

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

ПОЯСНЮВАЛЬНА ЗАПИСКА

до кваліфікаційної роботи освітнього ступеня «бакалавр»
за спеціальністю 121 «Інженерія програмного забезпечення»
(освітня програма «Веб-технології»)

на тему:

«Вебпортал приватної поліклініки»

Виконала студентка групи ВТ-22-2
КАРПЕНКО Аліна Сергіївна

Керівник роботи:
ФАНТ Микола Олександрович

Рецензент:
ЄФРЕМОВ Юрій Миколайович

Житомир – 2026

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ДЕРЖАВНИЙ УНІВЕРСИТЕТ «ЖИТОМИРСЬКА ПОЛІТЕХНІКА»
ФАКУЛЬТЕТ ІНФОРМАЦІЙНО-КОМП'ЮТЕРНИХ ТЕХНОЛОГІЙ
КАФЕДРА ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії
програмного забезпечення

Тетяна Вакалюк

«28» лютого 2026 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: КАРПЕНКО Аліна Сергіївна

Керівник роботи: ФАНТ Микола Олександрович

Тема роботи: **«Вебпортал приватної поліклініки»**,

затверджена Наказом закладу вищої освіти від «27» лютого 2026 р., №92/с

Вихідні дані для роботи: **вебпортал для керування медичним закладом.**

Консультанти з бакалаврської кваліфікаційної роботи із зазначенням розділів,
що їх стосуються:

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Фант М.О.	01.03.2026р.	01.03.2026р
2	Фант М.О.	06.04.2026р	06.04.2026р
3	Фант М.О.	04.05.2026р	04.05.2026р

РЕФЕРАТ

У кваліфікаційній роботі розглянуто питання проєктування та реалізації вебпорталу приватної поліклініки, призначеного для автоматизації основних процесів медичного закладу. Окрему увагу приділено побудові реляційної бази даних, механізмам розмежування доступу до медичної інформації, шифруванню персональних даних і файлів, а також реалізації функціоналу для адміністраторів, лікарів і пацієнтів. У роботі також описано структуру інтерфейсу, процес розгортання вебпорталу та результати мануального тестування основних сценаріїв використання.

У роботі проаналізовано предметну область, досліджено наявні аналоги медичних інформаційних систем, обґрунтовано вибір клієнт-серверної архітектури та інструментальних засобів розробки.

Метою роботи є аналіз, проєктування та реалізація вебпорталу приватної поліклініки для автоматизації внутрішніх процесів медичного закладу, забезпечення зручної взаємодії між пацієнтами, лікарями та адміністраторами, а також підвищення безпеки зберігання й обробки медичних даних.

Випускна робота містить 69 сторінок основного тексту, 33 ілюстрації, 24 таблиці, 5 додатків та 30 використаних літературних джерел.

КЛЮЧОВІ СЛОВА: КЛІЄНТ-СЕРВЕРНА АРХІТЕКТУРА, МЕДИЧНИЙ ЗАКЛАД, ПРИВАТНА ПОЛІКЛІНІКА, РОЗМЕЖУВАННЯ ДОСТУПУ, ВЕБПОРТАЛ

					ІПЗ.КР.Б-121.26 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата							
Розроб.		Карпенко А.С.			Вебпортал приватної поліклініки			Літ.	Арк.	Аркушів	
Керівник		Фант М.О.								3	75
Рецензент		Єфремов Ю.М.						ФІКТ, гр. ВТ-22-2			
Зав. Каф.		Вакалюк. Т.А.									

ABSTRACT

The qualification work considers the issue of designing and implementing a web portal for a private clinic designed to automate the main processes of a medical institution. Special attention is paid to building a relational database, mechanisms for separating access to medical information, encrypting personal data and files, as well as implementing functionality for administrators, doctors and patients. The work also describes the interface structure, the web portal deployment process, and the results of manual testing of the main usage scenarios.

The work analyzes the subject area, investigates existing analogues of medical information systems, and justifies the choice of a modular client-server architecture and modern development tools.

The purpose of the work is to analyze, design, and implement a web portal for a private clinic to automate the internal processes of a medical institution, ensure convenient interaction between patients, doctors, and administrators, and increase the security of storing and processing medical data.

The thesis contains 69 pages of main text, 33 illustrations, 24 tables, 5 appendices and 30 used literary sources.

KEYWORDS: CLIENT-SERVER ARCHITECTURE, MEDICAL INSTITUTION, PRIVATE POLYCLINIC, ACCESS DELIMITATION, WEB PORTAL

					ІПЗ.КР.Б-121.26 - ПЗ						
Змн.	Арк.	№ докум.	Підпис	Дата	Вебпортал приватної поліклініки			Літ.	Арк.	Аркушів	
Розроб.		Карпенко А.С.									
Керівник		Фант М.О.								4	75
								ФІКТ, зр. ВТ-22-2			
Рецензент		Єфремов Ю.М.									
Зав. Каф.		Вакалюк. Т.А.									

ЗМІСТ

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	6
ВСТУП	7
РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧ.....	9
1.1. Постановка задачі	9
1.2. Аналіз аналогів програмного забезпечення.....	11
1.3. Вибір архітектури web-орієнтованої системи.....	16
1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення	20
Висновки до першого розділу.....	23
РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПОРТАЛУ ПРИВАТНОЇ ПОЛІКЛІНІКИ.....	24
2.1. Визначення варіантів використання та структури вебпорталу	24
2.2. Розробка баз даних.....	30
2.3. Реалізація функціоналу вебпорталу приватної поліклініки	42
Висновки до другого розділу	44
РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБПОРТАЛУ	45
3.1. Розгортання вебпорталу	45
3.2. Структура інтерфейсу.....	48
3.3. Тестування роботи вебпорталу	62
Висновки до третього розділу	66
ВИСНОВКИ.....	67
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	69
ДОДАТКИ.....	72

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

БД – База даних

MUI – Material UI

UI – User Interface

JS – JavaScript

ORM – Object-Relational Mapping

HTTP – HyperText Transformer Protocol

JSX – JavaScript XML

СУБД – Система управління базою даних

SPA – Single Page Application

МІС – Медична інформаційна система

RBAC – Relationship-Based Access Control

API – Application Programming Interface

REST API – Representational State Transfer

ВСТУП

Актуальність теми сучасні web-орієнтовані системи дозволяють організувати зручну та доступну з будь-якого пристрою через браузер, вебпорталу для медичного закладу. Медичні заклади все частіше впроваджують інформаційні системи для автоматизації внутрішніх процесів, що забезпечило підвищення ефективної роботи, покращило облік пацієнтів та реалізувало безпечне зберігання медичних даних.

У більшості існуючих рішеннях проблемою є те, що пацієнти не мають інструментів контролю на поширенням доступу до своєї медичної історії, що заперечує принципам медичної таємниці. У зв'язку з цим виникає потреба у створенні нових рішень, які поєднують у собі модулі адміністрування медичним закладом та нададуть пацієнту повний контроль над своєю медичною історією лікування.

Таким чином, в умовах розвитку технологій, суттєвого збільшення обсягів інформації та критичної необхідності швидкого доступу до потрібних даних, традиційні паперові системи обліку стають неефективними.

Мета дослідження є аналіз та реалізація програмного комплексу вебпорталу приватної поліклініки, спрямованого на автоматизацію адміністративних та клінічних процесів закладу охорони здоров'я. Для досягнення цієї мети є розробка вебпорталу з чотирма функціональними частинами, що включають в себе повністю публічну сторінку, особистий кабінет пацієнта, лікаря та адміністратора. Також передбачається інтеграція симетричного шифрування електронних медичних карт та допоміжних документів із впровадженням моделі розмежування прав доступу. Також реалізація вебпорталу приватної поліклініки передбачає використання сучасних веб-технологій та інструментів.

Отже, практичне значення полягає у створенні вебпорталу приватної поліклініки, який стійкий до ризиків витоку інформації та несанкціонованого доступу до персональних та медичних даних із забезпеченням якості взаємодії між учасниками медичного процесу.

Завданням дослідження є досягнення поставленої мети, шляхом виконання визначених етапів:

- здійснення детального аналізу аналогів у сфері медичних інформаційних систем;
- дослідження існуючих архітектурних підходів;
- аналіз вимог щодо захисту персональних даних;
- проєктування архітектури вебпорталу;
- обґрунтування та вибір інструментальних засобів;
- реалізація моделі розмежування прав доступу та моделювання бізнес-процесів;
- дослідження існуючих алгоритмів шифрування, які забезпечать максимальну стійкість до злому при цьому гарантуючи цілісність даних на рівні бази даних;
- програмна реалізація серверної та клієнтської частини вебпорталу;
- проведення тестування працездатності роботи вебпорталу.

Методи аналізу та інформаційна база. Для вирішення поставлених завдань у роботі застосовано методи порівняльного моніторингу існуючих аналогів інформаційних систем та методи структурного аналізу, за допомогою діаграм прецедентів та проєктування схем даних.

Об'єкт дослідження є веб-технології для розробки вебпорталу приватної поліклініки, які забезпечують реалізацію функціоналу, який необхідний для автоматизації внутрішніх процесів закладу.

Предмет дослідження є структура та методи реалізації web-орієнтованого програмного комплексу для медичного закладу, що включають програмні алгоритми, архітектурні моделі та технологічні методи побудови односторінкових вебдодатків (SPA) для автоматизації взаємодії між учасниками медичного процесу, а також криптографічні механізми.

РОЗДІЛ 1. АНАЛІЗ ПРОБЛЕМАТИКИ, МЕТОДІВ ТА ЗАСОБІВ ВИРІШЕННЯ ЗАДАЧ

1.1. Постановка задачі

Наразі, медичні заклади характеризуються значним обсягом інформації, де введення цієї інформації вручну стає значно важче. Тому медичні заклади потребують систематизованого зберігання, швидкої обробки та оперативного доступу з даними які пов'язані з обліком пацієнтів, організацією прийомів та введення медичної документації.

У зв'язку з цим актуальною є задача розробки вебпорталу приватної поліклініки, яка і дозволить автоматизувати основні процеси та забезпечити доступ до системи з використанням стандартного веб-браузера. Основною функціональністю такої порталу є реєстрація, введення медичних карт та нотаток, розмежування прав доступу користувачів та управління розкладом прийому.

Для вирішення поставленої задачі є доцільним використання сучасних веб-технологій, що підтримують модульну організацію коду, ефективну взаємодію між клієнтською та серверною частиною та асинхронну обробку даних. Важливо зазначити, що застосування модульного підходу до організації програмного коду є одним із ключовим методів вирішення задачі, який передбачає поділ системи на окремі функціональні компоненти. Крім того, модульна організація дозволить ефективно розподілити відповідальність між компонентами системи.

Для реалізації модульної системи доцільніше використання порталу Node.js, яка забезпечить високу продуктивність та асинхронну модель обробки запитів. У якості серверного фреймворку краще використати Express.js, який надає засоби для реалізації REST-підходу. Клієнтська частина ж реалізовуватиметься з використанням бібліотеки React [22].

Використання даного підходу до програмної системи забезпечить масштабованість, зручність використання та можливість подальшого розширення функціоналу.

Оскільки вебпортал керує конфіденційною інформацією, виникає необхідність інтеграції з сучасними алгоритмами шифрування, які підходять для зберігання цілісності та забезпечення безпеки для інформації, яка зберігається у базі даних, тому що, електронні медичні картти, діагнози, нотатки лікарів та завантаженні файли не можуть зберігатися у відкритому вигляді через високі ризики несанкціонованого доступу. Вирішення цієї задачі вимагає використання симетричного алгоритму шифрування AES-256-GCM, вибір даного стандарту зумовлений здатністю автентифікації даних, що гарантує не лише конфіденційність, а й контроль цілісності за допомогою автоматичної генерації тегів автентифікації. Тому, будь яка спроба модифікації зашифрованих даних на рівні бази даних буде миттєво блокуватися системою на етапі дешифрування.

Для організації збереження інформації доцільно застосувати реляційну систему управління базами даних PostgreSQL, яка забезпечує високу стабільність під час роботи зі складними зв'язками, строгу структурування та підтримку транзакційності. Взаємодія між сервером та самою базою даних повинна відбуватися через об'єктно-реляційне відображення, що забезпечить безпеку від SQL-ін'єкцій та спрощення міграцій схем даних. Для реалізації даного зв'язку доцільно використати Sequalize ORM, яка працюватиме з таблицями бази даних за допомогою простих JS-об'єктів та також забезпечує безпеку.

Окремим та основним завданням у межах проектування бази даних є реалізація моделі розмежування прав доступу на основі відносин (RBAC). Створюваний вебпортал повинен надавати пацієнту конкретну роль повноправного володільця власної медичної історії та додаткових персональних даних. Реалізація передбачає розробку механізму запитів на доступ, де лікарю відповідно дозволу буде надано право на перегляд та додавання медичної інформації.

1.2. Аналіз аналогів програмного забезпечення

Наразі існує багато аналогів медичних систем для введення медичних записів, як міжнародних, так і українських. Найвідомішим в Україні, такою системою є Helsi (рис. 1.2), яка інтегрує роботу медичних закладів і забезпечує облік мільйонам електронних записів пацієнтів.

Дана система представлена у вигляді вебсайту, де користувач може сам зареєструватися та вибрати лікаря або медичний заклад відповідно до його потреб.

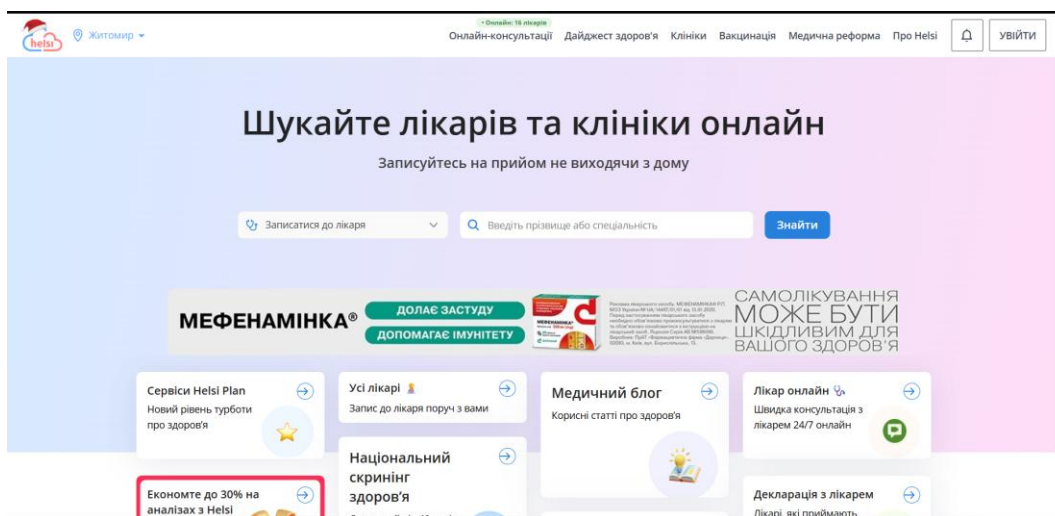


Рис. 1.2. Головна сторінка Helsi

З функціональної точки зору, реалізована система забезпечує зручний механізм пошуку медичних закладів, що спрямований на підвищення зручності взаємодії із системою. Користувач має розширений пошук з можливістю фільтрації за типом закладу (приватний або державним) та географічним розташуванням (рис 1.3). Такий підхід забезпечує швидкий доступ до актуальної інформації, зручний пошук знаходження медичної установи, що відповідають потребам пацієнтам. Також це сприяє ефективному плануванню медичного обслуговування пацієнтів. Окрему увагу приділено оптимізації швидкості фільтрації, що гарантує миттєве відображення результатів навіть при високому навантаженні на сервер. У підсумку розроблена інформаційна система підвищує загальну якість надання медичних послуг.

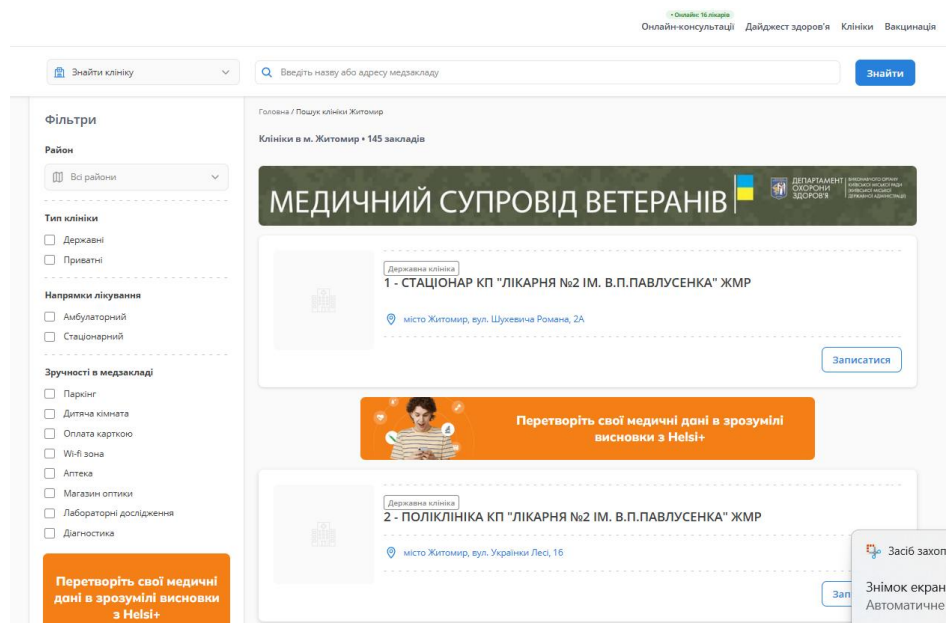


Рис. 1.3. Пошук медичного закладу

Аналогічний принцип реалізовано й для пошуку та фільтрації лікарів. Кожен лікар представлений у системі з інформацією про вартість прийому та відгуків пацієнтів. Таким чином, користувач має розширену змогу краще здійснити вибір спеціаліста (рис. 1.4).

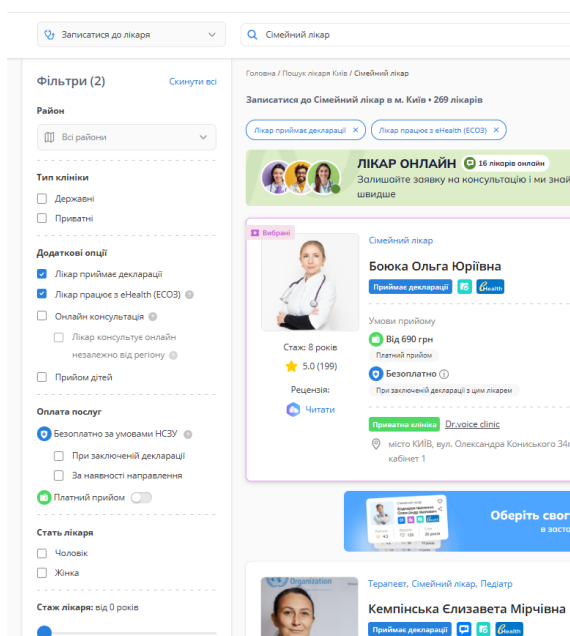
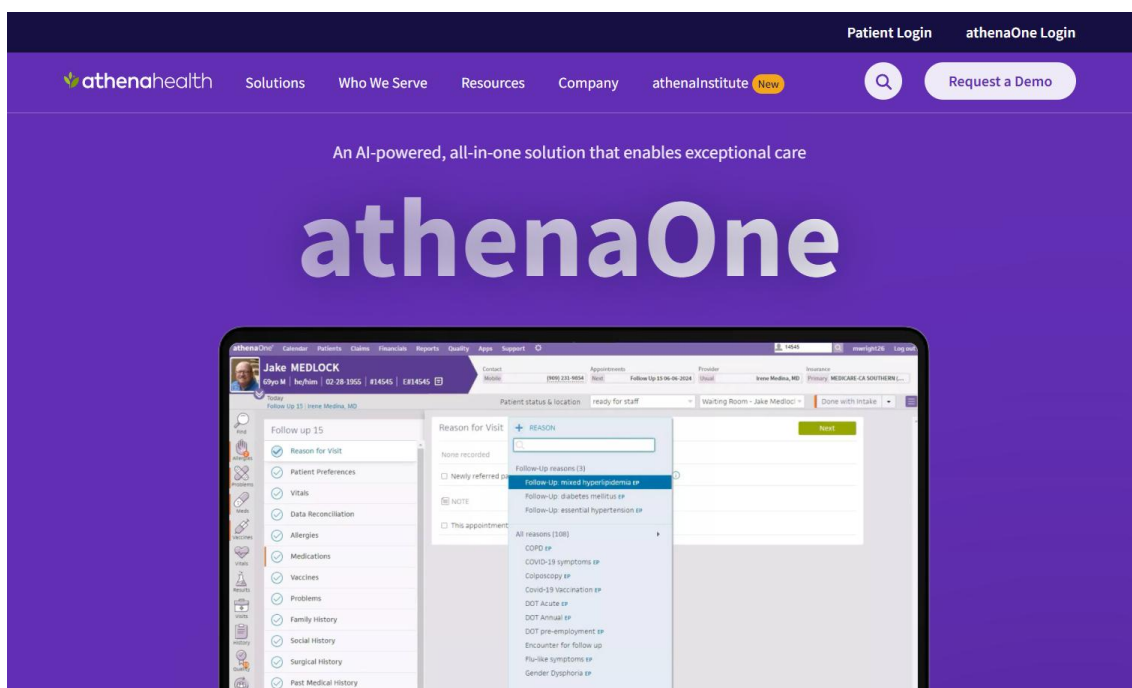


Рис. 1.4. Пошук лікаря

Аналогічні рішення використовуються і за кордоном, наприклад athenaOne (рис 1.5). Більшість таких систем забезпечують централізоване зберігання даних та автоматизацію клінічних процесів.



На головній сторінці, одразу видно, що платформа орієнтована на різні категорії користувачів, таких як пацієнти та медичні працівники. Інтерфейс передбачає окремий вхід для пацієнтів через персональний кабінет, що забезпечить контрольований доступ до власної медичної інформації.

Для того, щоб взаємодіяти надалі взаємодіяти з даною системою, реалізована форма, де користувачу пропонується обрати один із запропонованих варіантів, залежно від його ролі та цілей використання системи. Після заповнення форми з користувачем зв'язуються для подальшої роботи (рис. 1.6).

Інтерфейс має сучасний вигляд, виконаний у мінімалістичному стилі та орієнтований на швидку навігацію

Patient Login

athenaOne Login

athenahealth

Solutions

Who We Serve

Resources

Company

athenaInstitute

New

Request a Demo

Unlock productivity and drive efficiency

We will contact you within 2 business hours of form submission to schedule a brief call.

1

2

3

Which best describes you ?

(Choose One)

I'm a patient

I'm considering athenaOne for my organization

I'm a current athenaOne customer

Next

Рис. 1.6. Форма для взаємодії

Також, важливою перевагою порталу є система введення електронних медичних карт, що дозволяє лікарям швидко створювати, оновлювати медичні записи пацієнтів та іншу медичну інформацію в цифровому середовищі. Даний функціонал підвищує ефективність роботи медичного персоналу та спрощує доступ до важливої медичної інформації.

Особливо актуальним є те, що введення медичних карт інтегроване з AI, який забезпечує автоматизацію документації та оптимізує введення даних (рис. 1.7). Завдяки такому підходу зменшується кількість рутинних операцій та створюються умови для більш точного аналізу даних. Тому, завдяки використанню алгоритмів штучного інтелекту, впроваджено здатність виявляти приховані закономірності в анамнезі пацієнта та попереджати лікаря про потенційні ризики чи сумісність препаратів, що в свою чергу пришвидшить заповнення електронних медичних карток.

An AI-native EHR that simplifies care and documentation

Electronic Health Records

Deliver higher-quality, personalized care more efficiently with our AI-native EHR software. athenaOne® uses AI to enable better informed decision-making and improve care coordination while reducing your cognitive and administrative workload.

Request a demo

Our goal for AI-native

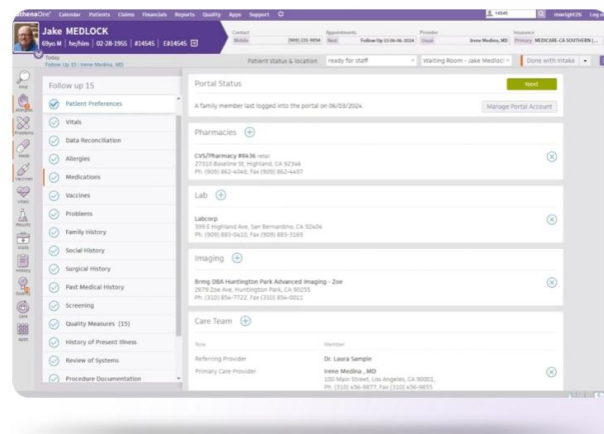


Рис. 1.7. Система введення електронних медичних карт

Отже, athenaOne демонструє сучасний підхід до розвитку медичних систем, поєднуючи електронні медичні записи та інший функціонал з інтелектуальними технологіями для підвищення зручності та продуктивності роботи медичного закладу.

Ще одним аналогом на українському ринку є ресурс Doc.ua (рис. 1.8), даний сервіс працює, як велика база лікарів, клінік та лабораторій і його основним завданням є максимально спростити для пацієнта пошук спеціаліста. Основними можливостями сервісу є пошук лікарів та клінік, справжні відгуки, які можуть залишати лише ті пацієнти, які дійсно були на прийомі та зручний запис з можливістю онлайн та офлайн-консультацій.

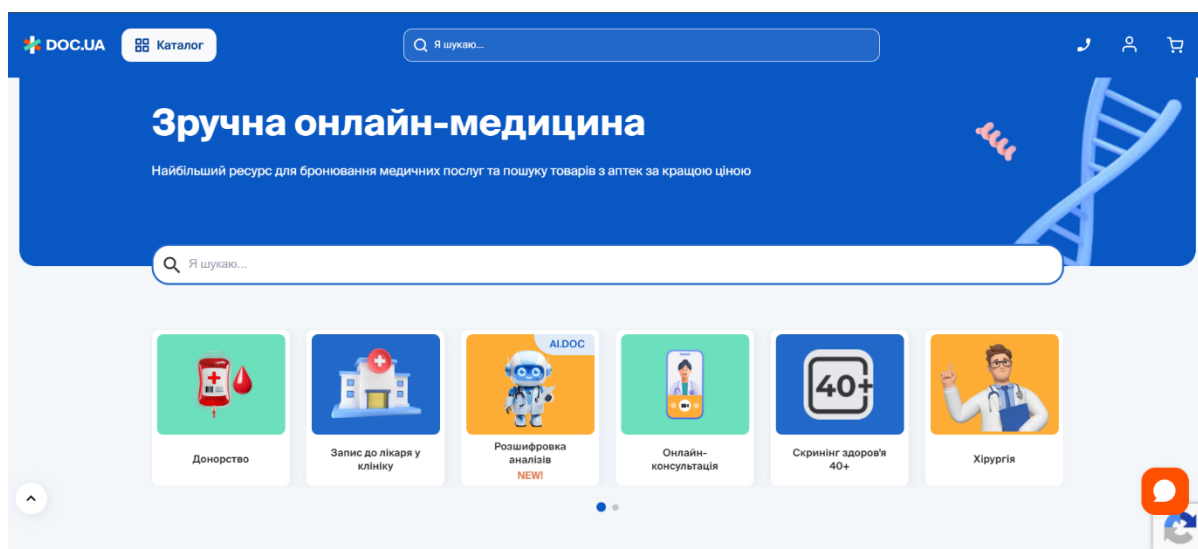


Рис. 1.8. Головна сторінка ресурсу Doc.ua

Таблиця порівняльного аналізу медичних систем

Критерії порівняння	Helsi	athenaOne	Doc.ua	Розроблюваний вебпортал
Зручний пошук та фільтрація спеціалістів	+	+	+	+
Онлайн-бронювання візитів	+	+	+	+
Повноцінний функціонал МІС	+	+	-	+
Наявність готової великої бази лікарів	+	+	+	-
Гнучкий контроль доступу до картки	-	-	-	+
Інтуїтивно зрозумілий, не перевантажений інтерфейс	+	+	+	+
Простий процес реєстрації	-	-	+	-
Інтеграція зі штучним інтелектом	-	+	-	-
Система верифікованих відгуків пацієнтів	+	-	+	-
Орієнтованість на український ринок	+	-	+	+

1.3. Вибір архітектури web-орієнтованої системи

Для реалізації вебпорталу приватної поліклініки було обрано модульну архітектуру [7], яка забезпечує логічне розділення програмного комплексу на окремі функціональні компоненти. Такий підхід дозволяє виокремити бізнес-логіку, обробку даних та інтерфейс, це сприяє підвищенню гнучкості системи та робить код більш читабельним та підтримуваним.

Для власної системи є важливий поетапний розвиток, це дозволить додавати нові функціональні компоненти без необхідності вносити суттєві зміни у загальну структуру системи. Таким чином, дана архітектура медичної системи спрощує оновлення та адаптації до змін вимог користувача.

Використання модульної архітектури також надає гнучкість системи та її масштабованість, що дозволяє додавати нові функції та компоненти без

порушення роботи вже існуючих, а також покращує тестованість завдяки можливості перевіряти кожен модуль незалежно від інших.

Також важливо зазначити, що модульна архітектура є одним із найбільш поширених підходів у розробці сучасних вебпорталів, вебсайтів та вебплатформ, тому що її основна ідея полягає в побудові програмного продукту як набору незалежних компонентів, кожен з яких відповідає за окрему частину функціональності. Така структура дозволяє уникнути випадків, коли всі елементи системи тісно пов'язані між собою, що досить ускладнює розвиток та підтримуваність програмного коду.

У сучасній практиці при побудові модульної архітектури виділяються окремі рівні: інтерфейс, бізнес-логіка, доступ до даних тощо. В свою чергу це забезпечує прозору організацію програмного коду, зменшує залежність між модулями та дозволяє вносити зміни або модернізувати окремі компоненти без впливу на всю систему загалом.

Таким чином, модульна архітектура є зручною для реалізації функціональних вимог та забезпечує технологічну основу для подальшого масштабування системи (табл. 1.2).

Таблиця 1.2

Відповідність архітектурних переваг модульного підходу

Вимога до вебпорталу	Проблема при монолітній архітектурі	Вирішення завдяки модульній архітектурі
Конфіденційність даних	Витік коду через інтерфейс дає повний доступ до сирих даних у БД	Через ізоляцію рівнів, шар інтерфейсу не має доступу та не знає про структуру БД
Гнучкість розкладу та записів	При зміні логіки валідації є потреба в переписуванні всього бекенду	Зміна логіки валідації оновлюється локально в одному сервісі
Масштабованість під навантаженням	Одночасний візит багатьох пацієнтів для перегляду медичних карт гальмує роботу всієї системи	Статичний фронтенд працює незалежно від обчислювального сервера

Надійність збереження медичної історії	Помилка при завантаженні довідки може пошкодити запис у картці пацієнта	Текстовий опис зберігається в СУБД, а файл йде в файлове сховище.
--	---	---

Зважаючи на переваги використання модульної архітектури, важливо зазначити, що вона є доцільною для систем медичного призначення, оскільки такі системи мають складну структуру даних, велику кількість інформаційних потоків та особливо підвищені вимоги до збереження конфіденційної інформації. Логічна структура розділення на окремі модулі дозволяє реалізувати механізми контролю доступу та забезпечити можливість інтеграції з іншими системами (табл. 1.3).

Таблиця 1.3

Структуризація серверної частини за архітектурними модулями

Назва директорії	Шар	Опис призначення
/routes	Рівень маршрутизації	Визначення HTTP-методів, кінцевих точок та підключення проміжного програмного забезпечення для перевірок.
/controllers	Рівень контролерів	Прийом запитів від маршрутів, первина валідація та виклики сервісів
/services	Рівень бізнес-логіки	Реалізація бізнес-перевірок, алгоритмів шифрування та формування відповідей
/models	Рівень моделей даних	Опис схем таблиць та налаштування зав'язків між ними
/config	Модуль конфігурації	Налаштування підключення до БД та до хмарного сховища

Кожен створений модуль вирішує лише одне завдання, такі як модуль маршрутизації реєструє доступні URL-адреси та передає обробку запитів відповідним контролерам. Контролери ізолюють логіку протоколу передачі даних від бізнес-процесів, тобто вони не знають, які саме операції виконуються для шифрування та бізнес-логіки загалом, а лише передають параметри на рівень сервісів.

Рівень бізнес-логіки дозволяє проводити необхідні бізнес-процеси медичного закладу, це допомагає зосередитися на реалізації комплексу дій, пов'язаних із координацією транзакційних процесів, де дії користувача може вимагати взаємодію кількох модулів та таблиць одночасно.

Важливою перевагою впровадження даної структури є можливість локалізації безпекових механізмів. У вебпорталі криптографічний модуль інтегрований безпосередньо в шар сервісів, завдяки цьому і реалізовано концепцію безпеки на стороні сервера. Тому, потоки даних, які містять чутливу інформацію (персональні дані, діагнози), потрапляють до модуля моделей вже у зашифрованому вигляді.

Оскільки серверна частина побудована на базі Node.js, взаємодія між модулями під час обробки важких операцій реалізована за допомогою асинхронних неблокуючих викликів. Це дозволяє серверу не блокувати головний потік виконання і продовжувати обробляти запити інших користувачів, які, наприклад, одночасно записуються на прийом

Отже, обрана архітектура відповідає сучасним підходам до реалізації web-орієнтованих систем та створює основу для подальшого створення власного вебпорталу, що сприятиме підвищенню стабільності роботи системи та якості програмного забезпечення загалом, де стабільність та безпека роботи забезпечується чітким логічним розмежуванням програмного комплексу на різних рівні модульні рівні, що дозволило повністю ізолювати клієнтську частину від структури БД. Тому, завдяки такій організації структури дозволило мінімізувати ризики витоку конфіденційної інформації пацієнтів.

1.4. Обґрунтування вибору інструментальних засобів та вимоги до апаратного забезпечення

Основною мовою розробки було обрано JavaScript [2], вибір даної мови визначено тим, що вона є універсальною, широко підтримується сучасними веб-технологіями та дозволяє використовуватися як на стороні клієнта, так і на стороні сервера.

Так як, система розділена на серверну та клієнтську частину, розглянемо реалізацію серверної частини системи. Для реалізації серверної частини використано середовище виконання Node.js [17]. Дане середовище забезпечує асинхронну модель без блокування інших завдань та обробки запитів, такий підхід є ефективним, тому що передбачає одночасну роботу великої кількості користувачів та активну взаємодію з базою даних. [6]

Однією з важливих та ключових особливостей Node.js є те, що забезпечує високу масштабованість серверної частини, оскільки дозволяє будувати застосунки, здатні працювати в умовах підвищеного навантаження. Для медичного порталу це є важливим, де одночасно можуть працювати користувачі різних ролей: лікарі, пацієнти та адміністратори [18].

Неможна не зазначити, що у Node.js є підтримка великої кількості бібліотек та модулів через npm, це дозволяє інтегрувати сучасні засоби авторизації, автентифікації, валідації даних, роботи з базами даних та багато інших засобів які оптимізують та полегшують розробку програмного продукту.

Щоб реалізувати саме побудову модульної структури на серверній частині, було вирішено, що доцільно використати фреймворк Express.js, який надає зручні інструменти для створення веб-додатків та API. Використання даного фреймворку забезпечує спрощення маршрутизації, обробку HTTP-запитів [24] та інтеграцію з іншими бібліотеками. Загалом, фреймворки сприяють підвищеності масштабованості та покращує тестованість системи, також вони прискорюють розробку через уже готові рішення для роботи з базами даних, аутентифікацією та іншими аспектами бекенда.

У процесі проектування важливо застосувати сучасні архітектурні підходи та патерни, зокрема модель MVC [8]. Завдяки такому підходу система може за необхідності адаптована до мікросервісної архітектури.

Важливо виокремити забезпечення безпеки у системі, таку функціональність виконуватиме бібліотека Passport.js, завдяки її гнучким механізмам авторизації та різних стратегій автентифікації, надасть системі ефективно контролювати доступ до функціональних можливостей.

Для створення динамічних користувацьких інтерфейсів використовується бібліотека React. Бібліотека базується на компонентному підході, що дозволяє розділити інтерфейс користувача на незалежні багаторазово використовуванні компоненти [22].

Використання React забезпечить високу продуктивність завдяки використанню віртуального DOM [30], оскільки замість повного перерендерингу сторінки, React оновлює лише ті елементи, які змінилися. Це також є особливо важливим для медичних вебпорталів, де інтерфейс може містити значну кількість записів та багато іншої медичної інформації.

React підтримує декларативний стиль програмування, що зменшує складність розробки, а бібліотека автоматично забезпечує коректне відображення. Це також підвищує надійність відображення даних, які отримуються з серверної частини через REST API.

Крім того, бібліотека легко інтегрується з сучасними підходами керування станом застосунку, такими як Redux, що дозволяє ефективно реалізовувати різні сценарії використання користувача з системою. Завдяки цьому користувачі різних ролей можуть отримувати персоналізований інтерфейс та доступ до відповідних функцій.

Використання React у клієнтській частині вебпорталу для медичних закладів є чудовим рішенням, оскільки забезпечує високу продуктивність та зручність в умовах подальшого розвитку вебпорталу.

Отже, обраний набір інструментальних засобів і технологій відповідає сучасним вимогам до розробки web-орієнтованих систем.

Для забезпечення стабільної та ефективної роботи вебпорталу для медичного закладу визначено вимоги до апаратного забезпечення (табл. 1.2). Оскільки система працює у режимі постійного доступу та обробляє конфіденційні медичні дані, апаратна інфраструктура повинна гарантувати високу продуктивність, надійність та можливість масштабування.

Таблиця 1.2

Технічні вимоги до системи та середовища виконання

Вимоги	Складова	Мінімальні технічні вимоги
Середовище виконання	Процесор (CPU)	Тактова частота не менше 2.5 ГГц
	Оперативна пам'ять (RAM)	Обсягом не менше 8 ГБ
	Дисковий простір	Наявність вільного місця не менше 50 ГБ
	Операційна система	На базі ОС або Linux Microsoft Windows
	Мережеве з'єднання	Стабільне підключення до мережі інтернет
СУБД та зберігання даних	Архітектура БД	Використання реляційної системи керування БД
	Надійність збереження	Підтримка механізму резервного копіювання
Мережа та доступність	Режим роботи	Рольова модель обмеження прав та захист від несанкціонованого доступу
	Навантаження	Одночасна підтримка роботи значної кількості клієнтів
Безпека та надійність	Контроль доступу	Рольова модель обмеження прав та захист від несанкціонованого доступу
	Захист облікових даних	Обов'язкове хешування паролів користувачів
	Захист даних	Шифрування медичних даних

Висновки до першого розділу

У першому пункті було детально проаналізовано та поставлено задачу, визначено способи та методи вирішення проблеми та описано основні функції та засоби реалізації медичної системи. На основі аналізу сучасних тенденцій у галузі захисту персональних даних було здійснено детальну постановку задачі. Визначено ключові функціональні вимоги до майбутнього вебпорталу, які полягають у забезпеченні автоматичної взаємодії між учасниками медичного процесу та наданні пацієнту повного доступу та управління ним за власною медичною інформацією.

У другому пункті знайдено відомі аналоги медичній системі, визначивши переваги, недоліки та загалом функціональність для врахування їх у створенні власної системи. У результаті аналізу переваг та недоліків виявлено, що більшість аналогів не мають функціоналу для введення медичних записів та не мають можливості надання пацієнту повного контролю над своїми даними.

У третьому пункті було визначено та обґрунтовано вибір архітектури для web-орієнтованої системи. Зокрема, обґрунтовано вибір модульну клієнт-серверної архітектури, що забезпечує логічне структурування системи. Зокрема, пріоритет було надано на реалізацію SPA на базі бібліотеки React для клієнтської частини. Для серверної частини визначено, що доцільніше використовувати Node.js та фреймворку Express.js для реалізації REST API. Отже, такий архітектурний підхід у поєднанні з реляційною СУБД PostgreSQL гарантує чіткий розподіл відповідальності між компонентами вебпорталу, забезпечуючи продуктивність асинхронної обробки запитів.

Таким чином, аналіз предметної області та аналогів дозволив сформулювати концепцію нового медичного порталу. Виявлені недоліки та переваги, підтвердили потребу у розробці власного рішення для поліклініки з підвищеним рівнем контролю користувачів над власними персональними даними. Також, спроектована структура вебпорталу вирішує проблему автоматизації взаємодії між лікарями та пацієнтами.

РОЗДІЛ 2. ПРОЄКТУВАННЯ ТА РОЗРОБКА ВЕБПОРТАЛУ ПРИВАТНОЇ ПОЛІКЛІНІКИ

2.1. Визначення варіантів використання та структури вебпорталу

Архітектура вебпорталу побудована на основі рольової моделі доступу з розмежуванням функціональних можливостей користувачів та контроль взаємодії між ними. Основними ролями в системі є адміністратор, лікар та пацієнт. Кожна з цих ролей виконує визначену логічну роль у процесі створення та управлінням медичної інформації.

Адміністратор у системі виступає як відповідальний за організаційну структуру вебпорталу. Його діяльність не пов'язана із взаємодією з медичними даними, а спрямована на створення та підтримку структури системи. У його можливості входить реєстрація користувачів, редагування департаментів, а також встановлення зв'язків між лікарями та відповідними департаментами. Отже, адміністратор забезпечує безпеку та цілісність організаційної моделі системи.

Лікар у вебпорталі є ключовий суб'єкт, який відповідальний за створення та повне управління медичними даними пацієнта. Його функціональні можливості охоплюють як управління доступом до медичної інформації, так і введення різноманітної медичної документації в межах взаємодії з пацієнтом. Для початку взаємодії із пацієнтом, лікар здійснює розгляд заявки на доступ до медичної історії пацієнта, та приймає рішення щодо підтвердження або відхилення заявки, що забезпечує контроль конфіденційності даних. Окрім визначених функціональних можливостей, лікар має змогу організовувати та управляти зустрічами та на основі них формує медичні записи та забезпечує документування медичних процесів шляхом завантаження відповідних файлів. Також, найважливішою складовою, є підтримка механізму доповнень до фіналізованих записів, що дозволяє вносити уточнення без порушення цілісності та незмінності медичних записів.

Пацієнт, у контексту вебпорталу виступає як ініціатор взаємодії з медичними працівниками та власником медичних даних. Його можливості

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						24

спрямовані на перегляд медичних даних та формування запитів на поширення доступ до своїх даних так і на отримання медичних послуг. Пацієнт має можливість перегляду свого профілю та його редагування, здійснення пошуку лікарів, управління доступом до своїх медичних даних та планування зустрічі з лікарем відповідно до його графіку роботи. Додатково передбачено можливість завантаження власних файлів у межах зустрічі, що сприяє доповненню та актуальності медичної інформації.



Рис. 2.1. Діаграма використання

Опис основних прецедентів наведено у таблицях 2.1 – 2.3. Наведені таблиці слугують для детального представлення функціональних можливостей порталу. Кожна специфікація визначає дійових осіб, варіант використання та короткий опис визначених дій для конкретного актора та варіанту використання.

Таблиця 2.1

Опис прецеденту актора «Адміністратор»

Назва варіанту використання	Основні актори	Короткий опис
Реєстрація користувачів	Адміністратор	Створення облікових записів користувачів (лікарів та пацієнтів)
Управління відділами	Адміністратор	Створення та оновлення існуючих відділів
Організаційний контроль лікарів	Адміністратор	Прив'язка лікарів до відповідних відділів, внесення даних про стаж, вартість прийому та налаштування робочого розкладу

Даний блок охоплює базові налаштування вебпорталу при якому адміністратор не має доступу до медичних даних пацієнтів, але виконує роль організаційного менеджера: створює облікові записи, формує структуру поліклініки та керує робочими профілями лікарів, створюючи розклад та додаючи ціни з стажем.

Таблиця 2.2

Опис прецеденту актора «Лікар»

Назва варіанту використання	Основні актори	Короткий опис
Обробка заявок	Лікар	Перегляд вхідних заявок щодо доступу до медичних карток пацієнтів
Управління зустрічами	Лікар	Створення, редагування та скасування прийомів
Введення медичних записів	Лікар	Створення записів на основі проведених зустрічей, фіналізація записів та додавання уточнень
Призначення лікування	Лікар	Додавання рецептів для пацієнта у межах медичного запису

продовж. табл. 2.2

Робота з медичними файлами	Лікар	Завантаження різних медичних вкладень та прикріплення їх до медичної картки пацієнта
Перегляд медичної історії	Лікар	Доступ до повної медичної історії пацієнта

У цій таблиці зосереджені основні медичні процеси та взаємодії. Функціонал лікаря покриває повний цикл з пацієнтом від отримання згоди на перегляд даних до введення медичної документації, що включає в себе записи, рецепти та додавання уточнень. Також у межах функціоналу лікаря доступні проведення консультацій з пацієнтом.

Таблиця 2.3

Опис прецеденту актора «Пацієнт»

Назва варіанту використання	Основні актори	Короткий опис
Пошук лікарів	Пацієнт	Пошук потрібних спеціалістів у вебпорталі за допомогою розширених фільтрів
Ініціювання зустрічі	Пацієнт	Вибір вільного слоту в розкладі лікаря та створення запиту на проведення зустрічі
Управління доступом	Пацієнт	Надсилання заявок на поширення доступу до власної медичної карти
Перегляд власної медичної карти	Пацієнт	Перегляд власної історії лікування, що включає медичні записи, рецепти, файли та доповнення
Управління власними файлами	Пацієнт	Перегляд файлів, а також самостійне завантаження файлів до зустрічей

Цей розділ описує можливості пацієнта вебпорталу щодо контролю за своїм здоров'ям та приватністю. Функціонал орієнтовано на пошук потрібних спеціалістів які доступні на вебпорталі, керування доступом та дозволами на перегляд та внесення даних до медичної історії, планування візитів та повний доступ до власних медичних вкладень та загалом всієї медичної інформації.

Таким чином, структура вебпорталу має чітке розмежування ролей, контроль доступу до медичних даних та організовану взаємодію між користувачами, що дозволяє ефективно підтримувати процеси надання медичних послуг та зберігати цілісність інформації задля забезпечення безпечного функціонування в умовах обробки конфіденційних даних.

Програмний комплекс спроектовано за допомогою класичної клієнт-серверної архітектури (рис. 2.2), що гарантує чіткий розподіл відповідальності між компонентами вебпорталу. Також, вибір клієнт-серверної архітектури забезпечує високу відмовостійкість, гнучкість масштабування та незалежність розробки окремих шарів.

Інформаційна взаємодія між клієнтським та серверним компонентом реалізується через стандартні протоколи передачі гіпертексту HTTP та розширення з підтримкою шифрування HTTPS і побудована за архітектурним стилем REST API, де кожен запит є самодостатнім і містить в собі логіку та інформацію необхідну для успішної обробки. Клієнтська ж частина в даному випадку виступає, як інтерактивне середовище з яким взаємодіють різні типи користувачів, такі як пацієнти, лікарі, адміністратори та загалом користувачі, які не є авторизованими на вебпорталі. А вся логіка обробки даних зосереджена на серверній частині, що включає в себе процеси автентифікації, авторизації, збереження інформації та реалізація шифрування.

Отже, такий підхід дозволяє знизити вимоги до апаратних ресурсів кінцевих пристроїв користувачів, оскільки всі обчислювальні операції виконуються на ресурсах серверної інфраструктури.

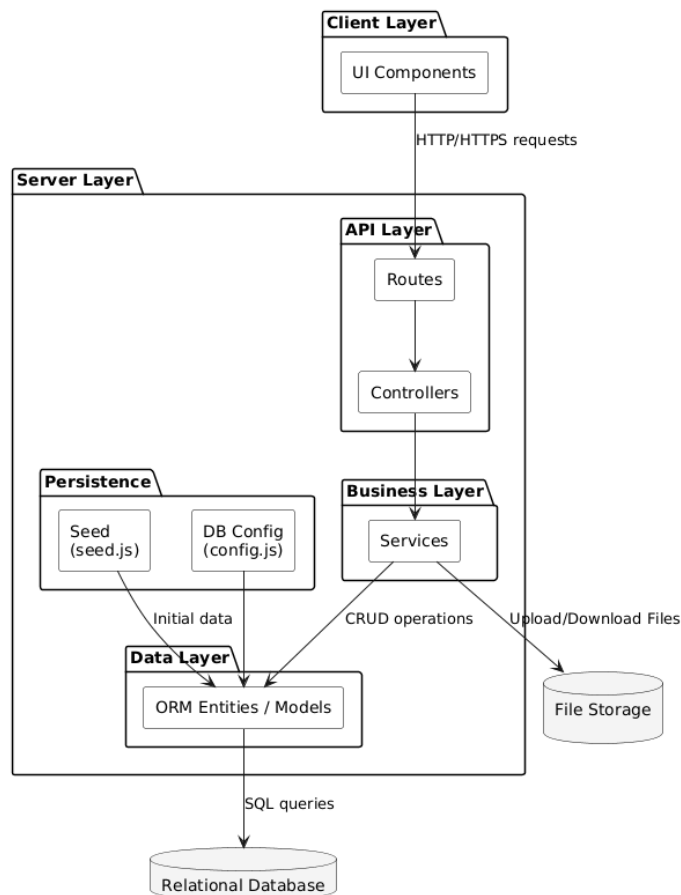


Рис. 2.2. Діаграма клієнт-серверної архітектури

Структуру даного архітектурного підходу можна розділити на 3 логічні рівні: рівень клієнта, рівень сервера та рівень зберігання даних.

Клієнтський рівень формується набором компонентів користувацького інтерфейсу, які відповідають за візуалізацію даних та забезпеченням взаємодії з кінцевими користувачами. Основною задачею цього рівня є генерація запитів до серверної частини з метою отримання та відображення відповідей у зручному графічному форматі.

Серверний рівень вебпорталу відповідає за всю основну роботу та обчислення порталу. Усі запити які приходять від клієнта спочатку потрапляють до API, який працює завдяки взаємодії маршрутизаторів та контролерів. Спочатку маршрутизатори виконують перевірку вхідних потоків інформації та направляють їх до відповідних контролерів за правильним маршрутом. Після того як дані прийшли до контролера, він виділяє із них необхідні параметри та передає їх до рівня бізнес-логіки. Рівень бізнес-логіки виконує найважливіші

процеси на сервері, такі як перевірка прав доступу, опрацювання зашифрованих даних, курування процесами завантаження та прикріплення файлів та інші специфічні завдання вебпорталу. Після того, як сервер успішно відпрацьовує усю логіку, сервіси формують і надсилають команди до рівня зберігання даних.

Завершальним структурним компонентом є рівень сховищ даних, щоб вебпортал працював швидко та стабільно, інформація зберігається роздільно. Усі чітко структуровані дані знаходяться у реляційній базі даних. Всі взаємодії з нею відбуваються за допомогою стандартних SQL-запитів.

Натомість великі та неструктуровані файли, наприклад результати аналізів, медичні документи або фотографії, зберігаються в окремому спеціалізованому файловому сховищі, а вже у самій БД залишаються посилання на ці файли та необхідна зашифрована інформація про них. Даний підхід дозволив не перевантажувати основну БД важкими документами, завдяки чому портал працює швидше, а доступ до всіх даних став значно зручнішим.

Отже, клієнт-серверна архітектура повністю відповідає вимогам розробки вебпорталу приватної поліклініки.

2.2. Розробка баз даних

Для повноцінної реалізації вебпорталу було розроблено реляційну модель бази даних, яка забезпечує зберігання, обробку та цілісність медичної та персональної інформації. Для реалізації баз даних було використано систему управління базами даних PostgreSQL, що забезпечує надійність, масштабованість, механізми транзакційності та підтримку складних зв'язків між сутностями. Для взаємодії порталу та бази даних було використано ORM-бібліотеку Sequelize, яка дозволяє працювати з даними у вигляді об'єктів та спрощує реалізацію різних операцій.

Розроблена структура баз даних відображає набір взаємопов'язаних сутностей з урахуванням принципів нормалізації. Тобто, це означає що сутності розділені таким чином, щоб мінімізувати дублювання даних, а зв'язки між ними реалізовані через зовнішні ключі. Центральними сутностями є користувач, лікар

					ІПЗ.КР.Б-121.26 - ПЗ	Арк.
						30

та пацієнт, які реалізують рольову модель порталу. У сутності користувача містяться лише ідентифікаційні дані, тоді як додаткова інформація винесена в окрему таблицю (`user_details`), що дозволяє уникнути надлишковості структури.

Значну роль у структурі та загалом функціонування порталу відіграє сутність зустрічі, яка виступає центральним елементом взаємодії між лікарем і пацієнтом. Вона містить інформацію про дату та формат проведення зустрічі, а також статус, що відображає його життєвий цикл. На основі проведених зустрічей формуються медичні записи, які містять діагноз та загальний клінічний висновок. Також, сутність медичних записів пов'язана із сутністю доповнень, який дозволяє вносити уточнення без змін основного запису.

Окремі ж сутності відповідають за процес лікування, такі як таблиця рецептів, яка містить інформацію про призначене лікування, а зв'язок із медикаментами реалізується через проміжну таблицю, що дозволяє враховувати множинні призначення та деталізувати параметри прийому препаратів. Такий підхід забезпечує гнучкість та можливість масштабування функціональності.

Для роботи з медичними документами передбачено сутність медичних файлів, яка забезпечує зберігання інформації про файли, а також проміжну таблицю, що реалізує контекст того, де ці файли зберігаються. Дана проміжна таблиця використовується для забезпечення повторного використання файлів у різних частинах вебпорталу без порушення цілісності та дублювання інформації.

Важливо зазначити, що реалізовано механізм контролю доступу до медичних даних через сутності `access_request` та `access_permission`, що забезпечує керований та контрольований доступ лікарів до медичної історії пацієнтів. Таким чином, даний підхід відповідає принципам конфіденційності та надає можливість пацієнтам контролювати, хто має доступ до його медичної інформації. Додатково передбачено також, журнал дій користувачів, який зберігає та фіксує лише ті дії, які пов'язані із зміною даних. Це дозволяє відстежувати критичні дії та можливістю реагувати на потенційні порушення або помилки.

Для того, щоб додатково забезпечити безпеку даних від несанкціонованих доступів, деяка чутлива інформація зберігається суто у зашифрованому вигляді з використанням окремих полів для вектора ініціалізації та тегу аутентифікації, що забезпечує додатковий рівень безпеки. Такий підхід мінімізує ризики несанкціонованого доступу та відповідає сучасним вимогам до захисту медичної інформації.

Детальний опис таблиць описано у таблицях 2.4 – 2.20.

Таблиця 2.4

Таблиця users

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
role	user_role	Роль користувача (admin, doctor, patient).
first_name	varchar(50)	Ім'я
last_name	varchar(50)	Прізвище.
email	varchar(50)	Електронна пошта
password	varchar(255)	Пароль
need_pass_reset	boolean	Прапорець, що вимагає скидання пароля
is_active	boolean	Статус активності облікового запису

Таблиця 2.5

Таблиця user_details

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
phone	varchar(50)	Номер телефону
address_encrypted	varchar(255)	Зашифрована адреса

продовж. табл. 2.5

address_iv	varchar(32)	Вектор ініціалізації для дешифрування
address_tag	varchar(32)	Тег автентифікації для зашифрованої адреси
date_of_birth	datetime	Дата народження
gender	user_gender	Стать користувача
user_id	int	Зв'язок із базовим записом користувача

Таблиця 2.6

Таблиця schedule

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
doctor_id	int	Зв'язок із лікарем
day	int	День тижня
start_time	time	Час початку роботи
end_time	time	Час завершення роботи
duration	int	Тривалість одного прийому (у хвиликах)

Таблиця 2.7

Таблиця register_request

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
first_name	varchar(50)	Ім'я
last_name	varchar(50)	Прізвище

продовж. табл. 2.7

email	varchar(50)	Електронна пошта
number	varchar(50)	Контактний номер телефону
created_at	timestampz	Час створення запиту

Таблиця 2.8

Таблиця doctor

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
user_id	int	Зв'язок із таблицею user
department_id	int	Зв'язок із відділенням
experience	varchar(50)	Досвід роботи (стаж)
avatar	varchar(255)	Посилання на фото профілю
specialization	varchar(255)	Спеціалізація лікаря
price	decimal(10,2)	Вартість консультації/прийому

Таблиця 2.9

Таблиця department

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	varchar(255)	Назва відділення
description	text	Детальний опис відділення
created_at	timestampz	Час створення запису

продовж. табл. 2.9

updated_at	timestamp_tz	Час останнього оновлення запису
------------	--------------	---------------------------------

Таблиця 2.10

Таблиця patient

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
user_id	int	Зв'язок із таблицею user

Таблиця 2.11

Таблиця medical_notes

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
title	varchar(255)	Заголовок медичного запису
diagnosis_encrypted	text	Зашифрований діагноз
diagnosis_iv	varchar(32)	Вектор ініціалізації для дешифрування
diagnosis_tag	varchar(32)	Тег автентифікації для діагнозу
notes_encrypted	text	Зашифрований текст медичної нотатки
notes_iv	varchar(32)	Вектор ініціалізації для дешифрування
notes_tag	varchar(32)	Тег автентифікації для нотатки
date	timestamp_tz	Дата створення/внесення нотатки

продовж. табл. 2.11

doctor_id	int	Лікар, що створив запис
patient_id	int	Пацієнт, якого стосується запис
appointment_id	int	Зв'язок із конкретним прийомом
created_at	timestamp_tz	Дата і час системного збереження
status	status_medical_note	Статус (DRAFT, FINAL)

Таблиця 2.12

Таблиця medical_amendments

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
medical_note_id	int	До якої нотатки внесено доповнення
doctor_id	int	Лікар, який вніс зміни
text_encrypted	text	Зашифрований текст уточнення
text_iv	varchar(32)	Вектор ініціалізації для дешифрування
text_tag	varchar(32)	Тег автентифікації для тексту уточнення
reason_encrypted	text	Зашифрована причина внесення змін
reason_iv	varchar(32)	Вектор ініціалізації для дешифрування
reason_tag	varchar(32)	Тег автентифікації для причини змін

продовж. табл. 2.12

title	varchar(50)	Заголовок для доповнення
created_at	timestampz	Дата та час внесення змін

Таблиця 2.13

Таблиця appointments

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
patient_id	Int	Пацієнт, записаний на прийом
doctor_id	int	Лікар, який проводить прийом
meeting_date	timestampz	Дата та час проведення прийому
created_at	timestampz	Дата та час створення запису
format_meeting	format_type	Формат зустрічі (ONLINE, OFFLINE)
url_meeting	varchar(350)	Посилання для онлайн-зустрічі
address_meeting	varchar(350)	Фізична адреса для офлайн-зустрічі
status	status_type	Статус прийому (Scheduled, Completed, Canceled, Pending, Waiting).
notes_encrypted	text	Зашифровані попередні скарги

продовж. табл. 2.13

notes_iv	varchar(32)	Вектор ініціалізації для дешифрування
notes_tag	varchar(32)	Тег автентифікації для коментарів

Таблиця 2.14

Таблиця prescriptions

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
date	timestampz	Дата виписування рецепта
description_encrypted	varchar(255)	Зашифрований загальний опис рецепта
description_iv	varchar(32)	Вектор ініціалізації для дешифрування
description_tag	varchar(32)	Тег автентифікації для опису рецепта
status	status_medical_note	Статус рецепта (DRAFT, FINAL, REVOKED).
doctor_id	int	Лікар, що виписав рецепт
patient_id	int	Пацієнт, якому призначено рецепт
medical_notes_id	int	Зв'язок із медичною нотаткою

Таблиця 2.15

Таблиця access_permission

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
doctor_id	int	Лікар, якому надано доступ
patient_id	int	Пацієнт, чії дані доступні
granted_at	timestampz	Час надання дозволу

Таблиця 2.16

Таблиця medication

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
name	varchar(100)	Назва препарату
form	varchar(100)	Форма випуску

Таблиця 2.17

Таблиця access_requests

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
to_doctor_id	int	Лікар, який робить запит на доступ
patient_id	int	Пацієнт, у якого запитують доступ до даних
status	statusname	Статус запиту

продовж. табл. 2.17

message	text	Повідомлення з обґрунтуванням запиту
created_at	timestampz	Час створення запиту
Responded_at	timestampz	Час відповіді на запит

Таблиця 2.18

Таблиця medical_files

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
file_name	varchar(100)	Оригінальна назва файлу
file_path	varchar(255)	Шлях до файлу
file_type	varchar(255)	MIME-тип або розширення файлу
file_iv	varchar(32)	Вектор ініціалізації для дешифрування
file_tag	varchar(32)	Тег автентифікації для вмісту файлу
description_encrypted	text	Зашифрований опис файлу
description_iv	varchar(32)	Вектор ініціалізації для дешифрування
description_tag	varchar(32)	Тег автентифікації для опису
uploaded_by	int	Користувач, який завантажив файл
uploaded_at	timestampz	Дата завантаження файлу

Таблиця 2.19

Таблиця medical_files_link

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
med_note_id	int	Прив'язка до медичної нотатки
file_id	int	Прив'язка до завантаженого файлу
amendment_id	int	Прив'язка до уточнення нотатки
appointment_id	int	Прив'язка до зустрічі
linked_by	int	Користувач, який прив'язав файл до запису
linked_at	timestampz	Дата створення прив'язки

Таблиця 2.20

Таблиця prescription_medication

Поле	Тип	Призначення
id	int	Унікальний ідентифікатор
prescription_id	int	Прив'язка до загального рецепта
medication_id	int	Прив'язка до медикаменту з довідника
dosage	varchar(100)	Дозування
duration_days	varchar(100)	Тривалість прийому
frequency	varchar(100)	Частота прийому

Отже, розроблена база даних забезпечує ефективну організацію зберігання та підтримує складні зв'язки між сутностями, тим самим гарантуючи цілісність та безпеку даних, а також створює надійну основу для функціонування вебпорталу в умовах обробки чутливої медичної інформації.

2.3. Реалізація функціоналу вебпорталу приватної поліклініки

У вебпорталу приватної поліклініки реалізовано надійний захист персональної інформації та завантажених файлів [27]. Основою механізму захисту виступає алгоритм симетричного шифрування Advanced Encryption Standard із довжиною ключа 256 біт, який функціонує в режимі лічильника з автентифікацією Галуа, відомому як AES-256-GCM. Даний алгоритм обрано тому що він одночасно здатен забезпечувати високу конфіденційність та цілісність інформації.

Для запуску процесу шифрування даних система використовує секретний ключ, який зберігається у файлі змінних середовища (.env) гарантуючи, що навіть у випадку витоку вихідного коду злоумисники не зможуть розшифрувати медичні дані користувачів.

Для шифрування текстової інформації реалізовано функцію шифрування (див. Додаток А), яка на першому етапі генерує унікальний псевдовипадковий вектор ініціалізації довжиною дванадцять байтів, його використання дозволяє багаторазове шифрування, яке формує абсолютно різний вихідний шифротекст для, наприклад, однакових діагнозів. На фінальному етапі обчислюється унікальний тег автентифікації, який функція повертає разом із зашифрованими даними та вектором для їх збереження у БД.

Функція механізму шифрування файлів працює за аналогічним принципом, однак замість рядкових перетворень працює з буферами пам'яті, шифруючи їх перед відправленням до зовнішнього сховища.

Зворотній процес дешифрування тексту чи файлів працює за принципом звіряння збереженого тегу автентифікації, тому, якщо дані у БД чи файлового

сховищі були змінені чи пошкоджені, процес дешифрування стає неможливим та доступ до даних блокується.

Також, важливим елементом забезпечення безпеки є функція перевірки прав доступу, яка регулює взаємодію між лікарями та медичною інформацією пацієнтів (див. Додаток Б). Перевірка відбувається через отримання вхідних даних обох учасників процесу (ідентифікатор лікаря та пацієнта) та здійснює пошук дозволу у відповідній таблиці зв'язків (`access_permission`). Якщо запис у відповідній таблиці відсутній то система забороняє доступ до виконання подальших дій та процес завершується з помилкою 403. Такий підхід є базовою перевіркою прав доступу та створює бар'єр проти несанкціонованого доступу до перегляду медичних даних та гарантує дотримання принципів лікарської таємниці.

Для забезпечення життєвого циклу зустрічі було вирішено створити фоновий процес для перевірки стану зустрічі. Це забезпечить логічному скасуванню зустрічей та запобігає засміченню бази даних зустрічами, які не відбулися, але все одно мають статус «Очікування». Даний фоновий процес (див. Додаток В), працює із заданою періодичністю та виконує перевірку таблиці зустрічей. Якщо зустріч має статус «Очікується», але з моменту її початку пройшла рівно доба, система автоматично змінює її статус на «Скасовано». Також це забезпечує підтримку актуальності прийомів без необхідності ручного втручання адміністратора або лікаря, тому що використання методів об'єктно-реляційного відображення дозволяє знаходити та масово оновлювати такі протерміновані зустрічі на рівні БД.

Основною частиною роботи вебпорталу є створення електронного медичного запису на основі проведеного прийому, який автоматично завершує зустріч або переводить її у стан «Очікується документація». Дану функцію спроектовано із багаторівневою валідацією даних та з забезпеченням транзакційності (див. Додаток Г).

На початковому етапі виконуються три перевірки на автентичність зв'язку лікаря з конкретним прийомом, валідацію поточного статусу зустрічі та

перевірка унікальності для запобігання дублюванню медичних записів для однієї зустрічі.

Безпека доступу забезпечується викликом сервісом (див. Додаток Б) для перевірки наявності активного дозволу на перегляд медичної історії пацієнта. Функція створення медичного запису працює таким чином, щоб при його додаванні, залежно від статусу запису «Завершений» або «Чернетка», вебпортал автоматично переводить сутність прийому у відповідний стан «Завершено» чи «Очікується документація». На рівні збереження конфіденційної інформації викликається функція шифрування даних та формує рядки діагнозу та опису у зашифровані бінарні рядки разом із генерацією векторів ініціалізації та тегом автентифікації, усі ж інші дані зберігаються у звичайному форматі та записуються у відповідну таблицю.

Висновки до другого розділу

У першому пункті було детально розглянуто та проаналізовано ключові аспекти проєктування медичного порталу, де основну увагу було приділено на формуванні функціональних можливостей порталу відповідно до ролей користувачів, а саме адміністратора, лікаря та пацієнта, а також на визначенні їх прав в межах порталу.

Також детально описано саме логіку взаємодії між користувачами, зокрема процес доступу, організація медичних зустрічей та формування медичних записів.

У другому пункті значна частина була присвячена проєктуванню реалізаційної баз даних, яка є основою порталу. Було розглянуто її структуру, сутності та зв'язки між ними. Описано використання систем управління базами даних PostgreSQL та ORM-бібліотеки Sequelize.

У третьому пункті детально розглянуто основна функціональна реалізація, яка відповідає забезпеченню безпеки, розподілу доступу до медичних даних та представлено функцію, яка відповідає за створення медичних записів на основі прийому, що є основою для роботи вебпорталу.

РОЗДІЛ 3. ПОРЯДОК РОБОТИ ТА ТЕСТУВАННЯ ВЕБПОРТАЛУ

3.1. Розгортання вебпорталу

Для опису архітектури та процесів розгортання була створена діаграма розгортання (див. Додаток Д), яка надає можливість продемонструвати, як саме програмні компоненти вебпорталу взаємодіють між собою та розподіляються між сервісами. Дана діаграма дозволяє зрозуміти загальну архітектуру системи на етапі її підготовки до використання у робочому середовищі.

Клієнтська частина архітектури ініціюється на пристроях кінцевих користувачів, де функціонує веббраузер, який завантажує зібрані файли клієнтського додатка.

Обробка всієї бізнес-логіки та різноманітних обчислювальних операціях відбувається на окремому бекенд-сервері. Дане середовище безперервно очікує запити від браузера та виступає головним координатором системи від перевірки прав доступу до обробки вхідних даних та передачею відповіді клієнту.

Для забезпечення надійності та безпеки, зберігання інформації ізольоване від обчислювальних процесів та розділено на два незалежні вузли. Уся текстова інформація знаходиться на окремому СУБД, з яким бекенд взаємодіє через закриті внутрішні мережеві протоколи. Окремо від БД інтегровано хмарне сховище, яке призначене включно для безпечного розміщення зашифрованих файлів та звичайних фото.

Логіка такої структури полягає у забезпеченні можливості масштабування та відмовостійкості, тому що, завдяки децентралізованому підходу кожен компонент інфраструктури виконує лише його роль у функціонуванні вебпорталу. Це дозволяє незалежно оновлювати клієнтський інтерфейс, збільшувати можливості бекенд-сервера при великій навантаженості користувачами та розширювати обсяг файлового сховища без необхідності повної зупинки вебпорталу приватної поліклініки.

Тому діаграма розгортання став важливим інструментом візуалізації фізичної архітектури вебпорталу.

Практична реалізація описаної архітектури розгортання базується на використанні сучасних хмарних платформ та безсерверних технологіях (табл. 3.1), що дозволяє автоматично масштабувати функціональні частини вебпорталу.

Таблиця 3.1

Специфікація хмарної інфраструктури та сервісів розгортання

Компонент	Хмарна платформа	Тип контенту/Технологічний стек	Архітектурна роль
Frontend Client	Vercel	Статичні файли (HTML5, CSS3, JS), зібрані через Vite	Налаштовано перенаправлення всіх запитів на index.html для коректної роботи.
Backend Server	Render	Node.js, Express.js	Обробка бізнес-логіки, валідація, шифрування
Database	Neon	Реляційна СУБД PostgreSQL	Ізольоване зберігання структурованих даних
File Storage	Cloudinary	Бінарні об'єкти, зображення, зашифровані медичні файли	Хмарне сховище, інтегроване через API. Реалізовано розділення на публічний медіа-контент та захищені медичні файли.

Клієнтська частина вебпорталу, представлена у вигляді статичних файлів HTML, CSS та JS-коду після збірки інструментом Vite та розгорнута на хмарній платформі Vercel. Даний сервіс гарантує мінімальні затримки при завантаженні інтерфейсу, оскільки статичні ресурси кешуються на найближчих до клієнтів проксі-серверах, додатково усі вхідні запити перенаправляються на кореневий файл index.html, забезпечуючи коректну роботу клієнтського роутингу React Router запобігаючи виникненню помилок 404 при перезавантаженні сторінок браузера.

Серверна частина, що реалізована на платформі Node.js з використанням фреймвору Express.js, ізольована у вигляді окремого веб-сервісу на платформі Render. Дана платформа має автоматичні тригери, які періодично опитують ендпоінти сервера запобігаючи переходу процесів у сплячий режим та гарантують миттєвий відгук на запити клієнтів. Обмін даними між хмарним сервером додатків та БД PostgreSQL здійснюється через шифроване SSL-з'єднання. Сама БД розгорнута на хмарній платформі Neon, що надає повну ізольованість таблиць від зовнішнього інтерфейсу.

Третім вузлом виступає хмарне сховище, де зберігаються різний медіа контент та зашифровані медичні файли. Цим сховищем є Cloudinary, який інтегрований у серверну частину вебпорталу через API. При завантаженні файлів у сховище, сервер приймає бінарний потік даних та перенаправляє їх у Cloudinary з розділенням контенту на медіа (фото користувачів) та захищені бінарні об'єкти, доступ до яких обмежено згенерованим тимчасовим токеном (рис. 3.1).

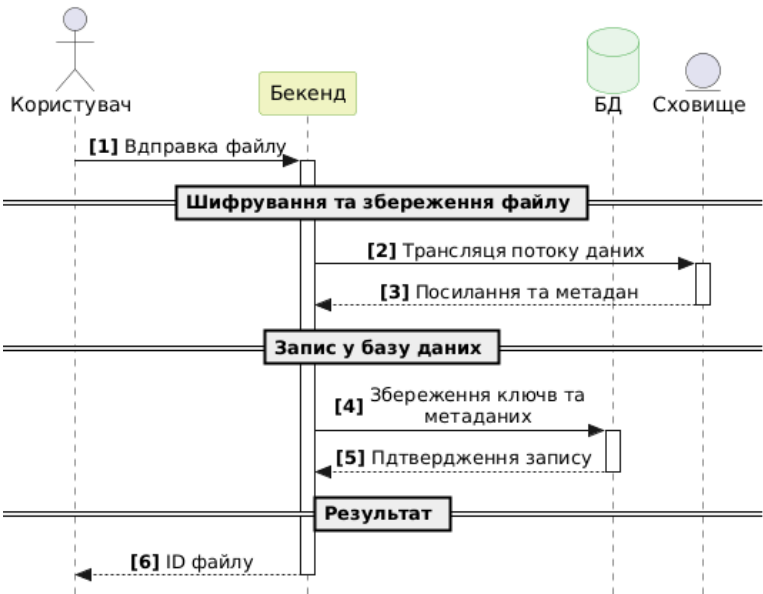


Рис. 3.1. Діаграма процесу завантаження файлу

Отже, реалізована інфраструктура розгортання вебпорталу приватної поліклініки повністю відповідає сучасним вимогам до архітектури хмарних застосунків. Поділ вебпорталу на ізольовані функціональні частини дозволив

досягти максимальної децентралізації обчислювальних ресурсів та гарантує високу доступність для кінцевих користувачів.

3.2. Структура інтерфейсу

Клієнтську частину вебпорталу було реалізовано у вигляді SPA, на базі бібліотеки React, що забезпечує високу швидкість відмальовування інтерфейсу без перенавантаження сторінок. Структура інтерфейсу базується на компонентному підході, що сприяє повторному використанню коду.

Візуальна частина порталу побудована з використанням бібліотеки компонентів MUI [15]. Завдяки чому інтерфейс має єдину типографіку, узгодженість кольорів та саму поведінку стилізованих компонентів. За допомогою даної стилізованої бібліотеки спрощує створення адаптивності вебпорталу, та взаємодія з користувачем доповнена модальними вікнами та сповіщеннями про успішність або помилки операцій.

Так, як вебпортал постійно робить запити на сервер потрібно забезпечити швидкість та реактивність інтерфейсу, для цього було використано бібліотеку TanStack Query [28]. Вона бере на себе керування серверним станом безпосередньо в компонентах React. Завдяки цьому реалізовано кешування даних та фонове оновлення.

Загалом логічна структура вебпорталу поділяється на чотири основні ізольовані зони: публічна сторінка, особистий кабінет лікаря, особистий кабінет пацієнта та особистий кабінет адміністратора.

Під час першого входу на вебпортал ми потрапляємо на публічну частину (рис. 3.2)

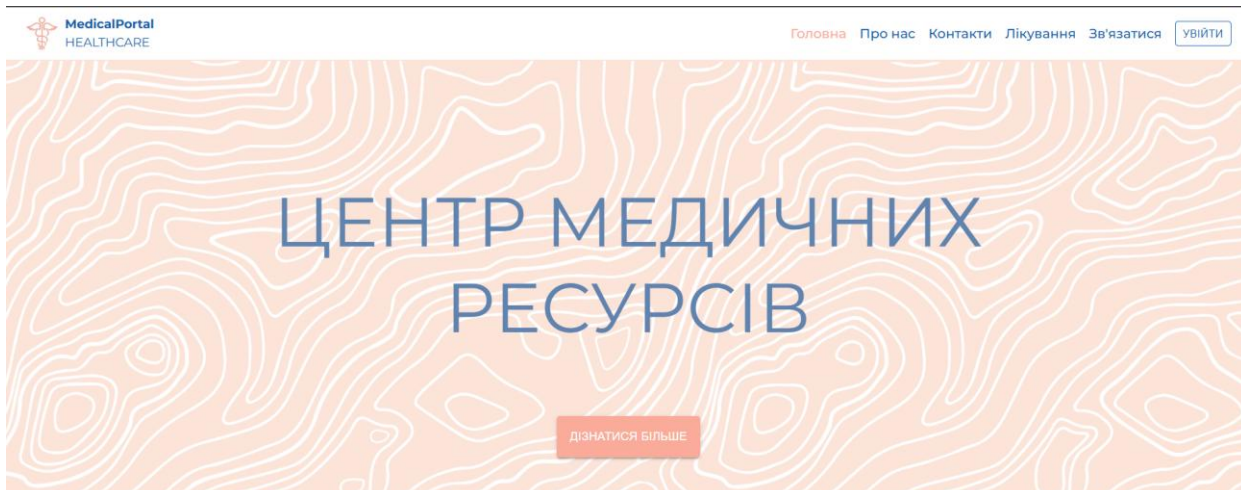


Рис. 3.2. Головна сторінка вебпорталу

На цій частині представлено основну інформацію про вебпортал та медичний заклад, можемо бачити зручну навігацію та кнопку яка перенаправляє нас на сторінку входу на вебпортал.

Так як реєстрація користувача відбувається на стороні адміністратора, було створено форму зворотнього зв'язку (рис. 3.3), щоб користувач надав початкові дані для зв'язку з ним та майбутнім створенням особистого кабінету на даному вебпорталі.

Рис. 3.3. Форма зворотнього зв'язку

Коли користувач надає свої персональні дані, для створення особистого кабінету і його було успішно внесено до порталу, він зможе зайти на свій

Особистий кабінет пацієнта має бокову панель з навігацією по основним структурним елементах вебпорталу, на головній сторінці відображені всі лікарі які зареєстровані з можливістю фільтрації по таких критеріях як: стаж, статъ, спеціалізація та загальний пошук по ПІБ. (рис. 3.4)



A screenshot of a web application showing a doctor's profile. The profile card for 'Андрій Петренко' (Andriy Petrenko) is visible, showing his status as 'Онлайн' (Online), a price of '450грн/прийом' (450 UAH/consultation), and a specialty of 'Психіатр' (Psychiatrist). A modal form is overlaid on the page, titled 'Вкажіть причину запису' (Specify the reason for the appointment). The form contains a text input field with the placeholder 'Введіть текст...' (Enter text...) and a blue 'Відправити' (Send) button. In the background, the left sidebar shows filters for 'Стать лікаря' (Doctor's gender) with options for 'Жіноча' (Female) and 'Чоловіча' (Male), and 'Стаж лікаря' (Doctor's experience) with a slider set to 16 years. The bottom of the sidebar shows 'Спеціальність' (Specialty) with a dropdown menu currently set to 'Не обрано' (Not selected).

					ИПЗ.КР.Б-121.26 - ПЗ	Арк.
						50

Після надсилання заявки, пацієнти можуть переглядати поточний стан своєї заявки, дату надсилання, дату відповіді та скасування якщо статус заявки знаходиться у стані «Очікується», схожа сторінка присутня і в особистому кабінеті лікаря, але з додатковим полем «Повідомлення», де знаходиться повідомлення від пацієнта (рис. 3.6).

№	Дата створення	Дата відповіді	Лікарі	Статус	Дії
8	15.05.2026 14:53:48	Немає відповіді	Матвій Загородний Невролог	Очікується	СКАСУВАТИ
7	11.04.2026 16:18:53	11.04.2026 16:20:09	Віктор Бондаренко Хірург	Завершено	СКАСУВАТИ
5	05.04.2026 14:22:06	05.04.2026 14:23:47	Олена Шевченко Стоматолог	Завершено	СКАСУВАТИ
1	22.03.2026 16:15:52	22.03.2026 17:27:16	Євген Гончарук Хірург	Завершено	СКАСУВАТИ

Рис. 3.6. Список поточних заявок

Найважливішим модулем у вебпорталі є сторінка з списком всіх медичних записів. Кожен запис містить коротку інформацію про нього: дата створення, заголовок, лікар, який створив запис та діагноз. Записи можна фільтрувати за пошуком та датою. Для лікаря створений схожий функціонал, але йому також відображаються свої записи у статусі чернетки та можливістю фільтрації за статусом (рис. 3.7).

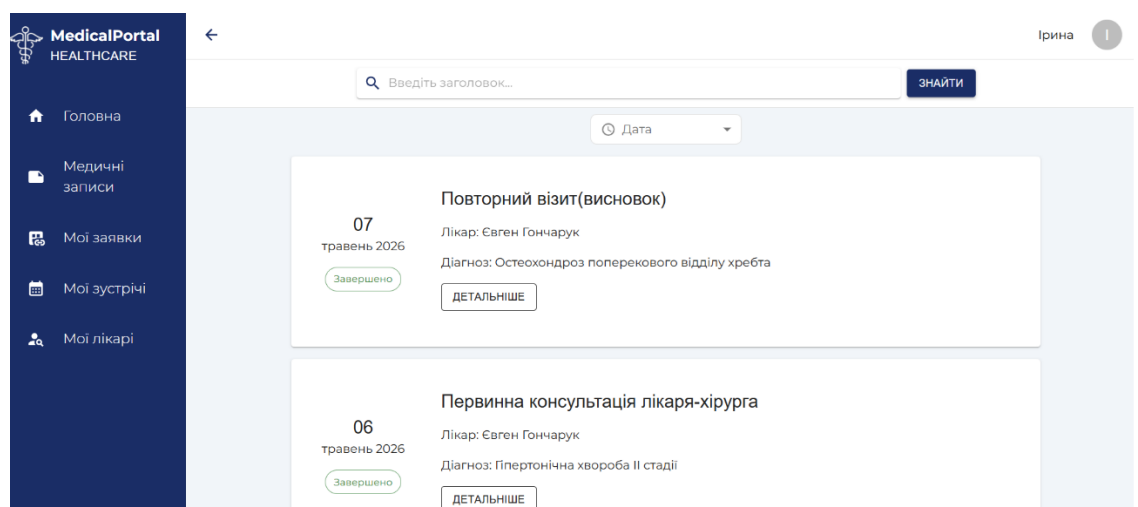


Рис. 3.7. Сторінка зі списком записів

Після натискання кнопки «Детальніше», що знаходиться на кожній картці запису, користувач переходить на додаткові деталі про запис, де на правій панелі можна побачити прив'язані доповнення, рецепти та файли (рис. 3.8).

Таке ж відображення доступне і для лікаря, але в його можливості входить ще додавати доповнення, рецепти та прив'язувати файли.

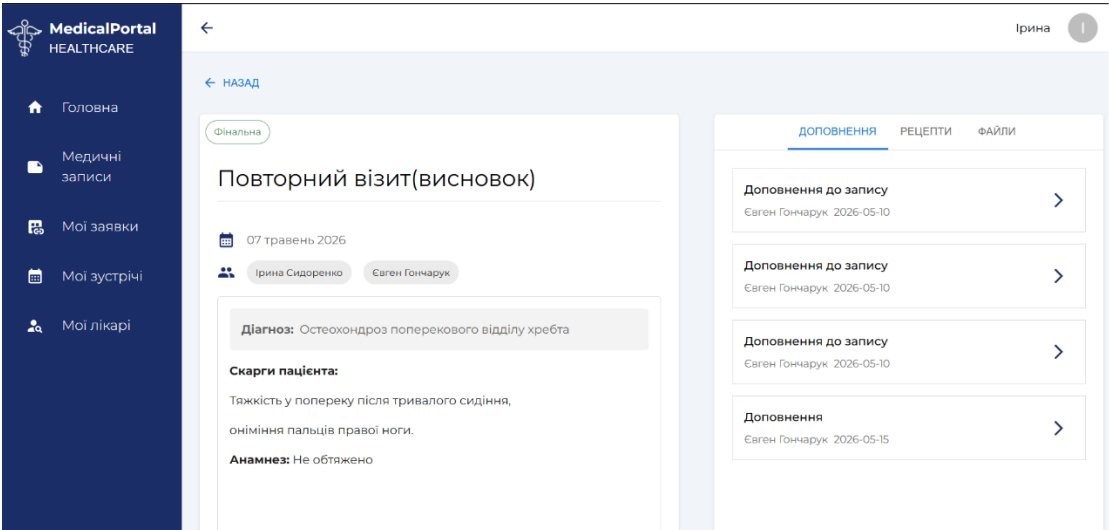


Рис. 3.8. Деталі про медичний запис

Як для пацієнта так і для лікаря, доступна сторінка з майбутніми прийомами так і загалом історія всіх прийомів. Кожен прийом представлений у вигляді карток з короткою інформацією про нього також із позначенням поточного статусу. Кожен прийом має свій життєвий цикл від створення до завершення, це допомагає отримати повну історію всіх прийомів та можливістю їх сортувати за статусом та датою (рис 3.9).

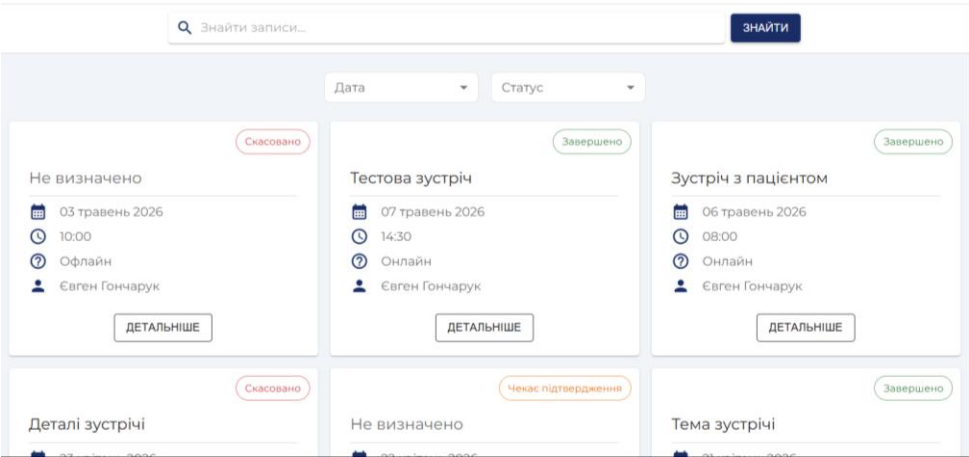


Рис. 3.9. Сторінка з історією прийомів

HEALTHCARE

Головна

Медичні записи

Мої заявки

Мої зустрічі

Мої лікарі

← НАЗАД

Завершено

Деталі

Тема: Тестова зустріч

07 травень 2026

14:30

Ірина Сидоренко

Євген Гончарук

Онлайн

<https://meet.google.com/ukx-sksf-dcy>

Тестова зустріч

Дії

СКАСУВАТИ

Прив'язані файли

Завантажити

Результати аналізів

PDF · 2026-05-07

Результати аналізів

PDF · 2026-05-07

Результати аналізів

PDF · 2026-05-07

Рис. 3.10. Деталі зустрічі

Для того, щоб отримати можливість створювати зустріч, лікар повинен підтвердити доступ до поширення прав доступу до медичної історії пацієнта, і тоді після підтвердження, пацієнт може бачити на окремій сторінці список лікарів, які мають до нього доступ (рис. 3.11).

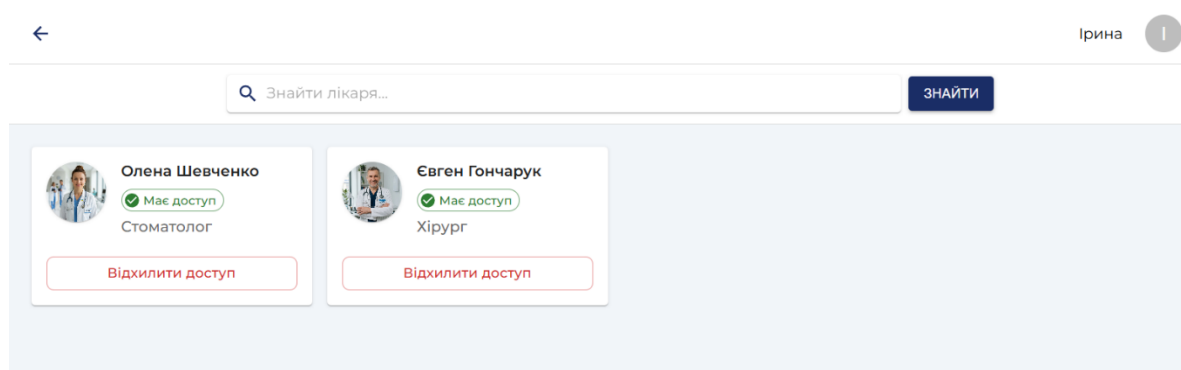


Рис. 3.11. Список підтверджених лікарів

До кожної картки з інформацією про лікаря є кнопка «Відхилити доступ», що означає те, що будь коли пацієнт може забрати права доступу лікаря на перегляд його медичної інформації та додавати зміни чи вкладення.

А якщо пацієнт просто натисне на картку лікаря, йому відкриється його розклад (рис. 3.12), де відображені дні та час його графіку з можливістю обрати вільний слот та заповнити форму для створення зустрічі.



Олена Шевченко



Має доступ

Стоматолог

Відхилити доступ



Євген Гончарук



Має доступ

Хірург

Відхилити доступ

Розклад лікаря

Євген Гончарук

+

ТРАВ. - ТРАВ.

18.05 - 22.05

+

ПН	ВТ	СР	ЧТ	ПТ
10:00	10:00	08:00	10:00	10:00
10:30	11:00	09:00	10:30	10:30
11:00	12:00	10:00	11:00	11:00
11:30	13:00	11:00	11:30	11:30
12:00	14:00	12:00	12:00	12:00
12:30	15:00	13:00	12:30	12:30
13:00	16:00	14:00	13:00	13:00
13:30			13:30	13:30
14:00			14:00	14:00
14:30			14:30	14:30

Рис. 3.12. Розклад лікаря

Також, пацієнт може переглядати свій профіль, в якому зазначено всю його основну інформацію, яку він може редагувати за потреби та панель з списком всіх виписаних рецептів від різних лікарів. Важливо зазначити, що пацієнт бачить як і дійсні рецепти так і анульовані, що забезпечує цілісність всієї історії лікування (рис. 3.13).

Ірина

Ірина Сидоренко

Основна інформація

Жіноча

Не вказано

Контакти

iriina@gmail.com

м.Житомир, вул. Гагаріна 22

Не вказано

Виписані рецепти

Від головного болю
Євген Гончарук 2026-05-15

>

Рецепт (інший)
Євген Гончарук 2026-05-15

>

Рецепт
Євген Гончарук 2026-05-15

>

Лікування спини
Євген Гончарук 2026-05-10

>

Рис. 3.13. Профіль пацієнта

Іншою ізольованою зоною у роботі вебпорталу є особистий кабінет лікаря (рис. 3.14). При вході лікар потрапляє на головну сторінку свого кабінету, де відображуються дві панелі із списком прийомів на сьогодні та список всіх медичних записів, які досі знаходяться на стадії «Чернетка». Це допомагає лікарю не пропустити прийом та акцентує увагу на завершенні записів.

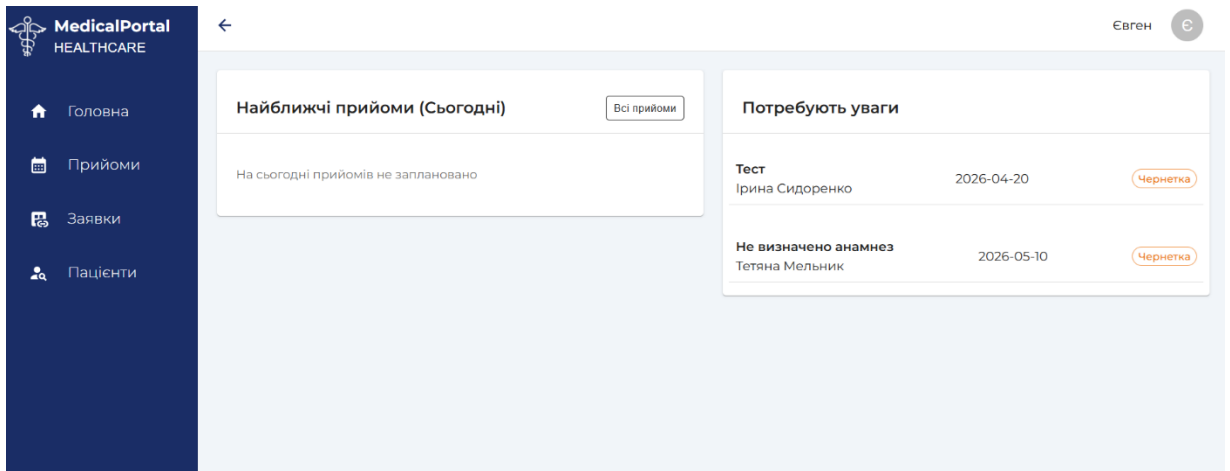


Рис. 3.14. Головна сторінка особистого кабінету лікаря

Єдиною відмінністю на сторінці деталей зустрічі є те, що лікар має змогу прикріпити вкладений файл до запису або доповнення та додати медичний запис. Якщо запис на стадії чернетки то файл прикріплюється до самого запису, а якщо ж запис фінальний то відкривається інший список зі всіма доповненнями до даного запису (рис. 3.15).

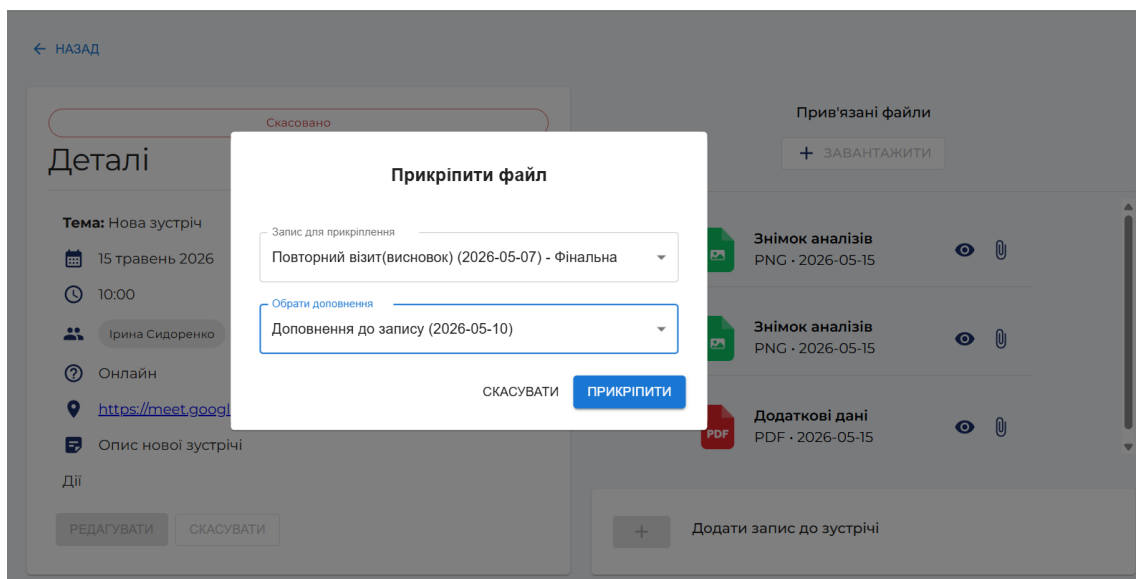


Рис. 3.15. Прикріплення файлу

Додавання медичного запису до прийому, означає логічне завершення зустрічі з пацієнтом. У формі додання медичного запису (рис. 3.16) відображено такі поля як: заголовок, діагноз та опис з можливістю додання мінімального форматування до тексту, що забезпечить кращу читабельність самого медичного запису.

Рис. 3.16. Форма додання медичного запису

Кожен лікар має сторінку зі списком пацієнтів, які він консультує та має доступ до медичної історії. Кожен пацієнт відображений у вигляді картки з короткою інформацією про нього (рис. 3.17).

Рис. 3.17. Список пацієнтів

При переході на детальну інформацію про пацієнта, лікарю відображається список всіх його медичних записів, сторінка відображення аналогічна до (рис.

3.6.). Також лікарю доступна можливість самому додати дату та час прийому для конкретного пацієнта (рис. 3.18).

←

← НАЗАД

травень 2026

За

Назва зустрічі

Введіть назву зустрічі

Опис зустрічі

Введіть опис зустрічі

Дата зустрічі

дд.мм.рррр

Пацієнт

Ірина Сидоренко

Формат проведення

Оберіть формат

Введіть адресу

Введіть місце проведення

Введіть час

--:--

ЗНАЙТИ

СТВОРИТИ

СКАСУВАТИ

Рис. 3.18. Форма додання зустрічі

У межах функціональності лікаря входить також створення рецептів, з вказуванням опису та детального опису препаратів доданих до рецепту. Кожен рецепт можна зберегти як фінальний варіант, який буде доступний і для пацієнта, так і як чернетку, яка буде доступна лише автору рецепту (рис. 3.19).

← НАЗАД

Фінальна

Повторення

РЕЦЕПТИ

СТВОРИТИ

Лікування спини
Ірина Гончарук 2026-05-10

Діагноз: Остеохондроз
Скарги пацієнта
Тяжкість у попереку
оніміння пальців
Анамнез: Неврастичний синдром

Опис рецепту

Кому
Ірина Сидоренко

Препарат №1

Ліки (назва)

Форма

Доза

Прийом

Частота

+ ДОДАТИ ЩЕ ПРЕПАРАТ

СКАСУВАТИ

ЧЕРНЕТКА

ВИПИСАТИ РЕЦЕПТ

Рис. 3.19. Форма створення рецепту

Як і у випадку з пацієнтом, у лікаря також є свій профіль (рис. 3.20) з можливістю його налаштування та перегляду робочого розкладу. Дані які може змінювати лікар є лише: дата народження, номер, адрес, стать та фото. Інша інформація змінюється лише адміністратором.



Євген Гончарук

Рік початку практики: 2001

Спеціальність: Хірург

ЗМІНИТИ

Основна інформація

 Чоловіча

 1979-04-05

 Хірургічне відділення

Контакти

 evgen@medical.com

 Не вказано

 0962435866

День тижня	Час початку	Час закінчення	Тривалість консультації
1	10:00:00	17:00:00	30
3	08:00:00	15:00:00	60
4	10:00:00	17:00:00	30
2	10:00:00	17:00:00	60
5	10:00:00	17:00:00	60

Рис. 3.20. Профіль лікаря

Останнім і найважливішим ізольованим середовищем на вебпорталі є адміністративна панель. На головній сторінці адміністративної панелі представлена у вигляді інформаційної панелі, що в режимі реального часу показує ключові метричні показники діяльності медичного закладу (рис. 3.21). Відображено такі дані як: сумарна кількість зареєстрованих користувачів, наявна кількість активних відділень та загальний обсяг запланованих на поточний день медичних консультацій. Додатково реалізовано аналітичний графік, який динамічно формується на основі групування всіх поточних статусів зустрічей, що допомагає візуально оцінювати ефективність проведення прийомів. Окремим показником є розрахунок середнього навантаження на медичний персонал. Для обчислення система виконує запит до БД та підраховує виключно унікальних пацієнтів, які зверталися до кожного конкретного лікаря, відкидаючи при цьому повторні візити. Після цього система сумує отриманні значення та вираховує середнє арифметичне число. Отриманий показник дозволяє керівництву закладу оперативно відстежувати ефективність роботи медичного закладу, що дозволяє

стратегічно планувати та оптимізовувати внутрішні процеси та ресурси поліклініки.

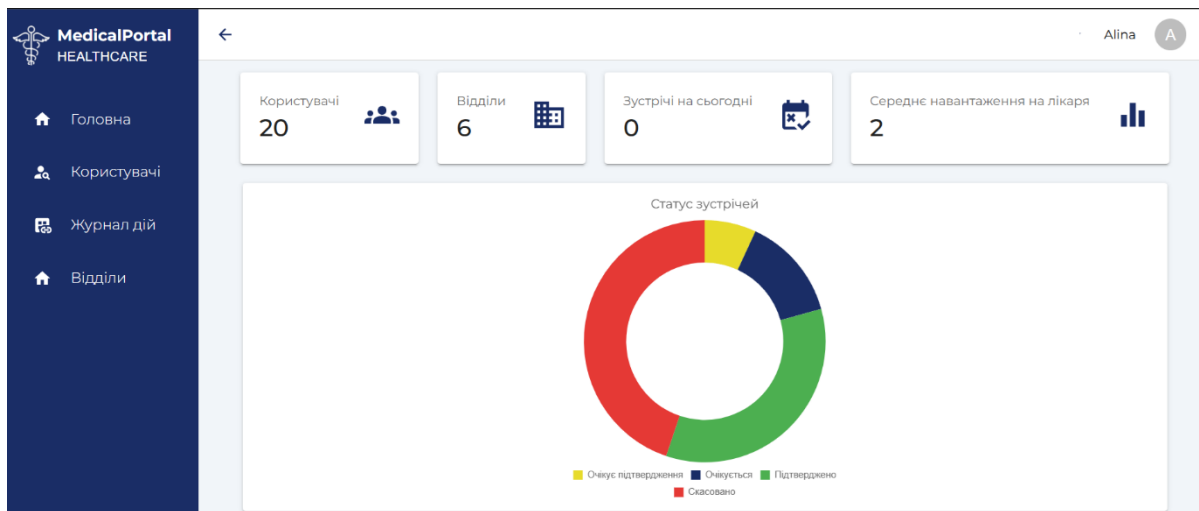


Рис. 3.21. Візуалізація метричних даних

Важливою частиною у роботі адміністратора є перегляд всіх користувачів у вебпорталі. Дана інформація відображена у вигляді таблиць з детальною інформацією про кожного користувача (рис. 3.22).

ДОДАТИ КОРИСТУВАЧА

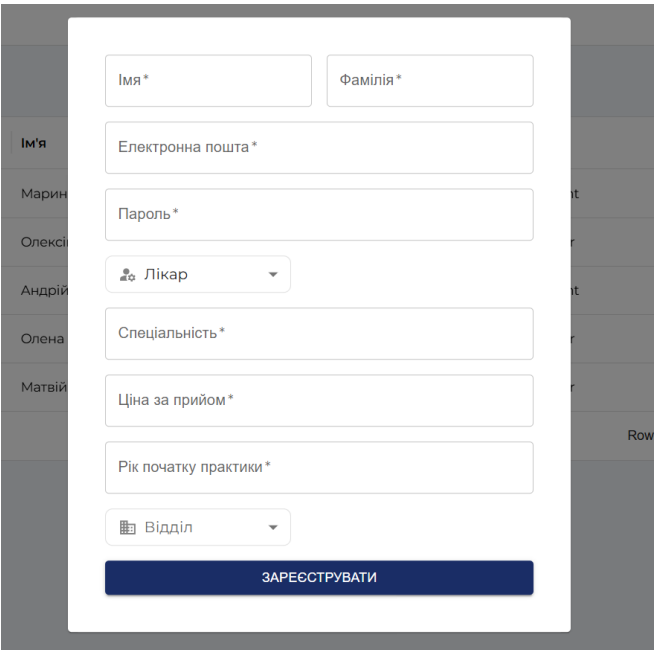
ID	Ім'я	Прізвище	Електронна пошта	Роль	Спеціальність	Дії
12	Марина	Ружна	marin1996@gmail.com	patient	—	 
2	Олексій	Іваненко	oleksiy@medical.com	doctor	—	 
8	Андрій	Петренко	andy@gmail.com	patient	Психіатр	 
6	Олена	Шевченко	olena@medical.com	doctor	Стоматолог	 
16	Матвій	Загородний	matvey@medical.com	doctor	Невролог	 

Rows per page: 5 1-5 of 20

Рис. 3.22. Сторінка перегляду користувачів

Функціональність адміністратора включає також реєстрація користувачів. У формі реєстрації адміністратору потрібно увести основну інформацію про користувача (електронна пошта, пароль, ім'я та прізвище), цієї інформації буде досить, якщо ми реєструємо пацієнта, але у випадку з лікарем додатково

відображуються такі поля як: стаж, спеціальність, відділ та ціна за прийом (рис. 3.23).



The registration form contains the following fields:

- Ім'я*
- Фамілія*
- Електронна пошта*
- Пароль*
- Лікар (dropdown menu)
- Спеціальність*
- Ціна за прийом*
- Рік початку практики*
- Відділ (dropdown menu)
- ЗАРЕЄСТРУВАТИ button

Рис. 3.23. Форма реєстрації користувачів

При перегляді додаткової інформації про лікаря, адміністратор може змінювати ціну за прийом та створювати або редагувати розклад роботи вибраного лікаря. Це забезпечує повний контроль у системі та дозволяє підтримувати актуальні дані для кожного лікаря, який зареєстрований у системі (рис. 3.24).



Євген Гончарук

Рік початку практики: 2001

Спеціальність: Хірург

ЗМІНИТИ

Основна інформація

Чоловіча

1979-04-05

Хірургічне відділення

Контакти

evgen@medical.com

Не вказано

0962435867

Вартість консультації

400.00 грн

РЕДАГУВАТИ

День тижня	Час початку	Час закінчення	Тривалість консультації
1	10:00:00	17:00:00	30
2	10:00:00	17:00:00	60
3	08:00:00	15:00:00	60
4	10:00:00	17:00:00	30
5	10:00:00	17:00:00	30

Рис. 3.24. Детальна інформація про лікаря

Окремою, і також важливою частиною у роботі адміністратора є перегляд структурних відділень медичного закладу. Дана інформація також представлена у вигляді таблиць з детальною інформацією про кожен відділ, включаючи кількість лікарів у кожному відділі. В обов'язки адміністратора також входить оновлення відділень та створення нових відділень (рис. 3.25).

ДОДАТИ ВІДДІЛЕННЯ					
ID	Назва	Кількість лікарів	Дата створення	Дата оновлення	Дії
6	Кардіологічне відділення	0	2026-05-14	—	 
1	Психіатричне (Психологічне) ...	2	2026-03-22	2026-05-06	 
3	Неврологічне відділення	3	2026-03-22	—	 
5	Педіатричне відділення	0	2026-04-29	—	 
2	Хірургічне відділення	2	2026-03-22	—	 
Rows per page: 5 ▾ 1–5 of 6 < >					

Рис. 3.25. Таблиця відділень

При перегляді деталей відділення, відображається опис самого відділення та список лікарів прив'язаних до обраного відділення (рис. 3.26).

Неврологічне відділення						
Відділення, що спеціалізується на лікуванні неврологічних захворювань						
				Кількість лікарів 3 Активних спеціалістів		
ID	Ім'я	Прізвище	Електронна пошта	Роль	Спеціальність	Дії
6	Олена	Шевченко	olena@medical.com	doctor	Стоматолог	 
16	Матвій	Загородний	matvey@medical.com	doctor	Невролог	 
17	Олег	Довгий	oleg@medical.com	doctor	Невролог	 
Rows per page: 5 ▾ 1–3 of 3 < >						

Рис. 3.26. Деталі відділення

Розділ інтерфейсу з списком надісланих заявок, спеціально спроектований для централізованої обробки та моніторингу, що надходять через комунікаційну форму зворотного зв'язку, розміщену у відкритій публічній зоні вебпорталу. Коли потенційний пацієнт або медичний спеціаліст ініціює контакт із закладом,

система автоматично структурує дану інформацію, створюючи відповідний запис у БД. Візуально ця сторінка реалізована у вигляді інтерактивної таблиці, яка відображає основні відомості про кожного потенційного клієнта (рис. 3.27).

Надіслані заявки					
ID	Ім'я	Прізвище	Електронна пошта	Номер	Дата створення
120	Наталія	Мельник	nataliiii@gmail.com	0964567654	2026-05-10
Rows per page: 10 ▾ 1–1 of 1 < >					

Рис. 3.27. Таблиця з даними про кожного заявника

3.3. Тестування роботи вебпорталу

Для перевірки працездатності вебпорталу приватної поліклініки було проведено етап комплексного мануального тестування. Вибір ручного підходу був зумовленим необхідністю детальної оцінки користувацького досвіду та специфікацією складної взаємодії різних категорій користувачів із інтерфейсом. Головною метою даного етапу є виявлення можливих логічних помилок, перевірка коректності обробки інформації та оцінка стабільності системи в умовах, які максимально наближені до сценаріїв реального використання вебпорталу.

Крім перевірки роботи основного функціоналу, було проведено тестування валідації форм та аналіз поведінки системи при спробах введення невалідних даних.

Початок тестування відбувається з першого входу на публічну частину вебпорталу, де дані вводить незареєстрований користувач. Метою перевірки було переконатися, що інтерфейс адекватно реагує на некоректні або неповні дані та надає користувачеві зрозумілі повідомлення на помилки, запобігаючи відправки на сервер хибних запитів. У цій формі перевірялася робота регулярних виразів для контактних даних, де спроба ввести цифри у поле «Прізвище», введення електронної пошти без символу «@» та довільний набір цифр у поле

«Номер телефону» було миттєво заблоковано з відображенням повідомлення про помилки (рис. 3.28).

Рис. 3.28. Валідація контактної форми

Проводилося тестування від ролі лікаря, при додаванні доповнення. Тестувалася валідація текстових полів на відповідність типу символів для запобігання створенню беззмисловних медичних записів. Тому при спробі ввести невалідні дані, поля відображають помилки та миттєво блокують запит до БД (рис. 3.29).

Рис. 3.29. Валідація форми створення доповнення

Перевірялася форма створення зустрічі при введенні дати та часу уже запланованого прийому. Тестування пройшло успішно, тому що вебпортал відловив це виключення та продемонстрував користувачеві глобальне спливаюче вікно червоного кольору про помилку створення такої зустрічі. Це гарантує, що користувач не зможе додати два прийоми на один і той самий час (рис. 3.30).

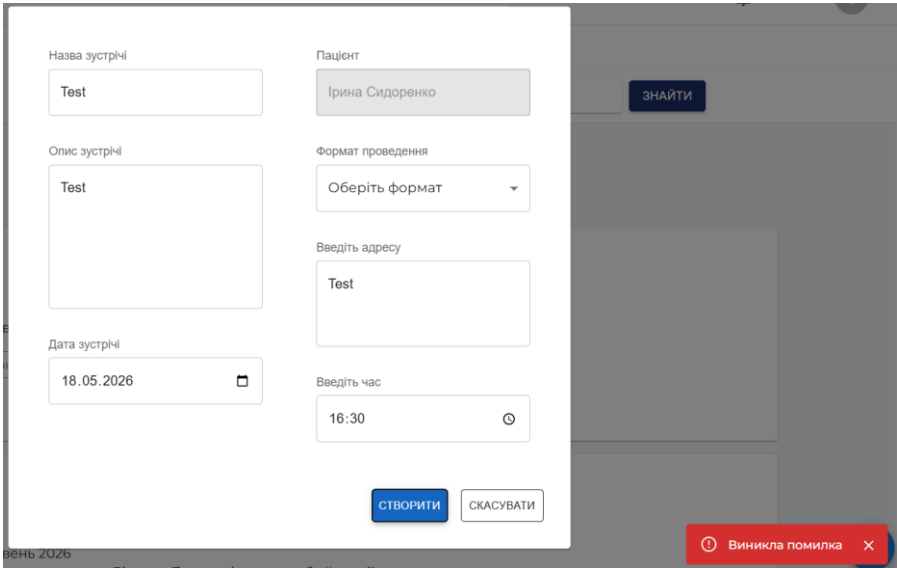


Рис. 3.30. Валідація форми створення зустрічі

Також, було проведено тестування форми створення зустрічі зі сторони пацієнта. В даній формі було введено час та дату у минулому. Вебпортал автоматично заблокував створити таку зустріч та відобразив сповіщення про неможливість створити зустріч на минулу дату (рис. 3.31).

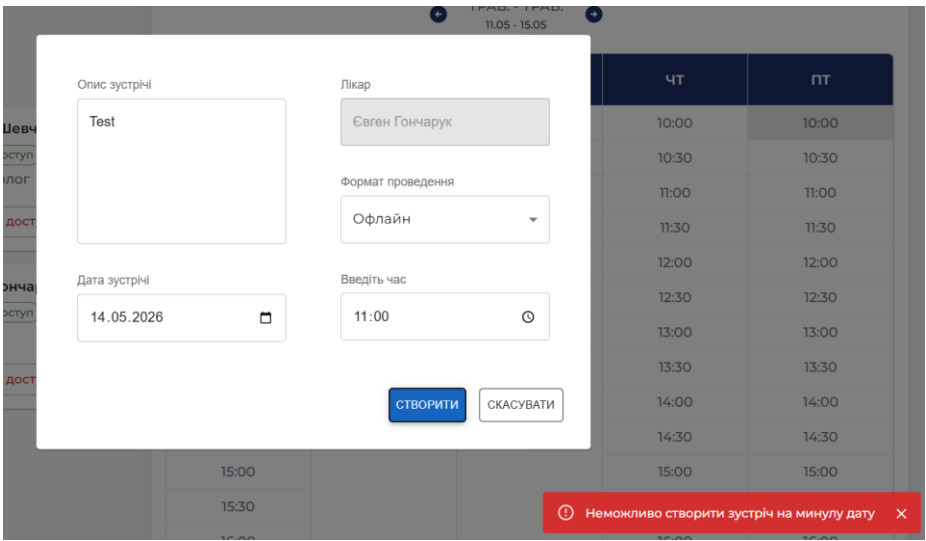


Рис. 3.31. Валідація форми створення зустрічі на стороні пацієнта

Проведено комплексне негативне тестування полів різних типів (текст, пароль, числа) під час створення нового профілю спеціаліста адміністратором. Валідація форми пройшла успішно, тому що при введенні некоректних даних у відповідні поля підсвічуються червоним з допоміжним текстом про помилку при цьому блокується відправка невалідних даних на сервер (рис. 3.32)

Ім'я*
test

Фамілія*
1
Поле має містити лише літери

Електронна пошта*
nataly@gmail.com

Пароль*
.....
Пароль повинен містити щонайменше 8 символів

Лікар

Спеціальність*
1
Поле має містити лише літери

Ціна за прийом*
test
Введіть коректний прайс

Рік початку практики*
1
Введіть коректний рік

Неврологічн...

ЗАРЕЄСТРУВАТИ

Рис. 3.32. Валідація форми реєстрації користувачів

Також на стороні адміністратора було проведено тестування форми створення та редагування робочого графіку лікаря. У межах тестування було перевірено поле введення для тривалості прийому. При спробі ввести час тривалості у менше ніж 10 хвилин, поле підсвічується червоним кольором та відображається допоміжна підказка про те, що потрібно ввести більше не менше ніж 10 хвилин тривалості (рис. 3.33). Таке обмеження на рівні інтерфейсу дозволяє уникнути хаотичного формування розкладу та забезпечує лікарю достатню кількість часу для якісного проведення огляду, консультації та заповнення медичної картки.

Рис. 3.33. Валідація форми створення та редагування графіку

Висновки до третього розділу

У третьому розділі було розглянуто роботу з інтерфейсом користувача та наведено опис і діаграму розгортання.

У першому підрозділі детально описано використану архітектуру для створення вебпорталу та наведено детальну діаграму розгортання.

У другому підрозділі було продемонстровано основні функціональності доступні на вебпорталі, зокрема розглянуто, як саме працюють ізольовані особисті кабінети різних ролей користувачів та життєві цикли медичних записів, зустрічей та рецептів. Візуалізація інтерфейсу надала змогу об'єктивно оцінити зручність користування вебпорталом для кінцевих користувачів.

У третьому підрозділі описано процес мануального тестування вебпорталу, перевірено роботу валідації полів та роботу логічних компонентів таких як: додавання запису на минулий час та додавання запису на одну і ту ж дату.

Отже, описана архітектурна модель та діаграма розгортання підтвердили раціональність вибору клієнт-серверної архітектури. Детальний аналіз кабінетів користувачів продемонстрував успішну реалізацію безпеки розмежування прав доступу.

ВИСНОВКИ

Під час виконання дипломної роботи було спроектовано та реалізовано захищений вебпортал приватної поліклініки з обґрунтуванням доцільності створення вебпорталу як єдиної точки входу для трьох видів користувачів таких як, пацієнтів, лікарів та адміністраторів. Комплексний підхід дозволив автоматизувати основні медичні процеси та забезпечив надійний рівень захисту конфіденційної інформації.

В основу реалізованого вебпорталу було закладено клієнт-серверну архітектуру, яка забезпечує масштабованість та чітке розділення бізнес-логіки. В основу побудови клієнт-серверної архітектури було покладено триланкову архітектуру, яка забезпечує декомпонування вебпорталу на три незалежні шари: шар представлення, серверний (бізнес-логіки) та шар БД. Перевагами такої архітектури стало те, що вона розділяє обов'язки всіх шарів, де кожна частина ізольована одна від одної, тому клієнтська частина виконує суто роль представлення без важких обчислюваних операцій, що підтримує продуктивність. Між клієнтом та сервером комунікація реалізована через HTTP-запити на основі архітектурного стилю REST, а сам сервер спроектовано без збереження стану, де він не зберігає інформацію в своїй оперативній пам'яті, тому ця відповідальність покладена на окремий шар, в шар – БД.

Клієнтську частину реалізовано за допомогою бібліотеки React та допоміжними засобами для дизайну та кешування даних. Для серверної частини було вирішено побудувати на базі середовища Node.js із використанням фреймворку Express.js. Для надійного збереження та структуризації інформації обрано реляційну БД PostgreSQL у поєднанні зі Sequelize ORM, що гарантує цілісність та транзакційність записів.

У процесі розробки створено функціонал, що автоматизує взаємодію між лікарями та пацієнтами. Практична частина охоплює безпечну реєстрацію та автентифікацію користувачів, управління робочим розкладом лікаря, механізм створення зустрічей з підтримкою обрання онлайн- та офлайн-форматів, а також

повноцінну електронну медичну карту з можливістю додання рецептів, доповнень та різних додаткових медичних вкладень.

Ключовим досягненням було впровадження захисту чутливих даних на основі симетричного алгоритму шифрування AES-256-GCM, який унеможливив витік клінічних даних. Даний криптографічний модуль забезпечує не лише дотримання конфіденційності, а й виконує критичну функцію безпеки, а саме – автентифікацію даних.

Проектування криптографічного модуля було реалізовано за допомогою вбудованого пакета crypto платформи Node.js. Тому, задля забезпечення конфіденційності, поля медичних записів, опису прийомів, файлів та самого вмісту файлів, які в подальшому зберігаються на захищеному хмарному сховищі Cloudinary є зашифрованими.

Отже, розроблений вебпортал приватної поліклініки ефективно оптимізує робочі процеси медичного закладу та забезпечує безпеку персональної інформації. Таким чином, результати кваліфікаційної роботи мають практичну цінність та гарантують суворе дотримання медичної таємниці та закладають потенціал до подальшого масштабування.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. About the Unified Modeling Language Specification Version 2.5.1. *Object Management Group*. URL: <https://www.omg.org/spec/UML/> (дата звернення: 10.04.2026)
2. An Introduction to JavaScript. *The Modern JavaScript Tutorial*. URL: <https://javascript.info/intro> (дата звернення: 10.04.2026)
3. Cherif Y. Understanding the Layered Architecture Pattern: A Comprehensive Guide. *DEV Community*. URL: <https://dev.to/yasmineddec94f4d4/understanding-the-layered-architecture-pattern-a-comprehensive-guide-1e2j> (дата звернення: 12.04.2026)
4. Cross-Origin Resource Sharing (CORS) - HTTP | MDN. *MDN Web Docs*. URL: <https://developer.mozilla.org/en-US/docs/Web/HTTP/Guides/CORS> (дата звернення: 12.04.2026)
5. Crypto | Node.js v26.1.0 Documentation. *Node.js – Run JavaScript Everywhere*. URL: <https://nodejs.org/docs/latest/api/crypto.html> (дата звернення: 15.04.2026)
6. Дос.уа - зручна онлайн-медицина. *Дос.уа*. URL: <https://doc.ua/> (дата звернення: 15.04.2026)
7. Fielding R. T. Architectural Styles and the Design of Network-based Software Architectures: Ph.D. dissertation. – University of California, Irvine, 2000. – 162 p.
8. GeeksforGeeks. MVC Design Pattern GeeksforGeeks. *GeeksforGeeks*. URL: <https://www.geeksforgeeks.org/system-design/mvc-design-pattern/#what-is-the-mvc-design-pattern> (дата звернення: 20.04.2026)
9. Get started in Postman | Postman Docs. *Postman Docs | Postman Docs*. URL: <https://learning.postman.com/docs/introduction/overview/> (дата звернення: 21.04.2026)
10. Kurose J. F., Ross K. W. Computer Networking: A Top-Down Approach (8th Edition). – Pearson, 2020. – 864 p.
11. Electronic Health Record-Related Safety Concerns: A Cross-Sectional Survey of Electronic Health Record Users. *JMIR Medical Informatics*. URL: <https://medinform.jmir.org/2016/2/e13/> (дата звернення: 15.04.2026)

12. Layered architecture. *slys.dev* | *Jakub Slys* | *Substack*. URL: <https://iam.slys.dev/p/layered-architecture> (дата звернення: 25.04.2026)
13. Медична інформаційна система Helsi. Офіційний портал. *helsi.me*. URL: <https://helsi.me/> (дата звернення: 20.04.2026)
14. Meneghello F., Calore E., Zucchetto D., Polese M., Zanella A. IoT: Internet of Threats? A Survey of Practical Security Vulnerabilities in Real IoT Devices // IEEE Internet of Things Journal. – 2019.
15. MUI: The React component library you always wanted. *MUI: The React component library you always wanted*. URL: <https://mui.com/> (дата звернення: 30.04.2026)
16. Nielsen Norman Group: UX Training, Consulting, & Research. *Nielsen Norman Group*. URL: <https://www.nngroup.com/> (дата звернення: 01.05.2026)
17. Node.js – Introduction to Node.js. *Node.js – Run JavaScript Everywhere*. URL: <https://nodejs.org/en/learn/getting-started/introduction-to-nodejs> (дата звернення: 01.05.2026)
18. Node.js – Overview of Blocking vs Non-Blocking. *Node.js – Run JavaScript Everywhere*. URL: <https://nodejs.org/en/learn/asynchronous-work/overview-of-blocking-vs-non-blocking> (дата звернення: 01.05.2026)
19. Passport.js. *Passport.js*. URL: <https://www.passportjs.org/> (дата звернення: 03.04.2026)
20. PlantUML Language Reference Guide. *PlantUML.com*. URL: <https://plantuml.com/guide> (дата звернення: 05.04.2026)
21. PostgreSQL: Documentation. *PostgreSQL: The world's most advanced open source database*. URL: <https://www.postgresql.org/docs/> (дата звернення: 10.05.2026)
22. React DOM Components – React. *React*. URL: <https://react.dev/reference/react-dom/components> (дата звернення: 10.05.2026)
23. React Router Official Documentation. *React Router Official Documentation*. URL: <https://reactrouter.com/> (дата звернення: 11.05.2026)

24. RFC 7230: Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing. *IETF Datatracker*. URL: <https://datatracker.ietf.org/doc/html/rfc7230> (дата звернення: 15.05.2026)
25. Richards M., Ford N. Fundamentals of Software Architecture: An Engineering Approach. – O'Reilly Media, 2020. – 432 p.
26. Sequelize. *Sequelize | Feature-rich ORM for modern TypeScript & JavaScript*. URL: <https://sequelize.org/> (дата звернення: 15.05.2026)
27. Stallings W. Cryptography and Network Security: Principles and Practice (8th Edition). – Pearson, 2020. – 768 p.
28. TanStack Query. *TanStack | The open-source application stack for the web*. URL: <https://tanstack.com/query/latest> (дата звернення: 01.06.2026)
29. Топ найкращих фреймворків Node.js. *FoxmindEd*. URL: <https://foxminded.ua/freimvorky-node-js/> (дата звернення: 30.05.2026)
30. Virtual DOM and Internals – React. *React – A JavaScript library for building user interfaces*. URL: <https://legacy.reactjs.org/docs/faq-internals.html> (дата звернення: 10.04.2026)

ДОДАТКИ

Реалізація функцій шифрування

```
import crypto from 'node:crypto'
import dotenv from 'dotenv'
dotenv.config()

const ALGORITHM = 'aes-256-gcm'
const KEY = Buffer.from(process.env.SECRET_ENCRYPT_KEY, 'hex')
const IV_LENGTH = 12;

export function encrypt(content) {
  if (!content) return null;

  const iv = crypto.randomBytes(IV_LENGTH);
  const cipher = crypto.createCipheriv(ALGORITHM, KEY, iv);
  let encrypted = cipher.update(content, 'utf8', 'hex');
  encrypted = cipher.final('hex') + encrypted

  const tag = cipher.getAuthTag()

  return {
    encrypted,
    iv: iv.toString('hex'),
    tag: tag.toString('hex')
  }
}

export function decrypt(encryption, iv, tag) {
  if(!encryption || !iv || !tag) return null

  const decipher = crypto.createDecipheriv(ALGORITHM, KEY, Buffer.from(iv,
'hex'))

  decipher.setAuthTag(Buffer.from(tag, 'hex'))
  let decrypted = decipher.update(encryption, 'hex', 'utf8')
  decrypted += decipher.final('utf8')
  return decrypted
}
```

```
export function encryptFileContent(fileContent) {
  const iv = crypto.randomBytes(IV_LENGTH);
  const cipher = crypto.createCipheriv(ALGORITHM, Buffer.from(KEY), iv)

  const encrypted = Buffer.concat([cipher.update(fileContent), cipher.final()])
  const tag = cipher.getAuthTag()

  return {
    encrypted,
    iv,
    tag
  }
}

export function decryptFileContent(fileContent, iv, tag) {
  if (!fileContent || !iv || !tag) return null

  const decipher = crypto.createDecipheriv(ALGORITHM, Buffer.from(KEY),
    Buffer.from(iv, 'hex'))

  decipher.setAuthTag(Buffer.from(tag, 'hex'))

  return Buffer.concat([decipher.update(fileContent), decipher.final()])
}
```

Реалізація функції перевірки прав доступу

```
import AccessPermission from "../Models/AccessPermission.js";
import {getDoctorId, getPatientId} from "../resolvers.js";
import {ForbiddenException} from "../ErrorHandling/CatchError.js";

export async function checkAccess(doctor, patient) {
  const doctor_id = await getDoctorId(doctor)
  const patient_id = await getPatientId(patient)

  const access_id = await AccessPermission.findOne({
    where:{
      doctor_id: doctor_id.id,
      patient_id: patient_id.id,
    }
  })

  if (!access_id) {
    throw new ForbiddenException('Forbidden')
  }

  return {doctor_id, patient_id};
}
```

Реалізація функції автоматичної перевірки зустрічей

```
import Appointments from "../../Models/Appointments.js";
import {Op} from "sequelize";

export function checkAppointmentsStatus() {
  console.log("checkAppointmentsStatus()");

  const interval = 60 * 60 * 1000
  const check = async () => {
    try{
      const deadline = new Date(Date.now() - 24 * 60 * 60 * 1000)
      const [updateAppointments] = await Appointments.update(
        {status: 'Canceled'},
        {
          where: {
            status: 'Scheduled',
            meeting_date: {[Op.lt]: deadline},
          }
        }
      )

      if(updateAppointments.length > 0) {
        console.log('updated appointments')
      } else {
        console.log('no updated appointments')
      }
    } catch (e) {
      console.error('Something went wrong...',e)
    }
  }

  check()
  setInterval(check, interval)
}
```

Функція створення медичного запису

```
export async function createNotes(userId, body) {
  const doctor = await getDoctorId(userId)

  const appointments = await Appointments.findOne({
    where: {
      id: body.appointmentId,
      doctor_id: doctor.id,
    }
  });

  if (!appointments) {
    throw new NotFoundException("No appointment found.");
  }

  if (appointments.status !== 'Scheduled') {
    throw new ConflictException("You cannot create note for this
appointment");
  }

  const existing = await MedicalNotes.findOne({
    where: {appointment_id: appointments.id},
  });

  if (existing) {
    throw new ConflictException("Medical note already exists");
  }

  const {doctor_id, patient_id} = await checkAccess(userId,
appointments.patient_id)

  const trans = await sequelize.transaction(async (trx) => {
    if (body.status === 'FINAL') {
      appointments.status = 'Completed'
      await appointments.save({transaction: trx})
    } else if (body.status === 'DRAFT') {
      appointments.status = 'Waiting'
      await appointments.save({transaction: trx})
    }
  })
}
```

```
const encNotes = encrypt(body.notes)
const encDiagnosis = encrypt(body.diagnosis)

const notes = await MedicalNotes.create({
  appointment_id: appointments.id,
  doctor_id: doctor_id.id,
  patient_id: patient_id.id,
  title: body.title ?? null,
  diagnosis_encrypted: encDiagnosis?.encrypted ?? null,
  diagnosis_iv: encDiagnosis?.iv,
  diagnosis_tag: encDiagnosis?.tag,
  notes_encrypted: encNotes?.encrypted ?? null,
  notes_iv: encNotes?.iv,
  notes_tag: encNotes?.tag,
  status: body.status,
  date: appointments.meeting_date,
  created_at: new Date().toISOString(),
}, {transaction: trx})

return notes
})
return trans
```

Діаграма розгортання вебпорталу приватної поліклініки

