МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

МАТЕМАТИЧНИЙ ФАКУЛЬТЕТ
Кафедра комп'ютерних наук

КВАЛІФІКАЦІЙНА РОБОТА БАКАЛАВРА

на тему: **«РОЗРОБКА ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ УПРАВЛІННЯ МЕДИЧНОЮ БАЗОЮ ДАНИХ З ВИКОРИСТАННЯМ ФРЕЙМВОРКУ LARAVEL»**

Виконав: студент 4 курсу, групи 6.1221
спеціальності 122 Комп'ютерні науки
(шифр і назва спеціальності)
освітньої програми Комп'ютерні науки
(назва освітньої програми)
А.А. Кроль
(ініціали та прізвище)
Керівник доцент кафедри комп'ютерних наук, PhD, Калюжняк А. В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)
Рецензент професор кафедри програмної інженерії, доцент, к.ф.-м.н., Кудін О.В.
(посада, вчене звання, науковий ступінь, прізвище та ініціали)

Запоріжжя 2025

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

ЗАПОРІЗЬКИЙ НАЦІОНАЛЬНИЙ УНІВЕРСИТЕТ

Факультет математичний

Кафедра комп'ютерних наук

Рівень вищої освіти бакалавр

Спеціальність 122 Комп'ютерні науки
(шифр і назва)

Освітня програма Комп'ютерні науки

ЗАТВЕРДЖУЮ
Завідувач кафедри комп'ютерних наук,
д.т.н., професор

_____ Шило Г.М.
(підпис)

" ____ " _____ 2025 р.

З А В Д А Н Н Я

НА КВАЛІФІКАЦІЙНУ РОБОТУ СТУДЕНТОВІ

Кролю Артему Андрійовичу
(прізвище, ім'я та по-батькові)

1. Тема роботи Розробка інформаційної системи для управління медичною базою даних з використанням фреймворку Laravel

керівник роботи Калюжняк Анастасія Вікторівна, PhD
(прізвище, ім'я та по-батькові, науковий ступінь, вчене звання)

затверджені наказом ЗНУ від « 20 » грудня 2024 року № 2001-с

2. Строк подання студентом роботи 23.05.2025

3. Вихідні дані до роботи 1. Постановка задачі.
2. Перелік літератури.

4. Зміст розрахунково-пояснювальної записки (перелік питань, які потрібно розробити)
1. Постановка задачі.
2. Основні теоретичні відомості.
3. Система управління базою даних.
4. Розробка системи.

5. Перелік графічного матеріалу (з точним зазначенням обов'язкових креслень)
презентація

6. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		завдання видав	завдання прийняв

7. Дата видачі завдання 20.12.2024

К А Л Е Н Д А Р Н И Й П Л А Н

№	Назва етапів кваліфікаційної роботи	Строк виконання етапів роботи	Примітка
1.	Вивчення рекомендованої літератури	05.01.2025	
2.	Постановка задачі	10.01.2025	
3.	Аналіз предметної області	15.01.2025	
4.	Дослідження існуючих МІС	20.01.2025	
5.	Обґрунтування вибору технічних засобів	25.01.2025	
6.	Розробка вимог до ПЗ	31.01.2025	
7.	Проектування БД	10.02.2025	
8.	Верстка інтерфейсу	05.03.2025	
9.	Розробка backend на Laravel	25.03.2025	
10.	Написання пояснювальної записки	12.05.2025	
11.	Подання роботи на перевірку нормоконтролю	23.05.2025	
12.	Проходження нормоконтролю	27.05.2025	
13.	Захист дипломної роботи	18.06.2025	

Студент _____ А.А. Кроль
(підпис) (ініціали та прізвище)

Керівник роботи _____ А.В. Калюжняк
(підпис) (ініціали та прізвище)

Нормоконтроль пройдено

Нормоконтролер _____ О.Г.Спиця
(підпис) (ініціали та прізвище)

РЕФЕРАТ

Кваліфікаційна робота бакалавра «Розробка інформаційної системи для управління медичною базою даних з використанням фреймворку Laravel»: 56 с., 13 рис., 13 табл., 36 джерел.

БАЗА ДАНИХ, ВЕБ-ДОДАТОК, ВЕБ-ІНТЕРФЕЙС, ЕЛЕКТРОННА СИСТЕМА ОХОРОНИ ЗДОРОВ'Я, ІНФОРМАЦІЙНА СИСТЕМА, МЕДИЦИНА, МЕДИЧНА ІНФОРМАЦІЙНА СИСТЕМА, ФРЕЙМВОРК, FIGMA, HTML, LARAVEL, MYSQL, PHP.

Об'єкт дослідження: медична інформаційна система.

Предмет дослідження: інформаційна система для управління медичною базою даних.

Мета роботи: – створити інформаційну систему, яка забезпечить ефективну обробку медичних даних та оптимальне управління процесами, пов'язаними з наданням медичних послуг, автоматизувати ведення документації.

Методи дослідження: дослідження існуючих інформаційних систем, аналіз предметної області, моделювання бізнес-процесів, проектування архітектури програмного забезпечення, використання об'єктно-орієнтованого програмування та компонентного підходу, реалізація інформаційної системи із застосуванням фреймворку Laravel.

В даній роботі проведено розробку WEB додатку для управління медичною базою даних на мові PHP з використанням фреймворку Laravel.

Для цього було досліджено сучасний стан та особливості застосування медичних інформаційних систем в Україні і перспективи підключення їх до центральної бази даних, створено WEB додаток і його дизайн для реалізації поставленої задачі, зроблені висновки за результатами проведеної роботи.

В результаті проведеної роботи було створено інформаційну систему для управління медичною базою даних, яка підвищить рівень надання медичних

послуг, відпрацьовано процес підключення до бази даних, формування запитів SQL, їх передачу і отримання результатів з наступним їх використанням.

Система дозволить швидко й ефективно налагодити електронний документообіг у медичних закладах, гнучко вибудовувати роботу з пацієнтами, а також підготувати заклади охорони здоров’я до підключення до централізованої електронної системи охорони здоров’я.

SUMMARY

Bachelor's Qualifying Thesis «Development of an Information System for Managing a Medical Database Using the Laravel Framework»: 56 pages, 13 figures, 13 tables, 36 references.

DATABASE, WEB APPLICATION, WEB INTERFACE, ELECTRONIC HEALTH CARE SYSTEM, INFORMATION SYSTEM, MEDICINE, MEDICAL INFORMATION SYSTEM, FRAMEWORK, FIGMA, HTML, LARAVEL, MYSQL, PHP.

The object of the study is medical information system.

The aim of the study is improving the working conditions of doctors.

The method of research is development of an information system using the Laravel framework.

In this work, the development of a WEB application for managing a medical database in the PHP language using the Laravel framework is carried out.

For this purpose, the current state and features of the use of medical information systems in Ukraine and the prospects for connecting them to the central database were studied, a WEB application and its design for the implementation of the task were created, conclusions were drawn based on the results of the work carried out.

As a result of the work carried out, an information system was created to manage the medical database, which will increase the level of medical services, the process of connecting to the database, forming SQL queries, transferring them and obtaining results with their subsequent use was worked out.

The system will allow you to quickly and efficiently establish electronic document management in medical institutions, flexibly build work with patients, as well as prepare healthcare institutions for connection to the centralized electronic health care system.

ЗМІСТ

Завдання на кваліфікаційну роботу 2
Реферат 4
Summary 6
Скорочення та умовні познаки 9
Вступ 10
1 Теоритичний аналіз особливостей розробки інформаційної системи для управління медичною базою даних 12
1.1 Аналіз предметної області 12
1.2 Дослідження існуючих систем управління медичних баз даних 15
1.3 Обґрунтування вибору програмних та технічних засобів для розробки системи управління медичною базою даних 20
1.4 Висновки до 1 розділу 21
2 Розробка ресурсу для управління медичною базою даних 22
2.1 Розробка вимог до програмного забезпечення 22
2.1.1 Варіанти використання програмного забезпечення 22
2.1.2 Розроблення функціональних вимог 24
2.1.3 Розроблення нефункціональних вимог 25
2.2 Розробка структури бази даних 26
2.3 Розробка дизайну системи для управління медичною базою даних 31
2.4 Висновки до 2 розділу 35
3 Реалізація WEB ресурсу для управління медичною базою даних 36
3.1 Вибір мови програмування та фреймворку для реалізації інформаційної системи для управління медичною базою даних 36
3.2 Встановлення Laravel 37
3.3 Створення бази даних системи 38
3.4 Створення моделей 41
3.5 Реалізація дизайну інтерфейсу системи 42

3.6 Реалізація основних функцій інформаційної системи для управління медичною базою даних 44
3.7 Розгортання та супровід системи 49
3.8 Висновки до 3 розділу 51
Висновки 53
Перелік посилань 54

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

API	–	Application programming interface, прикладний програмний Інтерфейс
IDE	–	Integrated Development Environment – інтегроване середовище розробки.
SDK	–	Software development kit
IT	–	Інформаційні технології
ER	–	Entity-Relation diagram
ОС	–	Операційна система.
БД	–	База даних.
JDK	–	Java development kit
ПЗ	–	Програмне забезпечення
ШІ	–	штучний інтелект
CMS	–	Content Management System – Система управління контентом
СУБД	–	система управління базами даних
VS Code	–	Visual Studio Code
MVC	–	Model-View-Controller
ORM	–	Object-Relational Mapping
CRUD	–	Create, Read, Update, Delete
MIC	–	Медична інформаційна система
ЦБД	–	центральна база даних
HIS	–	Hospital Information System

ВСТУП

Ще кілька років тому ніхто не замислювався про автоматизацію системи охорони здоров'я, всі документи виписувались вручну на папері, що впливало швидкість роботи, позначилося на якості обслуговування пацієнтів, приводило до лікарських помилок та великих витрат часу на заповнення документів. Тому зараз в системі охорони здоров'я Україні активно проводиться процес інформатизації і створення єдиного медичного інформаційного простору. Оскільки термін «медична інформаційна система» можна тлумачити по різному, то в даній роботі його будемо використовувати для конкретного випадку: медична інформаційна система для організації робочого місця лікаря (медсестри).

Медична інформаційна система це складна автоматизована система, яка об'єднує електронний документообіг медичного закладу як частину загального лікувального процесу, і тим самим прискорює час і якість медичного рішення і медичної допомоги. Саме тому питання розробки інформаційної системи для управління медичною базою даних є дуже важливим завданням сучасної медицини.

В охороні здоров'я використання інформаційних технологій повинно перш за все, вирішити такі завдання:

- створення електронної черги і запису на прийом до фахівців;
- моніторинг та управління якістю надання медичної допомоги;
- скорочення часу обстеження та лікування пацієнтів;
- створення єдиної інформаційної мережі, як всередині клініки, так і для взаємодії з аптеками та іншими установами;
- підвищення ефективності управлінських рішень;
- надання консультативної медичної підтримки фахівцям;
- моніторинг фізіологічних показників пацієнта;
- автоматизація різних медичних документів;
- створення та ведення бази даних пацієнтів з анамнезом лікування;
- надання дистанційної підтримки під час бойових дій.

Впровадження інформаційних технологій у сферу медицини та охорони здоров’я значно економить час медичного персоналу, а це означає, що працівники медичних закладів зможуть більше часу приділяють пацієнтам, зростає доступність медичних послуг.

Важливими показниками при підборі засобів розробки є технологічні можливості програми, її зручність у використанні, наявність фахівців на ринку, перспективи розвитку, надійність компанії виробника, цінові показники. Для одних підприємств більш важливим завданням автоматизації є збір і формування статистичних даних, для інших – зниження навантаження на персонал, фінансові питання чи інше. Однією з проблем впровадження інформаційних технологій у медицину є також те, що рівень оснащення системи охорони здоров’я сучасними інформаційно-комунікаційними технологіями є вкрай неоднорідним, немає уніфікації використовуваних програмних та апаратних платформ, тому дуже важлива інтеграція з іншими вже працюючими системами. З іншого боку, саме така різноманітність показує, що існуючі інформаційні системи для управління медичною базою даних ще далекі від ідеалу і потребують подальшого дослідження, чому і присвячена дана робота.

В процесі дослідження потрібно вирішити наступні завдання:

- дослідити і проаналізувати існуючі методи і підходи для створення інформаційних систем управління медичними базами даних;
- обрати програмні та технічні засоби для розробки інформаційної системи;
- розробити структуру бази даних;
- розробити структуру меню та концепт інтерфейсу WEB ресурсу;
- розробити дизайн сайту;
- обрати мову програмування та фреймворк для розробки;
- виконати верстку сайту за розробленим дизайном;
- реалізувати основний функціонал сайту.

1 ТЕОРИТИЧНИЙ АНАЛІЗ ОСОБЛИВОСТЕЙ РОЗРОБКИ ІНФОРМАЦІЙНОЇ СИСТЕМИ ДЛЯ УПРАВЛІННЯ МЕДИЧНОЮ БАЗОЮ ДАНИХ

1.1 Аналіз предметної області

В останні кілька років відбулися значні позитивні зміни в напрямку цифровізації медицини. На сайті МОЗ України [1] викладено офіційні підходи до розробки Медичних інформаційних систем [2] як складової частини загальної Електронної системи охорони здоров'я України. Затверджено технічні вимоги для підключення медичної інформаційної системи до центральної бази даних, що дозволить привести всі ці системи до одного стандарту і забезпечить їх взаємодію. [3] [4] [5]. Згідно постанови КМУ від 25.04.2018 № 411 всі заклади охорони здоров'я незалежно від форми власності мають працювати в електронній системі охорони здоров'я eHealth (ЕСОЗ). [6]

Дана постанова дає таке визначення електронній медичній інформаційній системі: «До складу електронної системи охорони здоров'я входять центральна база даних та електронні медичні інформаційні системи, між якими забезпечено автоматизований обмін інформацією, даними та документами через відкритий програмний інтерфейс (API).» Власником центральної бази даних є держава у особі НСЗУ. [6]

Також Урядом України схвалено п'ятирічну Концепцію розвитку електронної охорони здоров'я [7], розпочато розроблення ключових процесів та реєстрів, необхідних для існування інформаційного середовища у сфері охорони здоров'я, почав формуватися ринок виробників спеціалізованого програмного забезпечення. На сьогодні електронна система охорони здоров'я забезпечує укладення електронних декларацій із сімейними лікарями, виписку електронних рецептів за програмою "Доступні ліки" та електронних направлень, ведення електронної медичної картки тощо. Разом з тим існує ще питань щодо розвитку

електронної системи охорони здоров’я, пов’язаних з паралельним існуванням паперових та електронних форм, що нерідко суперечать один одному [7].

Шляхи вирішення цих питань лежать в чіткому розподілі функцій держави і функцій закладів охорони здоров’я. До функцій держави відноситься в першу чергу впровадження централізованих інформаційно-комунікаційних систем, центральної бази даних, забезпечення обміну даними, сумісності інформаційно-комунікаційних систем за допомогою технічного регулювання та стандартизації.

У функції бізнес-спільноти входить створення інформаційно-комунікаційних систем для кінцевих користувачів, що забезпечать якнайкращий рівень надання медичних послуг.

Як правило, кожна МІС складається з блоків, що відповідають за автоматизацію різних складових діяльності медичної установи. Серед них:

- реєстратура та електронні медичні картки пацієнтів;
- дані медичних досліджень;
- робочі місця лікаря і медсестри;
- розподіл ресурсів установи, їхній розклад;
- управління фінансами й облік;
- адміністративна інформація і засоби комунікації співробітників;
- лікарські призначення, журнал призначень;
- стандарти надання медичної допомоги та багато іншого.

Комплексні медичні інформаційні системи, як правило, складаються з модулів. Це дозволяє зібрати і налаштувати МІС у потрібній конфігурації для установ різного типу і забезпечити необхідний функціонал із можливістю подальшого додавання/видалення модулів.

Окремі компоненти МІС можна об’єднати у кілька великих груп:

- аналітичні й управлінські компоненти. Модулі та засоби ведення управлінського обліку, інструменти аналізу якості та ефективності медичних послуг. Ці складові МІС дозволяють проаналізувати стан вашої медичної організації, виявити проблемні місця й оптимізувати бізнес-

процеси. Результати аналізу можна вивести на екран у вигляді графіків чи таблиць.

- медичні компоненти. Усі модулі, пов’язані з реєстрацією пацієнтів, ведення реєстру електронних медичних карт, облік лікарняних листів, ведення протоколів лікування, інформаційний супровід лікування пацієнтів у різних типах установ (амбулаторія, поліклініка, стаціонар), медична статистика й аналітика, історія хвороби та багато іншого.
- фінансово-економічні компоненти. До них відносяться інструменти ведення обліку медикаментів, управління запасами, розрахунок собівартості лікування і тарифів на надання медичних послуг, розрахунок надбавок лікарям, інструменти проведення економічного аналізу діяльності організації і т.д.
- компоненти обміну даними. Ведення уніфікованих реєстрів, каталогів і довідників, обмін даними з іншими системами, обробка отриманих даних.
- загально технічні компоненти. Контроль доступу користувачів і захист бази даних, підтримка можливостей інтеграції з іншими системами і програмами.

Якщо медична інформаційна система була вибрана вдало, її впровадження позитивно відобразиться на роботі організації, але все буде залежати від її можливостей та специфіки конкретного закладу. В першу чергу це позбавляє від заповнення зайвих паперів: лікарі і персонал, що мають доступ до карти пацієнта, можуть ознайомитися з історією хвороби, ходом лікування, результатами досліджень і іншою інформацією з єдиної бази даних. Автоматизований документообіг дозволяє зменшити кількість паперової роботи, лікарям набагато простіше ставити діагноз, знижується ризик втрати важливих даних, з’являється можливість проконсультуватися в реальному часі з колегами та іншими фахівцями стосовно постановки діагнозу (особливо, в екстреній ситуації), вести дистанційний моніторинг стану хворих, уникнути черги з системою онлайн-запис.

Для кожного конкретного закладу питання вибору і використання МІС потрібно вирішувати окремо, їхній вибір залежить від типу закладу та повсякденних завдань, котрі належить оптимізувати.

В загальному випадку є три способи вирішити питання:

- обираємо готове програмне рішення і працюємо на базовому функціоналі для виконання всіх поточних завдань;
- інтеграція готового серверного рішення з подальшим доопрацюванням його під конкретні потреби організації. Набагато простіше найняти розробників і переписати/дописати функціонал, ніж перебудовувати роботу всієї організації під конкретне ПЗ.
- індивідуально замовити систему з нуля чи доповнити вже існуючу систему новими можливостями.

В основному, індивідуальна розробка МІС – це окремі випадки для великих комерційних клінік. Більшість медзакладів використовують готові програмні рішення, враховуючи, що українське законодавство дозволяє використання тільки тих систем, що пройшли перевірку МОЗ і були підключені до системи eHealth.[8]

З цієї точки зору логічним вибором буде в процесі виконання дипломного проекту розробити програму, яка дозволить охопити невелику групу завдань закладу і створить каркас для майбутньої великої інформаційної системи.

1.2 Дослідження існуючих систем управління медичних баз даних

Історія інформаційних систем у світі починається з 1970 1980х років, коли були запроваджені перші медичні інформаційні системи, які надалі стали вміщувати і детальну інформацію про стан здоров’я пацієнтів. У 2000х почали з’являтися об’єднані національні електронні медичні інформаційні системи.

В Україні розрізнені спроби автоматизувати облік медичних послуг отримали новий поштовх з початком проведення у 2016 році медичної реформи, а з 2018 року почав діяти закон про реформу системи охорони здоров’я в Україні. В рамках

впровадження реформи було створено було створено електронну інформаційно-телекомунікаційна систему охорони здоров'я eHealth або eZdorovya [9], яка забезпечує автоматизацію ведення обліку медичних послуг та управління медичною інформацією в електронному вигляді. Її розділили на два компоненти: центральну базу даних (ЦБД), якою керує держава і медичні інформаційні системи, що підтримуються комерційними розробниками програмного забезпечення.

Адміністратором eHealth системи є створене з цією метою державне підприємство «Електронне здоров'я». Адміністратор вимагає від розробників МІС строго дотримуватися вимог до надійності, безпеки і конфіденційності даних, якими їх системи обмінюватися з центральним компонентом. Система також пройшла сертифікацію Державної служби спеціального зв'язку та захисту інформації України.

Загальна структура системи eHealth показана на рисунку 1.1.



Рисунок 1.1 – Загальна структура системи eHealth

Медичні заклади зможуть обирати будь-яку медичну інформаційну систему з-поміж тих, які пройшли перевірку і підключені до центральної бази «eHealth».

Для підключення до системи eHealth закладам охорони здоров'я необхідно укласти договір з Національною службою здоров'я України (НСЗУ), вибрати МІС, яка підключена до ЦБД та володіє достатнім функціоналом для роботи в ЕСОЗ, і укласти договір на обслуговування з компанією розробником МІС. Повний перелік МІС, що пройшли перевірку і підключені до центральної бази даних електронної системи охорони здоров'я (ЦБД eHealth) наведений на офіційному сайті eHealth [10]. Перелік підключених систем постійно зростає і на даний момент в списку вже більше 40 позицій, причому більшість з них мають власні сайти, де можна ознайомитись з їх можливостями. Розглянемо деякі з них.

Helsi [11] – одна з найбільших в Україні і покриває велику долю ринку. За офіційними даними Національної служби здоров'я України через систему Helsi укладено приблизно 44% всіх декларацій між лікарями та пацієнтами. Система розроблена у форматі веб-застосунку, що складається з трьох частин: helsi.me частина системи призначена для пацієнтів, helsi.pro для лікарів і персоналу медичного закладу, reform.helsi.me для адміністрації закладу. Серед інших особливостей даної системи є наявність великої кількісті різноманітного функціоналу, наприклад особистий кабінет, реєстрація дітей у системі, онлайн замовлення ліків, запис у електронну чергу, можливість взяти участь у державних медичних програмах, укладення декларацій з лікарем і багато іншого. Серед недоліків можна виділити відсутність можливостей для інтеграції з іншими інформаційними системами, що не дає пацієнтам переносити електронні медичні картки з однієї платформи на іншу, порівняно низька швидкість роботи, а також те, система занадто громіздка для малих установ, а інтерфейс у всіх трьох підсистемах різний, тому перехід з reform.helsi.me на helsi.pro викликає у лікарів труднощі.

Doctor Eleks [12] друга по розповсюдженості медична інформаційна система, яка з'явилась ще у 2006 році і є однією з перших великих медичних інформаційних систем на території України. Основним недоліком є те, що основний модуль системи для робочого місця лікаря розповсюджується у форматі десктопного застосунку (рисунок 1.2) під операційну систему Windows, що робить неможливим його використання на інших платформах, а також досить висока вартість впровадження. Система не має функціоналу для пацієнтів, але швидка, забезпечує

зручну роботу лікарів і можливості підключення великої кількості працівників. Має дуже потужний функціонал, наявність комунікаційного сервера для обміну даними у форматі HL7 із суміжними ІС, зовнішніми лабораторіями, страховими компаніями. Присутній інтеграція з Toshiba УЗД, підтримується імпорт DICOM-зображень, підключення DICOM-сумісного обладнання та багато іншого.

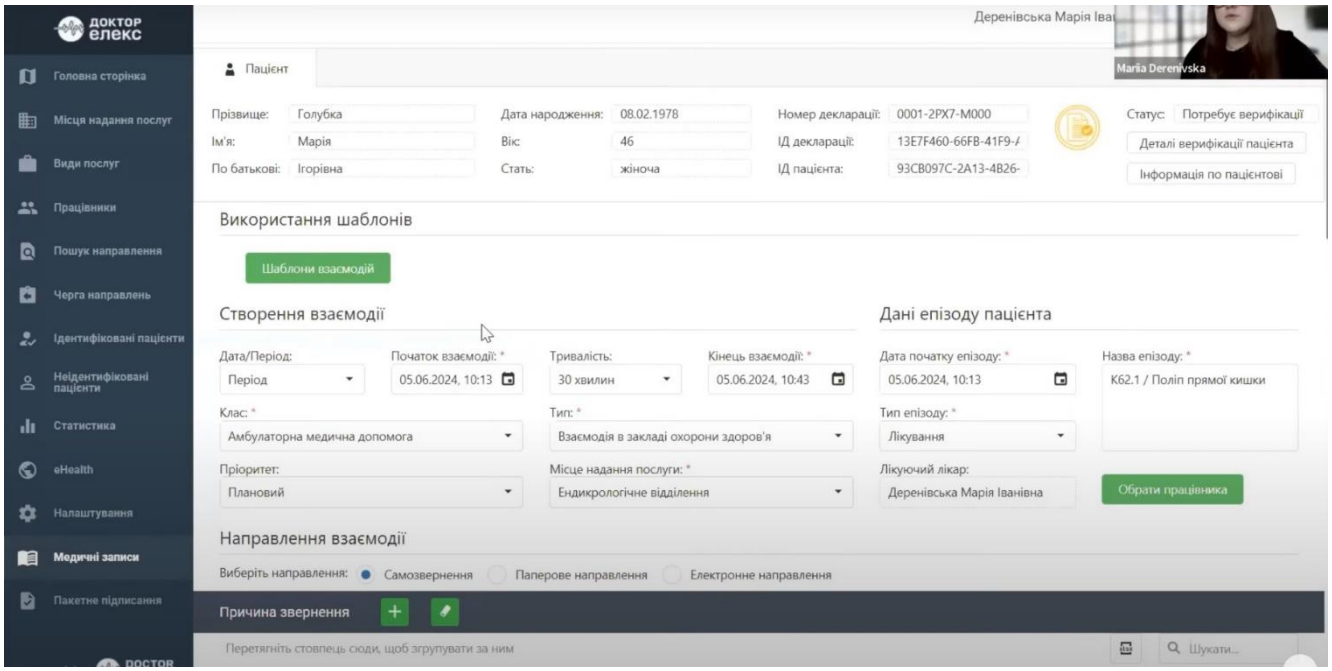


Рисунок 1.2 – Інтерфейс МІС «Doctor Eleks»

ASKEP [13] перевагами цієї системи є велика кількість функціоналу для пацієнтів, лікарів, а також для адміністрації медичного закладу. Система надає пацієнтам доступ до особистого кабінету з історією хвороби, рецептами і іншою особистою інформацією. Лікарям надається декілька модулів з усією інформацією про огляд пацієнта та історію хвороби. Для адміністрації система надає внутрішню статистику, звітність та інші потрібні функції.

EMCiMED [14] українська медична інформаційна система для медичних установ, приватних клінік і лабораторій, України. Складається з модулів, що легко можна збирати в потрібній конфігурації для кожного окремого закладу. За необхідності вони можуть поставлятися в складі пакетів EMCiMED-Поліклініка і EMCiMED-Стаціонар. Система має гнучке налаштування і потужну функціональну

складову, добре захищена завдяки використанню USB-брелків та шифрування всієї інформації, підтримує інтеграцію з іншими продуктами. (рисунок 1.3).

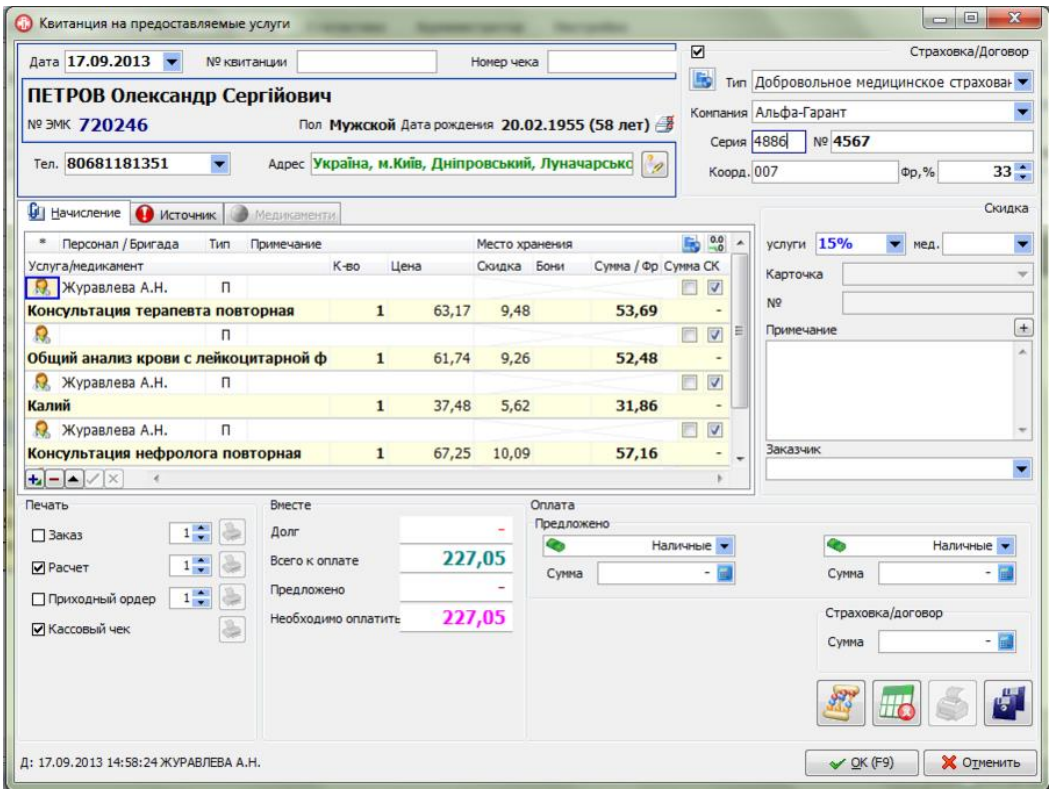


Рисунок 1.3 – Інтерфейс МІС «EMCImed»

Всі ці системи мають багато спільного і призначені в першу чергу для підвищення якості діагностики і надання послуг шляхом досконалої обробки медичної інформації, а також ліквідації трудомістких процесів ручної обробки й аналізу медичних даних. В той же час вони виконують і інші функції: задачі кадрові, навчальні, матеріально-технічного постачання, статистичні звіти, автоматизація лабораторних досліджень, накопичення відомостей про кожного хворого для подальшого аналізу матеріалу, обробку інформації відповідним математичним забезпеченням, організацію роботи аптек, удосконалювання процесів керування і планування і інше. Саме тому на сучасному етапі відбувається перехід від окремих інформаційних систем до інформаційних середовищ, що інтегрують в рамках єдиного технологічного процесу МІС різних класів, що пронизані єдиним інформаційним потоком.

1.3 Обґрунтування вибору програмних та технічних засобів для розробки системи управління медичною базою даних

Далі потрібно проаналізувати те, які саме можливості для створення системи управління медичною базою даних повинна забезпечити програма, потім сформулювати необхідні вимоги до неї і вибрати, якими саме засобами це можна забезпечити.

В даному випадку програма повинна мати вільний багатокористувацький доступ з мережі інтернет, можливість повноцінної реєстрації і авторизації а також зберігати і обробляти великі об’єми інформації. Це звичайні вимоги для створення більшості більш-менш великих сайтів. Відповідно, і інструменти краще брати оптимальні і широко поширені.

Найпростіше сайт можна створити, використавши онлайн-конструктори сайтів, які ідеально підходять для початківців, оскільки не потребують знань програмування або верстки, до того ж, пропонують готові шаблони дизайну. Але всі вони мають порівняно мало можливостей для налаштування функціональності, тому не підходять для великих і складних сайтів.

Дещо кращим варіантом є використання Content Management System (CMS), які мають більш гнучкі можливості і дозволяють створювати сайти з великою кількістю сторінок і функціоналу. Однак для роботи з ними потрібні хоча б базові знання HTML і CSS, крім того для роботи сайту потрібен окремий хостинг, хоча в багатьох випадках CMS провайдери надають для таких сайтів безкоштовний хостинг і URL адресу на своєму сервері, найчастіше з розміщенням реклами [15].

Недоліків у сайтів на CMS і конструкторах дуже багато, але основним недоліком є те, що у гонитві за універсальністю розробники закладають багато функцій, які для одного проєкту не потрібні, внаслідок чого значно знижується швидкість роботи. Крім того, при потребі відійти від шаблонних рішень, розібратися в величезній кількості коду дуже важко, а іноді при зміні програмного коду сайт може перестати працювати зовсім.

Зважаючи на проведений порівняльний аналіз, пропонується розробка системи у вигляді Веб-додатку з серверною і клієнтською частинами з використанням однієї з популярних мов програмування і її фреймворків. Незалежність від цільової платформи, легкість встановлення, підтримки та обслуговування, можливість доступу користувачів із будь-якого пристрою з підключенням до мережі Інтернет стали факторами вибору даного способу. Це надає ширші можливості в реалізації інтерфейсу користувача та забезпеченні спільного доступу групи користувачів.

1.4 Висновки до 1 розділу

В даному розділі розглянуто популярні нині Медичні інформаційні системи різних класів, що пройшли перевірку і підключені до центральної бази даних електронної системи охорони здоров’я (ЦБД eHealth). Перелік таких систем наведений на офіційному сайті eHealth [10] і в даний момент займає вже більше 40 позицій, причому більшість з них мають власні сайти, де можна ознайомитись з їх можливостями. Розглянуті сайти мають різне оформлення та наповнення, в одних більш переважає зручність використання, ніж зовнішній вигляд, в інших, поряд з зручним інтерфейсом, присутній і приємний зовнішній вигляд та велика різноманітність модулів для ефективного користування сайтом. Також сформовано вимоги до інформаційної системи, що розробляється, визначено перелік апаратних та програмних засобів для ведення розробки та подальшої підтримки.

Розглянуто велику кількість систем, але не всі вони призначені саме для управління медичними базами даних, що призводить до вузької спеціалізації в їх використанні, тому основною задач5ею, що необхідно вирішити при створенні автоматизованої системи управління медичними базами даних є індивідуальний підхід до створення бази даних і розробки інтерфейсу майбутньої програми, уважного відбору необхідних функцій і шляхів їх реалізації.

2 РОЗРОБКА РЕСУРСУ ДЛЯ УПРАВЛІННЯ МЕДИЧНОЮ БАЗОЮ ДАНИХ

2.1 Розробка вимог до програмного забезпечення

2.1.1 Варіанти використання програмного забезпечення

Для системи управління медичною базою даних головною функцією є можливість лікаря ознайомитись з історією хвороби пацієнта, попередніми діагнозами і призначеним лікуванням, а також можливість добавити нові записи. Само собою, повинні бути функції реєстрації і авторизації, а також і допоміжні можливості: додати коментар, пошук і подібне. На основі цієї інформації будуємо діаграму варіантів використання системи (рисунок 2.1).



Рисунок 2.1 – Діаграма варіантів використання (діаграма прецедентів)

Користуючись діаграмою варіантів (рисунок 2.1), детально розробляємо ці варіанти. В таблицях 2.1 – 2.2 наведено варіанти використання для реєстрації користувача і перегляду головної сторінки.

Таблиця 2.1 – Варіант використання UC-1

Use case name	Реєстрація користувача
Use case ID	UC-01
Goals	Реєстрація нового користувача в системі
Actors	Гість (незареєстрований користувач)
Trigger	Користувач бажає зареєструватися
Pre-conditions	-
Flow of Events	Користувач заповнює поля форми, натискає кнопку реєстрації і перенаправляється на сторінку входу.
Extension	В випадку введення не коректних даних, відповідне поле підсвічується червоним надписом.
Post-Condition	Створення сторінки користувача, перехід на сторінку входу

Таблиця 2.2 – Варіант використання UC-2

Use case name	Перегляд головної сторінки
Use case ID	UC-02
Goals	Вибрати потрібний режим роботи чи знайти історію хвороби
Actors	-
Trigger	Користувач обрає один з варіантів, користується фільтром чи пошуком і натискає відповідну кнопку
Pre-conditions	-
Flow of Events	Користувач переходить в обраний режим роботи
Extension	-
Post-Condition	Перехід на головну сторінку

Аналогічно створюємо варіанти використання і для інших ситуацій: авторизація, пошук пацієнта в БД, користування фільтром і так далі.

2.1.2 Розроблення функціональних вимог

Наступним кроком є розроблення функціональних вимог до програмного забезпечення. Програма розділена на модулі, кожен з яких має свій певний набір функцій. В таблиці 2.3 наведено загальну модель вимог, а в таблиці 2.4 наведено опис функціональних вимог до перегляду історії хвороби. Для інших ситуацій теж аналогічно створюємо описи функціональних вимог.

Таблиця 2.3 – Загальна модель вимог

№	Назва	ID Вимоги	Пріоритети	Ризики
1	Перегляд історії хвороби.	FR-1	Низький	Низький
2	Система авторизації користувача.	FR-2	Високий	Високий
2.1	Реєстрація користувача.	FR-3	Високий	Високий
2.2	Авторизація користувача.	FR-4	Високий	Високий
2.3	Вихід із акаунту.	FR-5	Середній	Низький
3	Новий запис до історії хвороби	FR-6	Високий	Високий
4	Створення картки	FR-7	Високий	Високий
5	Редагування картки	FR-8	Високий	Високий

Таблиця 2.4 – Функціональна вимога FR-1

Назва	Перегляд історії хвороби
Опис	повинна бути можливість переглянути історію хвороби пацієнта

Потім розробляємо загальну матрицю трасування вимог (таблиця 2.5).

Таблиця 2.5 – Матриця трасування вимог

	UC-1	UC-2	UC-3	UC-4	UC-5	UC-6	UC-7	UC-8
FR-1								
FR-2		+						
FR-3	+							
FR-4		+						
FR-5		+	+	+	+	+	+	+
FR-6			+	+	+	+		
FR-7			+		+	+	+	
FR-8			+				+	+

2.1.3 Розроблення нефункціональних вимог

Застосунок повинен мати зручний та простий у використанні інтерфейс для користувача, стабільно працюючий при будь яких умовах. Екран має бути оптимізованим до розмірів екрана користувача, повинна бути підтримка всіх основних типів браузерів і т.д.. В таблицях 2.6 2.7 наведено зразки описів нефункціональних вимог до програмного забезпечення.

Таблиця 2.6. – Нефункціональна вимога NFR-1

Назва	Надійність
Опис	Зміна розміру програми повинна відбуватися без суттєвих затримок.

Таблиця 2.7. – Нефункціональна вимога NFR-2

Назва	Продуктивність
Опис	Час від команди до відображення результатів повинен бути менше 5 секунд.

2.2 Розробка структури бази даних

Для розробки бази даних на першому етапі необхідно визначитися з переліком інформації, яка повинна бути присутня в інформаційній системі, продумати формат її зберігання і типи даних, організацію процесу лікування, процедуру введення і редагування даних.

На цьому етапі правильними діями будуть:

- поспілкуватись з лікарями і з'ясувати, які саме інформація і в якому вигляді потрібна в програмі;
- скласти список всіх необхідних даних і звести їх в таблицю.

В даному випадку є кілька зв'язаних між собою масивів відносно однотипних даних: списки лікарів і інших працівників, список пацієнтів, перелік ліків і можливих діагнозів і таке інше. Така структура інформації якнайкраще відповідає структурі реляційних баз даних, які широко розповсюджені і є основою для більшості подібних програм.

Враховуючи це, кожен з користувачів повинен мати свій рівень доступу, відповідно обов'язково повинна бути реєстрація і аутентифікація користувачів, з відповідним подальшим наданням потрібних прав. Користувачі, які не пройшли аутентифікацію, повинні бачити лише загальнодоступну титульну сторінку сайту.

Первинна реєстрація в системі може проводитись самостійно, але надання прав і розмежування прав доступу повинно проводитись лише адміністратором, що призначений керівництвом закладу.

З'ясувавши вимоги користувачів та перелік необхідної інформації, потрібно уважно їх розглянути з погляду розробника системи та визначити, які додаткові, не зазначені замовником дані повинні бути в базі даних. Для цього потрібно усвідомити схему взаємодії і те, хто і як використовувати систему.

Перша фаза процесу проектування передбачає створення концептуальної моделі даних. Концептуальне проектування включає визначення типів найважливіших сутностей та існуючих між ними зв'язків на основі наявної первісної інформації. Загальний підхід до створення таблиць у всіх реляційних

СУБД однаковий, тому на даному етапі можна ще не прив’язуватись до якоїсь конкретної СУБД.

Зведемо інформацію про потрібні нам дані в таблицю 2.8.

Таблиця 2.8 – Загальний список потрібної інформації

Поле	Опис
Лікарі	
ПІБ	Прізвище, ім’я, по-батькові
Спеціальність	
Пацієнти	
ПІБ	Прізвище, ім’я, по-батькові
Дата народження	
стать	
Діагноз	
Дата прийому	
Призначене лікування	
Аутентифікація користувачів:	
Логін	
Пароль	
Рівень доступу	

Наступним кроком необхідно розділити інформацію на окремі об’єкти (сутності), визначити їх атрибути та побудувати інфологічну модель БД діаграму «сутність-зв’язок», яка допоможе наочно побачити структуру бази даних, а також виявити та усунути її недоліки вже на початковому етапі, поки що без прив’язки до конкретної системи управління базами даних (СУБД). [16]

Відповідно потрібно виконати більш чіткий поділ інформації на сутності. Для реляційної бази даних це заготовки майбутніх таблиць.

Наступний кроком буде нормалізація бази даних. Теорію реляційних баз даних створив в 1970х роках Е.Кодд, і тоді ж він запропонував три нормальні форми, які допомагають побудувати правильну структуру бази даних.[16]

Спочатку потрібно привести таблиці до першої нормальної форми, яка характеризується тим, що значення всіх її полів атомарні (не можна розділити на дрібніші частини) і в її записах відсутні групи значень, що повторюються.

В таблиці 2.8 складене поле одне ПІБ. Розділяємо його на окремі атомарні складові: «Прізвище» «Ім'я» «По-батькові».

Далі проводимо аналіз полів, які мають повторювані значення. В даному випадку такими полями є поля «спеціальність», «Прізвище», «Ім'я», «По-батькові». Виділяємо їх в окремі таблиці, залишаючи в основній посилання на ID.

Крім того, в ній є поля «Діагноз» і «Призначене лікування», які можуть складатись з кількох записів в різні дні і мати складну непередбачувану структуру. Тому їх краще винести в окремі таблиці.

Для того щоб привести таблицю до другої нормальної форми потрібно визначити первинні ключі і добитись, щоб кожен неключовий атрибут повністю визначався первинним ключем і первинний ключ не був надлишковим. Ті атрибути, які залежать тільки від частини ключа, мають бути виділені в окремі таблиці.

Первинний ключ має бути унікальним у межах таблиці і не змінюватися, тому в усі таблиці вводимо нове поле типу лічильник, яке використовується як первинний ключ і має назву «ID».

Третя нормальна форма таблиця задовольняє умовам другої нормальної форми і жодне з неключових полів не ідентифікується іншим неключовим полем. Є ще форма Бойса-Кодда, четверта і п'ята нормальні форми, але в більшості випадків задовольняються приведенням БД до третьої нормальної форми.

На основі цього було створено таблиці: «Користувач» і «Пацієнт» (таблиці 2.9 і 2.10).

Таблиця 2.9 – Користувач

Поле	Опис
ID	Унікальне число
ID Прізвища	Посилання на таблицю Прізвище
ID імені	Посилання на таблицю Ім’я
ID по-батькові	Посилання на таблицю По-батькові
ID Посади	Посилання на таблицю посад
Адміністратор	Поле для визначення рівня доступу в програмі
Логін	
Пароль	

Таблиця 2.10 – Пацієнт

ID	Унікальне число
ID Прізвища	Посилання на таблицю Прізвище
ID імені	Посилання на таблицю Ім’я
ID по-батькові	Посилання на таблицю По-батькові
Дата народження	
стать	Ч/Ж

В цих таблиця немає, наприклад, прізвища, а є лише посилання на нього в таблицю-довідник, створену по такій структурі(таблиця 2.11).

Таблиця 2.11 – Прізвище

Поле	Опис
ID	Унікальне число
Прізвище	Прізвище

Аналогічно створюємо таблиці-довідники для імен, по-батькові, посад, діагнозів і ліків.

Крім того, оскільки кожен пацієнт може бути на прийомі у різних лікарів і кількість його відвідувань передбачити неможливо, то створюємо для цього окрему таблицю 2.12 і таблицю для призначеного лікування 2.13.

Таблиця 2.12 – Дати прийому

Поле	Опис
ID	Унікальне число
Дата прийому	
ID Лікаря	Посилання на таблицю Користувач
ID Пацієнта	Посилання на таблицю Пацієнт

Таблиця 2.13 – Призначене лікування

Поле	Опис
ID	Унікальне число
ID Пацієнта	Посилання на таблицю Пацієнт
ID дати	Посилання на таблицю Дати прийому
ID ліки	Посилання на таблицю Довідник ліків

Після розробки концептуальної моделі таблиць потрібно також визначити типи існуючих між ними зв’язків. В даному випадку фактично ми маємо дві основні таблиці: «Користувач»(Лікар) і "Пацієнт ", причому таблиці мають схожі атрибути «Прізвище» «Ім’я» «По-батькові». Крім того, велике значення мають таблиці "Діагноз" і " Призначене лікування ", які містять основну потрібну інформацію, і, в свою чергу, теж мають підлеглі таблиці. Всі інші таблиці бази даних мають підлеглий характер і виступають в ролі довідників.

Наступним етапом розробки є побудова інфологічної моделі розробленої бази даних у вигляді наочної ER-діаграми (рис. 2.2).

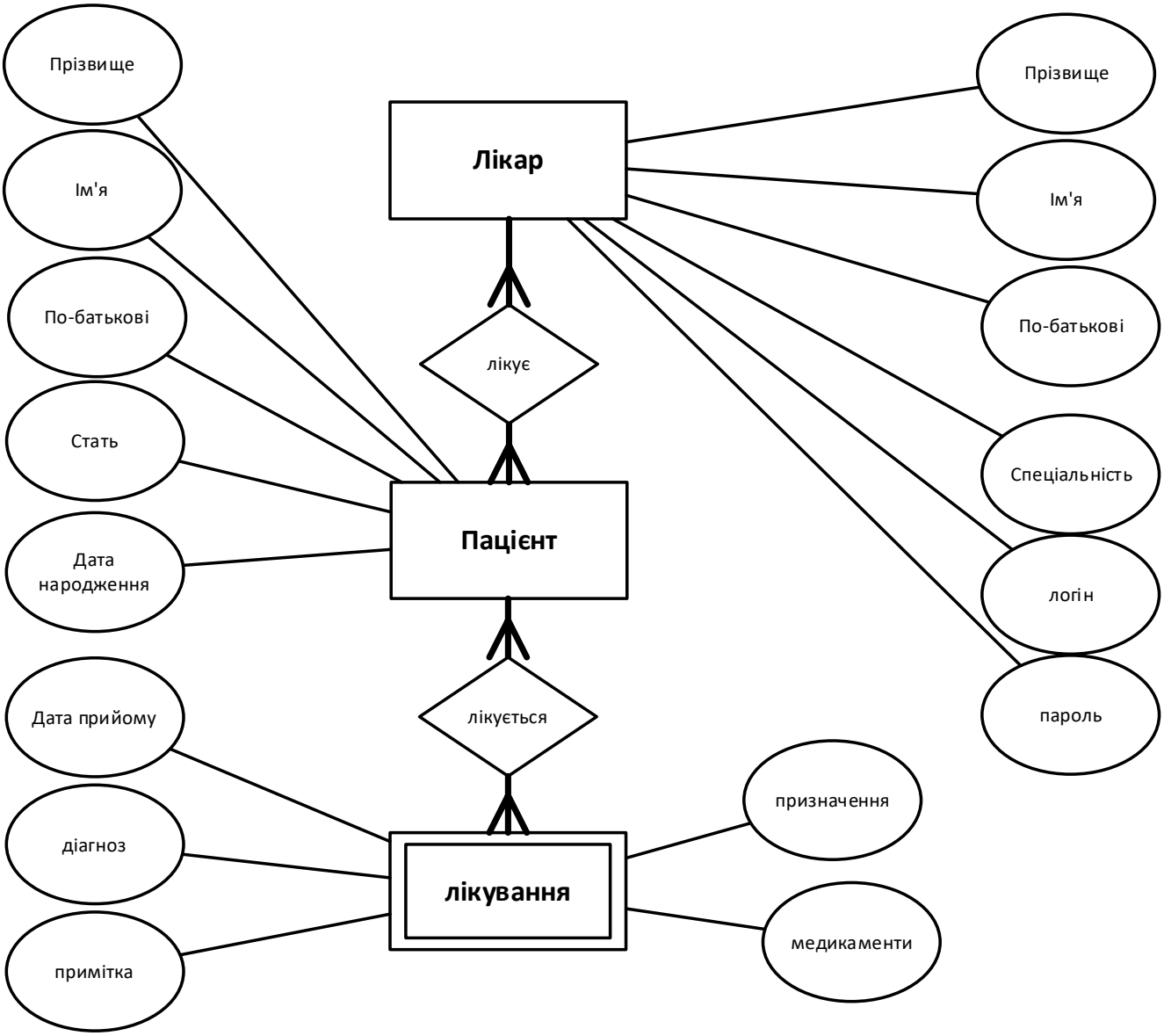


Рисунок 2.2 – ER-діаграма медичної бази даних

2.3 Розробка дизайну системи для управління медичною базою даних

Основною метою проекту є створення швидкого та зручного інструменту для роботи лікаря, тому він повинен бути максимально орієнтований на простоту і зручність використання. Розробка дизайну починається з прототипування. Це схематичне розташування елементів на сторінках, щоб уявити основну концепцію і зовнішній вигляд майбутнього сайту, дає можливість відпрацювати кілька версій і значно економити час. Найпростіше намалювати його від руки, але потім на цій основі створити більш якісний макет в спеціальній програмі, наприклад, можна

використати міжнародний онлайн сервіс NinjaMock [17] чи програми Figma [18] або Adobe Photoshop, які, крім того, дозволяють на його основі потім розробити повноцінний дизайн сайту. Для даного проекту використовуємо безкоштовну програму Figma, яка цілком задовольняє всім потрібним вимогам.

Прототип головної сторінки до авторизації показаний на рисунку 2.3.

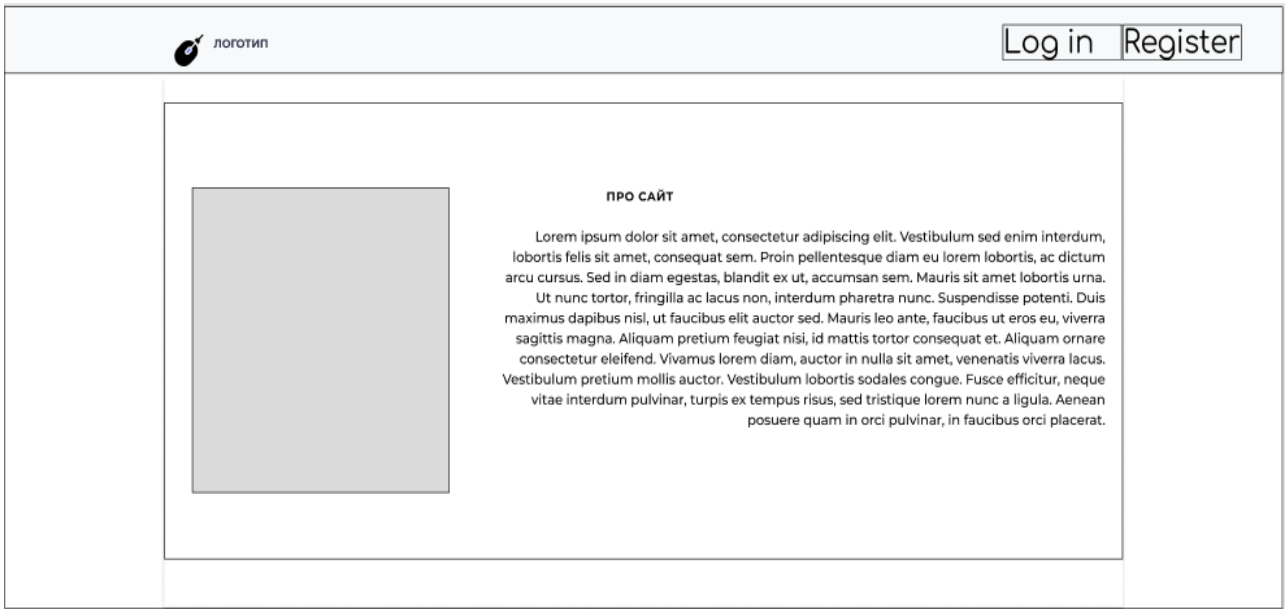


Рисунок 2.3 – Прототип головної сторінки до авторизації

Аналогічно створюємо прототипи сторінок реєстрації (рисунок 2.4) і авторизації. Вони дуже схожі і відрізняються лише двома полями.



Рисунок 2.4 – Прототип сторінки реєстрації

Після процедури авторизації на головній сторінці з’являється екран для роботи з карткою пацієнта (рисунок 2.5).

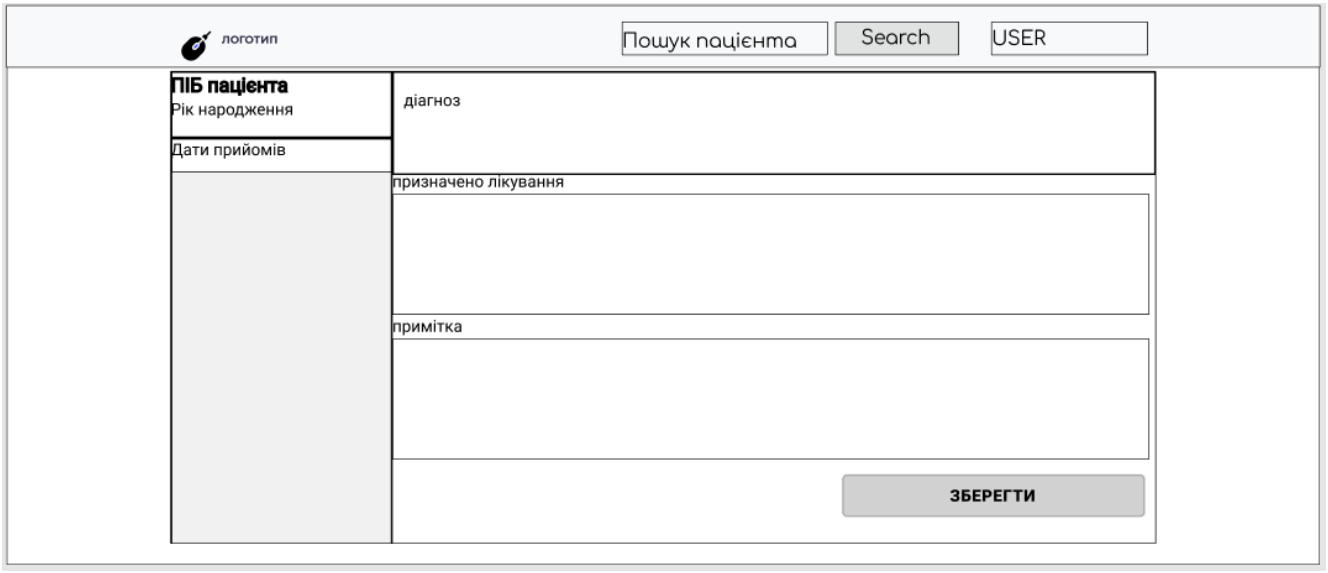


Рисунок 2.5 – Прототип екрану картки пацієнта

Після створення, доопрацювання і уважного вивчення прототипу, обирається кольорове рішення і розробляється повноцінний дизайн сайту. Для цього використовуємо програму Figma, яка створювалась для роботи з WEB ресурсами і зручна як для дизайнера сайтів, так і для верстальника. Крім того у Figma є безкоштовна версія, можливостей якої цілком достатньо для середньостатистичного дизайнера. У ній можна відразу переглянути і скопіювати CSS-стиль будь-якого об’єкта, добре видно всі розміри та відстані до об’єктів, легко експортувати елементи дизайну, іконки тощо у зручному форматі.

Одна з перших задач дизайну це підбір та поєднання кольорів. Колірна гама має величезний вплив як на зручність сприйняття дизайну, так і на емоційний стан користувача. Для роботи лікаря кольорове рішення особливо важливе, оскільки від нього залежать особливості психологічного стану і, як наслідок, працездатність лікаря і результати його роботи. Завдяки правильно підібраній палітрі кольорів, тексти стає зручно читати, кнопки дії в інтерфейсі привертають до себе увагу, забороняючі знаки дають попередження.

При підборі кольорів корисно пам’ятати і про те, що на рівні сприйняття користувачі часто схильні наділяти деякі кольори та їхні комбінації певним контекстом. Це пов’язано з так званим законом попереднього досвіду. Наприклад, червоний це забороняючий колір світлофора, «стоп», помилка на письмі чи попередження про небезпеку, тому часто кнопки незворотної дії (наприклад «видалити») в інтерфейсах зображають саме червоними. [19].

Враховуючи те, що інформаційна система управління медичною базою даних повинна використовуватись лише для роботи, причому такої, що вимагає максимальної зібраності, відповідальності і уваги, використання яскравих колірних рішень буде явно недоречним і відволікатиме увагу, в даному випадку правильним кроком буде використання максимально строгого дизайну: чорний текст на білому фоні, з використанням кольорових деталей лише для елементів навігації, і також відсканованих документів (наприклад енцефалограми).

Обравши кольорове рішення, створюємо в Figma на основі розробленого прототипу (рисунок 2.2) дизайн стартової сторінки сайту до авторизації користувача, показаний на рисунку 2.6.

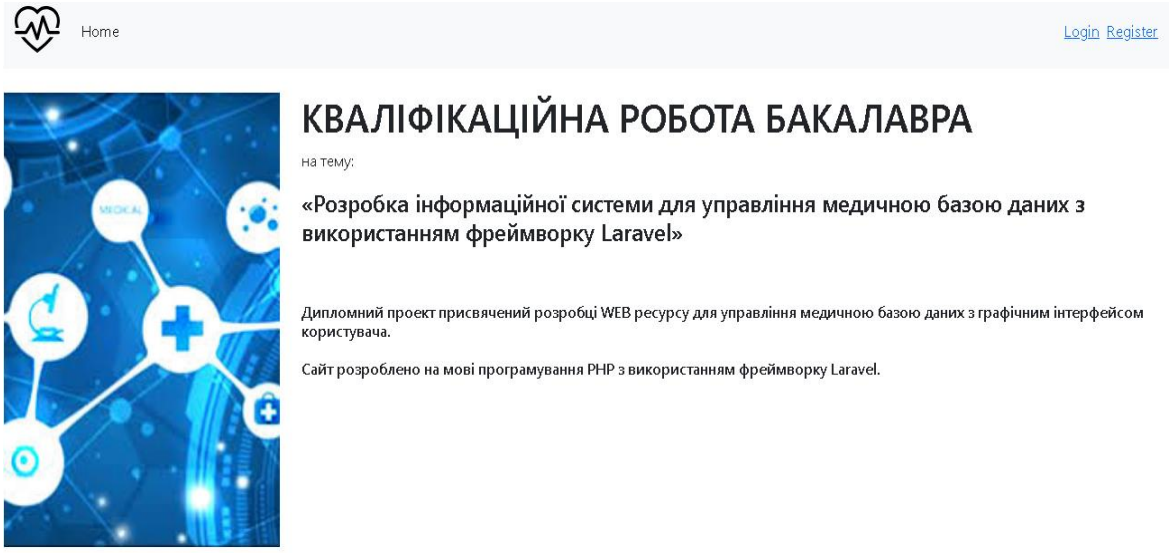


Рисунок 2.6 – Дизайн головної сторінки до авторизації

Аналогічно цьому на основі вже розроблених прототипів сторінок сайту (рисунки 2.3, 2.4 і 2.5) і обраного варіанту кольорового рішення, а також враховуючи загальний обраний при розробці стартового екрану (рисунок 2.6)

стиль, розробляємо і прототипи інших сторінок сайту, а також варіанти розміщення елементів дизайну на різних можливих варіантах екранів, від мобільного телефону і до сучасних телевізорів з великими екранами.

2.4 Висновки до 2 розділу

У цьому розділі було проаналізовано те, які саме можливості для реалізації інформаційної системи управління медичною базою даних повинні в ній бути, потім сформульовано вимоги і вибрано, якими засобами це можна забезпечити.

Було проведено аналіз предметної області, потрібних технічних рішень та алгоритмів для формулювання технічних вимог для розробки програми. Визначено головний функціонал розроблюваної системи, розглянуто та описано можливі варіанти її використання, розроблено діаграму варіантів використання системи, спроектовано бізнес-логіку додатку, прописано функціональні та нефункціональні вимоги, проведено їх аналіз, складено таблиці для відповідного сценарію використання та кожної функціональної вимоги.

В результаті аналізу було визначено, що програма повинна мати вільний багатокористувацький доступ з мережі інтернет, можливість повноцінної реєстрації і авторизації, а також зберігати і обробляти великі об’єми інформації. Це звичайні вимоги для створення більшості більш-менш великих сайтів. Відповідно, і інструменти для цієї мети краще брати оптимальні і широко поширені.

Було розроблено концептуальну модель бази даних, визначено структуру і склад її таблиць, типи зв’язків між ними. Створено прототипи сторінок сайту, обрано кольорове рішення і розроблено повноцінний дизайн в програмі Figma.

В результаті цього була виконана вся підготовча робота, що дозволяє приступити до фізичної реалізації інформаційної системи для управління медичною базою даних з використанням фреймворку Laravel.

3 РЕАЛІЗАЦІЯ WEB РЕСУРСУ ДЛЯ УПРАВЛІННЯ МЕДИЧНОЮ БАЗОЮ ДАНИХ

3.1 Вибір мови програмування та фреймворку для реалізації інформаційної системи для управління медичною базою даних

Враховуючи те, що WEB ресурс для управління медичною базою даних повинен динамічно використовувати велику кількість інформації з баз даних, потрібно вибирати інструмент, який призначений для роботи в першу чергу в Backend. З популярних мов програмування лідерство в цій сфері багато років тримала PHP, яка використана в якості серверної мови на 74.5% сучасних сайтів[23]. У мережі зараз активно рекламується і просувається Python, але PHP лідирує в швидкості і буде продовжувати це робити ще протягом багатьох років. У тестах швидкості Python програє PHP майже у 18 разів! Навіть у випадку використання скомпільованого коду відставання Python складає майже шість раз [24]. Мови Java і C# більш складні у освоєнні і використанні, тому для інформаційної системи управління медичною базою даних вибираємо PHP [25].

Важливий елемент для роботи розробника програмного забезпечення – правильно підібране середовище розробки. Для PHP це в першу чергу PhpStorm [26], але PhpStorm платний і потребує порівняно багато ресурсів, що теж важливо.

Як альтернатива PhpStorm для невеликих проектів часто використовується Visual Studio Code[27]. Це безкоштовне інтегроване середовище розробки, яке підтримує багато мов програмування, а при грамотному підборі плагінів його можливості наближаються до PhpStorm і цілком достатні для комфортної роботи. Враховуючи це, обираємо Visual Studio Code.

В наш час на чистих мовах програмування майже не працюють, оскільки для кожної з популярних мов є інструменти, які значно полегшують і прискорюють розробку. В першу чергу, це CMS – системи (Content Management System): «конструктори» з готовими шаблонами, але підходить вони лише для вирішення типових завдань, типу блогу чи інтернет-магазину. Для нетипових проектів

використовують фреймворки: набори типових бібліотек, що дозволяють реалізувати будь-який сайт. Продуктивність його буде вищою, хоча сама розробка складніша. Найпопулярнішим серед PHP фреймворків на сьогодні є Laravel , який на сьогодні вважається найкращим для подібних. [28] [29]

Laravel підтримує наступні п’ять типів баз даних [29]: MySQL, MariaDB, PostgreSQL, SQLite і SQL Server 2017+. Проаналізувавши як можливості самих БД, так і їх використання з Laravel обираємо СУБД MySQL, яка з даного списку є найбільш поширеною і простою для використання.

3.2 Встановлення Laravel

Для початку роботи потрібно встановити середовище розробки, в даному проекті було обрано Visual Studio Code. Скачуємо його з офіційного сайту, встановлюємо і налаштовуємо для роботи з PHP[27][30]. Крім того, на робочому місці повинен бути встановлений Composer і PHP[31][25].

Далі вибираємо папку в якій буде розміщено проект, потім в командному вікні перейти в неї і виконати команду для установки Laravel наприклад:

```
composer create-project laravel/laravel Medic
```

Composer по цій команді створює каталог Medic, потім скачує з офіційного сайту і встановлює у нього Laravel.[29] В Laravel вмонтований власний локальний сервер: в командному вікні переходимо в каталог Medic і запускаємо локальний сервер командою *php artisan serve* . Тепер локальний сервер доступний за адресою http://127.0.0.1:8000. Набираємо її в браузері і перевіряємо правильність установки: повинно відкритись стартове вікно Laravel.

3.3 Створення бази даних системи

Далі налаштовуємо підключення до бази даних. Для цього на робочому місці потрібно розвернути повноцінний локальний сервер, наприклад XAMPP чи Open Server. Запускаємо його, відкриваємо phpMyAdmin і створюємо базу даних. Назвемо її теж Medic. В файлі .env прописуємо її тип, ім’я користувача і пароль:

DB_DATABASE= Medic
DB_USERNAME=root
DB_PASSWORD=

Після цього можна приступати до безпосереднього створення таблиць. Найпростіше можна створити таблиці phpMyAdmin, але у Laravel для цього застосовано спеціальний механізм: міграції. Це файли, в яких описано структуру таблиць. В Laravel в каталогові database\migrations вже присутні міграції, що створюють п’ять стандартних таблиць (рисунок 3.3).

	Таблиця	Дія	Рядки	Тип	Зіставлення	Розмір	Фрагментовані
☐	failed_jobs		0	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
☐	migrations		4	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
☐	password_resets		0	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
☐	personal_access_tokens		0	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
☐	users		0	InnoDB	utf8mb4_unicode_ci	16.0 КБ	-
	5 таблиць	Всього	4	InnoDB	utf8mb4_0900_ai_ci	80.0 КБ	0 Б

Рисунок 3.3 – Стандартні таблиці в Laravel за замовчуванням

Крім власне таблиць, Laravel вимагає створити до кожної з таблиць окрему модель, за допомогою якої відбувається взаємодія програми з БД.

Але краще одночасно створити і модель, і міграцію командою:

php artisan make:model ім’я -m.

Створюємо моделі і міграції для розроблених таблиць.

```
php artisan make:model Family -m
php artisan make:model firstName -m
php artisan make:model middleName -m
php artisan make:model Positions -m
php artisan make:model patient -m
php artisan make:model dateReception -m
php artisan make:model diagnosis -m
php artisan make:model DiagnosisGuide -m
php artisan make:model therapy -m
php artisan make:model pharmacy -m
```

Команда створює лише каркас міграції, за замовчуванням в якому автоматично створюються поля *ID*, *created_at* і *updated_at* (ID, час створення і час останнього редагування). Щоб добавити інші поля, треба відкрити файл у редакторі і вручну записати команди для їх створення в функції up(). Наприклад для таблиці ‘families’ це буде так:

```
public function up(): void
{
    Schema::create('families', function (Blueprint $table) {
        $table->id();
        $table->string('family')->nullable();
        $table->timestamps();
    });
}
```

Таблиця “users” вже існує, але в ній немає полів для прізвища, імені, по-батькові, посади і визначення рівня доступу (прав) в програмі, тому добавляємо їх окремою міграцією, не створюючи саму таблицю командою:

```
php artisan make:migration add_field_to_users_table
```

Крім того, потрібно створити зв’язки між таблицями. Наприклад, в таблиці «користувач» (*user)* для поля прізвища потрібно створити посилання на таблицю «прізвище» (families), створивши поле ‘family_id’ таким чином:

```
$table->foreignId(‘family_id’)->nullable()
->constrained()      //створення зв’язку
->cascadeOnDelete()  //поведінка при видаленнні
->cascadeOnUpdate();
```

Заповнюємо всі міграції і виконуємо їх командою:

php artisan migrate.

При наявності помилок усуваємо їх, виконуємо міграції повторно і перевіряємо отриману структуру БД візуально в Designer phpMyAdmin (рисунок 3.4) .

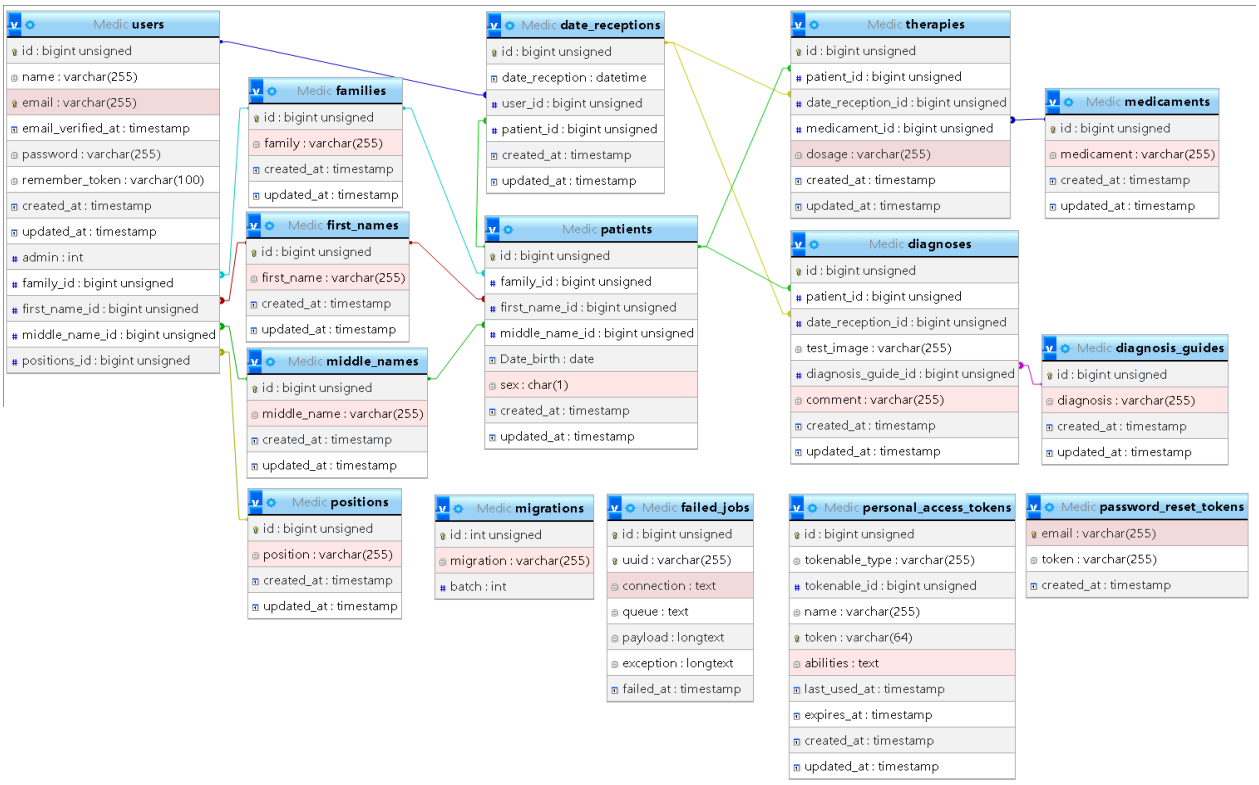


Рисунок 3.4 – Структура БД в Designer phpMyAdmin

3.4 Створення моделей

В Laravel для взаємодії програми з таблицями БД використовуються моделі, що в даному випадку були створені одночасно з міграціями і розміщені в каталогові app\Models. Аналогічно міграціям, Laravel створив лише каркаси моделей, які потрібно відкрити у редакторі і при необхідності вручну підправити, в першу чергу створивши списки полів і функції для зв'язків між таблицями.

Наприклад, в модель для таблиці «Пацієнт» потрібно добавити:

Список полів:

```
protected $fillable = ['family_id','first_name_id','middle_name_id','Date_birth','sex'];
```

Функцію для створення зв'язку з підлеглою таблицею дат прийому:

```
public function dateReception()
{
  return $this->hasMany(dateReception::class);
}
```

Функцію для створення зв'язку з головною таблицею прізвищ:

```
public function Family()
{
   return $this->belongsTo(Family::class);
}
```

Функції для зв'язків з іншими таблицями створюються аналогічно. Повний текст даної моделі наведено в додатку.

3.5 Реалізація дизайну інтерфейсу системи

Для створення сторінок сайту, потрібно на основі створеного дизайну виконати їх верстку, використовуючи HTML, CSS і при необхідності, JavaScript. В поєднанні з фреймворком Laravel це дасть можливість написати складний проект з дуже невеликою кількістю програмного коду, який буде швидко оброблений, а швидкість завантаження сайту значно зростає, що, в свою чергу, підвищує його рейтинг в пошукових системах, а в підсумку збільшує відвідуваність і прибутки для комерційних сайтів.

В той же час це досить трудомістка робота, що потребує дуже уважної розробки з врахування стилів CSS для всіх елементів, а також забезпечення адаптивності дизайну для всіх можливих варіантів екранів і популярних браузерів. Тому майже завжди при верстці використовують різні фреймворки і інструменти, що полегшують роботу, найпопулярнішим з яких є Bootstrap [32]. Фреймворк поставляється з величезним асортиментом компонентів: форми, кнопки, таблиці, меню, випадаючі списки, індикатори прогресу, і т.д., які можна легко додати на свій сайт, причому вони будуть адаптуватися в залежності від розміру екрану. Дизайн компонентів Bootstrap дійсно лаконічний і служить прикладом для дизайнера.

Для використання Bootstrap в даному випадку достатньо мінімальної конфігурації: файлів bootstrap.min.css і bootstrap.min.js. Скачуємо з офіційного сайту [32] файл *bootstrap.min.css*, поміщаємо його в каталог *\public\assets\css*, а в шапку файлу HTML поміщаємо посилання на нього – рядок:

<link rel="stylesheet" href="{{asset('assets/css/bootstrap.min.css')}}">.

На першому етапі на основі розробленого в Figma дизайну створюємо окремі HTML сторінки сторінок сайту, поки що в спрощеному вигляді, з тим, щоб потім інтегрувати їх в Laravel, підключити базу даних і відпрацювати всі функції WEB ресурсу. Іконку для логотипу в даному випадку вибрано на спеціалізованому сайті

«Icons» [34], але можна і створити самостійно в одному з редакторів векторної графіки чи замовити у спеціаліста.

Для створення сторінок у Laravel використовується Blade – простий, але потужний шаблонізатор, що надає зручний та виразний синтаксис для створення динамічних сторінок. Він дозволяє легко вставляти PHP-код безпосередньо в HTML-розмітку за допомогою спеціальних директив, що робить його ідеальним рішенням для створення складних інтерфейсів.

Макети Blade в Laravel служать базовою структурою для веб-сторінок, дозволяючи розробникам розміщати загальні елементи, такі як заголовки, колонтитули та меню в одному файлі. Це сприяє повторному використанню коду, послідовності в дизайні та розділенню проблем між презентацією та логікою. Файли шаблонів Blade мають розширення .blade.php і зберігаються в директорії resources/views, але зазвичай для зручності в ній ще створюють окремі папки для логічного поділу файлів, наприклад «/pages».

Для цього створюємо один файл «main.blade.php», розміщений в каталозі \resources\views\templates\, де створюємо у якому базову структуру HTML, що буде спільною для кількох представлень, і куди за допомогою директив ‘@yield включаються динамічні частини, розміщені в інших файлах. Відповідно, на початку файлів для кожної сторінки ставимо адресу файлу шаблону @extends(‘templates.main’), а частину, яку потрібно вставити, огортаємо в обгортку:

@section(‘…назва блоку…’) …. @endsection.

Для спільного меню створюємо окремий файл components\navbar.blade.php.

Далі відкриваємо файл routes\web.php і створюємо в ньому маршрути для відкриття сторінок Blade. Наприклад, для файлу pages\hello.blade.php це буде так:

```
Route::get(‘/hello’, function () {
    return view(‘pages.hello’);
});
```

Виконуємо верстку сторінок в чорновому варіанті, перевіряємо їх відкриття, зовнішній вигляд (поки що без малюнків), і, при потребі, виправляємо помилки. Оскільки змінний контент сторінок в основному формується на основі інформації з бази даних, то кінцеве формування сторінок краще робити в процесі налаштування програми після підключення бази даних.

3.6 Реалізація основних функцій інформаційної системи для управління медичною базою даних

Далі потрібно налаштовати роботу форм у відповідності з потрібними функціями інформаційної системи для управління медичною базою даних.

Для виконання цього потрібно мати підключення до бази даних і створити контролери, що будуть відповідати за виконання потрібних операцій. Файли контролерів зберігаються в директорії resources/views, але, щоб запобігти плутанині, додатково в ній ще створюємо для них окремий каталог /Application. В один контролер можна включити багато операцій (функцій), але для зручності краще створити окремі контролери для виконання певних груп дій: реєстрації, авторизації, роботи з карткою пацієнта і так далі.

Перша дія користувача після відкриття стартової сторінки це реєстрація в системі, тому спочатку відпрацьовуємо цю дію.

В командному рядку створюємо контролер для реєстрації:

php artisan make:controller Application/RegisterController

Крім того, створюємо контролер для відображення сторінок за маршрутами:

php artisan make:controller Application/PagesController

Підключаємо їх в файл маршрутів routes\web.php:

use App\Http\Controllers\Application\PagesController as ApplicationPagesController;

use App\Http\Controllers\Application\RegisterController;

З стартової сторінки (Рисунок 2.6) для відкриття форми реєстрації користувач повинен натиснути посилання «*Register*», відповідно в файлі головного меню resources\views\components\navbar.blade.php повинен бути рядок:

Register,

При натисканні на який Laravel шукає в файлі маршрутів routes\web.php відповідний рядок. Для цього коригуємо наявний там маршрут, підключаючи потрібний контролер RegisterController і посилання на функцію в ньому, наприклад, «index»:

```
Route::controller(RegisterController::class) ->group(function () {
   Route::get("/register", 'index')->name('register.form');
});
```

Відповідно в RegisterController створюємо функцію, яка буде відкривати форму реєстрації:

```
    public function index()    {
      return view('pages.register');
   }
```

Сама форма повинна мати приблизно таку структуру:

```
@extends('templates.main')  //підключення форми до загального шаблону HTML
@section('main')        // секція 'main' в до загальному шаблоні HTML
  <div class=".....">
    <form action="{{route('register.action')}}" method="POST"> //шапка форми
      @csrf  //обов'язковий компонент форми – захист від злому
```

```
<div class="....">
```
//поле для редагування:
```
<input type="..." name="..." value="{{ ... }}" id="...">
</div>
                    ……інші поля……
<button type="submit" class="btn ... "> Create account </button>   //кнопка:
</form> //кінець форми
</div>
@endsection //кінець
```

Додатково в форму для реєстрації потрібно включити контроль формату вводу, перевірку на помилки і так далі при потребі. В подвійних фігурних дужках ({{….}}) в структуру Blade можна включати дещо змінені команди і змінні PHP.

Як видно з шапки форми, при натисканні на кнопку «*Create account*» повинен відбутись перехід по маршруту *{{route('register.action')}}",* тобто потрібно створити цей маршрут в файлі web.php, добавивши рядок

```
Route::post("/register", 'register')->name('register.action');
```

Відповідно в контролері потрібно створити функцію «register», яка, використовуючи клас «RegisterRequest», перевіряє введені користувачем значення, кодує пароль і створює запис в БД:

```
public function register(RegisterRequest $request)
  {
    $data = $request->validated();
    $data['password'] = Hash::make($data['password']);  // шифрує пароль
    User::create($data);                    //запис в БД
    return redirect()->back()->with('success', 'Register success'); //повернення
  }
```

Само собою, також потрібно створити і клас «RegisterRequest» – набрати в командному рядку:

php artisan make:request User/RegisterRequest,

Це створить потрібний request-клас в файлі app\Http\Requests\User\RegisterRequest.php. В ньому в функції «rules» можна перевірити введені поля на відповідність формату: типу даних, довжині рядка і т.д.:

```
public function rules()
{
   return [
      'name' => ['required', 'string', 'max:255'],
      'email' => ['required', 'string', 'max:255', 'unique:users,email'],
      'password' => ['required', 'string', 'min:6', 'confirmed']
         // «confirmed» – перевірка пароля
   ];
}
```

На цьому всі потрібні файли і функції для реєстрації користувача створено.

Включаємо локальний сервер, відкриваємо стартову сторінку (рисунок 2.6), обираємо в меню «Register»- відкривається форма реєстрації (рисунок 3.5).

Заповняємо всі поля і натискаємо кнопку «*Create account*». Проходить перевірка інформації і в разі успіху проходить запис в БД. Отримуємо повідомлення 'Register success' (реєстрація успішна).

За таким принципом проводимо налаштування всіх інших сторінок сайту, починаючи з сторінки авторизації і закінчуючи призначенням лікування. З алгоритмом роботи лікаря в подібних програмах можна ознайомитись, наприклад на сайті «Медична справа» [35].

Проводимо заповнення кількох карток абонентів тестовими даними і детально перевіряємо роботу програми, усуваючи виявлені помилки.

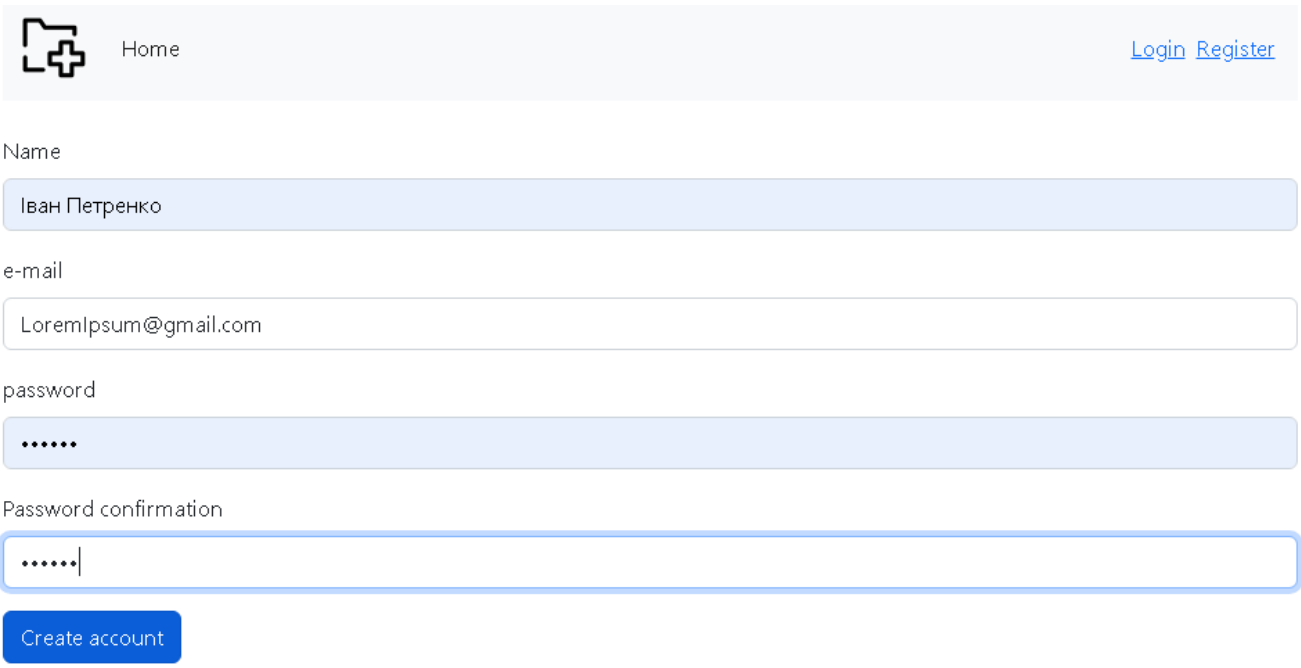


Рисунок 3.5 – Форма реєстрації в програмі

На цьому розробку сайту можна вважати закінченою. Тексти основних розроблених в процесі роботи файлів наведені в додатку.

Якщо перевірка пройшла успішно можна розпочинати тестовий період експлуатації програми.

3.7 Розгортання та супровід системи

Наступним кроком потрібно вирішити, де і як саме розмістити WEB ресурс для загального користування, оскільки при розробці він був в локальному варіанті.

Для цього в першу чергу потрібно визначитись з тим, хто буде користуватись застосунком. Якщо система для управління медичною базою даних потрібна для великої кількості користувачів, то потрібно щоб вона мала зручну адресу для запам’ятовування і займала якомога вищі показники в рейтингах пошукових систем. Першим кроком для досягнення такої мети потрібно обрати доменне ім’я для сайту, таке, щоб мало логічний зміст і легко запам’ятовувалося. Наприклад, для проектованої системи це можуть бути такі імена, як *medic.ua*, *medic.com* чи подібні. Перевірити можливість використання обраного доменного імені можна на будь якому з чисельних сайтів, наприклад на hostiq.ua [36]. Там же можна ознайомитись з цінами і купити обране доменне ім’я. Причому треба мати на увазі, що ціни на різні імена можуть досить сильно відрізнятись, наприклад *medic.com.ua* обійдеться в 483,00 грн. на рік, а на *medic.tech* потрібно вже 300 931,68 грн./рік.

Обравши і оплативши доменне ім’я, потрібно визначити, де розмістити сервер. Якщо потрібен лише один відносно невеликий сайт, найкращий варіант – викласти WEB ресурс на один з хмарних хостингів, яких існує дуже багато, як українських так і міжнародних. Наприклад, вже згаданий hostiq.ua [36] теж надає такі послуги, і в мінімальному варіанті для одного сайту розміром до 0,5ГБ місячний тариф складає 102,90 грн. При більш дорогих тарифних планах розміри і кількість сайтів може бути збільшена. Користувачам надається багатофункціональна панель керування (рисунок 3.6), де можна кількома кліками з бібліотеки автовстановлювача встановити Laravel і підключити БД. Провайдер обіцяє за рахунок заміни веб-сервера Apache на LiteSpeed та HDD-дисків на SSD пришвидшення швидкості роботи сайтів в 5 разів і підвищення надійності в 7 раз. В цілому при використанні хмарних провайдерів можна налаштувати автоматизований контроль практично всього, не надто вдаючись у технічні нюанси.

Далі потрібно виконати розгортання системи на обраному сервері. Найбільш правильним варіантом є використанням розділу Deployment з офіційної документації фреймворку [37] з врахуванням особливостей як обраного сервера, так і використаного програмного забезпечення. Одним з кращих варіантів є автоматичне встановлення Laravel, якщо таку послугу надає провайдер хостингу, з наступним копіюванням через FTP файлів, що створені користувачем. На практиці часто повністю копіюють всі каталоги проекту, крім vendor, node_modules і .git, а також каталогу public\storage. Файли з кореневого каталогу копіювати не потрібно.

Альтернативою автоматичному встановленню є повне завантаження Laravel через FTP у вигляді архіву з наступним розпакуванням, що значно прискорює процес. Ще краще використати GIT: налаштувати з'єднання і зробити *git clone* на сервері (попередньо налаштувавши *.gitignore*), що завантажить програму на сервер. Це складніший і довший варіант, але налаштування GIT все одно доведеться робити в процесі подальшого супроводу програми, оскільки копіювати файли вручну Вам набридне вже після першої ж зміни в програмі.

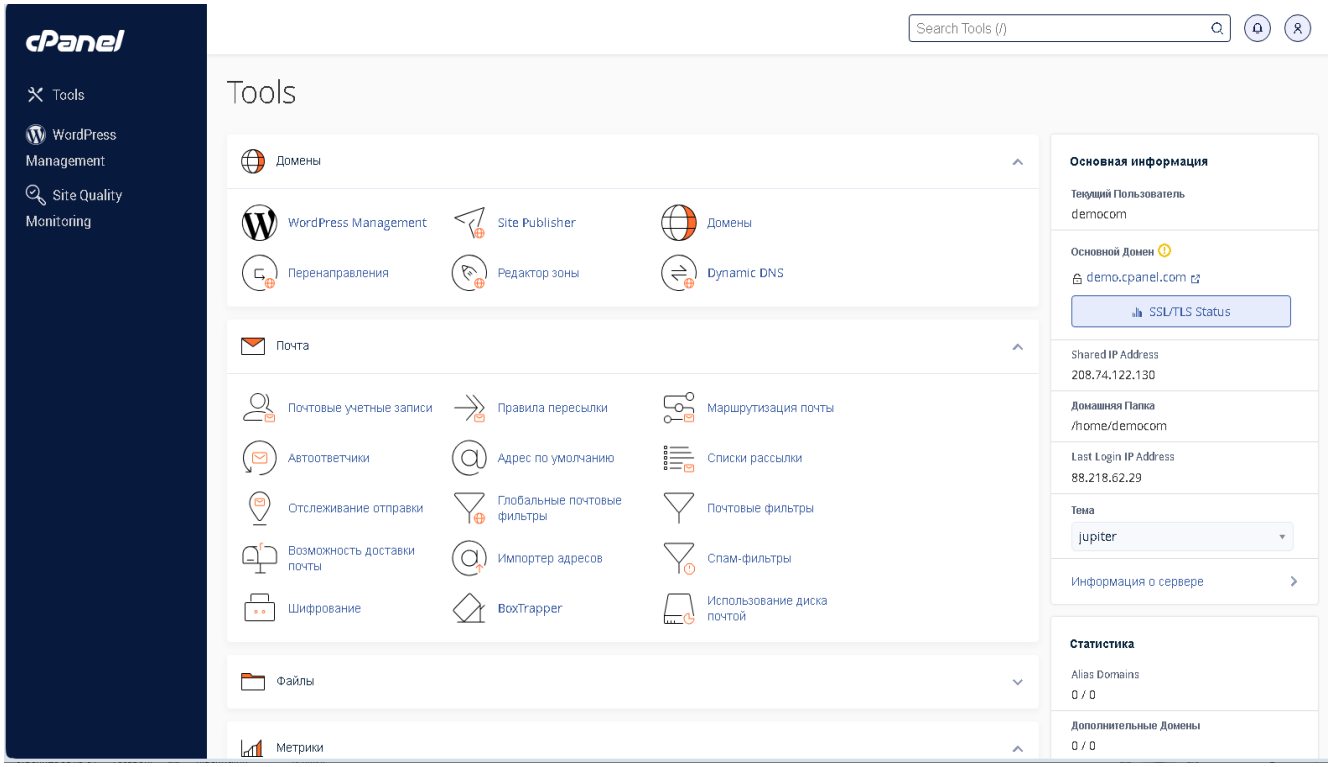


Рисунок 3.6 – Приклад інтерфейсу панелі керування хостингом [36]

Після завантаження файлів потрібно на сервері виконати *composer install* і *npm install,* налаштувати підключення до БД в файлі .env, потім відкрити *phpMyAdmin* і створити нову БД. Таблиці в БД створюємо з файлів міграцій командою *php artisan migrate*. Як варіант: можна завантажити дамп бази даних з локального сервера. Після цього варто скористатись інструкцією з налаштування Laravel на хостингу для конфігурації сервера і оптимізації роботи, щоб переконатися, що програма працює якомога ефективніше. [37]

Далі перевіряємо роботу, реєструємо користувача, який матиме права адміністратора і вручну в *phpMyAdmin* ставимо для нього 1 в полі *Admin* таблиці *User.*

Проводимо тестування програми з виправленням виявлених помилок (при необхідності).

Після введення сайту в експлуатацію починається етап супроводу програмного забезпечення. В першу чергу потрібно ще раз детальніше протестувати сайт на роботу в різних браузерах, для різних операційних систем та різних розмірів екрану. Уважно вивчати зауваження і побажання користувачів щодо покращення роботи системи.

Всі виявлені помилки виправляємо на локальному сервері, налаштовуємо підключення GIT до сервера, якщо цього не було зроблено ще на етапі розгортання програми і клонуємо зміни на сервер.

В подальшому постійно продовжуємо цю роботу: тестуємо програму, вивчаємо побажання користувачів і менеджерів сайту, вносимо запропоновані зміни і клонуємо їх на сервер.

3.8 Висновки до 3 розділу

В даному розділі було проведено реалізацію функцій розроблюваного ресурсу:

– виконано аналіз потрібних технічних рішень та варіантів їх реалізації;

- виконано верстку сторінок сайту;
- встановлено фреймворк Laravel на локальний хостинг;
- підключено створені сторінки сайту до Laravel,
- створено базу даних,
- описано процес підключення бази даних і її заповнення, а також шляхи для оптимізації і підвищення ефективності її роботи.
- реалізовано і відпрацьовано роботу основних функцій системи для управління медичною базою даних.
- розглянуто процес розгортання програмного забезпечення і його супроводу;
- проведено аналіз можливих варіантів розміщення системи на сервері і отримання доменного імені;
- описано процес встановлення на хостинг і варіанти шляхів для його виконання,
- розглянуто процес технічної підтримки сайту після вводу в експлуатацію і шляхи подальшого покращення його роботи.

ВИСНОВКИ

Даний дипломний проєкт присвячений розробці інформаційної системи для управління медичною базою даних на мові програмування PHP з використанням фреймворку Laravel і графічним інтерфейсом користувача.

В процесі розробки використовувалось середовище розробки Visual Studio Code. WEB ресурс був встановлений на локальний сервер з використанням програми Open Server і базою даних MySQL.

Перед розробкою програми було вивчено існуючі системи аналогічного призначення, проведено аналіз їх можливостей, варіантів оформлення, зручності використання і на цій основі розроблено загальну концепцію WEB ресурсу.

Потім створено прототипи сторінок сайту. На їх основі розроблено детальний дизайн в програмі Figma, а потім виконано верстку HTML сторінок сайту з використанням стилів CSS і фреймворку Bootstrap.

Основну частину роботи по розробці інформаційної системи для управління медичною базою даних було виконано в найбільш популярному на даний час фреймворку Laravel. До каркасу Laravel підключено підготовлену верстку сторінок сайту, засобами даного фреймворку створена база даних, після чого реалізовані основні функції системи.

В результаті проведеної роботи було створено інформаційну систему для управління медичною базою даних, яка підвищить рівень надання медичних послуг, відпрацьовано процес підключення до бази даних, формування запитів SQL, їх передачу і отримання результатів з наступним їх використанням.

Система дозволить швидко й ефективно налагодити електронний документообіг у медичних закладах, гнучко вибудовувати роботу з пацієнтами, а також підготувати заклади охорони здоров’я для підключення до електронної системи охорони здоров’я eHealth згідно постанови КМУ від 25.04.2018 №411 [7].

ПЕРЕЛІК ПОСИЛАНЬ

1. Міністерство охорони здоров'я України. Офіційний веб-сайт Веб-сайт. URL : https://moz.gov.ua/uk (дата звернення : 02.03.2025)

2. Медичні інформаційні системи як складова ЕСОЗ. Веб-сайт. URL : https://moz.gov.ua/uk/medichni-informacijni-sistemi-yak-skladova-esoz (дата звернення : 02.03.2025)

3. Про затвердження технічних вимог до електронної медичної інформаційної системи для її підключення до центральної бази даних електронної системи охорони здоров'я. Наказ НСЗУ № 5 від 02.01.2025. Веб-сайт. URL : https://e-health-ua.atlassian.net/wiki/spaces/EH/pages/1346764809 (дата звернення : 02.03.2025)

4. ТЕХНІЧНІ ВИМОГИ до електронної медичної інформаційної системи для її підключення до центральної бази даних електронної системи охорони здоров'я. В редакції наказу НСЗУ № 5 від 02.01.2025. Веб-сайт. URL : https://e-health-ua.atlassian.net/wiki/spaces/EH/pages/1346764809 (дата звернення : 02.03.2025)

5. ТЕСТОВА ПРОГРАМА встановлення відповідності електронних медичних інформаційних систем технічним вимогам URL : https://e-health-ua.atlassian.net/wiki/spaces/EH/pages/1346764809 (дата звернення : 02.03.2025)

6. Деякі питання електронної системи охорони здоров'я. Постанова КМУ від 25.04.2018 №411. URL: https://zakon.rada.gov.ua/laws/show/411-2018-п#Text (дата звернення : 02.03.2025)

7. КОНЦЕПЦІЯ розвитку електронної охорони здоров'я. Постанова КМУ від 20.12.2020 №1671-р URL: https://zakon.rada.gov.ua/laws/show/1671-2020-р#Text https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 2.03.2025)

8. Медичні інформаційні системи: огляд можливостей і приклади використання. URL : https://evergreens.com.ua/ua/articles/medical-information-systems.html#oglyad https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

9. Електронна система охорони здоров'я в Україні. URL : https://ehealth.gov.ua (дата звернення : 02.03.2025)

10. Підключені до ЦБД Медичні Інформаційні Системи. URL : https://ehealth.gov.ua/pidklyucheni-do-ehealth-mis/
https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

11. Helsi – зручна медична інформаційна система. URL: https://reform.helsi.me
https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

12. Doctor Eleks: URL : https://doctor.eleks.com
https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

13. MIC «Askep.net» URL : https://askep.net
https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

14. EMCImed. Медична інформаційна система. URL : https://emci.ua
https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

15. Що таке CMS? Огляд кращих CMS. URL: https://www.ukraine.com.ua/blog/site-administration/chto-takoe-cms-obzor-luchshih-cms.html#Найпопулярніші%20CMS (дата звернення : 20.02.2025)

16. Гайна Г.А. Основи проектування баз даних. Навчальний посібник. Київ: Кондор,. 2018.

17. NinjaMock. Інструмент для створення каркасів і прототипів. Веб-сайт. URL : https://ninjamock.com/uk/ (дата звернення : 02.03.2025)

18. Figma: The Collaborative Interface Design Tool. Веб-сайт. URL : https://www.figma.com (дата звернення : 02.03.2025)

19. Робота з кольором в дизайні. Веб-сайт. https://cases.media/article/robota-z-kolorom-v-dizaini (дата звернення : 02.03.2025)

20. TIOBE Index for March 2025. Веб-сайт. URL : https://www.tiobe.com/tiobe-index/ (дата звернення : 02.03.2025)

21. Рейтинг мов програмування 2025. Веб-сайт. URL : https://dou.ua/lenta/articles/language-rating-2025/ (дата звернення : 02.03.2025)

22. Как менялся топ самых популярных языков GitHub последние 10 лет. Веб-сайт. URL: https://devby.io/news/kak-menyalsya-top-samyh-populyarnyh-yazykov-github-poslednie-10-let (дата звернення : 02.03.2025)

23. Usage statistics of server-side programming languages for websites. Веб-сайт. URL: https://w3techs.com/technologies/overview/programming_language (дата звернення : 02.03.2025)

24. PHP проти Python: Що обрати для веб-розробки? Веб-сайт. URL: https://kwo-master.com/blog/nedostatki-python/ (дата звернення : 02.03.2025)

25. PHP. Веб-сайт. URL: https://www.php.net (дата звернення : 02.03.2025)

26. PhpStorm. The Lightning-Smart PHP IDE. Веб-сайт. URL: https://www.jetbrains.com/phpstorm/ (дата звернення : 02.03.2025)

27. Visual Studio Code – Code Editing. Redefined. Веб-сайт. URL: https://code.visualstudio.com (дата звернення : 02.03.2025)

28. Популярні PHP фреймворки: як вибрати? Веб-сайт. URL: https://foxminded.ua/php-freimvorky/ (дата звернення : 02.03.2025)

29. Laravel. The PHP Framework for Web Artisans. Веб-сайт. URL: https://laravel.com (дата звернення : 02.03.2025)

30. VS Code как альтернатива PhpStorm. URL: https://dev.to/wildtuna/visual-studio-code-kak-altiernativa-phpstorm-dlia-backend-razrabotchika-1d7k (дата звернення : 02.03.2025)

31. Composer. Веб-сайт. URL: https://getcomposer.org (дата звернення : 02.03.2025)

32. Build fast, responsive sites with Bootstrap. Веб-сайт. URL: https://getbootstrap.com (дата звернення : 02.03.2025)

33. Web technology for developers. Documentation for Web developers. Веб-сайт. URL: https://developer.mozilla.org/en-US/docs/Web (дата звернення : 07.04.2025)

34. Icons. URL: https://www.iconfinder.com/icons (дата звернення : 02.11.2025)

35. Електронна медична карта: як створити та працювати в Україні URL:: https://medplatforma.com.ua/article/1769-elektronna-medichna-karta-yak-vprovadjuvatimut https://zakon.rada.gov.ua/laws/show/411-2018-п – Text (дата звернення : 02.03.2025)

36. Зареєструвати домен. URL: https://hostiq.ua/ukr/domains/ (дата звернення : 02.04.2025)