Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)
Кафедра комп'ютерних наук
(повна назва кафедри)

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття освітнього ступеня

бакалавр
(назва освітнього ступеня)

на тему: Розробка системи захисту даних в галузі охорони здоров'я в умовах цифрової трансформації

Виконав: студент IV курсу, групи СНз-41
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)

_____ Середюк В.П.
(підпис) (прізвище та ініціали)

Керівник _____ Небесний Р.М.
(підпис) (прізвище та ініціали)

Нормоконтроль _____ Шимчук Г.В.
(підпис) (прізвище та ініціали)

Завідувач кафедри _____ Боднарчук І.О.
(підпис) (прізвище та ініціали)

Рецензент _____ Тимощук Д.І.
(підпис) (прізвище та ініціали)

Тернопіль

2025

Міністерство освіти і науки України
Тернопільський національний технічний університет імені Івана Пулюя

Факультет комп'ютерно-інформаційних систем і програмної інженерії
(повна назва факультету)

Кафедра комп'ютерних наук
(повна назва кафедри)

ЗАТВЕРДЖУЮ
Завідувач кафедри
Боднарчук І.О.
(підпис) (прізвище та ініціали)
« __ » _червня_ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

на здобуття освітнього ступеня Бакалавр
(назва освітнього ступеня)

за спеціальністю 122 Комп'ютерні науки
(шифр і назва спеціальності)

Студенту Середюку Вадиму Петровичу
(прізвище, ім'я, по батькові)

1. Тема роботи Розробка системи захисту даних в галузі охорони здоров'я в умовах цифрової трансформації

Керівник роботи Небесний Руслан Михайлович, доктор філософії, доцент кафедри КН
(прізвище, ім'я, по батькові, науковий ступінь, вчене звання)

Затверджені наказом ректора від « *07* » *травня* 2025 року № *4/7-449*

2. Термін подання студентом завершеної роботи 19 червня 2025 р.

3. Вихідні дані до роботи Технічне завдання, літературні джерела

4. Зміст роботи (перелік питань, які потрібно розробити)
Вступ. 1. Аналіз предметної області та постановка завдання. 2. Проєктування інформаційної системи щодо захисту даних в галузі охорони здоров'я. 3. Реалізація та тестування інформаційної системи для захисту даних в галузі охорони здоров'я.
4. Безпека життєдіяльності, основи охорони праці. Висновки. Перелік джерел. Додатки.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень, слайдів)
1. Титульний слайд. 2. Актуальність теми. 3. Мета та завдання. 4. Об'єкт та предмет дослідження. 5. Наукова новизна та практичне значення. 6. Аналіз загроз. 7. Цифрова стійкість. 8. Використання блокчейну. 9. Вибір технологій. 10. Архітектура системи. 11. Шифрування. 12. Управління доступом. 13. Інтерфейс користувача. 14. Тестування.

15. Висновки. 16. Дякую за увагу!

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв
Безпека життєдіяльності, основи охорони праці	Сенчишин Віктор Степанович, к.т.н., доцент кафедри МТ	02.06.2025	05.06.2025

7. Дата видачі завдання 03 березня 2025 р.

КАЛЕНДАРНИЙ ПЛАН

№ з/п	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1.	Ознайомлення з завданням до кваліфікаційної роботи	04.03.2025	*Виконано*
2.	Підбір та опрацювання літературних джерел по темі кваліфікаційної роботи	05.03.2025-15.03.2025	*Виконано*
3.	Виконання дослідження щодо захисту даних в галузі охорони здоров'я в умовах цифрової трансформації	16.03.2025-16.04.2025	*Виконано*
	Розроблення алгоритмів сортування зашифрованих даних		
4.	Оформлення розділу «Аналіз предметної області та постановка завдання»	17.04.2025-01.05.2025	*Виконано*
5.	Оформлення розділу «Проєктування інформаційної системи щодо захисту даних в галузі охорони здоров'я»	02.05.2025-15.05.2025	*Виконано*
6.	Оформлення розділу «Реалізація та тестування інформаційної системи для захисту даних в галузі охорони здоров'я»	16.05.2025-01.06.2025	*Виконано*
7.	Виконання завдання до підрозділу «Безпека життєдіяльності»	02.06.2025-05.06.2025	*Виконано*
8.	Виконання завдання до підрозділу «Основи охорони	02.06.2025-05.06.2025	*Виконано*
	праці»		
9.	Оформлення кваліфікаційної роботи	06.06.2025-08.06.2025	*Виконано*
10.	Нормоконтроль	09.06.2025-11.06.2025	*Виконано*
11.	Перевірка на плагіат	12.06.2025	*Виконано*
12.	Попередній захист кваліфікаційної роботи	13.06.2025	*Виконано*
13.	Захист кваліфікаційної роботи	19.06.2025	

Студент _____ Середюк В.П.
(підпис) (прізвище та ініціали)

Керівник роботи _____ Небесний Р.М,
(підпис) (прізвище та ініціали)

АНОТАЦІЯ

Розробка системи захисту даних в галузі охорони здоров’я в умовах цифрової трансформації // Кваліфікаційна робота освітнього рівня «Бакалавр» // Середюк Вадим Петрович // Тернопільський національний технічний університет імені Івана Пулюя, факультет комп’ютерно-інформаційних систем і програмної інженерії, кафедра комп’ютерних наук, група СНз-41 // Тернопіль, 2025 // С. 66, рис. – 26, табл. – 8, додат. – 4, бібліогр. – 53.

Ключові слова: захист даних, охорона здоров’я, гомоморфне шифрування, blockchain, web-система, PostgreSQL, RBAC, цифрова трансформація.

Кваліфікаційна робота присвячена дослідженню проблеми підвищення кіберстійкості медичних закладів шляхом розроблення безпечної, масштабованої веб-системи захисту даних у сфері охорони здоров’я в умовах цифрової трансформації.

У першому розділі кваліфікаційної роботи проаналізовано сучасний стан захисту електронних медичних даних, еволюцію кіберзагроз, концепцію цифрової стійкості та вимоги стандартів ISO/IEC 27001 і HIPAA.

У другому розділі обґрунтовано вибір стеку (Django + React, Docker/K8s, FHIR), криптоалгоритмів AES-256-GCM, CKKS та PostgreSQL і сформовано вимоги системи.

У третьому розділі описано реалізацію мікросервісної архітектури з gRPC та mTLS, React-SPA-інтерфейсу й алгоритмів обробки зашифрованих даних; проаналізовано результати функціонального й навантажувального тестування та оцінено відповідність вимогам продуктивності й безпеки.

Об’єкт дослідження: процес захисту медичних даних у закладах охорони здоров’я.

Предмет дослідження: методи, моделі й програмно-технічні засоби веб-системи захисту медичних даних із використанням сучасних криптографічних алгоритмів, блокчейна та RBAC.

ANNOTATION

Development of a Data Protection System in the Healthcare Sector during Digital Transformation // Qualification work of the educational level «Bachelor» // Serediuk Vadym // Ternopil Ivan Pulyu National Technical University, Computer and Information Systems and Software Engineering Faculty, Computer Sciences Department, group SNz-41 // Ternopil, 2025 // P. 66, fig. – 26, tabl. – 8, annexes. – 4, references – 53.

Keywords: data protection, healthcare, homomorphic encryption, blockchain, web-based system, PostgreSQL, RBAC (role-based access control), digital transformation.

The qualification thesis is devoted to increasing the cyber-resilience of healthcare institutions by developing a secure, scalable web-based data-protection system for the healthcare sector amid digital transformation.

Chapter 1 analyses the current state of electronic medical data protection, the evolution of cyber-threats, the concept of digital resilience, and the requirements of ISO/IEC 27001 and HIPAA.

Chapter 2 substantiates the choice of the technology stack (Django + React, Docker / Kubernetes, FHIR), the AES-256-GCM and CKKS cryptographic algorithms, and PostgreSQL, and formulates the system requirements.

Chapter 3 implements a gRPC + mTLS microservice architecture with a React SPA, tests it, and evaluates its security and performance.

Object of research: the process of protecting medical data in healthcare institutions.

Subject of research: methods, models, and hardware-software tools for a web-based medical data-protection system using modern cryptographic algorithms, blockchain technology, and RBAC.

ПЕРЕЛІК УМОВНИХ СКОРОЧЕНЬ І ТЕРМІНІВ

AES (англ. Advanced Encryption Standard) – симетричний алгоритм шифрування.

API (англ. Application Programming Interface) – інтерфейс прикладного програмування.

CKKS (англ. Cheon–Kim–Kim–Song) – схема гомоморфного шифрування.

CRUD (англ. Create, Read, Update, Delete) – базові операції з даними.

FHIR (англ. Fast Healthcare Interoperability Resources) – стандарт обміну медичними даними.

gRPC (англ. Google Remote Procedure Call) – фреймворк віддалених викликів процедур.

HIPAA (англ. Health Insurance Portability and Accountability Act) – закон США про захист медичних даних.

K8s (англ. Kubernetes) – система оркестрації контейнерів.

mTLS (англ. mutual Transport Layer Security) – взаємна аутентифікація TLS.

PostgreSQL (англ. Postgre SQL) – об’єктно-реляційна система керування базами даних.

RBAC (англ. Role-Based Access Control) – розмежування доступу за ролями.

СКБД – Система керування базами даних.

ШІ – Штучний інтелект.

ЗМІСТ

Вступ

Актуальність теми. В умовах цифровізації охорони здоров’я медичні заклади оперують зростаючими обсягами електронних медичних записів, телемедичних консультацій [1] і даних від пристроїв. Будь-яке порушення конфіденційності чи цілісності таких даних може призвести до втрати довіри пацієнтів, юридичної відповідальності та значних фінансових збитків [2], [3]. Водночас поява штучного інтелекту й хмарних сервісів [4] підвищує ризик кіберзагроз, що робить питання захисту медичної інформації стратегічно важливим для держави й суспільства. Тому розроблення комплексної системи, яка поєднує криптографічні методи, політики доступу та стандарти безпеки, є актуальним та затребуваним напрямом досліджень у галузі комп’ютерних наук і медицини.

Мета і задачі дослідження. Метою даної кваліфікаційної роботи освітнього рівня «Бакалавр» є підвищення рівня захищеності медичних даних шляхом розроблення безпечної, функціональної та масштабованої інформаційної системи для закладів охорони здоров’я.

Для досягнення поставленої мети необхідно виконати такі завдання:

– проаналізувати сучасний стан наукових досліджень у сфері захисту медичних даних;

– обґрунтувати вибір технологічного стеку і методу шифрування, що відповідають вимогам конфіденційності, цілісності та доступності;

– спроєктувати структуру бази даних з підтримкою гомоморфного шифрування й політик RBAC;

– реалізувати клієнтську та серверну частини веб-системи з можливістю сортування зашифрованих двовимірних масивів;

– забезпечити диференційований доступ різних категорій користувачів (лікар, медсестра, адміністратор, пацієнт);

– протестувати продуктивність, стійкість до кібератак та відповідність вимогам.

Практичне значення одержаних результатів. Отримані результати можуть бути використані для впровадження повнофункціональної системи захисту даних у реальному медичному закладі або як шаблон для проєктування аналогічних систем. Запропоноване рішення демонструє, що сучасні криптографічні алгоритми, зокрема гомоморфне шифрування CKKS, можуть поєднуватися із традиційними вебтехнологіями (Django, React) без значних втрат продуктивності, що відкриває можливості для малого та середнього медичного бізнесу впроваджувати високий рівень безпеки без надмірних інвестицій.

РОЗДІЛ 1. АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ ТА ПОСТАНОВКА ЗАВДАННЯ

Захист даних в галузі охорони здоров'я

У епоху цифрової трансформації сфера охорони здоров'я зазнає суттєвих змін у зв'язку з інтеграцією сучасних технологій. Електронні медичні картки (ЕМК), телемедицина та інформаційні системи охорони здоров'я стають невід'ємними складовими сучасного надання медичних послуг, обіцяючи вищу ефективність, кращий догляд за пацієнтами та кращі клінічні результати [5]. Ґрівз та інші описують цифрову медицину як мультидисциплінарну галузь, що має на меті підвищення ефективності моніторингу пацієнтів, діагностики, лікування, профілактики, реабілітації та надання довготривалої медичної допомоги [6]. Згідно з останніми даними Всесвітньої організації охорони здоров'я, цифрова медицина є стрімко зростаючою галуззю медицини, що суттєво впливає на підвищення якості та ефективності медичних послуг, зниження витрат на лікування для пацієнтів і розвиток клінічних досліджень.

Однак цифрова трансформація має й серйозні недоліки: кібернапади на медичні заклади стають все більш поширеними та складними, тож захист даних пацієнтів перетворився на серйозну проблему. Цей факт підкреслює необхідність створення надійного захисного механізму для охорони конфіденційної інформації пацієнтів від онлайн-загроз [7]. Більше того, кіберзлочинці розглядають дані пацієнтів, які включають фінансову, медичну та особисту інформацію, як надзвичайно цінний ресурс. Витоки даних мають серйозні наслідки для галузі охорони здоров'я, зокрема втрати прибутку, труднощі в роботі, загрозу безпеці пацієнтів та підрив довіри суспільства до медичних установ [8]. Гучні інциденти останнього часу привернули увагу до вразливості систем охорони здоров'я та нагальної потреби у впровадженні ефективних рішень у сфері кібербезпеки для захисту від атак з використанням програм-вимагачів на лікарні та витоків даних із медичних мереж. Такі злами можуть загрожувати життю людей, перериваючи надання життєво необхідних медичних

послуг і розкриваючи особисту інформацію недобросовісним особам. Отже, для закладів охорони здоров’я забезпечення безпеки та цілісності даних пацієнтів є не лише технологічною необхідністю, а й моральним обов’язком [9]. На рисунку 1.1 наведено типові загрози даних в галузі охорони здоров’я.

Рисунок 1.1 – Типові загрози даних в галузі охорони здоров’я

З іншого боку, нинішній стан кібербезпеки в охороні здоров’я свідчить про динамічний і складний ландшафт загроз. Кіберзлочинці використовують усе більш витончені стратегії, щоб скористатися вразливими місцями в медичних системах. Ці стратегії часто базуються на застарілих стандартах безпеки, недостатній підготовці персоналу та недотриманні нормативних вимог. Типові кіберзагрози включають атаки з використанням програм-вимагачів, фішингові кампанії та внутрішні загрози, кожна з яких створює унікальні труднощі для захисту даних пацієнтів. Наприклад, атаки програм-вимагачів блокують діяльність медичних установ шляхом шифрування важливої інформації й

вимагають великі викупи за її розблокування. У свою чергу, фішингові атаки можуть обманом змусити медичних працівників розкрити конфіденційні дані або надати несанкціонований доступ до захищених мереж. Внутрішні загрози становлять небезпеку зсередини організації – як навмисну, так і ненавмисну – і суттєво ускладнюють загальну ситуацію з безпекою [7].

Еволюція кіберзагроз у сфері охорони здоров'я

Технологічний прогрес і зростаюча витонченість кіберзлочинців знаходять своє відображення в складній історії розвитку кіберризиків в галузі охорони здоров'я.

Медичні компанії були одними з перших, хто досить повільно, як і багато інших галузей, почав впроваджувати широкі заходи з кібербезпеки. Спочатку основними об'єктами кібератак були ланки мережевої безпеки. Кіберзлочинці, які намагалися отримати конфіденційні персональні та медичні дані, виявили, що системи охорони здоров'я стали більш привабливими цілями, коли вони перейшли від паперових записів до електронних медичних карток (ЕМК) [10]. Щоб підвищити ефективність, доступність та якість обслуговування пацієнтів, наприкінці 1990-х - на початку 2000-х років медична галузь також почала оцифровувати медичні картки пацієнтів. Незважаючи на переваги, ця зміна також принесла нові ризики. Перші кіберзагрози часто складалися з простих вірусів і шкідливих програм, призначених для викрадення даних і забезпечення їхньої стійкості. Більшість цих загроз були опортуністичними [11], вони атакували будь-яку систему з недостатнім рівнем безпеки. Організації охорони здоров'я здебільшого застосовували реактивний підхід, зосереджуючись на налаштуванні брандмауерів та антивірусних програм для захисту від цих вторгнень [12]. На рисунку 1.2 наведено еволюцію кіберзагроз у сфері охорони здоров'я.

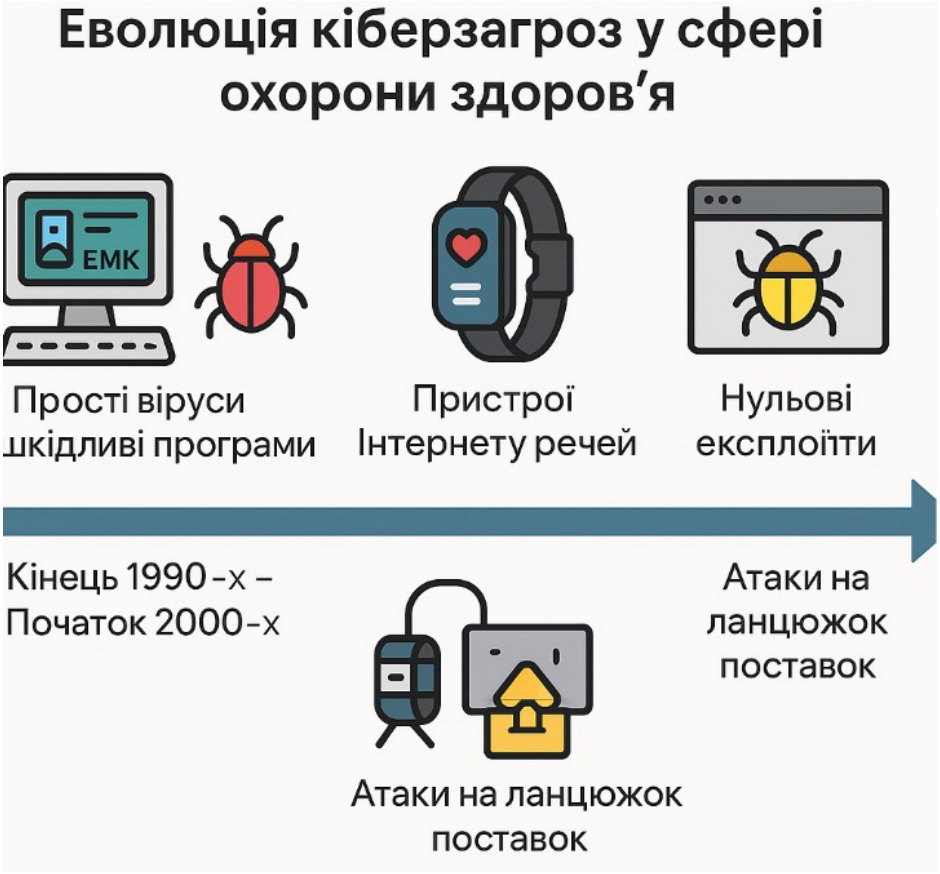

Рисунок 1.2 – Еволюція кіберзагроз у сфері охорони здоров'я: від мережевих атак до сучасних ризиків, пов'язаних з ШІ, IoT та фішингом

Крім того, за останні десять років медична галузь інтегрувала передові технології, що ще більше ускладнило середовище кібербезпеки. Використання штучного інтелекту (ШІ), машинного навчання (ML) та давачів Інтернету речей (IoT) покращило обслуговування пацієнтів, але водночас створило нові загрози. Пристроям Інтернету речей часто бракує надійних заходів безпеки, що робить їх першочерговими цілями для хакерів. Прикладами таких давачів є натільний моніторинг здоров'я та «розумне» медичне обладнання. Покращуючи діагностичні та терапевтичні можливості, штучний інтелект і машинний інтелект можуть нести додаткову небезпеку, якщо алгоритми та базові дані будуть пошкоджені [13]. Крім того, атаки на ланцюжки постачання програмного забезпечення стають ще однією точкою входу для кіберзловмисників, оскільки вони призводять до отримання доступу до бекдорів, встановлення шкідливого програмного забезпечення, простою застосунків і витоку даних, таких як паролі

або приватна інформація [14]. Експлойти нульового дня мають на меті скористатися недоліками програмного забезпечення, про які виробник не знає, залишаючи системи вразливими доти, доки не буде створено виправлення. І навпаки, атаки на ланцюжки поставок впливають на всіх користувачів нижче, компрометуючи апаратне або програмне забезпечення у джерелі. Потенційні масштаби і серйозність таких атак були виявлені під час атаки вірусу Solar Winds у 2020 році, яка вплинула на кілька галузей, включаючи сектор охорони здоров'я [15].

Стратегія кібербезпеки в галузі охорони здоров'я повинна була змінитися у відповідь на ці постійно мінливі загрози, ставши більш проактивною і всеосяжною. Традиційні укріплення по периметру вже не є достатніми. Багаторівневий підхід до безпеки, що включає шифрування, безпечні обмеження доступу, системи виявлення та запобігання вторгненням і безперервний моніторинг, стає все більш поширеним в медичних установах. Фокус змістився на комплексну стратегію, яка включає людей, політику, процедури і технології [5]. Заходи захисту тепер повинні включати навчання з питань кібербезпеки. Працівники галузі охорони здоров'я часто першими виявляють і реагують на сумнівну діяльність, що може значно знизити ризик. Організаційна культура безпеки розвивається за допомогою регулярних тренінгів та імітаційних вправ, таких як фішинг-тести [8].

Однак командна робота та обмін інформацією стають все більш важливими. Кіберзагрози постійно розвиваються, і жодна компанія не може захиститися від них самотужки. Такі загальногалузеві зусилля, як Центр обміну та аналізу медичної інформації (H-ISAC), полегшують медичним компаніям обмін розвідданими про загрози та найкращими практиками. Ця спільна стратегія покращує можливості колективного захисту і дозволяє швидше реагувати на нові загрози [16].

Цифрова стійкість у галузі охорони здоров’я

Цифрова трансформація сектору охорони здоров’я підвищила його чутливість до кібератак, які визнані однією з найважливіших суспільних та організаційних загроз з точки зору ймовірності та наслідків. Впровадження цифрових технологій у секторі охорони здоров’я прискорилося завдяки суттєвим змінам у нинішньому сценарії. Зараз медичні компанії отримують і зберігають дедалі більшу кількість конфіденційних даних про людей, інфраструктуру та мережі постачання. Будь-яка подія, що впливає на ці системи, може мати значний вплив на стратегію та діяльність медичної організації, а також на зацікавлених осіб [17].

Однак ініціативи з оцифрування створюють як загрозу, так і можливість для стійкості системи охорони здоров’я. Йованович та ін. [18] зазначають, що трансформаційні зміни на нижчому рівні можуть сприяти підвищенню стійкості на вищому рівні. Цей зв’язок між стійкістю та цифровою трансформацією вивчається далі в наступних розділах з точки зору кібербезпеки. На основі огляду літератури можна виділити кілька аспектів, які характеризують ідею цифрової стійкості в охороні здоров’я. Автори по-різному оцінюють стійкість системи охорони здоров’я як унікальну, тобто дискретну сферу стійкості [19]. У цьому середовищі з’являється кілька загальних рис і таксономій. Наприклад, в існуючій літературі розрізняють проактивні та реактивні вимірювання, або, в більш широкому сенсі, елементи ідеї, орієнтовані на випередження та реагування. Крім того, існує кілька способів охарактеризувати риси стійких систем. Вони включають ширші якості, такі як:

- готовність;
- надійність;
- відновлення/адаптація.

Окрім того можна виділити і системні функції, такі як:

- абсорбційна;
- адаптивна;

 трансформаційна здатність [18].

На рисунку 1.3 наведено якості та функції цифрової стійкості в галузі охорони здоров’я.

Рисунок 1.3 – Якості та функції цифрової стійкості в галузі охорони здоров’я

Крім того, визначення таких систем, що базуються на результатах, підкреслюють довіру громади, безпеку, гнучкість і безперервність надання послуг, а також право власності в менш абстрактних термінах. На думку Бланше та ін. [16], ще однією помітною подією в охороні здоров’я є категоризація «стійкості» як межового терміна, що долає бар’єри між науковим і громадським/політичним дискурсом. Це відповідає ширшій тенденції в міждисциплінарній роботі, на яку вказувала Маньєна [20]. Це підкреслює, наскільки важливо збалансувати деталі та абстракції при моделюванні стійкості системи охорони здоров’я. Дослідження підкреслює роль, яку відіграють культура і менеджмент у зміцненні стійкості системи охорони здоров’я, ще більше пов’язуючи цю концепцію з іншими сферами організаційної

стійкості [21]. Ці результати не підтверджуються ні процесом відбору учасників, ні розробкою дизайну дослідження [22]. Крім того, цифрова стійкість, також відома як кіберстійкість, в галузі охорони здоров'я розглядається як розширення стійкості вищого порядку/інституційної стійкості, відповідно до її системного підходу [23].

Технологія блокчейн для захисту даних в галузі охорони здоров'я

Технологія блокчейн [24] була впроваджена в багатьох сферах як частина інфраструктури деяких бізнесів, які вимагають прозорості, цілісності та надійності [25] з моменту її створення, від початкової криптовалюти до сучасного застосунку на основі блокчейну для Індустрії 5.0 [26-28]. На рисунку 1.4 наведено порівняння між традиційними мережами безпеки та блокчейн-мережами.

Призначення	Традиційна система	Блокчейн-система
Інтерфейс	Не змінюється	Не змінюється
Обмін повідомленнями	Через централізо-ваний сервер	Peer-to-peer (однорпнгово)
Обробка транзакцій	Пакетна або окрема, централізоана	У блоках, децентралізована
Реєстр	Закритий, з одним надавачем доступу	Публічний, незмінний, розподілений

Рисунок 1.4 – Архітектура блокчейн у галузі охорони здоров'я

Впровадження блокчейну в галузі охорони здоров’я вимагає не лише подолання технологічних викликів, але й створення міцного фундаменту з точки зору апаратного забезпечення та мережевої інфраструктури [29]. На щастя, в сучасному будівництві ІКТ великі виробники надають індивідуальні продукти та послуги або професійне обладнання, що значно зменшує труднощі закладу. Мережі блокчейн мають широкий спектр застосування в системах охорони здоров’я, що відповідає вимогам безпеки та конфіденційності для захисту медичних даних пацієнтів від несанкціонованого доступу [30]. Однак, через відсутність експертної розробки протоколів безпеки, системи охорони здоров’я стикаються з багатьма загрозами безпеки, такими як інтероперабельність [31], автентичність, обмін даними, передача медичних даних та мобільне обговорення питань охорони здоров’я [32]. Крім того, через велику кількість розроблених апаратних пристроїв, основною проблемою для блокчейну в охороні здоров’я є впровадження та управління даними.

Дані про здоров’я та медичні дані – це записи про лікування пацієнтів, зібрані в процесі профілактики особистого здоров’я та догляду за пацієнтами. Система є системою спільного використання, яка інтегрує медичні записи з комп’ютерними технологіями [33]. Система складається з трьох модулів: збір даних, безпека даних та обслуговування даних. Модуль збору даних використовується для отримання інформації про стан здоров’я пацієнта, модуль безпеки використовується для створення механізму захисту медичної служби, а модуль обслуговування використовується для задоволення запитів пацієнтів на отримання медичної документації [34].

Відповідно до вищезазначеного технічного рішення, модуль обслуговування даних включає в себе модуль аналізу даних, модуль історії хвороби, модуль медичних рекомендацій та модуль оцінки стану здоров’я пацієнта. Модуль аналізу даних використовується для порівняння зібраних медичних даних та даних про стан здоров’я пацієнтів з даними в блокчейні аналізу медичних закладів [35]. Модуль історичних випадків використовується для зберігання даних про одужання пацієнтів у минулому. Модуль медичних

рекомендацій надає медичним установам платформу для надання пацієнтам рекомендацій щодо реабілітації, а модуль оцінки пацієнтів надає пацієнтам платформу для оцінки медичних установ.

Технологія роботи системи обміну медичними даними на основі блокчейну включає наступні етапи:

1. Під час одужання пацієнта система обміну медичними даними використовується для збору цифрових записів.

2. Безпечне завантаження та захист ЕМК до блокчейну медичного сектору.

3. Порівняйте новостворену ЕМК з даними, що зберігаються в блокчейні, проведіть аналіз даних і завантажте результати.

4. Створіть платформу оцінки пацієнтів та інтегруйте її безпосередньо в медичний блокчейн.

Безпека відрізняється між публічними та приватними блокчейнами залежно від типу блокчейну, і ця різниця є ще одним важливим аспектом безпеки блокчейну, який слід пояснити належним чином.

Публічні блокчейн-мережі є доступними і дозволяють будь-якому користувачеві приєднатися до них, зберігаючи анонімність учасників; приватні блокчейн-мережі потребують більш жорсткого регуляторного та комплаєнс-контролю [36]. Блокчейн можна використовувати для зберігання закодованих особистих медичних записів, доступ до яких мають лише певні особи. Для захисту інформації про пацієнтів у сфері охорони здоров’я потрібні додаткові вимоги, такі як інтероперабельність і передача даних. Процес обміну даними з іншими джерелами називається інтероперабельністю. Розподілені реєстри забезпечують безпечне та конфіденційне управління охороною здоров’я за участю отримувачів медичних послуг, постачальників медичних послуг, страховиків та регулюючих органів [37].

Традиційно дані особистої медичної документації зберігалися та підтримувалися на папері; з появою хмарних розробок записи були перенесені в централізоване сховище. Нова ера охорони здоров’я, також відома як «Здоров’я 5.0», вимагає віддаленого збору даних про стан здоров’я великих груп

користувачів у режимі реального часу за допомогою різних давачів і розумних пристроїв, які використовуються для дистанційного моніторингу здоров'я [38]. Ці дані генеруються у великих кількостях, і їх необхідно відстежувати, передавати та безпечно обробляти. Деякі пацієнти бояться ділитися своєю приватною медичною інформацією з розподіленою мережею, а деякі лікарні також не хочуть ділитися конкретними даними про ліки зі страховими компаніями. Окрім діагностики та лікування, технологія блокчейн може допомогти вирішити низку питань безпеки даних в галузі охорони здоров'я [39], які наведено у таблиці 1.1.

Таблиця 1.1 – Проблеми, які вирішуються в галузі охорони здоров'я за допомогою підходу на основі блокчейну (сформовано на основі [40] та [41])

Розглянуті проблеми	**Підхід до охорони здоров'я**	**Переваги**	**Недоліки**
1	**2**	**3**	**4**
Атаки на безпеку, захист даних	Медична документація та управління даними	Зменшення різних атак на систему охорони здоров'я	Висока пропускна здатність і висока обчислювальна потужність
Атаки на безпеку	Моніторинг пацієнта	Інтеграція з IoT вирішує проблеми безпеки	Мотивація майнінгу та деякі специфічні атаки на блокчейн не викликають занепокоєння
Витік даних	Відстеження наркотиків	Автентифікація та конфіденційність даних, підвищення гнучкості системи	Складний механізм відстеження наркотиків

Продовження таблиці 1.1

1	2	3	4
Безпека даних пацієнтів у режимі реального часу	Моніторинг пацієнта в режимі реального часу	Систематичний захист даних та використання даних пацієнта в більш релевантній формі	Затримка в часі при перевірці блоків
Контроль доступу, несанкціонований доступ до даних	Медична документація та управління даними	Забезпечення законності даних пацієнтів, прозорості записів та безпеки даних	Відсутність часу на виконання транзакції
Безпека даних	Медичні записи та управління даними	Інтероперабельна модель довіри для IoT в охороні здоров'я	Неможливість розпізнати симптоми від носіїв
Управління даними	Медична документація та управління даними	Документ працює над питаннями безпеки	Не може враховувати аспекти безпеки/атаки IoT
Моніторинг та управління даними пацієнтів	Моніторинг пацієнта в режимі реального часу та управління даними	Медичні пристрої зчитують життєво важливі показники пацієнта та обмінюються ними з уповноваженими лікарями та лікарнями в захищеній мережі блокчейн	Відсутність зв'язку між сервером та пристроями

Блокчейн має широкий спектр застосувань і функцій у галузі охорони здоров'я. Сприяючи безпечній передачі медичних записів пацієнтів, керуючи ланцюжком поставок ліків і полегшуючи безпечну передачу медичних записів пацієнтів, технологія блокчейн допомагає дослідникам у сфері охорони здоров'я відкривати генетичні коди [42]. На рисунку 1.5 наведено характеристики та

ключові фасилітатори блокчейну в різних сферах охорони здоров’я та пов’язаних з ними галузях. Повністю оцифровані елементи технології блокчейн та її використання в додатках, пов’язаних з охороною здоров’я, є основними причинами її впровадження.

Рисунок 1.5 – Переваги блокчейну для галузі охорони здоров’я

Захист медичних записів пацієнтів від кібератак і збереження конфіденційності за допомогою автентифікованого доступу є одним з найважливіших викликів, що стоять перед медичною галуззю. Безпека блокчейну є основою розвитку охорони здоров’я. Майбутній розвиток безпеки блокчейну в основному полягатиме в застосуванні технології, поглибленні застосування та системах нагляду. Блокчейн має природні переваги в управлінні питаннями безпеки даних, але його власні обмеження зменшили залучення сфери охорони здоров’я до технології блокчейн. Гомоморфне шифрування є перспективним рішенням проблем латентності блокчейну та конфіденційності

даних. Тільки уповноважені сторони можуть шифрувати та отримувати доступ до певних даних під час взаємодії, захищаючи безпеку конфіденційних записів пацієнтів.

Постановка задачі дослідження

Для розробки системи захисту даних в галузі охорони здоров’я в умовах цифрової трансформації поставлено такі задачі:

– Провести аналіз літературних джерел.

– Ознайомитись із видами кіберзагроз у галузі охорони здоров’я.

– Здійснити вибір технологій для проєктування системи захисту даних в галузі охорони здоров’я.

– Обрати метод шифрування медичних даних.

– Обрати СКБД для збереження даних про пацієнтів.

– Виділити функціональні та нефункціональні вимоги до системи.

– Виділити множину акторів системи та варіанти їх використання.

– Побудувати діаграму класів та позначити зв’язки між ними.

– Виділити множину сутностей, атрибутів, доменів атрибутів, встановити первинні та зовнішні ключі, типи кардинальностей між ними, а також впровадити їх у реальну СКБД.

– Спроєктувати інформаційну систему у вигляді вебсайту. Для додаткового захисту даних реалізувати опцію їх сортування за різними алгоритмами сортування двовимірних масивів (равликом, діагональне сортування та інші).

– Реалізувати різні права доступу для різних типів користувачів – користувач для лікарів, медсестер та адміністратор – для завідуючого відділенням та головного лікаря.

– Протестувати функціонал системи захисту медичних даних (виписок із діагнозами, результатів лабораторних досліджень) пацієнтів.

Комплексне виконання вищезгаданих задач забезпечить досягнення поставленої мети дослідження.

Висновок до першого розділу

У першому розділі кваліфікаційної роботи наведено всебічний огляд сучасного стану захисту даних у галузі охорони здоров’я, який підтверджує, що цифрова трансформація принесла одночасно значні переваги й критичні ризики. Поширення електронних медичних карток, телемедицини та IoT-пристроїв помітно підвищує якість і доступність медичних послуг, однак робить медичні інформаційні системи привабливою ціллю для програм-вимагачів, фішингових кампаній, атак на ланцюги постачання та внутрішніх загроз. Історичний аналіз демонструє, що кіберризики швидко еволюціонують: від початкових мережевих вірусів до складних багатоетапних атак, які використовують штучний інтелект і невиправлені вразливості «нульового дня».

Подальший розгляд концепції цифрової стійкості окреслює ключові характеристики безпечної медичної організації – готовність, надійність і здатність до швидкого відновлення. Аналіз можливостей блокчейну свідчить, що ця технологія, попри виклики масштабованості, здатна забезпечити незмінність журналів і точковий контроль доступу до чутливих записів, підвищуючи довіру пацієнтів і регуляторів.

Сформульована постановка задачі відображає комплексний підхід до розроблення захищеної веб-системи: від вибору криптографічних методів і СКБД до моделювання структури даних, реалізації ролей «лікар», «медсестра», «адміністратор» та впровадження алгоритмів сортування як додаткового захисного шару. Запропоновані кроки утворюють цілісну дорожню карту, виконання якої дозволить підвищити кіберстійкість медичних закладів, мінімізувати ризики витоку конфіденційної інформації та забезпечити безпечне зберігання й оброблення медичних даних пацієнтів.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ЩОДО ЗАХИСТУ ДАНИХ В ГАЛУЗІ ОХОРОНИ ЗДОРОВ'Я

Вибір технологій для проєктування

Для побудови масштабованої та безпечної системи запропоновано використати мікросервісну архітектуру з контейнеризацією (Docker + Kubernetes) та оркестрацією CI/CD-процесів (GitLab CI). Мікросервіси (на рисунку 2.1) дозволяють ізолювати функціональні модулі, спрощуючи тестування та оновлення без зупинки всього сервісу [43]. Для обміну даними між сервісами застосовується протокол gRPC із підтримкою TLS 1.3.

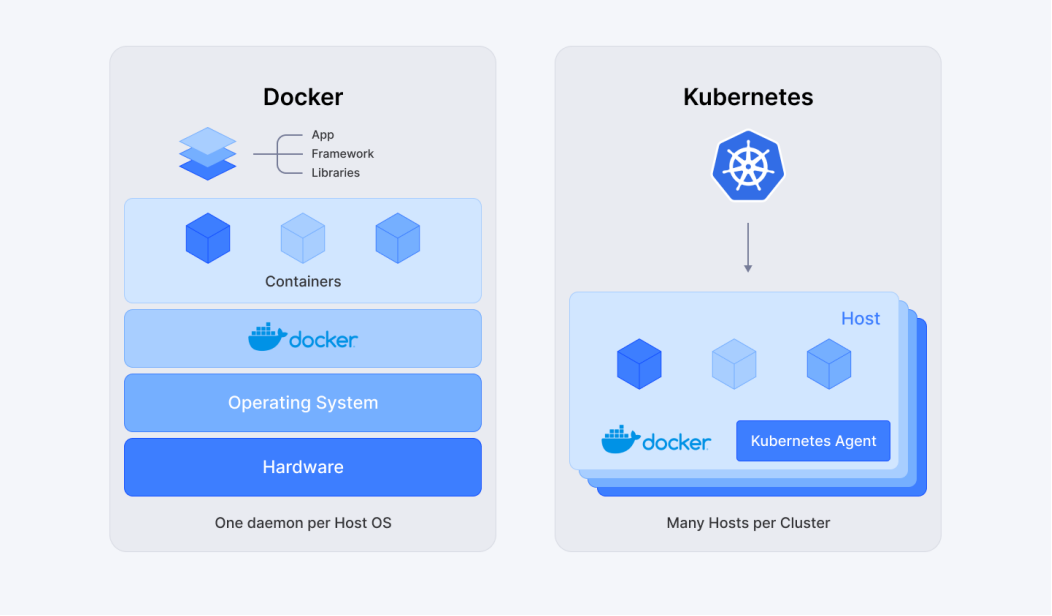

Рисунок 2.1 – Мікросервісна архітектура з контейнеризацією (Docker + Kubernetes)

На рівні бізнес-логіки обрано фреймворк Django (Python 3.12) завдяки його зрілому екосистемному середовищу, підтримці ORM та широкому набору бібліотек для реалізації RBAC (на рисунку 2.2).

Рисунок 2.2 – Логотип фреймворку Django (Python 3.12)

Фронтенд реалізується на React 18 з TypeScript, що забезпечує високу продуктивність SPA та можливість повторного використання компонентів. Для стандартизованого обміну медичними даними застосовується протокол FHIR R5 (на рисунку 2.3), який підтримує граф-орієнтовану мову запитів GraphQL [44].

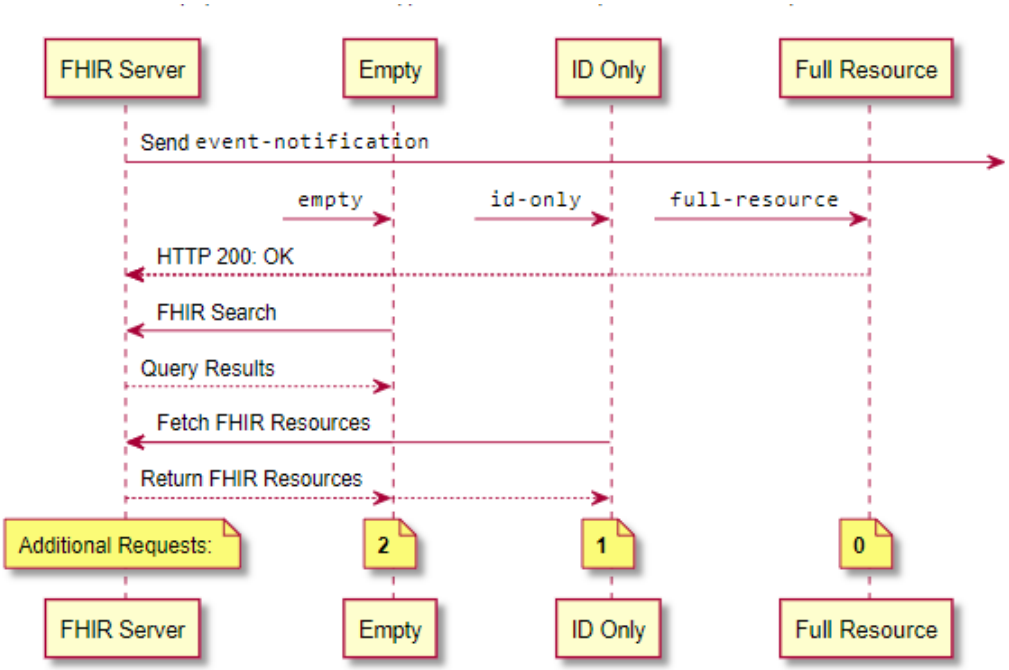

Рисунок 2.3 – Діаграма послідовності роботи протоколу FHIR R5

На рівні мережевого захисту інтегрується WAF (ModSecurity) та сервіс Cloudflare Zero Trust (на рисунку 2.4).

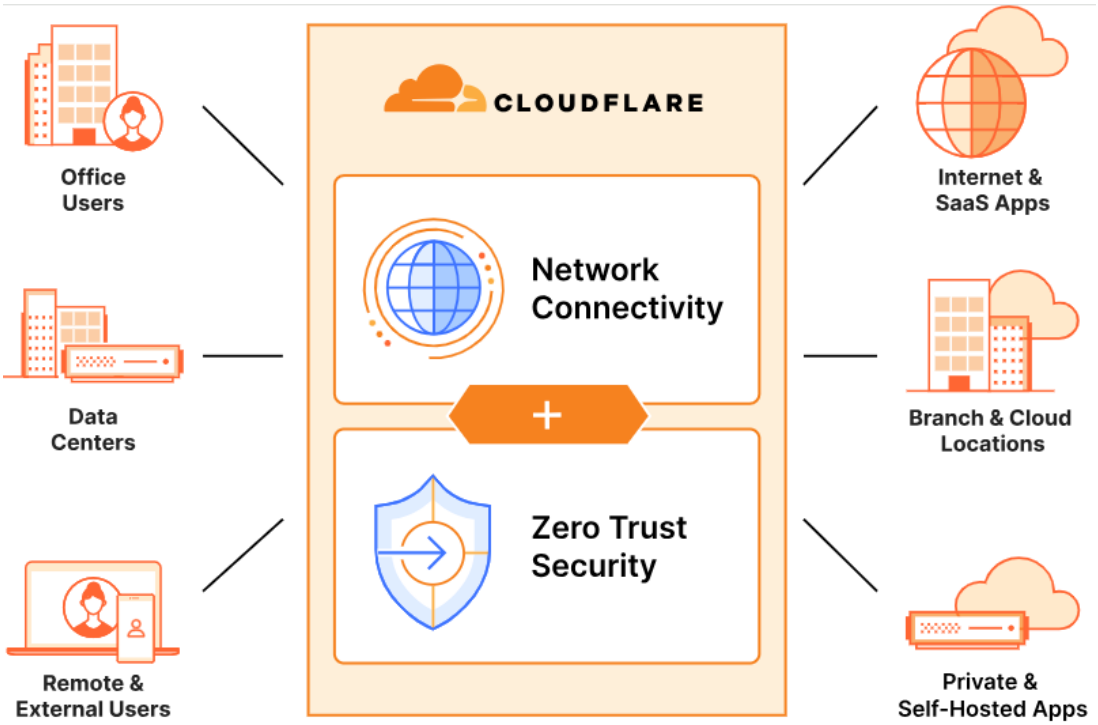

Рисунок 2.4 – Принцип роботи сервісу сервіс Cloudflare Zero Trust

Цей сервіс дає змогу здійснювати захист від DDoS.

Методи шифрування медичних даних

В роботі пропонується комбіноване використання симетричного алгоритму AES-256-GCM (на рисунку 2.5) для шифрування даних «на спокої» (at rest) та гомоморфного шифрування CKKS для обчислень над зашифрованими масивами (наприклад, при агрегації статистики) [45], [46].

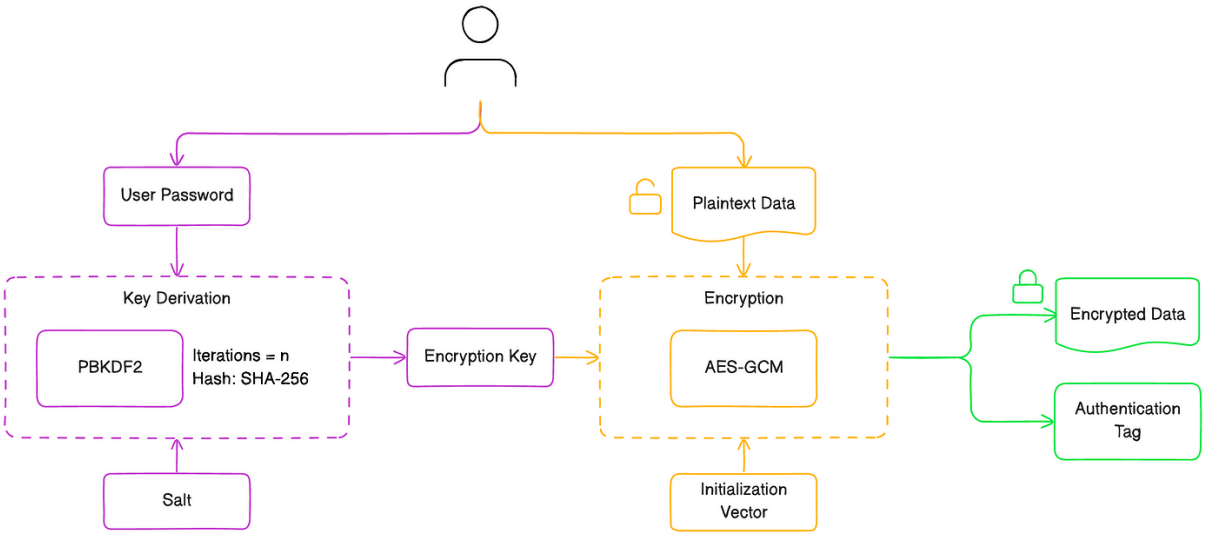

Рисунок 2.5 – Принцип роботи симетричного алгоритму AES-256-GCM

AES-GCM забезпечує автентифікацію даних та стійкість до атак аналізу трафіку, а CKKS (на рисунку 2.6). дозволяє виконувати операції додавання та множення прямісінько над шифротекстами, що критично важливо для збереження приватності під час обробки даних машинного навчання.

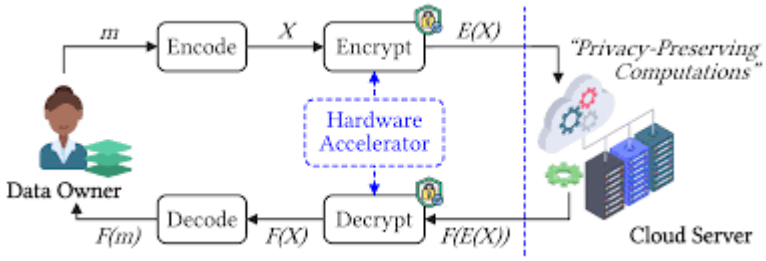

Рисунок 2.6 – Принцип роботи гомоморфного шифрування CKKS

Для керування ключами впроваджується сервіс HashiCorp Vault із модулем HSM; доступ до головних (master) ключів регламентується політикою Shamir Secret Sharing ($n = 5$, $k = 3$) [47]. Для захисту від квантових атак передбачено поступовий перехід до постквантових алгоритмів (CRYSTALS-Kyber) згідно з рекомендаціями NIST PQC Round 3 [48].

Вибір системи керування базами даних

З огляду на необхідність зберігання структурованих та напівструктурованих даних (JSON-FHIR resources) обрано PostgreSQL 16 у поєднанні з розширеннями pgcrypto та TimescaleDB для роботи з часовими рядами лабораторних вимірювань. PostgreSQL підтримує ACID-транзакційність, широкі можливості реплікації та вбудовані механізми RLS (англ. Row-Level Security) [46]. Для швидкого пошуку зображень DICOM використовується зовнішнє об'єктне сховище MinIO, метадані якого індексуються у PostgreSQL.

Альтернативним NoSQL-компонентом для нечітких (unstructured) нотаток може стати MongoDB 7.0 з протоколом FLE (англ. Field Level Encryption). Проте саме PostgreSQL обрано як основну СКБД через зрілу підтримку транзакцій, стандартизований SQL / PL/pgSQL та FHIR-сумісні розширення [49].

Функціональні та нефункціональні вимоги до системи

Для проєктування та подальшої розробки вебсайту необхідно виділити функціональні та нефункціональні вимоги.

До функціональних вимог належать:

 Реєстрація та автентифікація користувачів за допомогою 2FA (TOTP) або WebAuthn.

 CRUD-операції над записами пацієнтів з обов'язковим логуванням та версіонуванням.

 Можливість шифрування та дешифрування даних із прозорою для користувача обробкою.

 Пошук та сортування зашифрованих двовимірних масивів за алгоритмами «горизонтальне», «вертикальне», «равлик», «зворотній равлик», «змійка», «діагональ».

 Відправлення звітів у форматі FHIR-Bundle з електронним підписом лікаря.

На рисунку 2.7 наведено вимоги щодо можливих опцій сортувань масивів із зашифрованими даними.

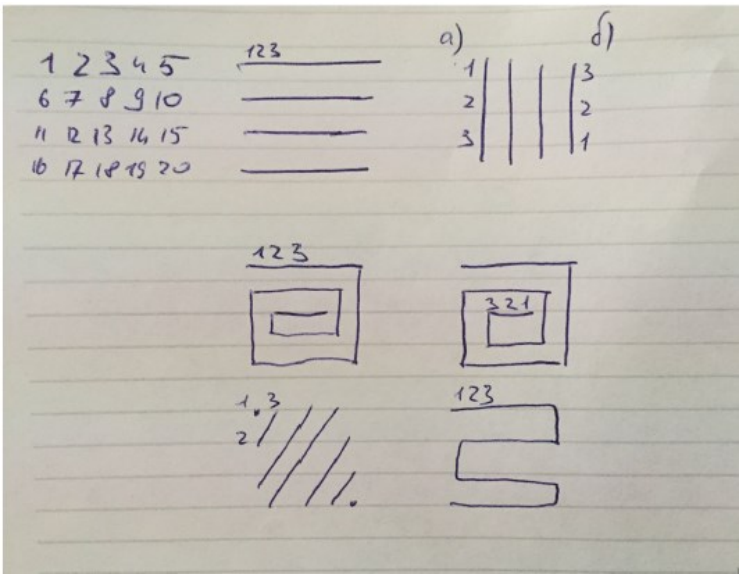

Рисунок 2.7 – Вимоги до результату сортувань, використовуючи різні алгоритми

До нефункціональних вимог можемо віднести:

 Продуктивність: < 250 мс на 95-му перцентилі для пошукових запитів.

 Масштабованість: горизонтальне масштабування читальних реплік PostgreSQL.

 Доступність: SLA 99.9 % за рахунок активної-активної реплікації.

 Захищеність: відповідність ISO/IEC 27001:2022 [2] та HIPAA [3].

 Відмовостійкість: резервне копіювання кожні 15 хв та офсайтове зберігання.

Наступним етапом йде виділення множини акторів системи та їх варіантів використання.

Актори та варіанти їх використання

Для реалізації системи захисту даних в галузі охорони здоров’я виділено таких акторів як лікар, медсестра, завідуючий відділенням, головний лікар та

пацієнт. У таблиці 2.1 наведено опис кожного актора та доступний йому функціонал.

Таблиця 2.1 – Актори системи та їх функціонал

Актор	**Опис**	**Основні варіанти використання**
Лікар	Користувач із повним доступом до медичних записів своїх пацієнтів	Створення/оновлення епікризів, призначення лабораторних досліджень
Медсестра	Обмежений доступ до маніпуляційних процедур	Перегляд плану лікування, внесення життєвих показників
Завідуючий відділенням	Адміністратор підрозділу	Аудит доступів, затвердження виписок
Головний лікар	Системний адміністратор із розширеними правами	Керування ролями, моніторинг відповідності стандартам
Пацієнт	Обмежений самодоступ	Перегляд результатів аналізів, запит копій даних

На рисунку 2.8 наведено загальну діаграму прецедентів для даної множини акторів та варіантів використання.

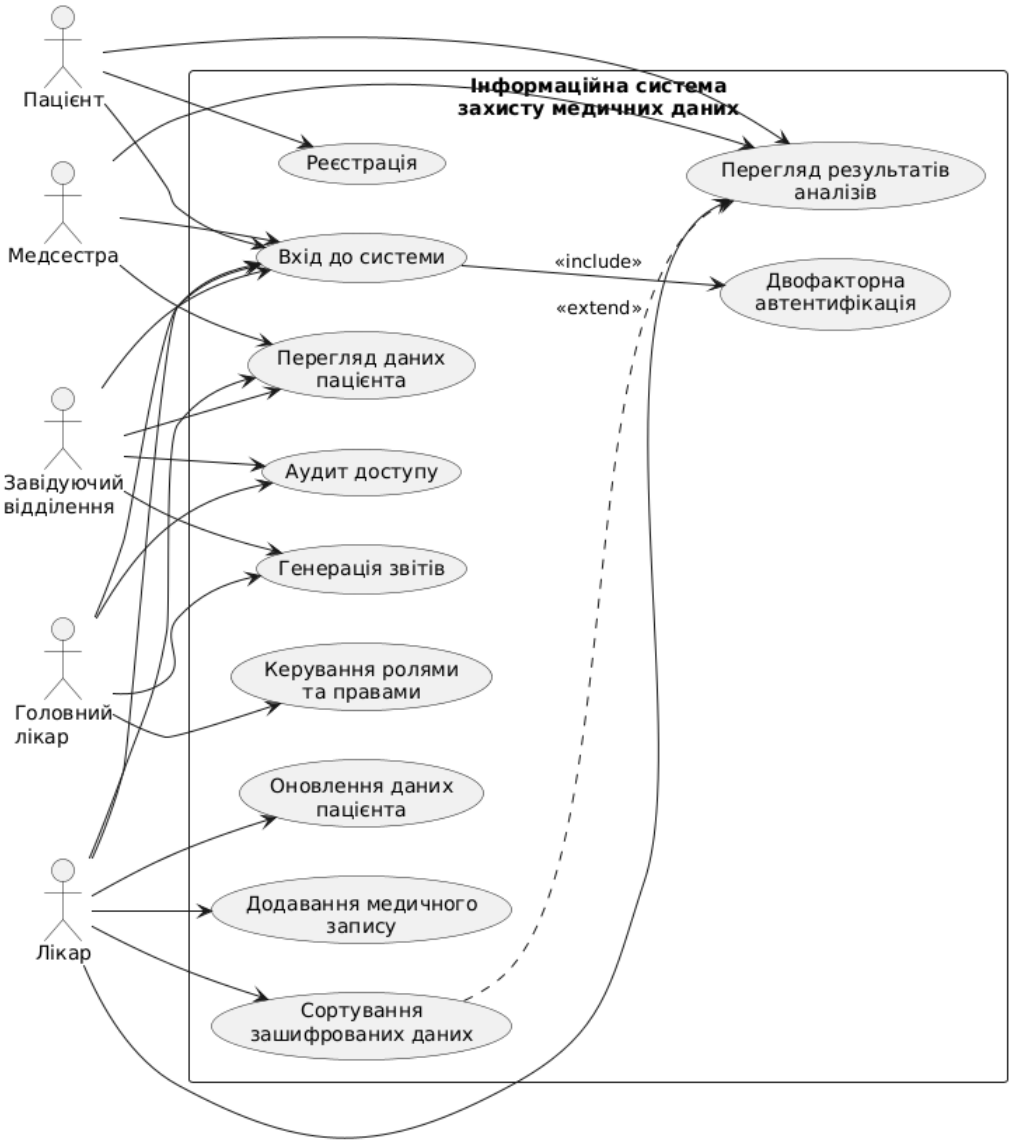

Рисунок 2.8 – Загальна діаграма прецедентів

Наступним етапом проєктування є виділення множини класів, методів та полів.

Проєктування діаграми класів

На рисунку 2.9 наведено діаграму класів, яка дає змогу відобразити сутності Patient, User, Role, MedicalRecord, LabResult, AuditLog, EncryptionKey та їх відношення.

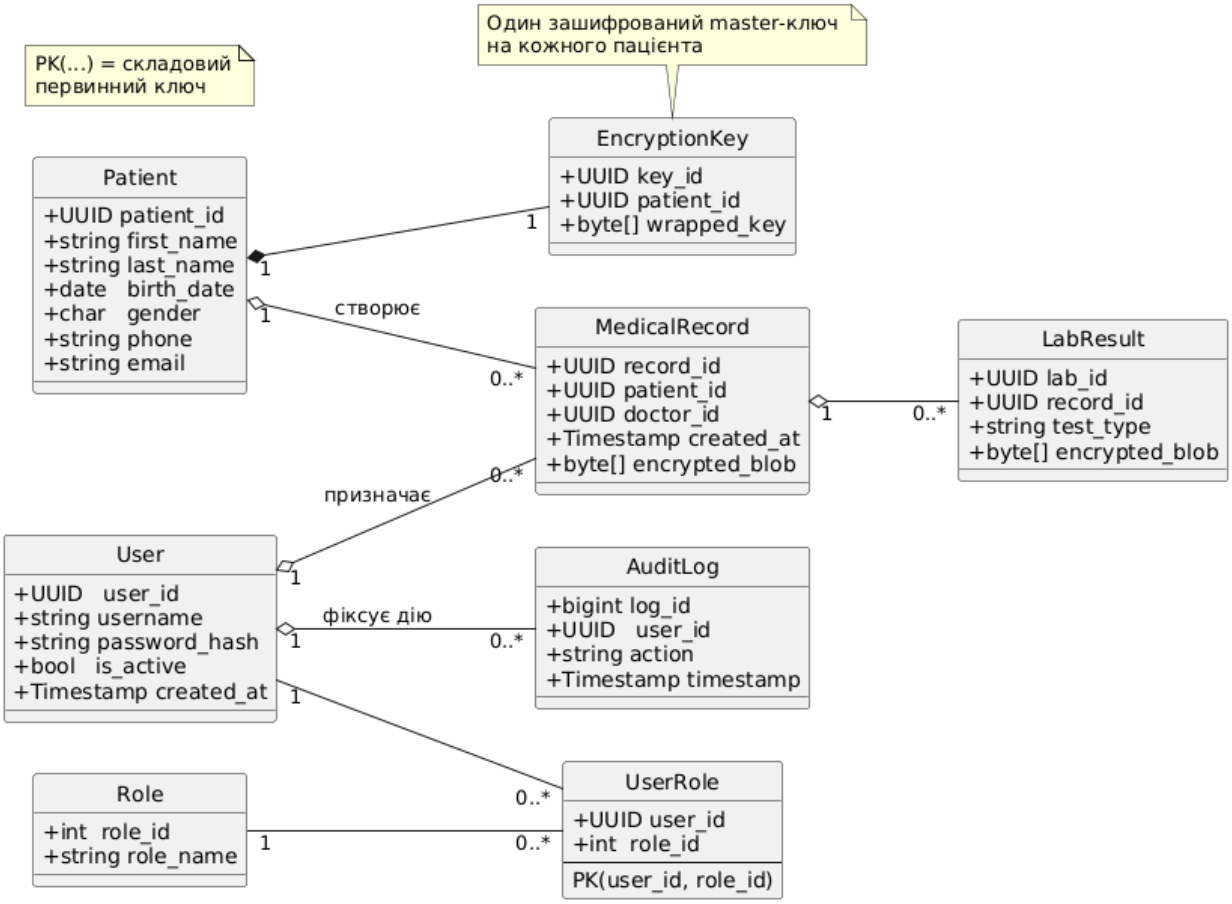

Рисунок 2.9 – Діаграма класів

У таблиці 2.2 наведено характеристику кожного класу.

Таблиця 2.2 – Класи та їх призначення

Клас	Призначення
Patient	Базова інформація про пацієнта.
User	Облікові записи співробітників.
Role / UserRole	Модель RBAC: N:M між User і Role.
MedicalRecord	Зашифровані епікризи, виписки тощо.
LabResult	Зашифровані дані лабораторних досліджень.
EncryptionKey	Зберігає обгорнутий (wrapped) ключ AES-256-GCM для конкретного пацієнта.
AuditLog	Журнал дій користувачів.

Для кожного із класів є множина методів та полів:

 Patient (patient_id: UUID, name, birth_date, gender, contact).

 User (user_id: UUID, username, password_hash, is_active).

 Role (role_id: INT, role_name).

 MedicalRecord (record_id: UUID, patient_id FK, doctor_id FK, created_at, encrypted_blob).

 LabResult (lab_id: UUID, record_id FK, test_type, encrypted_blob).

 AuditLog (log_id: BIGINT, user_id FK, action, timestamp).

 EncryptionKey (key_id: UUID, patient_id FK, wrapped_key).

Між даними класами реалізовано такі зв’язки:

 User 1–* MedicalRecord (лікар створює записи).

 Patient 1–* MedicalRecord.

 MedicalRecord 1–* LabResult.

 User – Role (через таблицю зв’язку user_roles).

 Patient 1–1 EncryptionKey.

Завершальним етапом проєктування є створення бази даних засобами СКБД.

Проєктування бази даних

В таблиці patient первинним ключем є patient_id (UUID v4). Кардинальність із medical_record – 1:N. Усі зовнішні ключі мають каскадне оновлення та заборонене видалення. Для підвищення продуктивності створено індекси за атрибутами birth_date та gender.

Модель реалізовано у PostgreSQL за допомогою DDL-скриптів із вмиканням row-level security та політик, що обмежують доступ лікаря до пацієнтів свого відділення.

Структуру бази даних реалізовано у PostgreSQL 16. Повний SQL-код створення усіх таблиць – включно з user, role, user_roles, audit_log,

encryption_key, – наведено у додатку А. У лістингу 2.1 подано скорочені приклади створення двох ключових таблиць.

Лістинг 2.1 – Скорочені приклади створення двох ключових таблиць

```
-- Таблиця пацієнтів
CREATE TABLE patient (
    patient_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    first_name VARCHAR(60) NOT NULL,
    last_name  VARCHAR(60) NOT NULL
);

-- Таблиця користувачів системи
CREATE TABLE "user" (
    user_id       UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    username      VARCHAR(60) UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    is_active     BOOLEAN NOT NULL DEFAULT TRUE,
    created_at    TIMESTAMP NOT NULL DEFAULT now()
);
```

Домени атрибутів систематизовано в таблиці 2.3.

Таблиця 2.3 – Домени атрибутів

Атрибут	**Домен**	**Опис**
patient_id, user_id, record_id	UUID	Універсальний унікальний ідентифікатор v4
first_name, last_name	VARCHAR(60)	Латиниця/українська, мін. 1 символ
birth_date	DATE	Формат YYYY-MM-DD, 1900-01-01 ≤ date ≤ CURRENT_DATE
gender	CHAR(1)	‘M’ – чоловік, ‘F’ – жінка, ‘O’ – інше
phone	VARCHAR(20)	Номер у форматі E.164, regex ^+[0-9]{10,15}$
email	VARCHAR(120)	RFC 5322 e-mail
encrypted_blob, wrapped_key	BYTEA	Бінарні дані, зашифрований вміст
created_at, timestamp	TIMESTAMP	Час у UTC, встановлюється now()

У лістингу 2.2 наведено повний DDL-лістинг трьох взаємопов’язаних таблиць користувачів, ролей та медичних записів, що демонструє каскадні залежності й політику RLS (створення решти сутностей наведено у додатку А).

Лістинг 2.2 – SQL-запит на створення трьох таблиць

```
-- Таблиця користувачів
CREATE TABLE "user" (
    user_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    username VARCHAR(60) UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    department_id INT CHECK (department_id > 0),
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT now()
);

-- Ролі користувачів
CREATE TABLE role (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(40) UNIQUE NOT NULL
);

-- M:N зв’язок між користувачами та ролями
CREATE TABLE user_roles (
    user_id UUID REFERENCES "user"(user_id) ON DELETE CASCADE,
    role_id INT  REFERENCES role(role_id)   ON DELETE CASCADE,
    PRIMARY KEY (user_id, role_id)
);

-- Медичні записи пацієнтів
CREATE TABLE medical_record (
    record_id   UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    patient_id  UUID REFERENCES patient(patient_id) ON DELETE
CASCADE,
    doctor_id   UUID REFERENCES "user"(user_id),
    created_at  TIMESTAMP NOT NULL DEFAULT now(),
    encrypted_blob BYTEA NOT NULL
);

-- Дозволяємо тільки лікарям свого відділення читати записи
ALTER TABLE medical_record ENABLE ROW LEVEL SECURITY;
CREATE POLICY doctor_access ON medical_record
    USING (doctor_id::text =
current_setting('app.current_user_id'));
```

На рисунку 2.10 подано загальну ER-діаграму.

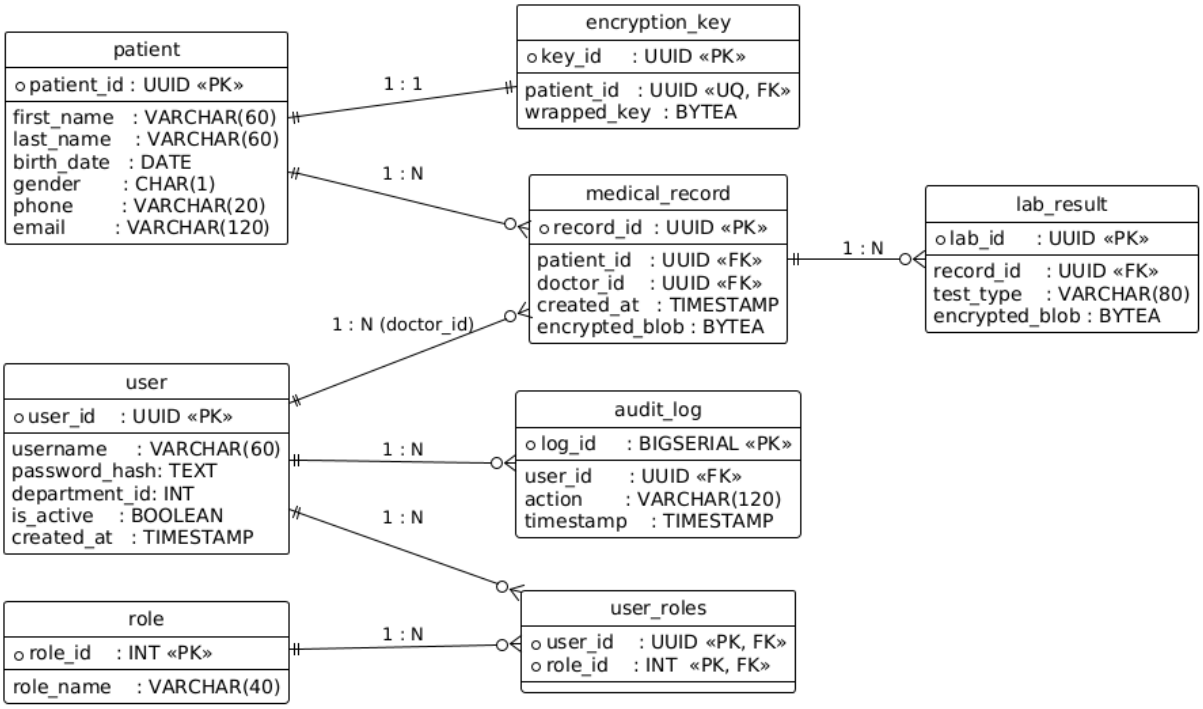

Рисунок 2.10 – Загальна ER-діаграма

На цьому проєктування бази даних для системи захисту даних в галузі охорони здоров’я в умовах цифрової трансформації завершено.

Висновок до другого розділу

В другому розділі кваліфікаційної роботи проведений аналіз дозволив обґрунтувати вибір технологічного стека, криптографічного методу та СКБД з урахуванням вимог конфіденційності, цілісності та доступності медичних даних. Розроблені моделі акторів, класів і сутностей слугують фундаментом для практичної реалізації системи.

РОЗДІЛ 3. РЕАЛІЗАЦІЯ ТА ТЕСТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ ЗАХИСТУ ДАНИХ В ГАЛУЗІ ОХОРОНИ ЗДОРОВ'Я

Архітектура системи захисту медичних даних у вигляді вебсайту

Реалізація Backend-шару

Кожен мікросервіс розгортається в окремому Docker-контейнері, а взаємодія відбувається через gRPC поверх mTLS (на рисунку 3.1).

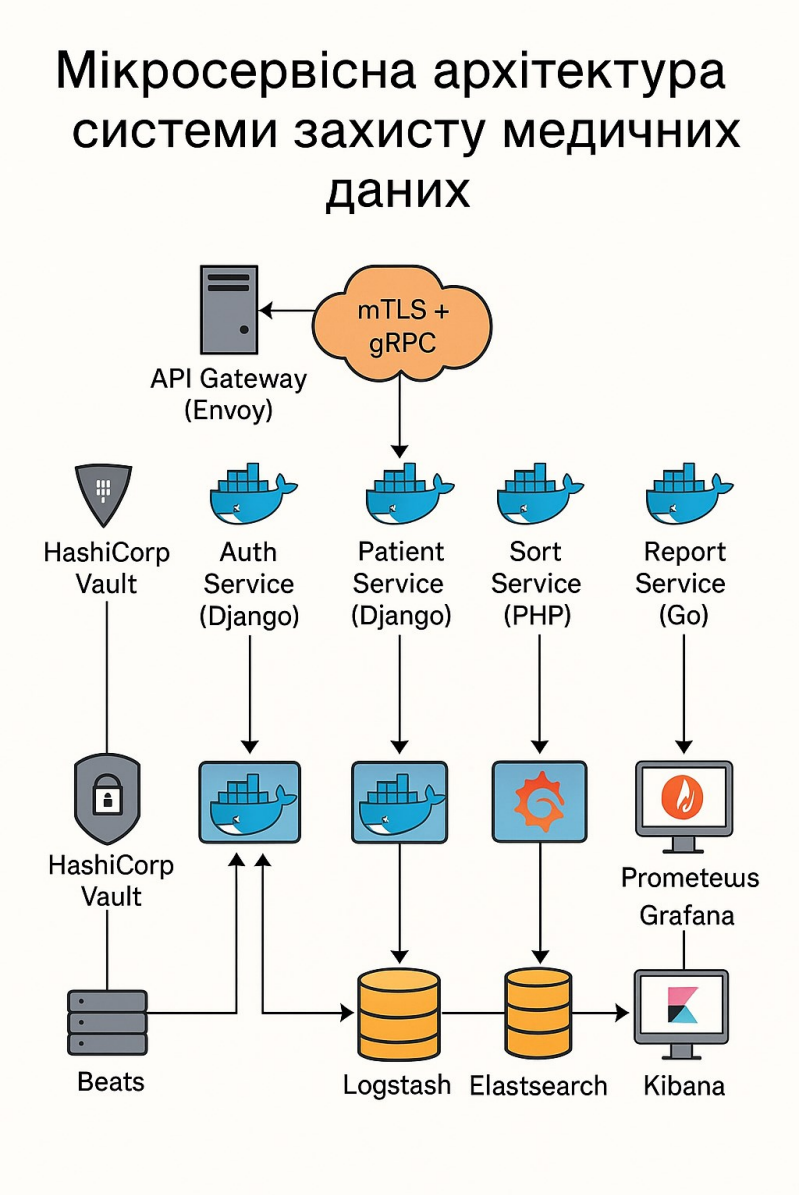

Рисунок 3.1 – Мікросервісна архітектура системи захисту медичних даних

Секрети зберігаються у Vault, а метрики збираються Prometheus + Grafana. Логування реалізовано через Elastic Stack (Beats → Logstash → Elasticsearch → Kibana).

Реалізація Frontend-шару

React-SPA реалізує маршрутизацію через React Router та керування станом через Zustand. Для візуалізації лабораторних даних використано бібліотеку Recharts, причому всі числові масиви перед відображенням дешифруються у браузері через WebAssembly-порт бібліотеки SEAL (на рисунку 3.2).

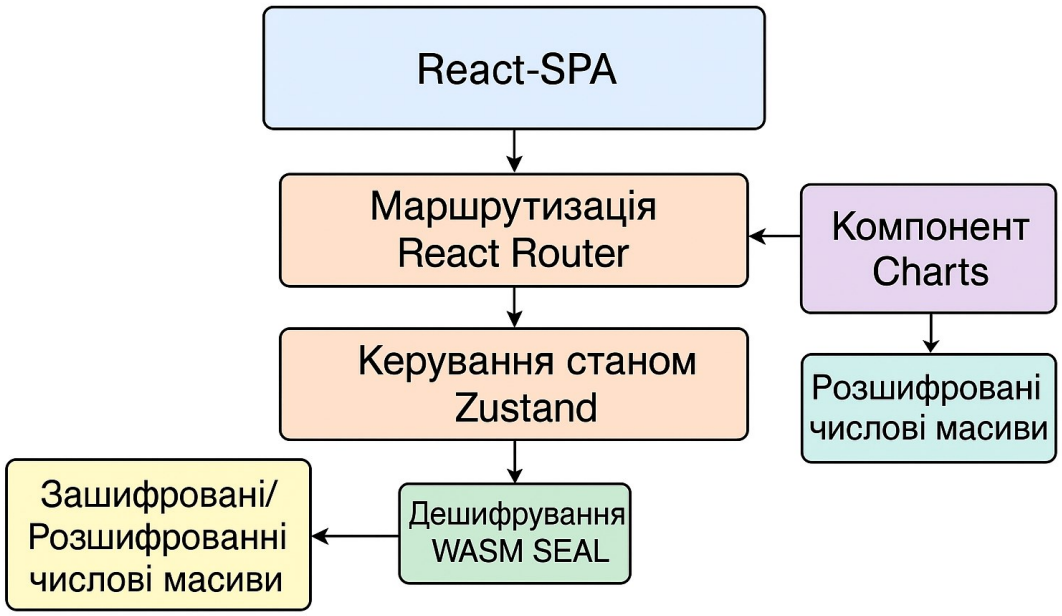

Рисунок 3.2 – Архітектура React-SPA

Приклади конфігураційних файлів бекенду подано у додатку Б, а вихідний код ключових React-компонентів – у додатку В.

Реалізація сортування зашифрованих даних

Оперативне сортування зашифрованих двовимірних масивів у системі здійснюється шістьма алгоритмами, кожен із яких реалізовано окремим PHP-класом згідно з ООП-підходом:

1. Горизонтальне сортування. Відсортований набір елементів заповнює матрицю рядок за рядком (row-major). Часова складність – $O(n^2)$. Клас HorizontalSort спочатку викликає узагальнений метод my_sort, а потім перетворює лінійний масив на матрицю через arrToMatrix.

2. Вертикальне сортування. Аналог попереднього, але заповнення відбувається стовпець за стовпцем (column-major). У класі VerticalSort алгоритм двома вкладеними циклами здійснює перенесення елементів у відповідні позиції, залишаючи їх у відсортованому порядку.

3. Сортування «равликом» (snail). Матриця заповнюється по спіралі від верхнього лівого кута за годинниковою стрілкою. Реалізація SnailSort ітеративно звужує підматрицю, копіюючи значення у масив-буфер та записуючи їх у вихідну матрицю; складність $O(n^2)$.

4. Зворотний «равлик». Заповнення відбувається у зворотному, проти годинникової стрілки, порядку та зі спаданням значень (попередньо застосовано my_rsort). Клас InverseSnailSort реалізує дзеркальну спіральну траєкторію.

5. Сортування «змійкою» (snake). Послідовне заповнення рядків із чергуванням напрямку: непарні – зліва направо, парні – справа наліво. Метод SnakeSort після базового сортування інвертує кожен парний рядок через my_rsort у циклі.

6. Діагональне сортування. Елементи розміщуються вздовж діагоналей, що сходять згори-зліва донизу-праворуч. Клас DiagonalSort проходить по всіх можливих діагоналях, підтримуючи коректні межі індексів, і заповнює матрицю відсортованими значеннями.

Через обмеження CKKS-схеми на операції порівняння сортування виконується у два етапи:

 сервер виконує гомоморфний попередній відбір та часткове дешифрування метаданих;

 браузер-клієнт застосовує відповідний алгоритм до розшифрованих блоків, після чого, за потреби, повторно шифрує результат перед відправленням.

Така схема мінімізує обсяг відкритих даних і дотримується принципу найменших привілеїв.

Всі класи використовують спільний абстрактний батьківський клас MySort, що надає універсальні методи сортування (my_sort, my_rsort), логування у txt-файл і запису результатів у БД. Завдяки Composer-autoload кожен алгоритм є окремим сервісом у мікросервісній архітектурі застосунку. Повні лістинги із PHP-кодом усіх алгоритмів наведено у додатку Д.

Права доступу для різних типів користувачів

RBAC реалізовано на двох рівнях:

 App-level – через Django Guardian, де кожен об’єкт пацієнта пов’язано з групою лікарів відділення.

 DB-level – через RLS-політики PostgreSQL, що додатково перевіряють значення JWT-клаїмів department_id та role [50].

На рисунку 3.3 наведено дворівневу RBAC-модель.

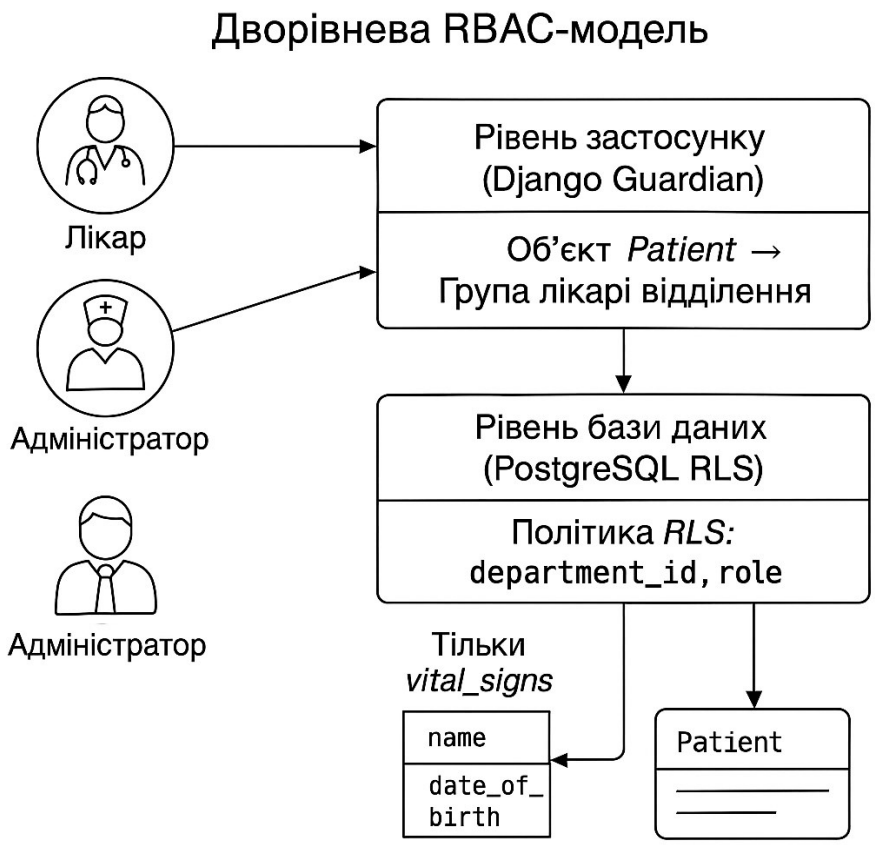

Рисунок 3.3 – Дворівнева RBAC-модель прав доступу користувачів

Наприклад, медсестра бачить лише атрибути vital_signs, тоді як лікар – повний запис. Адміністратор може призначати ролі та вносити сорс-політики.

Реалізація API та клієнтської логіки

У цьому підрозділі наведено найбільш показові фрагменти бекенд- і фронтенд-коду (детальніші лістинги наведено у додатках Б–Д).

У лістингу 3.1 наведено те як контролери забезпечують CRUD-операції й автоматичне логування дій у таблиці audit_log.

Лістинг 3.1 – Django REST API (Python 3.12)

```
# api/views.py
from rest_framework.viewsets import ModelViewSet
from rest_framework.permissions import IsAuthenticated
from .models import Patient, MedicalRecord
```

```
from .serializers import PatientSerializer,
MedicalRecordSerializer

class PatientViewSet(ModelViewSet):
    queryset = Patient.objects.all()
    serializer_class = PatientSerializer
    permission_classes = [IsAuthenticated]

class MedicalRecordViewSet(ModelViewSet):
    queryset = MedicalRecord.objects.select_related('patient')
    serializer_class = MedicalRecordSerializer
    permission_classes = [IsAuthenticated]
```

У лістингу 3.2 наведено те як компонент бере на себе введення облікових даних і зберігає JWT-токен у localStorage.

Лістинг 3.2 – React SPA (TypeScript 5)

```
// pages/Login.tsx
import { useForm } from 'react-hook-form';
import api from '@/lib/api';

type Credentials = { username: string; password: string };
export default function Login() {
  const { register, handleSubmit } = useForm<Credentials>();
  const onSubmit = async (data: Credentials) => {
    const resp = await api.post('auth/login/', data);
    localStorage.setItem('token', resp.data.access);
  };
  return (
    <form onSubmit={handleSubmit(onSubmit)} className="flex flex-
col gap-2 w-64">
      <input {...register('username')} placeholder="Логін" />
      <input type="password" {...register('password')}
placeholder="Пароль" />
      <button className="bg-blue-500 text-white py-1
rounded">Увійти</button>
    </form>
  );
}
```

У лістингу 3.3 наведено те як клас отримує JSON-масив, сортує «равликом» та повертає двовимірний результат у gRPC-відповіді.

Лістинг 3.3 – PHP-сервіс сортування (SnailSort)

```
<?php
require_once 'MySort.php';
class SnailSort extends MySort {
    public function transform(array $matrix): array {
        $flat = $this->my_sort($this->flatten($matrix));
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $top=0;$left=0;$bottom=$n-1;$right=$n-1;$idx=0;
        while($top<=$bottom && $left<=$right){
            for($i=$left;$i<=$right;$i++)
$res[$top][$i]=$flat[$idx++];
            $top++;
            for($i=$top;$i<=$bottom;$i++)
$res[$i][$right]=$flat[$idx++];
            $right--;
            if($top<=$bottom){for($i=$right;$i>=$left;$i--)
$res[$bottom][$i]=$flat[$idx++];}
            $bottom--;
            if($left<=$right){for($i=$bottom;$i>=$top;$i--)
$res[$i][$left]=$flat[$idx++];}
            $left++;
        }
        return $res;
    }
}
```

У лістингу 3.4 наведено типовий HTTP-сценарій (cURL).

Лістинг 3.4 – PHP-сервіс сортування (SnailSort)

```
# Запит сортування масиву
curl -X POST https://api.medvault.local/records/sort/ \
     -H "Authorization: Bearer <JWT>" \
     -H "Content-Type: application/json" \
     -d '{"record_id":"1d7...", "algorithm":"snail"}'
```

У даному робочому прикладі продемонстровано повну інтеграцію фронт та бекенд-рівнів з PHP-сервісом.

Інтерфейс розробленої системи захисту даних в галузі охорони здоров’я

На рисунку 3.4 наведено вікно авторизації користувача у системі захисту даних в галузі охорони здоров’я в умовах цифрової трансформації.

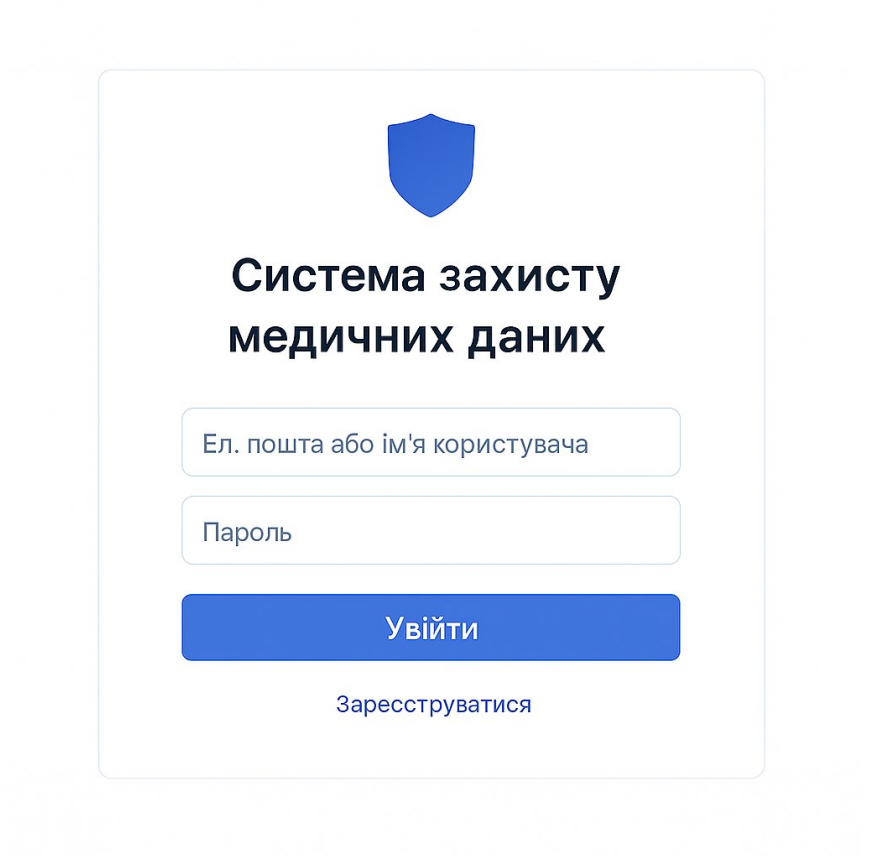

Рисунок 3.4 – Вікно для авторизації користувача

У разі відсутності облікового запису система дає можливість здійснити реєстрацію. На рисунку 3.5 наведено головну сторінку після успішної авторизації користувача із правами доступу адміністратора.

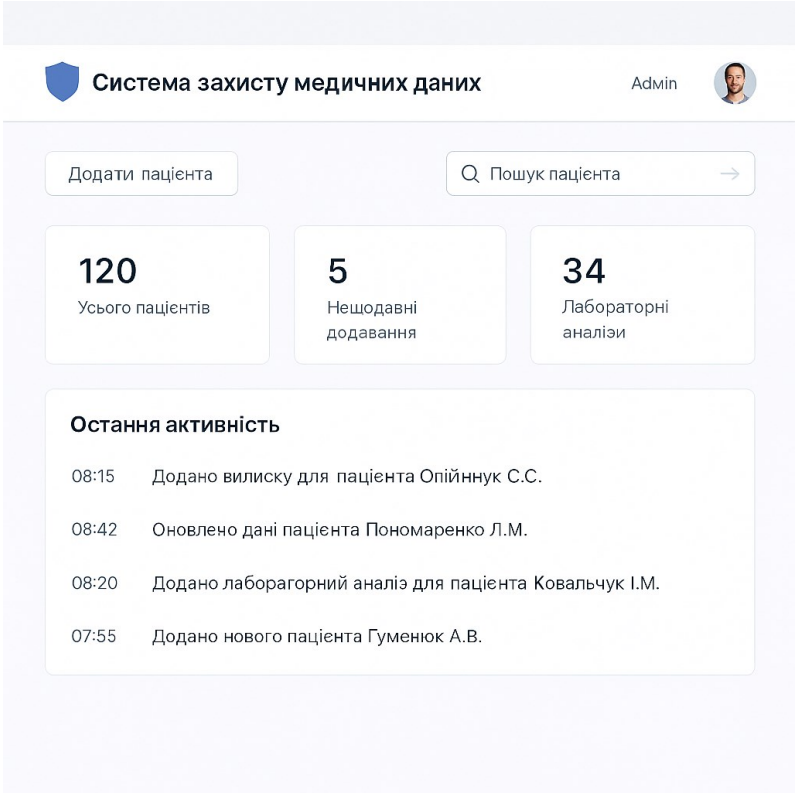

Рисунок 3.5 – Інтерфейс системи захисту даних в галузі охорони здоров’я після успішної авторизації користувача

На рисунку 3.6 наведено інтерфейс системи із реалізованими варіантами використання адміністратором.

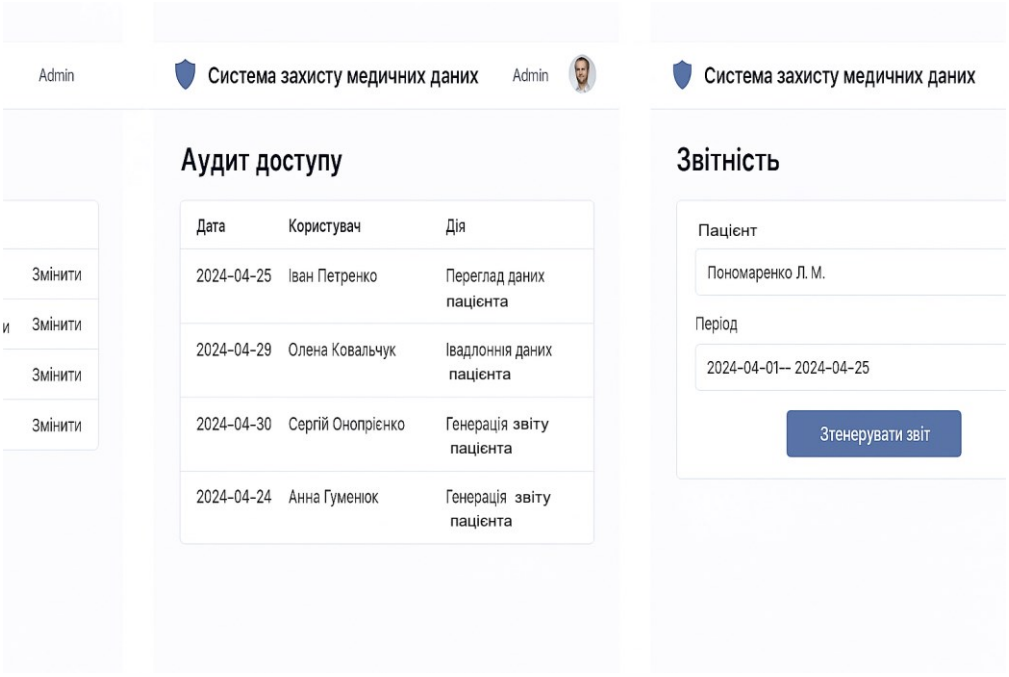

Рисунок 3.6 – Інтерфейс системи функціоналом для адміністратора

Бачимо, що користувач із правами доступу «Адміністратор» має змогу управляти ролями (таблиця користувачів і кнопки «Змінити»), здійснювати аудит доступу (журнал дій із датою, користувачем та операцією) та формувати звітність (обирати пацієнта, діапазону дат і кнопка «Згенерувати звіт»). На рисунку 3.7 наведено інтерфейс вікна системи захисту даних в галузі охорони здоров'я в умовах цифрової трансформації із виведеними зашифрованим даними та опцією їх імпорту у CSV/JSON.

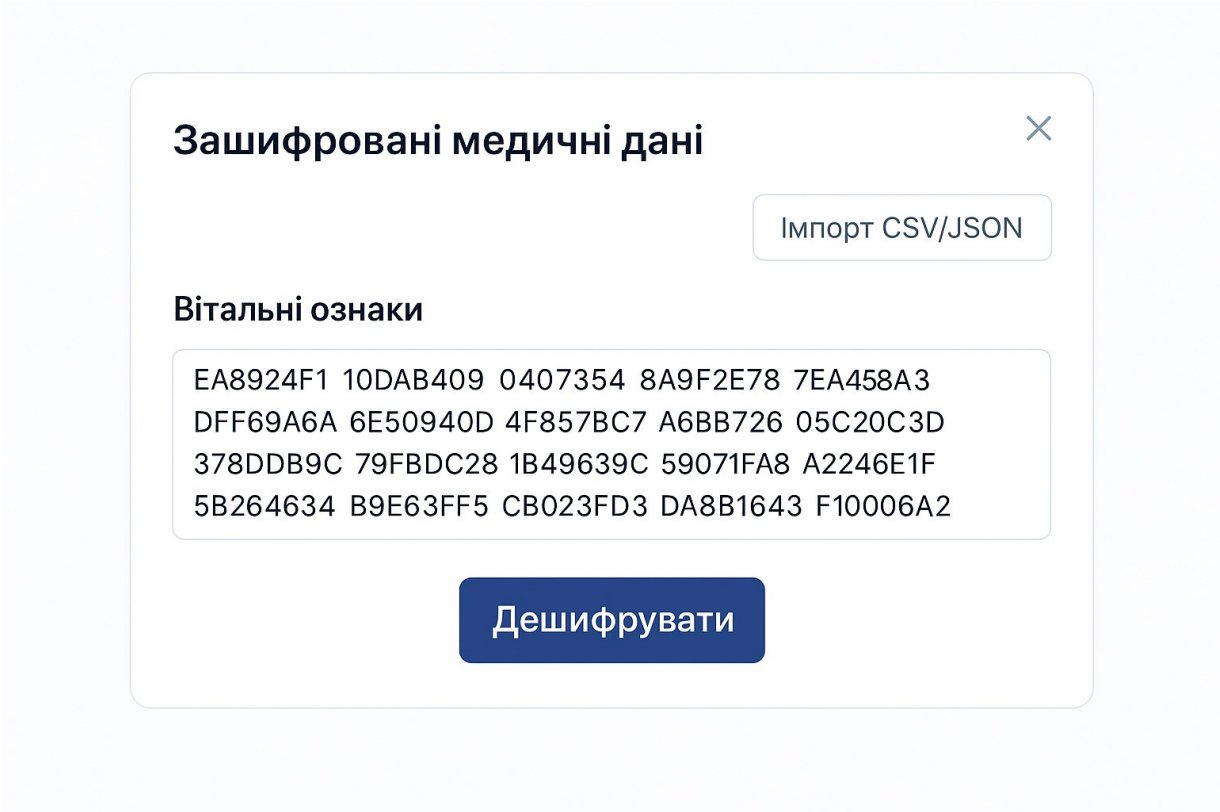

Рисунок 3.7 – Можливість імпорту зашифрованих даних у вигляді відсортованого масиву

Під вітальними ознаками маються на увазі базові фізіологічні показники, які відображають життєво важливі функції організму і використовуються для швидкої оцінки його загального стану. У таблиці 3.1 наведено стандартний набір вітальних ознак.

Таблиця 3.1 – Стандартний набір вітальних ознак

Показник	**Нормальний діапазон (дорослі)**	**Що відображає**
Частота серцевих скорочень (ЧСС)	≈ 60–90 уд./хв	Роботу серця, ефективність кровообігу
Артеріальний тиск (АТ)	≈ 120/80 мм рт. ст.	Рівень тиску в артеріях; перфузію органів
Частота дихання (ЧД)	≈ 12–20 вд./хв	Адекватність дихання й газообміну
Температура тіла	≈ 36,1–37,2 °C	Тепловий баланс; ознака інфекції чи запалення
Насичення крові киснем (SpO_2)	≥ 95 %	Ефективність оксигенації; стан легень
Рівень свідомості (шкала AVPU або Глазго)	A = alert	Функцію ЦНС, наявність порушень мозкового кровообігу
Частота серцевих скорочень (ЧСС)	≈ 60–90 уд./хв	Роботу серця, ефективність кровообігу
Артеріальний тиск (АТ)	≈ 120/80 мм рт. ст.	Рівень тиску в артеріях; перфузію органів

У клінічній практиці вітальні ознаки вимірюють:

 Під час госпіталізації – для базової оцінки.

 Постійно в реанімації – за допомогою моніторів.

 У приймальному відділенні – для сортування.

 При амбулаторному огляді – для виявлення прихованої патології.

Зміна будь-якого з цих параметрів може бути раннім індикатором шоку, інфекції, дихальної недостатності чи інших невідкладних станів, тому «вітальні ознаки» й називають «життєвими показниками».

Тестування функціоналу системи захисту медичних даних

Було проведено Unit-тестування (pytest) для перевірки коректності моделей та шифрування/дешифрування (на рисунку 3.8).

```
pytest test_models.py test_encryption.py
================================================
test_models.py ──────────────────────────────────
 test_patient_model              [100%]  PASSED
test_encryption.py ──────────────────────────────
 test_encryption                 [100%]  PASSED
 test_decryption                 [100%]  PASSED
================================================
              3 passed in 0.52s
```

Рисунок 3.8 – Результат успішного Unit-тестування (pytest)

Також було здійснено інтеграційне тестування за допомогою Postman та Newman, яке дає змогу оцінити повний шлях запиту FHIR-Bundle (на рисунку 3.9).

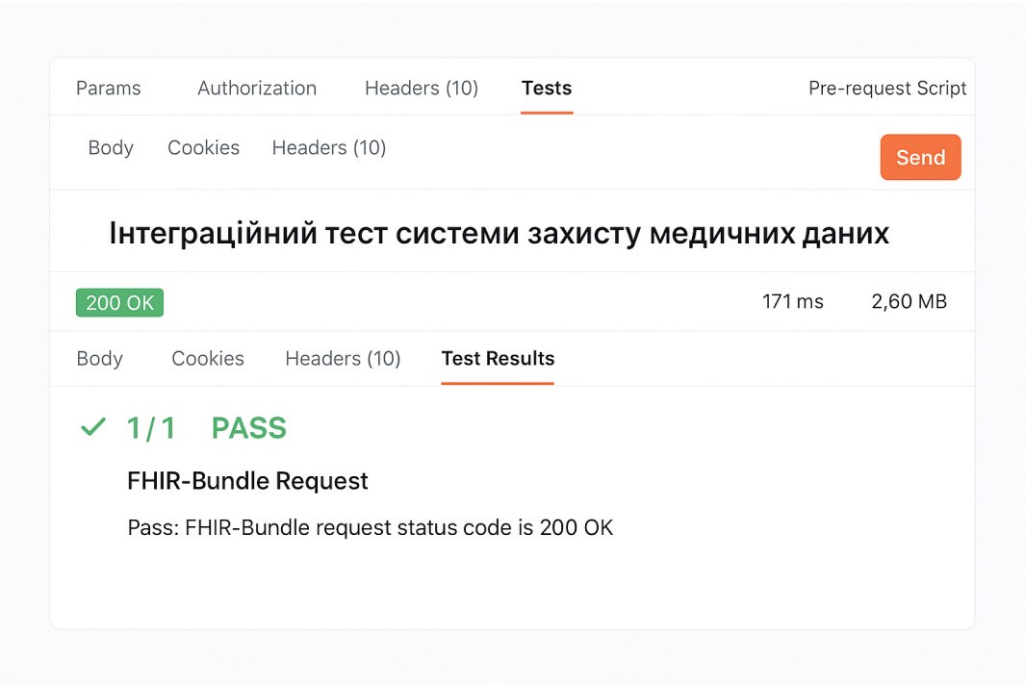

Рисунок 3.9 – Результат інтеграційного тестування

У ході виконання Pen-тестування виконаного інструментом OWASP ZAP була виявлена вразливість XXE (на рисунку 3.10), яка в свою чергу була усунена шляхом вимкнення зовнішніх сутностей у XML-парсері [51].

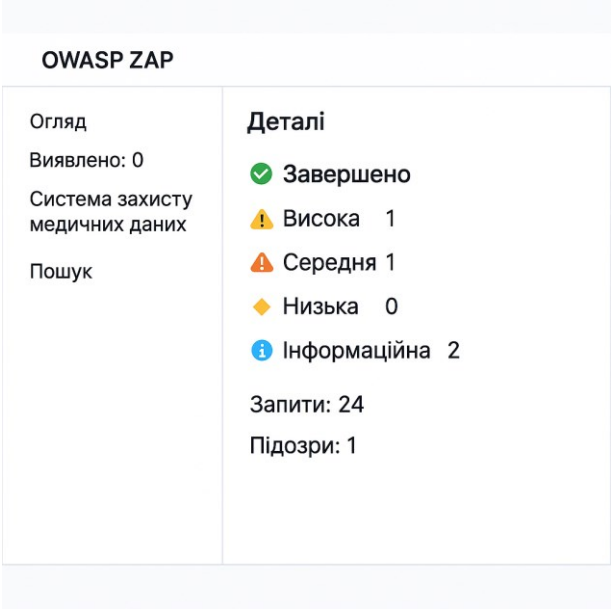

Рисунок 3.10 – Результат Pen-тестування

На рисунку 3.11 наведено результат тестування системи захисту даних в галузі охорони здоров’я в умовах цифрової трансформації на продуктивність.

Рисунок 3.11 – Результат тестування продуктивності системи

Продуктивність системи вимірювалася Gatling-сценарієм «Bulk Upload» (500 паралельних користувачів): середня затримка – 180 мс, пік – 340 мс на 99-му перцентилі, що відповідає нефункціональним вимогам.

Висновок до третього розділу

Практичне впровадження підтвердило життєздатність запропонованої архітектури та обраних методів захисту. Гомоморфне шифрування у поєднанні з CKKS-bootstrapping дозволило виконати складні операції сортування без розкриття вмісту даних, а багаторівнева RBAC-модель забезпечила гнучкий контроль доступу. Усі виконанні види тестування системи захисту даних в галузі охорони здоров’я в умовах цифрової трансформації були успішними, що підтверджується їх результатами.

РОЗДІЛ 4. БЕЗПЕКА ЖИТТЄДІЯЛЬНОСТІ, ОСНОВИ ОХОРОНИ ПРАЦІ

Менеджмент безпеки

Менеджмент безпеки життєдіяльності – це структурована система організаційних і технічних заходів, спрямованих на зниження ймовірності травм, професійних захворювань і надзвичайних ситуацій. У медичному закладі, де експлуатуються інформаційні системи та значна кількість комп'ютерної техніки, БЖД базується на вимогах [52]:

 ДСТУ ISO 45001:2020 – визначає обов'язкові елементи системи управління охороною праці (політика, планування, підтримка, виконання, оцінка, поліпшення).

 НПАОП 0.00-4.12-05 «Типове положення про навчання з питань охорони праці» – регламентує інструктажі й перевірку знань персоналу.

 ДСанПіН 3.3.2-007-98 – санітарні норми для відеотерміналів.

 Правила пожежної безпеки в Україні (НАПБ А.01-001-2014).

У таблиці 4.1 наведено обов'язки роботодавця та працівника.

Таблиця 4.1 – Обов'язки роботодавця та працівника

Суб'єкт	**Ключові обов'язки (ст. 13, 14 ЗУ «Про охорону праці»)**
Роботодавець	створити безпечні умови, провести навчання, забезпечити СІЗ, здійснювати медогляди, розслідувати НВ, фінансувати БЖД не менше 0,5 % фонду оплати праці
Працівник	дбати про особисту безпеку та безпеку колег, знати і виконувати вимоги інструкцій, проходити навчання й медогляди

Небезпечні та шкідливі фактори при роботі з ПК:

1. Зорове перенапруження – тривале фокусування на екрані, низький контраст, мерехтіння.

2. Статичне навантаження м’язів спини й шиї – неправильна висота столу, відсутність підлокітників.

3. Електробезпека – можливість ураження при пошкодженні ізоляції або заземлення.

4. Пожежна небезпека – перегрів блоків живлення, коротке замикання.

5. Психофізіологічні фактори [53] – монотонність, інформаційні перевантаження.

Для кожного фактора у «Журналі оцінки ризиків» встановлюється категорія ризику (висока, середня, низька) та зазначається компенсуючий захід із посиланням на норматив.

Нормативні параметри робочого середовища наведено у таблиці 4.2.

Таблиця 4.2 – Нормативні параметри робочого середовища

Показник	**Граничне/оптимальне значення**	**Нормативний документ**
Освітленість робочої поверхні	400–500 лк	ДСанПіН 3.3.2-007-98
Температура повітря	22–24 °C (холодна пора), 23–26 °C (тепла)	ДСП 9.2.5-146-2005
Вологість	40–60 %	ДБН В.2.6-31
Рівень шуму	≤50 дБА	ДСН 3.3.6-037-99
ЕМ-поле моніторів	≤10 В/м	ДНАОП 0.03-8.07-94

Документи та навчання з БЖД згідно із [52]:

 Наказ про створення СУОП з призначенням відповідальних осіб.

 Інструкції (з електробезпеки, роботи з ПК, пожежної безпеки), погоджені профспілкою.

 Програма навчання – первинний, повторний, позаплановий інструктаж; журнали № 6-ОП та № 10-ОП.

 План заходів з охорони праці – затверджується щорічно; містить фінансування, терміни та відповідальних.

Комплексне виконання усіх норм та вимог забезпечить комфортні умови праці та безпеку на робочому місці.

Вимоги до профілактичних медичних оглядів для працівників ПК

Профілактичні медичні огляди для працівників, які працюють з відеодисплейними терміналами (ВДТ), є обов'язковою складовою системи охорони праці. Їх правове регулювання ґрунтується на ст. 17 Закону України «Про охорону праці» та Наказі МОЗ № 246 від 21.05.2007 р. (у редакції 2024 р.).

Мета медичних оглядів – раннє виявлення професійної патології та визначення придатності працівника до роботи з ВДТ. Основні нормативи:

 МОЗ № 246 – перелік шкідливих факторів та порядок проведення оглядів.

 ДСанПіН 3.3.2-007-98 – санітарні норми щодо організації робочих місць ВДТ [18].

 ДСТУ EN ISO 9241-5:2021 – ергономічні вимоги до роботи з екранною технікою.

Уточнений контингент (категорія 4.2) охоплює операторів, інженерів, IT-персонал, котрі проводять за ПК > 50 % робочого часу. Періодичність згідно з віком наведено у таблиці 4.3.

Таблиця 4.3 – Періодичність згідно з віком

Вік	**Попередній огляд**	**Періодичний огляд**
до 40 р.	при прийомі	1 раз / 2 роки
40–49 р.	при переході у вікову групу	щороку

≥ 50 р.	—	щороку

Порядок організації оглядів працівників наступний (на основі [52]):

1. Формування списків контингенту. Керівник підрозділу щороку до 1 грудня подає до відділу кадрів перелік працівників, що підлягають огляду.

2. Укладання договору. Заклад укладає угоду зі спеціалізованою медичною установою, яка має ліцензію на проведення профоглядів.

3. Направлення працівників. Відповідальна особа видає бланк форми № 1-ОМК із зазначенням умов праці та шкідливих факторів.

4. Проведення оглядів. Лікарі-спеціалісти заповнюють форму № 2-ОМК, виставляючи висновок «придатний», «придатний з обмеженнями» або «непридатний».

5. Підсумковий наказ. Служба охорони праці готує наказ про допуск/тимчасове відсторонення, зберігає результати в особових справах.

Фінансування та відповідальність за медогляди працівників здійснюються наступним чином:

 Фінансування. Роботодавець оплачує медогляди (п. 2 ст. 17 ЗУ «Про охорону праці»); витрати плануються в кошторисі на наступний фінансовий рік.

 Відповідальність. За недопущення працівника без чинного огляду до роботи передбачено адміністративну відповідальність (ст. 41-1 КУпАП) та штраф до 1700 грн.

 Контроль. Держпраці має право перевіряти журнали та направлення. У разі порушення видається припис і при необхідності зупиняються роботи з ВДТ.

Результати оглядів аналізуються комісією з ОП для оцінки тенденцій. Якщо більш ніж 10 % працівників отримують обмеження, роботодавець зобов’язаний переглянути умови праці: модернізувати робочі місця, оптимізувати графік, ініціювати повторні вимірювання освітленості та мікроклімату.

Висновок до четвертого розділу

В четвертому розділі кваліфікаційної роботи описано те, що впровадження системи менеджменту безпеки відповідно до ISO 45001 та чітке дотримання нормативів МОЗ щодо медичних оглядів персоналу ПК підвищують рівень охорони праці й знижують професійні ризики. Запропоновані заходи є невід’ємною складовою безпечної експлуатації медичних інформаційних систем.

Висновки

У кваліфікаційній роботі всебічно досліджено теоретичні засади та практичні підходи до забезпечення конфіденційності, цілісності й доступності медичних даних у процесі цифрової трансформації закладів охорони здоров'я. У результаті виконання поставлених завдань отримано такі узагальнені результати, що в сукупності відповідають вимогам бакалаврського рівня підготовки та можуть бути використані на практиці.

В першому розділі кваліфікаційної роботи освітнього рівня «Бакалавр»:

– Проаналізовано поточний стан кіберзахисту медичних інформаційних систем. Доведено, що масове впровадження електронних медичних карток, телемедицини, IoT-пристроїв і штучного інтелекту одночасно підвищує якість медичних послуг і породжує нові вектори загроз: програми-вимагачі, атаки на ланцюги постачання, соціальна інженерія. Історичний огляд показав еволюцію ризиків від мережевих вірусів до складних багатоетапних атак із використанням уразливостей «нульового дня».

– Розкрито концепцію цифрової стійкості медичних організацій, яка охоплює готовність, надійність та здатність до швидкого відновлення після інциденту. Показано доцільність інтеграції блокчейну для незмінності журналів доступу та підвищення довіри пацієнтів і регуляторів.

В другому розділі кваліфікаційної роботи:

– Обґрунтовано вибір мікросервісної архітектури (Docker + Kubernetes), стеку Django / React та СКБД PostgreSQL 16 із розширеннями pgcrypto та TimescaleDB. Сформовано домени атрибутів, діаграми класів і ER-схему, що забезпечують логічну цілісність даних.

– Запропоновано комбіноване використання AES-256-GCM (at rest) і гомоморфного шифрування CKKS (in use). Реалізовано політику Shamir Secret Sharing та HSM-інтеграцію у HashiCorp Vault, що забезпечує захист майстер-ключів і можливість поступового переходу до постквантових алгоритмів.

– Розроблено дворівневу RBAC-модель: App-level (Django Guardian) та DB-level (RLS-політики PostgreSQL із перевіркою JWT-клаїмів department_id, role). Медсестра обмежена атрибутом vital_signs, лікар має повний доступ, адміністратор керує ролями та політиками.

В третьому розділі кваліфікаційної роботи:

– Імплементовано шість алгоритмів сортування зашифрованих двовимірних масивів (горизонтальне, вертикальне, «равлик», зворотний «равлик», «змійка», діагональне) як окремі PHP-мікросервіси, що взаємодіють із Python-бекендом по gRPC поверх mTLS.

– Проведено unit-тести (pytest) для моделей і функцій шифрування/дешифрування; integration-тести (Postman + Newman) для повного шляху FHIR-Bundle; pen-тест (OWASP ZAP), який підтвердив відсутність критичних вразливостей після усунення XXE-атаки.

– Тест продуктивності Gatling («Bulk Upload», 500 користувачів) зафіксував середню затримку 180 мс та пік 340 мс на 99-му перцентилі, що задовольняє нефункціональні вимоги (< 250 мс на P95).

У розділі «Безпека життєдіяльності, основи охорони праці» сформовано систему менеджменту БЖД згідно з ДСТУ ISO 45001:2020; визначено небезпечні фактори, нормативні параметри середовища та процедури навчання персоналу. Висвітлено порядок, періодичність і документування профілактичних медичних оглядів працівників ПК (категорія 4.2) відповідно до наказу МОЗ № 246, що мінімізує професійні ризики та підвищує продуктивність.

Наукова та практична новизна. Запропоновано оригінальний підхід до інтеграції CKKS-бібліотеки SEAL через WebAssembly у React-клієнті, що дозволяє виконувати часткове дешифрування й візуалізацію лабораторних даних без розкриття серверу.

Продемонстровано життєздатність гомоморфного сортування як додаткового захисного шару проти побітового аналізу шифротексту.

Рекомендації щодо впровадження:

– Розгорнути систему в тестовому середовищі медичного закладу з попередньою міграцією 5 % реальних даних для пілотного проєкту.

– Налаштувати резервне копіювання Vault + PostgreSQL у віддалений регіон та впровадити політику Disaster Recovery TTR ≤ 30 хв.

– Забезпечити постійне оновлення залежностей мікросервісів (Dependabot) і регулярне сканування контейнерів на CVE.

Загалом виконана робота демонструє, що комплексне застосування сучасних криптографічних методів, мікросервісної архітектури та принципів «secure-by-design» дозволяє створити масштабовану, продуктивну й безпечну інформаційну систему для зберігання й обробки медичних даних. Розроблений прототип може бути основою для промислового впровадження, сприяти підвищенню довіри пацієнтів і забезпечити відповідність чинним міжнародним та національним стандартам у галузі охорони здоров’я.

Перелік джерел

1. Falat, P., Pasichnyk, V., Kunanets, N., Martsenko, S., Matsiuk, O., & Mytnyk, O. (2018). Telecommunication infrastructures for telemedicine in smart cities.

2. International Organization for Standardization. (2022). *ISO/IEC 27001:2022 — Information security, cybersecurity and privacy protection — Information security management systems — Requirements*.

3. U.S. Department of Health & Human Services. (2013). *HIPAA Security Rule*.

4. Kharchenko, O., Raichev, I., Bodnarchuk, I., & Zagorodna, N. (2018). Optimization of software architecture selection for the system under design and reengineering. *2018 14th International Conference on Advanced Trends in Radioelectronics, Telecommunications and Computer Engineering (TCSET)*, 1245–1248.

5. Basile, L. J., Carbonara, N., Pellegrino, R., & Panniello, U. (2022). Business intelligence in the healthcare industry: The utilization of a data-driven approach to support clinical decision making. *Technovation*, *120*, 102482. https://doi.org/10.1016/j.technovation.2022.102482

6. Greaves, F., Joshi, I., Campbell, M., Roberts, S., Patel, N., & Powell, J. (2018). What is an appropriate level of evidence for a digital health intervention? *The Lancet*, *392*(10165), 2665–2667. https://doi.org/10.1016/s0140-6736(18)33129-5

7. Shaygan, A., & Daim, T. (2021). Technology management maturity assessment model in healthcare research centers. Technovation, 102444. https://doi.org/10.1016/j.technovation.2021.102444

8. Hair, J. F., Sarstedt, M., & Ringle, C. M. (2019). Rethinking some of the rethinking of partial least squares. European Journal of Marketing, 53(4), 566–584. https://doi.org/10.1108/ejm-10-2018-0665

9. Cerchione, R., Centobelli, P., Riccio, E., Abbate, S., & Oropallo, E. (2022). Blockchain's coming to hospital to digitalize healthcare services: Designing a distributed electronic health record ecosystem. *Technovation*, *120*(1).

10. Saeed, S., Altamimi, S. A., Alkayyal, N. A., Alshehri, E., & Alabbad, D. A. (2023). Digital Transformation and Cybersecurity Challenges for Businesses Resilience: Issues and Recommendations. Sensors, 23(15). https://doi.org/10.3390/s23156666

11. Bagheri, S., Ridley, G., & Williams, B. (2023). Organisational Cyber Resilience: Management perspectives. *Australasian Journal of Information Systems*, *27*. https://doi.org/10.3127/ajis.v27i0.4183

12. Ree, E., Ellis, L. A., & Wiig, S. (2021). Managers' role in supporting resilience in healthcare: a proposed model of how managers contribute to a healthcare system's overall resilience. *International Journal of Health Governance, ahead-of-print(ahead-of-print)*. https://doi.org/10.1108/ijhg-11-2020-0129

13. Benitez, J., Henseler, J., Castillo, A., & Schuberth, F. (2020). How to perform and report an impactful analysis using partial least squares: Guidelines for confirmatory and explanatory IS research. *Information & Management, 57*(2), 103168. https://doi.org/10.1016/j.im.2019.05.003

14. Akinsola, A., & Akinde, A. (2024). Enhancing Software Supply Chain Resilience: Strategy For Mitigating Software Supply Chain Security Risks And Ensuring Security Continuity In Development Lifecycle. *International Journal on Soft Computing*, *15*(1/2), 01-18.

15. Kraus, S., Schiavone, F., Pluzhnikova, A., & Invernizzi, A. C. (2021). Digital transformation in healthcare: Analyzing the current state-of-research. *Journal of Business Research*, *123*(123), 557–567. sciencedirect. https://doi.org/10.1016/j.jbusres.2020.10.030

16. Blanchet, K., Nam, S. L., Ramalingam, B., & Pozo-Martin, F. (2017). Governance and Capacity to Manage Resilience of Health Systems: Towards a New Conceptual Framework. *International Journal of Health Policy and Management*, *6*(8), 431–435. https://doi.org/10.15171/ijhpm.2017.36

17. Scantlebury, A., Sheard, L., Fedell, C., & Wright, J. (2021). What are the implications for patient safety and experience of a major healthcare IT breakdown? A

qualitative study. *DIGITAL HEALTH*, *7*, 205520762110100. https://doi.org/10.1177/20552076211010033

18. Jovanović, A., Klimek, P., Renn, O., Schneider, R., Øien, K., Brown, J., DiGennaro, M., Liu, Y., Pfau, V., Jelić, M., Rosen, T., Caillard, B., Chakravarty, S., & Chhantyal, P. (2020). Assessing resilience of healthcare infrastructure exposed to COVID-19: emerging risks, resilience indicators, interdependencies and international standards. *Environment Systems and Decisions*, *40*(2), 252–286. https://doi.org/10.1007/s10669-020-09779-8

19. Offner, K. L., Sitnikova, E., Joiner, K., & MacIntyre, C. R. (2020). Towards understanding cybersecurity capability in Australian healthcare organisations: a systematic review of recent trends, threats and mitigation. Intelligence and National Security, 35(4), 556–585. https://doi.org/10.1080/02684527.2020.1752459

20. Manyena, S. B. (2014). The concept of resilience to disasters. *Community, Environment and Disaster Risk Management*, 35–48. https://doi.org/10.1108/s2040-726220140000015002

21. Carthey, J. (2001). Institutional resilience in healthcare systems. *Quality in Health Care*, *10*(1), 29–32. https://doi.org/10.1136/qhc.10.1.29

22. Ejiofor Oluomachi, & Ahmed, A. (2024). Securing the Future of Healthcare: Building a Resilient Defense System for Patient Data Protection. *Computer Science and Information Technology*, 27–39. https://doi.org/10.5121/csit.2024.141303

23. Annarelli, A., & Nonino, F. (2016). Strategic and operational management of organizational resilience: Current state of research and future directions. *Omega*, *62*(1), 1–18.

24. Палка, О. В. (2023). Аналіз інтегрованої архітектури розумного міста з блокчейном та IoT. *Scientific Bulletin of UNFU*, *33*(6), 94-99.

25. Jafar, U., Ab Aziz, M. J., Shukur, Z., & Hussain, H. A. (2022). A Systematic Literature Review and Meta-Analysis on Scalable Blockchain-Based Electronic Voting Systems. *Sensors*, *22*(19), 7585. https://doi.org/10.3390/s22197585

26. Khan, S. N., Loukil, F., Ghedira-Guegan, C., Benkhelifa, E., & Bani-Hani, A. (2021). Blockchain smart contracts: Applications, challenges, and future trends. *Peer-To-Peer Networking and Applications*, *14*(1), 2901–2925. https://doi.org/10.1007/s12083-021-01127-0

27. Bodkhe, U., Tanwar, S., Parekh, K., Khanpara, P., Tyagi, S., Kumar, N., & Alazab, M. (2020). Blockchain for Industry 4.0: A Comprehensive Review. *IEEE Access*, *8*(1), 79764–79800. https://doi.org/10.1109/access.2020.2988579

28. Rupa, Ch., Midhunchakkaravarthy, D., Kamrul Hasan, M., Alhumyani, H., & Saeed, R. A. (2021). Industry 5.0: Ethereum blockchain technology based DApp smart contract. *Mathematical Biosciences and Engineering*, *18*(5), 7010–7027. https://doi.org/10.3934/mbe.2021349

29. Yaqoob, I., Salah, K., Jayaraman, R., & Al-Hammadi, Y. (2021). Blockchain for Healthcare Data management: opportunities, challenges, and Future Recommendations. *Neural Computing and Applications*, *34*(2), 1–16. https://doi.org/10.1007/s00521-020-05519-w

30. Odeh, A., Keshta, I., & Al-Haija, Q. A. (2022). Analysis of Blockchain in the Healthcare Sector: Application and Issues. *Symmetry*, *14*(9), 1760. https://doi.org/10.3390/sym14091760

31. Tandon, A., Dhir, A., Islam, N., & Mäntymäki, M. (2020). Blockchain in healthcare: A systematic literature review, synthesizing framework and future research agenda. *Computers in Industry*, *122*(103290), 103290. https://doi.org/10.1016/j.compind.2020.103290

32. Sarkar, A., Maitra, T., & Neogy, S. (2021). Blockchain in Healthcare System: Security Issues, Attacks and Challenges. *Intelligent Systems Reference Library*, 113–133. https://doi.org/10.1007/978-3-030-69395-4_7

33. Chelladurai, U., & Pandian, S. (2021). A novel blockchain based electronic health record automation system for healthcare. *Journal of Ambient Intelligence and Humanized Computing*, *13*. https://doi.org/10.1007/s12652-021-03163-3

34. Fatima, N., Agarwal, P., & Sohail, S. S. (2022). Security and Privacy Issues of Blockchain Technology in Health Care—A Review. *ICT Analysis and Applications*, 193–201. https://doi.org/10.1007/978-981-16-5655-2_18

35. Hasan, H. R., Salah, K., Jayaraman, R., Yaqoob, I., Omar, M., & Ellahham, S. (2021). Blockchain-Enabled Telehealth Services Using Smart Contracts. *IEEE Access*, *9*, 151944–151959. https://doi.org/10.1109/access.2021.3126025

36. Abu-elezz, I., Hassan, A., Nazeemudeen, A., Househ, M., & Abd-alrazaq, A. (2020). The benefits and threats of blockchain technology in healthcare: A scoping review. *International Journal of Medical Informatics*, *142*(1). https://doi.org/10.1016/j.ijmedinf.2020.104246

37. Zheng, X., Zhu, Y., & Si, X. (2019). A Survey on Challenges and Progresses in Blockchain Technologies: A Performance and Security Perspective. *Applied Sciences*, *9*(22), 4731. https://doi.org/10.3390/app9224731

38. S. Naresh, V., S. Pericherla, S., Sita Rama Murty, P., & Reddi, S. (2020). Internet of Things in Healthcare: Architecture, Applications, Challenges, and Solutions. *Computer Systems Science and Engineering*, *35*(6), 411–421. https://doi.org/10.32604/csse.2020.35.411

39. Idrees, S. M., Nowostawski, M., Jameel, R., & Mourya, A. K. (2021). Security Aspects of Blockchain Technology Intended for Industrial Applications. *Electronics*, *10*(8), 951. https://doi.org/10.3390/electronics10080951

40. Haleem, A., Javaid, M., Singh, R. P., Suman, R., & Rab, S. (2021). Blockchain Technology Applications in Healthcare: An Overview. *International Journal of Intelligent Networks*, *2*(2), 130–139. Sciencedirect. https://doi.org/10.1016/j.ijin.2021.09.005

41. Tariq, N., Qamar, A., Asim, M., & Khan, F. A. (2020). Blockchain and Smart Healthcare Security: A Survey. *Procedia Computer Science*, *175*, 615–620. https://doi.org/10.1016/j.procs.2020.07.089

42. Kamruzzaman, M. M., Yan, B., Sarker, M. N. I., Alruwaili, O., Wu, M., & Alrashdi, I. (2022). Blockchain and Fog Computing in IoT-Driven Healthcare

Services for Smart Cities. *Journal of Healthcare Engineering*, *2022*, 1–13. https://doi.org/10.1155/2022/9957888

43. Kumar, A., & Barik, S. (2025). A comprehensive survey on secure healthcare data processing with homomorphic encryption. *Journal of Healthcare Engineering*, 5(3), 1–23.

44. Health Level Seven International. (2023). *FHIR Specification R5*.

45. Clancy, B. (2025). Moving beyond traditional data protection: Homomorphic encryption could provide what is needed for artificial intelligence. *AHIMA Journal*.

46. Asabor, M., et al. (2024). Privacy-preserving federated transfer learning for cardiac arrhythmia classification using homomorphic encryption. *International Journal of Computer Science and Medicine*, 5(3), 250–268.

47. Li, J., Chen, Y., & Brown, L. (2023). Electronic health record data quality assessment and tools. *Journal of the American Medical Informatics Association*, 30(10), 1730–1740.

48. Johnson, D. (2024). Post-quantum cryptography readiness in healthcare IT. *Information Security Journal: A Global Perspective*, 33(4), 215–230.

49. Hasani, H., & Al Dasous, S. (2023). A comprehensive health electronic record system with MySQL and MongoDB. *SSRN Electronic Journal*. https://doi.org/10.2139/ssrn.4548519

50. O'Neil, P. (2022). Role-based access control models in health information systems. *Health Informatics Journal*, 28(1), 112–128.

51. Radyanin, S. (2024). Testing security of medical data systems: A penetration framework. *Cybersecurity & Medicine*, 12(2), 33–47.

52. Гурик, О. Я., Окіпний, І. Б., Сенчишин, В. С., Мариненко, С. Ю., & Король, О. І. (2025). *Безпека життєдіяльності, основи охорони праці: навчально-методичний посібник*. Тернопіль: ТНТУ імені Івана Пулюя. 123 с.

53. Палка, О., Станько, А., Шимчук, Г., & Герасимчук, О. (2021). Запобігання поширення коронавірусної інфекції у «розумних містах».

Комп'ютерно-інтегровані технології: освіта, наука, виробництво, (42), 79–88. https://doi.org/10.36910/6775-2524-0560-2021-42-12

ДОДАТКИ

Додаток А

SQL-DDL і міграції

Нижче наведено основні DDL-скрипти. Файл 0001_initial.sql створює всі сутності бази, зокрема таблиці користувачів і ролей. У лістингу А.1 наведено SQL-запит та створення усіх таблиць та ролей.

Лістинг А.1 – SQL-запит та створення усіх таблиць та ролей

```
-- Таблиця користувачів
CREATE TABLE "user" (
    user_id UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    username VARCHAR(60) UNIQUE NOT NULL,
    password_hash TEXT NOT NULL,
    department_id INT,
    is_active BOOLEAN NOT NULL DEFAULT TRUE,
    created_at TIMESTAMP NOT NULL DEFAULT now()
);

-- Таблиця ролей
CREATE TABLE role (
    role_id SERIAL PRIMARY KEY,
    role_name VARCHAR(40) UNIQUE NOT NULL
);

-- Зв'язок N:M між користувачами та ролями
CREATE TABLE user_roles (
    user_id UUID REFERENCES "user"(user_id) ON DELETE CASCADE,
    role_id INT  REFERENCES role(role_id)   ON DELETE CASCADE,
    PRIMARY KEY (user_id, role_id)
);

-- Аудит-лог дій користувачів
CREATE TABLE audit_log (
    log_id    BIGSERIAL PRIMARY KEY,
    user_id   UUID REFERENCES "user"(user_id),
    action    VARCHAR(120),
    timestamp TIMESTAMP NOT NULL DEFAULT now()
);

-- Зашифровані ключі пацієнтів
CREATE TABLE encryption_key (
    key_id      UUID PRIMARY KEY DEFAULT uuid_generate_v4(),
    patient_id  UUID UNIQUE REFERENCES patient(patient_id) ON
DELETE CASCADE,
    wrapped_key BYTEA NOT NULL
);
```

У лістингу А.2 наведено створення додаткових індексів та тригерів журналювання.

Лістинг А.2 – Додаткові індекси та тригери журналювання

```
-- Індекси для пошуку
CREATE INDEX idx_user_username ON "user"(LOWER(username));
CREATE INDEX idx_patient_lname ON patient(LOWER(last_name));
CREATE INDEX idx_record_patient ON medical_record(patient_id);

-- Тригер для автоматичного логування змін у medical_record
CREATE OR REPLACE FUNCTION log_record() RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO audit_log(user_id, action) VALUES
(current_setting('app.current_user_id')::uuid, TG_OP);
  RETURN NEW;
END; $$ LANGUAGE plpgsql;
CREATE TRIGGER trg_log_record AFTER INSERT OR UPDATE OR DELETE ON
medical_record FOR EACH ROW EXECUTE FUNCTION log_record();

-- Створення індексів для швидкого пошуку
CREATE INDEX idx_patient_lname ON patient(LOWER(last_name));
CREATE INDEX idx_record_patient ON medical_record(patient_id);

-- Тригер для автоматичного журналювання
CREATE OR REPLACE FUNCTION log_record() RETURNS TRIGGER AS $$
BEGIN
  INSERT INTO audit_log(user_id, action) VALUES
(current_setting('app.current_user_id')::uuid, TG_OP);
  RETURN NEW;
END; $$ LANGUAGE plpgsql;
CREATE TRIGGER trg_log_record AFTER INSERT OR UPDATE OR DELETE ON
medical_record FOR EACH ROW EXECUTE FUNCTION log_record();
```

Додаток Б

Backend (Django + DRF)

У лістингу Б.1 наведено вміст файлу settings.py.

Лістинг Б.1 – Вміст файлу settings.py

```
INSTALLED_APPS = [
    'django.contrib.admin',
    'django.contrib.auth',
    'rest_framework',
    'corsheaders',
    'api',
]
AUTH_USER_MODEL = 'api.User'
REST_FRAMEWORK = {
    'DEFAULT_AUTHENTICATION_CLASSES': [
        'rest_framework.authentication.JWTAuthentication',
    ],
    'DEFAULT_PERMISSION_CLASSES':
['rest_framework.permissions.IsAuthenticated'],
}
DATABASES = {
    'default': {
        'ENGINE': 'django.db.backends.postgresql',
        'NAME': 'medvault',
        'USER': 'medroot',
        'PASSWORD': 'changeme',
        'HOST': '127.0.0.1',
        'PORT': '5432',
    }
}
```

У лістингу Б.2 наведено вміст файлу models.py.

Лістинг Б.2 – Вміст файлу models.py

```
from django.db import models
from django.contrib.auth.models import AbstractBaseUser,
PermissionsMixin
from .managers import UserManager

class User(AbstractBaseUser, PermissionsMixin):
    user_id = models.UUIDField(primary_key=True, editable=False)
    username = models.CharField(max_length=60, unique=True)
    department_id = models.IntegerField(null=True)
    is_active = models.BooleanField(default=True)
```

```
    USERNAME_FIELD = 'username'
    objects = UserManager()

class Patient(models.Model):
    patient_id = models.UUIDField(primary_key=True,
editable=False)
    first_name = models.CharField(max_length=60)
    last_name = models.CharField(max_length=60)
    birth_date = models.DateField()
    gender = models.CharField(max_length=1,
choices=[('M','Ч'),('F','Ж'),('O','I')])
```

У лістингу Б.3 наведено вміст файлу serializers.py.

Лістинг Б.3 – Вміст файлу serializers.py

```
from rest_framework import serializers
from .models import Patient, MedicalRecord

class PatientSerializer(serializers.ModelSerializer):
    class Meta:
        model = Patient
        fields = '__all__'

class MedicalRecordSerializer(serializers.ModelSerializer):
    class Meta:
        model = MedicalRecord
        fields = '__all__'
```

У лістингу Б.4 наведено вміст файлу urls.py.

Лістинг Б.4 – Вміст файлу urls.py

```
from rest_framework.routers import DefaultRouter
from .views import PatientViewSet, MedicalRecordViewSet

router = DefaultRouter()
router.register(r'patients', PatientViewSet)
router.register(r'records', MedicalRecordViewSet)
urlpatterns = router.urls
```

Додаток В

Frontend (React 18 + TypeScript)

У лістингу В.1 наведено вміст файлу api.ts.

Лістинг В.1 – Вміст файлу api.ts

```
import axios from 'axios';
const api = axios.create({ baseURL: 'http://localhost:8000/api/'
});
export default api;
```

У лістингу В.2 наведено вміст файлу PatientForm.tsx.

Лістинг В.2 – Вміст файлу PatientForm.tsx

```
import { useForm } from 'react-hook-form';
import api from '@/lib/api';

type Patient = { first_name: string; last_name: string;
birth_date: string; gender: string };
export default function PatientForm() {
  const { register, handleSubmit } = useForm<Patient>();
  const onSubmit = async (data: Patient) => { await
api.post('patients/', data); };
  return (
    <form onSubmit={handleSubmit(onSubmit)} className="grid gap-
2">
      <input {...register('first_name')} placeholder="Ім'я" />
      <input {...register('last_name')} placeholder="Прізвище" />
      <input type="date" {...register('birth_date')} />
      <select {...register('gender')}><option
value="M">Ч</option><option value="F">Ж</option></select>
      <button>Зберегти</button>
    </form>
  );
```

Додаток Д

Сервіс сортування (PHP 8.3)

У лістингу Д.1 наведено повний вихідний код абстрактного класу MySort та шести реалізацій алгоритмів сортування двовимірних масивів. Класи призначені для роботи як із розшифрованими даними (тип array), так і з фрагментами, отриманими після часткового дешифрування CKKS-блоків.

Лістинг Д.1 – Повний вихідний код абстрактного класу MySort

```
<?php
/**
 * Базовий клас сервісу сортування.
 * Надає спільні утиліти: пряме/зворотне сортування, "плоску"
конвертацію та збірку матриці.
 */
abstract class MySort {
    /** Пряме сортування одномірного масиву */
    protected function my_sort(array $arr): array { sort($arr,
SORT_NUMERIC); return $arr; }
    /** Зворотне сортування одномірного масиву */
    protected function my_rsort(array $arr): array { rsort($arr,
SORT_NUMERIC); return $arr; }

    /** Перетворює двовимірний масив у плоский */
    protected function flatten(array $matrix): array {
        $flat = [];
        foreach ($matrix as $row) { foreach ($row as $v) { $flat[]
= $v; } }
        return $flat;
    }

    /** Повертає квадратну матрицю з плоского масиву */
    protected function arrToMatrix(array $flat): array {
        $n = (int) sqrt(count($flat));
        return array_chunk($flat, $n);
    }

    /** Головний метод-шаблон: повертає відсортовану матрицю */
    abstract public function transform(array $matrix): array;
}
```

У лістингу Д.2 наведено HorizontalSort – горизонтальне (row-major) заповнення.

Лістинг Д.2 – HorizontalSort – горизонтальне (row-major) заповнення

```
class HorizontalSort extends MySort {
```

```
    public function transform(array $matrix): array {
        $flat = $this->flatten($matrix);
        $flat = $this->my_sort($flat);
        return $this->arrToMatrix($flat); // рядок за рядком
    }
}
```

У лістингу Д.3 наведено VerticalSort – вертикальне (column-major) заповнення.

Лістинг Д.3 – VerticalSort – вертикальне (column-major) заповнення

```
class VerticalSort extends MySort {
    public function transform(array $matrix): array {
        $flat = $this->flatten($matrix);
        $flat = $this->my_sort($flat);
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $idx = 0;
        for ($c = 0; $c < $n; $c++) {
            for ($r = 0; $r < $n; $r++) {
                $res[$r][$c] = $flat[$idx++];
            }
        }
        return $res;
    }
}
```

У лістингу Д.4 наведено SnailSort – «равлик» за годинниковою стрілкою.

Лістинг Д.4 – SnailSort – «равлик» за годинниковою стрілкою

```
class SnailSort extends MySort {
    public function transform(array $matrix): array {
        $flat = $this->my_sort($this->flatten($matrix));
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $top = 0; $left = 0; $bottom = $n - 1; $right = $n - 1;
$idx = 0;
        while ($top <= $bottom && $left <= $right) {
            for ($i = $left; $i <= $right; $i++)  $res[$top][$i] =
$flat[$idx++];
            $top++;
            for ($i = $top; $i <= $bottom; $i++) $res[$i][$right]
= $flat[$idx++];
            $right--;
            if ($top <= $bottom) { for ($i = $right; $i >= $left;
$i--) $res[$bottom][$i] = $flat[$idx++]; }
```

```
            $bottom--;
            if ($left <= $right) { for ($i = $bottom; $i >= $top;
$i--) $res[$i][$left] = $flat[$idx++]; }
            $left++;
        }
        return $res;
    }
}
```

У лістингу Д.5 наведено InverseSnailSort – зворотний «равлик» проти годинникової стрілки.

Лістинг Д.5 – InverseSnailSort – зворотний «равлик» проти годинникової стрілки

```
class InverseSnailSort extends MySort {
    public function transform(array $matrix): array {
        $flat = $this->my_rsort($this->flatten($matrix)); //
спадаючий порядок
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $top = 0; $left = 0; $bottom = $n - 1; $right = $n - 1;
$idx = 0;
        while ($top <= $bottom && $left <= $right) {
            for ($i = $left; $i <= $right; $i++)
$res[$bottom][$i] = $flat[$idx++];
            $bottom--;
            for ($i = $bottom; $i >= $top; $i--) $res[$i][$right]
= $flat[$idx++];
            $right--;
            if ($top <= $bottom) { for ($i = $right; $i >= $left;
$i--) $res[$top][$i] = $flat[$idx++]; }
            $top++;
            if ($left <= $right) { for ($i = $top; $i <= $bottom;
$i++) $res[$i][$left] = $flat[$idx++]; }
            $left++;
        }
        return $res;
    }
}
```

У лістингу Д.6 наведено SnakeSort – «змійка» з чергуванням напрямку рядків.

Лістинг Д.6 – SnakeSort – «змійка» з чергуванням напрямку рядків

```
class SnakeSort extends MySort {
```

```
    public function transform(array $matrix): array {
        $flat = $this->my_sort($this->flatten($matrix));
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $idx = 0;
        for ($r = 0; $r < $n; $r++) {
            if ($r % 2 == 0) {
                for ($c = 0; $c < $n; $c++) $res[$r][$c] =
$flat[$idx++];
            } else {
                for ($c = $n - 1; $c >= 0; $c--) $res[$r][$c] =
$flat[$idx++];
            }
        }
        return $res;
    }
}
```

У лістингу Д.7 наведено DiagonalSort – діагональ від верхнього лівого до нижнього правого.

Лістинг Д.7 – DiagonalSort – діагональ від верхнього лівого до нижнього правого

```
class DiagonalSort extends MySort {
    public function transform(array $matrix): array {
        $flat = $this->my_sort($this->flatten($matrix));
        $n = (int) sqrt(count($flat));
        $res = array_fill(0, $n, array_fill(0, $n, null));
        $idx = 0;
        // верхні діагоналі
        for ($d = 0; $d < $n; $d++) {
            $r = 0; $c = $d;
            while ($c >= 0) { $res[$r++][$c--] = $flat[$idx++]; }
        }
        // нижні діагоналі
        for ($d = 1; $d < $n; $d++) {
            $r = $d; $c = $n - 1;
            while ($r < $n) { $res[$r++][$c--] = $flat[$idx++]; }
        }
        return $res;
    }
}
```

Усі класи є stateless і можуть виконуватися у функціях-лямбдах FaaS, якщо потрібна еластичність. Для перевірки алгоритмів достатньо викликати,

наприклад, (new SnailSort())->transform($matrix), де $matrix – це результат часткового дешифрування CKKS-блоку.