Міністерство освіти і науки України
Харківський національний університет радіоелектроніки

Факультет комп'ютерних наук
(повна назва)

Кафедра програмної інженерії
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА
Пояснювальна записка

рівень вищої освіти перший (бакалаврський)

Програмна система для медичних закладів «LifeLine». Back-end
(тема)

Виконав:
здобувач четвертого року навчання
групи ПЗПІ-21-6

Анастасія ЧЕРНОВА
(Власне ім'я, ПРІЗВИЩЕ)

Спеціальність 121 – Інженерія програмного забезпечення
(код і повна назва спеціальності)

Тип програми освітньо-професійна

Освітня програма Програмна інженерія
(повна назва освітньої програми)

Керівник доц. кафедри ПІ Віктор КАУК
(посада, Власне ім'я, ПРІЗВИЩЕ)

Допускається до захисту
Зав. кафедри _____ Кирило СМЕЛЯКОВ
(підпис) (Власне ім'я, ПРІЗВИЩЕ)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерних наук
Кафедра програмної інженерії
Рівень вищої освіти перший (бакалаврський)
Спеціальність 121 – Інженерія програмного забезпечення
Тип програми Освітньо-професійна
Освітня програма Програмна Інженерія
(шифр і назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
«___» _____ 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Черновій Анастасії Максимівні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмна система для медичних закладів «LifeLine». Back-end

Затверджена наказом по університету від 19.05.2025р. № 397Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 16.06.2025

3. Вихідні дані до роботи Розробити програмну систему для медичних закладів «LifeLine» на базі JavaScript (Node.js) з використанням Express.js та PostgreSQL. Система має забезпечувати керування користувачами, прийомами, медичними записами та послугами, з урахуванням ролей доступу.

4. Перелік питань, що потрібно опрацювати в роботі

Вступ, аналіз предметної галузі, формування вимог до програмної системи, архітектура та проєктування програмного забезпечення, опис прийнятих програмних рішень, тестування розробленого програмного забезпечення, висновки додатки.

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Термін виконання етапів роботи	Примітка
1	Аналіз предметної галузі	10.04.2025	*виконано*
2	Створення специфікації ПЗ	12.04.2025	*виконано*
3	Проєктування ПЗ	16.04.2025	*виконано*
4	Розробка ПЗ	22.04.2025	*виконано*
5	Тестування ПЗ	17.05.2025	*виконано*
6	Оформлення пояснювальної записки	26.05.2025	*виконано*
7	Підготовка презентації та доповіді	02.06.2025	*виконано*
8	Попередній захист	13.06.2025	*виконано*
9	Нормоконтроль, рецензування	13.06.2025	*виконано*
10	Здача роботи у електронний архів	13.06.2025	*виконано*
11	Допуск до захисту у зав. кафедри	15.06.2025	*виконано*

Дата видачі завдання « 09 » « квітня » 2025р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. кафедри ПІ Віктор КАУК
(підпис) (посада, Власне ім’я, ПРІЗВИЩЕ)

РЕФЕРАТ

Пояснювальна записка до кваліфікаційної роботи бакалавра, 75 стор., 9 рис., 1 табл., 14 джерел, 5 додатків.

АДМІНІСТРАТОР, АНАЛІЗИ, АРХІТЕКТУРА, БАЗА ДАНИХ, ВІДГУКИ, ДІАГНОЗ, ЕЛЕКТРОННІ ЗАПИСИ, КЕРУВАННЯ ПРИЙОМАМИ, КОРИСТУВАЧ, ЛІКАР, МЕДИЧНІ ПОСЛУГИ, МЕДИЧНІ ЗАПИСИ, НАПРАВЛЕННЯ, ОБЛІК ПАЦІЄНТІВ, ПЛАНУВАННЯ, ПАЦІЄНТ, ПРИЗНАЧЕННЯ, ПРОГРАМНА СИСТЕМА, РЕЦЕПТИ, РОЛІ, СТАТИСТИКА, ФІНАНСОВА ЗВІТНІСТЬ, API, BACK-END, JAVASCRIPT, JSON WEB TOKEN, NODE JS, POSTGRESQL, REST API, SEQUELIZE.

Об’єкт розробки - програмна система для автоматизації управління медичними закладами «LifeLine».

Мета розробки - розробити серверну частину системи, що забезпечує зручне керування прийомами, облік пацієнтів, зберігання електронних медичних записів, статистику, фінансову звітність і взаємодію користувачів різних ролей (лікар, пацієнт, адміністратор).

Метод рішення - середовище розробки Visual Studio Code, платформа Node.js, мова програмування JavaScript, ORM Sequelize, архітектура REST API, формат передачі даних JSON, база даних PostgreSQL, авторизація через JSON Web Token (JWT).

У результаті розробки створено back-end частину системи «LifeLine», що автоматизує процеси лікарнь, керує записами, розкладами, діагнозами, призначеннями ліків, має можливість зворотного зв’язку пацієнтів та статистичних звітів.

ABSTRACT

ADMINISTRATOR, ANALYZES, ARCHITECTURE, DATABASE, FEEDBACK, DIAGNOSIS, ELECTRONIC RECORDS, APPOINTMENT MANAGEMENT, USER, DOCTOR, MEDICAL SERVICES, MEDICAL RECORDS, REFERRALS, PATIENT RECORDS, SCHEDULING, PATIENT, PRESCRIPTIONS, SOFTWARE SYSTEM, PRESCRIPTIONS, ROLES, STATISTICS, FINANCIAL STATEMENTS, API, BACK-END, JAVASCRIPT, JSON WEB TOKEN, NODE JS, POSTGRESQL, REST API, SEQUELIZE.

The object of development is a software system for automating the management of medical institutions called LifeLine.

Development goal - to develop the server part of the system that provides convenient appointment management, patient accounting, electronic medical records storage, statistics, financial reporting and user interaction of different roles (doctor, patient, administrator).

The solution method is Visual Studio Code development environment, Node.js platform, JavaScript programming language, Sequelize ORM, REST API architecture, JSON data transfer format, PostgreSQL database, authorization via JSON Web Token (JWT).

As a result of the development, the back-end part of the LifeLine system was created, which automates hospital processes, manages records, schedules, diagnoses, prescriptions, and has the ability to provide patient feedback and statistical reports.

ЗМІСТ

Перелік скорочень....8
Вступ....9
1.1 Загальний опис проблеми....11
1.2 Аналіз існуючих аналогів....13
1.2.1 Потреби клієнтів та ринку....16
1.2.2 Монетизація....17
1.3 Постановка задачі....18
1.3.1 Цільова аудиторія....19
2 Формування вимог до програмної системи....21
2.1 Окреслення концепції....21
2.2 Головна функціональність....22
2.3 Нефункціональні вимоги....23
2.4 Рамки первинного випуску....24
2.5 Рамки наступних випусків....25
2.6 Обмеження та винятки....26
2.7 Робоче середовище....27
3 Архітектура та проєктування програмного забезпечення....29
3.1 Архітектурні рішення та використані технології....29
3.2 Моделювання програмної системи....31
3.3 Проєктування моделі даних....34
3.4 Специфікація REST....37
4 Опис прийнятих програмних рішень....41
4.1 Опис обраних програмних рішень для серверної частин....41
4.2 Безпека, авторизація та управління доступом....43
4.3 Додаткові технічні рішення....46

5 Тестування програмного забезпечення....48
5.1 Модульне тестування (Unit testing)....48
5.2 Ручне тестування через Postman....49
5.3 Тестування безпеки....50
Висновки....52
Перелік джерел посилання....53
Додаток А....55
Додаток Б....56
Додаток В....63
Додаток Г....68
Додаток Д....75

ПЕРЕЛІК СКОРОЧЕНЬ

AES-256-GCM – Advanced Encryption Standard with 256-bit key

API – Application Programming Interface

bcrypt – Blowfish-based crypt

CSS – Cascading Style Sheets

HTML – HyperText Markup Language

HTTP – HyperText Transfer Protocol

IT – Information Technology

JSON – JavaScript Object Notation

JWT – JSON Web Token

OAuth2 – Open Authorization 2.0

ORM – Object-Relational Mapping

PDF – Portable Document Format

SaaS – Software as a Service

SQL – Structured Query Language

ВСТУП

Сфера медицини зараз стрімко розвивається [1]. Більшість лікарень переходять на використання цифрових рішень, бо старі підходи вже не працюють. Запис пацієнтів, зберігання даних, результати аналізів, комунікація між лікарями та пацієнтами повинне бути швидким, зручним і безпечним [2].

На практиці стикаємось з численними труднощами. Деяку частину документації ведуть вручну, багато систем застарілі і дані не захищені як слід [3]. Усе це створює ризики і для пацієнтів, і для лікарів.

Темою кваліфікаційної роботи є back-end розробка програмної системи для медичних закладів під назвою «LifeLine». Мета даної програмної системи – забезпечення ефективного управління медичними прийомами, аналізами та відгуками та спрощення комунікації між пацієнтами, лікарями та адміністрацією лікарні.

Система «LifeLine» вирішуватиме перелічені вище проблеми. У межах проєкту планується створити продуктивну серверну частину, що забезпечить ефективну організацію та обробку даних пацієнтів, надасть захищений доступ до персональної інформації пацієнтів, підтримуватиме ведення електронних медичних записів, адміністрування прийомів, збереження результатів діагностичних процедур, а також реалізовуватиме інтеграцію з клієнтськими застосунками.

Впровадження цієї системи дозволить медичним установам оптимізувати внутрішні процеси, підвищити якість надання медичних послуг і посилити безпеку медичних даних. Незважаючи на існуюче розмаїття програмних продуктів на ринку [4], більшість із них не відповідає вимогам сучасності - вони або мають обмежену функціональність, або не гарантують належного рівня безпеки, або ж не адаптовані до специфіки українського медичного середовища.

Технології розвиваються у напрямку веб-архітектур на базі мікросервісів. Вони гнучкі, надійні й легко адаптуються [5]. Практика показує, що сучасна медична система знижує кількість помилок і покращує обслуговування.

Цей проєкт створює безпечну та продуктивну серверну архітектуру. «LifeLine» - основа для повноцінного медичного комплексу, який відповідає потребам українських користувачів.

Мета - створити серверний застосунок із REST API для керування розкладами, послугами, аналізами й генерації PDF-звітів. Система підтримує ролі користувачів і захищає доступ до даних. Використано Node.js, Express.js, Sequelize, PostgreSQL. Для безпеки - шифрування паролів, захист даних і авторизація через JWT. Розробка виконувалась у Visual Studio Code, а тестування в Postman.

1 АНАЛІЗ ПРЕДМЕТНОЇ ГАЛУЗІ

1.1 Загальний опис проблеми

Лікарні щодня мають справу з обробкою великих обсягів дуже важливої інформації. Серед основних можно виділити: персональні дані пацієнтів, результати лабораторних досліджень, історії хвороб, призначення та розклади прийому. Все це потрібно швидко знайти та оброботи, але у більшості випадків ця інформація зберігається у вигляді паперових носіїв або у старих інформаційних системах, які не взаємодіють одна з одною [6]. Виходить, що лікар шукає потрібну медичну картку пацієнта, не знаходить її, через що втрачає час та росте ризик помилки.

Програмні продукти з якими працює більшість лікарень застарілі, вони повільно працюють, мають затримки сервера, проблеми з захистом даних та їх можливості доволі обмежені[7]. Це призводить до зниження ефективності роботи персоналу, затримок у прийомі пацієнтів, складнощів у веденні електронної медичної документації та труднощів з дотриманням стандартів конфіденційності.

У період останніх років можемо помітити стрімке збільшення кількості пацієнтів та зростаючих вимог до швидкості обробки інформації. Особливо актуальними стають питання безпеки персональних даних. Зокрема, в Україні набирає обертів впровадження системи eHealth, що вимагає нових системних рішень, здатних забезпечити стабільну та безпечну роботу з медичними даними [8].

Якщо згадати 2020 рік, тоді викликом стала пандемія коронавірусу, яка виявила системні слабкі місця в організації обліку пацієнтів, оперативного прийняття рішень та передачі медичних даних. У період карантинних обмежень, коли важливо було мінімізувати особисті контакти між людьми та перевести якомога більше процесів в онлайн, виявилася критична потреба у надійній цифровій інфраструктурі. Багато лікарень не мали ніякого стабільного програмного забезпечення для ведення електронних карток, віддаленого консультування чи обробки заявок, що призвело до перевантаження персоналу та втрати оперативності в наданні допомоги.

Такий досвід показав як важливо цифровізувати медецину, тоді і виникло пришвидшення розвитку електронних сервісів у сфері охорони здоров’я. Проте, навіть попри показові результати зрушення ринку, багато проблем залишаються актуальними.

Зараз коли ми живемо в умовах повномасштабної війни в Україні навантаження на медичну систему значно зросло. Медичні заклади змушені працювати в умовах нестабільного електропостачання, відсутності інтернету, постійного переміщення персоналу та пацієнтів, обмеженого доступу до ресурсів. Це значно ускладнює зберігання, передачу та обробку медичної інформації, особливо якщо вона існує виключно в паперовому вигляді або в закритих локальних системах.

Системи охорони здоров'я стають ціллю для кібератак з боку ворога [9]. Це створює додаткову потребу у використанні безпечних та надійних програмних рішень, здатних працювати в умовах надзвичайних ситуацій. Особливо важливо забезпечити безперервний доступ до медичних записів, результатів аналізів, розкладів та призначень лікарів - незалежно від фізичного місцезнаходження пацієнта чи лікаря.

В умовах воєнного стану в державі для медичних систем утворюються специфічні виклики для медичних установ:

- необхідність швидкого налагодження медичних процесів;
- потреба у захисті від кібератак з боку агресора;
- потреба у стійкості до збоїв системи;
- зв’язок всіх необхідних функцій у одному програмному рішені.

Тому виникає потреба у створенні ефективної серверної частини медичної інформаційної системи, яка забезпечуватиме весь необхідний функціонал та належну безпеку, а також інтеграцію з іншими цифровими сервісами та системами управління медичним процесом. Така система має бути адаптована до потреб українських медичних закладів і відповідати сучасним технологічним стандартам.

1.2 Аналіз існуючих аналогів

На сьогодні на ринку представлено багато медичних інформаційних систем, які в певній мірі покращують процеси в медичних установах. Більшість із них мають широку функціональність, однак не завжди відповідають потребам українських користувачів, мають складний інтерфейс, проблеми з освоєнням функціоналу працівниками медичної сфери та пацієнтами або вимогам до безпеки. Найбільш відомими серед них є Helsi та Health24 з якими навіть була можливість ознайомитись на реальному прикладі як пацієнтці під час відвідування лікарів. Ці системи працюють у рамках державної ініціативи eHealth для покращення роботи лікарень.

Helsi - це одна з перших медичних інформаційних систем в Україні, яка працює з державним сервісом eHealth [10]. Основний акцент зроблено на запис до лікаря онлайн, перегляд електронних медичних записів, взаємодію пацієнта з медичними установами. Система також використовується лікарями для ведення електронних медичних карт, виписки е-рецептів та направлень.

Переваги Helsi:

- повна інтеграція з eHealth;
- онлайн-запис на прийом до лікаря;
- доступ до електронної медичної картки;
- можливість перегляду історії прийомів та призначень;
- підтримка виписки е-рецептів та направлень;
- реальне використання у багатьох медичних закладах.

Недоліки Helsi:

- обмежена можливість налаштування інтерфейсу під специфіку закладу;
- інтуїтивно незрозумілий інтерфейс, що ускладнює роботу персоналу;
- проблеми зі стабільністю роботи, коли пацієнтів дуже багато;
- відсутність адаптації для окремих клінічних підрозділів;
- недостатньо гнучке управління правами доступу користувачів.

Helsi є достатньо важкою для розуміння та користування особливо для літніх людей. Лікарі також досі мають проблеми з використанням цієї системи,

особливо під час зміни сімейного лікаря для пацієнта або переходу дитини до дорослого лікаря, коли потрібно переоформити декларацію одного з батьків на іншу особу. На особистому досвіді було помічено, що в таких випадках виникають проблеми: навіть після зміни декларації всі направлення надходили до батьків, а сам пацієнт не мав можливості їх отримати.

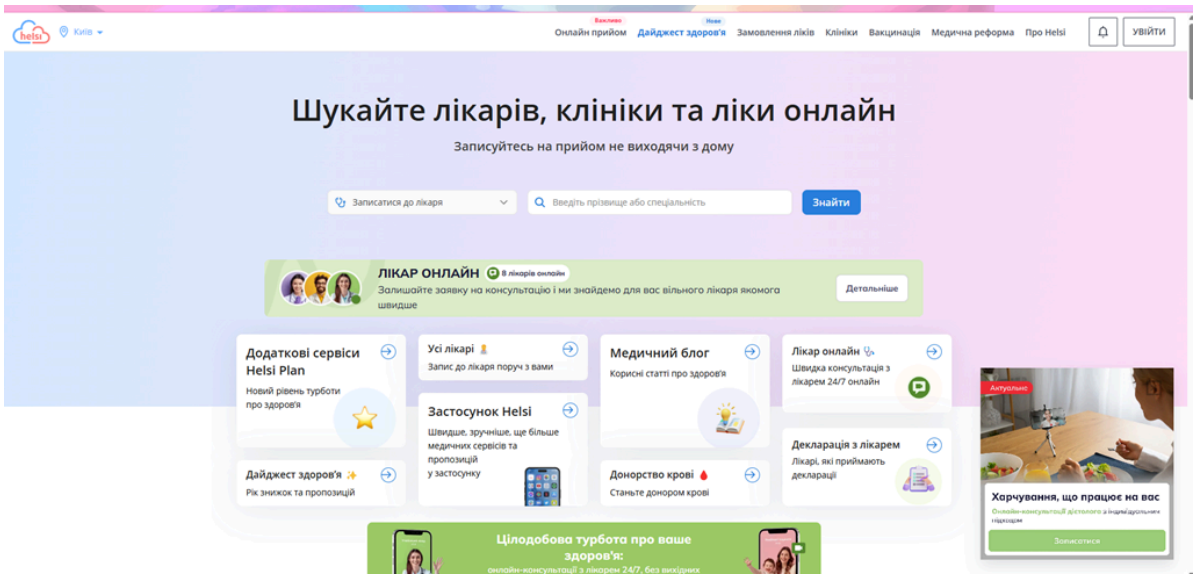

Рисунок 1.1 – Сторінка Helsi (за даними [11])

Health24 - ще одна українська система, що поєднує функції управління медичними записами, комунікації з пацієнтами та обліку медичних послуг. Платформа позиціонується як зручне рішення для приватних клінік і амбулаторій.

Переваги:

- мобільний додаток для лікарів і пацієнтів;
- підтримка обміну даними з eHealth;
- хмарне рішення - не вимагає власної інфраструктури;
- електронні картки, графіки, візити, звіти.

Недоліки:

- частина функцій доступна лише за підпискою;
- менше підтримки для великих багатопрофільних установ;
- обмеження у адаптації;
- слабша техпідтримка в порівнянні з Helsi.

У контексті повномасштабної війни система має як переваги, так і недоліки. Завдяки хмарному рішенні Health24 можна швидко розгорнути у нових локаціях або мобільних клініках, проте відсутність офлайн-режиму обмежує її використання в умовах перебоїв із живленням та інтернетом [12]. Також для державних закладів, які потребують інтеграції з локальними інформаційними системами, можливості Health24 можуть бути недостатніми через закриту архітектуру та фіксований набір сервісів.

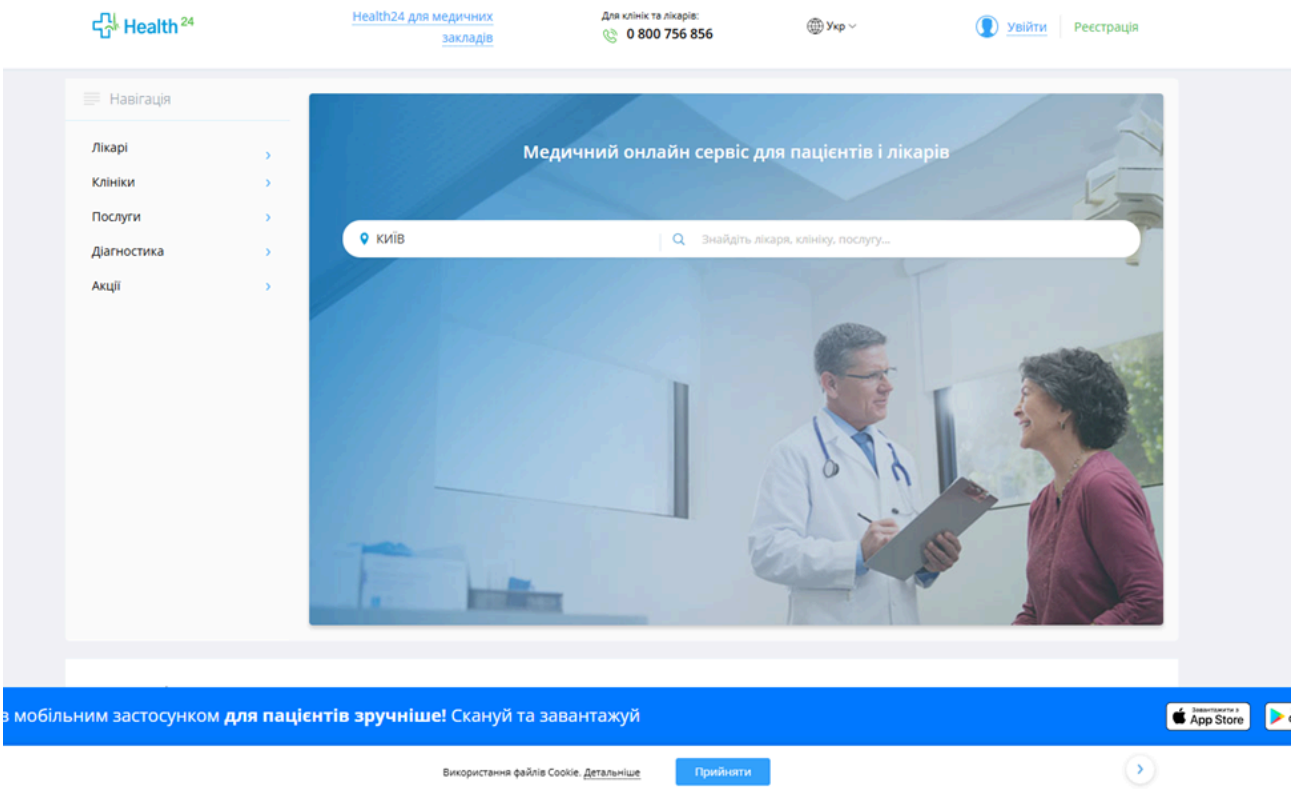

Рисунок 1.2 – Сторінка Health24 (за даними [13])

Деякі функції Health24 працюють з великою затримкою. Виникали труднощі з реєстрацією в системі та отриманні зворотнього зв’язку під час онлайн запису до сімейного лікаря. Також відсутність чітких інструкцій або певних підказок ускладнювало навігацію по сторінкам, особливо це стало б великою проблемою для користувачів, які не мають належного технічного досвіду.

Провівши аналіз існуючих аналогів можемо побачити наявність значних обмежень у вже впроваджених рішеннях. Вони або не дозволяють повною мірою

контролювати функціональність і адаптувати систему під конкретні вимоги, або мають високу вартість та складність інтеграції. Це створює потребу у створенні власної back-end частини програмної системи, яка буде відкритою, масштабованою, технологічно гнучкою, безпечною та повністю адаптованою до умов українських медичних закладів.

Виходячи з результатів аналізу програмна система «LifeLine» буде зосереджена на:

- безпеці, включаючи шифрування чутливої інформації;
- архітектурі, що дозволить легко масштабувати систему;
- зрозумілий інтерфейс взаємодії через REST API;
- реально необхідному функціоналі для справжніх сценаріїв роботи лікарень.

Ринок конкурентів в Україні дуже розвивається та знаходиться зараз у стадії стрімкого росту, існуючі зараз лідери мають впевнені та стійкі позиції, але все таки дають простір для нових рішень. LifeLine має потенціал зайняти місце серед медичних закладів, які хотят нових технічних рішень та готові навчатись новому для зручності у майбутньому.

1.2.1 Потреби клієнтів та ринку

Головними клієнтами даної системи є головуючі та власники медичних закладів, лікарі, медичний персонал та пацієнти. Для них критично важливо мати зрозумілий та продуктивний інструмент для щоденної роботи з медичними даними, який дозволяє швидко приймати рішення, уникати помилок у веденні документації та покращувати якість надання медичних послуг. Back-end частина програмної системи має забезпечувати безперебійну роботу всіх функцій, підтримувати великий обсяг даних та бути готовою до інтеграції з іншими сервісами, зокрема з державною системою eHealth.

Основні потреби клієнтів та ринку, які враховані при розробці серверної частини програмної системи для медичних закладів:

- ефективне управління медичними записами;

- оптимізація та автоматизація внутрішніх процесів;
- зменшення навантаження на адміністративний персонал;
- підвищення якості обслуговування пацієнтів;
- забезпечення швидкої взаємодії лікаря з пацієнтом, зменшення черг;
- високий рівень безпеки та конфіденційності відповідно до чинного законодавства;
- контроль доступу;
- підтримка системи;
- інтеграція з іншими системами;
- стабільна робота серверної частини при великій кількості одночасних користувачів;
- мінімальна затримка;
- можливість швидкого внесення змін відповідно до нових вимог.

1.2.2 Монетизація

Монетизація програмної системи «LifeLine» базується на поєднанні кількох моделей, що дозволяють адаптувати її до потреб різних типів медичних закладів. Основою є модель ліцензування, наприклад, клініки або медичні центри сплачують фіксовану щомісячну або щорічну плату за використання системи. Можливий тип підписки за моделлю SaaS, автоматичні оновлення та технічну підтримку. Для малих приватних кабінетів може застосовуватись freemium-модель: базові можливості надаються безкоштовно, тоді як доступ до розширених модулів, таких як фінансовий облік, розширена аналітика чи інтеграція з лабораторіями, надається на умовах додаткової оплати. Додатковим джерелом доходу є індивідуальні впровадження під замовника, що передбачають оплату налаштування та технічну підтримку. У разі використання у державному секторі можливе фінансування через держбюджет або донорську підтримку.

1.3 Постановка задачі

Проєкт передбачає створення комплексної програмної системи для медичних закладів, яка складається з двох частин: серверної та клієнтської.

Серверна частина має забезпечити стабільну обробку запитів, зберігання та доступ до даних пацієнтів, організацію взаємодії між медичним персоналом і пацієнтами. Основне завдання - реалізувати функціональну, захищену та продуктивну систему, що відповідатиме потребам закладів охорони здоров’я.

Клієнтська частина відповідатиме за інтуїтивно зрозумілий інтерфейс для всіх користувачів системи, за з’єднання функціональних можливостей між сервером та клієнтом і має забезпечити комфортну роботу з системою. До основних функцій належить: реєстрація, авторизація, перегляд, редагування медичної інформації, бронювання прийомів, розклади лікарів, створення та прегляд призначень.

Клієнтська частина взаємодіє з серверною через REST API - кожна дія користувача викликає відповідний запит до сервера, він обробляється та повертає результат клієнту. Така взаємодія дозволяє бути адаптованою на різних пристроях.

На першому етапі розробки важливо чітко окреслити функції, які система має виконувати, щоб максимально відповідати очікуванням користувачів. Сюди входить побудова логіки реєстрації та доступу користувачів із різними рівнями прав, створення цифрової моделі медичних карток, розробка механізмів запису на прийом, ведення розкладу лікарів. Окремим напрямом є обробка медичних записів - зберігання інформації про стан здоров’я, призначення, рецепти, історію відвідувань.

Також необхідно забезпечити передачу даних через мережу у форматі REST API, реалізувати шифрування критичних елементів (паролів та особистих даних пацієнтів). Не мало важливим є обмежити доступ до даних за ролями та запобігти несанкціонованим діям. Всі дії в системі мають бути зрозумілими, пов’язаними між собою та добре налаштованими.

Оскільки робота системи пов’язана з чутливою інформацією, ключовими аспектами є стабільність, контроль помилок і захист даних. При побудові

архітектури важливо враховувати розширюваність рішення та можливість доповнення новими модулями. Після визначення функціонального наповнення необхідно сформувати структуру бази даних, що відображатиме всі основні сутності та взаємозв’язки між ними.

Далі потрібно зробити прототипи сторінок для клієнтської частини системи. Це дозволить ще до початку реалізації перевірити логіку взаємодії користувача з інтерфейсом, оптимально розмістити елементи управління та врахувати потреби різних типів користувачів: лікарів, пацієнтів та адміністративного персоналу.

Після побудови логічної моделі - перейти до реалізації: написання коду, налагодження серверного середовища, підключення до бази даних, створення точок взаємодії для клієнтських додатків. Важливими етапами є перевірка роботи окремих частин системи, налаштування середовища для її розміщення та аналіз продуктивності. В цей самий час розробляється клієнтська частина, яка інтегрується з API.

Завершальним етапом є запуск основного функціоналу системи, її тестування в умовах, наближених до реального використання.

Таким чином, результатом вирішення поставлених задач має стати повнофункціональна медична інформаційна система з клієнтською та серверною частиною.

1.3.1 Цільова аудиторія

Програмна система «LifeLine» орієнтована на користувачів, які:

- працюють у сфері охорони здоров’я;
- здійснюють прийом пацієнтів або займаються медичним обліком;
- керують роботу медичних установ;
- відповідають за зберігання та захист медичних даних;
- потребують швидкого доступу до електронної медичної інформації;
- різного віку;
- різної спеціалізації (медичної або технічної);
- працюють у державних або у приватних лікарнях;

- хочуть позбутись паперової документації
- без досвіду в роботі з подібним програмним забезпеченням.

Система LifeLine розробляється з урахуванням широкого спектру користувачів, її інтерфейс має бути інтуїтивно зрозумілим, а функціонал адаптованим. Це дозволить ефективно використовувати її як у державних, так і в приватних медичних структурах, включаючи стаціонарні клініки, амбулаторії, мобільні пункти допомоги та медичні заклади тимчасового призначення.

2 ФОРМУВАННЯ ВИМОГ ДО ПРОГРАМНОЇ СИСТЕМИ

2.1 Окреслення концепції

У більшості українських лікарень документація, історії хвороб досі зберігаються на папері. Такий підхід забирає надто багато часу. Це не тільки незручно, а й заважає швидко приймати рішення, особливо в критичних моментах. «LifeLine» створена для того, щоб змінити це: дати лікарям, пацієнтам і медичному персоналу простий і надійний інструмент для роботи з даними - без зайвого ускладнення, без ризиків і без втрат часу.

Роль системи у роботі закладу:

- орієнтація на медичну сферу, враховуючи її специфіку;
- максимальне усунення ручної роботи;
- інтерфейс розроблений таким чином, щоб не вимагати тривалого навчання;
- зберігає всі дані в безпечному цифровому вигляді - нічого не загубиться й не зникне;
- зменшує кількість помилок, які виникають під час ручного введення даних;
- підтримує розмежування доступу: кожен бачить тільки те, що йому дозволено;
- модульна архітектура дає можливість змінювати функціональність відповідно до потреб конкретної клініки.

Система створена, щоб спростити щоденну роботу медичних працівників (ведення медичних карток, формування звітності, планування прийомів, взаємодія з пацієнтами) та забезпечити зручний сервіс для пацієнтів (онлайн-запис, доступ до результатів, отримання повідомлень про візити, рецепти). Впровадження цієї системи сприяє зменшенню паперового документообігу, підвищенню ефективності закладу та покращенню якості обслуговування.

Використання сучасних технологій (REST API, авторизація через OAuth2, захищене шифрування даних, мікросервісна архітектура) дозволяє системі

забезпечити високу надійність, стабільність роботи при великих навантаженнях та можливість інтеграції з мобільними або веб-додатками.

Безпека - є дуже важливою для подібних систем. Передбачено застосування сучасних криптографічних стандартів для зберігання і передачі персональних та медичних даних, система автентифікації користувачів, логування дій персоналу. Регулярні оновлення безпеки допомагають зменшити ризики несанкціонованого доступу та відповідають вимогам законодавства у сфері захисту персональних даних.

Також система передбачає технічну підтримку та супровід, щоб користувачі могли легко працювати в системі без зайвих проблем. Адміністратори й медичний персонал можуть звертатися по допомогу у налаштуванні, оновленнях, інтеграції, що забезпечує стабільну роботу на всіх етапах експлуатації.

Загалом, «LifeLine» має на меті створити повну цифровізацію процесів у медичних установах, підвищення ефективності роботи персоналу, зниження адміністративного навантаження, зручність для пацієнтів і загальне покращення якості медичних послуг. Система здатна легко підлаштовуватись під різні потреби. Вона однаково добре працюватиме в маленьких та великих закладах. Програмна система буде зростати разом з закладами. З часом будуть додаватись нові функції, в залежності від розвитку медичної сфери, нічого не потрібно робити заново.

2.2 Головна функціональність

Серверна частина системи «LifeLine» виконує ключову роль у забезпеченні стабільної, безпечної та структурованої роботи всіх внутрішніх процесів медичного закладу. Основна функціональність back-end складової спрямована на обробку, зберігання, захист і надання доступу до медичних даних для авторизованих користувачів, таких як лікарі, медичний персонал та адміністрація.

До основних функцій системи належать:

- управління електронними медичними картками;
- реєстрація та обробка прийомів ;
- створення електронних рецептів та направлень;

- адміністрування доступу користувачів за ролями (Admin, Doctor, Patient);
- ведення розкладу лікарів, лабораторних та медичних послуг;
- формування звітності та статистики;
- захист та шифрування даних, автентифікація та авторизація;
- запис на прийом до лікаря або послугу онлайн;
- резервне копіювання даних;
- перегляд завантаженості лікарів, контроль відвідувань та ефективності;
- запис на прийом онлайн;
- модуль обробки медичних послуг;
- автоматичне формування медичних довідок та результатів у форматі PDF;
- інтерфейс API для зовнішніх інтеграцій;
- модуль оплати послуг (інтеграція з платіжними сервісами);
- панель адміністратора для керування установами, працівниками, пацієнтами;
- аналітичні панелі для перегляду активності пацієнтів та лікарів;
- модуль обліку та управління лабораторними аналізами (призначення, результати, розклади);
- управління фінансовими звітами;
- створення, редагування та перегляд медичних послуг за лікарями або закладами;
- система оцінювання та перегляду відгуків на лікарів і клініки;
- підтримка персоналізованих розкладів лікарів і послуг;
- відображення історії дій користувачів та зміни статусів записів.

2.3 Нефункціональні вимоги

Виходячи з постановки задачі було визначено важливі нефункціональні вимоги для забезпечення довготривалої, ефективної роботи серверної частини програмного забезпечення у сфері охорони здоров’я:

- система має працювати швидко, навіть коли одночасно багато користувачів;
- має бути можливість легко додавати нові модулі, лікарні чи сервери без переробки всієї системи;
- система має працювати постійно, без перебоїв;
- приватні дані мають зберігатися у зашифрованому вигляді, з підтримкою сучасних стандартів криптографії, доступ до них має бути тільки у тих, хто має права.
- система має відповідати вимогам законодавства щодо захисту персональних даних;
- інтерфейс повинен бути простим і зрозумілим, а API бути добре описаними;
- архітектура системи має бути побудована так, щоб забезпечувати легку перевірку працездатності її компонентів;
- система має підтримувати українську мову як основну, а також мати можливість додавання інших мов для зручності використання в різних країнах;
- система має бути сумісна з різними типами клієнтів;
- оновлення системи повинні бути безпечними, передбачаючи безперервність роботи без втрати даних.

2.4 Рамки первинного випуску

Первинний випуск серверної частини програмної системи «LifeLine» охоплює базовий набір функціональності, достатній для впровадження у невеликому або середньому медичному закладі.

- створення, редагування та зберігання інформації про пацієнтів;
- електронна медична картка з усіма оглядами, діагнозами, призначеннями;
- запис на прийом до лікаря - пацієнт може бачити доступні години і обирати зручний час;

- управління графіком лікарів - адміністратор задає дні та години роботи;
- вхід до системи з розподілом прав: лікарі бачать своїх пацієнтів, пацієнти - тільки свою інформацію;
- усі паролі зберігаються зашифрованими, доступ до медичних даних мають лише уповноважені;
- формування довідок, направлень і рецептів у PDF-форматі;
- основна звітність;
- реалізація основних ендпоінтів для взаємодії з клієнтськими застосунками;
- перевірка правильності введених даних - система попереджає про помилки у формі;
- облік оплат прийомів, аналізів, медичних послуг, інтеграція з онлайн-оплатою;
- пацієнти можуть оцінити якість прийому та залишити відгук з автоматичним надісланням його на пошту медичного закладу;
- підтримка резервного копіювання, щоб не втрачати дані в разі збою;
- швидкий пошук пацієнта, лікаря або документа;
- зручна фільтрація пацієнтів за діагнозами, датами візитів або лікарями;
- можливість швидкого розширення функцій у майбутніх оновленнях.

2.5 Рамки наступних випусків

Після реалізації первинної версії серверної частини системи «LifeLine» заплановано поступове розширення функціональності. Це дозволить адаптувати систему до складніших бізнес-процесів медичних закладів.

Очікується, що в наступних випусках будуть реалізовані такі можливості:

- надсилання пацієнтам нагадувань про візити, змін у графіках, готові аналізи через email, SMS або push-нотифікації;
- мобільний API та оптимізація: адаптація REST API для мобільних додатків, такі як оптимізація запитів, кешування, швидка синхронізація;

- створення віртуальних прийомів, відеозв’язку між пацієнтом і лікарем, збереження протоколу онлайн-консультацій;
- обмін повідомленнями між співробітниками у межах медичного закладу, фіксація повідомлень у контексті візитів;
- формалізована система прийому й обробки скарг, пропозицій, запитів, з прив'язкою до користувача та відслідковуванням статусу;
- підтримка електронного підпису для юридично значущих документів;
- багаторівнева система ролей з гнучкими дозволами, адаптованими до великих установ;
- гнучка система створення шаблонів документів;
- інтеграція з eHealth - повна відповідність державним протоколам щодо електронних направлень, е-рецептів, декларацій, медичних подій.

Ці функціональні модулі будуть поступово впроваджуватись у наступних версіях системи на основі запитів користувачів.

2.6 Обмеження та винятки

Система може мати такі обмеження та винятки:

- працює лише в умовах наявного інтернет-з’єднання (віддалене або локальне), що обмежує можливість автономної роботи;
- не передбачено підтримки застарілих версій браузерів і операційних систем, які не відповідають сучасним вимогам;
- встановлення системи на застаріле серверне або мережеве обладнання може призвести до часткової втрати функціональності або нестабільної роботи через несумісність із сучасними компонентами;
- деякі функціональні модулі (наприклад, глибока аналітика, інтеграція з лабораторіями, автоматичні повідомлення) поки що не входять до поточної версії системи і заплановані на наступні етапи розробки;
- при використанні нових технологічних рішень (наприклад, REST API останніх версій, сучасні стандарти безпеки) можуть виникати складнощі

при взаємодії з зовнішніми системами, що використовують застарілі протоколи чи методи обміну даними.

2.7 Робоче середовище

Серверна частина програмної системи «LifeLine» розробляється з урахуванням сучасних технологічних вимог і потреб користувачів з різних медичних закладів. У розробці особливо важлива увага приділена безпеці, швидкості роботи та зручності підтримки.

Передбачається використання інтернет-з'єднання для доступу до функціональності системи як з боку персоналу медичних установ, так і пацієнтів. Завдяки централізованій архітектурі система дозволяє зберігати дані на єдиному сервері, що забезпечує цілісність та доступність інформації в реальному часі.

Для безпеки даних система використовує перевірені засоби автентифікації та авторизації. Паролі зберігаються у хешованому вигляді з використанням алгоритму bcrypt. Кожен користувач після входу отримує унікальний JWT-токен, який дозволяє захищено взаємодіяти з серверною частиною без потреби повторної авторизації при кожному запиті. Це гарантує надійний контроль доступу до ресурсів, захищає від несанкціонованого втручання та відповідає загальноприйнятим стандартам захисту персональних даних.

Одним із пріоритетів у розробці є стабільність системи. У разі виникнення технічних збоїв сервер повинен відновити роботу без втрати даних або необхідності ручного втручання. Розподілена логіка збереження даних, записування подій та обробка помилок сприяють підтримці безперервного функціонування. Система має демонструвати високу продуктивність навіть за умов пікового навантаження, що включає мінімальні затримки при обробці запитів та швидку відповідь API.

Очікується, що користувачі зможуть працювати із системою через сучасні браузери (Google Chrome, Mozilla Firefox, Microsoft Edge, Safari), які підтримують актуальні стандарти HTML5, CSS3 та JavaScript. Інтерфейси будуть адаптовані під

різні роздільні здатності екранів, що забезпечить коректне відображення як на комп’ютерах, так і на мобільних пристроях.

Серверна частина буде реалізована на мові програмування JavaScript із використанням середовища виконання Node.js. Це дозволяє створити подієво-орієнтовану систему, яка ефективно обробляє велику кількість одночасних запитів. Для керування базами даних використовується PostgreSQL - надійне рішення з підтримкою транзакцій, реляційної структури та високої продуктивності. Додатково використовуються ORM-бібліотека Sequelize для спрощення роботи з базою даних і підтримки взаємозв’язків між сутностями.

Система «LifeLine» побудована на перевірених та надійних технологіях. Вони дозволяють їй працювати без затримок, безпечно, стійко. Буде адаптивно до різних типів медичних установ та розширюватись дуже легко при необхідності.

3 АРХІТЕКТУРА ТА ПРОЄКТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

3.1 Архітектурні рішення та використані технології

Back-end частина системи створена з використанням мови JavaScript та середовища Node.js, яке забезпечує обробку запитів, виконання логіки додатку та взаємодію з базою даних. Для зберігання структурованої інформації застосовується система керування базами даних PostgreSQL, яка відзначається стабільністю та гнучкістю у роботі з великими обсягами медичних даних.

Усі структури, зв’язки та правила між сутностями системи реалізовані за допомогою бібліотеки Sequelize, що дозволяє працювати з базою не напряму через SQL, а через зрозумілі програмні моделі. Для зручності налаштувань середовища використовується окрема конфігурація через змінні .env.

На схемі представлено основні компоненти серверної частини програмної системи «LifeLine» та їх взаємодію (див.рис.3.1).

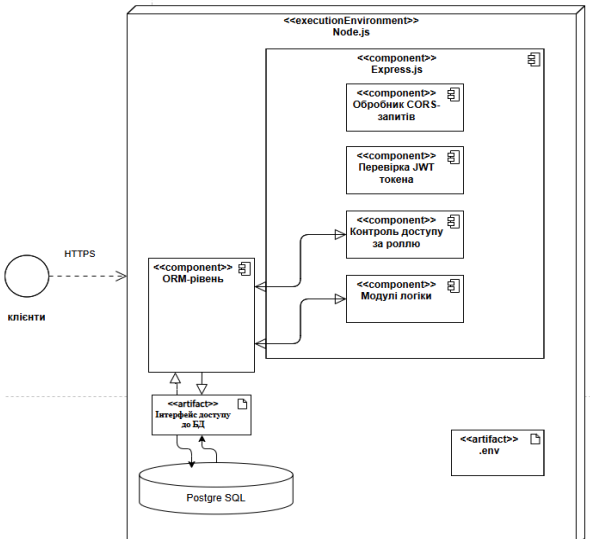

Рисунок 3.1 – Діаграма розгортання програмної системи (Рисунок виконано самостійно)

Контроль доступу до системи реалізовано через токени авторизації, які створюються після входу користувача та перевіряються при кожному запиті. Паролі та особисті дані пацієнтів зберігаються у зашифрованому вигляді, що підвищує рівень захисту.

У якості основи для зберігання даних у системі використовується PostgreSQL - стійка та функціональна база, яка добре підходить для медичних проєктів завдяки своїй надійності та підтримці складних зв’язків між таблицями. Всі дані зберігаються саме там.

Ключові процеси системи реалізовані на серверній стороні. Вона відповідає за основну логіку роботи, перевірку прав доступу, обробку запитів, взаємодію з іншими частинами застосунку. Комунікація між сервером і зовнішніми клієнтами (браузером або мобільним додатком) здійснюється через захищені HTTP-запити. Обмін відбувається у форматі REST, що робить взаємодію передбачуваною та структурованою.

Щоб зберегти безпеку під час передавання даних, використовується HTTPS - зашифрований протокол, який захищає інформацію між клієнтом і сервером. Це важливо при роботі з медичною інформацією або особистими даними пацієнтів.

Авторизація в системі реалізована через токени доступу. Після того як користувач успішно проходить перевірку, система створює спеціальний токен (JWT), в якому закодовано інформацію про користувача: його ID, роль та строк дії. Цей токен зберігається на боці клієнта і автоматично додається до всіх наступних запитів. Якщо токен не дійсний або прострочений (через 24 години), користувач має повторно увійти в систему. Це дозволяє чітко контролювати, хто і коли звертається до захищених частин застосунку.

Паролі користувачів не зберігаються у відкритому вигляді. Перед тим, як потрапити до бази, вони проходять через процес хешування, що унеможливлює їх. У системі реалізовано шифрування чутливої інформації пацієнтів (ПІБ, email, адреса, телефон, алергії тощо) за допомогою алгоритму AES-256-GCM. Дані зберігаються не у відкритому вигляді, а у зашифрованому. Навіть якщо хтось отримає доступ до бази, він не зможе прочитати ці дані без спеціального ключа.

Перед тим як записати дані в базу, система додає випадковий код і захисний тег автентичності, щоб можна було перевірити, чи дані не було змінено. А при читанні дані автоматично розшифровуються - якщо є правильний ключ. Це допомагає надійно захищати приватну інформацію користувачів.

Для роботи з базою та керування її станом передбачено набір API-запитів, через які можна, наприклад, оновити структуру або керувати записами. Програмна логіка організована в окремі блоки, тобто код поділений на незалежні частини, що полегшує навігацію та подальше розширення. У процесі реалізації активно використовуються сторонні модулі та бібліотеки.

3.2 Моделювання програмної системи

Для відображення основних функціональних можливостей серверної частини медичної інформаційної системи «LifeLine» була створена діаграма прецедентів (див. рис. 3.2). Вона моделює ключові сценарії взаємодії користувачів із системою та визначає, які саме дії доступні представникам кожної ролі.

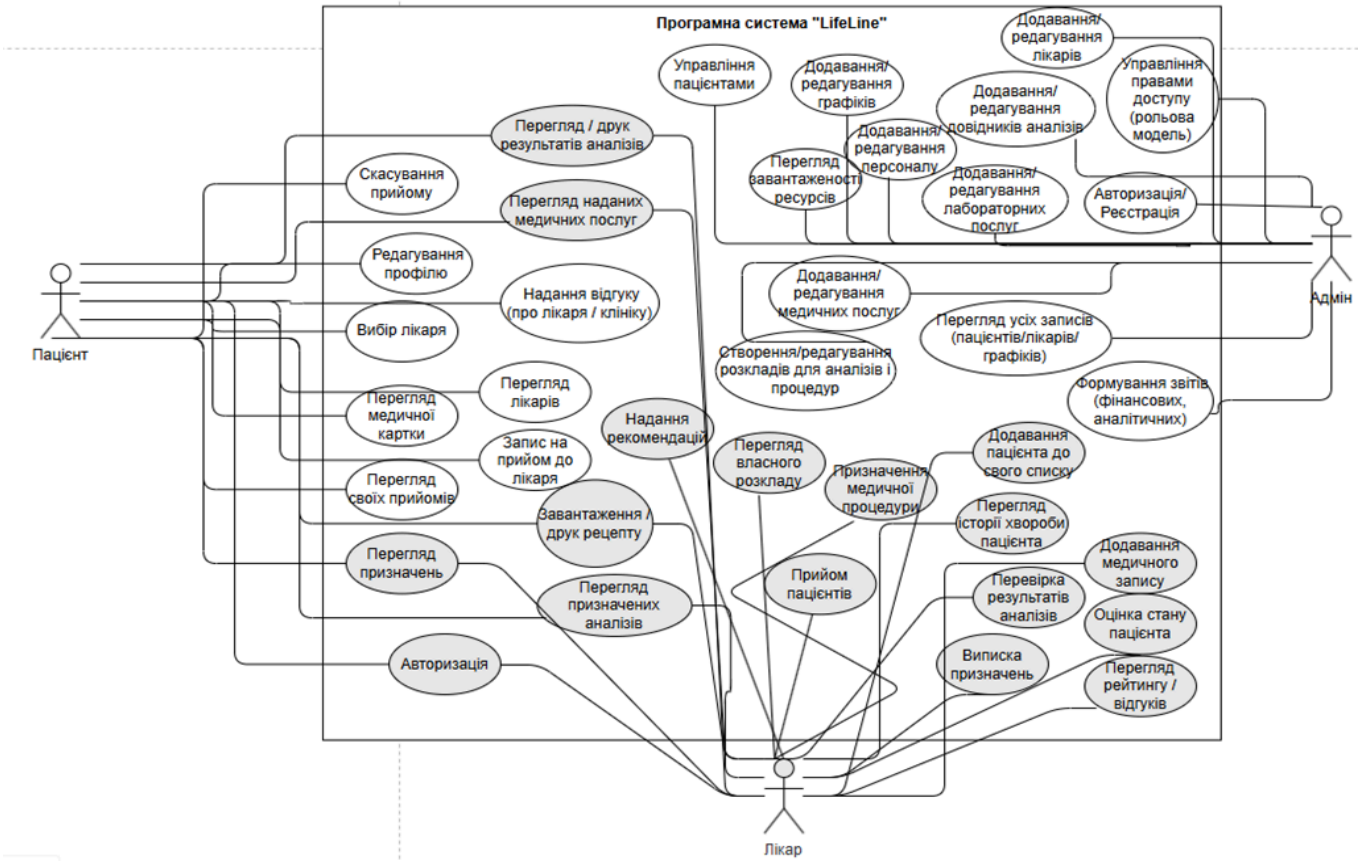

Рисунок 3.2 – Діаграма прецедентів (Рисунок виконано самостійно)

У системі передбачено три основні категорії користувачів: Пацієнт, Лікар та Адміністратор, кожна з яких має власний набір дозволених операцій.

Пацієнт - це користувач, який отримує медичні послуги. Після входу в систему він має змогу переглядати та редагувати свій профіль, змінювати фото та контактні дані. Вбудовані функції дозволяють швидко знайти лікаря за спеціальністю або назвою закладу, перевірити його розклад і записатись на прийом у зручний час.

Пацієнт може скасувати запис, переглянути історію відвідувань, переглядати результати аналізів, обстежень, електронні рецепти й призначення, які формуються автоматично та доступні для завантаження у PDF. Система дає доступ до електронної медичної картки з усією історією лікування. Існує змога записуватись на лабораторні та інші медичні послуги, оплачувати їх через інтегровані сервіси, а також переглядати графіки надання цих послуг. Після отримання допомоги пацієнт може залишити відгук про лікаря або клініку, поставити оцінку, що формує загальний рейтинг якості обслуговування.

Лікар - зареєстрований користувач із розширеними правами, який надає медичні послуги. Йому доступний розклад власних прийомів, можливість перегляду списку пацієнтів, записаних на візит. Під час прийому лікар може заповнювати медичну картку пацієнта, створювати медичні записи, призначати процедури, лабораторні дослідження, виписувати електронні рецепти. Додатково можна оцінювати стан пацієнта, залишати медичні рекомендації, переглядати історію хвороби пацієнта, результати аналізів, оцінювати загальну динаміку лікування. У його повноваження також входить перегляд власного рейтингу на основі зворотного зв’язку пацієнтів.

Адміністратор - це користувач з максимально широкими правами доступу, який керує всіма ресурсами системи. Адміністратор відповідає за лікарів, медичний персонал, клініки, графіки лікарів, лабораторні аналізи, медичні послуги, довідкову інформацію. Саме він керує ролями та доступом користувачів.

Адміністратор формує звіти про кількість прийомів, доходи, навантаження лікарів, популярні послуги тощо. Це допомагає головуючим приймати рішення для покращення роботи клініки. Він стежить за тим, як працює система,

контролює активність користувачів, оновлює налаштування та відповідає за загальну стабільність роботи клініки.

Діаграма діяльності, зображена на рисунку 3.3, демонструє типовий сценарій взаємодії пацієнта з програмною системою «LifeLine» - від входу в систему до отримання результатів медичних процедур. Вона охоплює ключові дії, які виконує користувач під час запису до лікаря, отримання призначень та проходження медичних процедур.

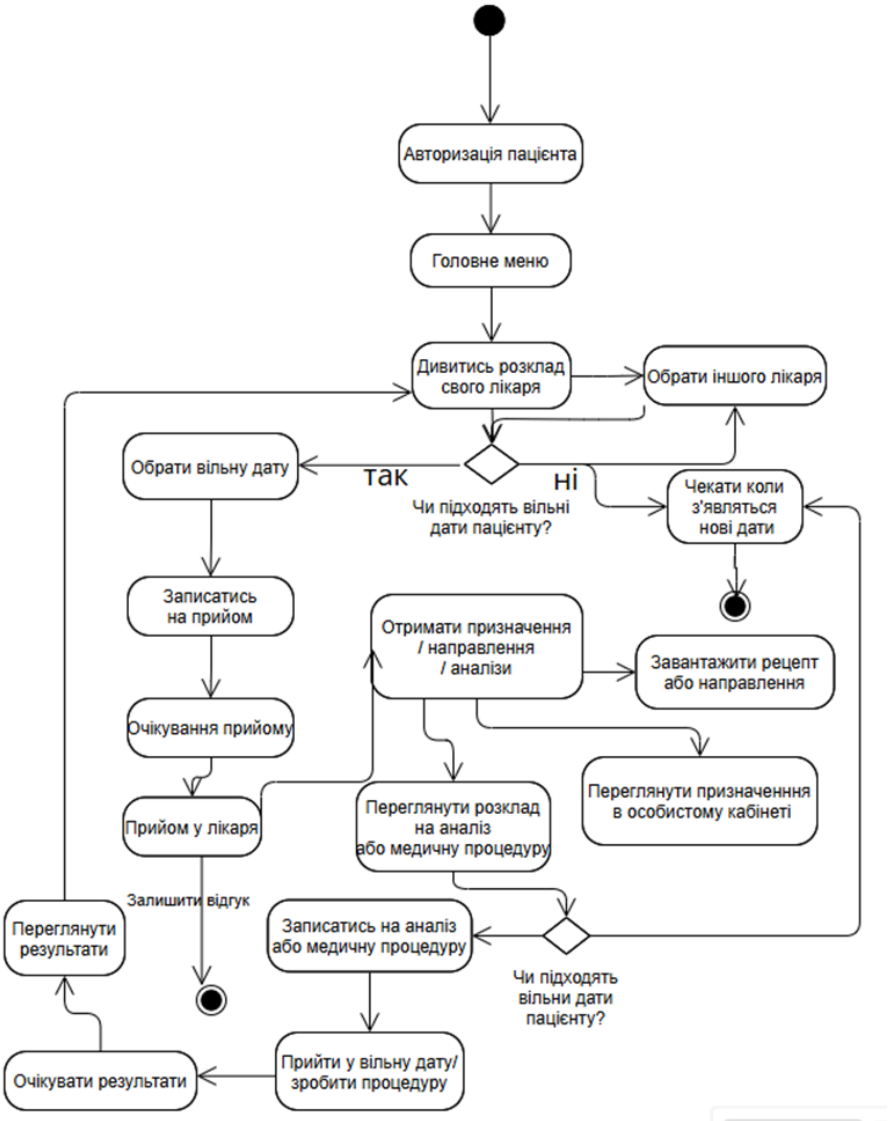

Рисунок 3.3 – Діаграма діяльності (Рисунок виконано самостійно)

Процес починається з авторизації пацієнта, після чого він переходить до головного меню, де йому доступна можливість переглянути розклад свого лікаря.

У випадку, якщо поточний лікар не має зручного графіка, пацієнт може обрати іншого лікаря.

На наступному етапі відбувається перевірка: чи підходять вільні дати пацієнту. Якщо так - пацієнт обирає дату, записується на прийом та очікує візиту. Після цього проходить прийом у лікаря, з можливістю залишити відгук. Якщо ж пацієнту призначено аналіз або медичну процедуру, він може переглянути розклад, записатись на вільну дату та прийти для проходження процедури. Після виконання процедури пацієнт очікує на результати та згодом може їх переглянути в системі.

Якщо ж вільні дати лікаря не підходять, користувач може або чекати, коли з’являться нові дати, або переглянути призначення в особистому кабінеті і завантажити рецепт або направлення.

Діаграма демонструє повний життєвий цикл взаємодії пацієнта з системою, забезпечуючи логічну послідовність дій. Даний підхід дозволяє пацієнтам управляти своїм лікуванням, отримувати якісний сервіс і зберігати медичну інформацію в зручному цифровому форматі.

3.3 Проєктування моделі даних

База даних програмної системи управління медичними закладами включає 21 таблицю, що відповідають ключовим сутностям і процесам у медичній галузі. Для побудови структури було використано ORM-бібліотеку Sequelize, яка дозволяє описувати моделі в об'єктно-орієнтованому стилі.

Список таблиць та їх призначення:

- User - облікові записи користувачів системи (пацієнтів, лікарів, адміністраторів), із валідацією email, пароля та ролями;
- Hospital - заклади охорони здоров’я з адресою, типом (державна або приватна), контактами та графіком роботи;
- HospitalStaff - працівники лікарень, які не є лікарями, з контактними даними та посадами;

- Patient - персональні дані пацієнтів, включаючи медичну історію, алергії, хронічні захворювання, кров, стать;
- Doctor - лікарі з прив’язкою до лікарні, спеціалізацією, досвідом, кабінетом, біографією та контактами;
- DoctorSchedule - графіки прийому лікарів, з часом, датою та можливістю запису;
- Appointment - запис пацієнтів на прийом до лікаря, з нотатками, статусом та датою;
- MedicalRecord - історія хвороби пацієнта: діагнози, призначення лікування, дати записів;
- Prescription - електронні рецепти з прив’язкою до медичних записів, дозуванням, строком дії;
- LabTestInfo - довідкова інформація про лабораторні аналізи: назва, ціна, опис, показання;
- HospitalLabService - перелік доступних аналізів у конкретній лікарні, які може призначити лікар;
- LabTestSchedule - графіки проведення аналізів: дата, час, бронювання;
- LabTest - конкретні проведені аналізи пацієнтів із результатами;
- FinancialReport - звіти про фінансову діяльність медичного закладу;
- Analytics - збереження показників ефективності: кількість візитів, доходів, витрат;
- Review - відгуки користувачів про лікарів або заклади, із оцінками та коментарями;
- MedicalServiceInfo - довідкова інформація про процедури: назва, опис, ціна, тривалість;
- HospitalMedicalService - доступність медичних процедур у закладі, призначених певними лікарями;
- MedicalServiceSchedule - розклад медичних послуг, який дозволяє записатись на процедуру;

- MedicalService - конкретна реалізація наданої медичної послуги з результатами;
- UsedOrder - підтверджені платіжні операції користувачів, із зазначенням призначення оплати (медична послуга, аналіз тощо), даними платника, сумою, комісією та часом підтвердження.

Ключові зв’язки між сутностями:

- один користувач може бути або пацієнтом, або лікарем, або працівником;
- пацієнт і лікар мають зв’язок через лікарню, записи на прийом, рецепти та медичні записи;
- лікар має графік прийому (DoctorSchedule), який використовується для створення запису (Appointment);
- медичний запис (MedicalRecord) може мати декілька рецептів (Prescription);
- кожна лікарня має власний список аналізів і процедур через HospitalLabService та HospitalMedicalService;
- відгуки можуть бути залишені як про лікарів, так і про установи;
- дані про медичні послуги, аналітику та фінансові звіти дають змогу формувати статистику по закладу;
- медичні послуги (MedicalService) та лабораторні аналізи (LabTest) належать до пацієнтів і пов’язані з відповідними графіками;
- зв’язки між лікарнею та аналізами/послугами реалізовані через проміжні сутності (HospitalLabService, HospitalMedicalService), що дозволяє додавати контекст (лікар, доступність);
- кожен запис або процедура може бути оплачена онлайн, а дані про транзакцію зберігаються у UsedOrder, включаючи тип послуги (lab, medical), платника, суму та комісію;
- оплачені послуги (UsedOrder) пов’язані з користувачем (User), що ініціював транзакцію, та містять інформацію про платіжну систему.

ER-модель (див. рис. Д.1) наочно демонструє ці зв’язки та сутності, що лежать в основі розробки бази даних.

3.4 Специфікація REST

Програмна система «LifeLine» включає реалізацію REST API, яка використовується для обміну даними між клієнтською частиною та сервером медичної інформаційної системи. API побудоване за принципами REST-архітектури та забезпечує доступ до основних функціональних модулів системи (див.таб.3.1 та Г.1 у додатку Г).

Таблиця 3.1 - Специфікація REST (таблиця виконана самостійно)

HTTP метод	Кінцева точка	Опис
POST	/api/user/login	Авторизація користувача в системі
POST	/api/user/registration	Реєстрація нового користувача (лише для Лікаря та Адміна)
GET	/api/user/auth	Перевірка токена користувача
GET	/api/user/:id	Отримання даних користувача за ID (лише авторизовані)
PUT	/api/user/update/:id	Оновлення даних користувача за ID (лише авторизовані)
POST	/api/review	Створити відгук (авторизований користувач)
PUT	/api/review/:id	Оновити відгук
DELETE	/api/review/:id	Видалити відгук

Продовження таблиці 3.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
DELETE	/api/user/delete/:id	Видалення користувача (лише Admin)
GET	/api/analytics/top-doctors	Отримати топ лікарів за рейтингом (лише Admin)
GET	/api/analytics/weekly-visits	Отримати тижневу кількість відвідувань (лише Admin)
GET	/api/analytics/monthly-income	Отримати місячний дохід лікарні (лише Admin)
GET	/api/analytics/average-doctor-rating	Середній рейтинг лікарів (лише Admin)
GET	/api/analytics/most-active-patients	Найактивніші пацієнти (лише Admin)
GET	/api/analytics/most-requested-lab-tests	Найпопулярніші аналізи (лише Admin)
GET	/api/analytics/most-used-medical-services	Найпопулярніші медичні послуги (лише Admin)
GET	/api/appointments/upcoming/doctor/:doctorId	Майбутні прийоми лікаря (лише Admin)
PATCH	/api/appointments/:id/complete	Позначити прийом як завершений (лише для Лікаря та Адміна)
GET	/api/appointments/upcoming/patient/:patientId	Майбутні прийоми пацієнта
GET	/api/appointments	Отримати всі прийоми (лише Admin)
GET	/api/appointments/:id	Отримати прийом по ID
POST	/api/appointments	Створити новий прийом
PUT	/api/appointments/:id	Оновити прийом (лише для Лікаря та Адміна)

Продовження таблиці 3.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
PATCH	/api/appointments/:id/cancel	Скасувати прийом (лише для Пацієнта)
GET	/api/appointments/patient/:patientId	Отримати всі прийоми пацієнта
GET	/api/appointments/doctor/:doctorId	Отримати всі прийоми лікаря (лише для Лікаря та Адміна)
GET	/api/doctors/specializations	Отримати унікальні спеціалізації лікарів
GET	/api/doctors/by-user/:userId	Отримати лікаря за userId (доступно для Admin, Doctor)
GET	/api/doctors	Отримати список усіх лікарів
GET	/api/doctors/:id	Отримати лікаря за Id
GET	/api/doctors/by-hospital/:hospitalId	Отримати лікарів певної лікарні
POST	/api/doctors	Створити лікаря (лише Admin)
PUT	/api/doctors/:id	Оновити лікаря за Id (лише Admin)
DELETE	/api/doctors/:id	Видалити лікаря за Id (лише Admin)
GET	/api/doctor-schedules/working-hours/:doctorId/:date	Отримати години роботи лікаря на конкретну дату
GET	/api/doctor-schedules/working-hours/:doctorId/:date	Отримати розклад лікаря на день
GET	/api/doctor-schedules	Отримати всі розклади (лише Admin)
GET	/api/doctor-schedules/hospital/:hospitalId	Отримати розклад за лікарнею

Продовження таблиці 3.1 - Специфікація REST

HTTP метод	Кінцева точка	Опис
GET	/api/doctor-schedules/doctor/:doctorId	Отримати розклад певного лікаря
POST	/api/doctor-schedules	Створити розклад (лише Admin)
PUT	/api/doctor-schedules/:id	Оновити розклад (лише Admin)
DELETE	/api/doctor-schedules/:id	Видалити розклад (лише Admin)
POST	/api/doctor-schedules/:scheduleId/book	Забронювати конкретний слот розкладу
GET	/api/financial-reports/report/day	Фінансовий звіт за день (лише Admin)
GET	/api/financial-reports/report/month	Фінансовий звіт за місяць (лише Admin)
GET	/api/financial-reports/report/year	Фінансовий звіт за рік (лише Admin)
GET	/api/hospital-lab-services	Отримати список всіх лабораторних послуг
GET	/api/hospital-lab-services/:id	Отримати лабораторну послугу за id
GET	/api/hospital-lab-services/hospital/:hospitalId	Отримати лабораторні послуги певної лікарні
POST	/api/hospital-lab-services	Створити лабораторну послугу (лише Admin)
PUT	/api/hospital-lab-services/:id	Оновити лабораторну послугу (лише Admin)
DELETE	/api/hospital-lab-services/:id	Видалити лабораторну послугу (лише Admin)
GET	/api/hospital-medical-services/hospital/:hospitalId	Отримати послуги лікарні

4 ОПИС ПРИЙНЯТИХ ПРОГРАМНИХ РІШЕНЬ

4.1 Опис обраних програмних рішень для серверної частин

При розробці програмного забезпечення LifeLine було особливо важливо обрати технологічні рішення, які б не тільки відповідали сучасним вимогам до продуктивності та безпеки але й створювали надійний фундамент для медичної інформаційної платформи. Враховуючи специфіку галузі охорони здоров'я, де швидкість обробки запитів, стабільність роботи та безпека даних є критично важливими.

В якості серверного середовища було обрано Node.js у поєднанні з фреймворком Express.js. Таке рішення дозволило реалізувати асинхронну обробку запитів, що є важливою перевагою для системи, де одночасно можуть надходити десятки і навіть сотні запитів - від запису пацієнтів до лікаря до збереження результатів лабораторних досліджень. Асинхронна обробка дозволяє серверу працювати швидко, не блокуючи потік даних і підтримуючи безперебійну роботу навіть при високому навантаженні. Node.js добре масштабується, а це означає, що система може рости разом з кількістю користувачів без втрати продуктивності. Express.js, в свою чергу, забезпечує зручну структуру коду, гнучкість у налаштуванні маршрутизації та розширюваність, що є значними перевагами для створення сучасного REST API.

Для управління базою даних було обрано PostgreSQL - сучасну реляційну СУБД, яка зарекомендувала себе як стабільне, безпечне та функціональне рішення. Основною причиною такого вибору стала її надійність та здатність обробляти складні запити.

Приклад визначення моделі:

```
const MedicalService = sequelize.define('MedicalService', {
  name: {
    type: DataTypes.STRING,
    allowNull: false
  },
  is_ready: {
```

```
        type: DataTypes.BOOLEAN,
        defaultValue: false
    }
})
```

Архітектура серверної частини програмної системи побудована за принципом Model-View-Controller, така модель дає можливість організувати код по розподілу відповідальності між окремими компонентами системи. Контролери виступають як суміжна частина між клієнтською частиною та моделями бази даних. Вони обробляють запити, ініціюють відповідні методи моделі для обробки даних та генерують відповіді, які бачить клієнт системи. Моделі відповідають за опис структури даних та їх взаємодію з базою даних, вони представляють основні сутності системи, такі як пацієнти, лікарі, лікарні, медичні послуги, розклади тощо. Для наочного представлення організації серверної частини програмної системи «LifeLine» було побудовано компонентну діаграму (див.рис.4.1).

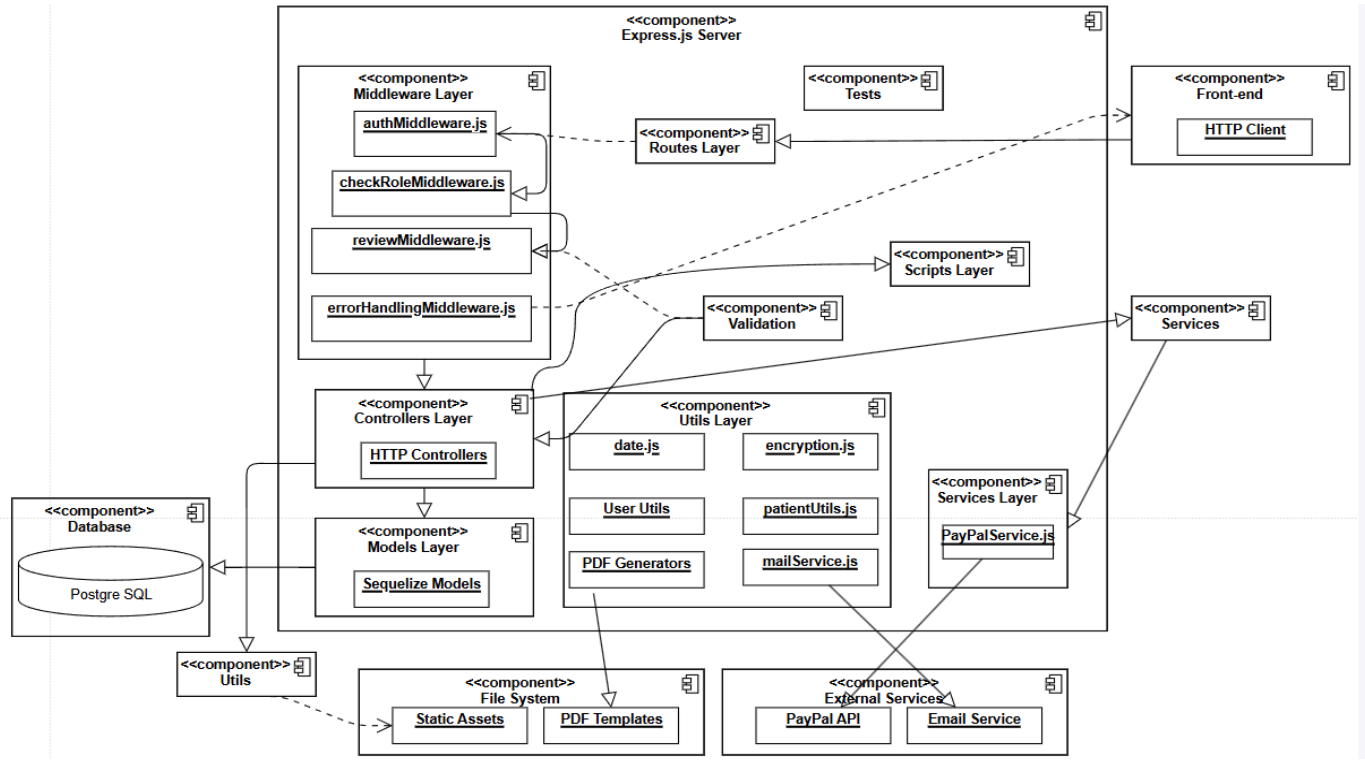

Рисунок 4.1 – Компонентна діаграма архітектури серверної частини системи «LifeLine» (рисунок виконано самостійно)

Кожна модель має чітко визначені атрибути, типи даних та зв'язки з іншими моделями. Маршрути становлять структуру доступу до функціональності додатку через кінцеві точки API. Вони поділені на окремі блоки для зручності та швидкої орієнтації в коді.

Компонентна діаграма ілюструє взаємозв’язок між основними логічними модулями серверної частини системи «LifeLine». Як видно з діаграми, система поділена на незалежні компоненти, кожен з яких виконує окрему функцію.

4.2 Безпека, авторизація та управління доступом

Дуже важливою складовою для медичних систем є безпека персональних даних, медичної інформації та контролю доступу до модулів системи. Після входу кожен кожному користувачу (пацієнту, лікарю або адміністратору) видається токен, який містить його унікальний ідентифікатор, роль та інші необхідні атрибути. Токен зберігається та надсилається з кожним HTTP-запитом для перевірки доступу та авторизації. На серверній частині програми перевіряється кожен запит middleware-функцією, така функція декодує токен та перевіряє на дійсність використовуючи секретний ключ, що зберігається на стороні серверу у файлі .env.

Приклад:

Наведено код перевірки авторизації:

```
module.exports = function (req, res, next) {
    if (req.method === "OPTIONS") return next();

    try {
        const authHeader = req.headers.authorization;
        if (!authHeader) {
                return res.status(401).json({ message: "Користувач не
авторизований" });
        }

        const token = authHeader.split(' ')[1];
        if (!token) {
```

```
                    return res.status(401).json({ message: "Користувач не
авторизований" });
            }

            const decoded = jwt.verify(token, process.env.SECRET_KEY);
            req.user = decoded;
            next();
        } catch (e) {
            console.error('authMiddleware error:', e.message);
                  return res.status(401).json({ message: "Користувач не
авторизований" });
        }
    };
```

Система також має перевірку на право доступу до того чи іншого функціоналу системи залежно від ролі (Patient, Doctor, Admin). Для такої перевірки використовується checkRoleMiddleware.

Приклад:

Наведено застосування у маршруті (Отримати всіх пацієнтів конкретного лікаря (Doctor, Admin)):

```
router.get(
  "/by-doctor/:doctorId",
  checkRole("Doctor", "Admin"),
  patientController.getByDoctor
);
```

Такий підхід дозволяє ефективно обмежувати доступ до забороненої для деяких ролей інформації.

Крім базового шифрування паролів для всіх користувачів системи, реалізовано захист персональної медичної інформації пацієнтів, зокрема адрес, номерів телефону, інформацію про результати обстежень, тощо. Для цього використовуються окремі утиліти шифрування, які застосовуються до вразливих полів під час запису до бази даних. Поля будуть зашифровані із використанням алгоритму AES.

Приклад:

Розглянемо приклад код для шифрування:

```
const crypto = require("crypto");
const algorithm = "aes-256-gcm";
const key = Buffer.from(process.env.ENCRYPTION_KEY, "utf8"); // 32 байти

function encrypt(plainText) {
  if (plainText === null || plainText === undefined) return null;
  const iv = crypto.randomBytes(12);
  const cipher = crypto.createCipheriv(algorithm, key, iv);
  const encrypted = Buffer.concat([
    cipher.update(plainText, "utf8"),
    cipher.final(),
  ]);
  const tag = cipher.getAuthTag();
  return Buffer.concat([iv, tag, encrypted]).toString("base64");
}
```

Зашифрована інформація зберігається у базі даних і виглядає як набір випадкових символів, які неможливо зрозуміти без спеціального ключа (див.рис.4.2), а розшифрування відбувається тільки для тих користувачів, які пройшли перевірку доступу за роллю. Таким чином, навіть у разі витоку БД сторонні особи не можуть прочитати зміст без секретного ключа.

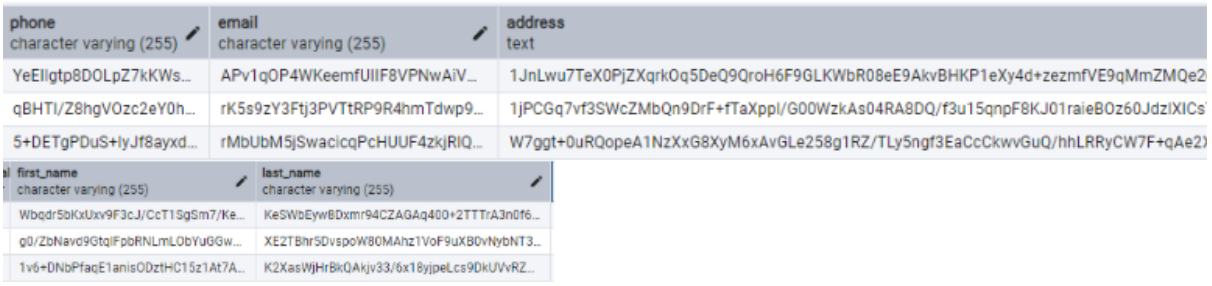

Рисунок 4.2 – Фрагмент збережених зашифрованих персональних даних у таблиці “Patient” у базі даних системи «LifeLine» (рисунок виконано самостійно)

Шифрування здійснюється автоматично під час збереження даних, а розшифрування лише у дозволених сценаріях використання. До того ж система не зберігає ключі у відкритому вигляді - вони захищені додатковим механізмом.

4.3 Додаткові технічні рішення

Система «LifeLine» передбачає можливість онлайн-оплати медичних послуг та аналізів. Для цього реалізовано інтеграцію з PayPal API, яка включає створення транзакції, перехід користувача на платіжну сторінку, перевірку статусу оплати та завершення бронювання.

Приклад:

Фрагмент сервісу підтвердження замовлення:

```
const paypal = require("@paypal/checkout-server-sdk");
const { UsedOrder, FinancialReport } = require("../models/models");
const { Op } = require("sequelize");

const captureOrder = async (orderId) => {
  const existing = await UsedOrder.findOne({ where: { order_id:
orderId } });
  if (existing) throw new Error("Цей платіж уже був використаний");

  const request = new paypal.orders.OrdersCaptureRequest(orderId);
  request.requestBody({});
  const response = await client.execute(request);

  if (response.result.status !== "COMPLETED") {
    throw new Error("Оплата не підтверджена");
  }

  const capture =
response.result.purchase_units[0].payments.captures[0];
  const breakdown = capture.seller_receivable_breakdown;
  const payer = response.result.payer;
```

Модуль pdfGenerator.js створює довідки, історії відвідувань та розклади в PDF-форматі на основі шаблонів. Для реалізації використовується pdfkit, а шаблони зберігаються у директорії templates/. Документи можуть бути завантажені користувачем або збережені на сервері для подальшого використання.

Для автоматизованої взаємодії з користувачами в системі реалізовано модуль поштової розсилки на основі бібліотеки nodemailer. Цей функціонал дозволяє надсилати підтвердження реєстрації, листи з PDF-звітами, сповіщення про записи на прийом тощо.

Приклад:

Фрагмент контроллера з надсилання відгуку на пошту при створенні:

```
// Відправка email з усіма даними
      await MailService.sendReviewNotification(REVIEW_RECEIVER_EMAIL,
{
        patientName,
        patientEmail,
        patientAge,
        patientPhoto,
        targetType: target_type,
        targetName,
        hospitalName,
        rating,
        comment,
        createdAt: review.createdAt,
        reviewId: review.id,
      });
```

Обрані технології забезпечують швидку та стабільну роботу, безпечне зберігання даних, зручну архітектуру та легке розширення в майбутньому. Система підтримує авторизацію, контроль доступу, шифрування персональних даних, онлайн-оплати, PDF-звіти та надсилання повідомлень на пошту, приклад коду для розкладу лікарів наведено в додатках В.1, В.2 .

5 ТЕСТУВАННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ

5.1 Модульне тестування (Unit testing)

Одним із базових підходів тестування роботи програмного забезпечення є модульне тестування, яке дозволяє ізолювати окремі частини коду та перевіряти їх логіку незалежно від інших компонентів системи. Такий тип тестування є ефективним інструментом виявлення помилок ще на етапі розробки, що дозволяє зменшити витрати на їх подальше усунення.

У рамках розробки серверної частини інформаційної системи «LifeLine» модульне тестування було застосовано для перевірки функцій контролерів, утиліт, middleware-функцій авторизації та обробки запитів. Особливу увагу приділено перевірці функцій, які реалізують ключову бізнес-логіку, працюють із базою даних або відповідають за шифрування та дешифрування персональної інформації.

Для цього було створено тестовий сценарій із використанням бібліотеки Supertest, яка дозволяє надсилати HTTP-запити до Express-додатку та перевіряти відповіді сервера.

Тест охоплює кілька важливих кейсів:

- авторизацію адміністратора;
- створення нового користувача з роллю «Doctor»;
- вхід під обліковими даними створеного лікаря;
- перевірку валідності токену доступу через захищену кінцеву точку.

Для middleware авторизації були створені тести, які перевіряють поведінку функції у випадках:

- відсутності токену;
- наявності недійсного або простроченого токену;
- коректного токену з валідними правами доступу.

Усі тести пройшли успішно, що підтверджує правильність роботи механізмів автентифікації, видачі та перевірки токенів, а також рольового доступу до системи. Тестування дозволило переконатися в тому, що базовий функціонал

авторизації працює коректно і є захищеним від типових помилок на рівні контролерів.

5.2 Ручне тестування через Postman

Для тестування REST API кінцевих точок медичної системи використовується інструмент Postman (див.рис.5.1), який дозволяє здійснювати комплексну перевірку функціональності серверної частини. Необхідно забезпечити правильну роботу всіх API методів з чим і допомагає такий підхід.

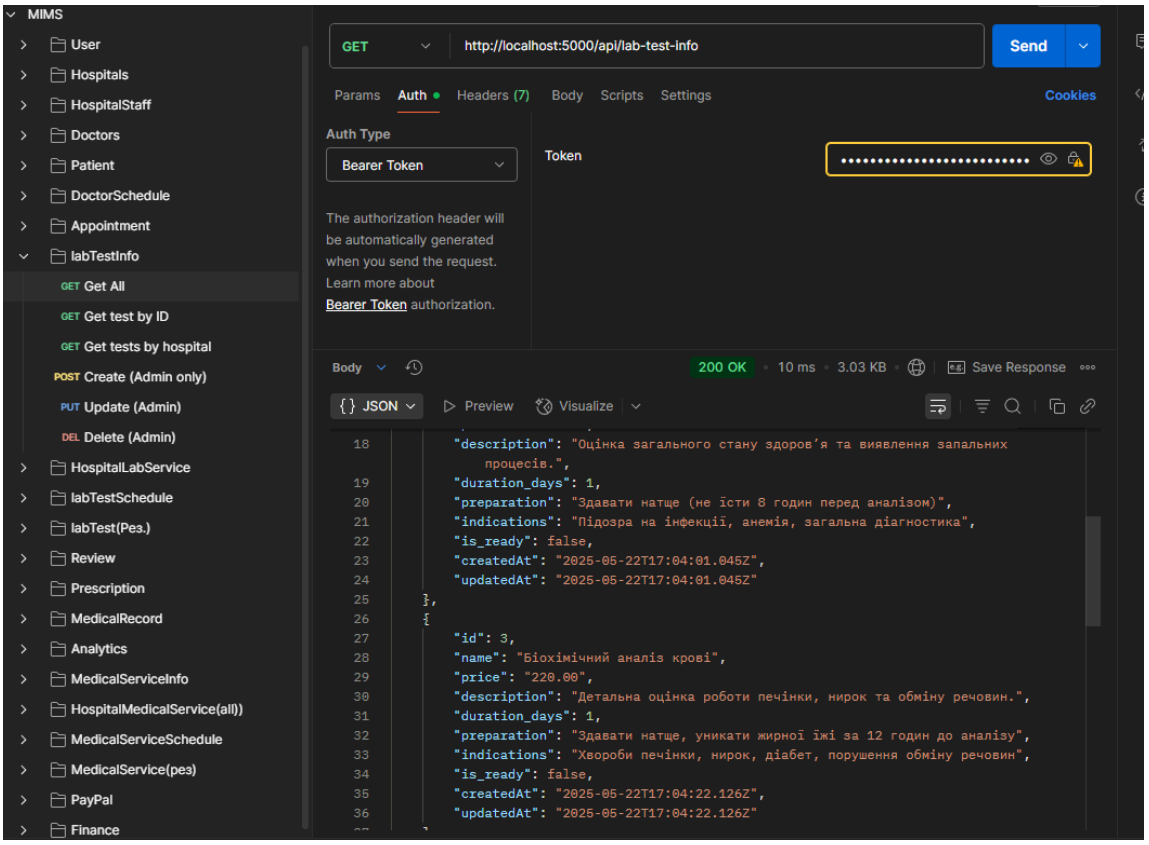

Рисунок 5.1 – Тестування API-запиту у Postman (рисунок виконано самостійно)

Postman надає зручний інтерфейс для створення та організації HTTP-запитів, що дозволяє тестувати різні сценарії використання системи. Через нього проводиться тестування таких ключових функцій як реєстрація та авторизація користувачів, управління медичними послугами, бронювання прийомів, інтеграція з платіжною системою PayPal, генерація медичних документів.

Перевагою використання цього інструменту є можливість створення колекцій тестів, які можуть бути збережені та використані повторно. Це дозволяє зробити стандарт процесу тестування та забезпечити послідовність перевірок при кожному оновленні системи. Postman підтримує автоматизацію тестів через скрипти, що дозволяє перевіряти структуру та зміст JSON-відповідей.

Інтеграція з платіжною системою PayPal потребує окремого тестування. Створюються тестові сценарії оплати, перевіряється обробка успішних платежів, відхилених транзакцій та помилок мережі. Було використано sandbox-середовище PayPal для безпечного тестування без реальних грошових переказів. Перевірялась також коректність оновлення статусу бронювання після успішної оплати.

Генерація медичних документів тестується через відповідні API endpoints з перевіркою правильності формування PDF-файлів, включення всієї необхідної інформації про пацієнта та процедуру. Було перевірено різні шаблони документів та коректність їх заповнення залежно від типу обраної медичної послуги.

5.3 Тестування безпеки

Для медичної програмної системи тестування безпеки є важливим моментом, бо система обробляє приватні дані пацієнтів, персональну інформацію та фінансові транзакції. Порушення безпеки в системі може призвести до серйозних наслідків, включаючи витік персональних медичних даних та порушення нормативних вимог.

Спочатку було застосовано тестування безпеки аутентифікації та авторизації включає перевірку міцності паролів для забезпечення їх достатньої складності. Тестується мінімальна довжина паролів (мінімум 8 символів), вимоги до використання різних типів символів (великі та малі літери, цифри, спеціальні символи) та обмеження на повторне використання попередніх паролів. Детально перевіряється безпека JSON Web Tokens, проходячи перевірку термінів дії та тестування поведінки системи при спробах використання змінених токенів.

Далі відбувається тестування рольового доступу для того, щоб користувачі могли отримати доступ лише до тих функцій та інформації, які відповідають їхнім

дозволам. Особлива увага приділяється перевірці того, що пацієнти можуть переглядати лише власні медичні записи, лікарі мають доступ тільки до призначених їм пацієнтів, а адміністратори не можуть переглядати конфіденційні медичні дані без відповідних прав.

Перевіряється правильність шифрування персональної інформації в базі даних, особисті дані пацієнтів (ПІБ, дата народження, адреса), медичні записи, діагнози. Тестується використання сучасних алгоритмів шифрування, правильність управління ключами шифрування через змінні оточення та захист ключів від несанкціонованого доступу. Перевіряється, що паролі зберігаються у хешованому вигляді з використанням bcrypt.

Комплексне тестування безпеки забезпечує високий рівень захисту приватної інформації, фінансової та персональних даних користувачів, що грає дуже важливу роль для довіри пацієнтів та дотримання всіх правових норм. Створюється належний механізм для безпечної роботи системи та захищає всіх користувачів від потенційних кібератак.

ВИСНОВКИ

У ході виконання бакалаврської кваліфікаційної роботи було реалізовано серверну частину програмної системи для медичних закладів «LifeLine». Основною метою цього проєкту було створення стабільної, надійної, безпечної системи з змогою інтегрування з іншими цифровими системами, які стануть у нагоді в процесі надання медичних послуг.

Робота над програмною системою «LifeLine» почалась з опрацювання предметної галузі, під час чого було знайдено недоліки вже існуючих рішень на українському ринку, таких як застарілість програмного забезпечення, недостатній рівень безпеки, складнощі у разі кризових ситуацій. Виходячи з цих даних було сформовано постановку задачі, визначено функціональні та нефункціональні вимоги, описано цільову аудиторію та визначено ключові напрями монетизації системи.

Для розробки програмної системи було обрано сучасні технології, такі як Node.js у поєднанні з Express.js як базу для обробки запитів та сформувані API. Для зберігання даних було обрано СУБД PostgreSQL з ORM-бібліотекою Sequelize. Такий вибір забезпечив безпеку персональних даних, гнучкість та можливість інтеграції з іншими підсистемами.

На наступному етапі було спроєктовано архітектуру системи «LifeLine» з підходом Model-View-Controller, також побудовано логічну модель бази даних. Значну увагу було приділено питанням безпеки, шифруванню даних, ролям доступу та захисту від типових атак. На етапі тестування програми було використано автоматизовані тести та ручне тестування з Postman для перевірки всіх запитів системи.Також було проведено первинне тестування безпеки з метою виявлення вразливостей у логіці автентифікації та контролю доступу.

У підсумку кваліфікаційної роботи було реалізовано серверну частину системи «LifeLine», яка відповідає всім поставленим завданням і має здатність інтегруватися з іншими цифровими сервісами. Отримані результати демонструють ефективність використаних технологій і рішень, а також потенціал системи до подальшого розвитку та впровадження в реальні умови роботи медичних установ.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. SEO медичних клінік: як уникнути проблем з алгоритмами Google [Електронний ресурс] - URL: https://webpromoexperts.net/ua/blog/seo-medicinskih-klinik-kak-izbezhat-problem-s-algoritmami-google/ (дата звернення: 13.04.2025);

2. Бадіков Р. Медичний фронт: загрози та виклики [Електронний ресурс] - URL: https://blogs.pravda.com.ua/authors/badikov/5ee39c2943d7d/ (дата звернення: 13.04.2025);

3. Дослідження інституту масової інформації: Інформатизація медицини в Україні [Електронний ресурс] - URL: https://ippi.org.ua/sites/default/files/2024-6.pdf (дата звернення: 14.04.2025);

4. Підключені до eHealth МІС [Електронний ресурс] - URL: https://ehealth.gov.ua/pidklyucheni-do-ehealth-mis/ (дата звернення: 14.04.2025);

5. Мікросервісна архітектура [Електронний ресурс] – URL: https://foxminded.ua/mikroservisna-arkhitektura/ (дата звернення: 13.04.2025);

6. Концепція інформатизації охорони здоров'я [Електронний ресурс] – URL: https://vikisoft.kiev.ua/без-рубрики/концепція-інформатизації-охорони-зд/ (дата звернення: 15.06.2025);

7. План відновлення системи охорони здоров'я України на 2022–2032 роки (проєкт) [Електронний ресурс] – URL: https://moz.gov.ua/uploads/ckeditor/Новини/21-07-2022-Draft-Ukraine%20HC%20System%20Recovery%20Plan-2022-2032_UKR.pdf (дата звернення: 15.06.2025);

8. Розбудова ЕСОЗ: її методологічна та технічна архітектура [Електронний ресурс] – URL: https://moz.gov.ua/uk/rozbudova-esoz-yiyi-metodologichna-ta-tehnichna-arhitektura (дата звернення: 16.04.2025);

9. За час повномасштабної війни Росія атакувала медичну систему України майже 2 тисячі разів — ВООЗ [Електронний ресурс] – URL: https://life.pravda.com.ua/health/za-chas-povnomasshtabnoji-viyni-rosiya-atakuvala-medichnu-sistemu-ukrajini-mayzhe-2-tisyachi-raziv-vooz-303251/ (дата

звернення:17.04.2025);

10. Великий гід по Helsi: як зареєструватись і чим може допомогти [Електронний ресурс] – URL: https://bf.diia.gov.ua/articles/velikij-gid-po-helsi-yak-zareyestruvatis-i-chim-mozhe-dopomogti (дата звернення: 17.04.2025);

11. Медичний онлайн-сервіс Helsi [Електронний ресурс] - URL: https://helsi.me/ (дата звернення: 17.04.2025);

12. Що потрібно знати про Health24 [Електронний ресурс] – URL: https://blog.h24.ua/uk/shho-potribno-znaty-pro-health24/ (дата звернення: 18.04.2025);

13. Медичний сервіс для пацієнтів і лікарів Health24 [Електронний ресурс]] - URL: https://h24.ua/ (дата звернення: 18.04.2025);

14. GitHub-репозиторій: https://github.com/NureChernovaAnastasiia/2025_B_PI_PZPI-21-6_Chernova_A_M