Міністерство освіти і науки України

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління
(повна назва)

Кафедра електронних обчислювальних машин
(повна назва)

КВАЛІФІКАЦІЙНА РОБОТА

Пояснювальна записка

Рівень вищої освіти перший (бакалаврський)

Програмний застосунок для автоматизації заповнення медичних документів

(тема)

Виконав:

здобувач 3 року навчання,

групи КІУКІу-22-1

Анастасія СМОТРОВА
(власне ім’я, прізвище)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва спеціальності)

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

Освітня програма Комп’ютерна інженерія
(повна назва освітньої програми)

Керівник: доц. Георгій ІВАЩЕНКО
(посада, власне ім’я, прізвище)

Допускається до захисту

Завідувач кафедри ЕОМ _____ Андрій КОВАЛЕНКО
(підпис) (власне ім’я, прізвище)

2025 р.

Харківський національний університет радіоелектроніки

Факультет комп’ютерної інженерії та управління

Кафедра електронних обчислювальних машин

Рівень вищої освіти перший (бакалаврський)

Спеціальність 123 «Комп’ютерна інженерія»
(код і повна назва)

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

Освітня програма Комп’ютерна інженерія
(повна назва)

ЗАТВЕРДЖУЮ:
Зав. кафедри _____
(підпис)
“____” _____ 20__ р.

ЗАВДАННЯ

НА КВАЛІФІКАЦІЙНУ РОБОТУ

здобувачеві Смотровій Анастасії Олександрівні
(прізвище, ім’я, по батькові)

1. Тема роботи Програмний застосунок для автоматизації заповнення медичних документів

затверджена наказом по університету від “ 26 ” травня 2025 р. № 425 Ст

2. Термін подання здобувачем роботи до екзаменаційної комісії 14 липня 2025 р.

3. Вхідні дані до роботи 1) документація мови програмування C#; 2) опис засобів WPF; 3) шаблон медичної виписки ОГП ОЦО; 4) шаблон медичної виписки №066.

4. Перелік питань, що потрібно опрацювати у роботі
1) аналіз проблеми та огляд існуючих рішень;
2) аналіз поширених технологій розробки та інструментальних засобів;
3) створення графічного інтерфейсу;
4) розробка програмних модулів;
5) інструкція використання для користувача;
6) висновки.

5. Перелік графічного матеріалу із зазначенням креслеників, схем, плакатів, комп'ютерних ілюстрацій

Слайд-презентація – 12 слайдів

6. Консультанти розділів роботи (заповнюється за наявності консультантів згідно з наказом, зазначеним у п.1)

Найменування розділу	Консультант (посада, прізвище, ім'я, по батькові)	Позначка консультанта про виконання розділу	
		підпис	дата

КАЛЕНДАРНИЙ ПЛАН

№	Назва етапів роботи	Строк / терміни виконання етапів роботи	Примітка
1	Аналіз проблеми та огляд існуючих рішень	10.06.25-13.06.25	
2	Вибір технології розробки та	14.06.25-17.06.25	
	інструментальних засобів		
3	Розробка алгоритмічного забезпечення	18.06.25-21.06.25	
4	Розробка програмних модулів	23.06.25-28.06.25	
5	Відлагодження програмних модулів	30.06.25-02.07.25	
6	Оформлення матеріалів кваліфікаційної роботи	03.07.25-05.07.25	
7	Подання кваліфікаційної роботи керівникові та	07.07.25-09.07.25	
	її попередній захист		
8	Подання кваліфікаційної роботи на рецензу-	10.07.25-11.07.25	
	вання		

Дата видачі завдання " 09 " червня 2025 р.

Здобувач _____
(підпис)

Керівник роботи _____ доц. Георгій ІВАЩЕНКО
(підпис) (посада, власне ім'я, прізвище)

РЕФЕРАТ

Пояснювальна записка кваліфікаційної роботи: 107 с., 30 рис., 1 табл., 2 дод., 31 джерел.

МЕДИЧНИЙ ДОКУМЕНТООБІГ, CRM-СИСТЕМА, ІНТЕРФЕЙС КОРИСТУВАЧА, WPF, БАЗИ ДАНИХ, MONGODB, ГЕНЕРАЦІЯ ДОКУМЕНТІВ, EASYDOX.

Метою кваліфікаційної роботи є розробка програмного застосунку, що призначений для вирішення актуальної сучасної проблеми надмірного бюрократичного навантаження працівників медичної сфери за допомогою автоматизації процесу генерації медичних документів.

У ході виконання кваліфікаційної роботи були реалізовані наступні етапи: детальний аналіз обраної предметної області, дослідження альтернативних інструментів автоматизації процесів управління даними, формування функціональних вимог до застосунку на основі проведеного аналізу, розгляд поширених технологій розробки, реалізація інтерфейсу користувача та програмної логіки. Використано технологію розробки WPF, нереляційну СКБД MongoDB та бібліотеку для генерації документів EasyDox.

ABSTRACT

Bachelor’s thesis: 107 pages, 30 figures, 1 tables, 2 appendices, 31 sources.

MEDICAL DOCUMENT FLOW, CRM SYSTEM, USER INTERFACE, WPF, DATABASES, MONGODB, DOCUMENT GENERATION, EASYDOX.

The primary goal of the qualification work is to develop a software application designed to solve the present-day problem of excessive bureaucratic workload of medical professionals by automating the process of generating medical documents.

In the course of the qualification work, the following stages were implemented: a detailed analysis of the selected subject area, research of alternative tools for automating data management processes, preparation of functional requirements for the application, consideration of standard development technologies, implementation of the user interface and program logic. The following instruments were used: WPF development technology, MongoDB non-relational database, and EasyDox document generation tool.

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

МКХ – Міжнародна класифікація хвороб

МОЗ – Міністерство охорони здоров’я

ASP.NET – активні сторінки серверу для .NET (англ., Active Server Pages for .NET)

BSON – бінарна нотація об'єктів JavaScript (англ., Binary JavaScript Object Notation)

CI/CD – безперервна інтеграція та безперервне постачання (англ., Continuous Integration and Continuous Delivery/Deployment)

CRM – управління взаємовідносинами з клієнтами (англ., Customer Relationship Management)

ERP – планування ресурсів підприємства (англ., Enterprise Resource Planning)

GDPR – загальний регламент про захист даних (англ., General Data Protection Regulation)

GUI – графічний інтерфейс користувача (англ., Graphical User Interface)

HIPAA – закон про мобільність та відповідальність медичного страхування (англ., Health Insurance Portability and Accountability Act)

JSON – нотація об'єктів JavaScript (англ., JavaScript Object Notation)

XAML – розширювана мова розмітки для застосунків (англ., eXtensible Application Markup Language)

ВСТУП

Використання сучасних технологій стає невід’ємною складовою в усіх сферах життя. У деяких галузях, таких як медицина, оперативний доступ до інформації має особливе значення, оскільки може стати умовою для надання якісної лікарської допомоги. Саме тому реалізація систем автоматизації лікарського документообігу для забезпечення стабільного та безпечного керування документами суттєво підвищує ефективність у роботі медичних закладів [1, 2]. Інтеграція системи документообігу впливає на рівень надання медичних послуг, оскільки у такому випадку медичний персонал може скоротити час за рахунок використання електронного інформаційного ресурсу замість традиційної картотеки.

Наявні програмні рішення не завжди здатні повністю задовольнити спеціалізовані вимоги медичних установ. Поширені засоби мають обмеження щодо індивідуального налаштування відповідно до внутрішніх процесів, інтеграції з локальними інформаційними системами або потребують значних фінансових витрат на впровадження та підтримку.

Одним з аспектів автоматизації документообігу лікарні є забезпечення зберігання та управління даними пацієнтів, які описуються в медичних картах. Реалізація системи автоматизації заповнення такої медичної документації дозволить помітно скоротити час для обробки документів, зменшити вірогідність помилок та забезпечити оперативний доступ до необхідних даних. У якості засобів розробки програмного продукту доцільно застосувати мову програмування C# та технологію WPF (Windows Presentation Foundation). Технологія WPF надає інструменти для створення зручного інтерфейсу користувача. Широкий спектр можливостей налаштування візуального оформлення елементів сприяє прискоренню розробки та подальшій підтримці з урахуванням потреб користувачів.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Опис задачі

Адміністративна робота, пов’язана з оформленням медичної документації, становить значну частину щоденних обов’язків медичних працівників, що може ускладнити надання медичних послуг та збільшити ймовірність помилкових дій [3].

Дослідження [2], проведене на базі медичної установи, демонструє, що використання системи електронного документообігу дозволило зменшити час, необхідний для опрацювання медичних записів на 50-60%. Розглянуті у дослідженні програмні рішення дозволили оптимізувати робочі процеси лікарських установ та водночас позитивно вплинули на якість надання медичної допомоги, що є особливо важливою складовою удосконалення системи охорони здоров’я.

Існуючі системи автоматизації не завжди охоплюють усі етапи обробки медичних документів, особливо в частині локалізованих вимог кожного конкретного закладу. У багатьох випадках програмне забезпечення або неповністю адаптоване до української медичної практики [4], або не підтримує гнучкі механізми налаштування шаблонів документів згідно з вимогами МОЗ.

Крім того, відсутність єдиного підходу до стандартизації електронних форм ускладнює процес впровадження автоматизованого обліку в різних лікарнях, що призводить до дублювання даних, неузгодженості в записах та порушення структури ведення документації. Тому виникає потреба у створенні програмного застосунку, який дозволяє медичному персоналу формувати, редагувати, зберігати та швидко знаходити лікарські документи з меншими витратами часу та зусиль. Також важливо забезпечити можливість подальшого аналізу введених даних, створення звітів та експорт.

1.2 Аналіз існуючих рішень

Сучасні електронні системи документообігу використовуються як невід'ємний інструмент для автоматизації процесів управління даними [5]. Головна мета впровадження систем автоматизації – зростання ефективності щоденної робочої діяльності персоналу [6]. Вони надають можливість зручного зберігання різноманітних даних, що у поєднанні з простотою використання дозволяє помітно зменшити часові витрати на пошук та опрацювання інформаційного контенту.

Важливою особливістю систем електронного документообігу є високий ступінь захисту даних, реалізований через впровадження шифрування та механізмів авторизації користувачів, що має важливе значення в контексті медичної інформації у лікарських закладах [7]. Застосування електронно-цифрових підписів, функції автоматизованих сповіщень і механізми маршрутизації документів забезпечують зменшення навантаження персоналу і підвищення якості виконання медичних процедур.

Завдяки інтеграції з корпоративними платформами ERP та CRM [8, 9], системи реалізують єдиний безперервний потік інформації між функціональними блоками, що позитивно впливає на рівень узгодженості дій персоналу та покращення комплексної продуктивності.

Згідно з результатами досліджень [10], застосування систем електронного документообігу значно полегшує керування потоками даних та забезпечує більш раціональне залучення ресурсного потенціалу організації.

Для формування переліку функціональних можливостей розроблюваного програмного засобу, необхідно проаналізувати типовий функціонал існуючих аналогічних рішень. У процесі аналізу існуючих програмних засобів для автоматизації документообігу розглянуто найбільш поширені програмні рішення: медична платформа Helsi, автоматизована система управління документами Megapolis.DocNet, хмарна платформа DocuWare та масштабована інформаційна система OpenText.

1.2.1 Застосунок «Helsi»

Провідна українська інформаційна платформа Helsi орієнтована на потреби пацієнтів, медичного персоналу, а також установ охорони здоров'я незалежно від форми власності [11]. Система істотно підвищує доступність медичних послуг для населення, надаючи пацієнтам можливість переглядати власні медичні дані, реєструватися на вакцинацію, записуватись до бажаного лікаря, резервувати лікарські засоби та ознайомлюватись з електронними рецептами (рисунок 1.1).

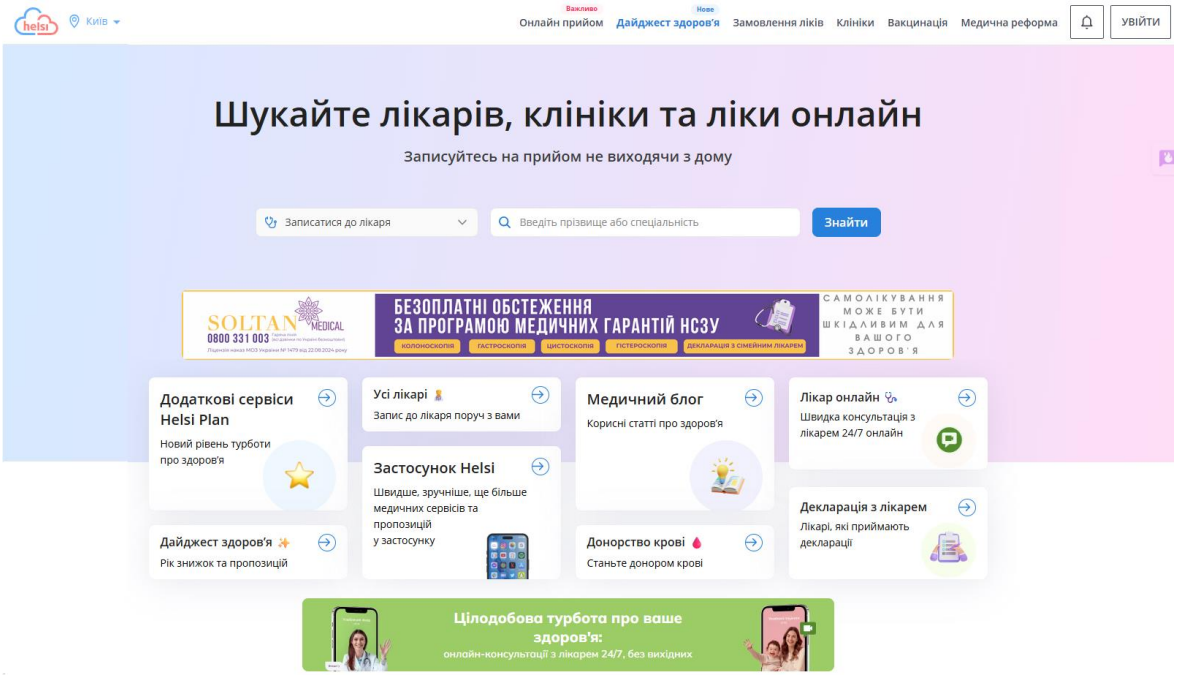

Рисунок 1.1 – Головна сторінка застосунку Helsi

Застосунок Helsi створений для підвищенню зручності взаємодії пацієнта з медичним закладом, зокрема шляхом попереднього запису, доступ до інформації про лікарів чи медичну установу та можливість проведення онлайн-консультації. На сьогоднішній день системою Helsi активно користуються понад 24 мільйони громадян України, що становить близько 55% загальної чисельності населення. Кожного місяця через платформу здійснюється більше ніж 2,5 мільйони записів на прийом до медичних

фахівців. Доступ до системи можливий через офіційний сайт або за допомогою мобільного застосунку, сумісного з операційними системами Android та iOS [12]. Серед основних переваг використання платформи Helsi для медичних працівників слід зазначити:

- зручність ведення електронної медичної карти та історії хвороб;
- швидкий доступ до результатів лабораторних досліджень та діагностичних процедур;
- можливість гнучкого керування графіком роботи;
- інтуїтивно зрозумілий інтерфейс особистого кабінету, призначеного для організації прийому пацієнтів.

Для медичних установ платформа Helsi пропонує низку функціональних можливостей:

- повна автоматизація внутрішніх процесів медичного закладу;
- гнучке налаштування платформи відносно до специфіки та потреб конкретної установи;
- інструменти для реалізації вимог реформи про «Порядок ведення Реєстру декларацій про вибір лікаря, що надає первинну медичну допомогу» (реєстрація декларації, закріплення пацієнтів за лікарями) [4];
- засоби моніторингу діяльності та формування управлінської статистики;
- автоматизоване створення звітності та статистики.

При використанні системи Helsi фахівці медичних закладів стикаються з певними недоліками. Серед них тривалий час переходу, необхідний для міграції існуючих медичних даних, що часто вимагає додаткової підготовки медичних працівників. Для установ з великим обсягом архівної документації у паперовому вигляді впровадження цієї системи може значно ускладнити міграцію і відповідно буде потребувати супутніх фінансових витрат на підготовку персоналу. Також існують випадки, коли медична установа використовує внутрішні форми та бланки специфічної структури, інтегрування яких у систему Helsi є технічно складною задачею. Відсутність

повної адаптації таких документів може ускладнити робочі процеси працівників та призвести до затримок у проведення медичної допомоги.

Зазначені недоліки є важливими для великих медичних установ, що характеризуються значною кількістю персоналу або обмеженими фінансовими ресурсами. У таких умовах адміністрація може зіткнутись з необхідністю витратити додаткові організаційні та матеріальні ресурси на впровадження Helsi, або повернутись до традиційних, менш ефективних методів ведення медичної документації.

1.2.2 Застосунок «Megapolis.DocNet»

Система Megapolis.DocNet представляє собою сучасний інструмент автоматизації документообігу, що орієнтується на використання в організаціях різного профілю, зокрема в сфері охорони здоров'я [13]. Метою функціонування системи Megapolis.DocNet є забезпечення цілісного інформаційного середовищу організації з чіткою структурою зберігання документів, високим рівнем доступності та захисту від несанкціонованого втручання (рисунок 1.2).

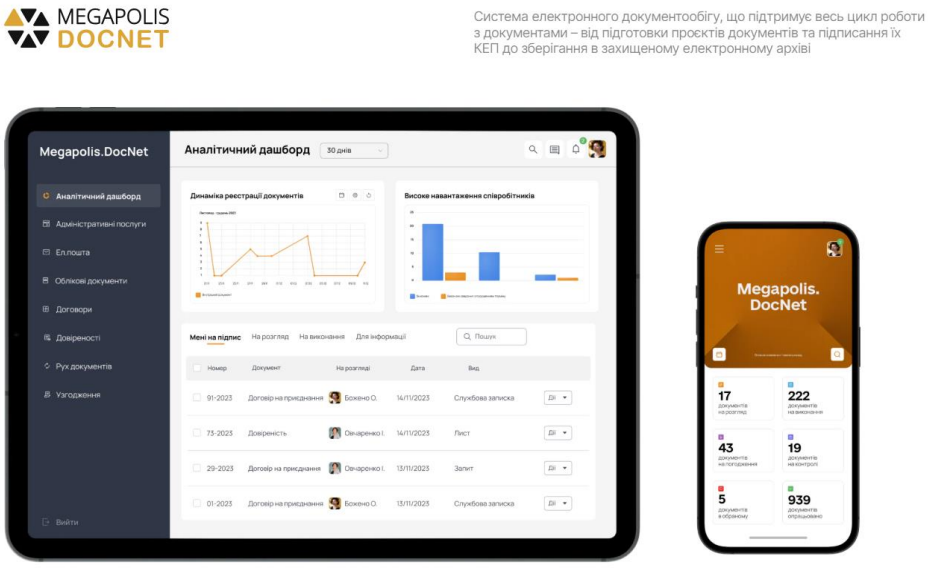

Рисунок 1.2 – Аналітичний дашборд системи Megapolis.DocNet

Завдяки централізованому підходу до збереження та опрацювання документів, система істотно збільшує ефективність функціонування установ та підвищує рівень прозорості їхньої діяльності.

Megapolis.DocNet забезпечує автоматизацію основних функцій документообігу, зокрема генерація, узгодження, верифікація та збереження документів, що сприяє суттєвій оптимізації часових та трудових ресурсів персоналу. Система містить інтегрований модуль для створення звітів та аналітики, що забезпечує зручне формування необхідної звітної документації та проведення внутрішнього аналізу даних.

Однією з головних переваг Megapolis.DocNet є можливість інтеграції з іншими програмними системами, що функціонують в організації. Така сумісність забезпечує безперервний обмін даними між різними інформаційними платформами та сприяє формуванню єдиного інформаційного середовища для підвищення ефективності роботи.

Розглянута система керування документацією Megapolis.DocNet сприяє мобільності користувачів, надаючи можливість дистанційного доступу до документів з будь-якого пристрою, що має підключення до Інтернету. Це сприяє гнучкій організації роботи та підвищенню зручності взаємодії з документами. Крім того, система характеризується високим рівнем захисту інформації шляхом реалізації багаторівневого механізму контролю доступу та використанню технологій шифрування, які гарантують конфіденційність та цілісність документів.

Одним із основних обмежень Megapolis.DocNet є висока загальна вартість володіння, яка включає ліцензійні платежі та витрати на впровадження, що можуть бути недоступними для медичних закладів з обмеженим фінансуванням. Окрім цього, процес впровадження вимагає глибокого навчання працівників, що передбачає додаткові значні витрати часу та фінансів.

Сервіс обмежений функціоналом електронного зберігання документів і не забезпечує їх створення чи редагування, тому його необхідно розглядати

як допоміжний інструмент, а не як автономне рішення для ведення медичної документації. Програмний засіб Megapolis.DocNet реалізується як великий проєкт із централізованим супроводом та широкою базою користувачів. У медичних закладів відсутня можливість прямої взаємодії з розробниками, оскільки всі комунікації відбуваються виключно через службу технічної підтримки, що може спричинити затримки у вирішення організаційних та технічних питань.

Отже, при ухваленні рішення про впровадження Megapolis.DocNet важливо зважено оцінити не лише переваги системи, але й проаналізувати можливі обмеження та супутні витрати. Доцільно врахувати рівень фінансових можливостей, кадрової спроможності до адаптації, а також сумісність системи з наявною IT-інфраструктурою.

1.2.3 Застосунок «DocuWare»

DocuWare – це сучасна багатофункціональна хмарна система для керування документами (рисунок 1.3), яка пропонує зручні інструменти для збереження, обробки та управління цифровими документами. Завдяки підтримці автоматизованої обробки завдань та повнотекстового індексування, DocuWare є інструментом для підвищення ефективності роботи з документами. Хмарна інфраструктура DocuWare Cloud відповідає стандартам інформаційної безпеки HIPAA ТА GDPR, та гарантує дистанційний доступ до даних у цілодобовому режимі [14].

Система надає адаптивну модель ліцензування за підпискою, що забезпечує гнучкість для компаній різного розміру. Кожен тарифний план включає розширений набір функцій, зокрема інтелектуальне індексування, управління робочими процесами, створення форм та інструменти інтеграції. Такий рівень універсальності сприяє ефективній автоматизації процесів орієнтованих на обробку документів і підвищенню загальної продуктивності організації.

DocuWare відрізняється високою надійністю та масштабованістю, надаючи захищене хмарне середовище для зберігання значних обсягів інформації, що дозволяє отримувати доступ до документів незалежно від місця перебування та часу. Додатково реалізована можливість створення резервних копій та відновлення інформації, що мінімізує ризики втрати важливих даних у непередбачуваних ситуаціях. Система також забезпечує сумісність із різними типами файлів, що робить можливим зберігання та обробку як текстових документів, так і таблиць, презентацій, зображень та відеофайлів.

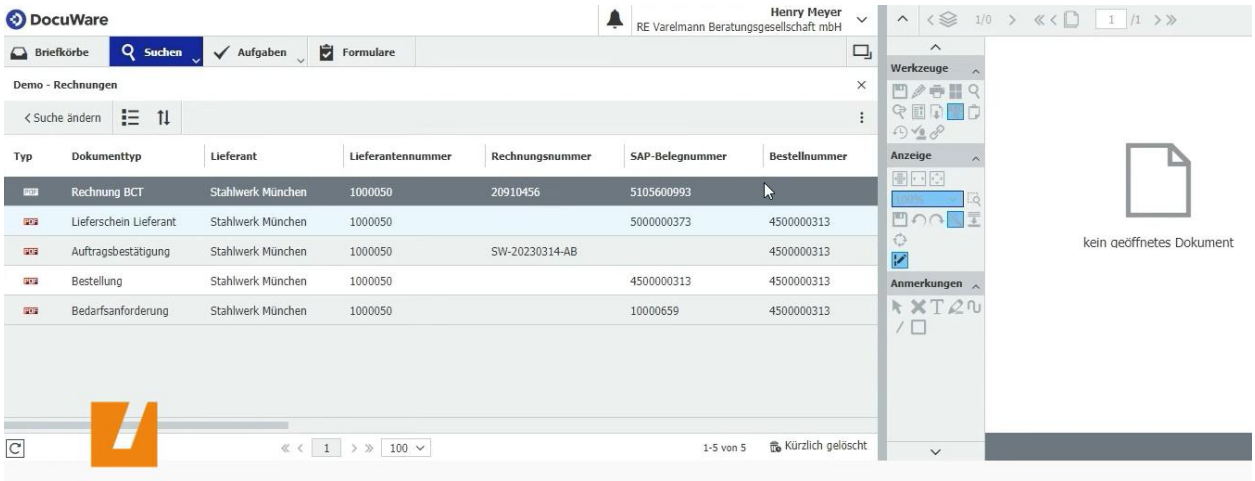

Рисунок 1.3 – Інтерфейс застосунку DocuWare

Серед основних переваг розглянутого програмного застосунку можна виокремити високий рівень захисту даних, досягнутий за рахунок використання сучасних криптографічних засобів, а також широкі можливості інтеграції з різноманітними бізнес-застосунками, що забезпечує гнучкий та зручний доступ до документів з будь-якого пристрою, підключеного до Інтернету.

До недоліків застосунку можна віднести непрозору модель ціноутворення, що передбачає необхідність індивідуального запиту для отримання комерційної інформації, а також високій поріг входу для нових користувачів через складність інтерфейсу.

1.2.4 Застосунок «OpenText»

Платформа OpenText представляє собою гнучке та масштабоване рішення, яке надає комплекс інструментів для ефективного керування документами й інформаційними процесами в організації (рисунок 1.4).

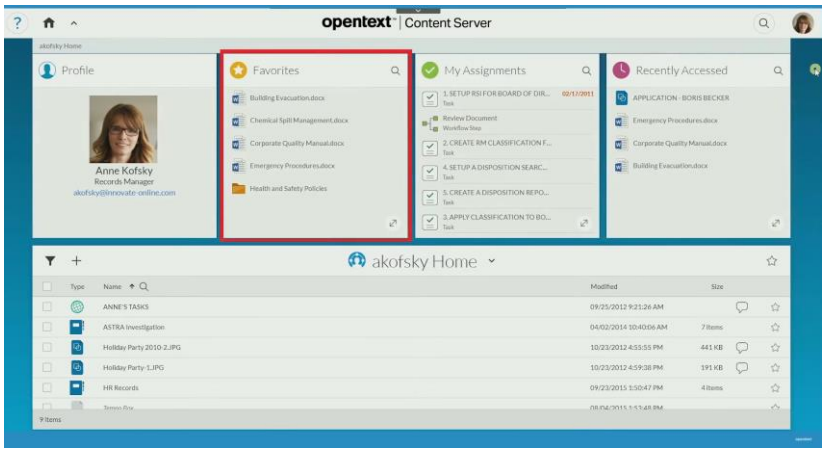

Рисунок 1.4 – Інтерфейс застосунку «OpenText»

Платформа розроблена з урахуванням специфічних потреб великих корпоративних структур, що мають високі стандарти щодо організації документообігу та обробки даних [15]. Завдяки високій адаптивності та розвиненому функціоналу, платформа забезпечує інтеграцію з різноманітними корпоративними системами, що дозволяє ефективно керувати інформаційними потоками навіть у складних бізнес-сценаріях.

Основні функціональні можливості системи включають:

- платформа забезпечує інтеграцію з провідними корпоративними системами, такими як Microsoft і SAP, що сприяє покращенню управління бізнес-процесами;

- платформа забезпечує високий рівень інформаційної безпеки та управління ризиками, пов’язаних з обробкою та зберіганням даних;

- платформа використовує технології обчислювального інтелекту для аналізу великих обсягів інформації, що сприяє оптимізації внутрішніх процесів;

- завдяки інструментам оптимізації даних, система сприяє раціональному використанню інформаційних ресурсів, що є актуальним для організацій з великими обсягами даних;

- платформа підтримує автоматизацію повного життєвого циклу HR-процесів, включаючи створення, подання, погодження, підписання документів з використанням електронного підпису, контроль виконання, переведення в архів і формування звітності;

- платформа підтримує інструменти для аналізу великих обсягів інформації.

На основі аналізу системи OpenText було визначено її основні переваги та недоліки. Серед переваг можна виокремити інтелектуальну автоматизацію, підтримку інтеграції з хмарними сервісами, обробку великих даних за допомогою засобів обчислювального інтелекту, високий рівень інформаційної безпеки та ефективну автоматизацію документообігу.

Серед основних недоліків системи слід зазначити тривалий період навчання, зумовлений загальною складністю інтерфейсу, платну ліцензійну політику, яка передбачає оплату навіть за демо-версію, а також відсутність підтримки кросплатформеного доступу.

1.2.5 Підсумки аналізу існуючих рішень

У результаті аналізу визначених недоліків та переваг поширених застосунків сформовані вимоги до розроблюваного засобу для автоматизації заповнення медичних документів, що включає в себе наявність автоматизації формування документів, підтримку багаторівневої аутентифікації та можливість локального збереження даних пацієнтів для роботи в офлайн-режимі. Проведений порівняльний аналіз таких аналогів систем автоматизації документообігу:

- медична платформа Helsi (№1 у таблиці 1.1);
- автоматизована система Megapolis.DocNet (№2 у таблиці 1.1);

- хмарна платформа DocuWare (№3 у таблиці 1.1);
- масштабована інформаційна система OpenText (№4 у таблиці 1.1).

Результати порівняльного аналізу представлені у таблиці 1.1. Знак «+» означає наявність відповідних можливостей у сервісі, знак «-» – означає його відсутність.

Таблиця 1.1 – Порівняльна характеристика існуючих систем

Критерії	№1	№2	№3	№4
Збереження документів	+	+	+	+
Створення та редагування документів	+	-	+	+
Інтеграція з іншими системами документообігу	+	+	+	+
Дистанційний доступ до документів	+	+	+	+
Інтуїтивність застосунку (user-friendly)	+	+	-	-
Швидке навчання працівників	+	-	-	-
Високий рівень захисту інформації	+	+	+	+
Безкоштовне використання	+	-	-	-
Масштабованість системи	+	+	+	+
Кросплатформеність системи	+	+	+	-
Підтримка створення резервної копії та відновлення інформації	-	+	+	+
Сумісність із різними типами документів	-	+	+	+
Можливість додавати внутрішні форми та бланки специфічної структури	-	-	+	+

1.3 Постановка задачі

Проєкт орієнтований на створення програмного продукту з простим та зрозумілим інтерфейсом для автоматизації заповнення медичних документів. Система виконує роль електронної платформи для зберігання інформації про пацієнтів та подальшого використання цих даних при створенні звітної документації, що сприятиме підвищенню продуктивності роботи.

Основна мета проєкту спрямована на оптимізацію процесів роботи з медичними документами через впровадження засобів для їх збереження та обробки. Цифровий каталог надає змогу систематизувати медичні записи, що полегшує процес створення, оновлення та пошуку необхідної інформації. Проєкт передбачає реалізацію низки технічних та нормативних вимог, необхідних для його повноцінного функціонування:

- проаналізувати необхідність і доцільність розробки застосунку;
- ознайомитися з функціональністю актуальних рішень на ринку з метою подальшого порівняльного аналізу їх ефективності;
- визначити функціональні компоненти системи, які забезпечать її перевагу порівняно з наявними аналогами;
- обґрунтувати вибір технологічного стеку для розробки застосунку;
- реалізація механізмів авторизації та аутентифікації користувачів із підтримкою кількох методів доступу та забезпеченням захисту даних;
- безпосередня реалізація функціональних модулів системи;
- здійснення перевірки програмного забезпечення на предмет дотримання цілісності та коректності структури даних.

Підходи до вирішення поставленої задачі ґрунтуються на застосуванні сучасних технологій програмної інженерії, що сприяє підвищенню ефективності при побудові інтерфейсів користувача. Таким чином, реалізація даного проєкту дозволяє оптимізувати робочі процеси лікарських установ та водночас позитивно впливає на якість надання медичної допомоги, що є особливо важливою складовою удосконалення системи охорони здоров’я.

2 АНАЛІЗ ВИКОРИСТАНИХ ТЕХНОЛОГІЙ

2.1 Засоби розробки

Спостерігається тенденція до заміни терміна «засоби розробки» на «технологічний стек» (англ. tech stack). Це поняття описується як комплекс технічних інструментів, до якого входять мови програмування, сервіси, фреймворки та бібліотеки, що використовуються на всіх етапах створення програмного застосунку [16].

До найбільш розповсюджених мов програмування відносяться C/C++, C#, JavaScript, Java, Python, Dart, Rust. Кожна з них має спеціалізовані фреймворки та бібліотеки, зокрема React, Angular, Vue, ASP.NET, WPF, Flutter, Entity Framework, що розширюють їхні можливості.

Процес створення програмного забезпечення вимагає застосування відповідних інструментів програмування, передусім редакторів коду та інтегрованих середовищ розробки (IDE). До найбільш популярних належать Visual Studio, NetBeans, PyCharm, Rider, WebStorm та інші.

Відповідно до визначених цілей та функціональних вимог застосунку, для його розробки обрано такий технічний стек:

- Visual Studio 2022, середовище для розробки програмного забезпечення випущене у 2021 році;
- мова програмування C# версії 10, яка використовується для розробки застосунків різного масштабу та призначення [17];
- платформа .NET версії 8, яка підтримує десяту версію мови програмування C#, випущена у 2023 році [18];
- Windows Presentation Foundation (WPF) – фреймворк, що надає інструменти для розробки програмних застосунків з графічним інтерфейсом на мовах C# та XAML.

2.2 Розробка Windows застосунків за допомогою фреймворку WPF

Розробка застосунків із використанням фреймворку Windows Presentation Foundation (WPF) ґрунтується на інтеграції мови розмітки XAML (eXtensible Application Markup Language) [19], яка відповідає за формування інтерфейсу користувача, та мови програмування C#, що використовується для реалізації програмної логіки.

Фреймворк WPF забезпечує розробника інструментами для створення графічного інтерфейсу користувача, зокрема контейнерами для розміщення елементів у межах вікна, елементами керування для реалізації взаємодії з користувачем, механізмами стилізації та шаблонізації, а також засобами прив'язки даних (англ. data binding), що дають змогу автоматично оновлювати інтерфейс згідно зі змінами у джерелі даних.

WPF також надає засоби для реалізації графічних можливостей, зокрема відображення векторної та растрової графіки, підтримки анімацій і візуальних ефектів [20]. Серед інших важливих аспектів розробки з використанням фреймворку WPF слід зазначити реалізацію взаємодії з користувачем за допомогою подій, а також застосування шаблону проектування MVVM (Model-View-ViewModel) (рисунок 2.1) для забезпечення структурованості коду та відокремлення логіки програми.

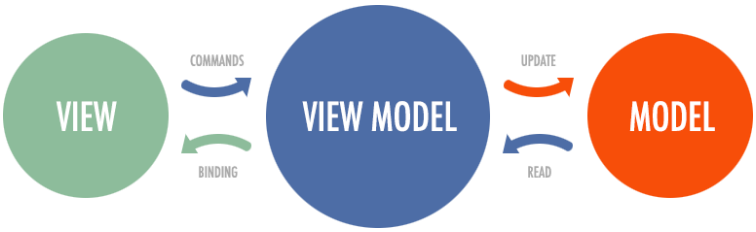

Рисунок 2.1 – Компоненти шаблону проектування MVVM

Розробка застосунків із використанням WPF вимагає від розробника володіння мовою програмування C#, розуміння основ GUI-програмування та практичний досвід роботи з компонентами фреймворку [20].

2.3 Використання бібліотек та NuGet пакетів у застосунках C#

Система керування пакетами NuGet є основним інструментом пошуку, завантаження та інтеграції бібліотек із відкритим кодом до проєктів .NET [21]. NuGet надає зручні засоби для керування зовнішніми бібліотеками у проектах C#, включаючи їх швидке підключення та оновлення. Використання таких пакетів у застосунках розширює функціональні можливості програмного забезпечення та спрощує інтеграцію сторонніх бібліотек і сервісів у процесі розробки [22].

2.3.1 Використання пакету «EasyDox»

NuGet-пакет EasyDox призначений для автоматизації процесу заповнення шаблонів документів Word формату .docx у застосунках на C#. Він дозволяє замінювати плейсхолдери у шаблоні на конкретні значення, що зручно для створення персоналізованих документів, таких як договори, сертифікати, звіти та інші. Механізм використання EasyDox базується на завантаженні шаблону документа, у якому вказано умовні поля у вигляді ключових міток. Далі, за допомогою вбудованого класу TemplateProcessor, виконується заповнення шаблону шляхом передачі словника ключ-значення, що відповідає конкретним змінним у документі. Результат обробки зберігається у новому .docx-файлі, який готовий до подальшого використання. Такий підхід дозволяє суттєво зменшити часові витрати при створенні документів однотипної структури.

Для розширення функціоналу є додатковий пакет EasyDox.Morpher, який надає можливість здійснювати граматичну обробку тексту, зокрема відмінювання слів та конвертацію числових значень у текстовий формат.

Таким чином, EasyDox є ефективним інструментом для автоматизації процесів роботи з документами Word формату .docx у програмному забезпеченні, орієнтованому на роботу з текстовими даними.

2.4 MongoDB

MongoDB є документоорієнтованою системою управління базами даних, яка зберігає інформацію у форматі документів, що характеризуються як структура типу ключ-значення. Документи зберігаються у форматі BSON (Binary JSON) – двійковому представленні JSON (JavaScript Object Notation), яке дозволяє ефективно обробляти складні, вкладені структури даних і зберігати типізовані значення [23].

У документоорієнтованій моделі база даних формується з колекцій, кожна з яких може містити множину документів зі змінною структурою та різними наборами атрибутів. Використання BSON як основного формату зберігання забезпечує більш швидкий доступ і компактне представлення даних у порівнянні з JSON (рисунок 2.2). Така організація дозволяє легко адаптуватися до змінних вимог і масштабів проєкту при роботі з неструктурованими або слабо структурованими даними [24].

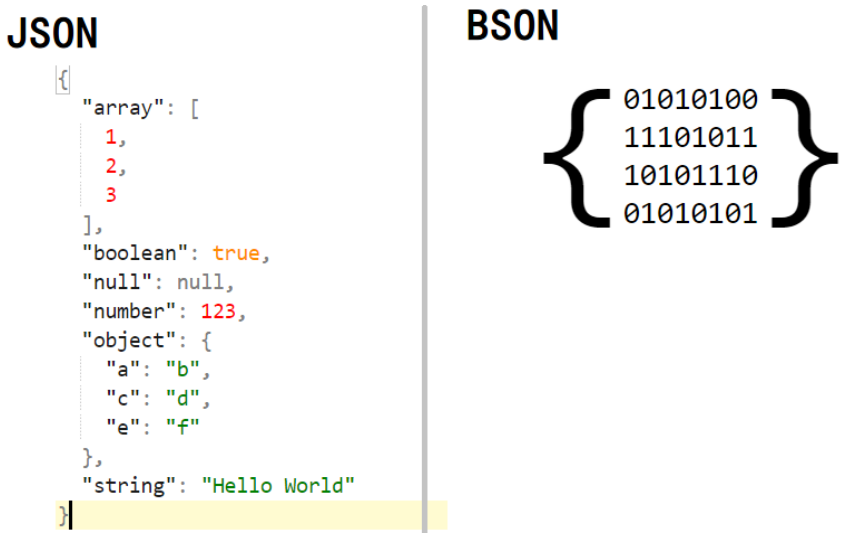

Рисунок 2.2 – Порівняння JSON та BSON форматів

Завдяки зазначеним особливостям MongoDB є ефективним інструментом для побудови високопродуктивних, масштабованих систем, орієнтованих на зберігання та обробку слабо структурованих даних.

2.5 Використання GitHub

GitHub є сучасною хмарною платформою для розробників, яка забезпечує зберігання, управління та спільну роботу над програмним кодом з використанням системи контролю версій Git. Платформа підтримує як відкриті, так і приватні репозиторії, що дозволяє адаптувати її до потреб як індивідуальних розробників, так і великих корпоративних команд [25].

Однією з основних функціональних переваг платформи GitHub є підтримка системи pull-запитів для рецензування коду, а також вбудована інтеграція з платформою GitHub Actions, яка забезпечує реалізацію процесів CI/CD. GitHub Actions дозволяє автоматизувати різні етапи життєвого циклу розробки програмного забезпечення, такі як тестування, деплоймент, а також повідомлення про результати виконання робочих процесів. Така автоматизація дає змогу виявляти помилки на ранніх стадіях, забезпечує стабільність доставки оновлень і сприяє підвищенню якості програмного продукту. Також GitHub надає інструменти для управління задачами, ведення технічних обговорень, хостингу статичних сайтів через GitHub Pages, а також створення хмарних середовищ розробки за допомогою GitHub Codespaces, що сприяє зростанню продуктивності роботи команд [26].

Організація командної роботи за допомогою GitHub базується на принципах розподіленої розробки. Кожен розробник команди працює з локальною копією репозиторію, вносить необхідні оновлення у окремих гілках (англ. branches) та створює pull-запити (англ. pull requests) для внесення змін в основну гілку проєкту. Такий підхід дає змогу досягти контрольованого внесення оновлень, рецензування коду, фіксацію історії змін, а також зменшує ймовірність виникнення конфліктів при об’єднанні (рисунок 2.3). Платформа надає різні механізми безпеки, включаючи сканування коду на вразливості та моніторинг залежностей. Ці функції допомагають розробникам виявляти та усувати потенційні загрози на ранніх етапах розробки, забезпечуючи надійність та захист програмних продуктів.

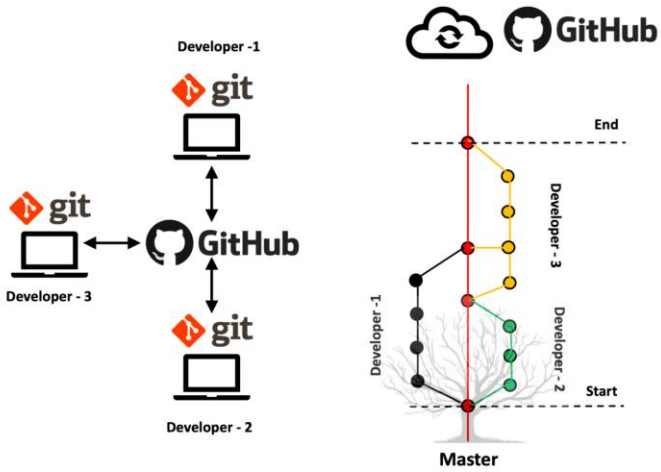

Рисунок 2.3 – Організація командної роботи з GitHub

GitHub також пропонує вбудовані інструменти для створення й розподілу задач (англ. issues), комунікації між учасниками, а також використання візуальних панелей проєктів (англ. projects) для організації етапів розробки. Такий підхід сприяє узгодженості, прозорості та підвищенню якості командної взаємодії в межах програмного проєкту.

Крім використання git для контролю версій коду проекту, його можна використовувати для координації того, як учасники команди працюють над проектом. Цього можна досягти, застосувавши одну з існуючих моделей розгалуження, таких як Centralized workflow, Feature Branch Workflow, Gitflow Workflow або Forking Workflow.

Centralized workflow – це модель розгалуження, у якій робочий процес (англ. workflow) використовує центральне сховище, яке слугує єдиною точкою входу для всіх змін у проекті. Гілка розробки за замовчуванням називається main, і всі зміни фіксуються у цій гілці. Ця модель не потребує жодних інших гілок, окрім основної. Репозиторій може бути локальним, без віддалених копій або зберігатися віддалено, де він може бути клонований або запущений [27].

Розробники починають з клонування центрального сховища. У своїх локальних копіях проекту вони редагують файли і фіксують зміни, але нові комміти зберігаються локально, вони повністю ізольовані від центрального

сховища. Це дозволяє розробникам відкласти синхронізацію висхідного потоку до того моменту, коли вони досягнуть зручної точки зупинки. Щоб опублікувати зміни в офіційному проекті, розробники завантажують (англ. push) свою локальну основну гілку до центрального сховища.

Centralized workflow є основою для інших варіантів побудови робочих процесів Git. Більшість популярних робочих процесів Git мають певний тип централізованого репозиторію, з якого окремі розробники будуть завантажувати та отримувати дані.

Особливості Feature Branch Workflow полягають у тому, що вся розробка функціональних можливостей повинна відбуватися у виділеній гілці, а не в основній гілці [28]. Така інкапсуляція дозволяє декільком розробникам зручно працювати над певною задачею, не порушуючи основну кодову базу. Це також означає, що головна гілка ніколи не міститиме непрацюючого коду, що є перевагою для середовищ безперервної інтеграції.

Gitflow Workflow – це альтернативна модель розгалуження Git, яка передбачає використання функціональних гілок і декількох первинних гілок. Замість однієї головної гілки цей робочий процес використовує дві гілки для запису історії проекту. Основна гілка (англ. main branch) зберігає історію офіційних релізів, а гілка розробки (англ. develop) слугує гілкою для інтеграції функцій. В такому робочому процесі комміти в основній гілці позначені номером версії [29].

При використанні Forking Workflow замість використання єдиного серверного репозиторію в якості «центральної» кодової бази, кожному розробнику надається власний серверний репозиторій. Це означає, що кожен учасник має не одне, а два сховища Git: приватне локальне і загальнодоступне серверне. Forking Workflow часто використовується в публічних проектах з відкритим вихідним кодом [30].

Завдяки масштабованості та великому набору інструментів, GitHub інтегрувався в розробку програмного забезпечення як основний компонент, що сприяє ефективній командній взаємодії.

3 ПРОГРАМНА РЕАЛІЗАЦІЯ

3.1 Структура проекту

Проекти, написані мовою C# .NET, які використовують фреймворк WPF, мають стандартну структуру, що спрощує розробку масштабованих, ефективних та легких у підтримці застосунків. WPF-проєкти містять кілька компонентів, включаючи головний проект, файли представлення (XAML), файли ресурсів, модельні класи та контролери [20]. В контексті поточної постановки задачі проект включає всі необхідні для запуску елементи. Файлова структура вихідного коду проєкту наведена на рисунку 3.1.

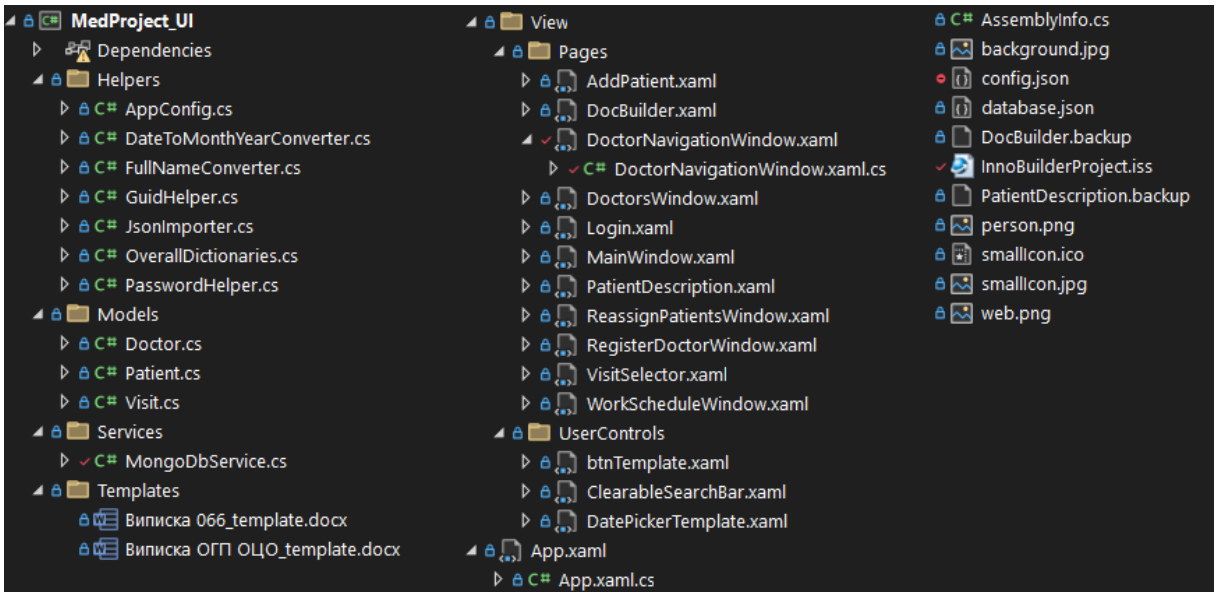

Рисунок 3.1 – Структура проєкту

Особливе значення мають файли DoctorNavigationWindow.xaml.cs, MongoDbService.cs та App.xaml.cs. DoctorNavigationWindow.xaml.cs є головним вікном застосунку, що надає вибір дій для користування застосунком. MongoDbService.cs зберігає необхідні операції для роботи з колекціями у базі даних. App.xaml.cs зберігає сесію та усі словники симптомів.

У WPF представлення створюються за допомогою мови XAML (eXtensible Application Markup Language) – декларативного інструмента для опису інтерфейсу користувача. Кожен XAML-файл супроводжується code-behind файлом, який містить відповідний програмний код [19]. Таким чином, представлення в WPF відокремлене від логіки, що дозволяє досягти поділу елементів архітектури.

Окрім зазначених компонентів, WPF застосунки можуть також включати додаткові елементи, такі як сервіси для обробки спеціалізованих завдань, менеджери навігації для керування переходами між різними вікнами застосунку, NuGet пакети, які являють собою бібліотеки заздалегідь готових рішень, набір ресурсів, таких як рисунки, файли та допоміжні класи. Гнучкість структури WPF-проектів у C# .NET дозволяє створювати сучасні застосунки, зручні як для користувачів, так і для розробників.

3.2 Реєстрація користувача

Під час реєстрації нові користувачі створюють пароль. Клас PasswordHelper реалізує безпечне хешування та перевірки паролів використовуючи вбудовану у C# бібліотеку System.Security.Cryptography, із використанням хеш-функції SHA-256. Основними параметрами класу (лістинг 3.1) є розмір солі 128 біт, довжина ключа 256 біт і кількість ітерацій 10 000, що забезпечує стійкість до атак.

Лістинг 3.1 – Параметри класу PasswordHelper

```
internal static class PasswordHelper
{    private const int SaltSize = 16;
    private const int KeySize = 32;
    private const int Iterations = 10000;…}
```

Метод HashPassword (лістинг 3.2) генерує випадкову сіль, обчислює хеш на основі введеного пароля та додає сіль до хешу, після чого результат кодується у форматі Base64.

Лістинг 3.2 – Функція хешування паролю (файл PasswordHealper.cs)

```
public static string HashPassword(string password)
{  using (var rng = RandomNumberGenerator.Create())
    {   var salt = new byte[SaltSize];
        rng.GetBytes(salt);
        using (var pbkdf2 = new Rfc2898DeriveBytes(password,
salt, Iterations, HashAlgorithmName.SHA256))
        {
       var key = pbkdf2.GetBytes(KeySize);
            var hashParts = new byte[SaltSize + KeySize];
            Buffer.BlockCopy(salt, 0, hashParts, 0, SaltSize);
            Buffer.BlockCopy(key, 0, hashParts, SaltSize,
KeySize);
            return Convert.ToBase64String(hashParts);
        }}}
```

Для прикладу, користувач ввів пароль «GoodPassword!». Його перетворення наступні:

- генерується випадкова сіль обсягом 16 байт, у даному випадку вона моє значення BC 4C D8 86 E8 F3 0E 72 74 AE F6 3D 36 4D 21 DF;

- алгоритм PBKDF2 застосовується до рядка паролю («GoodPassword!») разом із визначною сіллю 10 000 ітерацій, формуючи такий 32-байтний похідний ключ: 96 76 46 90 C0 95 09 F0 0C C9 12 30 1B E5 ED D0 6F D1 B9 1E F9 8D 7B F2 B1 27 19 29 0B 02 34 22;

- формується єдиний 48-байтний масив шляхом конкатенації солі та ключа (байти 0-15 – сіль, байти 16-47 – ключ);

- усі 48 байт кодуються у Base64, що дає наступний рядок: vEzYhujzDnJ0rvY9Nk0h35Z2RpDAlQnwDMkSMBvl7dBv0bke+Y178rEnGSkLAjQi. Отриманий рядок зберігається як зашифрований (хешований) пароль.

Метод VerifyPassword (лістинг 3.3) декодує збережений хеш, витягує з нього сіль і порівнює повторно обчислений хеш з оригінальним для перевірки коректності пароля.

Лістинг 3.3 – Функція перевірки паролю (файл PasswordHealper.cs)

```
public static bool VerifyPassword(string password, string
hashedPassword)
 {  var hashBytes = Convert.FromBase64String(hashedPassword);
     var salt = new byte[SaltSize];
```

```
    Buffer.BlockCopy(hashBytes, 0, salt, 0, SaltSize);
    using (var pbkdf2 = new Rfc2898DeriveBytes(password, salt,
Iterations, HashAlgorithmName.SHA256))
    {
        var key = pbkdf2.GetBytes(KeySize);
        for (var i = 0; i < KeySize; i++)
            if (hashBytes[SaltSize + i] != key[i])
                return false;
        return true;
    } }
```

Завдяки вбудованій підтримці криптографічних засобів .NET, реалізація є надійною та відповідає сучасним вимогам безпеки. Застосування солі унеможливлює використання попередньо обчислених хеш-таблиць (rainbow tables). Такий підхід до обробки облікових даних рекомендований для захисту користувацької автентифікації в програмних системах.

3.3 Заповнення даних пацієнтів

Під час створення або редагування даних про пацієнта частина полів для заповнення має заздалегідь заготовлені відповіді. Клас OverallDictionaries виконує роль централізованого сховища стандартизованих медичних класифікаторів для опису загального стану пацієнта. Він містить набір словників, кожен з яких відповідає окремому клінічному параметру або симптому, представленому у вигляді пари «ключ-значення», де ключ – це ціле число, а значення – текстовий опис медичного стану (лістинг 3.4).

Лістинг 3.4 – Словники для зберігання симптомів

```
public Dictionary<int, string> dictOverallItem1 = new()
{    { 0, "Задовільний" },
    { 1, "Відносно задовільний" },
    { 2, "Середньої важкості" },
    { 3, "Важкий" }
};
public Dictionary<int, string> dictOverallItem2 = new()
{    { 0, "Нормостенік" },
    { 1, "Астенік" },
    { 2, "Гиперстенік" }
};
```

Dictionaries, властивість класу OverallDictionaries, агрегує всі локальні словники у вигляді загального словника, що дозволяє зручно отримувати доступ до конкретного класифікатора за його символьним ідентифікатором (лістинг 3.5). Це значно спрощує доступ до уніфікованих значень під час заповнення форм, створення документів та обробки візитів у медичній інформаційній системі. Реалізація класу забезпечує гнучку та масштабовану модель представлення статичних довідкових даних.

Лістинг 3.5 – Клас OverallDictionaries та його конструктор

```
internal class OverallDictionaries
 {  public OverallDictionaries()
     {    Dictionaries = new Dictionary<string, Dictionary<int,
string>>()
         {   { "dictOverallItem1",  dictOverallItem1 },
             { "dictOverallItem2",  dictOverallItem2 },
             { "dictOverallItem3",  dictOverallItem3 },
             { "dictOverallItem4",  dictOverallItem4 },
             { "dictOverallItem5",  dictOverallItem5 },
             { "dictOverallItem6",  dictOverallItem6 },
             { "dictOverallItem8",  dictOverallItem8 },
             { "dictOverallItem10", dictOverallItem10 },
             { "dictOverallItem13", dictOverallItem13 },
             { "dictOverallItem14", dictOverallItem14 }
         };}
     public Dictionary<string, Dictionary<int, string>>
Dictionaries { get; set; } … }
```

3.4 Реалізація зміна дати госпіталізації

Оскільки пацієнт може бути госпіталізований декілька разів то візитів до лікарню теж може бути декілька. Метод btnChangeVisitTo (лістинг 3.6) реалізує механізм інтерактивного вибору візиту пацієнта через окреме діалогове вікно.

Лістинг 3.6 – Зміна дати госпіталізації

```
private void btnChangeVisitTo(object sender, RoutedEventArgs e)
{ if (_patient.Visits == null || _patient.Visits.Count == 0)
    {    MessageBox.Show("Немає доступних візитів для вибору.",
"Інформація", MessageBoxButton.OK, MessageBoxImage.Information);
```

```
        return;  } var visitSelector = new
VisitSelector(_patient.Visits, $"{_patient.LastName}
{_patient.FirstName} {_patient.MiddleName}");
    if (visitSelector.ShowDialog() == true)
    {   var selectedVisit = visitSelector.SelectedVisit;
        if (selectedVisit != null)   {  LoadData(_patient,
selectedVisit); }}}
```

Перед викликом діалогу здійснюється перевірка наявності візитів у пацієнта, і в разі їх відсутності користувачу виводиться відповідне інформаційне повідомлення. Якщо візити наявні, створюється екземпляр вікна VisitSelector (лістинг 3.7), якому передається список візитів разом із повним ім’ям пацієнта для відображення у заголовку. Після завершення викликається метод LoadData, що ініціює завантаження детальної інформації про обраний візит разом із даними пацієнта.

Лістинг 3.7 – Клас VisitSelector (файл DoctorNavigationWindow

```
public partial class VisitSelector : Window
{    public Visit SelectedVisit { get; private set; }
    public VisitSelector(List<Visit> visits, string
patientFullName = null)
    {   InitializeComponent();
        lbVisitDates.ItemsSource = visits.OrderByDescending(v =>
v.StartDate).ToList();
        lbVisitsLabel.Text = $"Перелік візитів пацієнта
{patientFullName ?? ""}"; }
    private void btnSelectVisit_Click(object sender,
RoutedEventArgs e) {  if (lbVisitDates.SelectedItem is Visit
selected) { SelectedVisit = selected; DialogResult = true;
Close();}
        else {…}}}
```

У конструкторі класу VisitSelector відбувається ініціалізація компонентів інтерфейсу та заповнення списку lbVisitDates переданими об'єктами типу Visit, упорядкованими за датою візиту. Метод btnSelectVisit_Click перевіряє, чи обрано візит у списку, і в разі успіху, призначає його властивості SelectedVisit і закриває вікно. Якщо жодного елементу не вибрано, користувач отримує повідомлення з проханням зробити вибір.

3.5 Автоматизація заповнення медичних документів

Метод BtnCreateF2Document призначений для автоматизованого формування медичного документа на основі шаблону, з використанням даних пацієнта та обраної дати візиту (лістинг 3.8). В межах методу створюється словник fieldValues, який містить пари ключ-значення, де ключі відповідають маркерам у шаблоні Word-документа, а значення – даним пацієнта, візиту та симптомів, отриманих за допомогою методу GetSymptom. Усі дати форматуються відповідно до українського формату представлення. Документ генерується за допомогою методу GenerateDocument. Згенерований документ зберігається під динамічно сформованою назвою, яка включає ПІБ пацієнта та дату візиту. Таким чином, метод реалізує інтеграцію медичних даних у систему документообігу, що є необхідним для автоматизованого створення уніфікованих медичних документів.

Лістинг 3.8 – Заповнення даними шаблону медичного документу

```
private void BtnCreateF2Document(object sender, RoutedEventArgs
e){  var fieldValues = new Dictionary<string, string>
    {{ "_docCardNumber", _patient.CardNumber },
        { "_docHospitalStartDate",
_visit.StartDate.ToString("dd.MM.yyyy") ?? "" },
        { "_docLastFirstMiddleName", $"{_patient.LastName}
{_patient.FirstName} {_patient.MiddleName}" },
        { "_docBirthDay",
_patient.BirthDate.ToString("dd.MM.yyyy") },
        { "_docAge", _patient.Age.ToString() },
        { "_docLivingAddress", _patient.Address },
        { "_docProfession", _patient.Profession },
        { "_docHospitalEndDate",
_visit.EndDate.ToString("dd.MM.yyyy") ?? "" },
        { "_docDiagnosisMain",
GetSymptom("_fieldFinalDiagnosis") },
        { "_docMKX", GetSymptom("_fieldMKX") }, …
        { "_docDoctor", GetSymptom("_fieldDoctor") },
        { "_docCreationDate",
DateTime.Now.ToString("dd.MM.yyyy") }};
    GenerateDocument("Виписка 066_template.docx", fieldValues,
$"Виписка_066_{_patient.LastName}_{_patient.FirstName}_{_patient
.MiddleName}");
}
```

Метод GenerateDocument (лістинг 3.9) відповідає за створення текстового документа на основі заздалегідь підготовленого шаблону у форматі Microsoft Word (.docx). У процесі виконання метод ініціалізує діалогове вікно збереження файлу, де користувач може вказати бажане ім’я та місце розміщення документа. Якщо користувач підтверджує дію, викликається метод Merge зовнішнього шаблонізатора Engine, який виконує підстановку значень із словника значень (англ. values) у відповідні позиції шаблону. У разі успішного завершення операції користувачеві виводиться інформаційне повідомлення, після чого створений документ автоматично відкривається. Метод також реалізує обробку виключень із наданням користувачу специфічного повідомлення залежно від типу помилки, зокрема при відкритому файлі або відсутності шаблону.

Лістинг 3.9 – Генерація документу

```
private void GenerateDocument(string templateFileName,
Dictionary<string, string> values, string outputName)
{
  var engine = new Engine();
    try{
         var dialog = new SaveFileDialog
        { FileName = outputName,
            DefaultExt = ".docx",
            Filter = "Word Documents (.docx)|*.docx" };
        if (dialog.ShowDialog() == true)
        { engine.Merge($"..\\..\\..\\{templateFileName}",
values, dialog.FileName);
            MessageBox.Show("Документ було створено!",
"Документ", MessageBoxButton.OK, MessageBoxImage.Information);
            new Process            { StartInfo = new
ProcessStartInfo(dialog.FileName) { UseShellExecute = true }
}.Start(); }}
    catch (Exception ex)
    {
      var msg = ex.Message.Contains("being used") ? "Документ
вже створений та відкритий! Перевірте будь-ласка запущені
процеси." : ex.Message.Contains("Could not find file")
                ? "Шаблон документу не знайдено. Будь-ласка
зверніться до підтримки!"  : "Неочікувана помилка. Будь-ласка
зверніться до підтримки!";
        MessageBox.Show(msg, "Помилка", MessageBoxButton.OK,
MessageBoxImage.Error); }  Close();
}
```

3.6 Завантаження даних та використання фільтрів

Метод LoadDoctors (лістинг 3.10) виконує асинхронне завантаження інформації про всіх лікарів із бази даних MongoDB шляхом виклику методу GetAllDoctorsAsync. Попередньо завантажуються конфігураційні параметри з локального файлу, зокрема рядок з'єднання та назва бази даних, що забезпечує динамічне оновлення даних всередині вікна. Після отримання списку лікарів кожному з них присвоюється інформація про найближчу робочу зміну (OnDutyStatus), яка визначається за допомогою окремої логіки, реалізованої у методі GetOnDutyStatus. Це дозволяє додатково класифікувати лікарів для подальшої візуалізації та фільтрації. Останнім викликається метод ApplyFilters, який застосовує поточні критерії відбору та оновлює відображення.

Лістинг 3.10 – Асинхронне завантаження інформації із бази даних

```
private async void LoadDoctors()
{
   var config = AppConfig.Load();
    var mongoService = new
MongoDbService(config.MongoDbConnection, config.DatabaseName);
    allDoctors = await mongoService.GetAllDoctorsAsync();
    foreach (var doctor in allDoctors) doctor.OnDutyStatus =
GetOnDutyStatus(doctor);
 ApplyFilters();
}
```

Метод ApplyFilters (лістинг 3.11) відповідає за динамічне фільтрування списку лікарів відповідно до вказаних критеріїв користувача. У першому етапі здійснюється перевірка наявності текстового запиту в полі tbSearch, після чого здійснюється пошук за ПІБ у різних варіантах порядку, що підвищує гнучкість пошуку. Пошукова операція виконується у нечутливому до регістру режимі. Додатково, якщо активовано прапорець cbOnDutyOnly, застосовується фільтр, що відображає лише тих лікарів, які перебувають на робочій зміні, на основі логіки методу IsDoctorOnDuty. Результат фільтрації

призначається як джерело даних для таблиці DoctorsGrid, що відповідає за відображення списку лікарів у вікні. Такий підхід дозволяє користувачу швидко отримати лише необхідні дані за допомогою фільтрів.

Лістинг 3.11 – Динамічне фільтрування списку лікарів

```
private void ApplyFilters()
{    var filtered = allDoctors;

    if (!string.IsNullOrWhiteSpace(tbSearch.Text))
    {
  var search = tbSearch.Text.Trim().ToLower();
        filtered = filtered.Where(d =>
            $"{d.LastName} {d.FirstName}
{d.MiddleName}".ToLower().Contains(search) ||
            $"{d.FirstName} {d.MiddleName}
{d.LastName}".ToLower().Contains(search) ||
            $"{d.LastName}
{d.FirstName}".ToLower().Contains(search)
        ).ToList();
}
    if (cbOnDutyOnly.IsChecked == true)
        filtered = filtered.Where(d =>
IsDoctorOnDuty(d)).ToList();
    DoctorsGrid.ItemsSource = filtered;
}
```

3.7 Експорт та імпорт даних

Метод ExportData (лістинг 3.12) реалізує механізм експорту даних пацієнтів у форматі ZIP-архіву. Залежно від ролі поточного користувача (лікар або головний лікар), відбувається асинхронне отримання відповідної підмножини пацієнтів з бази MongoDB. Дані серіалізуються у JSON у компактному представленні, що включає лише основні атрибути пацієнта. Користувачеві надається можливість обрати місце збереження архіву за допомогою стандартного діалогового вікна SaveFileDialog. Архів формується за допомогою класу ZipArchive, який дозволяє зберегти згенерований JSON-файл у стиснутому вигляді. У разі успішного завершення експорту користувач отримує відповідне інформаційне повідомлення про результат операції, інакше повідомлення про помилку.

Лістинг 3.12 – Реалізація експорту даних пацієнтів

```
using (var _mongoService = new
MongoDbService(AppConfig.Load().MongoDbConnection,
AppConfig.Load().DatabaseName)) {
     var json = JsonConvert.SerializeObject(minimalPatients,
Formatting.None);
    var saveDialog = new SaveFileDialog
    {   Title = "Зберегти архів пацієнтів",
        Filter = "ZIP files (*.zip)|*.zip",
        FileName = "exported_patients.zip" };
    if (saveDialog.ShowDialog() == true)
    {  using (var zipStream = new
FileStream(saveDialog.FileName, FileMode.Create))
        using (var archive = new ZipArchive(zipStream,
ZipArchiveMode.Create))
        {   var entry = archive.CreateEntry("patients.json",
CompressionLevel.Optimal);
            using var entryStream = entry.Open();
            using var writer = new StreamWriter(entryStream);
            writer.Write(json); }
        MessageBox.Show("Пацієнтів експортовано у архів
успішно.", "Успіх", MessageBoxButton.OK,
            MessageBoxImage.Information);}}
```

Метод ImportData забезпечує імпорт даних пацієнтів із зовнішнього ZIP-файлу, що містить JSON-файл у визначеній структурі. Перед обробкою виконується перевірка правильності архіву, зокрема наявність лише одного JSON-файлу всередині. Зміст файлу десеріалізується у список об’єктів типу Patient, після чого проводиться аналіз на наявність співпадінь за унікальними ідентифікаторами пацієнтів. Отримані дані розподіляються на дві категорії: нові записи, що потребують вставлення, та наявні – для оновлення. Користувачеві виводиться діалогове повідомлення з підсумком змін, що дозволяє підтвердити або скасувати імпорт. Далі погоджені зміни застосовуються до бази даних, а у випадку помилок виконується обробка винятків із відповідним інформуванням.

4 ІНСТРУКЦІЯ КОРИСТУВАЧА

4.1 Розгортання програмного забезпечення

Описуваний програмний засіб являє собою desktop-застосунок, розроблений з використанням мов програмування C# у рамках програмної платформи .NET та з використанням WPF як базової бібліотеки для побудови графічного інтерфейсу. Для зберігання та обробки даних використовується база даних у нереляційній СУБД MongoDB. Для коректного функціонування програмного забезпечення рекомендовано використовувати персональний комп’ютер із встановленою операційною системою Windows 10 або новішою, процесором з архітектурою x64, оперативною пам’яттю обсягом не менше 4 ГБ та наявністю вільного простору на диску щонайменше 300 МБ.

У поточній версії застосунку (v1.1) необхідною передумовою є наявність доступу до локальної мережі, оскільки програмний продукт припускає взаємодії з віддаленим сховищем даних у межах локальної мережі відділу лікарні. Усі допоміжні бібліотеки, які необхідні для функціонування, включені до пакету інсталяції. Разом з тим, деякі системні компоненти .NET, які входять до стандартного складу операційної системи Windows, повинні бути встановлені на ПК користувача. У разі їх відсутності система інсталяції або сама операційна система виведе відповідне повідомлення з пропозицією встановити відсутні компоненти.

Інсталяція програмного забезпечення здійснюється за допомогою окремого інсталяційного модуля, який забезпечує покрокове налаштування параметрів розгортання. Після запуску інсталятора користувачеві буде запропоновано вказати кінцеву директорію, у яку здійснюватиметься копіювання всіх файлів застосунку (рисунок 4.1). Також у процесі інсталяції можна підтвердити або скасувати створення ярлика для запуску програми на робочому столі або в меню «Пуск».

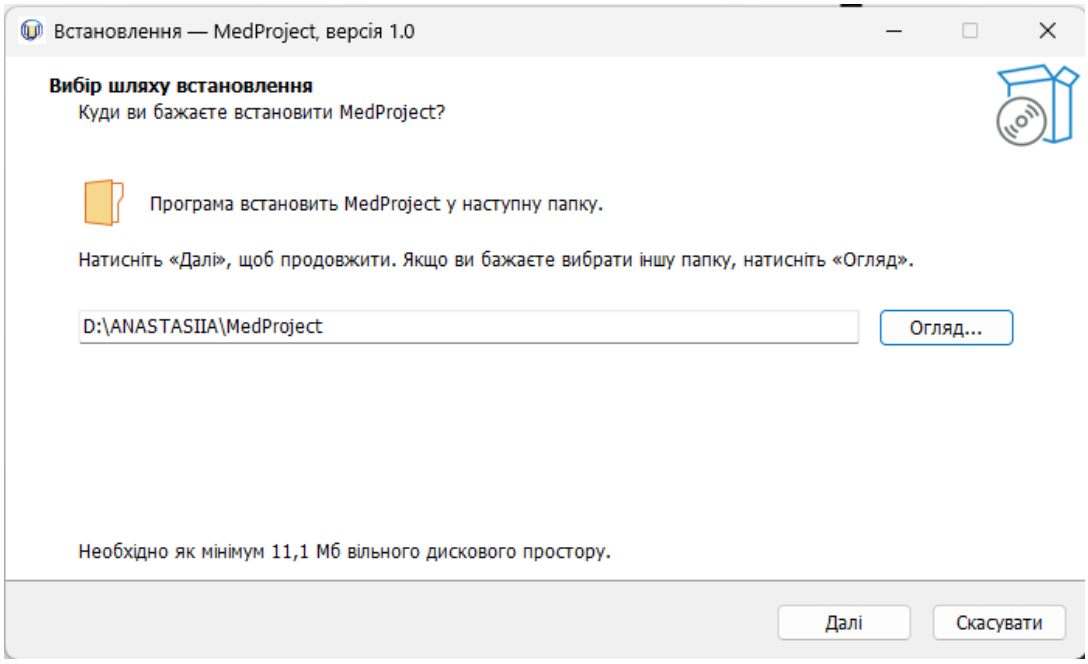

Рисунок 4.1 – Інсталяція програмного забезпечення

Назва застосунку відображається в інтерфейсі інсталятора, а також буде використовуватись у розділі «Програми та компоненти» операційної системи для подальшого керування. Після завершення встановлення з’явиться повідомлення з опцією автоматичного запуску програмного забезпечення. Рекомендується не активувати цю опцію та виконати перший запуск вручну після завершення усіх додаткових налаштувань.

Після інсталяції користувачу необхідно створити екземпляр бази даних у хмарному середовищі MongoDB Atlas. Для цього необхідно перейти на офіційний веб-сайт MongoDB та виконати реєстрацію нового користувача. Після реєстрації слід створити новий кластер у рамках безкоштовного тарифного плану (позначений як «Shared Cluster»), в якому буде зберігатися вся інформація. У процесі створення необхідно задати регіон розміщення, назву кластера та створити користувача з паролем, який буде використовуватись для автентифікації під час підключення до бази даних. Вибір регіону розміщення та назви кластеру залишається повністю за користувачем і не має впливу на подальший стан застосунку.

Після створення кластера у панелі керування буде доступна кнопка «Connect», за допомогою якої можна згенерувати стандартну строку

підключення у форматі MongoDB URI. Отриману строку необхідно скопіювати та вставити у файл config.json, який розміщується у директорії, де було встановлено програму. У цьому файлі необхідно змінити значення параметра MongoDbConnection, замінивши на отриману строку підключення. Усі інші параметри конфігураційного файлу змінювати не потрібно. Збереження змін у конфігураційному файлі дозволяє застосунку коректно ініціалізувати з'єднання з базою даних при першому запуску.

При локальному використання бази даних у встановленій MongoDB на ПК користувача чи у локальній мережі відділу лікарні, необхідно скорегувати рядок підключення.

4.2 Вхід у систему та реєстрація

Для запуску застосунку необхідно відкрити файл MedProject_UI.exe. Після запуску відкриється вікно для входу у систему з інтерфейсом, представленим на рисунку 4.2.

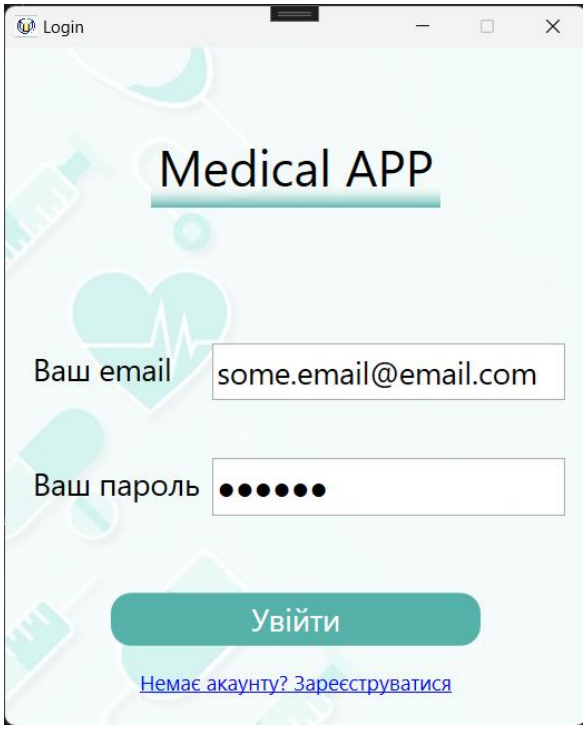

Рисунок 4.2 – Вікно входу у систему

Вікно входу у систему містить такі елементи:

- текстове поле для введення email;
- текстове поле для введення паролю;
- кнопка для входу в обліковий запис у системі;
- кнопка для реєстрації нових користувачів.

Незареєстровані користувачі можуть натиснути кнопку для реєстрації та побачити вікно для створення нового облікового запису (рисунок 4.3).

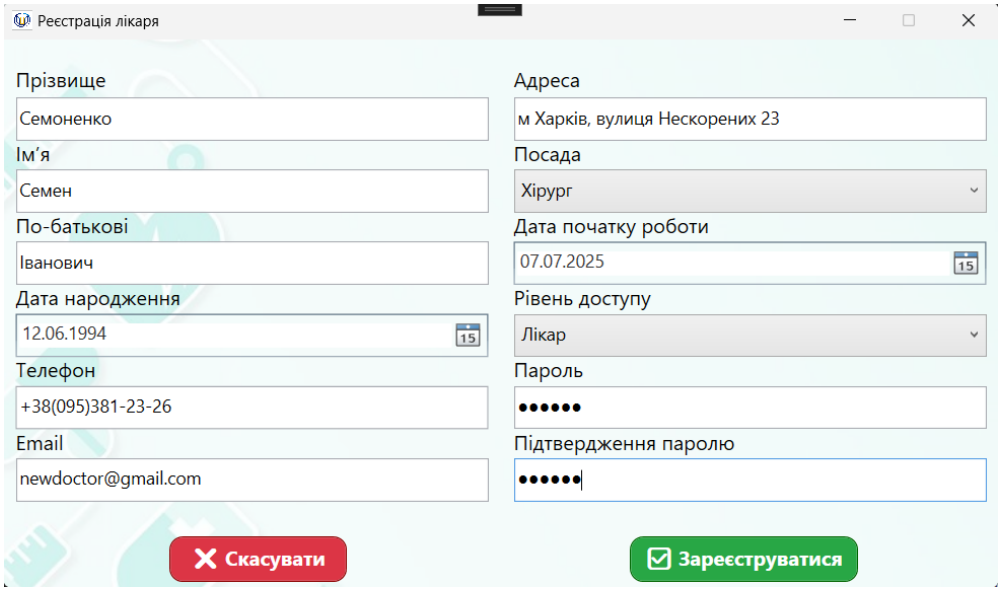

Рисунок 4.3 – Вікно для реєстрації нових лікарів

Вікно для реєстрації нових лікарів містить такі елементи:

- текстове поле для введення прізвища;
- текстове поле для введення ім'я;
- текстове поле для введення по-батькові;
- поле для вибору дати народження;
- текстове поле для введення email;
- текстове поле для введення адреси;
- випадаючий список для вибору посади;
- поле для вибору дати початку роботи;
- випадаючий список для вибору рівня доступу («Лікар» або «Відвідувач»);

- текстове поле для введення паролю;
- текстове поле для підтвердження паролю;
- кнопка для скасування реєстрації;
- кнопка для реєстрації нового облікового запису.

Після реєстрації буде виведено повідомлення про успішну реєстрацію у разі правильності заповнення усіх полів вікна. Далі відкриється вікно входу, зображеного на рисунку 4.2.

У вікні входу необхідно ввести свій email та пароль і натиснути на кнопку «Увійти». Відкриється головне вікно застосунку (рисунок 4.4).

Рисунок 4.4 – Головне вікно застосунку (для головного лікаря)

Головне вікно застосунку містить такі елементи:

- кнопка «Переглянути список докторів»;
- кнопка «Переглянути список пацієнтів»;

- кнопка «Редагувати власні дані»;
- кнопка «Експортувати пацієнтів» (присутня у користувача з рівнем доступу «Головний лікар» або «Лікар»);
- кнопка «Імпортувати пацієнтів» (присутня у користувача з рівнем доступу «Головний лікар»);
- кнопка «Редагувати графік» (присутня у користувача з рівнем доступу «Головний лікар» або «Адміністратор»);
- кнопка «Переглянути графік» (присутня у користувача з рівнем доступу «Лікар»);
- кнопка «Вихід».

4.3 Перегляд, редагування та видалення лікарів

Для перегляду списку лікарів необхідно на головній сторінці застосунку натиснути на кнопку «Переглянути список докторів». Відкриється вікно для перегляду переліку лікарів (рисунок 4.5).

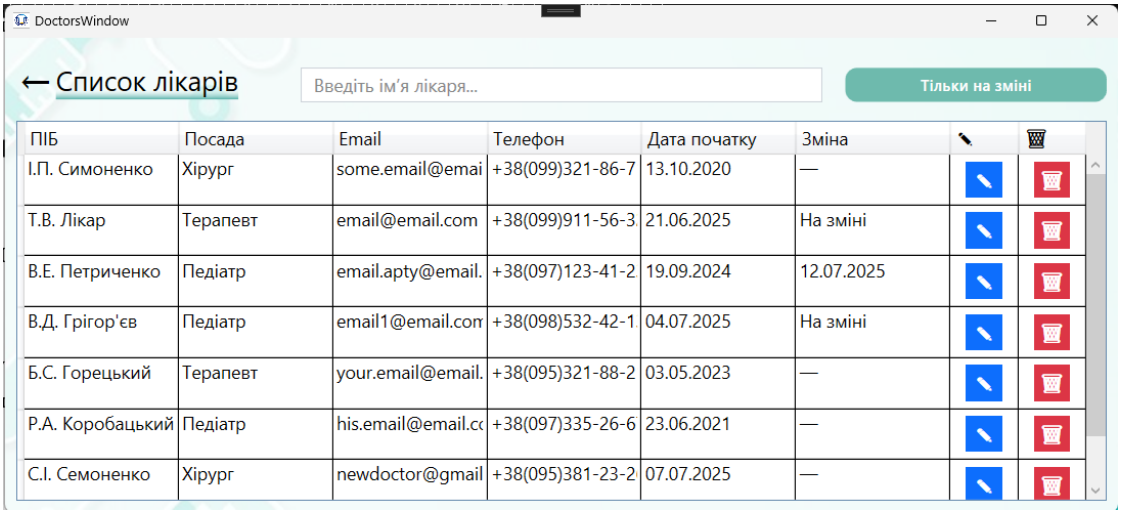

Рисунок 4.5 – Вікно для перегляду переліку лікарів

Вікно для перегляду списку лікарів містить такі елементи:

- таблиця переліку лікарів та їх особова інформація;
- пошуковий рядок;

- кнопка «Тільки на зміні»;
- кнопка «Редагувати» (присутня для користувачів, що мають рівень доступу «Головний лікар» або «Адміністратор»);
- кнопка «Видалити» (присутня для користувачів, що мають рівень доступу «Головний лікар» або «Адміністратор»).

Для пошуку необхідного лікаря можна скористатись пошуковим рядком, ввівши прізвище, ім'я або по-батькові. Кнопка «Тільки на зміні» відображає лікарів які, в день перегляду сторінки, присутні на робочому місці. Натиснувши на кнопку «Редагувати» відкриється вікно для редагування особової інформації лікаря (рисунку 4.6).

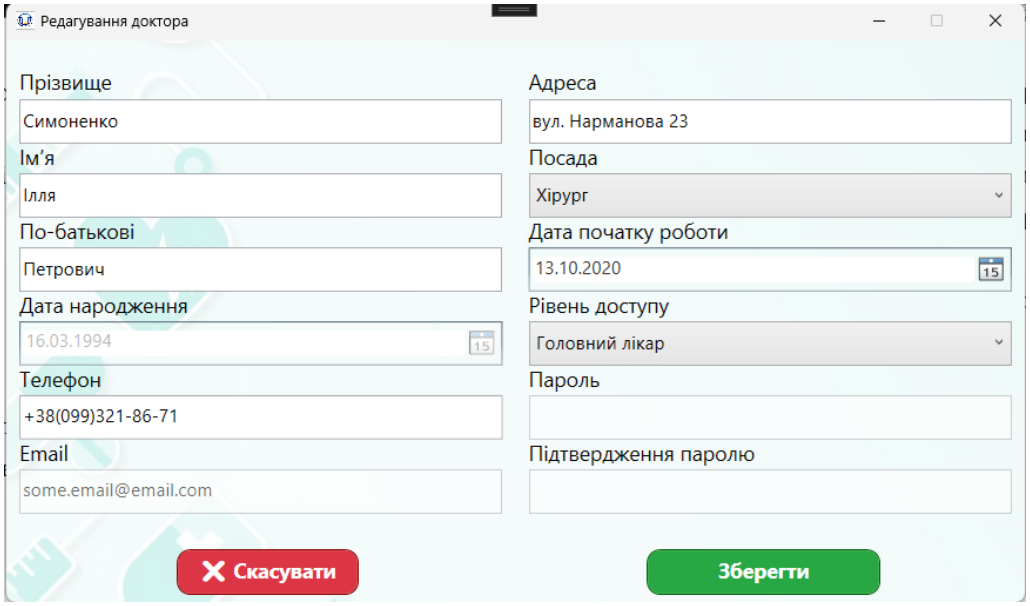

Рисунок 4.6 – Вікно для редагування даних лікаря

Вікно для редагування даних лікаря має такі ж елементи, як і вікно для створення нового облікового запису користувача застосунку. Єдиними відмінностями є блокування зміни дати народження, email, посади та рівня доступу. Дати народження може редагувати тільки користувачі з правом доступу «Адміністратор». Email можна редагувати тільки самому собі. Посаду та рівень доступу може редагувати тільки користувач, що має рівень доступу «Головний лікар» та «Адміністратор».

Після натискання на кнопку «Видалити» у вікні для перегляду переліку лікарів відкриється вікно для передачі пацієнтів іншому лікарю (рисунок 4.7). Дане вікно відкривається тільки у разі наявності у лікаря пацієнтів.

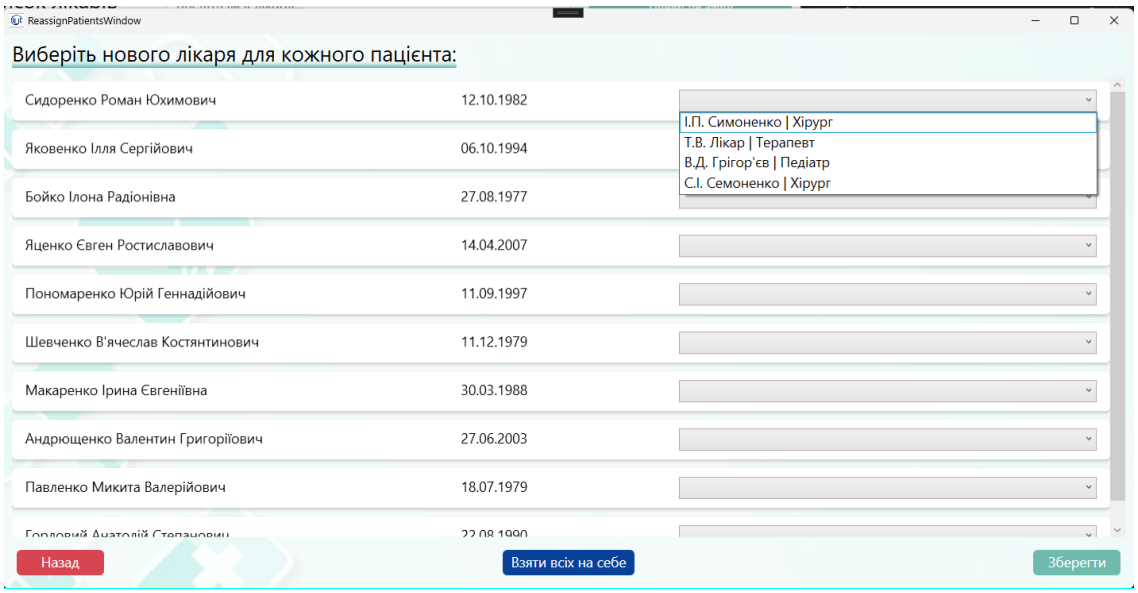

Рисунок 4.7 – Вікно для передачі пацієнтів іншому лікарю

Вікно для передачі пацієнтів іншому лікарю містить такі елементи:

- перелік пацієнтів;
- випадаючий список з переліком можливих лікарів;
- кнопка «Назад»;
- кнопка «Взяти всіх на себе»;
- кнопка «Зберегти».

Видаляти лікарів може тільки користувач, з рівнем доступу «Головний лікар». При натисканні на кнопку «Взяти всіх на себе» всі випадаючі поля автоматично заповнюються ім’ям головного лікаря.

4.4 Перегляд, додавання та редагування даних пацієнтів

У головному вікні застосунку користувач може натиснути на кнопку «Переглянути список пацієнтів». Після натискання відкриється вікно з переліком пацієнтів лікаря (рисунок 4.8).

Рисунок 4.8 – Вікно перегляду пацієнтів

Вікно перегляду пацієнтів має такі елементи:

- список пацієнтів у таблиці;
- SearchBar для пошуку за ПІБ;
- кнопка «Деталі»;
- кнопка «Видалити»;
- кнопка «Додати пацієнта»;

Більшу частину вікна займає таблиця, в якій вказані необхідні поля для швидкої ідентифікації пацієнта. Також в таблиці для кожного пацієнта є два додаткових поля: «Інформація» та «Видалення». Поле «видалення» містить кнопку, за допомогою якої лікар може видалити пацієнта. Поле «Інформація» містить кнопку, при натисканні на яку відкривається вікно «Інформація пацієнта». Для пошуку необхідного пацієнта користувач має ввести прізвище, ім’я або по-батькові у текстове поле.

4.4.1 Процес додавання нового пацієнту

Праворуч від текстового поля на головному вікні застосунка знаходиться кнопка «Додати пацієнта», яка, при натисканні на неї, відкриває вікно для додавання нового пацієнта. Вікно для додавання та редагування

інформації пацієнта складається з 8 розділів. Перше вікно для додавання нового пацієнта містить поля для заповнення основних даних пацієнта, як зображено на рисунку 4.9.

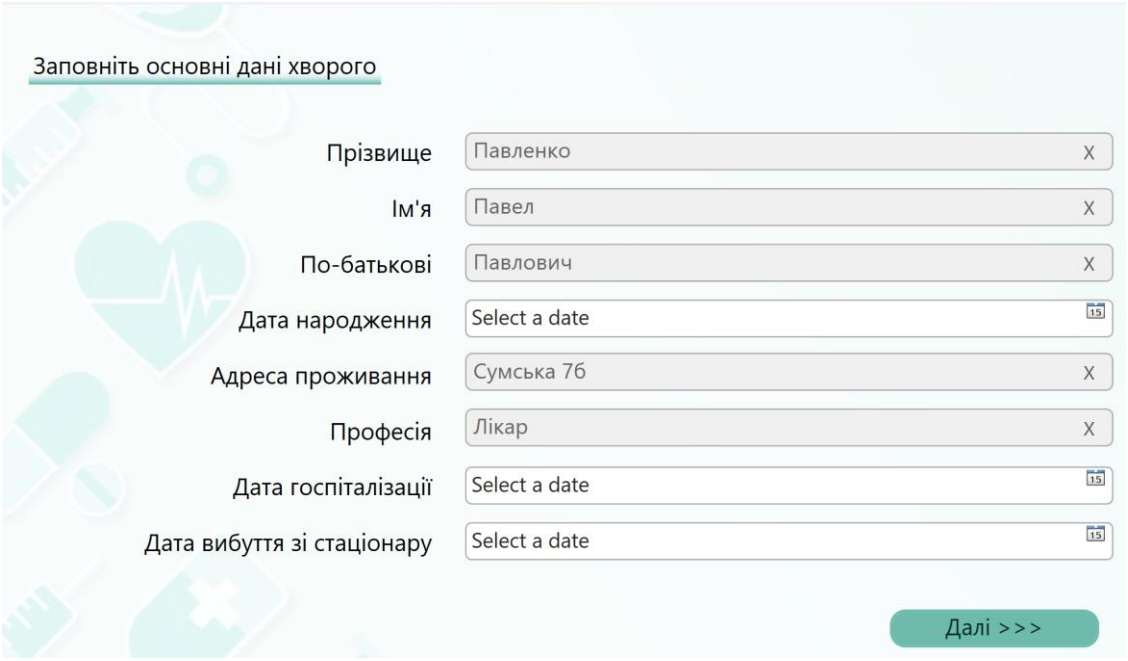

Рисунок 4.9 – Розділ для заповнення основних даних пацієнта

У вікні для заповнення основних даних пацієнта передбачені такі поля:

- прізвище (обов’язкове поле);
- ім’я (обов’язкове поле);
- по-батькові (обов’язкове поле);
- дата народження (обов’язкове поле);
- адреса проживання (обов’язкове поле);
- професія (обов’язкове поле);
- дата госпіталізації (обов’язкове поле);
- дата вибуття зі стаціонару (необов’язкове поле, заповнюється у майбутньому).

Обов’язкові поля мають бути заповненими до переходу до наступних розділів, а пусті поля в такому випадку будуть виділені червоним.

У нижній частині сторінки «Основні дані пацієнта» розміщено кнопку «Далі», при натисну на яку, за умови заповненості усіх обов’язкових полів, у

вікні відкриється наступний розділ для заповнення даних щодо діагнозів (рисунок 4.10).

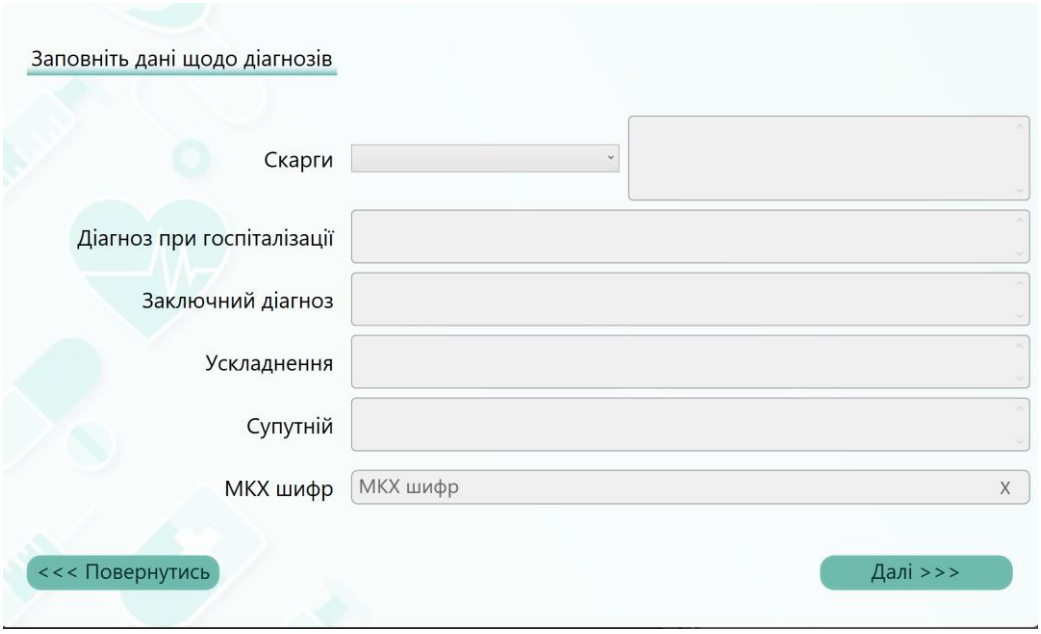

Рисунок 4.10 – Розділ для заповнення даних щодо діагнозів

Розділ даних щодо діагнозів містить такі поля для заповнення:

- скарги (обов’язкове поле, з запропонованими типовими описами скарг: кашель з слизовою, гнійною мокротою, сухий; гіпертермія (до певного значення у градусах Цельсія); набряк обличчя; загальна слабкість тощо);
- діагноз при госпіталізації (обов’язкове поле);
- заключний діагноз (необов’язкове поле);
- ускладнення (необов’язкове поле);
- супутній (необов’язкове поле);
- МКХ шифр (необов’язкове поле).

У нижній частині сторінки «Дані щодо діагнозів» зліва розміщено кнопку «Повернутись» за допомогою якої є можливість перейти на попередній розділ для заповнення основних даних пацієнта (рисунок 4.9). Праворуч, у нижній частині, розміщено кнопку «Далі». За умови, що усі обов’язкові поля заповнені, у вікні з’явиться наступний розділ для заповнення додаткових даних (рисунок 4.11).

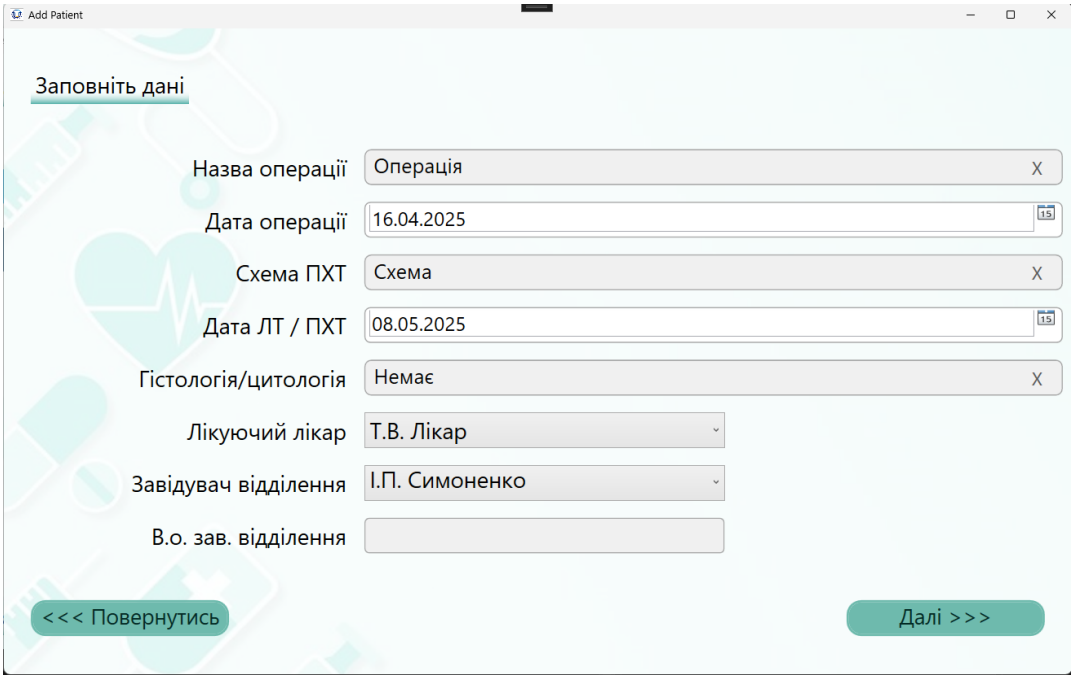

Рисунок 4.11 – Розділ для заповнення додаткових даних

У розділі додаткових даних зображено такі поля для заповнення, як:

- назва операції (необов’язкове поле);
- дата операції (необов’язкове поле);
- схема ПХТ (необов’язкове поле);
- дата ПХТ (необов’язкове поле);
- гістологія/цитологія (необов’язкове поле);
- лікуючий лікар (обов’язкове поле, вибір з переліку);
- завідувач відділення (обов’язкове поле, вибір з переліку);
- в.о. зав. відділення (необов’язкове поле).

Після заповнення розділу додаткових даних відкриється вікно для заповнення першої частини розділу об’єктивного статусу (рисунок 4.12). У першій частині розділу об’єктивного статусу зображено такі поля з варіантами вибору, як:

- загальний стан (обов’язкове поле);
- конституція (обов’язкове поле);
- харчування (обов’язкове поле);
- задишка (обов’язкове поле);

- стан шкіри (обов’язкове поле);
- язик (обов’язкове поле);
- печінка (обов’язкове поле);
- стілець (обов’язкове поле).

Рисунок 4.12 – Перший частина розділу об’єктивного статусу

Наступним розділом для заповнення є друга частина даних про об’єктивний статус (рисунок 4.13).

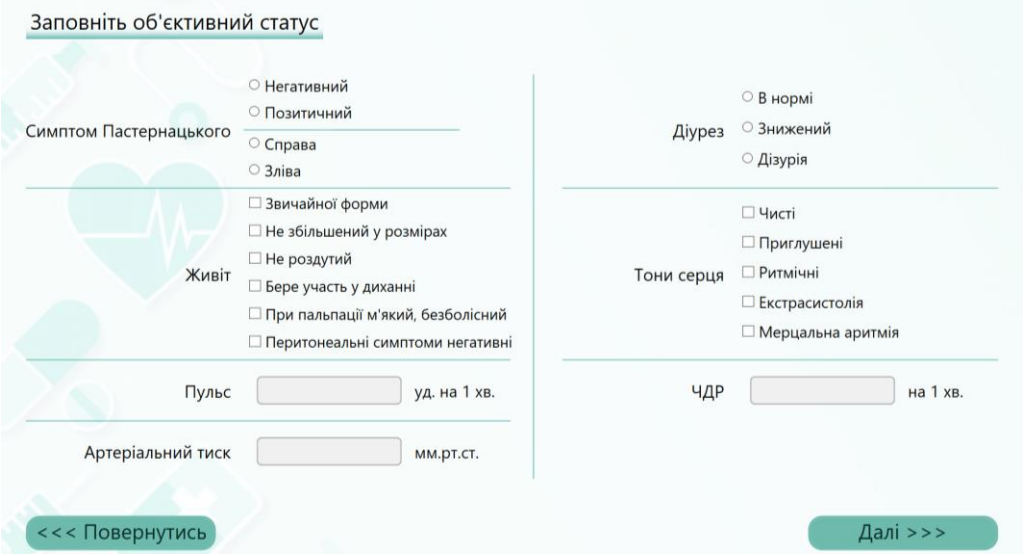

Рисунок 4.13 – Другий частина розділу об’єктивного статусу

Друга частина розділу об’єктивного статусу містить такі поля для заповнення, як:

- симптом Пастернацького (необов’язкове поле);
- живіт (обов’язкове поле з можливими варіантами);
- пульс (обов’язкове поле);
- артеріальній тиск (обов’язкове поле);
- діурез (обов’язкове поле);
- тони серця (обов’язкове поле);
- ЧДР (обов’язкове поле).

Після натискання на кнопку «Далі», за умови заповненості усіх обов’язкових полів, у вікні з’явиться наступний розділ щодо інформації про анамнез (рисунок 4.14).

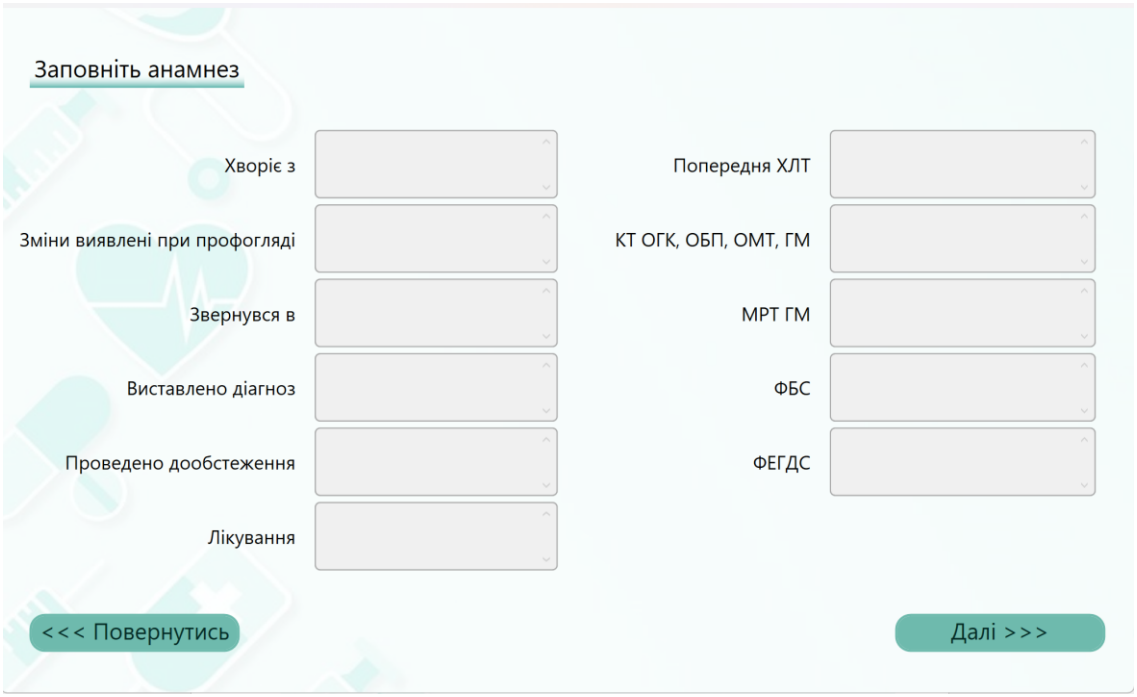

Рисунок 4.14 – Розділ для введення інформації про анамнез

У розділі для введення інформації про анамнез зображено такі поля для заповнення, як:

- хворіє з (необов’язкове поле);
- зміни виявлені при профогляді (необов’язкове поле);

- звернувся в (необов’язкове поле);
- виставлено діагноз (необов’язкове поле);
- проведено дообстеження (необов’язкове поле);
- лікування (необов’язкове поле);
- тони серця (необов’язкове поле);
- попередня ХЛТ (необов’язкове поле);
- КТ ОГК, ОБП, ОМТ, ГМ (необов’язкове поле);
- МРТ ГМ (необов’язкове поле);
- ФБС (необов’язкове поле);
- ФЕГДС (необов’язкове поле).

Після заповнення інформації анамнезу відкривається розділ для введення інформації про анамнез життя (рисунок 4.15).

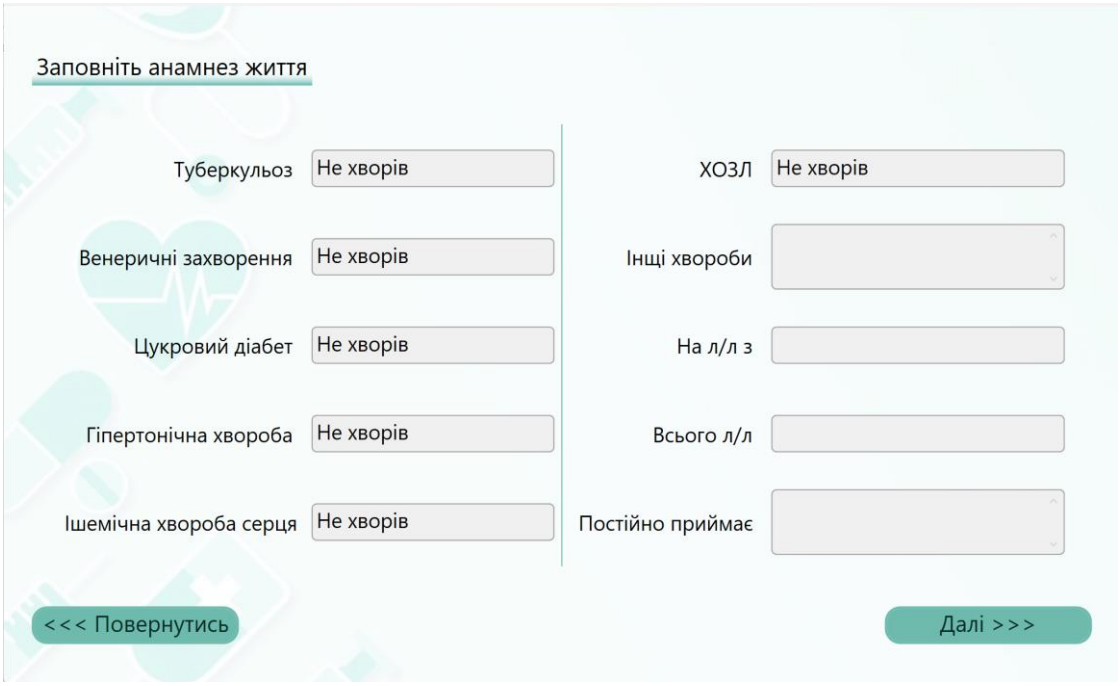

Рисунок 4.15 – Розділ для введення даних про анамнез життя

Розділ для введення даних про анамнез життя містить такі поля, як:

- туберкульоз (обов’язкове поле);
- венеричні захворювання (обов’язкове поле);
- цукровий діабет (обов’язкове поле);

- гіпертонічна хвороба (обов’язкове поле);
- ішемічна хвороба серця (обов’язкове поле);
- ХОЗЛ (обов’язкове поле);
- інші хвороби (необов’язкове поле);
- на л/л з (необов’язкове поле);
- всього л/л (необов’язкове поле);
- постійно приймає (необов’язкове поле).

Останнім розділ для створення нового пацієнта є розділ Locus Morbi, зображений на рисунку 4.16.

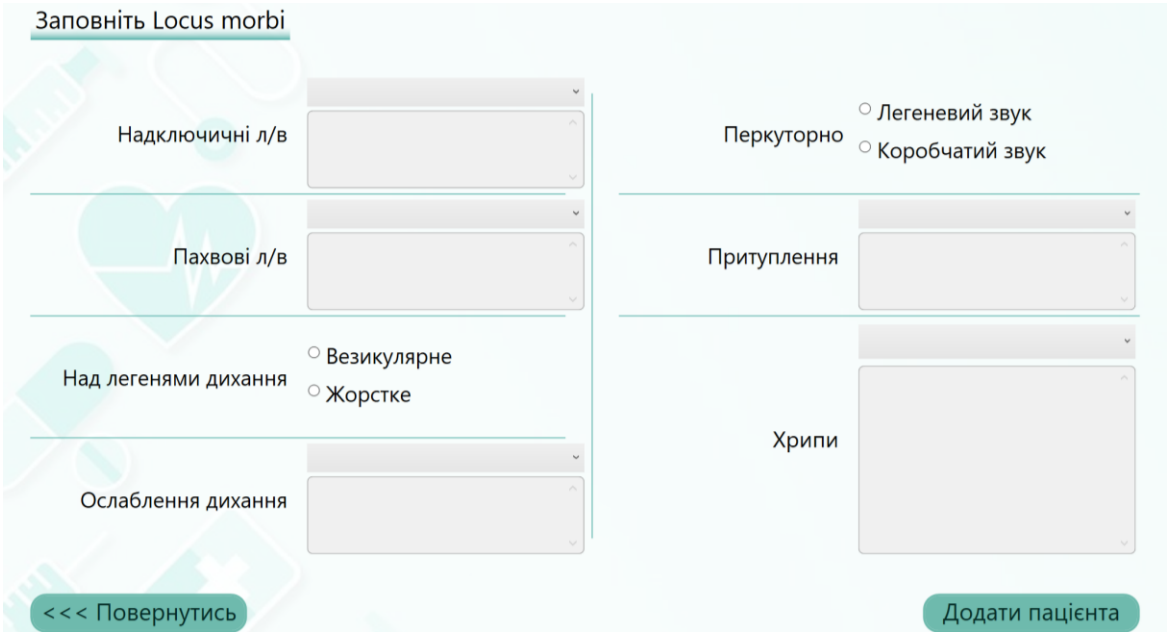

Рисунок 4.16 – Розділ Locus Morbi

У розділі Locus Morbi містяться такі поля для заповнення, як:

- надключичні л/в (необов’язкове поле, з запропонованими типовими описами);
- пахові л/в (необов’язкове поле, з запропонованими типовими описами);
- над легенями дихання (необов’язкове поле);
- ослаблення дихання (необов’язкове поле, з запропонованими типовими описами);

- перкуторно (необов’язкове поле);
- притуплення (необов’язкове поле, з запропонованими типовими описами);
- хрипи (необов’язкове поле, з запропонованими типовими описами).

В нижній частині у розділі Locus Morbi розміщено кнопку «Додати пацієнта». При натисканні на неї, буде створено нового пацієнта, закрито сторінку для додавання та відображено оновлену таблицю головній сторінці застосунку (рисунок 4.8). Новий пацієнт буде відображатись у кінці таблиці.

4.4.2 Перегляд та редагування інформації пацієнта

Після збереження нового пацієнта може виникнути потреба переглянути або відредагувати його дані. Для цього у головному вікні застосунку лікар має натиснути на кнопку «Деталі» у полі таблиці «Інформація». Після натискання на кнопку «Деталі» відкриється вікно детальної інформації про пацієнта (рисунку 4.17). Користуючись вікном детальної інформації про пацієнта, користувач у лівій частині може переглянути основну інформацію, яка ідентифікує пацієнта.

Рисунок 4.17 – Вікно детальної інформації про пацієнта

Нижче, під коротким описом пацієнта, розміщено кнопку для генерації документів. В лівому нижньому куті розміщено кнопку для зміни дати візиту пацієнта.

В лівій частині розміщено зону з переліком полів. В залежності від наявності даних у користувача ці поля будуть заповнені чи незаповнені. Над зоною з переліком полів розміщена кнопки «Додати нову госпіталізацію» та «Редагувати інформацію». Після натискання на кнопку для редагування відкривається вікно, остання сторінка якого зображена на рисунку 4.18.

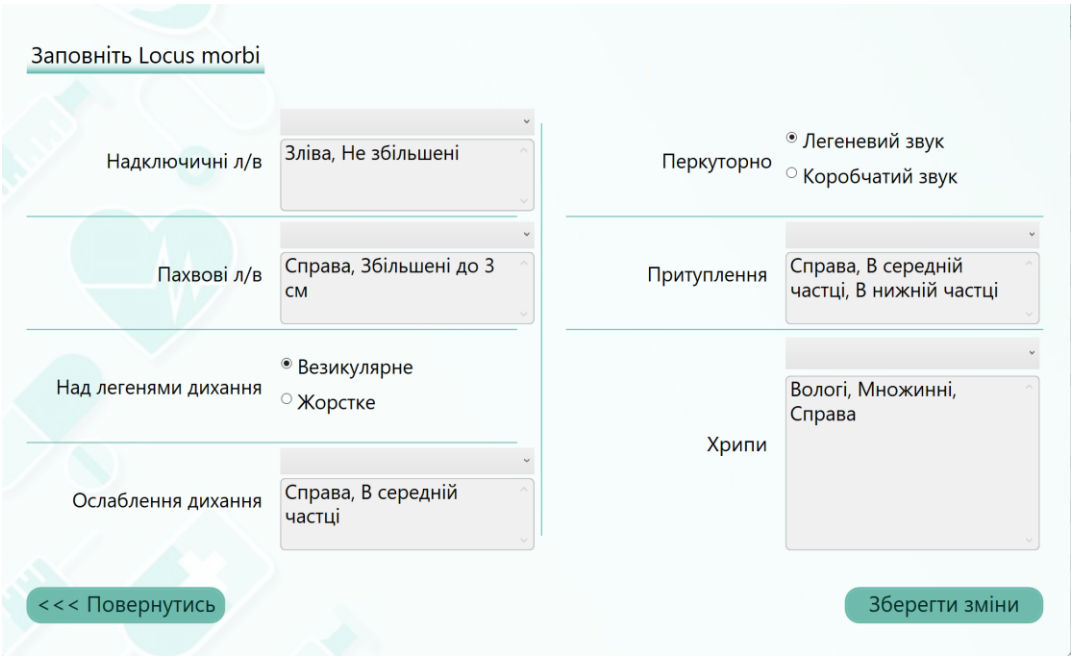

Рисунок 4.18 – Остання сторінка вікна для редагування даних пацієнта

Зовнішній вигляд та реалізація восьми розділів є спільними зі сторінкою для додавання нового пацієнта та сторінкою додавання нової госпіталізації. Відмінності – на сторінці для редагування інформації усі поля вже заповнені даними пацієнта, та у останньому розділі замість кнопки «Додати пацієнта» розміщено кнопку «Зберегти зміни». При додаванні нової госпіталізації основна інформація про пацієнта буде заблокованою для зміни та в кінці розміщена кнопка «Зберегти візит». Після натиску на кнопку «Зберегти зміни», зміни будуть збережені та вікно для редагування закриється.

4.5 Генерація документів

У користувача може виникнути потреба згенерувати документ. Для цього необхідно натиснути на кнопку «Згенерувати документ», яка відкриє вікно для вибору необхідної форми документу (рисунок 4.19).

Документи заповнюються інформацією пацієнта за обраним лікарем візитом. У вікні для вибору необхідної форми розміщено дві кнопки, для генерації відповідного документу.

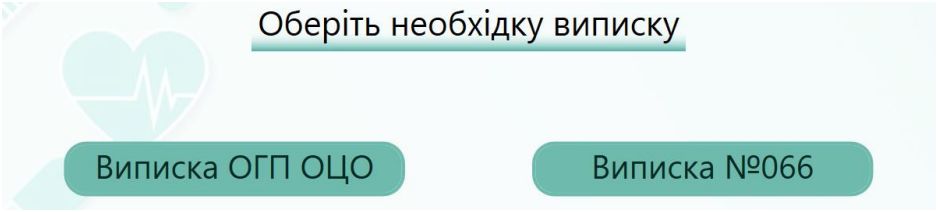

Рисунок 4.19 – Вікно для вибору необхідного форми документу

Перша кнопка під назвою «Виписка ОГП ОЦО» згенерує документ у формі виписки ОГП ОЦО. Друга кнопка «Виписка №066» відповідно згенерує документ у формі виписки №066. Після обрання необхідної медичної форми для заповнення, відкриється вікно для вибору місця зберігання майбутнього документа. Частина заповненого документа форми ОГП ОЦО зображено на рисунку 4.20.

Комунальне некомерційне підприємство
Обласний Центр Онкології
Regional Cancer Center
Municipal non-profit enterprise

ХАРКІВСЬКА ОБЛАСНА РАДА
KHARKIV REGIONAL COUNCIL

ВИПИСКА	IX	456	AK	123

з медичної карти стаціонарного хворого

Виписку направлено в ЛПУ за місцем проживання

1. Прізвище, ім'я, по батькові хворого	Павленко Миколай Сергійович		
2. Дата народження	12.07.1989		
3. Місце проживання хворого	Сумська 9		
3. Місце роботи (спеціальність)	Тестувальник		
5. Дата госпіталізації	11.04.2024	Дата вибуття зі стаціонару	06.06.2024

Рисунок 4.20 – Згенерований документи форми ОГЦ ОЦО

4.6 Експортування та імпортування даних

Лікар має можливість експортувати дані про своїх пацієнтів у обране ним місце на його комп'ютері. Це можна використовувати для створення резервних копій на випадок проблем з базою даних для можливості відновлення інформації або тимчасової відсутності підключення до мережі. Також, оскільки база даних у кожного лікаря своя, то ця функціональна можливість забезпечує подальший імпорт даних пацієнтів до бази «Головного лікаря». Для експортування даних необхідно на головному вікні застосунку натиснути на кнопку «Експортувати пацієнтів». Після цього відкриється вікно для вибору місця зберігання архіву.

Необхідно звернути увагу, що буде згенеровано архів формату .zip з одним файлом всередині. Будь-яка модифікація архіву може призвести до збоїв системи.

Імпортувати дані може лише «Головний лікар», оскільки тільки він потребує дані усіх пацієнтів для подальшого аналізу. Для імпортування даних необхідно на головному вікні застосунку натиснути на кнопку «Імпортувати пацієнтів». Після цього відкриється вікно для вибору необхідного архіву, та далі, після аналізу файлу в архіві, буде виведено повідомлення для підтвердження операції імпорту. Користувача буде повідомлено про кількість записів, що може бути додано та оновлено (рисунок 4.21). Після успішного імпорту даних на екран виведеться повідомлення.

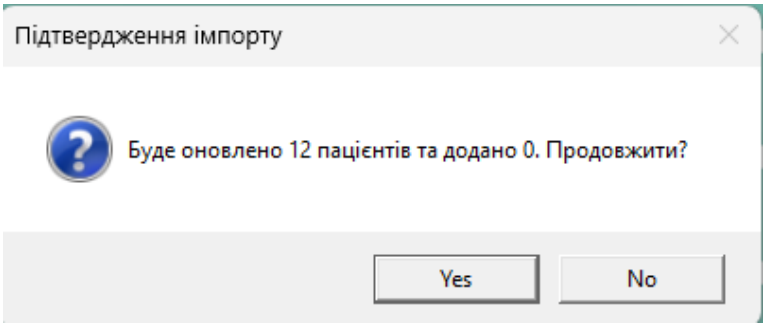

Рисунок 4.21 – Пірдвердження імпорту даних

4.7 Редагування графіку роботи лікарів

У застосунку присутня можливість вести графік роботи лікарів. Редагувати весь графік можуть лише ті особи, які мають рівень доступу «Головний лікар». Особи, з рівнем доступу «Лікар» можуть тільки переглядати та запропонувати свій рафік роботи. Для редагування графіку роботи необхідно на головному вікні застосунку натиснути на кнопку «Редагувати графік» (рисунок 4.22).

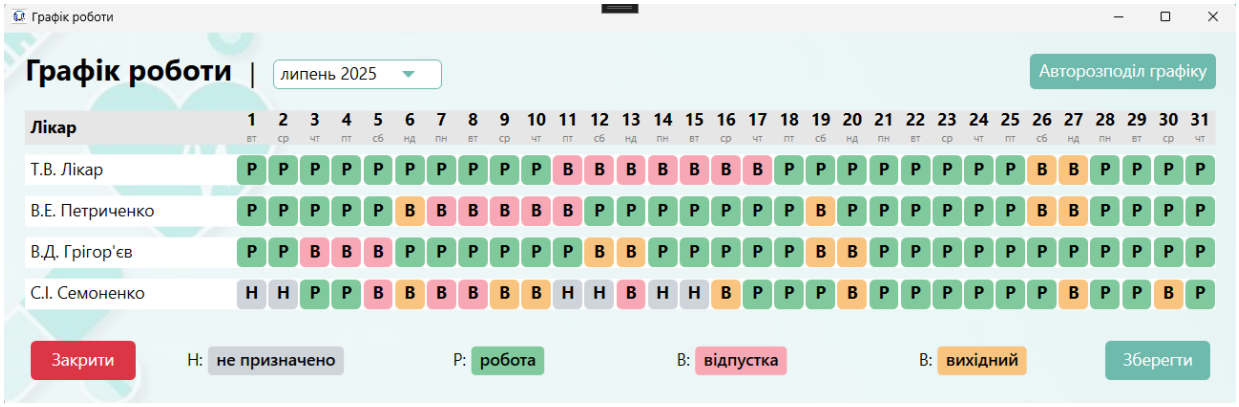

Рисунок 4.22 – Вікно редагування графіку роботи лікарів

Вікно для реєстрації нових лікарів містить такі елементи:

- календар;
- випадаючий список місяців;
- кнопка авторозподілу графіку;
- кнопка «Закрити»;
- кнопка «Зберегти».

У даному вікні головний лікар може змінювати статус робочого дня для кожного лікаря натискаючи на кнопки, з яких побудовано календар.

ВИСНОВКИ

Робота присвячена вирішенню актуальної сучасної проблеми надмірного бюрократичного навантаження на медичних працівників, яка негативно впливає на якість та швидкість надання медичних послуг.

Проведено детальний аналіз предметної області, пов’язаний з автоматизацією заповнення медичних документів. Розгляд існуючих програмних рішень виявив їхні обмеження щодо налаштування з урахуванням специфічних потреб окремих медичних закладів, а також складнощі з інтеграцією в локальні інформаційні системи. Зокрема, поширені системи документообігу виявилися складними у використанні або вимагають великих фінансових витрат на впровадження та підтримку. На основі проведеного аналізу визначено доцільність розробки нового програмного застосунку, який буде орієнтований на автоматизацію процесів заповнення медичної документації з урахуванням специфіки роботи українських медичних установ.

Використано мову програмування C# та технологію WPF (Windows Presentation Foundation) для створення зручного та інтуїтивно зрозумілого інтерфейсу користувача. Для зберігання та управління даними пацієнтів застосовано нереляційну СУБД MongoDB, яка забезпечує гнучкість у роботі з неструктурованими даними та високий рівень масштабованості.

Розроблений застосунок дозволить медичному персоналу швидко формувати, редагувати, зберігати та знаходити лікарські документи. У подальшому передбачається розширення функціоналу застосунку, включаючи покращення інтерфейсу та підтримку інших форм документів.

Результати роботи були опубліковані в рамках міжнародної науково-технічної конференції «Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління» [31].

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. Bonfatti F., Monari P. D., Martinelli L. Business Document Exchange between Small Companies. *Electronic Business Interoperability: Concepts, Opportunities and Challenges*. 2011. P. 482-510.

2. Ефективність систем електронного документообігу: Практичний приклад. URL: https://www.imcra-az.org/uploads/public_files/2022-05/07-kenan-abaci.pdf. (дата звернення: 10.06.2025).

3. Almacen A. M. B., Cabaluna A. Y. Electronic Document Management System Implementation: Implications for the Future of Digital Transformation in Philippine Healthcare. *Journal of Computer Science and Technology Studies*. 2021. Vol. 3, No 2. P. 82-90. DOI: https://doi.org/10.32996/jcsts.2021.3.2.8

4. Про затвердження Порядку ведення Реєстру декларацій про вибір лікаря, який надає первинну медичну допомогу, в електронній системі охорони здоров'я. URL: https://zakon.rada.gov.ua/laws/show/z0324-23#n22/ (дата звернення: 11.06.2025)

5. Ismael A., Okumus I. Design and Implementation of an Electronic Document Management System. *Mehmet Akif Ersoy Üniversitesi Uygulamalı*. 2017. No 1. P. 9-17. DOI: https://doi.org/10.31200/makuubd.321093

6. Mettler T., Tobias M., Peter R. Supplier Relationship Management In Practice (SRM Practice). Swiss Hospitals, 2023. 250 p.

7. Achachlouei M. A., Patil O., Joshi T., Nair V. N. Document Automation Architectures and Technologies: A Survey. *Corporate Model Risk*. 2021. P. 1-34. DOI: https://doi.org/10.48550/arXiv.2109.11603.

8. Payne A. Handbook of CRM. London: Routledge, 2005. 458 p.

9. Baashar Y., Alhussian H., Patel A., Alkawsi G., Alzahrani A. I., Alfarraj O., Hayder G. Customer relationship management systems (CRMS) in the healthcare environment: A systematic literature review. *Computer Standards & Interfaces*. 2020. No 71. P. 1-16. DOI: https://doi.org/10.1016/j.csi.2020.103442

10. Ralph H., Sprague Jr. Electronic Document Management: Challenges and Opportunities for Information Systems Managers. *Management Information Systems Research Center, University of Minnesota*. 1995. No 19. P. 29-49. DOI: https://doi.org/10.2307/249710

11. Гід по Helsi. URL: https://bf.diia.gov.ua/articles/velikij-gid-po-helsi-yak-zareyestruvatis-i-chim-mozhe-dopomogti/ (дата звернення: 12.06.2025).

12. Про Helsi. URL: https://helsi.me/about/ (дата звернення: 12.06.2025).

13. Деталі системи Megapolis.DocNet. URL: https://inbase.com.ua/vsi-produkty/megapolis/details/ (дата звернення: 12.06.2025).

14. DocuWare. URL: https://start.docuware.com/docuware-cloud/ (дата звернення: 12.06.2025).

15. Середовище OpenText. URL: https://www.opentext.com/ (дата звернення: 12.06.2025).

16. McConnell S. Code Complete: A Practical Handbook of Software Construction, Second Edition, Microsoft Press, 2004. 960 p.

17. Griffiths I. Programming C# 10 Build Cloud, Web, and Desktop Applications. O'Reilly Media, 2022. 836 p.

18. Liberty J. Programming C# Building .NET Applications with C#. O'Reilly Media, 2009. 672 p.

19. Likness J. Building Windows 8 Apps with C# and XAML. Pearson Education, 2012. 384 p.

20. MacDonald M. Pro WPF 4.5 in C#: Windows Presentation Foundation in .NET 4.5, 4th Edition, Apress, 2012. 1111 p.

21. NuGet. URL: https://learn.microsoft.com/en-us/dotnet/standard/library-guidance/nuget/ (дата звернення: 13.06.2025).

22. Kort D. W. Creating and Sharing Packages. *DevOps on the Microsoft Stack.* 2016. No 1. P. 189-201.

23. Borucki A., Domingues L., Hammad M. A., Hannouch E., Nair R., Palmer R. Mastering MongoDB 7.0: Achieve data excellence by unlocking the full potential of MongoDB, 4th Edition, Packt Publishing, 2024. 434 p.

24. Пунда С. Ю. Системи збереження даних для IT інфраструктури. *Проблеми програмування*. 2020. № 2-3. С. 82-93.

25. About GitHub and Git. URL: https://docs.github.com/en/get-started/start-your-journey/about-github-and-git/ (дата звернення: 17.06.2025).

26. GitHub Codespaces features. URL: https://docs.github.com/en/codespaces/about-codespaces/codespaces-features/ (дата звернення: 17.06.2025).

27. Comparing Git workflow. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/ (дата звернення: 18.06.2025).

28. Git feature branch workflow. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/feature-branch/ (дата звернення: 18.06.2025).

29. Gitflow workflow. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/gitflow-workflow (дата звернення: 18.06.2025).

30. Forking workflow. URL: https://www.atlassian.com/git/tutorials/comparing-workflows/forking-workflow (дата звернення: 18.06.2025).

31. Смотрова А. О., Іващенко Г. С. Автоматизація документообігу лікарні. *Сучасні напрями розвитку інформаційно-комунікаційних технологій та засобів управління.* 2025. Т. 2. С. 20. DOI: https://doi.org/10.32620/ICT.25.t2