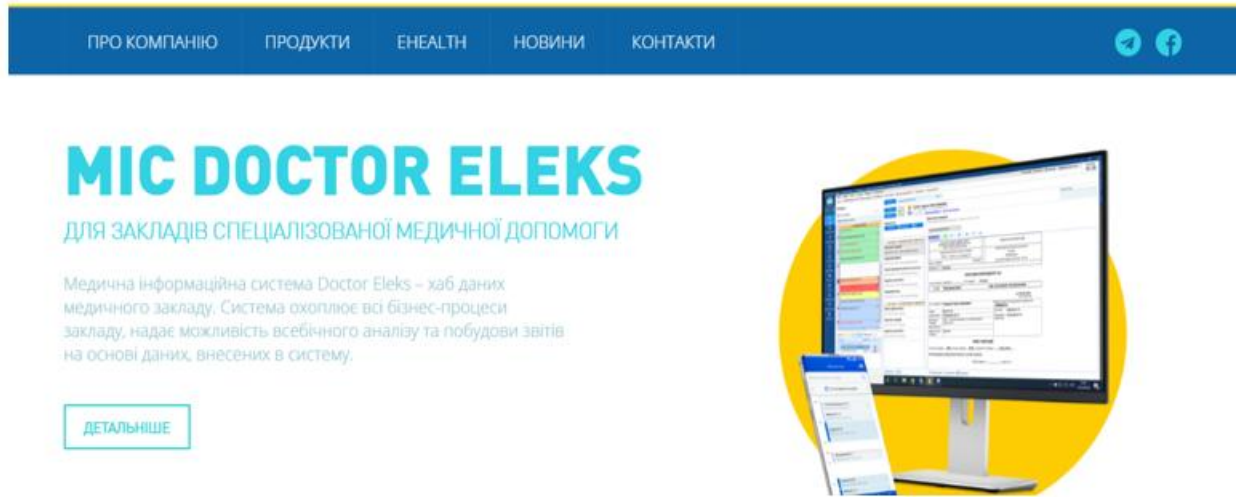


Рисунок 3.1– MIC Doctor Eleks

3.2 Модульна специфікація ІС «EMCiMED»

EMCiMED проектувалася як суворо розгалужена, модульна інформаційна система. Вона дозволяє гнучко збирати конфігурацію сервісів під конкретного замовника (модулі реєстратури, стаціонару, лабораторії, CRM тощо).

– Особливості навантаження: Модульна структура створює ізольовані потоки даних. Інтеграція з фінансовими програмами (наприклад, 1C/BAS) потребує постійної синхронізації та обміну даними через API-інтерфейси. Безпека даних забезпечується наскрізним шифруванням та апаратними ключами.

– Вимоги до ресурсів AWS: Архітектура EMCiMED ідеально адаптується під мікросервісний підхід у хмарі. Окремі модулі можуть бути розгорнуті в контейнерах за допомогою Amazon ECS або Amazon EKS (Kubernetes). Управління використанням тут базується на ізольованому масштабуванні конкретного модуля (наприклад, під час пікових навантажень на лабораторію масштабується лише контейнер лабораторії, що економить загальний бюджет інфраструктури) [5, 6].

На рис. 3.2 зображено медичну інформаційну систему EMCiMED.

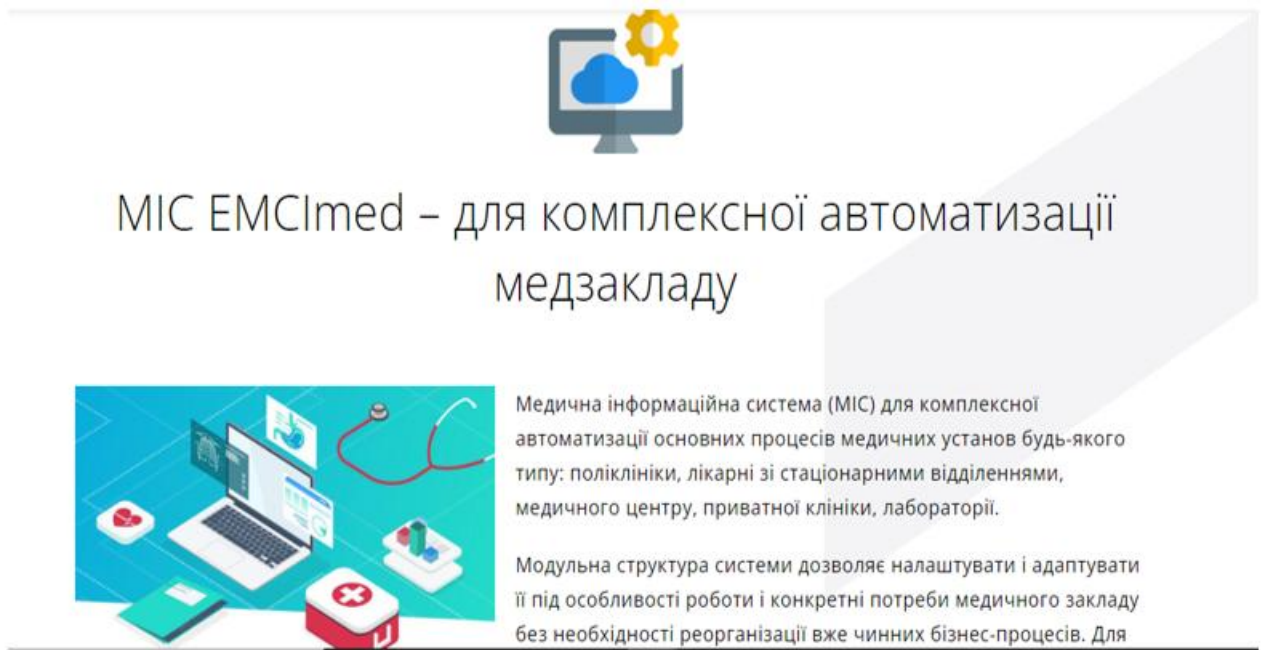


Рисунок 3.2 – МІС EMCiMED

3.3 Високонавантажений B2C-профіль IC «Helsi»

Helsi є однією з наймасштабніших цифрових платформ в Україні, яка одночасно обслуговує мільйони пацієнтів (B2C-сегмент) та тисячі лікарів і державних установ (B2B-сегмент).

– Особливості навантаження: величезний веб-трафік, зумовлений одночасною роботою мільйонів користувачів, які шукають лікарів, записуються на прийом та переглядають аналізи онлайн. Система підтримує міжнародні стандарти HL7 та формат DICOM, а також має сертифікат КСЗІ для захисту даних у дата-центрі.

– Вимоги до ресурсів AWS: профіль Helsi вимагає максимального рівня автоматизації управління ресурсами. Для обробки масових запитів критично важливим є використання балансувальників навантаження AWS Application Load

Balancer (ALB) та груп автоналаштування AWS Auto Scaling, які автоматично додають нові сервери під час ранкових піків записів та вимикають їх уночі, забезпечуючи нульове простування потужностей [4-7].

На рис. 3.3 зображено медичну інформаційну систему Helsi.

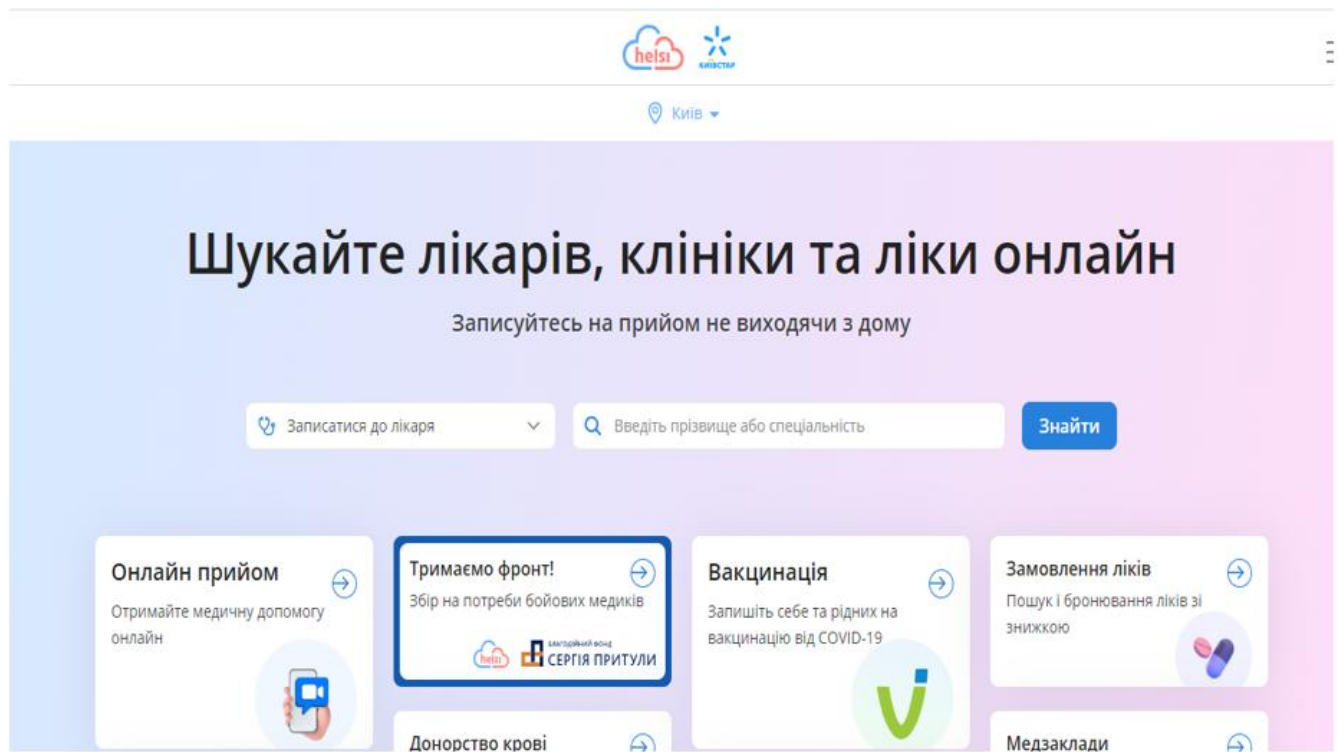


Рисунок 3.3 – MIC Helsi

3.4 Порівняльний аналіз моделей розгортання: Cloud-native (SaaS) проти On-Premise (Server)

Всі аналізовані системи пропонують дві кардинально різні моделі керування інфраструктурними ресурсами, описані у табл.3.1 [8, 9].

Таблиця 3.1 – Порівняння хмарних моделей Cloud-native (SaaS) і On-Premise (Server)

Критерій порівняння	Хмарна модель (SaaS / Cloud-native)	Локальна модель (On-Premise / Server)
Локація та контроль ресурсів	Інфраструктура повністю розміщена в хмарі (наприклад, AWS). Управління залізом здійснює провайдер.	Фізичні сервери розміщені всередині організації. Повний контроль заліза власною ІТ-службою.
Економічна модель	OpEx (Операційні витрати): оплата за фактично спожиті ресурси (Pay-as-you-go). Відсутність капітальних інвестицій в обладнання.	CapEx (Капітальні витрати): великі одноразові інвестиції в закупівлю серверів, ліцензій, систем охолодження та живлення.
Ефективність утилізації (Utilization Efficiency)	Максимальна: ресурси динамічно стискаються або розширюються під поточний трафік за допомогою інструментів автоматизації.	Низька: залізо купується із запасом під пікові навантаження, тому 70-80% часу сервери працюють «вхолосту».
Масштабованість (Scalability)	Миттєва: додавання нових потужностей, гігабайтів сховища чи ядер процесора відбувається за кілька кліків або автоматично.	Обмежена: для розширення потрібно купувати, чекати на доставку та фізично монтувати нове обладнання.
Безпека та ізоляція	Безпека забезпечується інструментами хмари (шифрування, AWS IAM) та сертифікатами відповідності (KC3I, ISO).	Повна ізоляція даних від третіх осіб. Ризик безпеки залежить виключно від кваліфікації внутрішньої команди.

ВИСНОВКИ ДО РОЗДІЛУ 3

У третьому розділі здійснено практичний порівняльний аналіз архітектурних профілів реальних інформаційних систем (Doctor Eleks, EMCiMED, Helsi). Доведено, що кожен тип системи формує унікальний інфраструктурний патерн (Workload), який вимагає специфічних інструментів AWS: від управління важкими сховищами для медіаданих до використання груп автоматичного масштабування (Auto Scaling) для обробки мільйонного користувацького веб-трафіку. Порівняння локальної (On-Premise) та хмарної (SaaS) моделей розгортання показало беззаперечну перевагу останньої в контексті управління ресурсами. Хмарна модель (OpEx) дозволяє досягти максимального коефіцієнта ефективності утилізації (Utilization Efficiency), усуває проблему простою дорогого обладнання та забезпечує миттєву масштабованість інфраструктури з оплатою лише за фактично спожиті потужності.

4 АНАЛІЗ ІНФРАСТРУКТУРНИХ МОЖЛИВОСТЕЙ ХМАРНОЇ ПЛАТФОРМИ AWS ТА АВТОМАТИЗАЦІЯ УПРАВЛІННЯ РЕСУРСАМИ ЗА ДОПОМОГОЮ TERRAFORM

Amazon Web Services (AWS) наразі є світовим лідером у сфері хмарних обчислень, утримуючи домінантну частку ринку. Платформа функціонує за моделлю Pay-as-you-go (оплата за фактичне споживання), що робить її ідеальним середовищем для реалізації стратегій Utilization Management. Глобальна інфраструктура AWS розгорнута в географічно ізольованих Регіонах (Regions), кожен з яких складається з кількох Зон доступності (Availability Zones – AZ). Центри обробки даних об'єднані приватною високошвидкісною оптоволоконною мережею, що гарантує відмовостійкість. Додатково периферійні локації (Edge Locations) у складі мережі доставки контенту (CDN) забезпечують мінімальну затримку (latency) для кінцевих користувачів у всьому світі.

4.1 Моделі хмарних сервісів у розрізі оптимізації утилізації

Ефективність управління об'єднанням ресурсів та контролю їх використання безпосередньо залежить від обраної моделі хмарних обчислень (шарів хмари), які формують ієрархічну структуру (рис. 4.1) [9, 10].

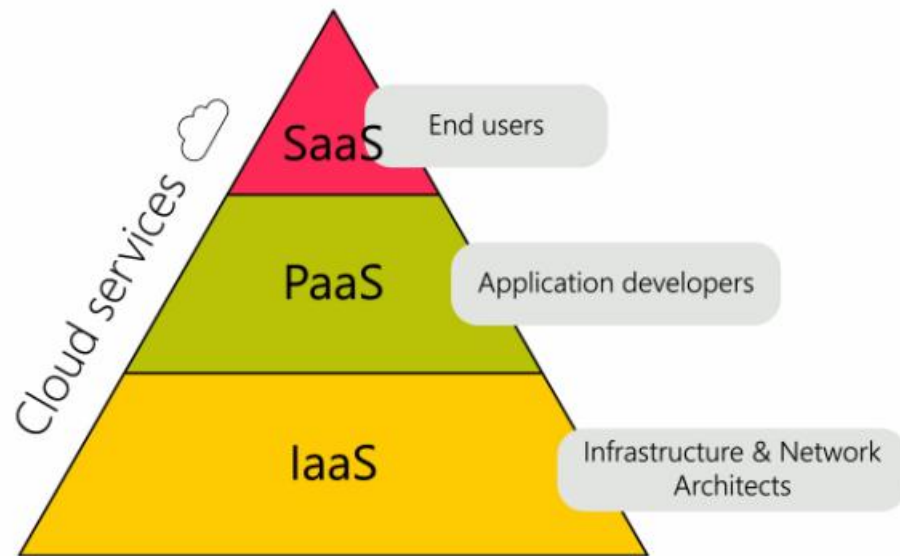


Рисунок 4.1 – Піраміда хмарних сервісів

– Інфраструктура як послуга (IaaS): фундамент хмарної архітектури. Провайдер управляє фізичним залізом, а клієнт отримує повний контроль над віртуальними серверами, мережами та сховищами. У контексті утилізації ресурсів IaaS надає максимальну гнучкість: інженери можуть точно підбирати конфігурації потужностей під поточні завдання, ліквідуючи капітальні витрати (CapEx). Приклади: Amazon EC2, Google Compute Engine.

– Платформа як послуга (PaaS): більш спеціалізований рівень, що надає готове середовище для розробки, тестування та розгортання коду без необхідності адмініструвати ОС чи системне ПЗ. З точки зору УМ, PaaS суттєво спрощує контроль утилізації, оскільки хмара автоматично балансує внутрішні ресурси платформи під навантаження програми. Приклади: AWS Elastic Beanstalk, Google App Engine.

– Програмне забезпечення як послуга (SaaS): готові програмні рішення, що надаються в оренду через веб-браузер. Усі турботи щодо масштабування, безпеки

та ефективності використання заліза повністю лежать на постачальнику послуги, звільняючи клієнта від інфраструктурного менеджменту. Приклади: Microsoft Office 365, Salesforce [10, 11].

4.2 Профільний аналіз сервісів AWS крізь призму Utilization Management

Для розгортання високонавантаженої МІС (наприклад, Hospital Management System) та моніторингу ефективності її ресурсів використовується стек базових сервісів AWS, кожен з яких має свої особливості утилізації:

- Amazon EC2 (Elastic Compute Cloud): віртуальні сервери, що забезпечують масштабовану обчислювальну потужність. Аналіз утилізації EC2 є критичним: моніторинг завантаження процесора (CPU) та оперативної пам'яті дозволяє вчасно проводити rightsizing (зміну типу інстансу на менший/дешевший або більший). Підтримка різних моделей покупки (наприклад, Spot Instances для тимчасових завдань зі знижкою до 90%) дозволяє гнучко керувати бюджетом.

- Amazon S3 (Simple Storage Service): об'єктне сховище з найвищим рівнем доступності та безпеки. Управління використанням в S3 реалізується через Lifecycle Policies (політики життєвого циклу). Наприклад, важкі медичні зображення PACS або старі логи автоматично переміщуються з дорогого шару S3 Standard до ультрадешевого архіву S3 Glacier, оптимізуючи вартість утилізації дискового простору.

- Amazon RDS (Relational Database Service): керована служба реляційних баз даних. Вона знімає з ІТ-команди завдання рутинного адміністрування (бек-апи, патчі). Управління утилізацією тут полягає в аналізі метрик зчитування/запису (IOPS) та використанні реплік для читання (Read Replicas), щоб розвантажити основну БД під час масових запитів.

– Amazon VPC (Virtual Private Cloud): логічно ізольована віртуальна мережа. Дозволяє гнучко налаштовувати підмережі, маршрутизацію та контролювати утилізацію мережевої пропускної здатності, захищаючи систему від зовнішніх загроз.

– AWS Lambda: Безсерверний (Serverless) обчислювальний сервіс. Код запускається суворо у відповідь на події (наприклад, завантаження файлу пацієнта) і масштабується автоматично. Це еталон ефективності утилізації ресурсів, оскільки коефіцієнт корисного використання дорівнює 100% – оплата здійснюється виключно за мілісекунди реальної роботи коду, а простій сервера коштує \$0.

– Amazon ECS (Elastic Container Service): служба оркестровки контейнерів для керування мікросервісами платформи. Дозволяє упакувати модулі системи в ізольовані контейнери та запускати їх або на базі EC2, або в безсерверному середовищі AWS Fargate, максимізуючи щільність використання (утилізацію) заліза.

– AWS ELB (Elastic Load Balancing): балансувальник, який автоматично розподіляє вхідний веб-трафік між серверами. ELB є ключовим компонентом УМ, оскільки він запобігає перевантаженню одних інстансів та простоюванню інших, забезпечуючи рівномірну утилізацію всієї інфраструктури [1, 9].

4.3 Концепція Infrastructure as Code (IaC) як інструмент фінансового контролю

Практика «Інфраструктура як код» (IaC) трансформує процес управління ресурсами, кодифікуючи залізо, мережі та бази даних у вигляді програмного забезпечення. Замість ручного налаштування інфраструктури через консоль, інженери використовують декларативні описові мови.

У контексті Utilization Management та FinOps підхід IaC забезпечує фундаментальні переваги:

- Швидкість та усунення Cloud Waste: автоматичне створення та знищення середовищ (наприклад, тестових «пісочниць» для розробників) за лічені хвилини. Написаний сценарій може містити код, який автоматично вимикає невикористовувані ресурси у позаробочий час, оптимізуючи коефіцієнт утилізації.

- Прозорість витрат: оскільки вся інфраструктура зафіксована в коді, команда чітко бачить кожен запущений компонент. Це спрощує процес аудиту витрат та дозволяє уникати «забутих» або «осиротілих» серверів, які таємно споживають бюджет.

- Стандартизоване тестування: команди контролю якості та безпеки отримують ідентичні копії робочого середовища для перевірок, що прискорює життєвий цикл розробки системи (CI/CD) [10].

4.4 Інструмент декларативного управління інфраструктурою Terraform

Terraform (від компанії HashiCorp) є провідним open-source інструментом IaC, що використовує декларативний стиль та принцип ідемпотентності. Це означає, що користувач описує у конфігураційних файлах фінальний (бажаний) стан інфраструктури, а Terraform самостійно порівнює його з поточним станом і виконує лише необхідні зміни, уникаючи дублювання дій.

Взаємодія з хмарними API здійснюється через спеціальні модулі — Провайдери (Providers). Завдяки цьому інструмент є портативним і дозволяє керувати мультихмарними архітектурами за допомогою єдиної мови HCL (HashiCorp Configuration Language).

Основний триетапний робочий процес Terraform (Workflow):

- Запис (Write): Опис необхідних ресурсів (VPC, EC2, RDS, ELB) та зв'язків між ними у файлах конфігурації.

- План (Plan): Генерація плану виконання. Terraform аналізує код і показує інженеру точний перелік дій: які ресурси будуть створені, модифіковані чи видалені. Це головний етап пре-верифікації утилізації.
- Застосувати (Apply): Безпосередній деплой інфраструктури в AWS у строгому порядку з урахуванням усіх архітектурних залежностей [11].

4.5 Практична реалізація та деплой інфраструктури МІС в AWS

У межах даного дослідження для демонстрації ефективності управління хмарними ресурсами було реалізовано розгортання веб-орієнтованої інформаційної системи керування лікарнею (Hospital Management System).

Базовий набір команд Terraform для управління життєвим циклом та утилізацією розгорнутого стенду містить:

- `terraform init` – ініціалізація робочого простору та автоматичне завантаження провайдера AWS;
- `terraform plan` – попередній технічний та фінансовий аудит запланованих хмарних потужностей;
- `terraform apply` – активація інфраструктури та запуск сервісів у хмарі;
- `terraform destroy` – повне та миттєве видалення всіх створених ресурсів AWS після завершення сесії використання стенду. Це ключова команда з точки зору економії, що гарантує відсутність нарахування плати за сервери після завершення дослідження [11].

Повний вихідний код розгорнутої системи доступний у відкритому репозиторії GitHub, технічна архітектура зв'язків інфраструктури AWS винесена у додаток А, а лістинг конфігураційного HCL-коду – у додатку Б.

У результаті успішного виконання сценарію `terraform apply` у хмарі AWS було піднято та налаштовано повний стек системи. Результат успішної ініціалізації веб-

інтерфейсу (вікно авторизації) та робочий простір запусненої програми відображено на рис. 4.3 та рис. 4.4 відповідно [6, 9].

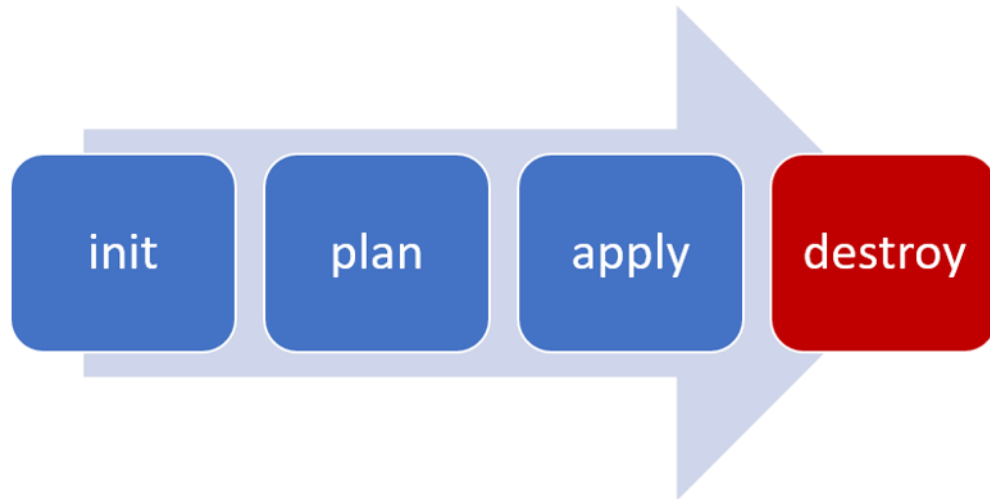


Рисунок 4.2 – Уніфікований робочий процес керування ресурсами в Terraform

The screenshot shows a web application interface for 'GLOBAL HOSPITALS'. The header includes a logo with a plus sign and the text 'GLOBAL HOSPITALS', and navigation links for 'HOME', 'ABOUT US', and 'CONTACT'. The main content area features a 'Welcome' message with a rocket icon on the left. On the right, there is a 'Register as Patient' form. The form has three tabs: 'Patient' (selected), 'Doctor', and 'Admin'. The registration fields include 'First Name *', 'Last Name *', 'Your Email *', 'Your Phone *', 'Password *', and 'Confirm Password *'. Below these fields are radio buttons for 'Male' and 'Female', and a link that says 'Already have an account?'. A 'Register' button is located at the bottom right of the form.

Рисунок 4.3 – Вікно реєстрації розгорнута в AWS

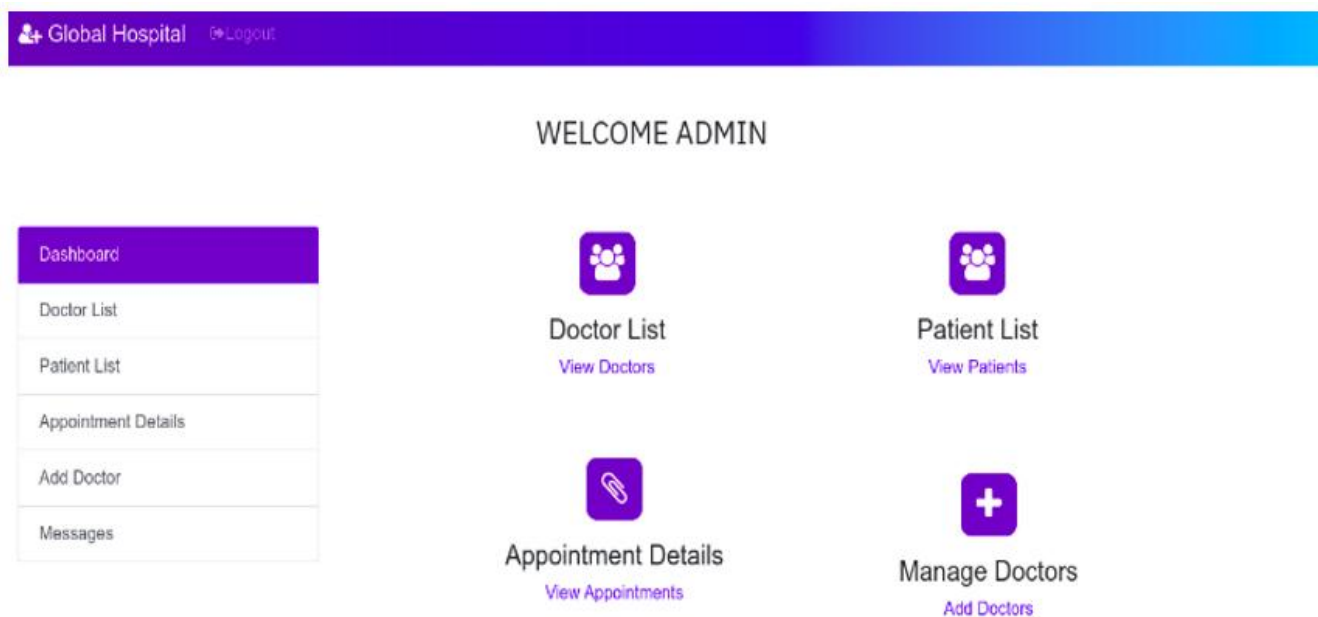


Рисунок 4.4 – Робоча програма розгорнута в AWS

ВИСНОВКИ ДО РОЗДІЛУ 4

У четвертому розділі проведено детальний аналіз технологічного потенціалу хмарної платформи AWS та методів оптимізації утилізації її ресурсів через підхід Infrastructure as Code. Доведено, що використання сервісів AWS (таких як EC2, S3, RDS, Lambda) у поєднанні з моделлю оплати за фактом споживання дозволяє гнучко підлаштовувати потужність інфраструктури під поточні потреби бізнесу. Встановлено, що впровадження інструменту декларативного управління Terraform кардинально підвищує ефективність Utilization Management. Завдяки автоматизації процесів деплою (terraform apply) та миттєвого очищення середовища (terraform destroy) повністю нівелюється проблема «хмарного марнотратства» (Cloud Waste), забезпечується 100% прозорість витрат, виключається людський фактор та гарантується точна утилізація виділених обчислювальних квот. Практична працездатність підходу підтверджена успішним розгортанням стенду інформаційної системи Hospital Management System в автоматичному режимі.

ВИСНОВКИ

У кваліфікаційній роботі розв'язано актуальне науково-практичне завдання, що полягає в теоретичному обґрунтуванні, аналізі та практичній реалізації методів автоматизованого управління використанням (утилізацією) обчислювальних ресурсів хмарної інфраструктури для комплексних інформаційних систем. Обґрунтовано роль Utilization Management у хмарі, встановлено, що в рамках сучасної методології FinOps управління утилізацією є базовим елементом контролю витрат. Визначено, що ефективна програма UM повинна базуватися на трьох фундаментальних принципах: ступеневому виділенні потужностей, обов'язковій попередній авторизації архітектурних змін та жорсткому контролю виділених лімітів (квот). Це дозволяє збалансувати технічні потреби інженерів та фінансові вимоги бізнесу.

Під час виконання роботи визначено архітектурні особливості комплексних систем на основі аналізу систем рівня eHealth доведено необхідність переходу від локальних віртуальних серверів до хмарних віртуальних дата-центрів. Розподілена модульна архітектура таких платформ генерує різнотипні навантаження (транзакційні, аналітичні, обробку Big Data), що вимагає динамічного розподілу ресурсів для забезпечення відмовостійкості та високої доступності даних відповідно до стандартів безпеки.

Проведено порівняльний аналіз моделей розгортання на прикладі реальних МІС (Doctor Eleks, EMCiMED, Helsi) досліджено їхні інфраструктурні патерни. Доведено беззаперечну перевагу хмарної моделі обслуговування (SaaS) над локальною (On-Premise) в контексті управління ресурсами. Модель OpEx (оплата за фактом споживання) повністю усуває проблему простою дорогого фізичного обладнання під час спадів трафіку та забезпечує миттєву масштабованість.

У роботі досліджено інфраструктурний потенціал платформи AWS, здійснено профільний аналіз базових сервісів екосистеми AWS (EC2, S3, RDS, Lambda, ECS, ELB) крізь призму ефективності утилізації заліза. Виявлено, що безсерверна модель (Serverless на прикладі AWS Lambda) забезпечує еталонний коефіцієнт корисного використання, де оплата здійснюється суворо за мілісекунди роботи коду, повністю виключаючи витрати на простій.

Також у роботі реалізовано автоматизацію утилізації за допомогою Terraform, а саме обґрунтовано переваги підходу Infrastructure as Code (IaC). На практиці продемонстровано розгортання веб-орієнтованої системи Hospital Management System в інфраструктурі AWS за допомогою декларативного інструменту Terraform. Доведено, що використання команд життєвого циклу інфраструктури (зокрема, швидкого аудиту через `terraform plan` та повного видалення стенду через `terraform destroy`) повністю нівелює «хмарне марнотратство» (Cloud Waste), виключає людський фактор та забезпечує 100% прозорість інфраструктурних витрат.

Результати дослідження, сформовані архітектурні підходи та розроблений Terraform-код мають універсальний інженерний характер. Створені декларативні шаблони та методики моніторингу Utilization Management можуть бути успішно масштабовані та застосовані у таких перспективних напрямках: у сфері корпоративного сектора та E-commerce, у муніципальних та державних реєстрах, у банківських платформах, в освітніх технологіях.

ПЕРЕЛІК ПОСИЛАНЬ

1. Gray B. H., Field M. J. Controlling Costs and Changing Patient Care? The Role of Utilization Management. Washington (DC): National Academies Press, 1989. 328 p.
2. AWS Well-Architected Framework: Cost Optimization Pillar [Electronic resource] / Amazon Web Services. URL: <https://docs.aws.amazon.com/wellarchitected/latest/cost-optimization-pillar/welcome.html>
3. FinOps Framework: Cloud Utilization and Efficiency [Electronic resource] / FinOps Foundation. URL: <https://framework.finops.org/>
4. Електронна система охорони здоров'я в Україні (eHealth): офіційний портал [Електронний ресурс] / Міністерство охорони здоров'я України. URL: <https://ehealth.gov.ua/>
5. Про електронну систему охорони здоров'я: Закон України від 19.10.2017 р. № 2168-VIII (із змінами та доповненнями) [Електронний ресурс] / Верховної Рада України. URL: <https://zakon.rada.gov.ua/>
6. Медичні інформаційні системи: огляд можливостей і приклади використання [Електронний ресурс] // Evergreens. URL: <https://evergreens.com.ua/ua/articles/medical-information-systems.html>
7. Hospital Management System Web Application Open-Source Repository [Electronic resource] / GitHub. URL: <https://github.com/kishan0725/Hospital-Management-System>
8. What is Cloud Computing? AWS Cloud Definition [Electronic resource] / Amazon Web Services. URL: <https://aws.amazon.com/what-is-cloud-computing/>
9. AWS Documentation: Core Services Overview (EC2, S3, RDS, VPC) [Electronic resource] / Amazon Web Services. URL: <https://docs.aws.amazon.com/>

10. What is Infrastructure as Code (IaC)? [Electronic resource] / Red Hat Inc. URL: <https://www.redhat.com/en/topics/automation/what-is-infrastructure-as-code-iac>

11. Infrastructure as Code with Terraform: AWS Get Started [Electronic resource] / HashiCorp Developer. URL: <https://developer.hashicorp.com/terraform/tutorials/aws-get-started/infrastructure-as-code>