МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ

Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами

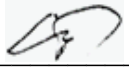
Кафедра програмного забезпечення автоматизованих систем

Спеціальність 121 «Інженерія програмного забезпечення»

Освітня програма «Інженерія програмного забезпечення»

«Допущений до захисту»

Завідувач кафедри

 проф. Приходько С.Б.

«__» _____ 2025 р.

КВАЛІФІКАЦІЙНА РОБОТА

на здобуття ступеня вищої освіти «бакалавр»

на тему: Розробка вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Виконав: студент групи 4151

 Смоквіна В.М

(підпис, ПІБ)

Керівник роботи:

_доцент кафедри ПЗАС, к.т.н.

(посада, науковий ступень, вчене звання)

__  ____Пухалевич А.В.

(підпис, ПІБ)

Миколаїв – 2025 р.

МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
Національний університет кораблебудування імені адмірала Макарова

Навчально-науковий інститут комп'ютерних наук та управління проектами
Кафедра програмного забезпечення автоматизованих систем
Спеціальність 121 «Інженерія програмного забезпечення»
Освітня програма «Інженерія програмного забезпечення»

«ЗАТВЕРДЖУЮ»
Гарант освітньої програми
_____ доц. Макарова Л.М.
(підпис)

« 08 » 04 2025 р.

ЗАВДАННЯ
НА КВАЛІФІКАЦІЙНУ РОБОТУ
на здобуття ступеня вищої освіти «бакалавр»

Студенту Смоквіні Вадиму Миколайовичу
(Прізвище, ім'я, по батькові)

1. Тема роботи: Розробка вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"
Керівник роботи доцент кафедри ПЗАС, к.т.н. Пухалевич Андрій Володимирович
Затверджені наказом ректора №153 від 08.04.2025 р.
2. Термін подання роботи: 02.06.2025 р.
3. Вихідні дані по роботі:
4. Перелік питань, що належать до розробки (найменування розділів)

Титульний аркуш; Завдання на кваліфікаційну роботу; Анотація (українською та англійською мовами); Зміст; Перелік умовних позначень, символів, одиниць та термінів (за потреби); Вступ; Аналіз практичної задачі інженерії програмного забезпечення; Проєкт програмного забезпечення; Результати розробки програмного забезпечення; Розділ з охорони праці; Висновки; Список використаних джерел; Додатки (технічне завдання, текст програми, опис програми, інструкція користувача, програма та методика випробувань програмного забезпечення).

5. Перелік презентаційних матеріалів 1. Титульний слайд, 2. Мета та завдання кваліфікаційної роботи, 3. Аналіз практичної задачі інженерії програмного забезпечення, 4. Аналіз вимог до ПЗ, 5. Діаграма варіантів використання, 6. Архітектура ПЗ, 7. Діаграма діяльності (Прецедент "Призначити новий прийом пацієнта до лікаря"), 8. Діаграма діяльності (Прецедент "Перегляд розкладу роботи певного лікаря"), 9. Концептуальна модель даних, 10. Діаграма класів, 11. Діаграма класів, 12. Діаграма послідовності (Прецедент "Призначити новий прийом пацієнта до лікаря"), 13. Логічна модель бази даних, 14. Фізична модель бази даних, 15. Результати розробки, 16. Висновки.

5. Консультанти розділів роботи

Розділ	Прізвище, ініціали та посада консультанта	Підпис, дата	
		Завдання видав	Завдання прийняв
4	Гурець Н.В., ст. викладач		

7. Дата видачі завдання 08.04.2025 р.

К А Л Е Н Д А Р Н И Й П Л А Н

Номер	Назва етапів роботи	Терміни виконання	Примітка
1	Підготовка розділу «ВСТУП»	31.03.2025	ПП*
2	Аналіз практичної задачі інженерії ПЗ	07.04.2025	ПП*
3	Аналіз вимог до ПЗ	14.04.2025	ПП*
4	Розробка архітектури ПЗ	21.04.2025	
5	Розробка проєкту ПЗ	05.05.2025	
6	Реалізація та тестування ПЗ	12.05.2025	
7	Підготовка розділу «Результати розробки ПЗ»	16.05.2025	
8	Підготовка розділу з охорони праці	19.05.2025	ПП*
9	Підготовка розділу «ВИСНОВКИ»	23.05.2025	
10	Оформлення списку використаних джерел та додатків	26.05.2025	
11	Подання на кафедру ПЗАС тексту остаточного варіанта роботи, підписаного її керівником, у роздрукованому та електронному форматі разом із заявами щодо самостійності виконання роботи та ідентичності друкованої та електронної версій роботи (дод. 1 і 2 «Порядку здійснення заходів з перевірки робіт на наявність текстових збігів / ідентичності / схожості з використанням програмно-технічних засобів», який введений у дію наказом ректора НУК за № 20 від 20.01.2020 р.)	02.06.2025	не пізніше ніж за 14 діб до захисту (згідно з п.4.1 зазначеного Порядку)
12	Підготовка презентації та доповіді	06.06.2025	
13	Попередній захист роботи на засіданні кафедри ПЗАС	09.06.2025	
14	Подання на кафедру ПЗАС електронних версій наступних документів у форматі pdf: кваліфікаційної роботи; файла-опису кваліфікаційної роботи (згідно з Додатком до наказу ректора НУК за № 287-уч від 19.05.2020 р.); презентації доповіді, а також Авторського договору з додатком	16.06.2025	

* За результатами переддипломної практики (ПП), яка була з 03.02.2025 по 23.03.2025 р.

Студент _____ Смоквіна В.М
(підпис) (ПІБ)

Керівник роботи _____ Пухалевич А.В.
(підпис) (ПІБ)

АНОТАЦІЯ

Кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

Було проаналізовано практичну задачу інженерії програмного забезпечення з розробки вебзастосунку з автоматизації роботи реєстратури медичної клініки "CompliMed", розглянуто існуючі аналоги та виконано постановку задачі на розробку вебзастосунку. Здійснено аналіз вимог до програмного забезпечення. Розроблено ескізний, технічний та робочий проєкти програмного забезпечення, представлено результати розробки та спеціальний розділ з охорони праці.

Кваліфікаційна робота викладена на 126 сторінках друкованого тексту та містить 27 рисунків, 17 таблиць, 5 додатків та список використаних джерел з 37 найменувань.

ABSTRACT

The qualification work is devoted to solving the practical problem of software engineering, characterized by complexity and uncertainty of conditions, namely the development of a web application for automating the work of the reception of the "CompliMed" medical clinic.

The practical task of software engineering for the development of a web application for automating the work of the reception of the CompliMed medical clinic was analyzed, existing analogues were considered, and the task of software development was formulated. The software requirements were analyzed. The sketch, technical and working projects of the software were developed, the development results and a special section on labor protection were presented.

The qualification work is presented on 126 pages of printed text and contains 27 figures, 17 tables, 5 appendices and a list of references of 37 titles.

ЗМІСТ

ВСТУП

Реєстратура медичної клініки є важливим вузлом в організації медичного обслуговування пацієнтів. Відсутність якісного програмного забезпечення в реєстратурі медичної клініки може призвести до втрати даних, труднощів з їх збереженням, помилок у веденні медичних карток та затримок у обслуговуванні пацієнтів, що ускладнює роботу персоналу та знижує ефективність [1].

Один зі способів полегшення роботи медичного персоналу та покращення обслуговування пацієнтів – використання спеціального вебзастосунку. З його допомогою реєстратори зможуть швидко та зручно призначати прийоми пацієнтів до лікарів в зручний для них час відповідно до графіків роботи, а лікарі вносити та переглядати результати обстежень [2].

Дана кваліфікаційна робота присвячена вирішенню практичної задачі інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розробці вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed". Ця розробка є ініціативною та узгоджена з медичною клінікою "CompliMed".

Обрана практична задача інженерії програмного забезпечення характеризується комплексністю та невизначеністю умов через необхідність прийняття до уваги багатьох важливих факторів, серед яких – підтримка різних рівнів доступу до інформації для персоналу медичної клініки, що вимагає ґрунтовної розробки механізмів контролю доступу, забезпечення цілісності даних та ретельно продуманих процедур для вирішення виключень. Поза тим, медичні заклади дотримуються певних інструкцій, коли йдеться про взаємодію з пацієнтами, планування прийомів та ведення документації. Зміни в законодавстві або внутрішній політиці медичної клініки можуть призвести до модифікації правил. Це вимагає гнучкості при проектуванні системи, щоб забезпечити адаптивність до варіативності бізнес-процесів.

Розглядаючи аналогічне програмне забезпечення, слід зазначити, що використання та адаптація вже існуючого програмного продукту може виявитися

не найкращим варіантом, якщо він не відповідає загальним потребам медичного закладу. Створення власного програмного забезпечення для автоматизації роботи реєстратури дозволяє розставити пріоритети по відношенню до необхідного функціоналу та максимально задовольнити потреби та очікування користувачів розробкою персонального рішення та виключенням непотрібного функціонального навантаження.

Метою кваліфікаційної роботи є розробка вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

Для досягнення поставленої мети кваліфікаційної роботи необхідно вирішити наступні завдання:

– провести аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed";

– провести аналіз існуючих аналогів програмного забезпечення для автоматизації роботи реєстратури медичних закладів та зробити висновок про доцільність розробки власного програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed";

– сформулювати постановку задачі на розробку вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" та розробити технічне завдання;

– розробити архітектуру вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed";

– розробити проект вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed";

– виконати тестування і випробовування вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed";

Об'єктом кваліфікаційної роботи є процес розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

1 АНАЛІЗ ПРАКТИЧНОЇ ЗАДАЧІ ІНЖЕНЕРІЇ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ТА ПОСТАНОВКА ЗАДАЧІ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ РЕЄСТРАТУРИ МЕДИЧНОЇ КЛІНІКИ "COMPLIMED"

1.1 Аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

В даній роботі розв'язується нетривіальна задача розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки “CompliMed”. Вирішення цієї задачі потрібує створення вебсайту, який буде містити сторінки, необхідні для автоматизації процесів, що відбуваються в реєстратурі медичної клініки. У межах дослідження аналізується робота реєстратури медичної клініки "CompliMed".

В реєстратурі відбуваються процеси обліку пацієнтів, лікарів, прийомів пацієнтів до лікарів та роботу з ними. Здебільшого при записі на прийом виникають певні складнощі через незручність роботи з паперовими носіями, а також нестачу легких і зрозумілих інформаційних засобів реєстрації.

Процес запису на прийом до лікаря можна сформулювати таким чином: пацієнт звертається в реєстратуру і дізнається розклад роботи необхідного йому лікаря та кабінет в якому він приймає. Реєстратура має інформацію про кожного з пацієнтів, яка зберігається на певній картці, в якій зазначено:

– прізвище; ім’я; по-батькові пацієнта;

– дата народження;

– номер телефону;

– контактний номер Viber/Telegram;

– домашня адреса;

– пільгова категорія.

Також в цій картці міститься інформація про історію звернень та захворювань пацієнта. Якщо пацієнт звертається вперше, то в реєстратурі заводять нову картку.

Звернувшись до реєстратури будь-яким зручним способ пацієнт отримує талон з направленням, у якому вказано прізвище, ім'я, по батькові пацієнта та

лікаря, спеціальність лікаря, дата та час прийому і номер кабінету. Реєстратура має таку інформацію про кожного з лікарів:

– прізвище, ім'я, по-батькові лікаря;

– дата народження;

– номер телефону;

– контактний номер Viber/Telegram

– домашня адреса;

– лікарська спеціальність;

– лікарська кваліфікаційна категорія;

– освіта;

– номер кабінету;

– дні прийому та відповідні години прийому для кожного дня по днях тижня (понеділок, вівторок, середа, четвер, п’ятниця, субота, неділя).

Існують наступні обмеження на інформацію:

– Не може бути одночасно кілька прийомів на один і той самий час в одного й того самого лікаря.

– Не можна призначити прийом пацієнту на час, на який у цього пацієнта вже є інший прийом.

– Не можна призначити прийом пацієнта до лікаря, якщо час прийому виходить за межі графіку роботи лікаря.

Для розробки вебзастосунків для автоматизації роботи реєстратури медичних закладів найпоширенішими є інтерпретовані мови програмування JavaScript/TypeScript та Python через їхню гнучкість та простоту. Водночас класичні строго типізовані мови, такі як Java та C# також часто використовуються, але їхня частка в українському IT-секторі поступово скорочується [3].

Для збереження даних використовуються як реляційні СУБД (PostgreSQL, MySQL, Oracle), так і нереляційні, наприклад MongoDB, враховуючи різні вимоги до структури та об'єму даних [4]. Так, в системі HELSI для реалізації функцій реального часу, таких як чат між лікарем та пацієнтом під час онлайн-консультацій, була обрана MongoDB, через її високу швидкість та ефективність [5]. Широко використовується контейнеризація на основі Docker, оркестрація на Kubernetes, а

також інструменти CI/CD для автоматизації збирання, тестування і розгортання вебзастосунку. Для комплексного моніторингу використовуються інструменти візуалізації даних, такі як Grafana, які працюють з базами даних часових рядів, наприклад Prometheus або VictoriaMetrics, в той час як ці бази даних отримують дані з різних джерел через відповідних агентів [6]. Це допомогає оперативно виявляти й усувати програмні збої до того, як вони вплинуть на користувачів.

Більшість таких вебзастосунків реалізовані з використанням RESTful API, що дає змогу створювати крос-платформені рішення для роботи через браузер та мобільні пристрої. Для мінімізації зв'язності компонентів і спрощення підтримки часто використовують мікросервісну архітектуру, в якій кожен сервіс вирішує окреме завдання. Такий підхід дає змогу розробляти та розгортати сервіси різними командами, різними мовами програмування, і виділяти під кожен окремий сервіс різну кількість ресурсів. У межах такої архітектури, механізми авторизації та автентифікації зазвичай будуються на стандартах OAuth 2.0 і OpenID Connect, із впровадженням делегованої реєстрації, де доступ до застосунку спочатку отримують керівники клінік, які потім створюють облікові записи для персоналу [7].

Облік процесів в реєстратурі медичної клініки у паперовому вигляді може призвести до вразливості втрати даних у випадку пожеж, витоків води, руйнувань або інших непередбачуваних дій. Недостатній простір або умови зберігання паперових документів можуть також призвести до їхнього псування. Ручне ведення медичних карток призводить до затримок та непродуктивності роботи реєстратури, що впливає на задоволення пацієнтів та їхню довіру до медичного закладу. Тому робимо висновок про необхідність розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed", що покращить ефективність процесів обслуговування пацієнтів, зменшить кількість помилок при плануванні прийомів та надасть оперативний централізований доступ до інформації для персоналу медичної клініки.

1.2 Аналіз існуючих програмних продуктів для автоматизації роботи реєстратури медичного закладу

Існує готове програмне забезпечення, яке автоматизує низку процесів медичної клініки. Таке програмне забезпечення зазвичай допомагає керувати записом пацієнтів, веденням електронних медичних карток, фінансовими операціями та кадровим адмініструванням, що значно підвищує ефективність роботи медичного закладу. Існуюче програмне забезпечення має свої переваги та недоліки. Розглянемо та проаналізуємо декілька відомих варіантів подібного програмного забезпечення.

MedCenter+. Даний вебзастосунок пропонує спектр функцій, необхідних для ефективної роботи медичної клініки. Дане рішення забезпечує ведення обліку медичних даних про пацієнтів, керування фінансами та організацію електронної черги на прийом згідно робочих кабінетів та розкладів роботи лікарів [8]. Інтерфейс програми зображений на рисунку 1.1. Перевагами є простий інтерфейс, гнучка аналітика з використанням інструменту QlikView, доступність пробного періоду для випробовування ПЗ та відкритість інформації про ціноутворення. Недоліком даного ПЗ є висока ціна та те, що воно є хмарним, тим самим для роботи необхідно стабільне інтернет підключення до серверів постачальника "MedCenter+", це викликає занепокоєння щодо конфіденційності та безпеки даних, окрім цього, у разі технічних збоїв у постачальника, клініка може втратити доступ до інформації.

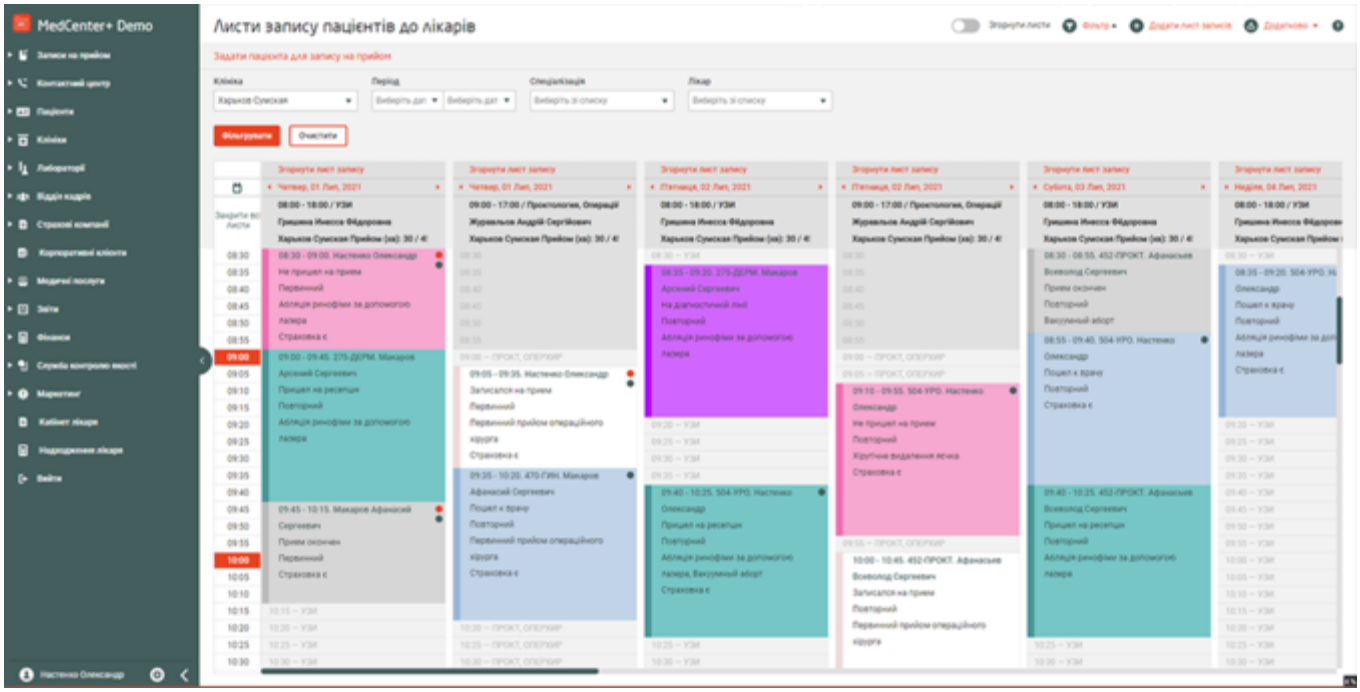


Рисунок 1.1 − Інтерфейс ПЗ “MedCenter+” [8]

Вебзастосунок "Clinica Web” створений для автоматизації медичних установ та надає низку важливого функціоналу: ведення обліку інформації про пацієнтів, їх медичних карток та виданих рецептів, створення графіків прийому, формування черги пацієнтів на прийоми до лікарів, збереження історії звернень та можливість розсилок за різними типами зв'язку, наприклад СМС або Telegram. [9]. Інтерфейс програми зображений на рисунку 1.2. Переваги: підтримка роботи особистого кабінета пацієнта, заявлене індивідуальне налаштування та можливість розробки нового функціоналу під потреби медичного закладу, інтеграції для медичного діагностичного обладнання, яке працює за протоколом DICOM. Недоліки: модель помісячної оплати, в довгостроковій перспективі ці витрати можуть стати обтяжливими, також даний застосунок є так само хмарним, як і "MedCenter+", отже може мати тіж самі проблеми, слід також додати, що в разі розширення медичного закладу, можуть виникнути проблеми з масштабуванням, у разі якщо хмарна платформа постачальника не надасть достатньої гнучкості або обчислювальної потужності для задоволення вимог.

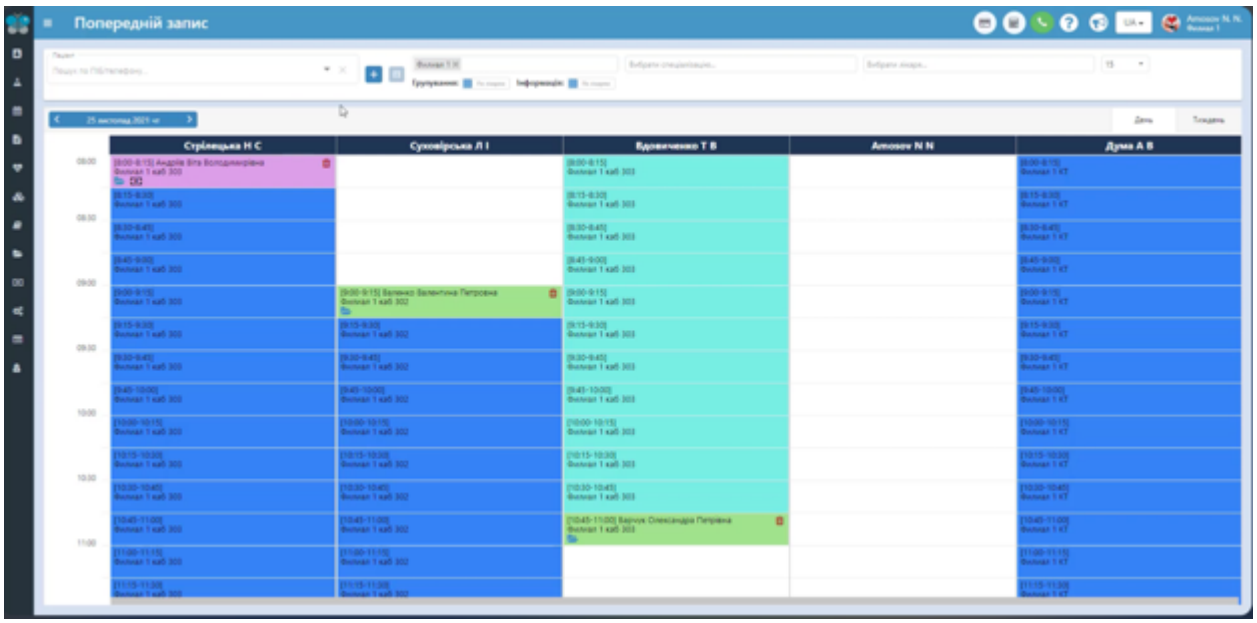


Рисунок 1.2 − Інтерфейс ПЗ “Clinica Web” [9]

Doktor Eleks – це програмне забезпечення для автоматизації медичних організацій різного рівня, від невеликих клінік до великих медичних центрів. Воно забезпечує управління медичними та адміністративними процесами, включаючи електронну медичну документацію, управління записами на прийом, фінансовий облік, управління персоналом та аналітику [10]. Інтерфейс програми зображений на рисунку 1.3.

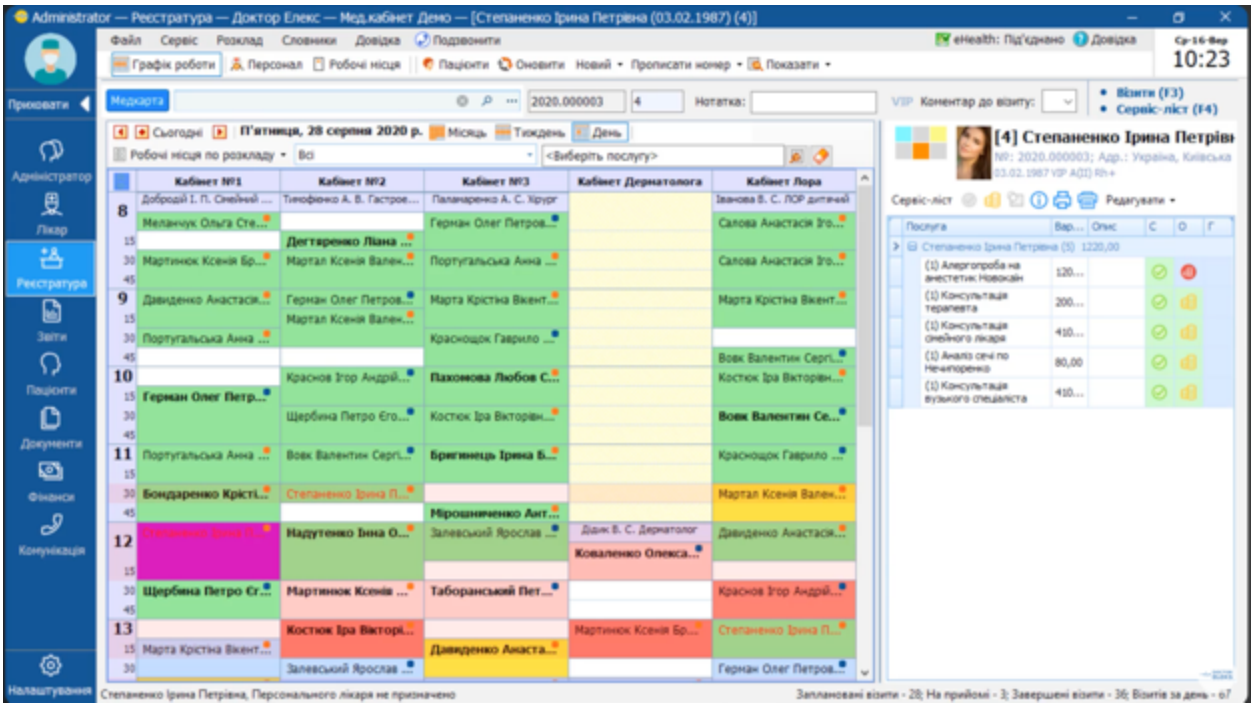


Рисунок 1.3 − Інтерфейс ПЗ “Doktor Eleks” [10]

Переваги:

– Підтримка роботи на Windows, а також вебверсія та мобільний застосунок.

– Підтримка телемедицини – можливість онлайн-консультацій через вбудовані інструменти.

Недоліки:

– Ціна – програмне забезпечення не безкоштовне, а вартість залежить від обраних модулів та кількості робочих місць. Вартість не вказана на сайті розробника, для отримання комерційної пропозиції треба залишити заявку, що ускладнює процес прийняття рішення.

– Для ефективного використання даного ПЗ може знадобитись наявність більш сучасного обладнання, що може вимагати додаткових інвестицій.

– Складність впровадження – для налаштування потрібна технічна підтримка та спеціалізоване навчання персоналу.

Хоча дане ПЗ має як десктопну так і вебверсію, слід зазначити що це ускладнює процес розробки та тестування, так як треба робити це для кожної окремої платформи, крім того це потребує окремої технічної підтримки. Також це може призвести до специфічних вимог до сумісності та синхронізації функцій і даних, що ускладнює процес підтримки актуальності всіх версій програмного забезпечення. Користувацький інтерфейс на різних платформах може виглядати або працювати дещо по різному згідно обмежень або особливостей операційної системи. Використовуючи один вебзастосунок для автоматизації роботи реєстратури, можна забезпечити однаковий досвід при використанні застосунку на різних пристроях.

З розглянутих готових програмних рішень, можна зробити висновок, що жодне не задоволняє вимогам медичної клініки “CompliMed”. Основними недоліками аналогічних продуктів є висока ціна, хмарність програмного забезпечення, зайве для медичної клініки "CompliMed" функціональне навантаження та складний процес впровадження та супроводу. Виходячи з цього виникає необхідність створення власного програмного забезпечення для вирішення задачі автоматизації роботи реєстратури медичної клініки "CompliMed".

1.3 Постановка задачі на розробку вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Виходячи з проведеного аналізу задачі інженерії програмного забезпечення розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed", яку має на меті розв'язувати кваліфікаційна робота, сформулюємо наступну постановку задачі: необхідно розробити вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed".

З даним застосунком повинні мати можливість працювати такі групи користувачів:

– лікарі;

– працівники реєстратури;

– адміністратори.

При роботі з застосунком **працівник реєстратури** повинен мати можливість вирішувати такі завдання:

1) Перегляд списків всіх пацієнтів, лікарів, прийомів, кабінетів разом із можливістю пошуку.

2) Перегляд персональних даних конкретного пацієнта.

3) Перегляд персональних даних конкретного лікаря та його графік роботи.

4) Перегляд даних про конкретний кабінет.

5) Перегляд даних про конкретний прийом пацієнта до лікаря.

6) Перегляд розкладу роботи певного лікаря на певну дату.

7) Додавання нових графіків роботи лікарів, коригування та видалення наявних графіків роботи лікарів до кожного дня тижня.

8) Додавання нових пацієнтів, коригування, та видалення даних про наявних пацієнтів.

9) Призначення, коригування та скасування прийомів пацієнтів з лікарями (без можливості редагування інформації про консультації в прийомах, таку як: симптоми, діагнози та медичні рекомендації).

Лікар повинен мати можливість вирішувати такі завдання:

1) Додавання нових пацієнтів, коригування, та видалення даних про наявних пацієнтів.

2) Перегляд списків всіх пацієнтів та прийомів разом із можливістю пошуку за доступними критеріями.

3) Перегляд персональних даних конкретного пацієнта.

4) Перегляд даних про конкретний прийом пацієнта до лікаря.

5) Перегляд свого розкладу роботи на певну дату.

6) Призначення, коригування та скасування прийомів пацієнтів до себе.

7) Перегляд інформації про консультації в усіх прийомах.

8) Внесення інформації про консультації в своїх прийомах.

Адміністратор успадковує всю функціональність **реєстратора** та має наступні додаткові функції:

1) Додавання нових реєстраторів (разом із створенням облікового запису реєстратора), коригування, видалення даних про наявних реєстраторів.

2) Додавання нових лікарів (разом із створенням облікового запису лікаря), коригування, та видалення даних про наявних лікарів.

3) Перегляд списків всіх реєстраторів разом із можливістю пошуку за доступними критеріями.

4) Перегляд персональних даних конкретного реєстратора.

5) Перегляд інформації про консультації в усіх прийомах.

6) Додавання нових кабінетів, коригування, та видалення даних про наявні кабінети.

Вимоги до складу виконуваних функцій

Вебзастосунок повинен забезпечити можливість виконання нижче перерахованих функцій

1) Авторизація

Вхідні дані: логін, пароль.

Вихідні дані: повідомлення про успішність авторизації.

2) Додавання нового пацієнта

Вхідні дані: номер пацієнта, прізвище, ім’я, по батькові пацієнта, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

Вихідні дані: повідомлення про додавання нового пацієнта.

3) Оновлення даних про пацієнта

Вхідні дані: номер пацієнта, прізвище, ім’я, по батькові пацієнта, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

Вихідні дані: повідомлення про оновлення даних про пацієнта.

4) Видалення пацієнта

Вхідні дані: номер пацієнта.

Вихідні дані: повідомлення про видалення пацієнта.

5) Додавання нового лікаря

Вхідні дані: номер лікаря, прізвище, ім'я, по батькові лікаря, лікарська спеціальність, лікарська кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер кабінету.

Вихідні дані: повідомлення про додавання нового лікаря з згенерованим логіном та паролем від створеного облікового запису.

6) Оновлення даних про лікаря

Вхідні дані: номер лікаря, прізвище, ім'я, по батькові лікаря, лікарська спеціальність, лікарська кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер кабінету.

Вихідні дані: повідомлення про оновлення даних про лікаря.

7) Видалення даних про лікаря

Вхідні дані: номер лікаря.

Вихідні дані: повідомлення про видалення лікаря.

8) Додавання нового прийому пацієнта до лікаря

Вхідні дані: номер прийому, номер пацієнта, номер лікаря, дата, час початку прийому, час закінчення прийому.

Вихідні дані: повідомлення про додавання нового прийому.

9) Оновлення даних про прийом

Вхідні дані: номер прийому, номер пацієнта, номер лікаря, дата, час початку прийому, час закінчення прийому.

Вихідні дані: повідомлення про оновлення даних про прийом.

10) Видалення даних про прийом

Вхідні дані: номер прийому.

Вихідні дані: повідомлення про видалення даних про прийом.

11) Додавання нового реєстратора

Вхідні дані: номер реєстратора, прізвище, ім'я, по батькові реєстратора, дата народження.

Вихідні дані: повідомлення про додавання нового реєстратора з згенерованим логіном та паролем від створеного облікового запису.

12) Оновлення даних про реєстратора

Вхідні дані: номер реєстратора, прізвище, ім'я, по батькові реєстратора, дата народження.

Вихідні дані: повідомлення про оновлення даних про реєстратора.

13) Видалення даних про реєстратора

Вхідні дані: номер реєстратора.

Вихідні дані: повідомлення про видалення даних про реєстратора.

14) Внесення даних про графік роботи лікаря

Вхідні дані: номер графіку роботи, номер лікаря, номер дня тижня, час початку роботи, час закінчення роботи.

Вихідні дані: повідомлення про оновлення графіку роботи.

15) Додавання нового кабінету

Вхідні дані: номер кабінету, номер кабінету в медичній клініці, назва кабінету.

Вихідні дані: повідомлення про додавання нового кабінету.

16) Оновлення даних про кабінет

Вхідні дані: номер кабінету, номер кабінету в медичній клініці, назва кабінету.

Вихідні дані: повідомлення про оновлення даних про кабінет.

17) Видалення даних про кабінет

Вхідні дані: номер кабінету.

Вихідні дані: повідомлення про видалення даних про кабінет.

18) Внесення даних про консультацію прийому

Вхідні дані: номер прийому, симптоми, діагноз, медичні рекомендації.

Вихідні дані: повідомлення про оновлення даних про консультацію.

19) Формування інформації про пацієнтів

Вхідні дані: запит на перегляд списку всіх пацієнтів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові пацієнта, датою народження, або номер пацієнта для перегляду інформації про певного пацієнта.

Вихідні дані:

– список усіх пацієнтів: номер пацієнта, прізвище, ім’я, по батькові, дата народження.

– інформація про певного пацієнта: номер пацієнта, прізвище, ім’я, по батькові, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

20) Формування інформації про лікарів

Вхідні дані: запит на перегляд списку всіх лікарів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові лікаря, датою народження, лікарською спеціальністю, або номер лікаря для перегляду інформації про певного лікаря.

Вихідні дані:

– список усіх лікарів: номер лікаря, прізвище, ім’я, по батькові, дата народження, лікарська спеціальність, номер кабінету.

– інформація про певного лікаря: номер лікаря, прізвище, ім’я, по батькові, лікарська спеціальність, кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер та назва кабінету.

21) Формування інформації про реєстраторів

Вхідні дані: запит на перегляд списку всіх реєстраторів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові реєстратора, датою народження, або номер реєстратора для перегляду інформації про певного реєстратора.

Вихідні дані:

– список усіх реєстраторів: номер реєстратора, прізвище, ім’я, по батькові, дата народження.

– інформація про певного реєстратора: номер реєстратора, прізвище, ім’я, по батькові, дата народження.

22) Формування інформації про прийоми

Вхідні дані: запит на перегляд списку всіх прийомів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові пацієнта, прізвищем, ім’ям, по батькові лікаря, датою прийому, часом початку прийому, або номер прийому для перегляду інформації про певний прийом.

Вихідні дані:

– список усіх прийомів: номер прийому, пацієнт (ПІБ), лікар (ПІБ), номер кабінету в медичній клініці, дата та час початку прийому.

– інформація про певний прийом: номер прийому, лікар (ПІБ), пацієнт (ПІБ), номер кабінету в медичній клініці та назва кабінету лікаря, дата прийому, час початку прийому, час закінчення прийому.

23) Формування інформації про консультації

Вхідні дані: номер прийому.

Вихідні дані: симптоми, діагноз, медичні рекомендації.

24) Формування інформації про графіки роботи лікарів

Вхідні дані: номер лікаря.

Вихідні дані: день тижня, час початку роботи, час закінчення роботи.

25) Формування інформації про кабінети

Вхідні дані: запит на перегляд списку всіх кабінетів з опціональними критеріями пошуку за назвою кабінету, номером кабінету, або номер кабінету для перегляду інформації про певний кабінет.

Вихідні дані:

– список усіх кабінетів: номер кабінету в медичній клініці, назва кабінету.

– інформація про певний кабінет: номер кабінету в медичній клініці, назва кабінету.

26) Формування інформації про розклад роботи лікарів

Вхідні дані: номер лікаря, дата.

Вихідні дані: набір часових проміжків по 30 хвилин згідно графіку роботи лікаря на певну дату, прийоми лікаря до кожного часового проміжку або вільний стан часового проміжку.

Вимоги до організації вхідних і вихідних даних

Вхідною інформацією є дані, які вводяться вручну у поля та форми. Залежно від типу поля, дані можуть бути: числові цілі, текстові, дати, або варіанти, обрані з представлених списків (для уникнення можливих помилок, у випадних списках можуть бути представлені готові варіанти вибору).

Вихідними даними є результати виконаних операцій, які відображаються у вигляді екранних форм або повідомлень.

Вимоги до складу й параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на сервері наступної мінімальної комплектації:

– Процесор: чотирьохядерний процесор з тактовою частотою не менше 2,0 Ггц.

– Оперативна пам’ять: 8 ГБ.

– Системний диск (HDD/SDD): 150 ГБ вільного дискового простору. Рекомендовано використовувати SSD для покращення швидкості завантаження та роботи застосунку.

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

Система повинна задовольняти вказаним вимогам на клієнтському комп’ютері наступної мінімальної комплектації:

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

Вимоги до інформаційної й програмної сумісності

На сервері повинна бути встановлена одна з операційних систем: Windows 11 не нижче версії 21H2, Linux Ubuntu не нижче версії 22.04.3 LTS, Linux Debian не нижче версії 12, Linux Fedora не нижче версії 38 або MacOS версії не нижче Monterey. На сервері повинен бути встановлений Docker Desktop не нижче версії 4.24.0. На клієнтському комп’ютері повинен бути встановлений веббраузер з підтримкою HTML5.

Повні вимоги викладені у технічному завданні, яке представлено у додатку А.

2 РОЗРОБКА ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ РЕЄСТРАТУРИ МЕДИЧНОЇ КЛІНІКИ "COMPLIMED"

2.1 Аналіз вимог до вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

У даному підрозділі формалізовано вимоги до вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" шляхом створення діаграми варіантів використання та надання специфікацій варіантів використання.

2.1.1 Побудова моделі варіантів використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

При аналізі предметної галузі було виділено три актори, що взаємодіють вебзастосунком: реєстратор, лікар, адміністратор. Опис акторів наведено в таблиці 2.1.

Таблиця 2.1 – Актори вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Актор	Короткий опис
1	2
Реєстратор	Особа, котра веде облік пацієнтів, графіків роботи лікарів та призначає прийоми пацієнтів до лікарів.
Лікар	Особа, котра проводить медичні консультацій та визначає лікування для пацієнтів в медичному центрі, вводить інформацію про діагнози та медичні рекомендації щодо лікування у прийомах пацієнтів. Має можливість на призначення, редагування та видалення прийомів пацієнтів до самого себе.
Адміністратор	Особа, котра має можливість додавання нових працівників, таких як лікарі та реєстратори, редагування та видалення даних про них. Має доступ до всього функціоналу застосунку.

Виявлені варіанти використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" представлені у таблиці 2.2.

Таблиця 2.2 – Виявлені варіанти використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Код	Основні діючі особи	Назва	Короткий опис
1	2	3	4
РЛА1	Реєстратор, Лікар, Адміністратор	Авторизація	Дозволяє увійти у свій обліковий запис для отримання доступу до застосунку.
РЛА2	Реєстратор, Лікар, Адміністратор,	Ввести інформацію про нового пацієнта	Дозволяє ввести інформацію про нового пацієнта.
РЛА3	Реєстратор, Лікар, Адміністратор	Редагувати інформацію про пацієнта	Дозволяє редагувати інформацію про пацієнта.
РЛА4	Реєстратор, Лікар, Адміністратор	Видалити інформацію про пацієнта	Дозволяє видалити інформацію про пацієнта.
А5	Адміністратор	Ввести інформацію про нового лікаря	Дозволяє ввести інформацію про нового лікаря.
А6	Адміністратор	Редагувати інформацію про лікаря	Дозволяє редагувати інформацію про лікаря.
А7	Адміністратор	Видалити інформацію про лікаря	Дозволяє видалити інформацію про лікаря.
РЛА8	Реєстратор, Лікар, Адміністратор	Призначити новий прийом	Дозволяє призначити новий прийом пацієнта до лікаря.
РЛА9	Реєстратор, Лікар, Адміністратор	Перепланувати прийом	Дозволяє перепланувати прийом пацієнта до лікаря, змінивши дату, час та лікаря у існуючого прийому.
РЛА10	Реєстратор, Лікар, Адміністратор,	Видалити інформацію про прийом	Дозволяє видалити інформацію про прийом пацієнта до лікаря.

Продовження табл. 2.2

1	2	3	4
РА11	Адміністратор	Ввести інформацію про новий кабінет	Дозволяє ввести інформацію про новий кабінет.
РА12	Адміністратор	Редагувати інформацію про кабінет	Дозволяє редагувати інформацію про кабінет.
РА13	Адміністратор	Видалити інформацію про кабінет	Дозволяє видалити інформацію кабінет.
РА14	Реєстратор, Адміністратор	Заповнити графік роботи лікаря	Дозволяє заповнити графік роботи лікаря.
РЛА15	Реєстратор, Лікар, Адміністратор	Перегляд списку всіх пацієнтів	Дозволяє переглянути список всіх пацієнтів.
РЛА16	Реєстратор, Лікар, Адміністратор	Перегляд повного звіту про пацієнта	Дозволяє переглядати повний звіт про певного пацієнта.
РА17	Реєстратор, Адміністратор	Перегляд списку всіх лікарів	Дозволяє переглядати список всіх лікарів.
РА18	Реєстратор, Адміністратор	Перегляд повного звіту про лікаря	Дозволяє переглядати повний звіт про певного лікаря.
РА19	Реєстратор, Адміністратор	Перегляд графіку роботи певного лікаря	Дозволяє переглянути графік роботи певного лікаря.
РЛА20	Реєстратор, Лікар, Адміністратор	Перегляд розкладу роботи певного лікаря	Дозволяє переглянути розклад роботи певного лікаря на певну дату.
РЛА21	Реєстратор, Адміністратор, Лікар	Перегляд списку всіх прийомів	Дозволяє переглядати список всіх прийомів пацієнтів до лікарів.
РЛА22	Реєстратор, Лікар, Адміністратор	Перегляд повного звіту про прийом	Дозволяє переглядати повний звіт про певний прийом пацієнта до лікаря.
РА23	Реєстратор, Адміністратор	Перегляд списку всіх кабінетів	Дозволяє переглядати список всіх кабінетів.

Продовження табл. 2.2

1	2	3	4
РА24	Реєстратор, Адміністратор	Перегляд повного звіту про кабінет	Дозволяє переглядати повний звіт про певний кабінет.
Л25	Лікар	Ввести інформацію про консультацію прийому	Дозволяє ввести інформацію про консультацію в прийомі пацієнта до лікаря.
ЛА26	Лікар, Адміністратор	Перегляд інформації про консультацію прийому	Дозволяє переглядати інформацію про консультацію в прийомах пацієнтів до лікарів.
А27	Адміністратор	Перегляд списку всіх реєстраторів	Дозволяє переглянути список всіх реєстраторів.
А28	Адміністратор	Перегляд повного звіту про реєстратора	Дозволяє переглянути повний звіт про реєстратора.
А29	Адміністратор	Ввести інформацію про нового реєстратора	Дозволяє ввести інформацію про реєстратора.
А30	Адміністратор	Редагувати інформацію про реєстратора	Дозволяє редагувати інформацію реєстратора.
А31	Адміністратор	Видалити інформацію про реєстратора	Дозволяє видалити інформацію про реєстратора.

Ці варіанти використання допомагають розуміти, як користувачі будуть взаємодіяти з вебзастосунком для автоматизації роботи реєстратури медичної клініки "CompliMed" та які функції і можливості повинні бути реалізовані для задоволення їх потреб. Аналіз та уточнення цих варіантів використання допомагає забезпечити ефективну взаємодію користувачів з застосунком.

На рисунку 2.1 зображено діаграму варіантів використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

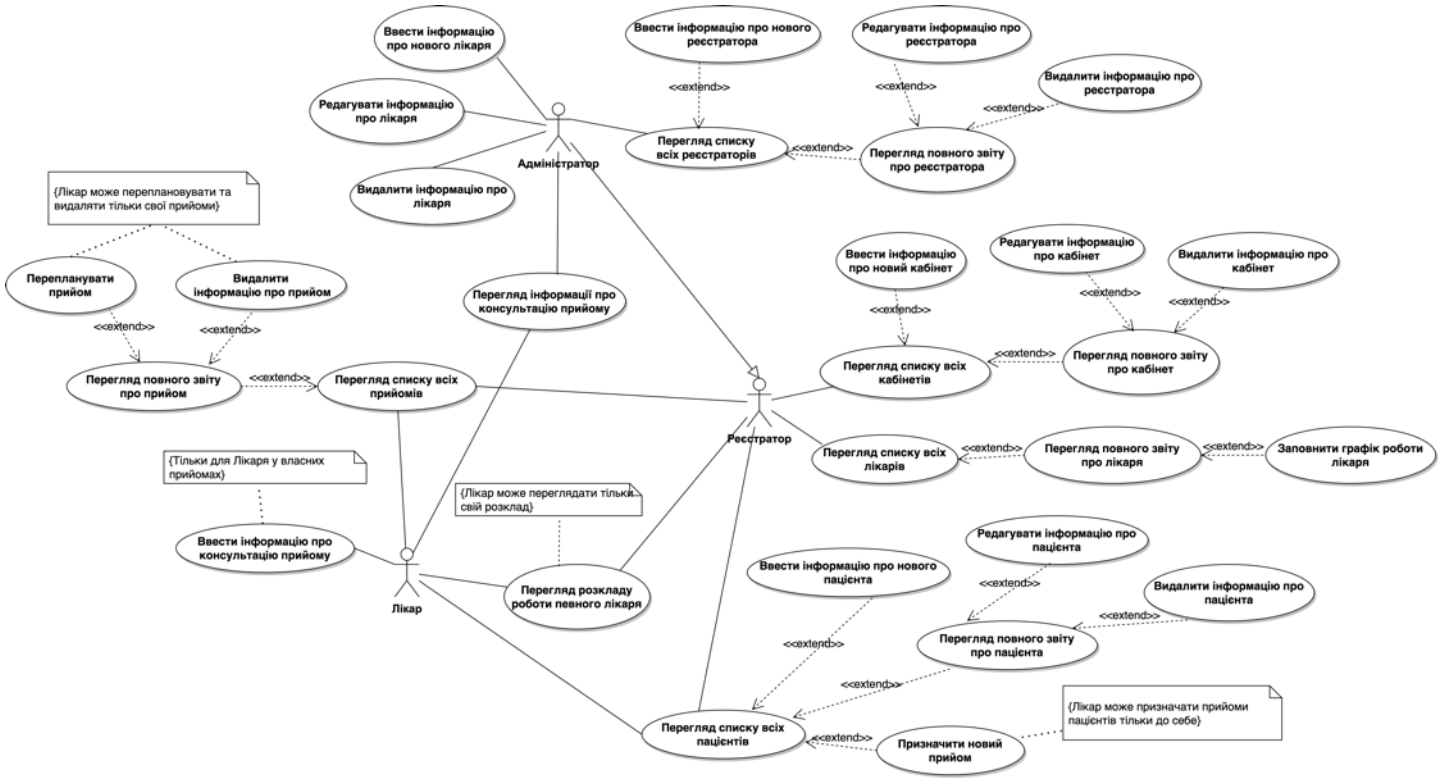


Рисунок 2.1 – Діаграма варіантів використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Діаграма варіантів використання слугує основою для окреслення архітектури програмного забезпечення.

2.1.2 Специфікації варіантів використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Для уточнення вимог було створено специфікації всіх варіантів використання. Специфікації варіантів використання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" характерні для всіх акторів представлені у таблицях 2.3 – 2.7.

Таблиця 2.3 – Специфікація варіанту використання “Ввести інформацію нового пацієнта”

Складова	Опис
1	2
Назва прецеденту	РЛА2. Ввести інформацію нового пацієнта
Опис прецеденту	Дозволяє ввести інформацію про нового пацієнта.
Дійові особи прецеденту	Реєстратор, лікар, адміністратор
Зв’язок з іншими варіантами використання	РЛА15. Перегляд списку всіх пацієнтів
Передумови	Успішна авторизація. Користувач знаходиться на сторінці списку всіх пацієнтів.
Базовий сценарій	1. Користувач натискає на кнопку ”Новий пацієнт”. 2. Система перенаправляє користувача на сторінку створення нового пацієнта та виводить форму для вводу необхідної інформації. 3. Користувач заповняє наступну інформацію про пацієнта: – прізвище; ім’я; по батькові пацієнта; – дата народження; – номер телефону; – контактний номер Viber/Telegram; – пільгова категорія; – домашня адреса. 4. Користувач підтверджує додавання нового пацієнта натискаючи на кнопку “Додати нового пацієнта”. 5. Система проводить валідацію введених даних. Дані проходять валідацію. Якщо дані не пройшли перевірку валідації. Запускається альтернативний сценарій №1. 6. Система зберігає інформацію про нового пацієнта та перенаправляє користувача на сторінку створеного пацієнта.
Альтернативний сценарій №1	1. Система виводить інформер з повідомленням про помилку біля неправильно введеного поля. 2. Сценарій продовжується від шагу №3 базового сценарію.
Постумови	Інформація про нового пацієнта збережена у системі
Особливості	Відсутні

Таблиця 2.4 – Специфікація варіанту використання “Призначити новий прийом”

Складова	Опис
1	2
Назва прецеденту	РЛА8. Призначити новий прийом
Опис прецеденту	Дозволяє призначити новий прийом пацієнта до лікаря
Дійові особи прецеденту	Реєстратор, адміністратор, лікар
Зв’язок з іншими варіантами використання	РЛА15. Перегляд списку всіх пацієнтів. РЛА20. Перегляд розкладу роботи певного лікаря
Передумови	Успішна авторизація. Користувач знаходиться на сторінці списку всіх пацієнтів.
Базовий сценарій	1. Користувач знаходячись на сторінці списку всіх пацієнтів, обирає необхідного пацієнта в таблиці та натискає на піктограму журналу у стовпчику “Дія” певного пацієнта. 2. Система перенаправляє користувача на сторінку розкладу роботи лікарів. 3. Користувач обирає необхідного лікаря з випадаючого списку лікарів. Користувач обирає бажану дату прийому. 4. Система виводить екранну форму з розкладом роботи обраного лікаря на обрану дату. Якщо лікар не має графіку роботи на обрану дату, запускається альтернативний сценарій №1. 5. Користувач натискає на кнопку “Новий прийом”. 6. Система виводить діалогове вікно з полями вводу часу початку та часу закінчення прийому на обрану дату. 7. Користувач заповнює наступні поля: – час початку; – час закінчення. 8. Користувач підтверджує призначення нового прийому натискаючи на кнопку “Призначити прийом”. 9. Система проводить перевірку введених даних про дату та час прийому на наявність конфліктів. Якщо лікар або пацієнт вже мають інший прийом на обрану дату та час запускається альтернативний сценарій №2.

Продовження табл 2.4

1	2
	10. Система перевіряє що час початку та закінчення прийому не виходить за межі графіку роботи лікаря на обрану дату. Якщо час початку або закінчення прийому виходить за межі графіку роботи лікаря на обрану дату запускається альтернативний сценарій №3 11. Система зберігає інформацію про новий прийом
Альтернативний сценарій №1	1. Система виводить повідомлення про відсутність графіку роботи лікаря на обрану дату та неможливість призначення прийому. 2. Сценарій продовжується від шагу №3 базового сценарію
Альтернативний сценарій №2	1. При введені дати та часу коли у пацієнта або лікаря вже є інший прийом система відображає повідомлення про помилку при призначенні прийому по причині конфлікту з іншими прийомами. 2. Сценарій продовжується від шагу №7 базового сценарію
Альтернативний сценарій №3	1. При введені дати, часу початку та закінчення прийому які виходять за межі графіку роботи лікаря на обрану дату система відображає повідомлення про помилку при призначенні прийому по причині неправильного часу прийому, який виходить за межі графіку роботи лікаря. 2. Сценарій продовжується від шагу №7 базового сценарію
Постумови	Інформація про новий прийом пацієнта до лікаря збережено у системі
Особливості	Реєстратор та Адміністратор можуть призначити прийом до будь-якого лікаря. Лікар може призначити прийом тільки до себе.

Таблиця 2.5 – Специфікація варіанту використання “Перегляд списку всіх пацієнтів”

Складова	Опис
1	2
Назва прецеденту	РЛА15. Перегляд списку всіх пацієнтів
Опис прецеденту	Дозволяє переглянути список всіх пацієнтів.
Дійові особи прецеденту	Реєстратор, адміністратор, лікар

Продовження табл. 2.5

1	2
Зв’язок з іншими варіантами використання	Відсутні
Передумови	Успішна авторизація
Базовий сценарій	1. Користувач натискає на кнопку “Пацієнти” у навігаційному меню 2. Система перенаправляє користувача на сторінку списку всіх пацієнтів. 3. Система виводить екранну форму з таблицею всіх пацієнтів по 5 пацієнтів на сторінку та можливість переключення сторінок. Для кожного пацієнта надається наступна інформація: – номер пацієнта; – прізвище; ім’я; по батькові пацієнта; – дата народження. 4. Користувач вводить наступні критерії у поля форми пошуку пацієнтів: – прізвище; ім’я; по батькові пацієнта; – дата народження. 5. Користувач натискає на кнопку “Пошук”. 6. Система шукає пацієнтів, що відповідають заданим критеріям та оновлює стан таблиці пацієнтів, виводячи знайдених пацієнтів. Якщо пацієнтів не знайдено запускається альтернативний сценарій №1.
Альтернативний сценарій №1	1. В випадку якщо пацієнтів за заданими критеріями не знайдено, система виводить повідомлення про відсутність пацієнтів за заданими критеріями. 2. Сценарій продовжується від шагу №4 базового сценарію.
Постумови	Відсутні
Особливості	Відсутні

Таблиця 2.6 – Специфікація варіанту використання “Перегляд повного звіту про пацієнта”

Складова	Опис
1	2
Назва прецеденту	РЛА16. Перегляд повного звіту про пацієнта
Опис прецеденту	Дозволяє переглядати повний звіт про певного пацієнта.
Дійові особи прецеденту	Реєстратор, адміністратор, лікар

Продовження табл. 2.6

1	2
Зв’язок з іншими варіантами використання	РЛА15. Перегляд списку всіх пацієнтів.
Передумови	Успішна авторизація. Користувач знаходиться на сторінці списку всіх пацієнтів
Базовий сценарій	1. Користувач знаходячись на сторінці списку всіх пацієнтів, обирає необхідного пацієнта в таблиці та натискає на піктограму ока у стовпчику “Дія” певного пацієнта. 2. Система перенаправляє користувача на сторінку певного пацієнта. 3. Система виводить екранну форму з наступною інформацією про обраного пацієнта: – прізвище; ім’я; по батькові пацієнта; – дата народження; – номер телефону; – контактний номер Viber/Telegram; – пільгова категорія – домашня адреса;
Альтернативні сценарії	Відсутні
Постумови	Відсутні
Особливості	Відсутні

Таблиця 2.7 – Специфікація варіанту використання “Перегляд розкладу роботи певного лікаря”

Складова	Опис
1	2
Назва прецеденту	РЛА20. Перегляд розкладу роботи певного лікаря
Опис прецеденту	Дозволяє переглядати розклад роботи лікаря на певну дату
Дійові особи прецеденту	Реєстратор, адміністратор, лікар
Зв’язок з іншими варіантами використання	Відсутні
Передумови	Успішна авторизація
Базовий сценарій	1. Користувач натискає на кнопку “Розклад” у навігаційному меню. 2. Система перенаправляє користувача на сторінку розкладу роботи лікарів. 3. Користувач обирає необхідного лікаря з випадаючого списку лікарів та бажану дату для перегляду розкладу роботи лікаря.

Продовження табл. 2.7

1	2
	4. Система виводить екранну форму з розкладом роботи лікаря на певну дату, в якому відображається набір часових проміжків по 30 хвилин згідно графіку роботи лікаря на певну дату та вільні й зайняті прийомами часові проміжки. Якщо лікар не має графіку роботи на обрану дату, запускається альтернативний сценарій №1.
Альтернативний сценарій №1	1. Система виводить повідомлення про відсутність графіку роботи лікаря на обрану дату та неможливість призначення прийому. 2. Сценарій продовжується від шагу №3 базового сценарію.
Постумови	Відсутні
Особливості	Реєстратор та Адміністратор можуть переглядати розклад всіх лікарів. Лікар може переглядати тільки свій розклад.

Специфікація варіанту використання “Заповнити графік роботи”, який характерний для акторів “Реєстратор” та “Адміністратор” представлена у таблиці 2.8.

Таблиця 2.8 – Специфікація варіанту використання “Заповнити графік роботи”

Складова	Опис
1	2
Назва прецеденту	РА14. Заповнити графік роботи
Опис прецеденту	Дозволяє заповнити графік роботи лікаря.
Дійові особи прецеденту	Реєстратор, адміністратор
Зв’язок з іншими варіантами використання	РА18. Перегляд повного звіту про лікаря
Передумови	Успішна авторизація. Користувач знаходиться на сторінці певного лікаря.
Базовий сценарій	1. Користувач знаходячись на сторінці певного лікаря має намір заповнити (відредагувати) графік роботи лікаря та натискає на кнопку навігації “Графік роботи” у картці інформації про лікаря. 2. Система відображає екранну форму з графіком роботи лікаря на кожен день тижня.

Продовження табл. 2.8

1	2
	3. Користувач натискає на піктограму олівця під даними про графік роботи для розблокування полей для редагування. 4. Система розблоковує поля графіку роботи та надає можливість редагування. 5. Користувач заповнює графік роботи до кожного дня тижня (понеділок - неділя), графік роботи включає в себе час початку та час закінчення роботи. Якщо лікар не працює в певний день тижня, то поле можна залишити пустим. 6. Користувач підтверджує внесені зміни, натискаючи на кнопку “Зберегти”. 7. Система проводить валідацію введених даних. Дані проходять валідацію. Якщо дані не пройшли перевірку валідації. Запускається альтернативний сценарій №1. 8. Система зберігає нову інформацію про графік роботи та виводить повідомлення про успішне редагування. Поля редагування знов блокуються.
Альтернативний сценарій №1	1. При спробі ввести час роботи де час кінця роботи менше часу початку роботи система відображає повідомлення про помилку збереження графіку роботи по причині некоректного часу роботи. 2. Сценарій продовжується від шагу №5 базового сценарію
Постумови	Інформація про графік роботи лікаря збережена у системі

Специфікація варіанту використання “Перегляд інформації про консультацію прийому”, який характерний для акторів “Лікар” та “Адміністратор” представлена у таблиці 2.9.

Таблиця 2.9 – Специфікація варіанту використання “Перегляд інформації про консультацію прийому”

Складова	Опис
1	2
Назва прецеденту	ЛА25. Перегляд інформації про консультацію прийому

Продовження табл. 2.9

1	2
Опис прецеденту	Дозволяє переглядати інформацію про консультацію в прийомах пацієнтів до лікарів.
Дійові особи прецеденту	Лікар, адміністратор
Зв’язок з іншими варіантами використання	РЛА22. Перегляд повного звіту про прийом
Передумови	Успішна авторизація. Користувач знаходиться на сторінці певного прийому
Базовий сценарій дій	1. Користувач знаходячись на сторінці певного прийому, на яку він попав виконавши варіант використання “РЛА22. Перегляд повного звіту про прийом” переглядає звіт про певний прийом формат якого описаний в зазначеному варіанту використання: 2. Система доповнює існуючу екранну форму з стандартною інформацією про прийом наступною консультаційною інформацією про прийом: – симптоми; – діагноз; – медичні рекомендації.
Альтернативні сценарії	Відсутні
Постумови	Відсутні
Особливості	Відсутні
Коментарі	Цей варіант використання є доповненням варіанту “РЛА22. Перегляд повного звіту про прийом”. Користувачу не потрібно виконувати ніяких додаткових дій для його виконання крім того що мати користувацьку роль “Лікар” або “Адміністратор”.

Специфікація варіанту використання “Ввести інформацію про консультацію прийому”, який характерний для актору “Лікар” представлена у таблиці 2.10.

Таблиця 2.10 – Специфікація варіанту використання “Ввести інформацію про консультацію прийому”

Складова	Опис
1	2
Назва прецеденту	Ввести інформацію про консультацію прийому
Опис прецеденту	Дозволяє ввести інформацію про консультацію в прийомі пацієнта до лікаря.

Дійові особи прецеденту	Лікар
Зв’язок з іншими варіантами використання	РЛА22. Перегляд повного звіту про прийом ЛА26. Перегляд інформації про консультацію прийому
Передумови	Успішна авторизація. Користувач знаходиться на сторінці певного прийому.
Базовий потік дій	1. Користувач, знаходячись на сторінці певного прийому, має намір ввести консультаційну інформацію про прийом пацієнта та натискає на піктограму олівця під даними про прийом для розблокування полей для редагування консультаційної інформації. 2. Система розблоковує поля консультації та надає можливість редагування. 3. Користувач редагує наступну інформацію про консультацію прийому: – симптоми; – діагноз; – медичні рекомендації. 4. Користувач підтверджує внесені зміни, натискаючи на кнопку “Зберегти”. 5. Система зберігає нову інформацію про консультацію прийому та виводить повідомлення про успішне редагування. Поля редагування знов блокуються.
Альтернативні сценарії	Відсутні
Постумови	Відсутні
Особливості	Лікар може редагувати консультаційну інформацію тільки в власних прийомах

За рахунок великої кількості та схожості певних варіантів використання були представлені специфікації тільки деяких варіантів використання.

2.2 Архітектура програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed"

Для реалізації вимог задачі розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" буде використаний патерн проєктування програмного забезпечення Model-View-Controller (MVC).

Компонент "Модель" в патерні проєктування MVC представляє дані та бізнес-логіку застосунку. Він відповідає за управління даними застосунку, обробку

бізнес-правил і відповіді на запити інформації від інших компонентів, таких як View і Controller.

Представлення (View) відображає дані з Моделі користувачу та надсилає вхідні дані користувача до Контролера. Він є пасивним і не взаємодіє безпосередньо з моделлю. Замість цього, він отримує дані від Моделі та надсилає вхідні дані користувача до Контролера для обробки.

Контролер діє як посередник між моделлю та представленням. Він обробляє вхідні дані користувача і відповідно оновлює Модель, а також оновлює Представлення, щоб відобразити зміни в Моделі. Він містить прикладну логіку, таку як перевірка вхідних даних і перетворення даних [11]. Діаграма взаємодії паттерну MVC зображена на рисунку 2.2.

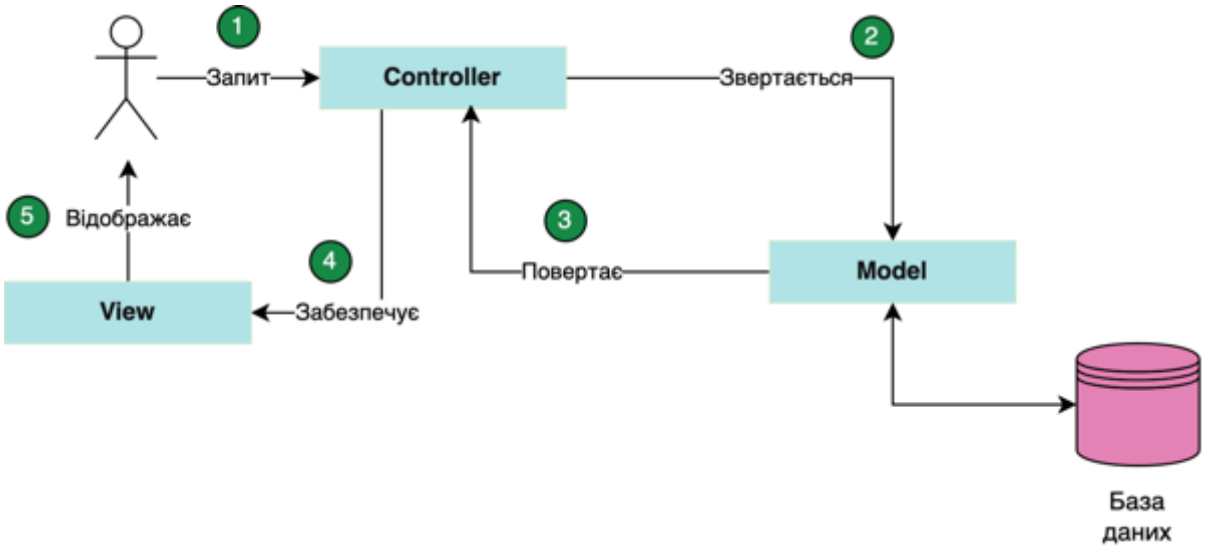


Рисунок 2.2 – Діаграма взаємодії паттерну MVC

Представити загальну структуру програмного забезпечення зручно за допомогою діаграми компонентів, яка належить до уніфікованої мови моделювання UML. Це структурна діаграма, яка використовується для зображення організації та зв'язків компонентів програмного забезпечення [12]. Діаграма компонентів має вищий рівень абстракції ніж діаграма класів, зазвичай один компонент реалізується через один або декілька класів [13].

На рисунку 2.3 представлено діаграму компонентів вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

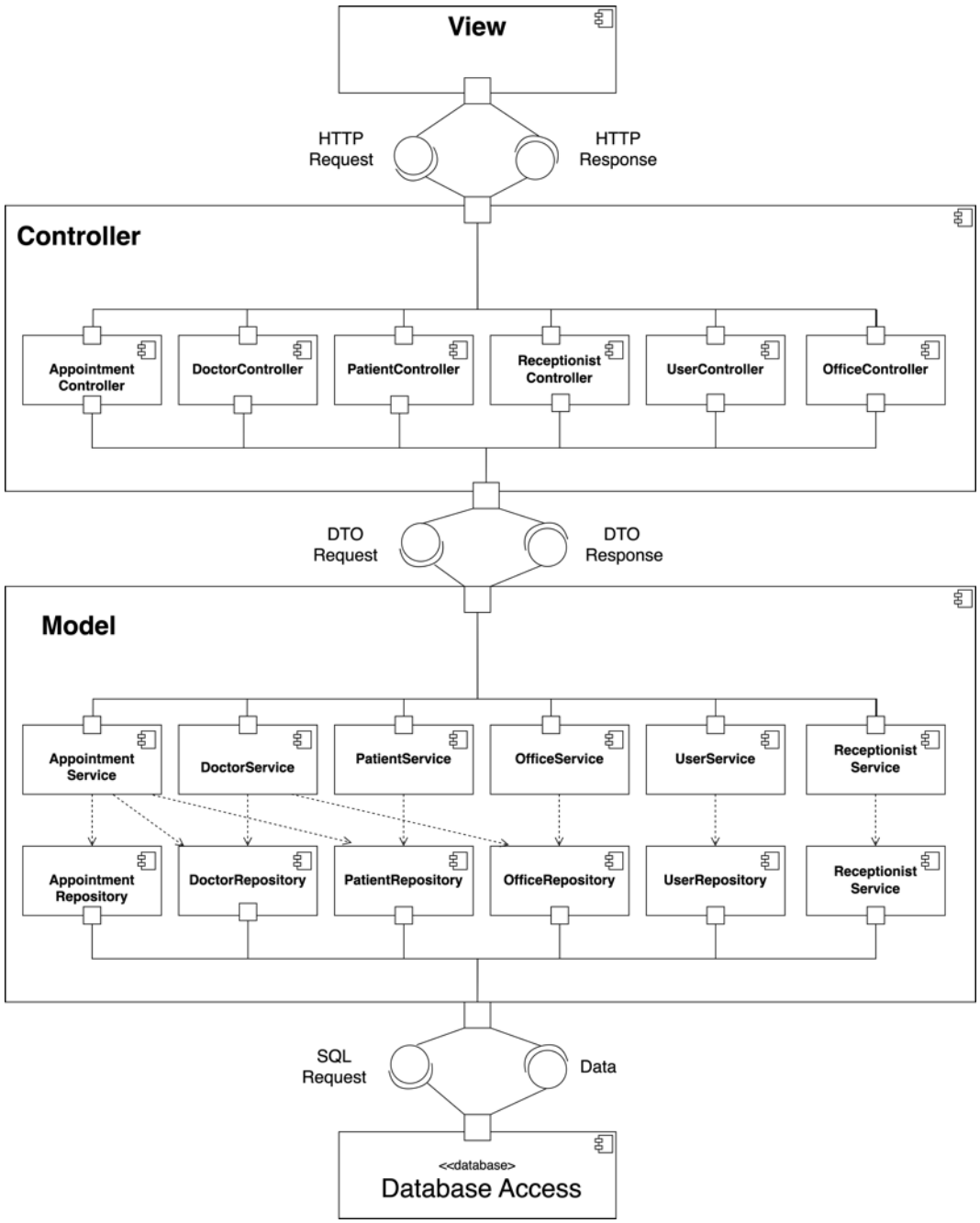


Рисунок 2.3 – Діаграма компонентів вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Для представлення фізичної архітектури системи є розумним використати діаграму розгортання UML. Діаграма розгортання є структурною діаграмою мови UML, яка показує розподіл фізичної обробки між апаратними та програмними блоками системи. На цій діаграмі описується фізична організація вузлів системи, артефакти та інші компоненти або елементи [14]. Діаграма розгортання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" представлена на рисунку 2.4.

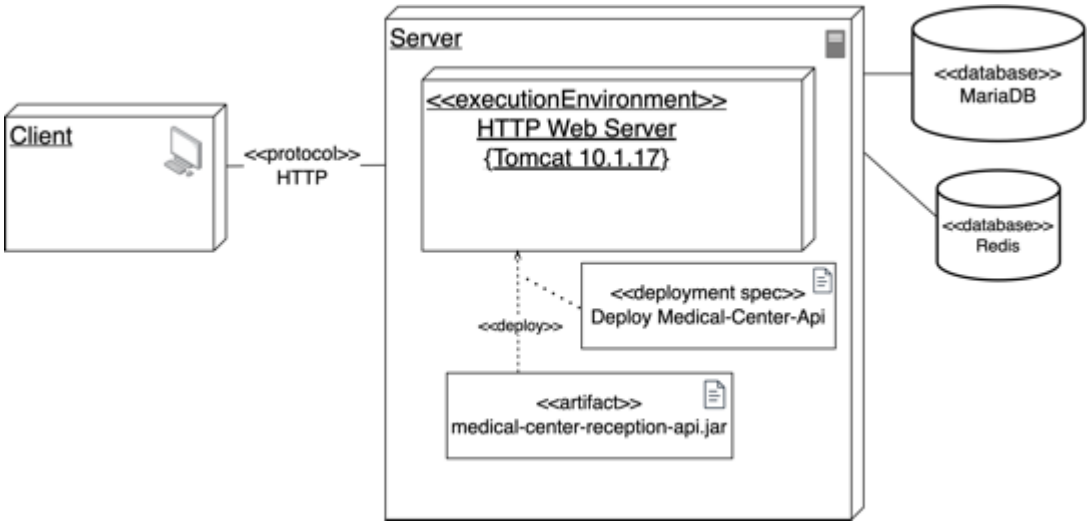


Рисунок 2.4 – Діаграма розгортання вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

У цій моделі розгортання більшість ресурсів і сервісів, до яких звертається клієнт, знаходяться, надаються та обслуговуються сервером. У цьому підході сервер, підключений до мережі, може обслуговувати одразу кілька клієнтів.

Обмін даними відбувається лише в одному напрямку, утворюючи цикл. Процес зазвичай ініціюється клієнтом, який надсилає певні дані для обробки сервером. У відповідь сервер передає клієнту необхідну інформацію за встановленим протоколом. Клієнти не спілкуються безпосередньо один з одним.

2.3 Проєкт вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

2.3.1 Ескізний проєкт вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

В рамках розробки ескізного проєкту програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed" було створено діаграми діяльності для уточнення сценаріїв варіантів використання, концептуальну модель у виді ER-діаграми та прототип користувацького інтерфейсу.

Діаграма діяльності мови UML представляє собою ілюстрації порядку дій. Ці діаграми ілюструють потік процесу від початкової точки до кінцевої, виділяючи

різні шляхи прийняття рішень, які присутні в послідовності подій в рамках роботи. На діаграмі використовуються вузли і ребра діяльності для моделювання потоку керування та даних між діями [15].

Діаграми діяльності прецедентів “Призначити новий прийом пацієнта до лікаря” та "Перегляд розкладу роботи певного лікаря" зображені на рисунках 2.5 та 2.6 відповідно.

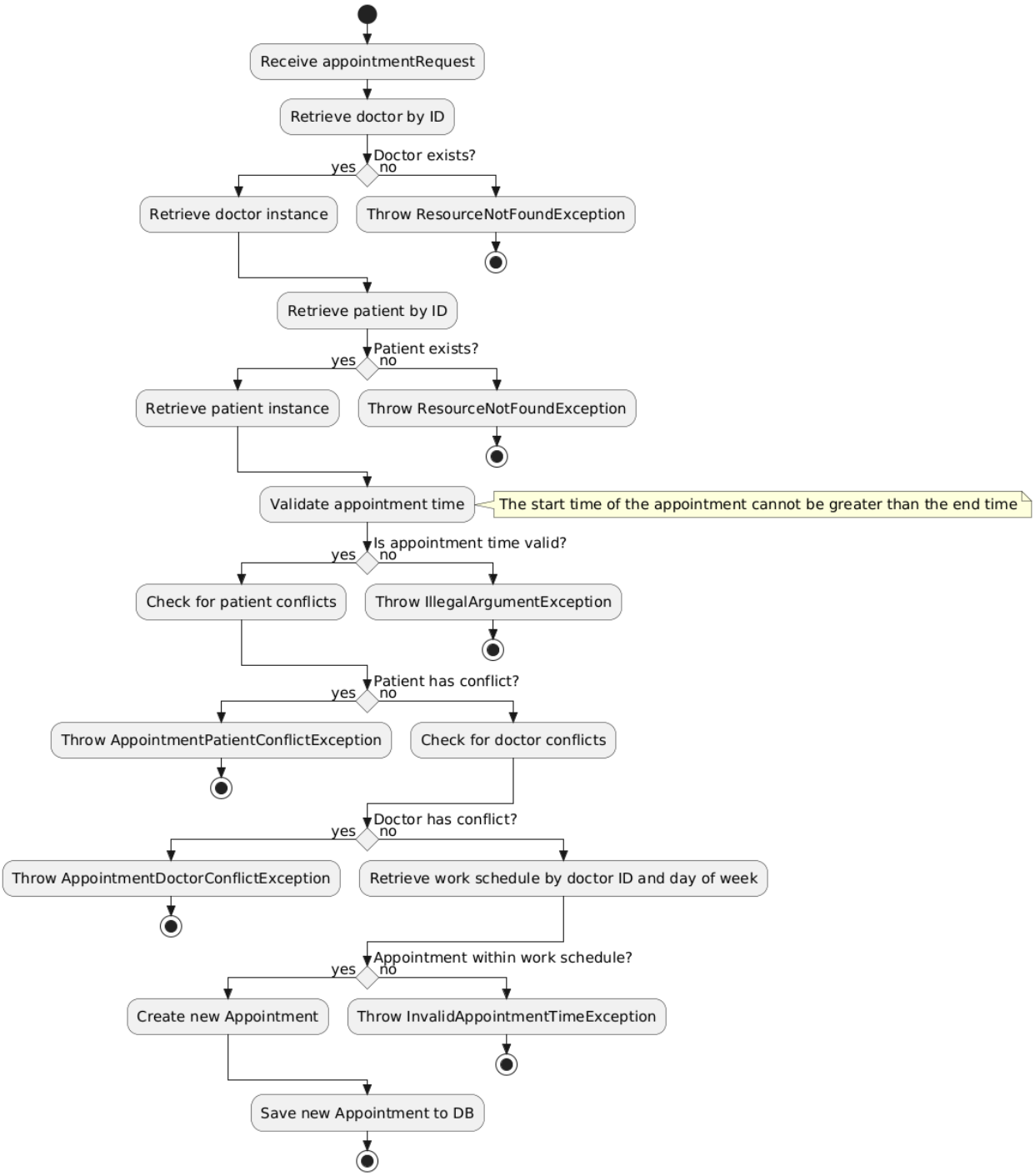


Рисунок 2.5 – Діаграма діяльності прецеденту “Призначити новий прийом пацієнта до лікаря”

Рисунок 2.6 – Діаграма діяльності прецеденту “Перегляд розкладу роботи певного лікаря”

На 2.7 представлено концептуальну модель даних у вигляді діаграми "сутність-зв'язок" (ER-модель). Діаграма “сутність-зв'язок” візуально представляє різні сутності, їх атрибути та зв'язки. Таке моделювання допомогає проаналізувати вимоги до даних для подальшого правильного проєктування бази даних [16].

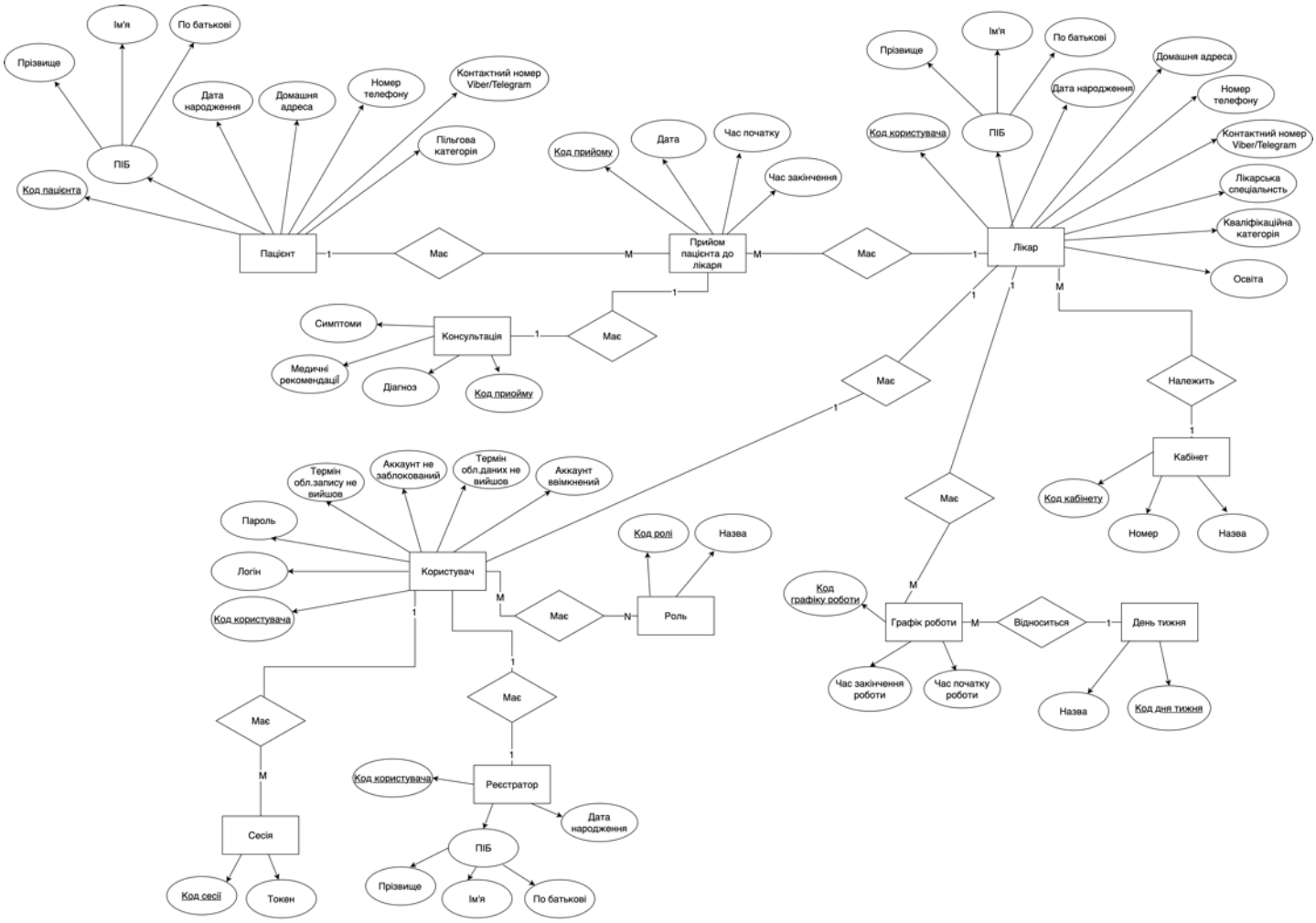


Рисунок 2.7 – Діаграма "сутність-зв'язок" вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Наступним етапом розробки ескізного проєкту програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed" є прототипування інтерфейсу користувача. На рисунках 2.8 – 2.12, зображено ескізи користувацького інтерфейсу.

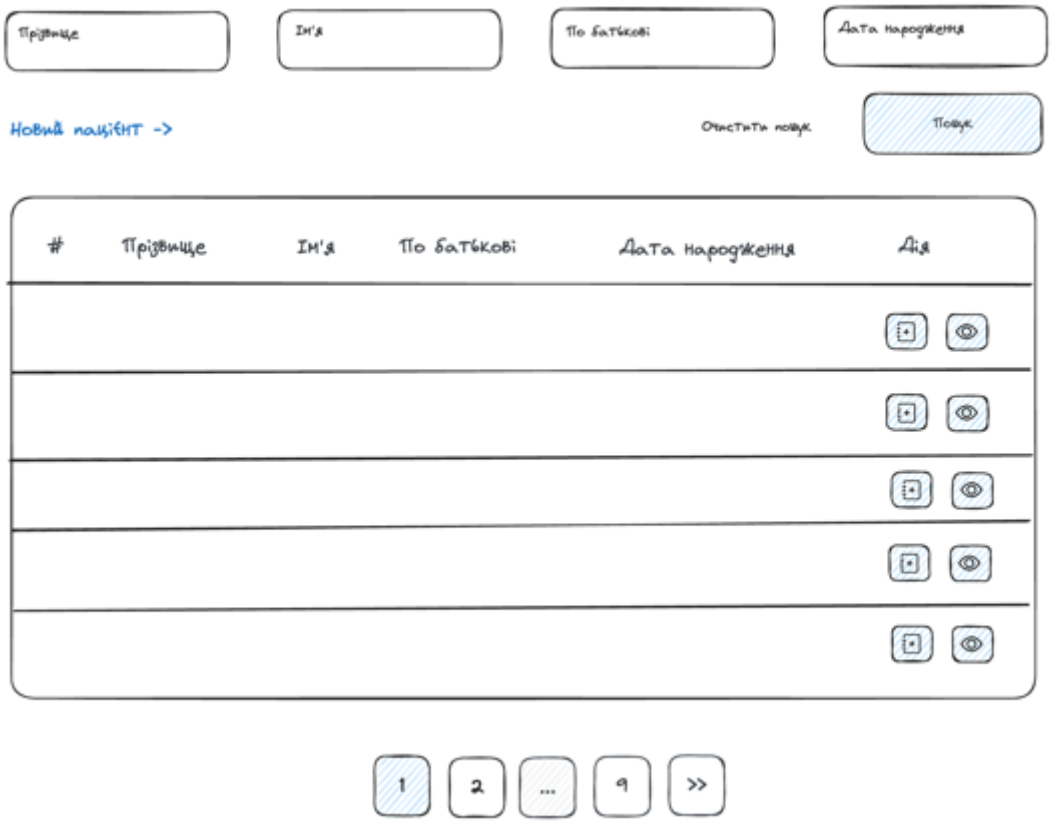


Рисунок 2.8 – Прототип сторінки “Пацієнти

Рисунок 2.9 – Прототип сторінки “Новий пацієнт”

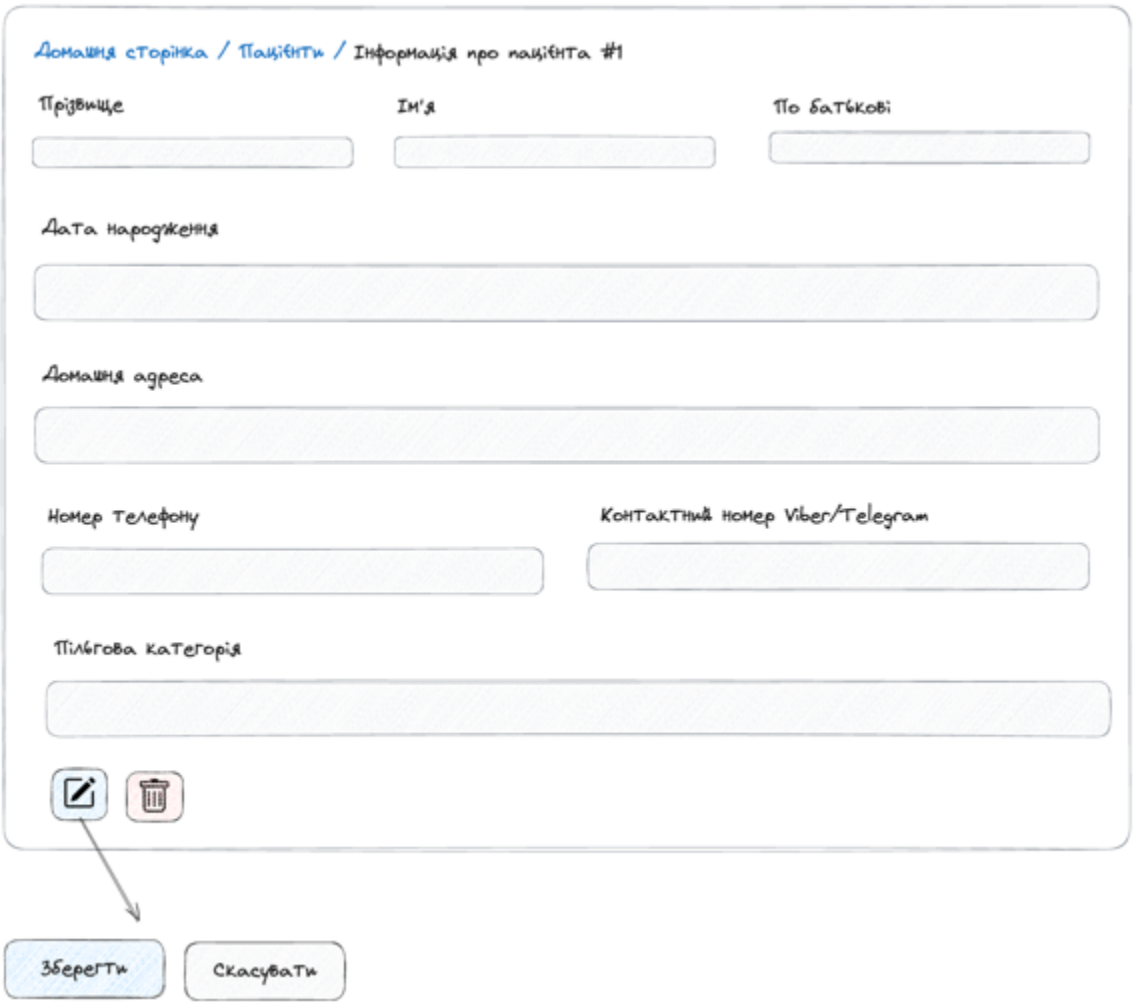


Рисунок 2.10 –Прототип сторінки "Інформація про певного пацієнта”

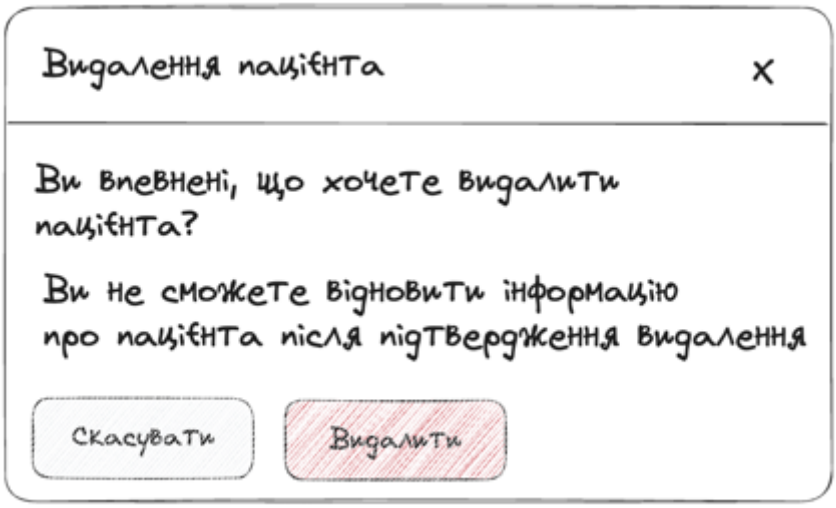


Рисунок 2.11 – Прототип вікна підтвердження видалення інформації про пацієнта

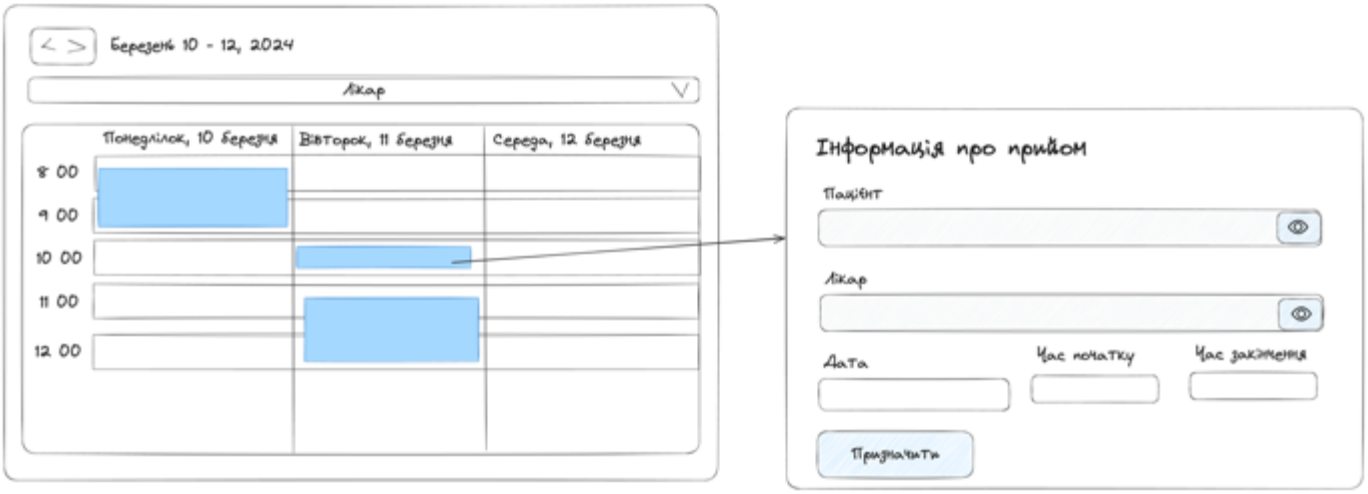


Рисунок 2.12 – Прототип сторінки “Новий прийом пацієнта до лікаря”

Інтерфейс застосунку грає ключову роль в покращенні роботи медичного персоналу, роблячи роботу більш зручною та ефективною. Інтуітивно зрозуміле управління та логічна організація функцій допомогають користувачам швидше орієнтуватися в системі без додаткового навчання.

2.3.2 Технічний проєкт вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

На основі створеного ескізного проєкту, було розроблено технічний проєкт програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed". Технічний проєкт містить побудову статичної та динамічної моделей програмного забезпечення, а також логічної моделі даних.

Діаграмою класів представлено статичну модель програмного забезпечення. Діаграма класів є структурною діаграмою мови UML, ця діаграма використовується для моделювання класів системи на етапі аналізу, подання їх взаємозв'язків, та опису того, що роблять ці класи та які функції вони надають [17].

Діаграма класів вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" представлена на рисунках 2.13 та 2.14.

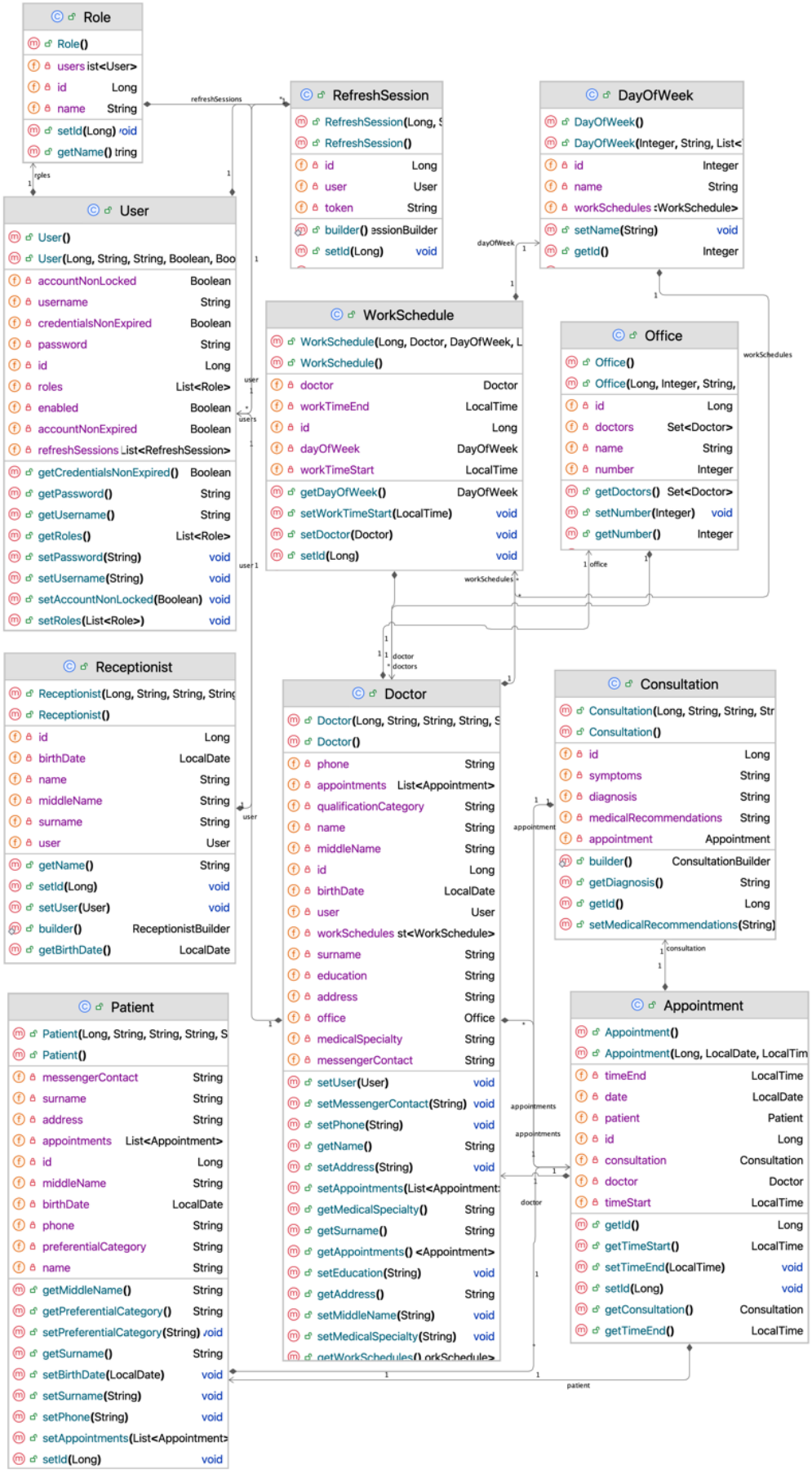


Рисунок 2.13 – Діаграма класів вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" (Частина 1)

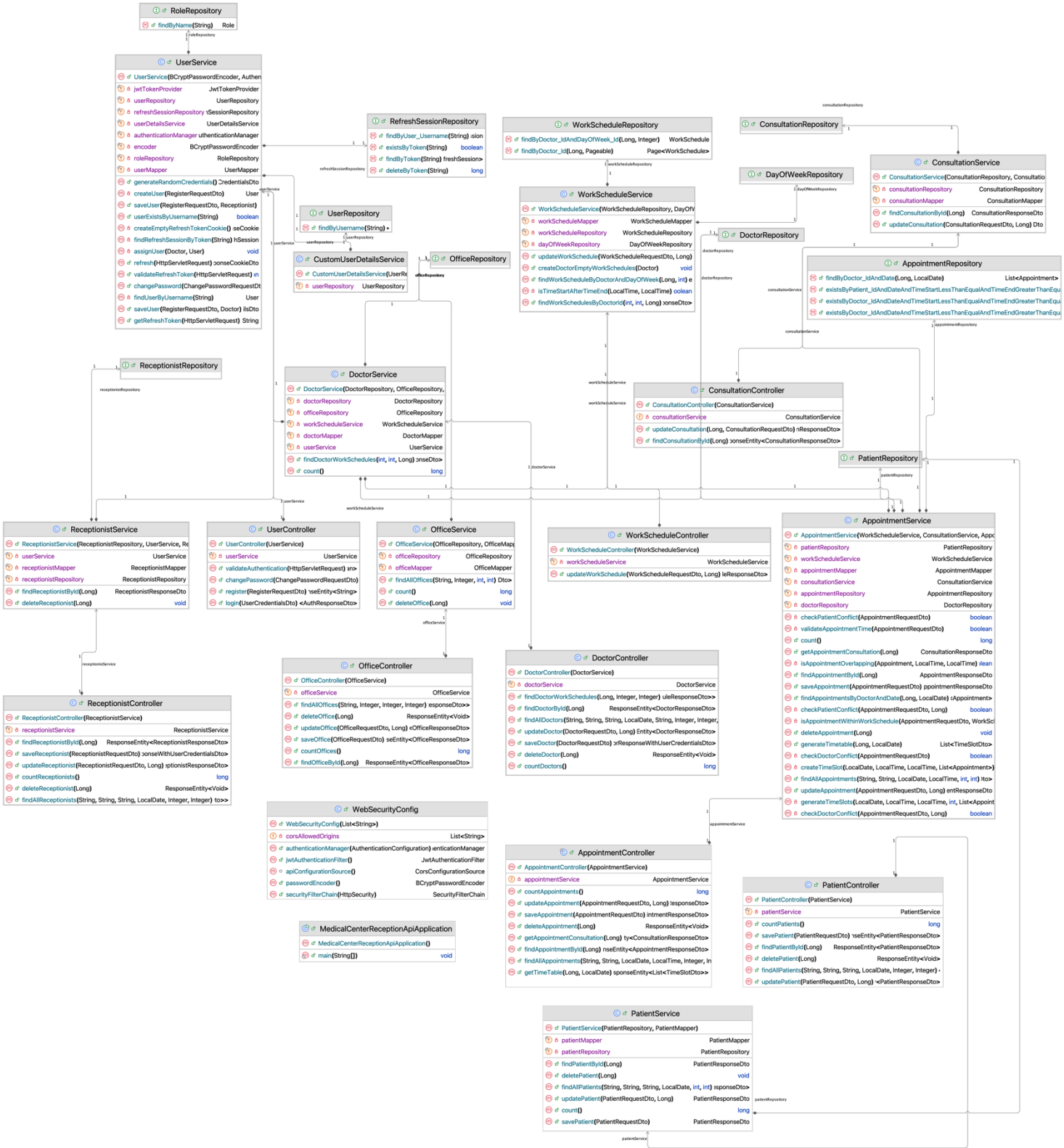


Рисунок 2.14 – Діаграма класів вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" (Частина 2)

Виявивши та проаналізувавши класи програмного забезпечення, для кожного з них було прописано детальні специфікації, які приведено нижче:

Специфікація класу “CustomUserDetailsService”

Назва: CustomUserDetailsService.

Відповідальність: Клас відповідає за завантаження інформації про певного користувача з бази даних для подальшої авторизації.

Атрибути:

– userRepository: UserRepository – інтерфейс Spring Data JPA для взаємодії з таблицею “Користувачі” в базі даних.

Методи:

– loadUserByUsername(): UserDetails – метод, що знаходить користувача за ім’ям в базі даних та повертає інформацію про нього у виді об’єкту UserDetails.

Специфікація класу “PatientController”

Назва: PatientController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про пацієнтів.

Атрибути:

– patientService: PatientService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із пацієнтами.

Методи:

– findAllPatients() – метод для отримання списку всіх пацієнтів;

– findPatientById() – метод для отримання звіту про певного пацієнта за його унікальним ідентифікатором;

– savePatient() – метод для додавання нового пацієнта;

– updatePatient() – метод для редагування інформації про певного пацієнта;

– deletePatient() – метод для видалення певного пацієнта.

Специфікація класу “ReceptionistController”

Назва: ReceptionistController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про реєстраторів.

Атрибути:

– receptionistService: ReceptionistService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із реєстраторами.

Методи:

– findAllReceptionists() – метод для отримання списку всіх реєстраторів;

– findReceptionistById() – метод для отримання звіту про певного реєстратора за його унікальним ідентифікатором;

– saveReceptionist() – метод для додавання нового реєстратора;

– updateReceptionist() – метод для редагування інформації про певного реєстратора;

– deleteReceptionist() – метод для видалення певного реєстратора.

Специфікація класу “DoctorController”

Назва: DoctorController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про лікарів.

Атрибути:

– doctorService: DoctorService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із лікарями.

Методи:

– findDoctorWorkSchedules() – метод для отримання робочих графіків лікаря;

– findAllDoctors() – метод для отримання списку всіх лікарів;

– findDoctorById() – метод для отримання звіту про певного лікаря за його унікальним ідентифікатором;

– saveDoctor() – метод для додавання нового лікаря;

– updateDoctor() – метод для редагування інформації про певного лікаря;

– deleteDoctor() – метод для видалення певного лікаря.

Специфікація класу “AppointmentController”

Назва: AppointmentController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про прийоми пацієнтів до лікарів.

Атрибути:

– appointmentService: AppointmentService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із прийомами.

Методи:

– findAllAppointments() – метод для отримання списку всіх прийомів;

– findAppointmentById() – метод для отримання звіту про певний прийом за його унікальним ідентифікатором;

– saveAppointment() – метод для додавання нового прийому;

– updateAppointment() – метод для редагування інформації про певний прийом;

– deleteAppointment() – метод для видалення певного прийому;

– getTimeTable() – метод для отримання розкладу роботи певного лікаря;

– getAppointmentConsultation() – метод для отримання консультації прийому.

Специфікація класу “ConsultationController”

Назва: ConsultationController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про консультації, пов’язані з прийомами пацієнтів.

Атрибути:

– consultationService: ConsultationService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із консультаціями.

Методи:

– findConsultationById() – метод для отримання звіту про певну консультацію за її унікальним ідентифікатором.

– updateConsultation() – метод для редагування інформації про певну консультацію.

Специфікація класу “WorkScheduleController”

Назва: WorkScheduleController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про графіки роботи лікарів.

Атрибути:

– workScheduleService: WorkScheduleService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із графіками роботи.

Методи:

– updateWorkSchedule() – метод для редагування інформації про певний графік роботи.

Специфікація класу “OfficeController”

Назва: OfficeController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про графіки роботи лікарів.

Атрибути:

– officeService: OfficeService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із кабінетами.

Методи:

– findAllOffices() – метод для отримання списку всіх кабінетів;

– findOfficeById() – метод для отримання звіту про певний кабінет за його унікальним ідентифікатором;

– saveOffice() – метод для додавання нового кабінету;

– updateOffice() – метод для редагування інформації про певний кабінет;

– deleteOffice() – метод для видалення певного кабінету.

Специфікація класу “UserController”

Назва: UserController.

Відповідальність: Клас відповідає за обробку запитів щодо інформації про користувачів.

Атрибути:

– userService: UserService – атрибут, що представляє собою об’єкт сервісу для виконання операцій пов’язаних із користувачами.

Методи:

– login() – метод для авторизації користувачів.

– refresh() – метод для оновлення JWT access-токену авторизації за допомогою refresh-токену

– getUserDetails() – метод для отримання інформації про обліковий запис користувача.

– validateAuthentication() – метод для валідації авторизаційної сесії користувача.

– logout() – метод для виходу з користувацького облікового запису та інвалідації авторизаційної сесії.

– changePassword() – метод для для зміни паролю від облікового запису користувача.

Через велику кількість класів вебзастосунку та обмежень щодо обсягу кваліфікаційної роботи було представлено специфікації тільки для деяких класів.

Розробивши діаграму класів та специфікацію класів до неї, можна переходити до наступного кроку розробки технічного проєкту програмного забезпечення, яким є побудова динамічної моделі програмного забезпечення для автоматизації реєстратури медичної клініки "CompliMed". Динамічна модель представлена діаграмою послідовності, яка є видом діаграми взаємодії мови UML. На цій діаграмі розглядається послідовність обміну повідомленнями між об'єктами в ході взаємодії, разом з їх життєвими циклами та структурами управління [18].

Діаграма послідовності прецеденту “Перегляд розкладу роботи певного лікаря” зображена на рисунку 2.15.

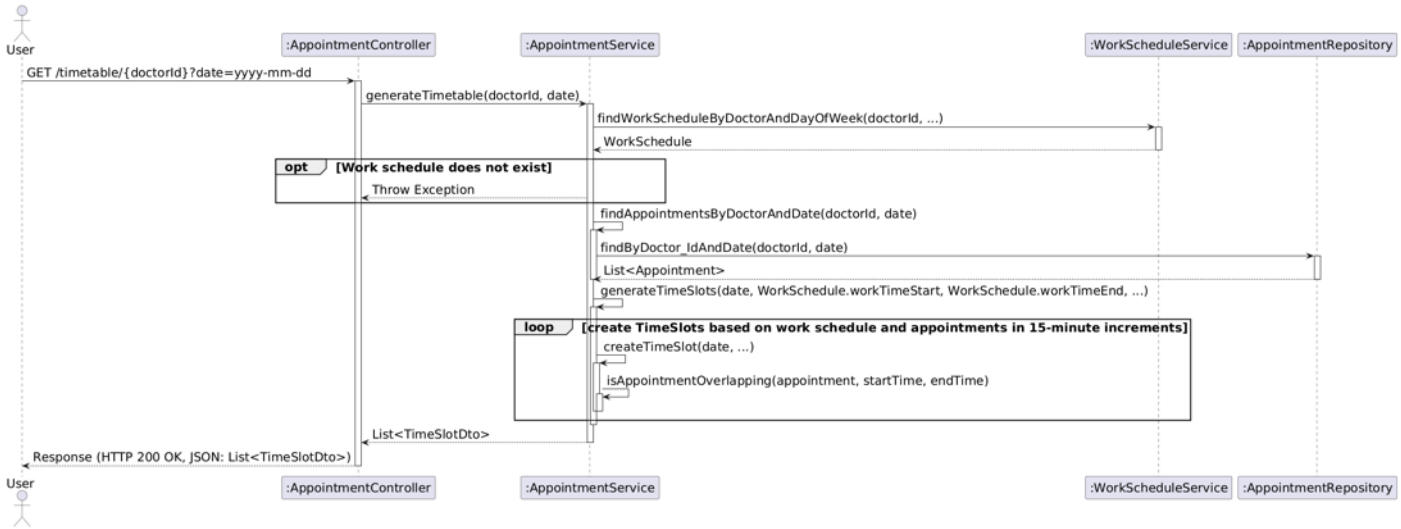


Рисунок 2.15 – Діаграма послідовності прецеденту “Перегляд розкладу роботи певного лікаря”

Діаграма послідовності прецеденту “ Призначити новий прийом пацієнта до лікаря ” зображена на рисунку 2.16.

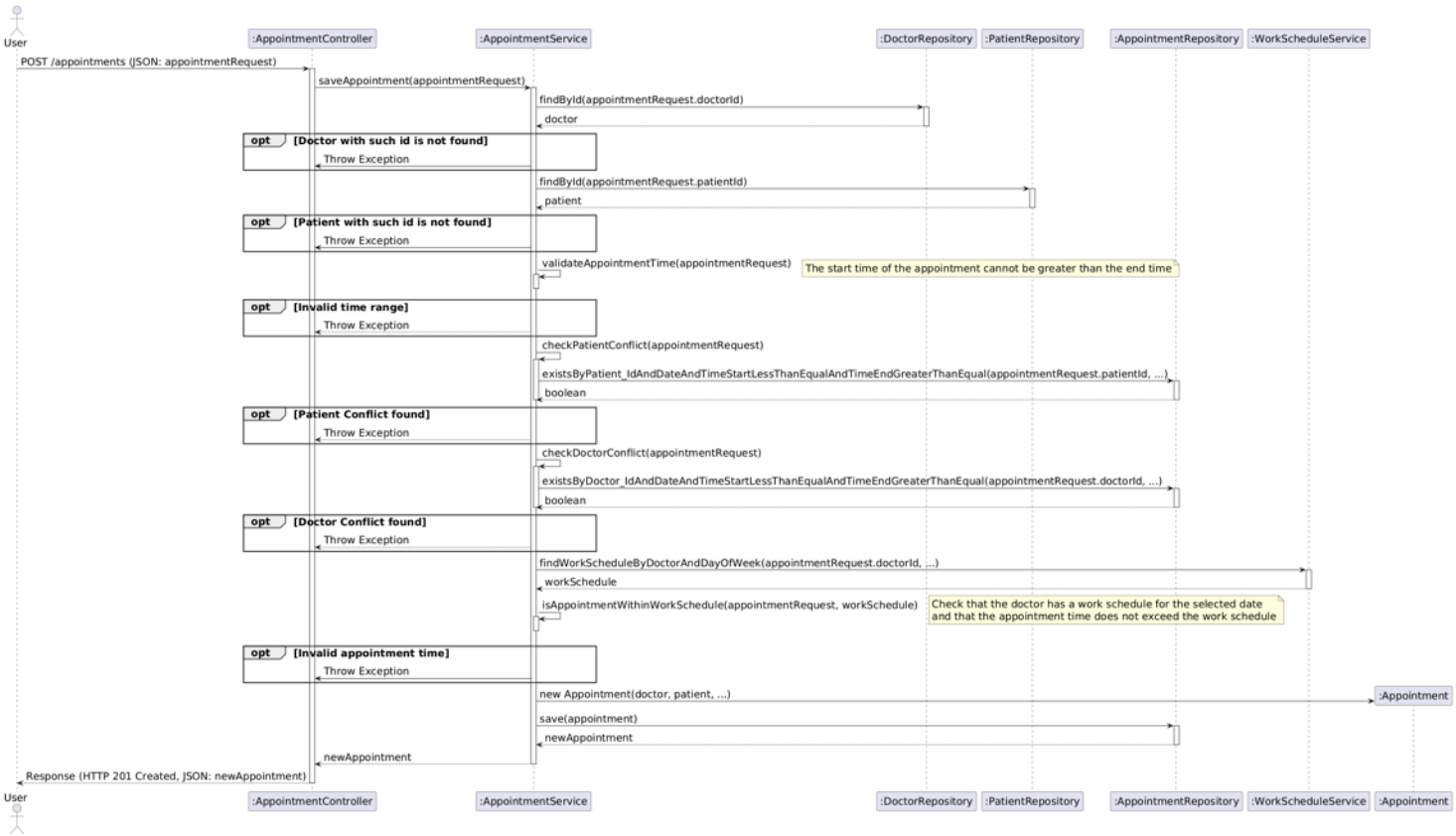


Рисунок 2.16 – Діаграма послідовності прецеденту “Призначити новий прийом пацієнта до лікаря”

На основі раніше створеної та представленої на рисунку 2.8 концептуальної моделі даних вебзастосунку для автоматизації роботи медичної клініки "CompliMed", необхідно розробити логічну модель даних для представлення предметної галузі, що розглядається. Логічна модель даних візуалізує інформацію для певної бізнес сфери, вона має більшу деталізацію ніж концептуальна модель, так як на ній включається визначення первинних ключів, зовнішніх ключів, кардинальність зв'язків, типи даних, тощо. Також, на логічній моделі дані повинні бути нормалізовані, але структура відображається незалежно від використовуваної системи керування базами даних [19].

Логічну модель даних вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" приведено на рисунку 2.17.

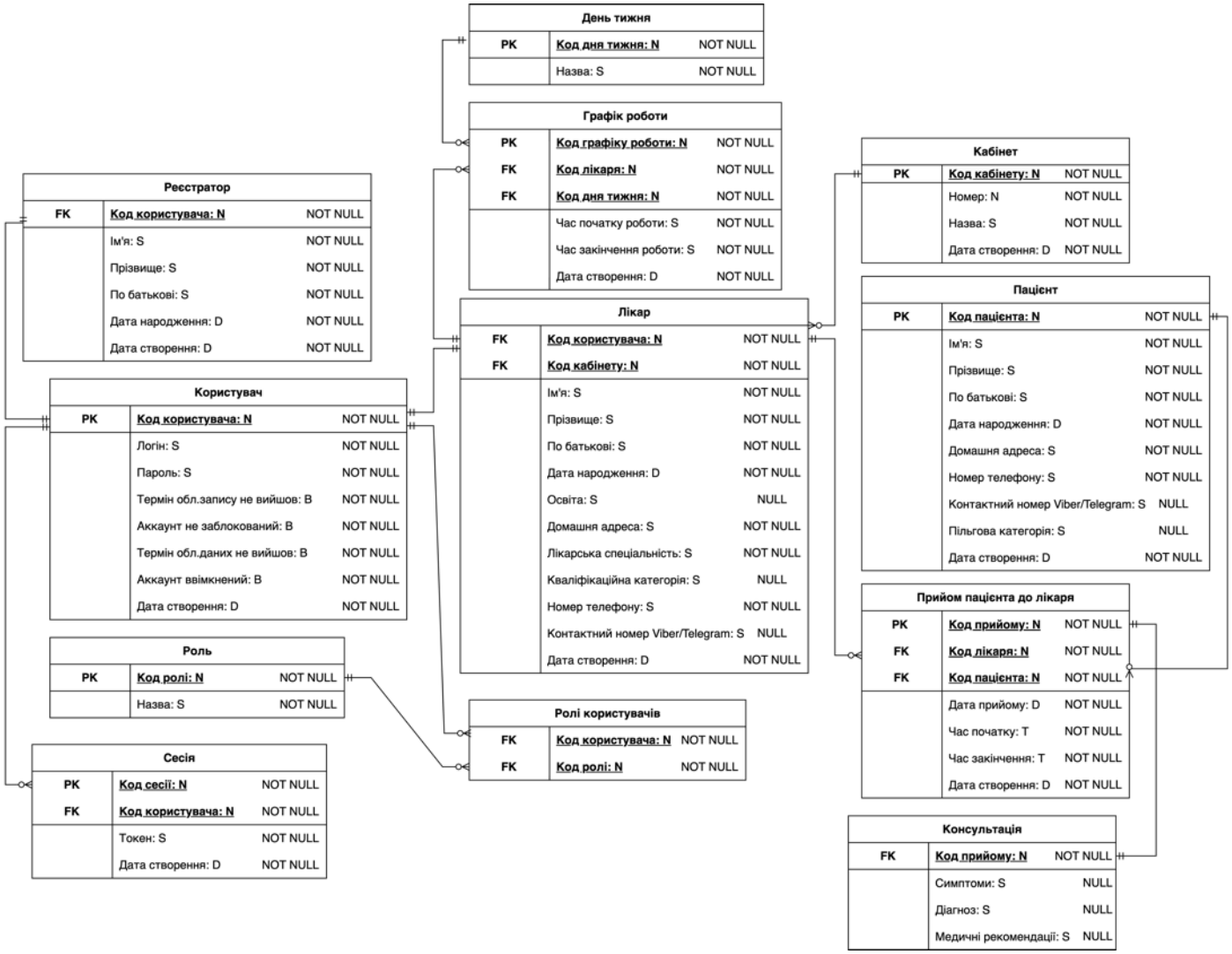


Рисунок 2.17 – Логічна модель бази даних вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Після розробки технічного проєкту програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed" можна приступати до наступного етапу проєктування, а саме до розробки робочого проєкту вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

2.3.3 Робочий проєкт вебзастосунку для автоматизації роботи реєстратури медичної клініки "Complimed"

У підрозділі, присвяченому робочому проєкту програмного забезпечення для автоматизації роботи реєстратури медичної клініки "CompliMed", було обгрунтовано вибір засобів розробки, реалізовано відповідне програмне забезпечення, а також проведено його тестування та випробовування.

Обгрунтування вибору засобів розробки

Для успішної реалізації проєкту, що вимагає ефективної системи управління базами даних (СУБД), важливо ретельно вибрати відповідну платформу. Одним із перспективних варіантів є MariaDB, розглянемо причини, через які цей вибір виправданий порівняно з іншими поширеними СУБД, такими як MySQL і PostgreSQL.

MariaDB заснована на відкритому вихідному коді, що дає користувачам повний контроль над кодом і можливість внесення змін відповідно до їхніх потреб. На відміну від комерційних аналогів, таких як Oracle Database, MariaDB надає безоплатну, відкриту ліцензію, що робить її більш доступною для більшої кількості користувачів [20].

MariaDB застосовує низку оптимізацій для підвищення швидкості запитів і обробки даних. Це включає використання індексів, оптимізовані алгоритми сортування і паралельну обробку запитів. У порівнянні з PostgreSQL, MariaDB може продемонструвати вищу продуктивність у деяких випадках, особливо під час роботи з великим обсягом транзакцій [21].

MariaDB підтримується активною спільнотою розробників, що забезпечує швидке виявлення і виправлення помилок, а також регулярні оновлення з новими функціями. Порівняно з MySQL, який, після придбання Sun Microsystems компанією Oracle, викликав занепокоєння в спільноті через питання незалежності розробки, MariaDB виглядає стабільнішим і відкритішим варіантом.

Отже, MariaDB надає великий набір функцій, високу продуктивність і відкритий вихідний код, роблячи її привабливим варіантом для використання в

розробці бази даних для вебзастосунку для автоматизації роботи реєстратури медичної клініки “CompliMed”.

Для кешування даних було обрано сховище даних в оперативній пам’яті Redis. Це сховище використовують як кеш, векторну базу даних, базу даних документів, потоковий движок і брокера повідомлень. Redis підтримує різні типи даних, такі як хеші, списки, рядки, множини, відсортовані множини та JSON [22].

Мовою програмування була обрана Java 17 версії. Java – строго типізована об'єктно-орієнтована мова програмування загального призначення, розроблена компанією Sun Microsystems. Програми Java зазвичай перетворюються у спеціальний байт-код, тому вони можуть працювати на будь-якій комп'ютерній архітектурі, для якої існує реалізація віртуальної Java-машини [23].

Для розробки проєкту мною було обрано програмне забезпечення JetBrains IntelliJ IDEA – інтегроване середовище розробки (IDE) для мов JVM, призначене для збільшення продуктивності розробників. Воно виконує рутинні та повторювані завдання, забезпечуючи розумне автодоповнення коду, статичний аналіз коду та рефакторинг. IntelliJ IDEA має інтеграцією з Gradle і Maven, що дозволяє автоматизувати процес збірки, пакування, запуск тестів, розгортання та інші дії [24].

Для збірки проєкту використовуватиметься Apache Maven. Це фреймворк для автоматизації збирання проєктів на основі опису їхньої структури у файлах мовою POM (англ. Project Object Model). Maven забезпечує декларативну, а не імперативну збірку проєкту. У файлах опису проєкту міститься його специфікація, а не окремі команди виконання. Усі завдання з обробки файлів, описані в специфікації, Maven виконує за допомогою їх опрацювання послідовністю вбудованих і зовнішніх плагінів [25].

Для створення архітектурної моделі вебзастосунку мною було обрано Spring MVC – це вебфреймворк на основі Java, який використовується для створення вебзастосунків. Він забезпечує архітектуру модель-вид-контролер (MVC), яка розділяє логіку застосунку на три взаємопов'язані компоненти: бізнес-логіку, користувацький інтерфейс та контроллер посередник між бізнес логікою та інтерфейсом [26]. Spring MVC фреймворк надає ряд функцій, таких як обробка запитів, зв'язування даних, валідація та інтернаціоналізація. RESTful API на Spring

MVC для цього проєкту забезпечить взаємодію між клієнтською та серверною частинами застосунку. Він дозволить клієнтам надсилати HTTP-запити для отримання, створення, оновлення та видалення даних за допомогою стандартних HTTP-методів, таких як GET, POST, PUT та DELETE та отримувати відповіді на запити у форматі JSON.

Для створення гіпертекстової розмітки вебсторінок я обрав мову розмітки гіпертексту HTML (HyperText Markup Language). HTML визначає зміст і структуру вебконтенту [27].

Для задання стилів та вигляду вебсторінок, написаних мовою HTML я обрав “Каскадні таблиці стилів” або CSS – у той час як HTML використовується для визначення структури та семантики контенту, CSS використовується для його стилізації та компонування. За допомогою CSS можна змінювати шрифт, колір, і розмір контенту, розбивати його на кілька колонок, додавати анімації та інші декоративні елементи [28].

Для створення динамічних вебсторінок я обрав React – це безкоштовна бібліотека з відкритим вихідним кодом для вебінтерфейсів та нативних інтерфейсів користувача. За його допомогою можна створювати користувацькі інтерфейси з окремих частин, які називаються компонентами, написаними на JavaScript. Однією з головних переваг React є те, що він уникає непотрібного повторного рендерингу незмінних елементів вебсторінки перемальовуючи лише ті частини сторінки, які змінилися [29].

Разом з React буде використано фреймворк Next.js. Цей інструмент дозволяє створювати full-stack вебзастосунки, розширюючи можливості React та надаючи можливості рендерингу на стороні серверу, статичну генерацію та JavaScript-інструментарій на основі Rust для збірки проєкту [30].

Для створення сучасного вебінтерфейсу я обрав React Bootstrap – це безкоштовний набір інструментів з відкритим кодом, призначений для створення інтерфейсів вебзастосунків, який містить шаблони React компонентів для типографіки, форм, кнопок, навігації та інших компонентів інтерфейсу. [31].

Розробка фізичної моделі бази даних

Фізична модель даних описує набір даних специфічно до певної бази даних. На цій моделі представляються реляційні об'єкти, такі як таблиці, стовпці, первинні ключі, зовнішні ключі, їхні зв'язки та типи даних. Так як на цій моделі відображаються елементи реальної бази даних, то необхідно дотримуватись відповідних до обраної бази даних обмежень [32]. Фізична модель бази даних вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" представлена на рисунку 2.18.

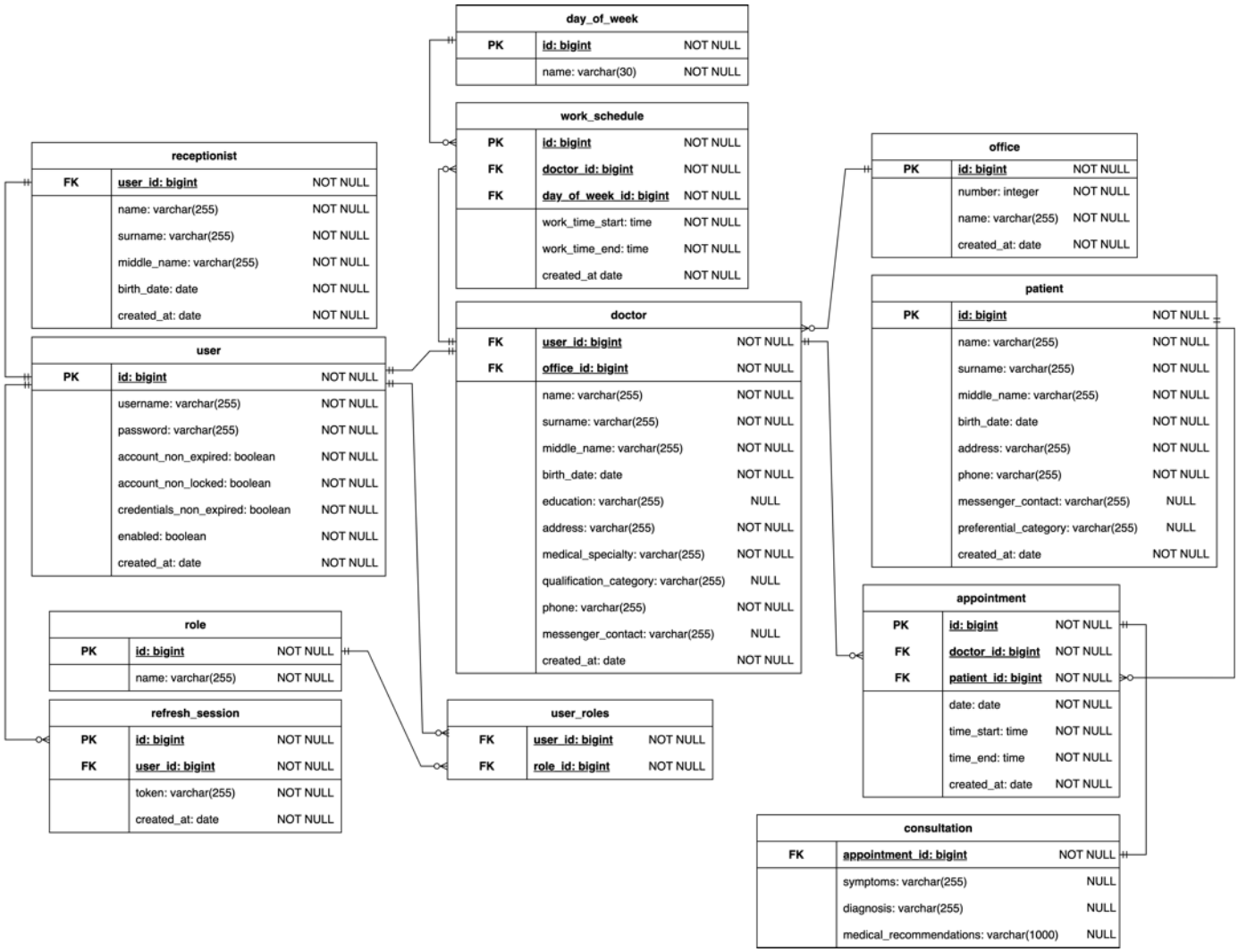


Рисунок 2.18 – Фізична модель бази даних вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Відповідність назв таблиць та атрибутів логічної та фізичної моделей представлені у таблиці 2.11.

Таблиця 2.11 – Відповідність назв таблиць та атрибутів логічної та фізичної моделей

Назва таблиці в логічний моделі	Назва таблиці в фізичній моделі	Атрибути логічної моделі	Атрибути фізичної моделі
1	2	3	4
Кабінет	office	Код кабінету	id
		Назва	name
		Номер	number
		Дата створення	created_at
Лікар	doctor	Код користувача	user_id
		Ім’я	name
		Прізвище	surname
		По батькові	middle_name
		Дата народження	birth_date
		Освіта	education
		Домашня адреса	address
		Лікарська спеціальність	medical_speciality
		Лікарська категорія	qualification_category
		Номер телефону	phone
		Контактний номер Viber/Telegram	messenger_contact
		Код кабінету	office_id
		Дата створення	created_at
Пацієнт	patient	Код пацієнта	id
		Ім’я	name
		Прізвище	surname
		По батькові	middle_name
		Дата народження	birth_date
		Домашня адреса	address
		Номер телефону	phone
		Контактний номер Viber/Telegram	messenger_contact
		Пільгова категорія	preferential_category
		Дата створення	created_at
Реєстратор	receptionist	Код користувача	user_id
		Ім’я	name
		Прізвище	surname
		По батькові	middle_name
		Дата народження	birth_date
		Дата створення	created_at

Продовження табл 2.11

1	2	3	4
Прийом пацієнта до лікаря	appointment	Код прийому	id
		Код лікаря	doctor_id
		Код пацієнта	patient_id
		Дата прийому	date
		Час початку	time_start
		Час закінчення	time_end
		Дата створення	created_at
Консультація	consultation	Код прийому	appointment_id
		Симптоми	symptoms
		Діагноз	diagnosis
		Медичні рекомендації	medical_recommendations
День тижня	day_of_week	Код дня тижня	id
		Назва	name
Графік роботи	work_schedule	Код графіку роботи	id
		Код лікаря	doctor_id
		Код дня тижня	day_of_week_id
		Час початку роботи	work_time_start
		Час закінчення роботи	work_time_end
		Дата створення	created_at
Користувач	user	Код користувача	id
		Логін	username
		Пароль	password
		Термін обл.запису не вийшов	account_non_expired
		Аккаунт не заблокований	account_non_locked
		Термін обл.даних не вийшов	credentials_non_expired
		Аккаунт ввімкнений	enabled
		Дата створення	created_at
Роль	role	Код ролі	id
		Назва	name
Ролі користувачів	user_roles	Код користувача	user_id
		Код ролі	role_id
Сесія	refresh_session	Код сесії	id
		Код користувача	user_id
		Токен	token
		Дата створення	created_at

Нижче надано фрагмент коду реалізації функції призначення нового прийому пацієнта до лікаря.

Фрагмент коду класу AppointmentService:

```
@PreAuthorize(
    "hasAnyRole('ADMIN', 'RECEPTIONIST') or"
        + "#appointmentRequestDto.doctorId ==
authentication.principal.id")
@CacheEvict(
    value = {"appointments", "timetable"},
    allEntries = true)
public AppointmentResponseDto saveAppointment(AppointmentRequestDto
appointmentRequestDto) {
  Doctor doctor =
      doctorRepository
          .findById(appointmentRequestDto.getDoctorId())
          .orElseThrow(
              () -> new ResourceNotFoundException("A doctor with
that id doesn't exist"));
  Patient patient =
      patientRepository
          .findById(appointmentRequestDto.getPatientId())
          .orElseThrow(
              () -> new ResourceNotFoundException("A patient with
that id doesn't exist"));
  if (!validateAppointmentTime(appointmentRequestDto)) {
    throw new IllegalArgumentException();
  }
  if (checkPatientConflict(appointmentRequestDto)) {
    throw new AppointmentPatientConflictException();
  }
  if (checkDoctorConflict(appointmentRequestDto)) {
    throw new AppointmentDoctorConflictException();
  }
  java.time.DayOfWeek dayOfWeek =
appointmentRequestDto.getDate().getDayOfWeek();
  WorkSchedule workSchedule =
      workScheduleService.findWorkScheduleByDoctorAndDayOfWeek(
          doctor.getId(), dayOfWeek.getValue());
  if (!isAppointmentWithinWorkSchedule(appointmentRequestDto,
workSchedule)) {
    throw new InvalidAppointmentTimeException();
  }
  Appointment newAppointment =

appointmentMapper.appointmentRequestDtoToAppointment(appointmentRequ
estDto);
  newAppointment.setDoctor(doctor);
  newAppointment.setPatient(patient);
```

```
  newAppointment = appointmentRepository.save(newAppointment);
  return
appointmentMapper.appointmentToAppointmentResponseDto(newAppointment
);
}
```

Анотація @PreAuthorize у цьому прикладі перевіряє, що новий прийом створюється адміністратором, реєстратором або лікарем. У випадку якщо прийом створюється лікарем, перевіряється, що лікар призначає прийом саме до себе, так як певний лікар не має права призначати прийоми другим лікарям.

Метод saveAppointment, отримує об'єкт appointmentRequestDto, який відображає інформацію про новий прийом. Далі виконується пошук відповідного лікаря та пацієнта в базі даних за переданими ідентифікаторами. Якщо пацієнта або лікаря не знайдено, виникає оброблена помилка. За допомогою допоміжних методів validateAppointmentTime, checkPatientConflict, checkDoctorConflict перевіряється правильність переданого часу прийому та відсутність конфліктів при призначенні прийому, таких як, наприклад, у разі якщо лікарю намагаються призначити прийом на час, який вже зайнятий іншим пацієнтом у цього лікаря. Зрештою, отримується графік роботи лікаря з бази даних, та перевіряється що час нового прийому знаходиться в межах графіку роботи лікаря на обрану дату. У разі успішного проходження всіх перевірок, створюється новий об'єкт Appointment та заноситься в базу даних за допомогою методу save об'єкту appointmentRepository.

Повний текст вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" представлено у Додатку Г.

Тестування вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Для тестування функціоналу вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" був використаний метод розбиття на класи еквівалентності, що належить до методів функціонального тестування. Цей метод передбачає розподіл множини вхдних та вихідних даних на еквівалентні класи. Розбиття виконується таким чином, що кожен тест, що входить до певного класу, вважається еквівалентним будь-якому іншому тесту цього класу, тобто програма однаково реагує на всі тести одного класу. Цей метод є підходом

тестування за стратегією чорної скриньки, де тестувальник не знає внутрішньої реалізації програми [33].

В ході першого тестування було протестовано варіант використання "Перегляд розкладу роботи певного лікаря":

Ідентифікатор: TEST_RLA20_1

Предмети тестування:

– Форма вибору лікаря та дати для перегляду розкладу.

– Відображення розкладу роботи лікаря.

Вхідні дані:

– номер лікаря* (вибір наявних лікарів із запропонованого списку);

– дата*;

Результати:

Користувачу надається інформація про розклад роботи обраного лікаря на певну дату.

Класи еквівалентності для функції “Перегляд розкладу роботи певного лікаря” представлені в таблиці 2.12.

Таблиця 2.12 – Класи еквівалентності функції “Перегляд розкладу роботи певного лікаря”

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
1	2	3
Номер лікаря*	Вибір лікаря із запропонованого списку лікарів 1. Перший лікар зі списку 2. Проміжний лікар зі списку. 3. Останній лікар зі списку	
Дата*	4. Дата у форматі дд[01-31].мм[01-12].рррр[1583-9999], де значення для дд (день), мм (місяць), рррр (рік) відповідають стандарту ISO 8601 5. Дата, на яку лікар має графік роботи	6. Рік в даті не входить у діапазон 1583-9999. 7. Місяць в даті не входить у діапазон 01-12. 8. День в даті не входить у діапазон 01-31.

Продовження табл. 2.12

1	2	3
Дата*		9. Кількість символів в компоненті дати не відповідає стандарту (наприклад, 9.8.2025, де одиничні числа не мають ведучих нулів). 10. День або місяць не відповідають дійсним датам (наприклад, 31.04.2025). 11. Дата, на яку лікар не має графіку роботи

Тести для правильних класів еквівалентності функції “Перегляд розкладу роботи певного лікаря” представлені у таблиці 2.13.

Таблиця 2.13 – Тести для правильних класів еквівалентності функції “Перегляд розкладу роботи певного лікаря”

№ теста	Вхідні умови	Результат
1	3	4
1	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 05.03.2025 Дата на яку лікар має графік роботи	Розклад роботи лікаря з номером 1 на 5 березня 2025 року, враховуючи графік роботи лікаря та наявні прийоми лікаря на обрану дату з кроком часових проміжків 15 хвилин. Де не зайняті прийомами часові проміжки позначаються вільними, а в зайнятих вказано номер прийому який зайняв проміжок.
2	Лікар: 5 – Броваренко Андрій Олексійович Дата: 12.03.2025 Дата на яку лікар має графік роботи	Розклад роботи лікаря з номером 5 на 12 березня 2025 року, враховуючи графік роботи лікаря та наявні прийоми лікаря на обрану дату з кроком часових проміжків 15 хвилин.
3	Лікар: 10 – Захарчук Ярослава Тарасівна Дата: 03.04.2025 Дата на яку лікар має графік роботи	Розклад роботи лікаря з номером 10 на 3 квітня 2025 року, враховуючи графік роботи лікаря та наявні прийоми лікаря на обрану дату з кроком часових проміжків 15 хвилин.

Тести для неправильних класів еквівалентності функції “Перегляд розкладу роботи певного лікаря” представлені у таблиці 2.14.

Таблиця 2.14 – Тести для неправильних класів еквівалентності функції “Перегляд розкладу роботи певного лікаря”

№ теста	Вхідні умови	Результат
1	2	3
1	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 12.02.12345	Помилка! Введіть дату у форматі дд.мм.рррр.
2	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 12.13.2025	Помилка! Введіть дату у форматі дд.мм.рррр.
3	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 33.03.2025	Помилка! Введіть дату у форматі дд.мм.рррр.
4	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 31.04.2025	Помилка! Введіть дату у форматі дд.мм.рррр.
5	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 3.04.2025	Помилка! Введіть дату у форматі дд.мм.рррр.
6	Лікар: 1 – Кравченко Єлизавета Борисівна Дата: 08.03.2025 Дата на яку лікар не має графіку роботи	Повідомлення про відсутність графіку роботи лікаря на обрану дату, розклад роботи не надається.
7	Лікар: 5 – Кравченко Єлизавета Борисівна Дата: 09.03.2025 Дата на яку лікар не має графіку роботи	Повідомлення про відсутність графіку роботи лікаря на обрану дату, розклад роботи не надається.

Тестування варіанту використання "Призначити новий прийом"

Ідентифікатор: TEST_RLA8_1.

Предмети тестування:

– Форма призначення нового прийому;

– Валідація введених даних;

– Обробка конфліктів та обмежень при призначенні прийому.

Вхідні дані:

– номер лікаря* (вибір наявних лікарів із запропонованого списку);

– номер пацієнта* (вибір наявних пацієнтів із запропонованого списку);

– дата (дд.мм.рррр)*;

– час початку (гг:хх)*;

– час закінчення (гг:хх)*;.

Результати:

Інформація про новий прийом пацієнта до лікаря збережено у системі.

Класи еквівалентності функції “Призначити новий прийом” представлені в таблиці 2.15.

Таблиця 2.15 – Класи еквівалентності функції “Призначити новий прийом”

Вхідні дані	Правильні класи еквівалентності	Неправильні класи еквівалентності
1	2	3
Номер лікаря*	Вибір лікаря із запропонованого списку лікарів 1. Перший лікар зі списку 2. Проміжний лікар зі списку. 3. Останній лікар зі списку	
Номер пацієнта*	Вибір пацієнта із запропонованого списку пацієнтів 4. Перший пацієнт зі списку 5. Проміжний пацієнт зі списку. 6. Останній пацієнт зі списку	
Дата та час прийому (початок-закінчення)	7. Дата у форматі дд[01-31].мм[01-12].рррр[1583-9999], де значення для дд (день), мм (місяць), рррр (рік) відповідають стандарту ISO 8601. 8. Дата, на яку лікар має графік роботи. 9. Час початку та закінчення прийому у форматі гг[00-23]:хх[00-59], де гг (години) та хх (хвилини) відповідають стандарту ISO 8601.	13. Рік в даті не входить у діапазон 1583-9999. 14. Місяць в даті не входить у діапазон 01-12. 15. День в даті не входить у діапазон 01-31. 16. Кількість символів в компоненті дати не відповідає стандарту (наприклад, 9.8.2025, де одиничні числа не мають ведучих нулів).

Продовження табл. 2.15

1	2	3
	10. Час закінчення має бути пізніше ніж час початку прийому. 11. Дата та час прийому не конфліктують з вже існуючими прийомами у лікаря та пацієнта. 12. Дата та час прийому не виходять за межі графіку роботи лікаря.	17. День або місяць не відповідають дійсним датам (наприклад, 31.04.2025). 18. Дата на яку лікар не має графіку роботи. 19. Години часу прийому виходять за межі діапазону 00-23. 20. Хвилини часу прийому виходять за межі діапазону 00-59. 21. Години або хвилини мають пропущені ведучі нулі. 22. Час закінчення раніше ніж час початку прийому. 23. Пацієнт або лікар вже мають інший прийом на обрану дату та час. 24. Дата та час прийому виходять за межі графіку роботи лікаря.

Тести для правильних класів еквівалентності функції “Призначити новий прийом” представлені у таблиці 2.16.

Таблиця 2.16 – Тести для правильних класів еквівалентності функції “Призначити новий прийом”

№ теста	Вхідні умови	Результат
1	2	3
1	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 05.03.2025, Час: 13:00 – 13:35 Дата, на яку лікар має графік роботи. Час закінчення пізніше ніж час початку прийому. Дата та час прийому не конфліктують з вже існуючими прийомами у лікаря та пацієнта. Дата та час прийому не виходять за межі графіку роботи лікаря.	Інформація про новий прийом пацієнта до лікаря збережена у системі

Продовження табл. 2.16

1	2	3
2	Лікар: 5 – Бровареико Андрій Олексійович Пацієнт: 1 – Луценко Михайло Володимирович Дата: 06.03.2025, Час: 9:00 – 11:00 Дата, на яку лікар має графік роботи. Час закінчення пізніше ніж час початку прийому. Дата та час прийому не конфліктують з вже існуючими прийомами у лікаря та пацієнта. Дата та час прийому не виходять за межі графіку роботи лікаря.	Інформація про новий прийом пацієнта до лікаря збережена у системі
3	Лікар: 10 – Захарчук Ярослава Тарасівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 03.04.2025, Час: 14:00 - 14:20 Дата, на яку лікар має графік роботи. Час закінчення пізніше ніж час початку прийому. Дата та час прийому не конфліктують з вже існуючими прийомами у лікаря та пацієнта. Дата та час прийому не виходять за межі графіку роботи лікаря.	Інформація про новий прийом пацієнта до лікаря збережена у системі
4	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 5 – Боднаренко Вадим Васильович Дата: 14.03.2025, Час: 13:00 – 13:35 Дата, на яку лікар має графік роботи. Час закінчення пізніше ніж час початку прийому. Дата та час прийому не конфліктують з вже існуючими прийомами у лікаря та пацієнта. Дата та час прийому не виходять за межі графіку роботи лікаря.	Інформація про новий прийом пацієнта до лікаря збережена у системі

Тести для неправильних класів еквівалентності функції “Призначити новий прийом” представлені у таблиці 2.17.

Таблиця 2.17 – Тести для неправильних класів еквівалентності функції “Призначити новий прийом”

№ теста	Вхідні умови	Результат
1	3	4
1	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 05.02.2025, Час: 20:00 – 20:30 Дата на яку лікар має графік роботи. Час прийому виходить за межі графіку роботи лікаря на обрану дату.	Помилка! Час прийому виходить за межі графіку роботи лікаря на обрану дату.
2	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 16.02.2025, Час: 13:00 – 13:35 Дата, на яку лікар не має графіку роботи	Помилка! Лікар не має графіку роботи на обрану дату.
3	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 05.02.2025, Час: 11:00 – 10:20 Час початку пізніше ніж час закінчення.	Помилка! Час закінчення повинен бути пізніше часу початку прийому.
4	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 12.02.2025, Час: 13:00 – 13:40 Дата та час на який у лікаря вже є інший прийом.	Помилка! Лікар вже має прийом на обраний час.
5	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 13.02.2025, Час: 11:40 – 12:20 Дата та час на який у пацієнта вже є інший прийом.	Помилка! Пацієнт вже має прийом на обраний час.
6	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 13.02.12345, Час: 9:00 – 9:45 Дата неправильного формату	Помилка! Дата повинна бути формату дд.мм.рррр.
7	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 12.13.2025, Час: 9:00 – 9:45 Дата неправильного формату	Помилка! Дата повинна бути формату дд.мм.рррр.
8	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 2.04.2025, Час: 9:00 – 9:45 Дата неправильного формату	Помилка! Дата повинна бути формату дд.мм.рррр.

Продовження табл. 2.17

1	2	3
9	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 33.03.2025, Час: 9:00 – 9:45 Дата неправильного формату	Помилка! Дата повинна бути формату дд.мм.рррр.
10	Лікар: 1 - Кравченко Єлизавета Борисівна Пацієнт: 1 - Луценко Михайло Володимирович Дата: 31.04.2025, Час: 9:00 – 9:45 Дата неправильного формату	Помилка! Дата повинна бути формату дд.мм.рррр.
11	Лікар: 1 – Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 13.02.2025, Час: 8:30 – 13:7 Час неправильного формату	Помилка! Час прийому повинен бути формату гг:хх.
12	Лікар: 1 - Кравченко Єлизавета Борисівна Пацієнт: 1 – Луценко Михайло Володимирович Дата: 13.02.2025, Час: 14:65 – 15:30 Час неправильного формату	Помилка! Час прийому повинен бути формату гг:хх.

По причині великого обсягу функціоналу вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" та складності модулів, в роботі представлено результати невеликої кількості тестів. Вебзастосунок був протестований повністю. За результатами тестування можна зробити висновок, що вебзастосунок працює правильно, валідує вхідні дані перевіряючи на відповідність типам, форматам, обмеженням та недопускає введення інших елементів, окрім стандартно запропонованих зі списків, де це передбачено.

Випробування вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed"

Випробування вебзастосунку є одним з завершальних етапів розробки та визначальним етапом життєвого циклу, де закріплюються та перевіряються досягнуті показники якості ПЗ. Метою такого випробування є виявлення реально досягнутих характеристик вебзастосунку та визначення його відповідності вимогам.

Випробування вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" було виконано згідно з програмою та методикою випробувань, яка представлена у додатку Д.

Випробування функції "Введення інформації про нового лікаря".

Перейшовши на сторінку "Лікарі" з навігаційного меню та натиснувши на кнопку "Новий лікар" була відкрита сторінка внесення інформації про нового лікаря. Далі були заповнені поля інформації про лікаря (див. рис 2.19).

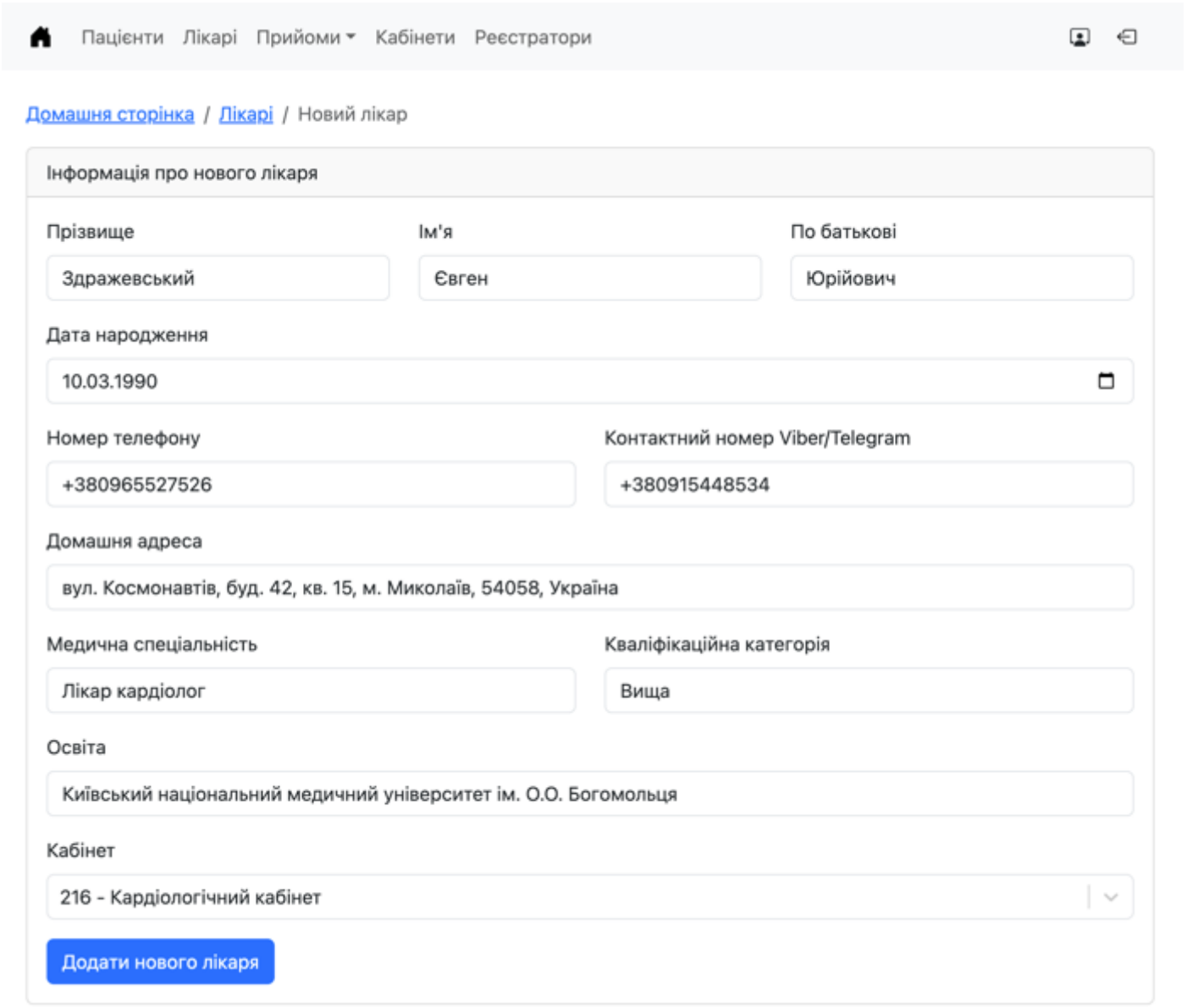


Рисунок 2.19 – Сторінка "Новий Лікар"

Після натискання кнопки "Додати нового лікаря" для підтвердження внесення інформації про нового лікаря, застосунок провів перевірку введених даних на правильність та після успішної перевірки новий лікар був створений.

Результат даної операції представлено на рисунках 2.20 та 2.21.

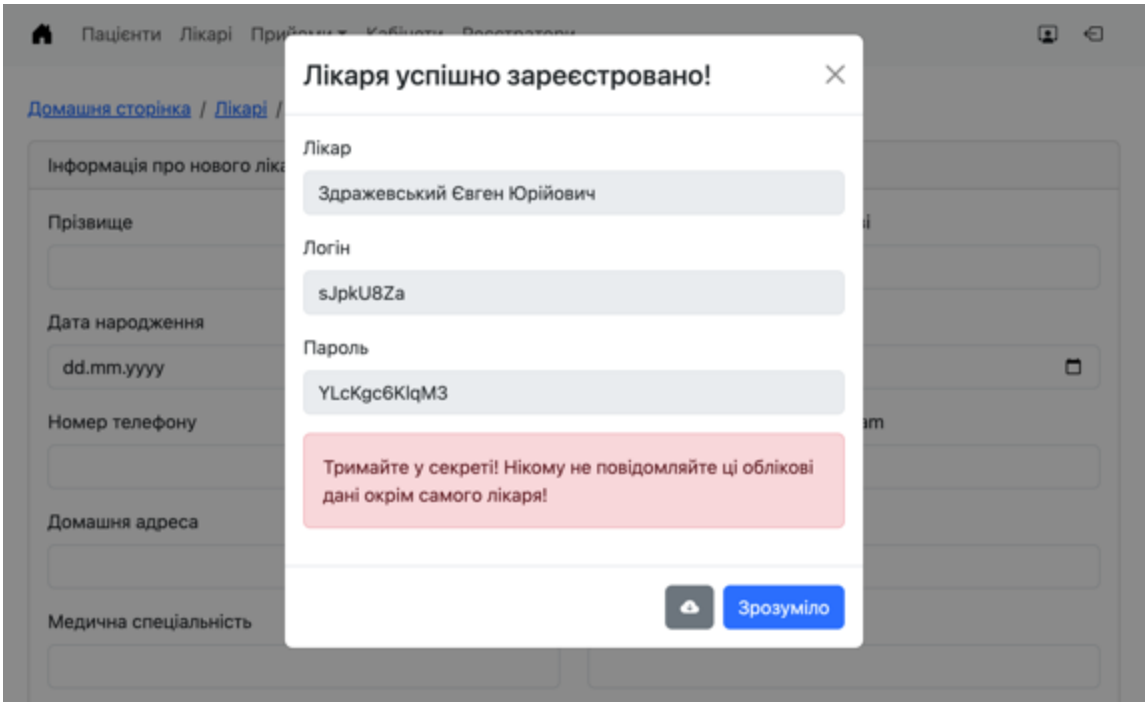


Рисунок 2.20 – Результат внесення інформації про нового лікаря (Створення облікового запису)

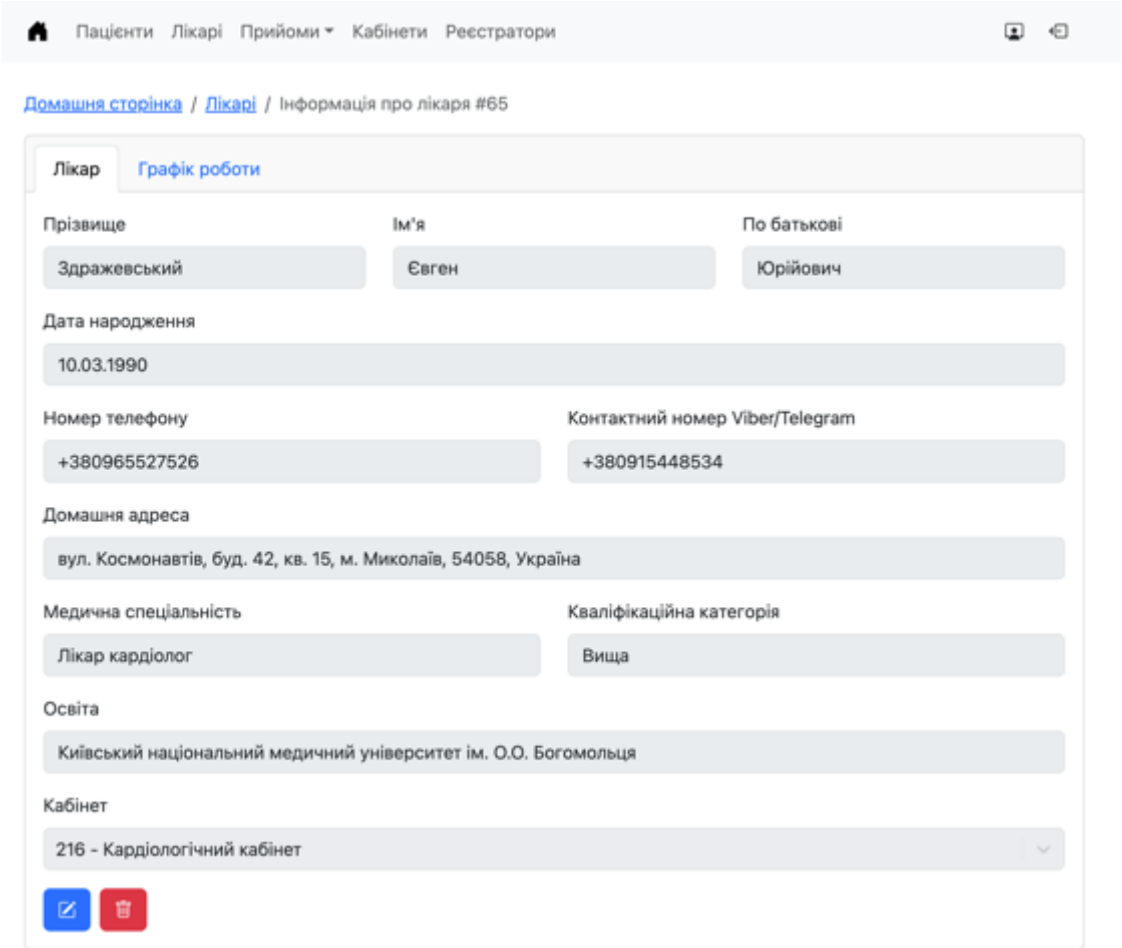


Рисунок 2.21 – Результат внесення інформації про нового лікаря (Інформація про нового лікаря)

Поведінка функції внесення інформації про нового лікаря відповідає очікуваному результату.

Випробування функції "Призначити новий прийом пацієнта до лікаря".

Для призначення нового прийому спочатку було здійснено перехід на сторінку "Пацієнти" з навігаційного меню. На цій сторінці був обраний бажаний пацієнт та ініційовано призначення йому нового прийому шляхом натискання на піктограму журналу поряд з ім'ям пацієнта у таблиці. Після цього користувача перевело на сторінку перегляду розкладу лікарів, де був обраний бажаний лікар та дата для призначення прийому. Далі був обраний час для прийому на таймлайні розкладу роботи лікаря, після чого автоматично відкрилось вікно підтвердження призначення прийому. Вибір часу прийому та вікно підтвердження призначення нового прийому представлено на рисунках 2.22 та 2.23 відповідно.

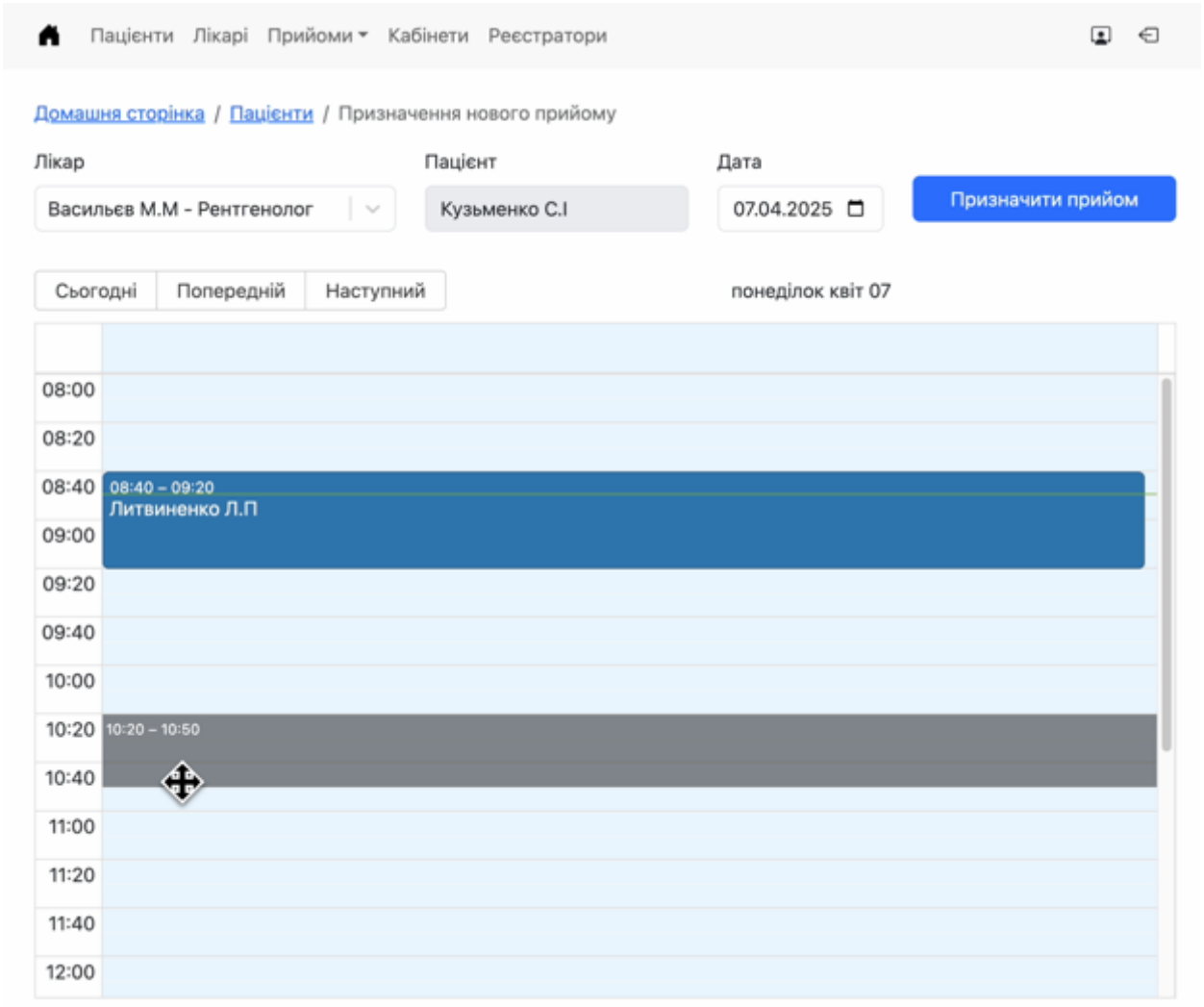


Рисунок 2.22 – Вибір часу прийому для призначення нового прийому пацієнта до лікаря

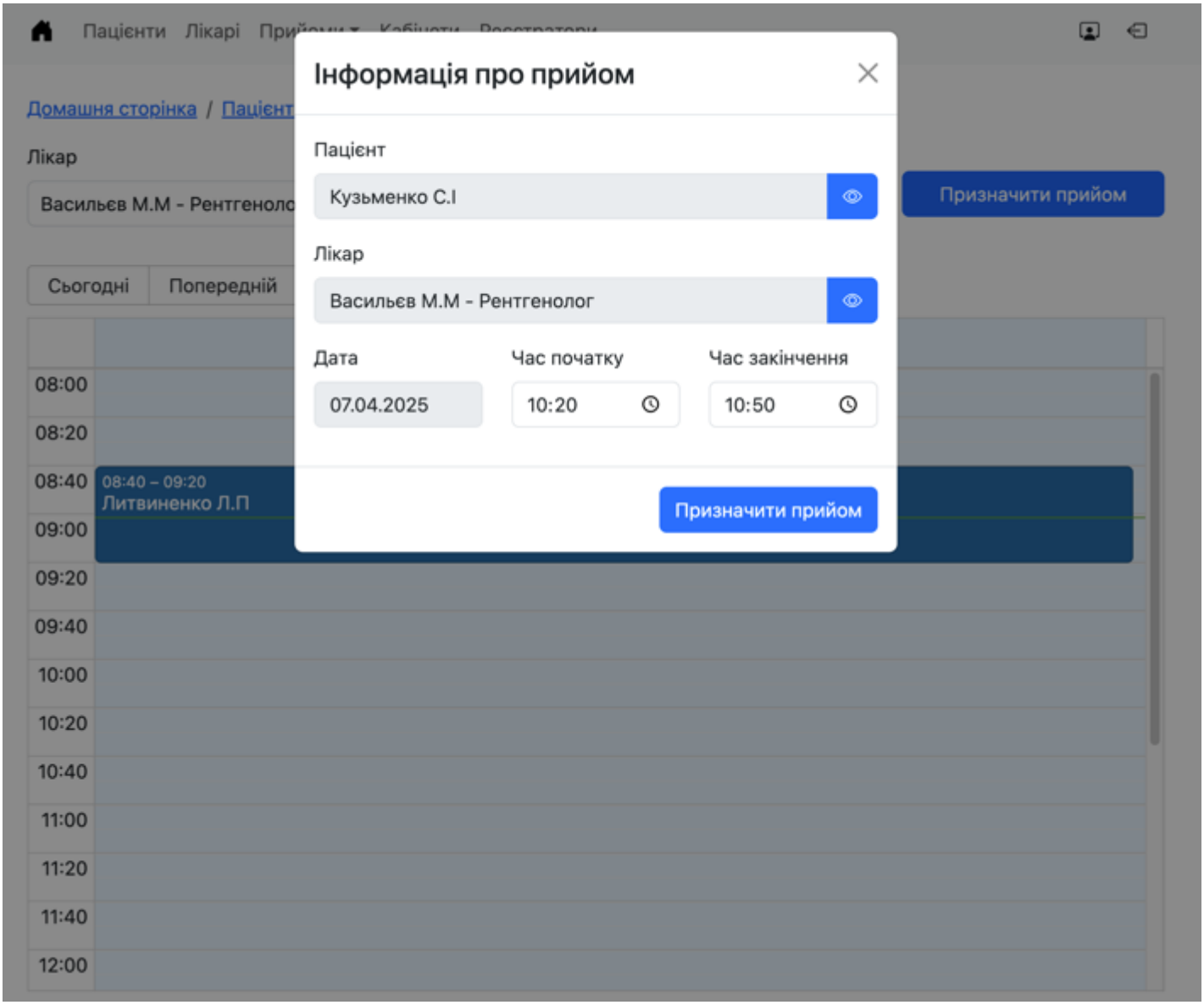


Рисунок 2.23 – Вікно підтвердження призначення нового прийому пацієнта до лікаря

Після підтвердження призначення нового прийому натиснувши на кнопку керування "Призначити прийом", інформація про новий прийом була успішно збережена та відображено відповідне повідомлення. Результат призначення нового прийому представлено на рисунку 2.24.

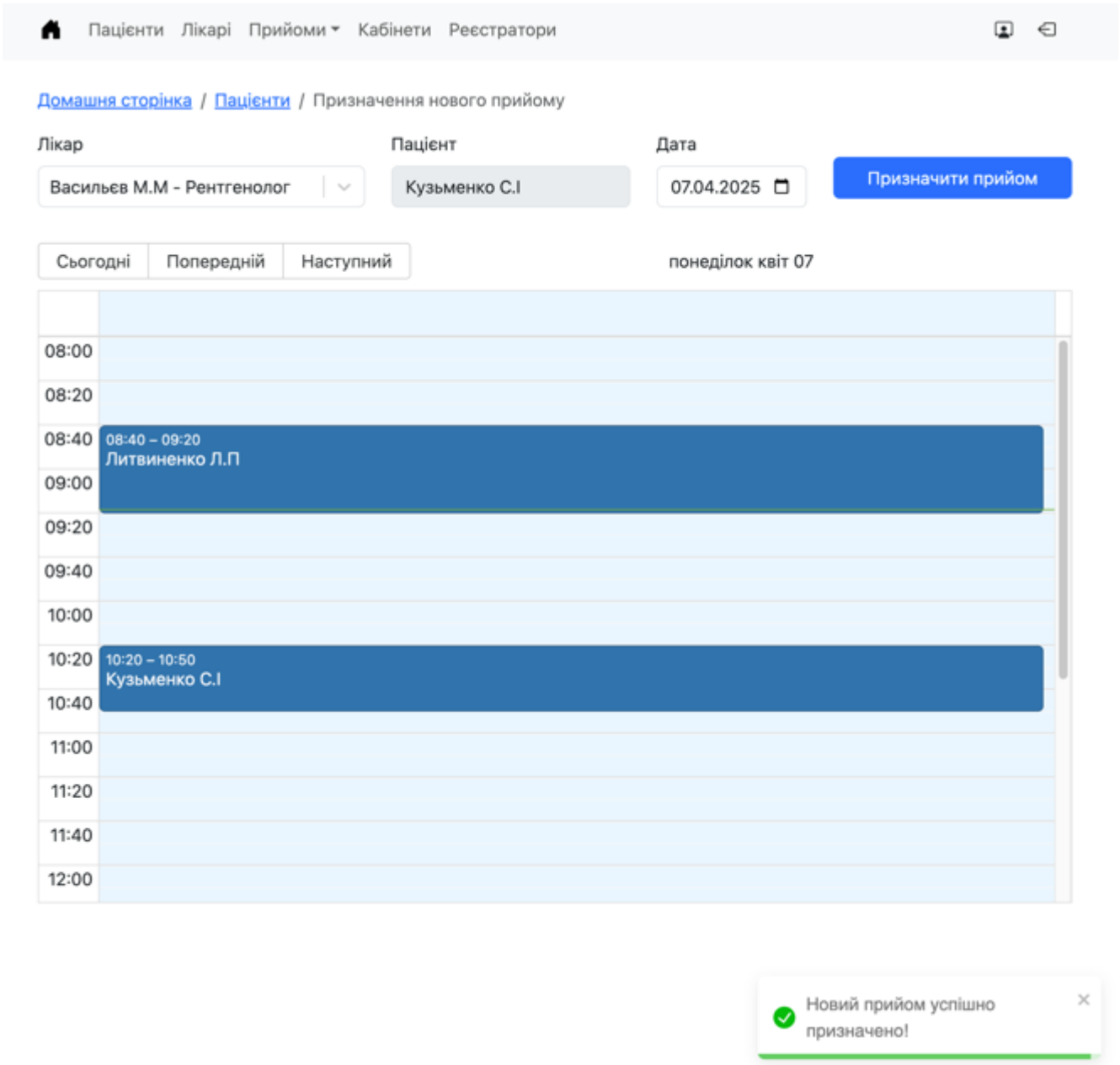


Рисунок 2.24 – Результат призначення нового прийому на сторінці розкладу роботи лікаря

Поведінка функції призначення нового прийому пацієнта до лікаря відповідає очікуваному результату.

Під час проведення всіх випробувань, помилок у роботі вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" не виявлено, випробування пройшло успішно. В результаті проведення тестування та випробування, можна зробити висновок, що вебзастосунок повністю працездатний та відповідає поставленим вимогам.

3 РЕЗУЛЬТАТИ РОЗРОБКИ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ РЕЄСТРАТУРИ МЕДИЧНОЇ КЛІНІКИ "COMPLIMED"

У результаті виконання кваліфікаційної роботи була розв'язана практична задача інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме розроблено вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed".

Було проведено аналіз роботи реєстратури медичної клініки "CompliMed" та досліджено існуючі програмні рішення для вирішення проблеми автоматизації роботи реєстратури медичного закладу.

Розроблено вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed": проаналізовано вимоги до вебзастосунку, спроєктовано архітектуру вебзастосунку, розроблено ескізний, технічний та робочий проєкти вебзастосунку. Функціональна модель вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" розроблена використовуючи об'єктно-орієнтовану методологію аналізу та проєктування розробки.

На етапі ескізного проєкту було розроблено діаграми діяльності для основних варіантів використання, побудовано концептуальну модель даних у вигляді діаграми "сутність-зв'язок" та створено прототип користувацького інтерфейсу вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

В рамках технічного проєкту було побудовано статичну та динамічну моделі вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed", а також розроблено логічну модель бази даних. Статична модель була представлена за допомогою діаграми класів та специфікацій прописаних до кожного класу. Динамічну модель подано діаграмами послідовності для основних варіантів використання.

У робочому проєкті було обгрунтовано вибір засобів розробки, розроблена фізична модель бази даних, реалізовано вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed", а також проведено його тестування та випробовування згідно програми та методики випробувань, яке продемонструвало

повну працездатність вебзастосунку та його відповідність вимогам, зазначеним у технічному завданні.

Розроблений вебзастосунок реалізує наступні функції:

– перегляд, додавання, редагування та видалення інформації про пацієнтів;

– перегляд, додавання, редагування та видалення інформації про лікарів;

– перегляд інформації про графіки роботи лікарів та їх редагування.

– перегляд розкладу роботи лікарів згідно графіків роботи;

– перегляд, додавання, редагування та видалення інформації про реєстраторів;

– перегляд, додавання, редагування та видалення інформації про кабінети медичної клініки;

– перегляд, призначення, коригування та скасування прийомів пацієнтів у відповідності з графіками роботи лікарів.

– внесення та перегляд інформації про консультації пацієнтів в прийомах.

– створення та видалення облікових записів користувачів: лікарів та реєстраторів.

Була розроблена вся необхідна програмна документація, яка вимагається технічним завданням:

– технічне завдання наведено у додатку А;

– текст програми наведено у додатку Б;

– опис програми наведено у додатку В;

– інструкцію користувача наведено у додатку Г;

– програму та методику випробувань наведено у додатку Д.

Основними інструментами для розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" були мова програмування Java, фреймоворк Spring Boot та база даних MariaDB.

Вихідний код розробленого вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" займає 5,4 МБ дискового простору. Вебзастосунок складається з 5145 рядків коду. Вихідний архів вебзастосунку з байт-кодом скомпільованого Java коду, який виконується на віртуальній машині Java займає 64,5 МБ дискового простору.

4 ОХОРОНА ПРАЦІ

4.1 Національне законодавство про охорону праці

Законодавство України про охорону праці складається з конституційних гарантій прав громадян у цій сфері, спеціального Закону України «Про охорону праці», Кодексу законів про працю України, ряду інших законів, пов'язаних з охороною життя й здоров'я громадян у процесі їх трудової діяльності. У Конституції України [34] затверджується: «Людина, її життя й здоров'я, безпека визнаються в Україні найвищою соціальною цінністю». В інших статтях проголошені права громадян на:

– належні, безпечні й здорові умови праці (ст. 43);

– соціальний захист громадян, забезпечення їх у випадку повної, часткової або тимчасової втрати працездатності, втрати годувальника (ст. 46);

– охорону здоров'я, медичну допомогу й страхування (ст.49);

– безпечне для життя й здоров'я навколишнє середовище й відшкодування заподіяної порушенням цього права шкоди (ст. 50).

Основним законодавчим документом у галузі охорони праці є Закон України «Про охорону праці» [35] , дія якого поширюється на всі підприємства, установи й організації незалежно від форм власності й видів діяльності, на всіх громадян, які працюють, а також притягнуті до праці на цих підприємствах. Цей Закон визначає основні положення по реалізації конституційного права трудящих на охорону їх життя й здоров'я в процесі трудової діяльності, на належні, безпечні й здоровіші умови праці, регулює відносини між роботодавцем і працівником з питань безпеки, гігієни праці й виробничого середовища й установлює єдиний порядок організації охорони праці на Україні.

Складовою частиною законодавства про охорону праці є Кодекс законів про працю (КЗпП) України [36], який регулює трудові відносини в цілому. У Кодексі питання охорони праці відображені в ряді статей і в спеціальному розділі відповідно Закону «Про охорону праці».

4.2 Ергономічні вимоги до організації робочих місць користувачів комп’ютерів

Організація робочого місця користувача комп’ютера повинна відповідати вимогами НПАОП 0.00-1.31-10. Так, площа, на якій розташовується одне робоче місце з відео дисплейним терміналом (ВДТ), повинна становити не менше 6.0 $м^2$, а об’єм приміщення – не менше 20 $м^3$. Робочі місця з ВДТ розміщуються на відстані не менше 1 м від стіни зі світловими прорізами; відстань між бічними поверхнями ВДТ має бути не менше 1,2 м; відстань між тильною поверхнею одного ВДТ та екраном іншого не повинна бути меншою за 2,5 м; прохід між рядами робочих місць має бути не менше метра. Необхідно також враховувати розміри меблів для комп’ютеризованих робочих місць, тобто висота 725 мм, ширина 600 – 1400 мм, глибина 800 – 1000 мм. Зокрема, розміри столу для ВДТ складають: ширина – 1200 мм, глибина – 800 мм. Особливу увагу необхідно звернути на розміщення відео терміналів. Для того щоб уникнути дзеркального відображення на екрані ВДТ джерел природного освітлення, їх необхідно розставити вздовж стіни з вікнами.

З метою зменшення потрапляння шуму з суміжних робочих місць та забезпечення концентрації уваги під час виконання робіт, що вимагають напруженості, необхідно відокремити робочі місця перегородками висотою 1,5 – 2 м.

Вагомим фактором у забезпеченні безпеки праці користувачів комп'ютерів є характер розташування на робочому місці відео термінала, клавіатури та принтера. Розташування екрана (дисплея) повинно забезпечувати зручність зорового спостереження у вертикальній площині під кутом ± 30° від лінії зору оператора. Найкращі зорові умови й можливість розпізнавання цифр, символів досягається тоді, коли верхній край відео термінала знаходиться на висоті очей, а погляд спрямований вниз на центр екрана. Оскільки при роботі з комп'ютером найбільш сприятливим вважається нахил голови вперед, приблизно на 20° від вертикалі (при такому положенні голови м'язи шиї розслабляються), то екран відео термінала також повинен бути нахилений назад на 20° від вертикалі. Екран відео термінала та клавіатура повинні розташовуватись на оптимальній відстані від очей

користувача комп'ютера, але не ближче 600 мм, з урахуванням розміру абетково-цифрових знаків і символів. Так, при розмірі екрана по діагоналі 35 см, відстань від монітора до очей повинна складати 60 – 70 см, при діагоналі 43 см – 70 см, при діагоналі 48 см – 80 см [37].

4.3 Освітлення приміщень

Головне завдання освітлення – створити найкращі умови для органів зору. Це завдання може бути вирішене тоді, коли виконуються такі вимоги до освітлення:

– Освітленість на робочому місці повинна відповідати характеру роботи органів зору, що визначається величиною найбільш дрібних предметів або їх частин, які необхідно відрізняти під час роботи, а також фоном та контрастом об'єкта розглядання і фону. Чим дрібніший об'єкт, темніший фон, менший контраст, тим більша величина освітленості потрібна для створення оптимальних умов праці.

– Необхідно забезпечувати достатньо рівномірне освітлення робочої поверхні, а також навколишнього простору, щоб у полі зору не було поверхні з яскравістю, що значно відрізняється від інших. У противному разі переведення погляду з ярко освітленої поверхні на слабко освітлену викликає необхідність у переадаптації органів зору, що призводить до їх швидкої втоми.

– На робочій поверхні не повинно бути різких тіней. Їх наявність створює нерівномірну яскравість поверхні в полі зору, що веде до швидкої втоми.

– У полі зору не повинно бути прямої та відображеної блискучості (підвищеної яскравості випромінюючої поверхні), що може призвести до тимчасового осліплення. Пряма блискучість зв'язана з джерелами світла. Її зменшують шляхом зниження яскравості джерел. Відображену блискучість зменшують відповідним вибором напрямку світлового потоку або зміною кута нахилу робочої поверхні.

– Величина освітленості повинна бути постійною у часі. Коливання освітленості виникають у разі змін напруги в електричній мережі, а також зв'язані з особливостями роботи джерел світла.

– Спектральний склад світла повинен по можливості забезпечувати правильну передачу кольору, тому штучне світло, що використовується на

підприємствах, за своїм спектральним складом має наближатися до природного.

– Освітлення повинно бути надійним, простим в експлуатації та економічним. Джерела світла не повинні створювати небезпечних та шкідливих факторів (шум, теплові випромінювання, небезпеку враження струмом, пожежота вибухонебезпечність) [37].

4.4 Організаційні заходи щодо попередження електротравм

Згідно з НПАОП 0.00-1.21-98 «Правила безпечної експлуатації електроустановок споживачів» відповідальність за організацію безпечної експлуатації електроустановок покладається на роботодавця, який створює необхідні для цього служби, призначає відповідальних осіб, розробляє та затверджує інструкції, забезпечує перевірку знань з електробезпеки тощо.

Згідно з чинними вимогами роботодавець повинен:

– призначити відповідального за справний стан і безпечну експлуатацію електроустановок (далі — відповідальний за електрогосподарство);

– створити і укомплектувати відповідно до потреб електротехнічну службу;

– розробити і затвердити посадові інструкції працівників електротехнічної служби та інструкції з безпечного виконання робіт в електроустановках з урахуванням їх особливостей;

– створити на підприємстві такі умови, щоб працівники, на яких покладено обов'язки з обслуговування електроустановок, відповідно до чинних вимог своєчасно здійснювали їх огляд, профілактичні, протиаварійні та приймально-здавальні випробування;

– забезпечити своєчасне навчання і перевірку знань працівників з питань електробезпеки.

На малих підприємствах за неможливості чи недоцільності створення електротехнічної служби власник, на договірних засадах, доручає електротехнічним службам споріднених підприємств або фізичним особам, які

мають відповідну підготовку, забезпечення справного стану і безпечної експлуатації електроустановок.

Обслуговування діючих електроустановок, проведення в них оперативних переключень, організація та виконання ремонтних, монтажних, налагоджувальних робіт та випробувань здійснюються спеціально підготовленим електротехнічним персоналом. Ці працівники повинні мати відповідну професійну підготовку, групу з електробезпеки (I – V), підтверджену посвідченням установленої форми, і не мати медичних протипоказань і вікових обмежень щодо можливості виконання роботи в електроустановках [37].

4.5 Організаційно-технічні заходи пожежної безпеки

Увесь комплекс організаційно-технічних заходів можна поділити на організаційні, технічні, режимні та експлуатаційні.

Організаційні заходи пожежної безпеки передбачають: створення пожежної охорони на об’єкті, проведення навчань з питань пожежної безпеки (включаючи інструктажі та пожежно-технічні мінімуми), застосування наочних засобів протипожежної пропаганди та агітації, організацією ДПД та ПТК, проведення перевірок, оглядів стану пожежної безпеки приміщень, будівель, об'єкта в цілому тощо.

До технічних заходів належать: дотримання правил і норм, визначених чинними нормативно-правовими актами при спорудженні та реконструкції приміщень, будівель й об’єктів, технічному переоснащенні виробництва, експлуатації чи можливому переобладнанні електромереж, опалення, вентиляції, освітлення і т. п.

Заходи режимного характеру передбачають заборону куріння та застосування відкритого вогню в недозволених місцях, недопущення появи сторонніх осіб у вибухонебезпечних приміщеннях чи об’єктах, регламентацію пожежної безпеки при проведенні вогневих робіт тощо.

Експлуатаційні заходи охоплюють своєчасне проведення профілактичних оглядів, випробувань, ремонтів технологічного та допоміжного устаткування, а

також інженерного господарства (електромереж, електроустановок, опалення, вентиляції).

Законом України «Про пожежну безпеку» державні органи управління наділяються певними повноваженнями та встановлюються обов’язки керівників підприємств і працівників з пожежної безпеки. Центральні органи виконавчої влади забезпечують проведення єдиної політики в галузі пожежної безпеки, розробку та затвердження державних стандартів, норм і правил пожежної безпеки, організацію навчання з пожежної безпеки, оперативне управління силами і технічними засобами, які залучаються до ліквідації великих пожеж, координацію роботи щодо створення і випуску пожежної техніки та засобів протипожежного захисту тощо.

Відповідальність за стан пожежної безпеки підприємства покладається на його власника (керівника) та уповноважених ним осіб. Вони зобов'язані розробляти комплексні заходи з пожежної безпеки, здійснювати постійний контроль за дотриманням чинних нормативно-правових актів з пожежної безпеки, розробляти та затверджувати нормативні акти, що діють у межах підприємства, організовувати навчання працівників, утримувати в справному стані засоби протипожежного захисту і зв'язку, створювати у разі потреби підрозділи пожежної охорони, проводити службове розслідування випадків пожеж тощо.

З метою координації робіт з пожежної безпеки та контролю за проведенням і виконанням протипожежних заходів у міністерствах, інших центральних органах виконавчої влади, в об'єднаннях підприємств різної форми власності створюються служби пожежної безпеки, діяльність яких регламентується Типовим положенням про службу пожежної безпеки, затвердженим наказом №220 МВС України 12 квітня 1995 р.

Для захисту життя і здоров'я громадян та матеріальних цінностей від пожеж, підтримки належного рівня пожежної безпеки на об'єктах і в населених пунктах створюється система пожежної охорони. Її основні завдання – це контроль за дотриманням протипожежних вимог, запобігання пожежам і нещасним випадкам, гасіння пожеж, рятування людей та надання допомоги в ліквідації наслідків аварій, катастроф та стихійного лиха [37].

ВИСНОВКИ

В результаті виконання кваліфікаційної роботи було розв'язано практичну задачу інженерії програмного забезпечення, що характеризується комплексністю та невизначеністю умов, а саме задачу розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

Було проведено аналіз практичної задачі інженерії програмного забезпечення з розробки вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed": проаналізовано роботу реєстратури медичної клініки "CompliMed" та досліджено сучасні підходи щодо розробки вебзастосунків для автоматизації роботи реєстратури у медичних закладах, проведено аналіз існуючих програмних продуктів для автоматизації роботи реєстратури медичних закладів, зроблено висновок про доцільність розробки власного програмного забезпечення та сформулювано постановку задачі на розробку вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed".

Проведено аналіз вимог до вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed", розроблена архітектура вебзастосунку, створено ескізний, технічний, та робочий проєкти. Здійснено тестування та випробовування вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed", яке свідчить про його цілковиту працездатність та відповідність вимогам.

Розроблено всю необхідну програмну документацію до вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" та представлено спеціальній розділ з охорони праці.

Мета кваліфікаційної роботи була досягнута. Всі завдання, які потрібно було вирішити для досягнення мети кваліфікаційної роботи були виконані в повному обсязі.

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. Медична інформаційна система, що це? URL: https://blog.h24.ua/uk/shho-take-mis/ (дата зверення 27.02.2025).

2. 5 можливостей медичної інформаційної системи для клініки. URL: https://imed.ua/mozhlyvosti-imed/5-mozhlivostej-medichnoi-informacijnoi-sistemi-dlya-kliniki (дата зверення 27.02.2025).

3. Рейтинг мов програмування 2025. URL: https://dou.ua/lenta/articles/language-rating-2025/ (дата звернення 28.02.2025).

4. DB-Engines Ranking. URL: https://db-engines.com/en/ranking (дата звернення 28.02.2025).

5. Розширення функціоналу онлайн-комунікації медичної інформаційної системі HELSI. URL: https://er.knutd.edu.ua/handle/123456789/23364 (дата звернення 28.02.2025).

6. How microservices support IT integration in healthcare. URL: https://www.redhat.com/en/topics/microservices/microservices-in-healthcare (дата звернення 28.02.2025)

7. Как разрабатывалась украинская цифровая платформа здравоохранения eHealth. URL: https://ain.ua/ru/2017/11/21/kak-razrabatyvalas-ehealth/ (дата звернення 28.02.2025).

8. MedCenter+. URL: https://medcentercrm.com/ (дата звернення 28.02.2025).

9. Clinica Web. URL: https://www.clinica-web.ua (дата зверення 28.02.2025).

10. Doktor Eleks. URL: https://doctor.eleks.com/ (дата звернення 28.03.2025).

11. MVC Design Pattern. URL: https://www.geeksforgeeks.org/mvc-design-pattern/ (дата звернення 28.02.2025).

12. Component Diagram. URL: https://plantuml.com/component-diagram (дата звернення 28.02.2025).

13. Component Diagrams. URL: https://sparxsystems.com/resources/tutorials/uml2/component-diagram.html (дата звернення 28.02.2025).

14. Deployment Diagrams. URL: https://www.ibm.com/docs/en/rational-soft-arch/9.7.0?topic=diagrams-deployment (дата звернення 28.02.2025).

15. Activity Diagram. URL: https://sparxsystems.com/resources/tutorials/uml2/activity-diagram.html (дата звернення 28.02.2025).

16. What is Entity Relationship Diagram? URL: https://www.baeldung.com/cs/erd (дата звернення 28.02.2025).

17. Class Diagrams. URL: https://www.ibm.com/docs/en/rsm/7.5.0?topic=structure-class-diagrams (дата звернення 03.03.2025).

18. UML Sequence Diagrams. URL: https://www.uml-diagrams.org/sequence-diagrams.html (дата звернення 03.03.2025).

19. Logical Data Modeling. URL: https://www.erwin.com/learn/logical.aspx (дата звернення 03.03.2025).

20. MariaDB. URL: https://mariadb.org/ (дата звернення: 04.03.2025).

21. Al Yasrebi. MariaDB vs. PostgreSQL Performance. URL: https://aliyasrebi.medium.com/database-comparison-c1778daaa7b4 (дата звернення: 04.03.2025).

22. Redis. URL: https://redis.io/docs/latest/get-started/ (дата звернення 04.03.2025).

23. What is Java technology and why do I need it? URL: https://www.java.com/en/download/help/whatis_java.html (дата звернення: 04.03.2025).

24. IntelliJ IDEA overview. URL: https://www.jetbrains.com/help/idea/discover-intellij-idea.html (дата звернення: 04.03.2025).

25. Apache Maven. URL: https://maven.apache.org/ (дата звернення: 04.03.2025).

26. Spring - MVC Framework. URL: https://www.geeksforgeeks.org/spring-mvc-framework/ (дата звернення: 04.03.2025).

27. HTML: HyperText Markup Language. URL: https://developer.mozilla.org/en-US/docs/Web/HTML (дата звернення: 04.03.2025).

28. Learn to style HTML using CSS. URL: https://developer.mozilla.org/en-US/docs/Learn/CSS (дата звернення: 04.03.2025).

29. React. URL: https://react.dev/ (дата звернення 04.03.2025).

30. Next.js. URL: https://nextjs.org/ (дата звернення 04.03.2025).

31. React Bootstrap. URL: https://react-bootstrap.netlify.app/ (дата звернення 04.03.2025).

32. Physical Data Models. URL: https://www.ibm.com/docs/en/ida/9.1.1?topic=modeling-physical-data-models (дата звернення 04.03.2025).

33. Jorgensen P. C. Software Testing. Boca Raton : CRC Press, 2014. С. 99–100.

34. Конституція України : від 28.06.1996 № 254к/96-ВР : станом на 1 січ. 2020 р. URL: https://zakon.rada.gov.ua/laws/show/254к/96-вр#Text (дата звернення 03.04.2025)

35. Про охорону праці : Закон України від 14.10.1992 № 2694-XII : станом на 1 січ. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/2694-12#Text (дата звернення: 03.04.2025).

36. Кодекс законів про працю України : Кодекс України від 10.12.1971 № 322-VIII : станом на 1 січ. 2025 р. URL: https://zakon.rada.gov.ua/laws/show/322-08#Text (дата звернення: 03.04.2025).

37. Голінько В.І., Іконніков М.Ю., Лебедєв Я.Я. Охорона праці в галузі інформаційних технологій : навч. посіб. М-во освіти і науки України, Нац. гірн. ун-т: Д. : НГУ, 2015. С. 97–98, 57–58, 202–203, 234–235.

ДОДАТОК А – ТЕХНІЧНЕ ЗАВДАННЯ НА РОЗРОБКУ ВЕБЗАСТОСУНКУ ДЛЯ АВТОМАТИЗАЦІЇ РОБОТИ РЕЄСТРАТУРИ МЕДИЧНОЇ КЛІНІКИ "COMPLIMED"

1 Вступ

Повна назва розроблюваного проекту: "Вебзастосунок для автоматизації роботи реєстратури медичного клініки CompliMed".

Галузь застосування: автоматизація реєстратури, управління записами пацієнтів, зберігання та обробка медичних даних, облік лікарів і кабінетів, запис на прийом, адміністрування медичної клініки, звітність.

2 Підстави для розробки

Підставою для розробки вебзастосунку є завдання на кваліфікаційну роботу на здобуття ступеня вищої освіти "бакалавр".

3 Призначення розробки

3.1 Експлуатаційне призначення

Експлуатаційним призначенням розробки є спрощення роботи реєстратури медичної клініки "CompliMed", та покращення ефективності роботи медичної клініки, а саме:

– покращення ефективності роботи реєстратури та медичного персоналу шляхом автоматизації рутинних операцій, зменшення ручних операцій та помилок при плануванні прийомів пацієнтів;

– забезпечення оперативного централізованого доступу до інформації для персоналу, що дозволяє швидко призначати прийоми та вести облік медичних даних пацієнтів;

– забезпечення безпеки даних: захист медичних даних від несанкціонованого доступу, резервне копіювання та відновлення даних для запобігання втрат інформації.

3.2 Функціональне призначення

Управління даними пацієнтів:

Перегляд, додавання, редагування та видалення інформації про пацієнтів.

Управління даними лікарів:

Перегляд, додавання, редагування та видалення інформації про лікарів.

Перегляд та управління графіками роботи лікарів, включаючи можливість додавання, коригування та видалення графіків роботи на кожний день тижня.

Перегляд розкладів роботи лікарів згідно графіків роботи.

Управління даними реєстраторів:

Перегляд, додавання, редагування та видалення інформації про реєстраторів.

Управління даними кабінетів:

Перегляд, додавання, редагування та видалення інформації про кабінети медичної клініки

Управління прийомами пацієнтів:

Перегляд, призначення, коригування та скасування прийомів пацієнтів у відповідності з графіками роботи лікарів.

Внесення та перегляд інформації про консультації пацієнтів в прийомах, включаючи симптоми, діагнози та медичні рекомендації.

Управління обліковими записами:

Створення та видалення облікових записів користувачів: лікарів та реєстраторів

4 Вимоги до програми та програмної документації

4.1 Вимоги до функціональних характеристик

4.1.1 Вимоги до складу виконуваних функцій

Вебзастосунок повинен забезпечити можливість виконання нижче перерахованих функцій

1) Авторизація

Вхідні дані: логін, пароль.

Вихідні дані: повідомлення про успішність авторизації.

2) Додавання нового пацієнта

Вхідні дані: номер пацієнта, прізвище, ім’я, по батькові пацієнта, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

Вихідні дані: повідомлення про додавання нового пацієнта.

3) Оновлення даних про пацієнта

Вхідні дані: номер пацієнта, прізвище, ім’я, по батькові пацієнта, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

Вихідні дані: повідомлення про оновлення даних про пацієнта.

4) Видалення пацієнта

Вхідні дані: номер пацієнта.

Вихідні дані: повідомлення про видалення пацієнта.

5) Додавання нового лікаря

Вхідні дані: номер лікаря, прізвище, ім'я, по батькові лікаря, лікарська спеціальність, лікарська кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер кабінету.

Вихідні дані: повідомлення про додавання нового лікаря з згенерованим логіном та паролем від створеного облікового запису.

6) Оновлення даних про лікаря

Вхідні дані: номер лікаря, прізвище, ім'я, по батькові лікаря, лікарська спеціальність, лікарська кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер кабінету.

Вихідні дані: повідомлення про оновлення даних про лікаря.

7) Видалення даних про лікаря

Вхідні дані: номер лікаря.

Вихідні дані: повідомлення про видалення лікаря.

8) Додавання нового прийому пацієнта до лікаря

Вхідні дані: номер прийому, номер пацієнта, номер лікаря, дата, час початку прийому, час закінчення прийому.

Вихідні дані: повідомлення про додавання нового прийому.

9) Оновлення даних про прийом

Вхідні дані: номер прийому, номер пацієнта, номер лікаря, дата, час початку прийому, час закінчення прийому.

Вихідні дані: повідомлення про оновлення даних про прийом.

10) Видалення даних про прийом

Вхідні дані: номер прийому.

Вихідні дані: повідомлення про видалення даних про прийом.

11) Додавання нового реєстратора

Вхідні дані: номер реєстратора, прізвище, ім'я, по батькові реєстратора, дата народження.

Вихідні дані: повідомлення про додавання нового реєстратора з згенерованим логіном та паролем від створеного облікового запису.

12) Оновлення даних про реєстратора

Вхідні дані: номер реєстратора, прізвище, ім'я, по батькові реєстратора, дата народження.

Вихідні дані: повідомлення про оновлення даних про реєстратора.

13) Видалення даних про реєстратора

Вхідні дані: номер реєстратора.

Вихідні дані: повідомлення про видалення даних про реєстратора.

14) Внесення даних про графік роботи лікаря

Вхідні дані: номер графіку роботи, номер лікаря, номер дня тижня, час початку роботи, час закінчення роботи.

Вихідні дані: повідомлення про оновлення графіку роботи.

15) Додавання нового кабінету

Вхідні дані: номер кабінету, номер кабінету в медичній клініці, назва кабінету.

Вихідні дані: повідомлення про додавання нового кабінету.

16) Оновлення даних про кабінет

Вхідні дані: номер кабінету, номер кабінету в медичній клініці, назва кабінету.

Вихідні дані: повідомлення про оновлення даних про кабінет.

17) Видалення даних про кабінет

Вхідні дані: номер кабінету.

Вихідні дані: повідомлення про видалення даних про кабінет.

18) Внесення даних про консультацію прийому

Вхідні дані: номер прийому, симптоми, діагноз, медичні рекомендації.

Вихідні дані: повідомлення про оновлення даних про консультацію.

19) Формування інформації про пацієнтів

Вхідні дані: запит на перегляд списку всіх пацієнтів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові пацієнта, датою народження, або номер пацієнта для перегляду інформації про певного пацієнта.

Вихідні дані:

– список усіх пацієнтів: номер пацієнта, прізвище, ім’я, по батькові, дата народження.

– інформація про певного пацієнта: номер пацієнта, прізвище, ім’я, по батькові, дата народження, номер телефону, контактний номер Viber/Telegram, пільгова категорія, домашня адреса.

20) Формування інформації про лікарів

Вхідні дані: запит на перегляд списку всіх лікарів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові лікаря, датою народження, лікарською спеціальністю, або номер лікаря для перегляду інформації про певного лікаря.

Вихідні дані:

– список усіх лікарів: номер лікаря, прізвище, ім’я, по батькові, дата народження, лікарська спеціальність, номер кабінету.

– інформація про певного лікаря: номер лікаря, прізвище, ім’я, по батькові, лікарська спеціальність, кваліфікаційна категорія, освіта, контактний номер Viber/Telegram, дата народження, номер телефону, домашня адреса, номер та назва кабінету.

21) Формування інформації про реєстраторів

Вхідні дані: запит на перегляд списку всіх реєстраторів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові реєстратора, датою народження, або номер реєстратора для перегляду інформації про певного реєстратора.

Вихідні дані:

– список усіх реєстраторів: номер реєстратора, прізвище, ім’я, по батькові, дата народження.

– інформація про певного реєстратора: номер реєстратора, прізвище, ім’я, по батькові, дата народження.

22) Формування інформації про прийоми

Вхідні дані: запит на перегляд списку всіх прийомів з опціональними критеріями пошуку за прізвищем, ім’ям, по батькові пацієнта, прізвищем, ім’ям, по батькові лікаря, датою прийому, часом початку прийому, або номер прийому для перегляду інформації про певний прийом.

Вихідні дані:

– список усіх прийомів: номер прийому, пацієнт (ПІБ), лікар (ПІБ), номер кабінету в медичній клініці, дата та час початку прийому.

– інформація про певний прийом: номер прийому, лікар (ПІБ), пацієнт (ПІБ), номер кабінету в медичній клініці та назва кабінету лікаря, дата прийому, час початку прийому, час закінчення прийому.

23) Формування інформації про консультації

Вхідні дані: номер прийому.

Вихідні дані: симптоми, діагноз, медичні рекомендації.

24) Формування інформації про графіки роботи лікарів

Вхідні дані: номер лікаря.

Вихідні дані: день тижня, час початку роботи, час закінчення роботи.

25) Формування інформації про кабінети

Вхідні дані: запит на перегляд списку всіх кабінетів з опціональними критеріями пошуку за назвою кабінету, номером кабінету, або номер кабінету для перегляду інформації про певний кабінет.

Вихідні дані:

– список усіх кабінетів: номер кабінету в медичній клініці, назва кабінету.

– інформація про певний кабінет: номер кабінету в медичній клініці, назва кабінету.

26) Формування інформації про розклад роботи лікарів

Вхідні дані: номер лікаря, дата.

Вихідні дані: набір часових проміжків по 30 хвилин згідно графіку роботи лікаря на певну дату, прийоми лікаря до кожного часового проміжку або вільний стан часового проміжку.

4.2 Вимоги до організації вхідних і вихідних даних

Вхідною інформацією є дані, які вводяться вручну у поля та форми. Залежно від типу поля, дані можуть бути: числові цілі, текстові, дати, або варіанти, обрані з представлених списків (для уникнення можливих помилок, у випадних списках можуть бути представлені готові варіанти вибору).

Вихідними даними є результати виконаних операцій, які відображаються у вигляді екранних форм або повідомлень.

4.3 Вимоги до надійності

Середній час безвідмовної роботи – 30 робочих днів.

Застосунок повинен бути доступним 24/7. Лікарі та працівники реєстратури можуть працювати в будь-який час, тому недоступність системи може вплинути на роботу медичної клініки та пацієнтів.

Недоступність – час, що витрачається на обслуговування системи не повинно перевищувати 3% від загального часу роботи.

Максимальна норма помилок або дефектів – 1 помилка на десять тисяч рядків коду.

Надійне функціонування програми має бути забезпечене виконанням сукупності організаційно-технічних заходів, перелік яких подано нижче:

– система повинна мати можливість автоматичного відновлення після відмови або помилки протягом 15 хвилин. Це може включати резервне копіювання бази даних, автоматичну перезавантаження серверів та інші заходи для забезпечення продовження роботи системи;

– захист даних від несанкціонованого доступу;

– валідація даних перед додаванням або оновленням будь-якої інформації, важливо перевіряти вхідні дані на відповідність формату та обмеженням.

4.4 Вимоги до складу й параметрів технічних засобів

Система повинна задовольняти вказаним вимогам на сервері наступної мінімальної комплектації:

– Процесор: чотирьохядерний процесор з тактовою частотою не менше 2,0 Ггц.

– Оперативна пам’ять: 8 ГБ.

– Системний диск (HDD/SDD): 150 ГБ вільного дискового простору. Рекомендовано використовувати SSD для покращення швидкості завантаження та роботи застосунку.

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

Система повинна задовольняти вказаним вимогам на клієнтському комп’ютері наступної мінімальної комплектації:

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

4.5 Вимоги до інформаційної й програмної сумісності

На сервері повинна бути встановлена одна з операційних систем: Windows 11 не нижче версії 21H2, Linux Ubuntu не нижче версії 22.04.3 LTS, Linux Debian не нижче версії 12, Linux Fedora не нижче версії 38 або MacOS версії не нижче Monterey. На сервері повинен бути встановлений Docker Desktop не нижче версії 4.24.0.

На клієнтському комп’ютері повинен бути встановлений веббраузер з підтримкою HTML5.

4.6 Вимоги до захисту інформації

Вимоги до захисту інформації:

– забезпечити шифрування та безпеку передачі даних між клієнтом і сервером за допомогою протоколу HTTPS;

– усі паролі користувачів повинні бути збережені у зашифрованому вигляді.

4.7 Вимоги до програмної документації

У програмну документацію повинні входити наступні документи:

– технічне завдання;

– програма та методика випробувань;

– керівництво користувача;

– опис програми;

– код програми.

5 Техніко-економічні показники

Техніко-економічні показники не розраховуються.

6 Стадії й етапи розробки

Стадії й етапи розробки представлені у таблиці А.1.

Таблиця А.1 – Стадії й етапи розробки

Номер	Назва етапів роботи	Термін виконання
1	Аналіз практичної задачі інженерії ПЗ	07.04.2025
2	Аналіз вимог до ПЗ	14.04.2025
3	Розробка архітектури ПЗ	21.04.2025
4	Розробка проекту ПЗ	05.05.2025
5	Реалізація та тестування ПЗ	12.05.2025

7 Порядок контролю й приймання

Здійснюється контроль аналізу та проектування кожної окремої частини вебзастосунку на всіх етапах з урахуванням вимог, визначених у технічному завдані. Кожна стадія розробки повинна бути представлена в зазначені строки та узгоджена із замовником.

Приймання проводиться відповідно до програми і методики випробувань та документується за допомогою протоколу проведення випробувань. У разі знаходження помилок під час приймання програмного виробу складається акт про знайдені помилки, який підписуються представниками замовника і розробника і затверджуються керівником організації замовника та організації розробника.

Розробник повинен на протязі не більше ніж 2 тижнів виправити зазначені помилки і оповістити замовника про повторне проведення перевірки.

ДОДАТОК Б – ТЕКСТ ПРОГРАМИ

Лістинг 1

AppointmentService.java

```
package com.medicalcenter.receptionapi.service;

import com.medicalcenter.receptionapi.annotation.CheckDoctorAppointmentOwnership;
import com.medicalcenter.receptionapi.domain.*;
import com.medicalcenter.receptionapi.dto.appointment.AppointmentRequestDto;
import com.medicalcenter.receptionapi.dto.appointment.AppointmentResponseDto;
import com.medicalcenter.receptionapi.dto.appointment.TimeSlotDto;
import com.medicalcenter.receptionapi.dto.consultation.ConsultationResponseDto;
import com.medicalcenter.receptionapi.exception.*;
import com.medicalcenter.receptionapi.mapper.AppointmentMapper;
import com.medicalcenter.receptionapi.repository.AppointmentRepository;
import com.medicalcenter.receptionapi.repository.DoctorRepository;
import com.medicalcenter.receptionapi.repository.PatientRepository;
import com.medicalcenter.receptionapi.specification.AppointmentSpecification;
import com.medicalcenter.receptionapi.util.BeanCopyUtils;
import java.time.LocalDate;
import java.time.LocalTime;
import java.util.ArrayList;
import java.util.Comparator;
import java.util.List;
import lombok.AllArgsConstructor;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

@Service
@AllArgsConstructor
public class AppointmentService {
  private final WorkScheduleService workScheduleService;
  private final ConsultationService consultationService;
  private final AppointmentRepository appointmentRepository;
  private final PatientRepository patientRepository;
  private final DoctorRepository doctorRepository;
  private final AppointmentMapper appointmentMapper;

  @Cacheable(value = "appointments", key = "'count'")
  public long count() {
    return appointmentRepository.count();
  }

  @Cacheable(
      value = "appointments",
      key =
          "#patient + '_' + #doctor + '_'"
              + "+ (#date != null ? #date.toString() : 'null') + '_'"
              + "+ (#timeStart != null ? #timeStart.toString() : 'null')"
              + "+ '_' + #page + '_' + #pageSize")
  public Page<AppointmentResponseDto> findAllAppointments(
      String patient, String doctor, LocalDate date, LocalTime timeStart, int
```

```
page, int pageSize) {
    Specification<Appointment> specs = Specification.where(null);
    if (date != null) {
      specs = specs.and(AppointmentSpecification.withDate(date));
    }
    if (timeStart != null) {
      specs = specs.and(AppointmentSpecification.withTimeStart(timeStart));
    }
    if (patient != null && !patient.isBlank()) {
      specs = specs.and(AppointmentSpecification.withPatient(patient));
    }
    if (doctor != null && !doctor.isBlank()) {
      specs = specs.and(AppointmentSpecification.withDoctor(doctor));
    }
    Sort sort = Sort.by(Sort.Direction.DESC, "id");
    Pageable pageable = PageRequest.of(page, pageSize, sort);
    return appointmentRepository
        .findAll(specs, pageable)
        .map(appointmentMapper::appointmentToAppointmentResponseDto);
  }

  @Cacheable(value = "appointments", key = "#id")
  public AppointmentResponseDto findAppointmentById(Long id) {
    return appointmentRepository
        .findById(id)
        .map(appointmentMapper::appointmentToAppointmentResponseDto)
        .orElseThrow(ResourceNotFoundException::new);
  }

  @Cacheable(value = "timetable", key = "#doctorId + '_' + #date")
  @PreAuthorize("hasAnyRole('ADMIN', 'RECEPTIONIST') or #doctorId ==
authentication.principal.id")
  public List<TimeSlotDto> generateTimetable(Long doctorId, LocalDate date) {
    java.time.DayOfWeek dayOfWeek = date.getDayOfWeek();
    WorkSchedule workSchedule =
        workScheduleService.findWorkScheduleByDoctorAndDayOfWeek(doctorId,
dayOfWeek.getValue());
    LocalTime workTimeStart = workSchedule.getWorkTimeStart();
    LocalTime workTimeEnd = workSchedule.getWorkTimeEnd();
    if (workTimeStart == null || workTimeEnd == null) {
      throw new WorkScheduleDoesNotExistException();
    }
    int minutesStep = 15;
    List<Appointment> appointments = findAppointmentsByDoctorAndDate(doctorId,
date);
    return generateTimeSlots(date, workTimeStart, workTimeEnd, minutesStep,
appointments);
  }

  @PreAuthorize(
      "hasAnyRole('ADMIN', 'RECEPTIONIST') or"
          + "#appointmentRequestDto.doctorId == authentication.principal.id")
  @CacheEvict(
      value = {"appointments", "timetable"},
      allEntries = true)
  public AppointmentResponseDto saveAppointment(AppointmentRequestDto
appointmentRequestDto) {
    Doctor doctor =
        doctorRepository
            .findById(appointmentRequestDto.getDoctorId())
            .orElseThrow(
                () -> new ResourceNotFoundException("A doctor with that id doesn't
exist"));
    Patient patient =
```

```
        patientRepository
            .findById(appointmentRequestDto.getPatientId())
            .orElseThrow(
                () -> new ResourceNotFoundException("A patient with that id
doesn't exist"));
    if (!validateAppointmentTime(appointmentRequestDto)) {
      throw new IllegalArgumentException();
    }
    if (checkPatientConflict(appointmentRequestDto)) {
      throw new AppointmentPatientConflictException();
    }
    if (checkDoctorConflict(appointmentRequestDto)) {
      throw new AppointmentDoctorConflictException();
    }
    java.time.DayOfWeek dayOfWeek =
appointmentRequestDto.getDate().getDayOfWeek();
    WorkSchedule workSchedule =
        workScheduleService.findWorkScheduleByDoctorAndDayOfWeek(
            doctor.getId(), dayOfWeek.getValue());
    if (!isAppointmentWithinWorkSchedule(appointmentRequestDto, workSchedule)) {
      throw new InvalidAppointmentTimeException();
    }
    Appointment newAppointment =

appointmentMapper.appointmentRequestDtoToAppointment(appointmentRequestDto);
    newAppointment.setDoctor(doctor);
    newAppointment.setPatient(patient);
    newAppointment = appointmentRepository.save(newAppointment);
    return appointmentMapper.appointmentToAppointmentResponseDto(newAppointment);
  }

  @PreAuthorize(
      "hasAnyRole('ADMIN', 'RECEPTIONIST') or"
          + "#appointmentRequestDto.doctorId == authentication.principal.id")
  @CacheEvict(
      value = {"appointments", "timetable"},
      allEntries = true)
  @CheckDoctorAppointmentOwnership
  public AppointmentResponseDto updateAppointment(
      AppointmentRequestDto appointmentRequestDto, Long appointmentId) {
    Appointment appointmentToUpdate =

appointmentRepository.findById(appointmentId).orElseThrow(ResourceNotFoundExceptio
n::new);
    Doctor doctor =
        doctorRepository
            .findById(appointmentRequestDto.getDoctorId())
            .orElseThrow(
                () -> new ResourceNotFoundException("A doctor with that id doesn't
exist"));
    if (!validateAppointmentTime(appointmentRequestDto)) {
      throw new IllegalArgumentException();
    }
    if (checkDoctorConflict(appointmentRequestDto, appointmentId)) {
      throw new AppointmentDoctorConflictException();
    }
    if (checkPatientConflict(appointmentRequestDto, appointmentId)) {
      throw new AppointmentPatientConflictException();
    }
    java.time.DayOfWeek dayOfWeek =
appointmentRequestDto.getDate().getDayOfWeek();
    WorkSchedule workSchedule =
        workScheduleService.findWorkScheduleByDoctorAndDayOfWeek(
            doctor.getId(), dayOfWeek.getValue());
```

```
    if (!isAppointmentWithinWorkSchedule(appointmentRequestDto, workSchedule)) {
      throw new InvalidAppointmentTimeException();
    }
    Appointment appointmentUpdateRequest =

appointmentMapper.appointmentRequestDtoToAppointment(appointmentRequestDto);
    appointmentUpdateRequest.setDoctor(doctor);
    BeanCopyUtils.copyNonNullProperties(
        appointmentUpdateRequest, appointmentToUpdate, "id", "patient");
    Appointment updatedAppointment =
appointmentRepository.save(appointmentToUpdate);
    return
appointmentMapper.appointmentToAppointmentResponseDto(updatedAppointment);
  }

  @CacheEvict(
      value = {"appointments", "timetable"},
      allEntries = true)
  @CheckDoctorAppointmentOwnership
  public void deleteAppointment(Long appointmentId) {
    Appointment appointmentToDelete =
        appointmentRepository
            .findById(appointmentId)
            .orElseThrow(
                () -> new ResourceNotFoundException("Appointment with such id is
not found"));
    appointmentRepository.delete(appointmentToDelete);
  }

  public ConsultationResponseDto getAppointmentConsultation(Long appointmentId) {
    return consultationService.findConsultationById(appointmentId);
  }

  public List<Appointment> findAppointmentsByDoctorAndDate(Long doctorId,
LocalDate date) {
    return appointmentRepository.findByDoctor_IdAndDate(doctorId, date);
  }

  private boolean isAppointmentWithinWorkSchedule(
      AppointmentRequestDto appointmentRequestDto, WorkSchedule workSchedule) {
    if (workSchedule.getWorkTimeStart() == null || workSchedule.getWorkTimeEnd()
== null) {
      return false;
    }
    return
!appointmentRequestDto.getTimeStart().isBefore(workSchedule.getWorkTimeStart())
        &&
!appointmentRequestDto.getTimeStart().isAfter(workSchedule.getWorkTimeEnd())
        &&
!appointmentRequestDto.getTimeEnd().isAfter(workSchedule.getWorkTimeEnd())
        &&
!appointmentRequestDto.getTimeEnd().isBefore(workSchedule.getWorkTimeStart());
  }

  private boolean validateAppointmentTime(AppointmentRequestDto
appointmentRequestDto) {
    return
appointmentRequestDto.getTimeStart().isBefore(appointmentRequestDto.getTimeEnd());
  }

  private boolean checkPatientConflict(AppointmentRequestDto
appointmentRequestDto) {
    return appointmentRepository
```

```
.existsByPatient_IdAndDateAndTimeStartLessThanEqualAndTimeEndGreaterThanEqual(
            appointmentRequestDto.getPatientId(),
            appointmentRequestDto.getDate(),
            appointmentRequestDto.getTimeEnd().minusMinutes(1),
            appointmentRequestDto.getTimeStart().plusMinutes(1));
  }

  private boolean checkPatientConflict(
      AppointmentRequestDto appointmentRequestDto, Long appointmentId) {
    return appointmentRepository

.existsByPatient_IdAndDateAndTimeStartLessThanEqualAndTimeEndGreaterThanEqualAndId
Not(
            appointmentRequestDto.getPatientId(),
            appointmentRequestDto.getDate(),
            appointmentRequestDto.getTimeEnd().minusMinutes(1),
            appointmentRequestDto.getTimeStart().plusMinutes(1),
            appointmentId);
  }

  private boolean checkDoctorConflict(AppointmentRequestDto appointmentRequestDto)
{
    return appointmentRepository

.existsByDoctor_IdAndDateAndTimeStartLessThanEqualAndTimeEndGreaterThanEqual(
            appointmentRequestDto.getDoctorId(),
            appointmentRequestDto.getDate(),
            appointmentRequestDto.getTimeEnd().minusMinutes(1),
            appointmentRequestDto.getTimeStart().plusMinutes(1));
  }

  private boolean checkDoctorConflict(
      AppointmentRequestDto appointmentRequestDto, Long appointmentId) {
    return appointmentRepository

.existsByDoctor_IdAndDateAndTimeStartLessThanEqualAndTimeEndGreaterThanEqualAndIdN
ot(
            appointmentRequestDto.getDoctorId(),
            appointmentRequestDto.getDate(),
            appointmentRequestDto.getTimeEnd().minusMinutes(1),
            appointmentRequestDto.getTimeStart().plusMinutes(1),
            appointmentId);
  }

  private boolean isAppointmentOverlapping(
      Appointment appointment, LocalTime startTime, LocalTime endTime) {
    return (appointment.getTimeStart().isBefore(endTime)
            && appointment.getTimeEnd().isAfter(startTime))
        || appointment.getTimeEnd().equals(startTime);
  }

  private TimeSlotDto createTimeSlot(
      LocalDate date, LocalTime startTime, LocalTime endTime, List<Appointment>
appointments) {
    TimeSlotDto timeSlot = new TimeSlotDto();
    timeSlot.setDate(date);
    timeSlot.setStartTime(startTime);
    timeSlot.setEndTime(endTime);
    List<AppointmentResponseDto> slotAppointments = new ArrayList<>();
    for (Appointment appointment : appointments) {
      if (isAppointmentOverlapping(appointment, startTime, endTime)) {

slotAppointments.add(appointmentMapper.appointmentToAppointmentResponseDto(appoint
ment));
```

```
      }
    }

slotAppointments.sort(Comparator.comparing(AppointmentResponseDto::getTimeStart));
    timeSlot.setAppointments(slotAppointments);
    return timeSlot;
  }

  private List<TimeSlotDto> generateTimeSlots(
      LocalDate date,
      LocalTime workTimeStart,
      LocalTime workTimeEnd,
      int minutesStep,
      List<Appointment> appointments) {
    List<TimeSlotDto> timeSlots = new ArrayList<>();
    LocalTime currentTime = workTimeStart;
    while (currentTime.isBefore(workTimeEnd)) {
      LocalTime slotEndTime = currentTime.plusMinutes(minutesStep);
      if (slotEndTime.isAfter(workTimeEnd)) {
        slotEndTime = workTimeEnd;
      }
      TimeSlotDto timeSlot = createTimeSlot(date, currentTime, slotEndTime,
appointments);
      timeSlots.add(timeSlot);
      currentTime = currentTime.plusMinutes(minutesStep);
    }
    return timeSlots;
  }
}
```

Лістинг 2

DoctorService.java

```
package com.medicalcenter.receptionapi.service;

import com.medicalcenter.receptionapi.domain.Doctor;
import com.medicalcenter.receptionapi.domain.Office;
import com.medicalcenter.receptionapi.dto.doctor.DoctorRequestDto;
import com.medicalcenter.receptionapi.dto.doctor.DoctorResponseDto;
import
com.medicalcenter.receptionapi.dto.doctor.DoctorResponseWithUserCredentialsDto;
import com.medicalcenter.receptionapi.dto.user.RegisterRequestDto;
import com.medicalcenter.receptionapi.dto.user.UserCredentialsDto;
import com.medicalcenter.receptionapi.dto.workschedule.WorkScheduleResponseDto;
import com.medicalcenter.receptionapi.exception.ResourceNotFoundException;
import com.medicalcenter.receptionapi.mapper.DoctorMapper;
import com.medicalcenter.receptionapi.repository.DoctorRepository;
import com.medicalcenter.receptionapi.repository.OfficeRepository;
import com.medicalcenter.receptionapi.security.enums.RoleAuthority;
import com.medicalcenter.receptionapi.specification.DoctorSpecification;
import java.time.LocalDate;
import java.util.Optional;
import lombok.AllArgsConstructor;
import org.springframework.beans.BeanUtils;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

@Service
```

```
@AllArgsConstructor
public class DoctorService {

  private final DoctorRepository doctorRepository;
  private final OfficeRepository officeRepository;
  private final WorkScheduleService workScheduleService;
  private final UserService userService;
  private final DoctorMapper doctorMapper;

  @Cacheable(value = "doctors", key = "'count'")
  public long count() {
    return doctorRepository.count();
  }

  @Cacheable(
      value = "doctors",
      key =
          "#surname + '_' + #name + '_' + #middleName + '_'"
              + "+ (#birthDate != null ? #birthDate.toString() : 'null' ) + '_'"
              + "+ #medicalSpecialty + '_' + #officeNumber + '_' + #page + '_' +
#pageSize")
  public Page<DoctorResponseDto> findAllDoctors(
      String surname,
      String name,
      String middleName,
      LocalDate birthDate,
      String medicalSpecialty,
      Integer officeNumber,
      int page,
      int pageSize) {
    Specification<Doctor> specs = Specification.where(null);
    if (surname != null && !surname.isBlank()) {
      specs = specs.and(DoctorSpecification.withSurname(surname));
    }
    if (name != null && !name.isBlank()) {
      specs = specs.and(DoctorSpecification.withName(name));
    }
    if (middleName != null && !middleName.isBlank()) {
      specs = specs.and(DoctorSpecification.withMiddleName(middleName));
    }
    if (medicalSpecialty != null && !medicalSpecialty.isBlank()) {
      specs =
specs.and(DoctorSpecification.withMedicalSpeciality(medicalSpecialty));
    }
    if (birthDate != null) {
      specs = specs.and(DoctorSpecification.withBirthDate(birthDate));
    }
    if (officeNumber != null) {
      specs = specs.and(DoctorSpecification.withOffice(officeNumber));
    }
    Sort sort = Sort.by(Sort.Direction.DESC, "id");
    Pageable pageable = PageRequest.of(page, pageSize, sort);
    return doctorRepository.findAll(specs,
pageable).map(doctorMapper::doctorToDoctorResponseDto);
  }

  @PreAuthorize("hasAnyRole('ADMIN', 'RECEPTIONIST') or #id ==
authentication.principal.id")
  @Cacheable(value = "doctors", key = "#id")
  public DoctorResponseDto findDoctorById(Long id) {
    return doctorRepository
        .findById(id)
        .map(doctorMapper::doctorToDoctorResponseDto)
        .orElseThrow(ResourceNotFoundException::new);
```

```
  }

  @PreAuthorize("hasAnyRole('ADMIN', 'RECEPTIONIST') or #doctorId ==
authentication.principal.id")
  public Page<WorkScheduleResponseDto> findDoctorWorkSchedules(
      int page, int pageSize, Long doctorId) {

doctorRepository.findById(doctorId).orElseThrow(ResourceNotFoundException::new);
    return workScheduleService.findWorkSchedulesByDoctorId(page, pageSize,
doctorId);
  }

  @CacheEvict(value = "doctors", allEntries = true)
  public DoctorResponseWithUserCredentialsDto saveDoctor(DoctorRequestDto
doctorRequestDto) {
    Doctor doctor = doctorMapper.doctorRequestDtoToDoctor(doctorRequestDto);
    if (doctorRequestDto.getOfficeId() != null) {
      Optional<Office> officeOptional =
officeRepository.findById(doctorRequestDto.getOfficeId());
      if (officeOptional.isPresent()) {
        Office office = officeOptional.get();
        doctor.setOffice(office);
      }
    }
    UserCredentialsDto userCredentialsDto =
userService.generateRandomCredentials();
    RegisterRequestDto registerRequestDto =
        RegisterRequestDto.builder()
            .userCredentialsDto(userCredentialsDto)
            .role(RoleAuthority.DOCTOR)
            .build();
    userService.saveUser(registerRequestDto, doctor);
    doctor = doctorRepository.save(doctor);
    return DoctorResponseWithUserCredentialsDto.builder()
        .doctorResponseDto(doctorMapper.doctorToDoctorResponseDto(doctor))
        .userCredentialsDto(userCredentialsDto)
        .build();
  }

  @CacheEvict(value = "doctors", allEntries = true)
  public DoctorResponseDto updateDoctor(DoctorRequestDto doctorRequestDto, Long
id) {
    Doctor doctorToUpdate =
        doctorRepository.findById(id).orElseThrow(ResourceNotFoundException::new);
    Doctor updateRequestDoctor =
doctorMapper.doctorRequestDtoToDoctor(doctorRequestDto);
    BeanUtils.copyProperties(
        updateRequestDoctor,
        doctorToUpdate,
        "id",
        "appointments",
        "office",
        "user",
        "workSchedules");
    if (doctorRequestDto.getOfficeId() != null) {
      Optional<Office> officeOptional =
officeRepository.findById(doctorRequestDto.getOfficeId());
      if (officeOptional.isPresent()) {
        Office office = officeOptional.get();
        doctorToUpdate.setOffice(office);
      }
    } else {
      doctorToUpdate.setOffice(null);
    }
```

```
    Doctor updatedDoctor = doctorRepository.save(doctorToUpdate);
    return doctorMapper.doctorToDoctorResponseDto(updatedDoctor);
  }

  @CacheEvict(value = "doctors", allEntries = true)
  public void deleteDoctor(Long id) {
    doctorRepository.deleteById(id);
  }
}
```

Лістинг 3

PatientService.java

```
package com.medicalcenter.receptionapi.service;

import com.medicalcenter.receptionapi.domain.Patient;
import com.medicalcenter.receptionapi.dto.patient.PatientRequestDto;
import com.medicalcenter.receptionapi.dto.patient.PatientResponseDto;
import com.medicalcenter.receptionapi.exception.ResourceNotFoundException;
import com.medicalcenter.receptionapi.mapper.PatientMapper;
import com.medicalcenter.receptionapi.repository.PatientRepository;
import com.medicalcenter.receptionapi.specification.PatientSpecification;
import java.time.LocalDate;
import lombok.AllArgsConstructor;
import org.springframework.beans.BeanUtils;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.stereotype.Service;

@Service
@AllArgsConstructor
public class PatientService {

  private final PatientRepository patientRepository;
  private final PatientMapper patientMapper;

  @Cacheable(value = "patients", key = "'count'")
  public long count() {
    return patientRepository.count();
  }

  @Cacheable(
      value = "patients",
      key =
          "#surname + '_' + #name + '_' + #middleName + '_'"
              + "+ (#birthDate != null ? #birthDate.toString() : 'null' )"
              + "+ '_' + #page + '_' + #pageSize")
  public Page<PatientResponseDto> findAllPatients(
      String surname, String name, String middleName, LocalDate birthDate, int
page, int pageSize) {

    Specification<Patient> spec = Specification.where(null);
    if (surname != null && !surname.isBlank()) {
      spec = spec.and(PatientSpecification.withSurname(surname));
    }
    if (name != null && !name.isBlank()) {
      spec = spec.and(PatientSpecification.withName(name));
    }
    if (middleName != null && !middleName.isBlank()) {
      spec = spec.and(PatientSpecification.withMiddleName(middleName));
```

```
    }
    if (birthDate != null) {
      spec = spec.and(PatientSpecification.withBirthDate(birthDate));
    }
    Sort sort = Sort.by(Sort.Direction.DESC, "id");
    Pageable pageable = PageRequest.of(page, pageSize, sort);
    return patientRepository
        .findAll(spec, pageable)
        .map(patientMapper::patientToPatientResponseDto);
  }

  @Cacheable(value = "patients", key = "#id")
  public PatientResponseDto findPatientById(Long id) {
    Patient patient =
patientRepository.findById(id).orElseThrow(ResourceNotFoundException::new);
    return patientMapper.patientToPatientResponseDto(patient);
  }

  @CacheEvict(value = "patients", allEntries = true)
  public PatientResponseDto savePatient(PatientRequestDto patientRequestDto) {
    Patient patient =

patientRepository.save(patientMapper.patientRequestDtoToPatient(patientRequestDto)
);
    return patientMapper.patientToPatientResponseDto(patient);
  }

  @CacheEvict(value = "patients", allEntries = true)
  public PatientResponseDto updatePatient(PatientRequestDto patientRequestDto,
Long id) {
    Patient patientToUpdate =

patientRepository.findById(id).orElseThrow(ResourceNotFoundException::new);
    BeanUtils.copyProperties(patientRequestDto, patientToUpdate, "id",
"appointments");
    Patient patient = patientRepository.save(patientToUpdate);
    return patientMapper.patientToPatientResponseDto(patient);
  }

  @CacheEvict(value = "patients", allEntries = true)
  public void deletePatient(Long id) {
    patientRepository.deleteById(id);
  }
}
```

Лістинг 4

ReceptionistService.java

```
package com.medicalcenter.receptionapi.service;

import com.medicalcenter.receptionapi.domain.Receptionist;
import com.medicalcenter.receptionapi.dto.receptionist.ReceptionistRequestDto;
import com.medicalcenter.receptionapi.dto.receptionist.ReceptionistResponseDto;
import
com.medicalcenter.receptionapi.dto.receptionist.ReceptionistResponseWithUserCreden
tialsDto;
import com.medicalcenter.receptionapi.dto.user.RegisterRequestDto;
import com.medicalcenter.receptionapi.dto.user.UserCredentialsDto;
import com.medicalcenter.receptionapi.exception.ResourceNotFoundException;
import com.medicalcenter.receptionapi.mapper.ReceptionistMapper;
import com.medicalcenter.receptionapi.repository.ReceptionistRepository;
import com.medicalcenter.receptionapi.security.enums.RoleAuthority;
import com.medicalcenter.receptionapi.specification.ReceptionistSpecification;
import java.time.LocalDate;
import lombok.AllArgsConstructor;
```

```
import org.springframework.beans.BeanUtils;
import org.springframework.cache.annotation.CacheEvict;
import org.springframework.cache.annotation.Cacheable;
import org.springframework.data.domain.Page;
import org.springframework.data.domain.PageRequest;
import org.springframework.data.domain.Pageable;
import org.springframework.data.domain.Sort;
import org.springframework.data.jpa.domain.Specification;
import org.springframework.security.access.prepost.PreAuthorize;
import org.springframework.stereotype.Service;

@Service
@AllArgsConstructor
public class ReceptionistService {

  private final ReceptionistRepository receptionistRepository;
  private final UserService userService;
  private final ReceptionistMapper receptionistMapper;

  @Cacheable(value = "receptionists", key = "'count'")
  public long count() {
    return receptionistRepository.count();
  }

  @Cacheable(
      value = "receptionists",
      key =
          "#surname + '_' + #name + '_' + #middleName + '_'"
              + "+ (#birthDate != null ? #birthDate.toString() : 'null' ) "
              + "+ '_' + '_'  + #page + '_' + #pageSize")
  public Page<ReceptionistResponseDto> findAllReceptionists(
      String surname, String name, String middleName, LocalDate birthDate, int
page, int pageSize) {
    Specification<Receptionist> specs = Specification.where(null);
    if (surname != null && !surname.isBlank()) {
      specs = specs.and(ReceptionistSpecification.withSurname(surname));
    }
    if (name != null && !name.isBlank()) {
      specs = specs.and(ReceptionistSpecification.withName(name));
    }
    if (middleName != null && !middleName.isBlank()) {
      specs = specs.and(ReceptionistSpecification.withMiddleName(middleName));
    }
    if (birthDate != null) {
      specs = specs.and(ReceptionistSpecification.withBirthDate(birthDate));
    }
    Sort sort = Sort.by(Sort.Direction.DESC, "id");
    Pageable pageable = PageRequest.of(page, pageSize, sort);
    return receptionistRepository
        .findAll(specs, pageable)
        .map(receptionistMapper::receptionistToReceptionistResponseDto);
  }

  @Cacheable(value = "receptionists", key = "#id")
  @PreAuthorize("hasAnyRole('ADMIN') or #id == authentication.principal.id")
  public ReceptionistResponseDto findReceptionistById(Long id) {
    return receptionistRepository
        .findById(id)
        .map(receptionistMapper::receptionistToReceptionistResponseDto)
        .orElseThrow(ResourceNotFoundException::new);
  }

  @CacheEvict(value = "receptionists", allEntries = true)
  public ReceptionistResponseWithUserCredentialsDto saveReceptionist(
```

```
      ReceptionistRequestDto receptionistRequestDto) {
    Receptionist receptionist =

receptionistMapper.receptionistRequestDtoToReceptionist(receptionistRequestDto);
    UserCredentialsDto userCredentialsDto =
userService.generateRandomCredentials();
    RegisterRequestDto registerRequestDto =
        RegisterRequestDto.builder()
            .userCredentialsDto(userCredentialsDto)
            .role(RoleAuthority.RECEPTIONIST)
            .build();
    userService.saveUser(registerRequestDto, receptionist);
    receptionist = receptionistRepository.save(receptionist);
    return ReceptionistResponseWithUserCredentialsDto.builder()
        .receptionistResponseDto(

receptionistMapper.receptionistToReceptionistResponseDto(receptionist))
        .userCredentialsDto(userCredentialsDto)
        .build();
  }

  @CacheEvict(value = "receptionists", allEntries = true)
  public ReceptionistResponseDto updateReceptionist(
      ReceptionistRequestDto receptionistRequestDto, Long id) {
    Receptionist updateRequestReceptionist =

receptionistMapper.receptionistRequestDtoToReceptionist(receptionistRequestDto);
    Receptionist receptionistToUpdate =

receptionistRepository.findById(id).orElseThrow(ResourceNotFoundException::new);
    BeanUtils.copyProperties(updateRequestReceptionist, receptionistToUpdate,
"id");
    Receptionist updatedReceptionist =
receptionistRepository.save(receptionistToUpdate);
    return
receptionistMapper.receptionistToReceptionistResponseDto(updatedReceptionist);
  }

  @CacheEvict(value = "receptionists", allEntries = true)
  public void deleteReceptionist(Long id) {
    receptionistRepository.deleteById(id);
  }
}
```

ДОДАТОК В – ОПИС ПРОГРАМИ

1 Загальні відомості

1.1 Позначення і найменування програми

Найменування: Вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed".

Позначення: вебзастосунок.

1.2 Програмне забезпечення, необхідне для функціонування програми

На сервері повинна бути встановлена одна з операційних систем: Windows 11 не нижче версії 21H2, Linux Ubuntu не нижче версії 22.04.3 LTS, Linux Debian не нижче версії 12, Linux Fedora не нижче версії 38 або MacOS версії не нижче Monterey. На сервері повинен бути встановлений Docker Desktop не нижче версії 4.24.0.

На клієнтському комп’ютері повинен бути встановлений веббраузер з підтримкою HTML5.

1.3 Мови програмування

Серверну частину вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" було розроблено за допомогою мови програмування Java та фреймворку Spring Boot. Для збереження даних було використано бази даних MariaDB та Redis.

Для розробки клієнтської частини застосунку було використано фреймворк Next.js та бібліотеку для розробки користувацьких інтерфейсів Bootstrap.

2 Функціональне призначення

2.1 Класи вирішуваних завдань

Вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed" є застосунком спрямованим на автоматизацію процесів, пов’язаних із роботою реєстратури в медичній клініці “CompliMed”. Вебзастосунок охоплює низку процесів необхідних для повноцінного функціонування реєстратурного

відділу медичного закладу: зберігання та обробка даних про пацієнтів, лікарів, прийомів пацієнтів до лікарів та кабінетів медичної клініки. Вебзастосунок надає можливість виконання нижче перерахованих функцій.

Управління даними про пацієнтів

Перегляд, додавання, редагування та видалення інформації про пацієнтів.

Управління даними про лікарів

Перегляд, додавання, редагування та видалення інформації про лікарів.

Перегляд та управління графіками роботи лікарів, включаючи можливість додавання, коригування та видалення графіків роботи на кожний день тижня.

Перегляд розкладу роботи лікарів згідно графіків роботи.

Управління даними про реєстраторів

Перегляд, додавання, редагування та видалення інформації про реєстраторів.

Управління даними про кабінети

Перегляд, додавання, редагування та видалення інформації про кабінети медичної клініки.

Управління даними про прийоми пацієнтів

Перегляд, призначення, коригування та скасування прийомів пацієнтів у відповідності з графіками роботи лікарів.

Внесення та перегляд інформації про консультації пацієнтів в прийомах, включаючи симптоми, діагнози та медичні рекомендації

Управління даними про облікові записи

Створення та видалення облікових записів користувачів: лікарів та реєстраторів

2.2 Відомості про функціональні обмеження на застосування

Головним функціональним обмеженням вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" є потреба підключення до мережі Інтернет.

3 Опис логічної структури

3.1 Структура програми

Основою структури вебзастосунку для автоматизації роботи реєстратури медичної клініки "CompliMed" є патерн проєктування Model-View-Controller (MVC). Програма поділяється на 3 взаємопов'язані частини:

– модель – керує даними та бізнес-логікою;

– подання – керує компонуванням та відображенням даних з моделі, надсилає вхідні дані до контролера;

– контролер – обробляє вхідні дані користувача та відповідно спрямовує команди до частин моделі і подання.

Вебзастосунок побудовано на базі слабкопов'язаних модулів, де кожен модуль відповідає за чітко відведену йому мету.

3.2 Алгоритми програми

Вебзастосунок розділений на дві частини: клієнтську та серверну. Клієнтом є браузер користувача, а сервером може бути один або декілька комп'ютерів, які надають послуги клієнту та на яких розгорнуто сам вебзастосунок і необхідні для його роботи інші програми, такі як бази даних MariaDB та Redis.

Типовий потік взаємодії клієнта з сервером виглядає наступним чином:

1) Користувач, знаходячись на сайті вебзастосунку робить запити на сервер через клієнтський застосунок.

2) Сервер під управлінням Apache Tomcat обробляє запит клієнта та передає його далі для подальшої обробки.

3) Сервер обробляє запит клієнта, проводить необхідні розрахунки для задоволення запиту клієнта. Якщо необхідно, сервер звертається до відповідної бази даних для отримання деяких даних.

4) База даних повертає запитувані дані назад на сервер.

5) Сервер обробляє дані та надсилає їх назад клієнту.

3.3 Використовувані методи

Вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed" розроблено на мові Java, яка є об'єктно-орієнтованою мовою

програмування. Розробка вебзастосунку ґрунтується на об'єктно-орієнтованій методології.

4 Необхідні технічні засоби

Система повинна задовольняти вказаним вимогам на сервері наступної мінімальної комплектації:

– Процесор: чотирьохядерний процесор з тактовою частотою не менше 2,0 Ггц.

– Оперативна пам’ять: 8 ГБ.

– Системний диск (HDD/SDD): 150 ГБ вільного дискового простору. Рекомендовано використовувати SSD для покращення швидкості завантаження та роботи застосунку.

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

Система повинна задовольняти вказаним вимогам на клієнтському комп’ютері наступної мінімальної комплектації:

– Мережеві можливості: мережевий адаптер з підтримкою Ethernet.

5 Виклик і завантаження

Вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed" розгортається на сервері закладу. Для запуску вебзастосунку використовується контейнеризація образу вебзастосунку за допомогою Docker, або безпосереднє виконання .jar архіву застосунку з інтегрованим в нього вебсервером Apache Tomcat за допомогою віртуальної машини Java.

Після запуску, застосунок стає доступним через веббраузер за сконфігурованою при запуску адресою.

6 Вхідні і вихідні дані

Вхідні дані вводяться користувачем за допомогою пристроїв введення даних (клавіатури або миші) у спеціально відведенні форми вручну, або обираються із наданих списків.

Виведення даних відбувається за допомогою пристроїв виведення у відведених до них графічних компонентах вебсторінок.

ДОДАТОК Г – ІНСТРУКЦІЯ КОРИСТУВАЧА

Г.1. Початок роботи з програмою

Для початку роботи необхідно перейти за адресою вебзастосунку у браузері. Адреса за замовчуванням: localhost:3001. При переході за адресою з'являється вікно доступу до застосунку. Далі необхідно пройти авторизацію, для чого слід ввести логін та пароль від облікового запису (рисунок Г.1).

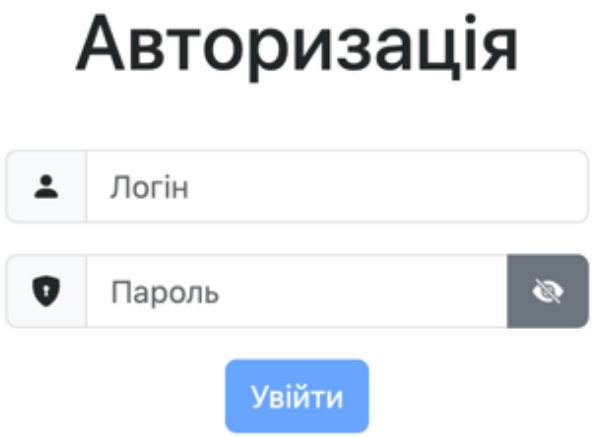


Рисунок Г.1 – Авторизація

При успішній авторизації відкриється головна сторінка застосунку (рисунок Г.2). Головна сторінка містить повідомлення вітання з ім'ям та роллю користувача. Доступ до подальшого функціоналу застосунку здійснюється через вкладки відповідних підсистем у навігаційному меню: Пацієнти, Лікарі, Прийоми, Кабінети, Реєстратори.

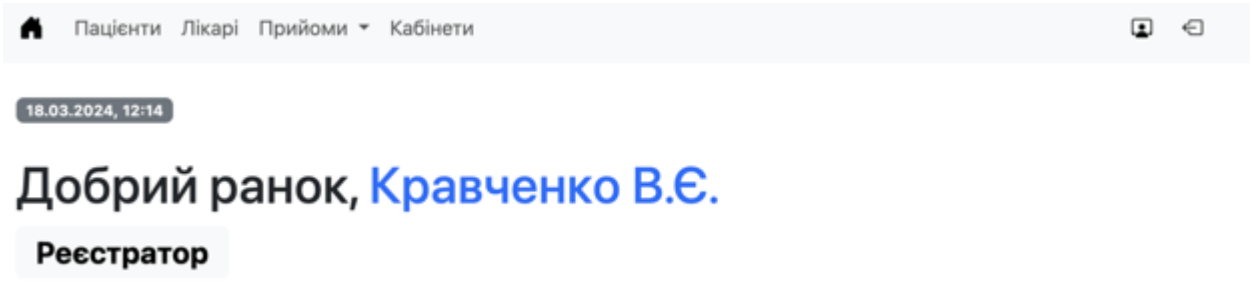


Рисунок Г.2 – Головна сторінка вебзастосунку

На всіх сторінках присутнє навігаційне меню у верхній частині екрану, за допомогою нього можна пересуватися між розділами застосунку.

Наступні сторінки будуть зображені від користувача з роллю “Адміністратор” для надання вигляду всіх сторінок, так як “Реєстратор” та “Лікар” мають обмежений доступ.

А.2. Загальні відомості

Сторінки “Пацієнти”, “Лікарі”, “Прийоми”, “Кабінети”, “Реєстратори” представляють собою сторінки зі списками відповідних сутностей та можливістю пошуку. Приклад такої сторінки представлено сторінкою "Пацієнти" на рисунку Г.3. Інші сторінки виглядають аналогічно.

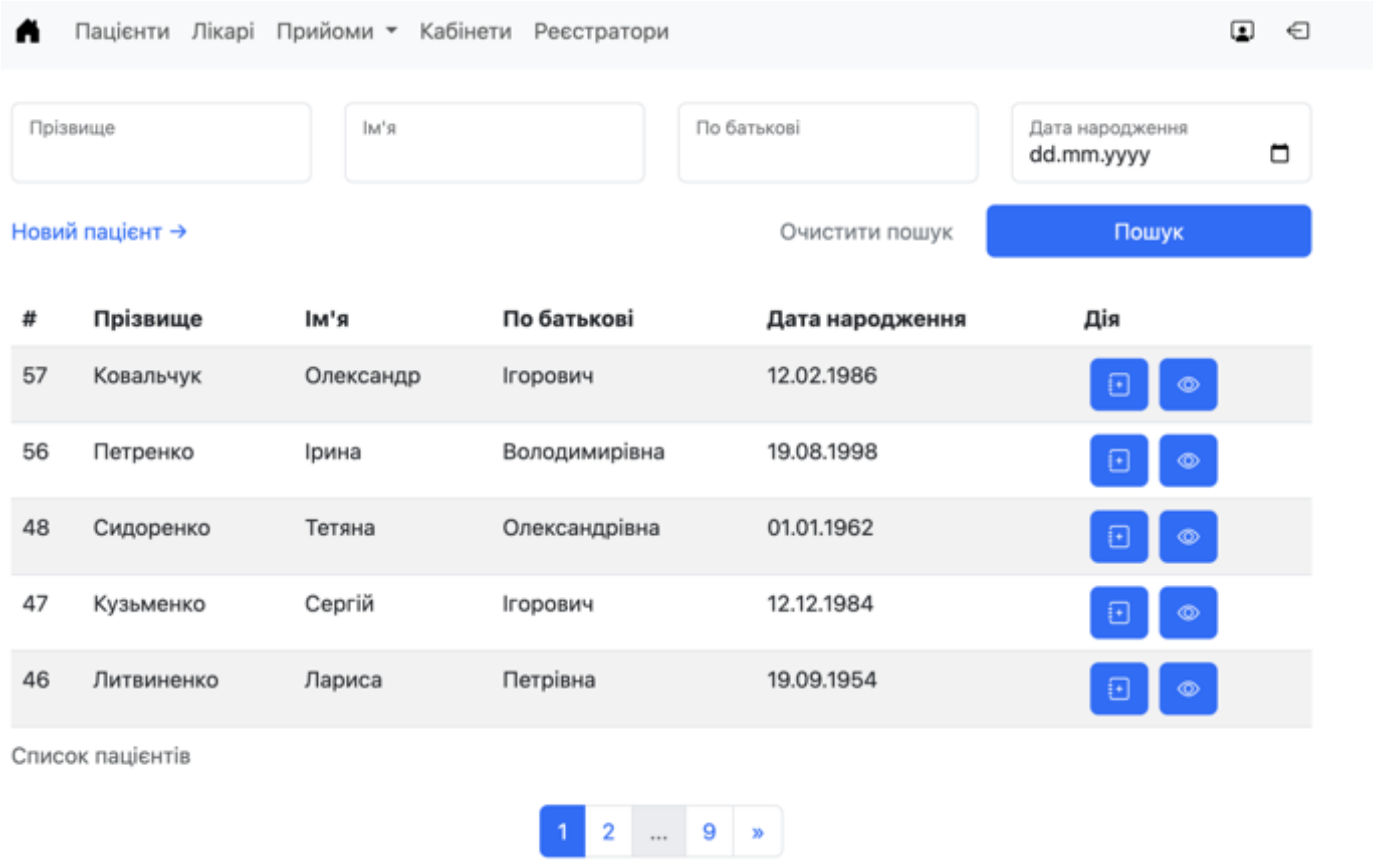


Рисунок Г.3 – Сторінка “Пацієнти”

Під списком надано кнопки пагінації для перемикання сторінок списку, на одній сторінці надано 5 сутностей.

Г.3. Пошук об'єктів у списку

Для виконання пошуку певного об'єкту у списку за атрибутами слід ввести інформацію в надані поля для пошуку та натиснути на кнопку керування “Пошук”. Як приклад, буде взято пошук лікарів на сторінці “Лікарі”.

Над списком лікарів надано доступні поля для пошуку. Після введення бажаних даних у поля слід натиснути кнопку “Пошук”, а для очищення параметрів пошуку слід натиснути на кнопку “Очистити пошук”. Форму пошуку представлено на рисунку Г.4.

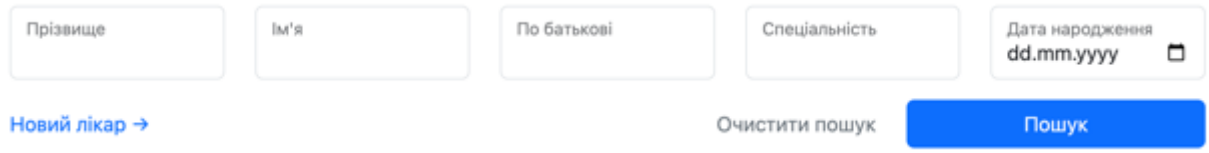


Рисунок Г.4 – Форма пошуку лікарів

Після натискання кнопки "Пошук" список лікарів буде оновлено відповідно до критеріїв пошуку.

Г.4. Додавання нової інформації про об'єкти

Зі сторінок “Пацієнти”, “Лікарі”, “Кабінети”, “Реєстратори” можна перейти до відповідних сторінок додавання нової інформації. Наприклад, для додавання нового пацієнта слід натиснути на кнопку “Новий пацієнт” (рисунок Г.5), розташовану зліва від кнопок керування пошуку на сторінці “Пацієнти”.

Рисунок Г.5 – Кнопка для переходу на сторінку додавання нового пацієнта.

Після натискання кнопки відбувається перенаправлення на сторінку додавання нового пацієнта. Сторінку “Новий пацієнт” зображено на рисунку Г.6.

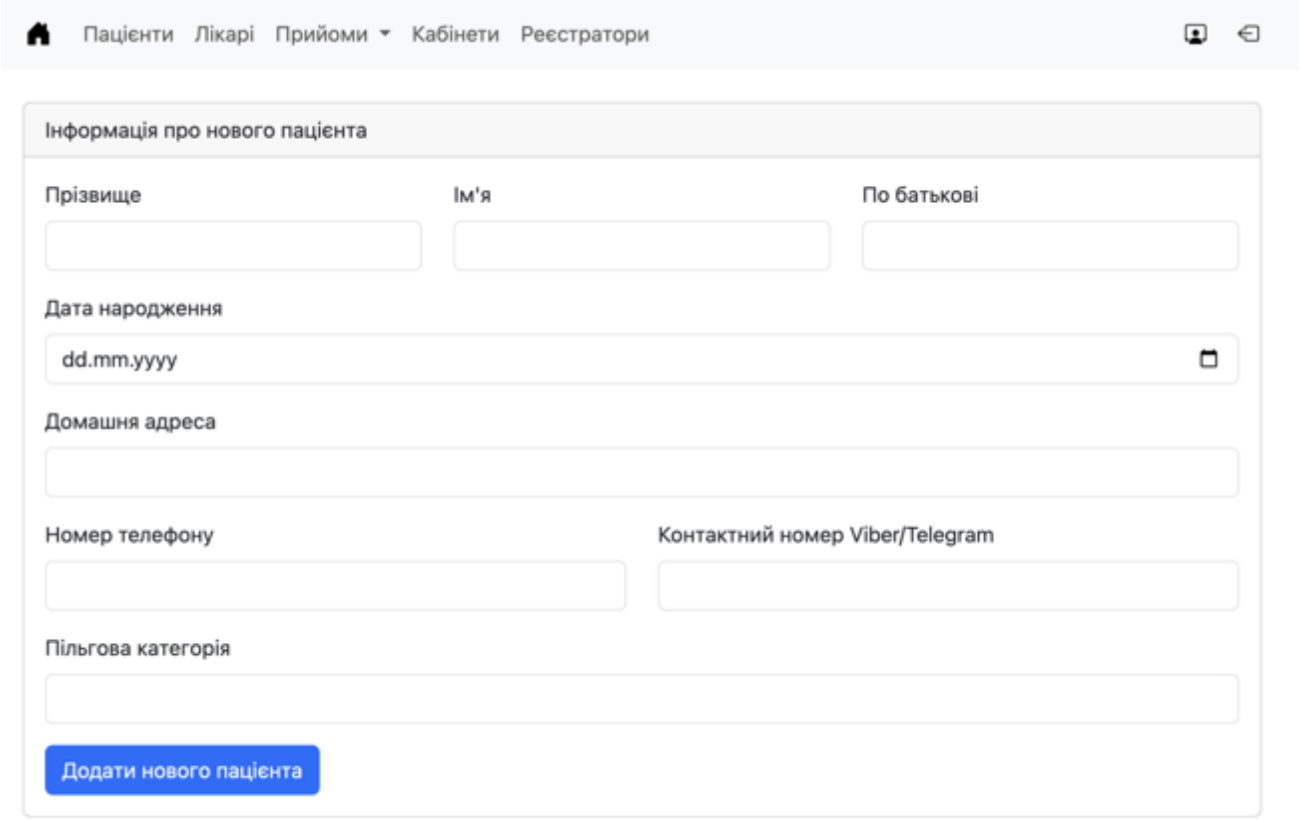


Рисунок Г.6 – Сторінка “Новий пацієнт”

Ввівши відповідну інформації до полів форми та натиснувши кнопку керування “Додати нового пацієнта”, можна внести нового пацієнта в систему. Після додавання нового пацієнта він з’явиться у списку на сторінці “Пацієнти”.

Додавання нової інформації про лікарів, реєстраторів та кабінети відбувається аналогічно до роботи з пацієнтами.

Г.5 Створення облікових записів лікарів та реєстраторів

При додаванні нового лікаря або реєстратора у застосунок для нього створюється користувацький обліковий запис з відповідними правами доступу. Розглянемо приклад додавання нового лікаря. Для додавання нового лікаря слід заповнити інформацію про нового лікаря на сторінці “Новий лікар”, та натиснути на кнопку “Додати нового лікаря”. Після підтвердження додавання нового лікаря, якщо не виникло помилок відкриється вікно з обліковими даними до новоствореного облікового запису лікаря (рисунок Г.7).

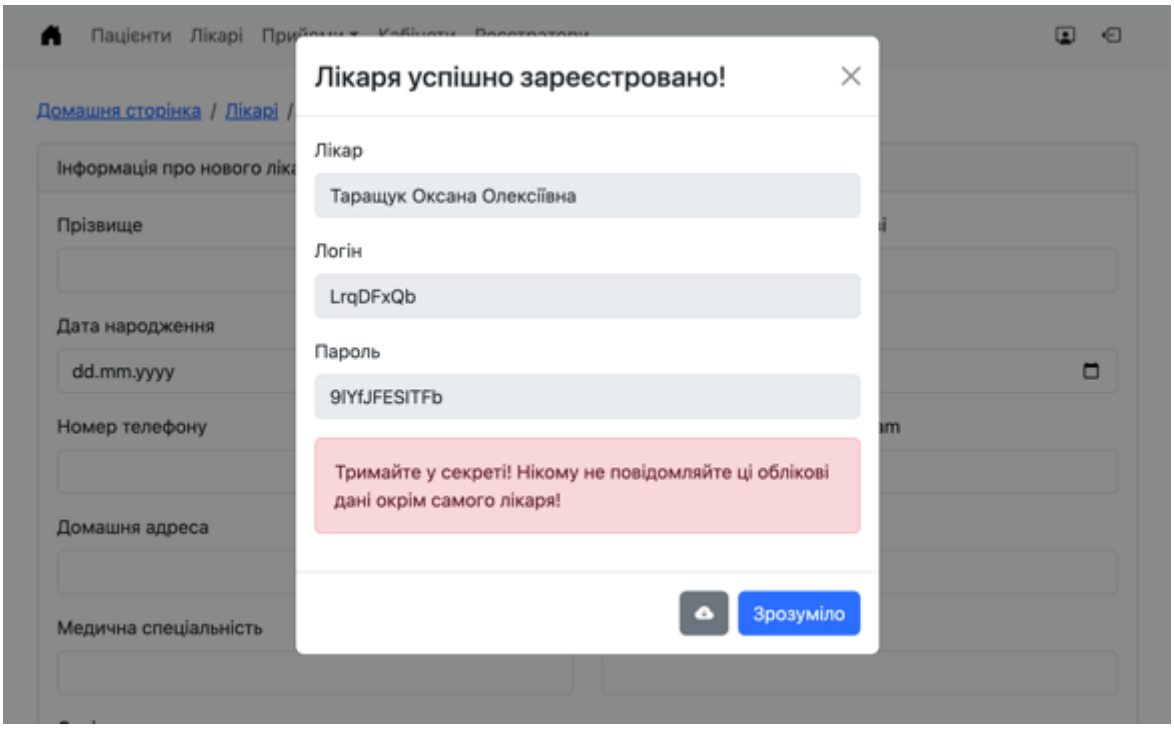


Рисунок Г.7 – Додавання нового лікаря

З згенерованими логіном та паролем лікар зможе увійти до свого облікового запису. Натиснувши на піктограму хмари, можна завантажити облікові дані файлом формату JSON.

У подальшому, для безпеки, лікар зможе змінити свій пароль, натиснувши на піктограму людини справа у навігаційному меню. Ця кнопка переадресує користувача на сторінку управління обліковим записом (рисунок Г.8).

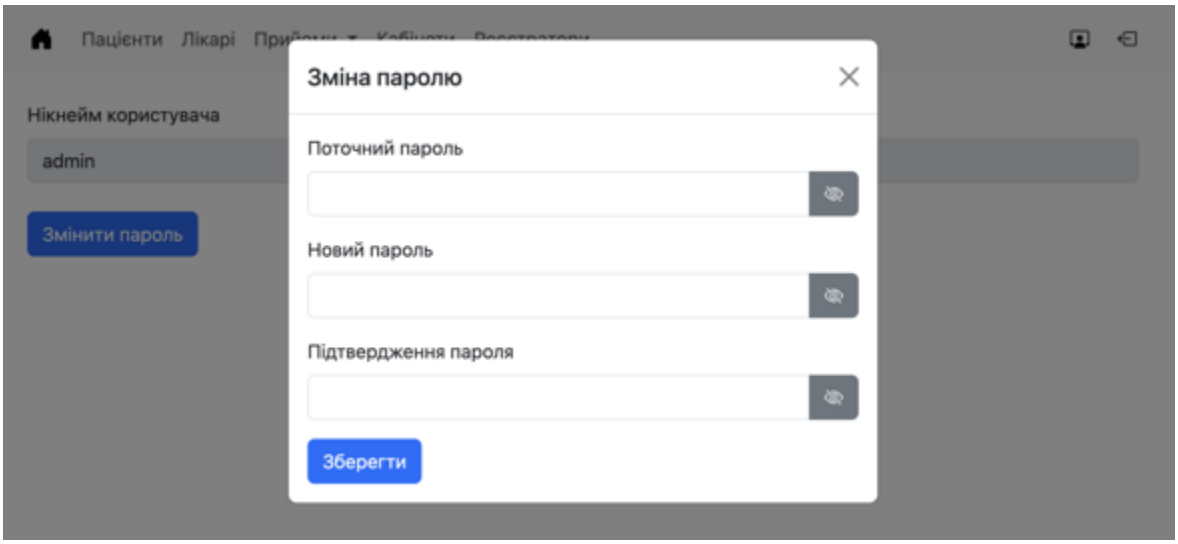


Рисунок Г.8 – Зміна паролю від облікового запису користувача

На сторінці управління обліковим записом, натиснувши на кнопку “Змінити пароль”, можна змінити поточний пароль на новий, ввівши бажаний новий пароль та підтвердивши новий пароль.

Г.6 Перегляд детального звіту про певний об'єкт

На сторінках “Пацієнти”, “Лікарі”, “Прийоми”, “Кабінети”, “Реєстратори” у кожному рядку таблиці є стовпчик “Дія”. Наприклад, для перегляду детального звіту про лікаря слід натиснути на піктограму ока у стовпчику “Дія” у рядку необхідного лікаря, після цього відкриється сторінка “Інформація про лікаря” (рисунок Г.9).

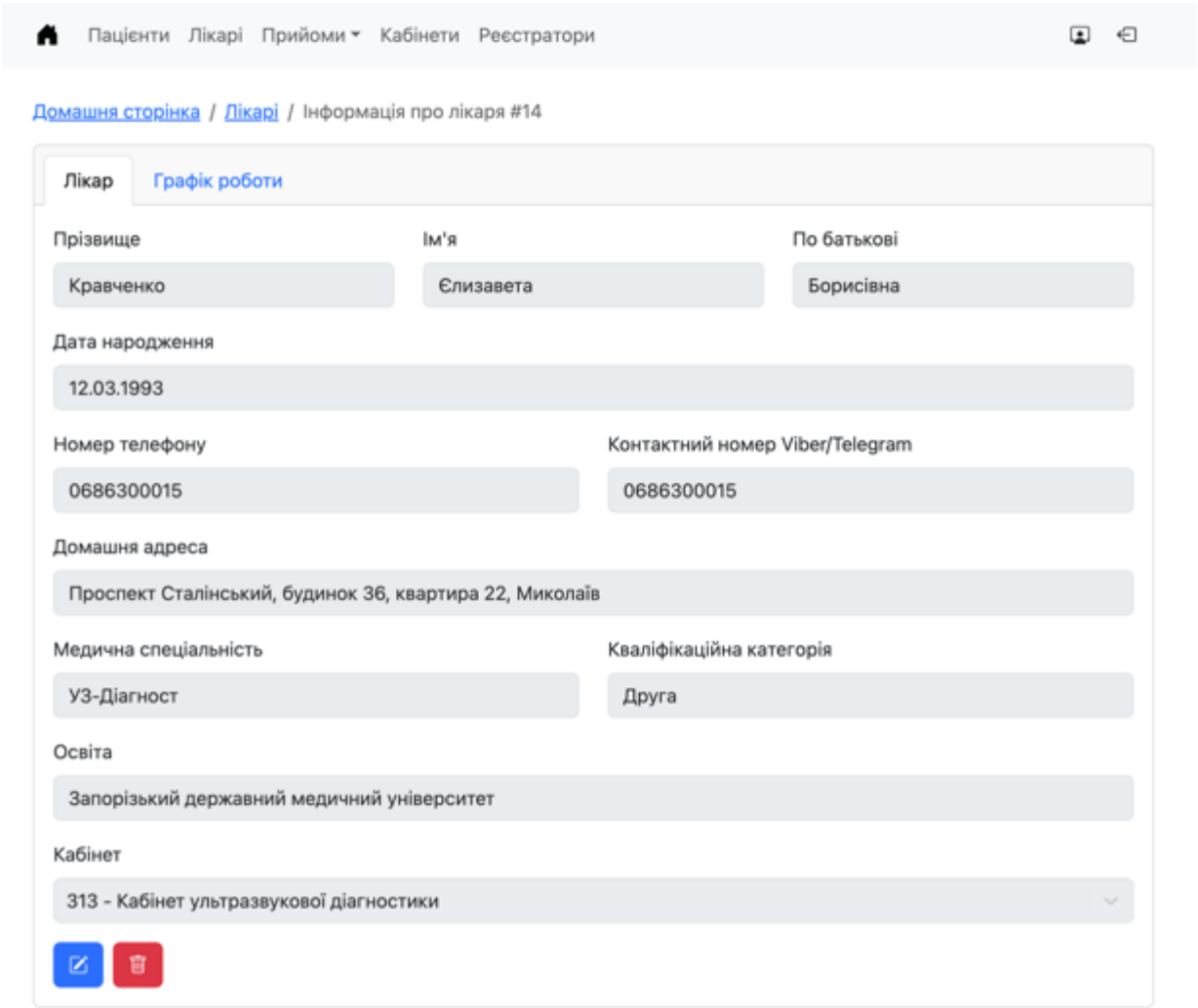


Рисунок Г.9 – Сторінка “Інформація про лікаря”

Перегляд звітів про пацієнтів, реєстраторів, кабінети та прийоми відбувається аналогічно до описаного процесу з лікарями.

Г.6.1 Редагування та видалення інформації про певний об'єкт

Сторінка перегляду інформації про лікаря надає можливість редагувати дані про обраного лікаря, для цього спочатку натисніть на піктограму олівця, потім оберіть бажане для редагування поле, введіть нову інформацію та натисніть на кнопку “Зберегти” для зберігання змін або “Скасувати” для скасування внесених змін (рисунок Г.10).

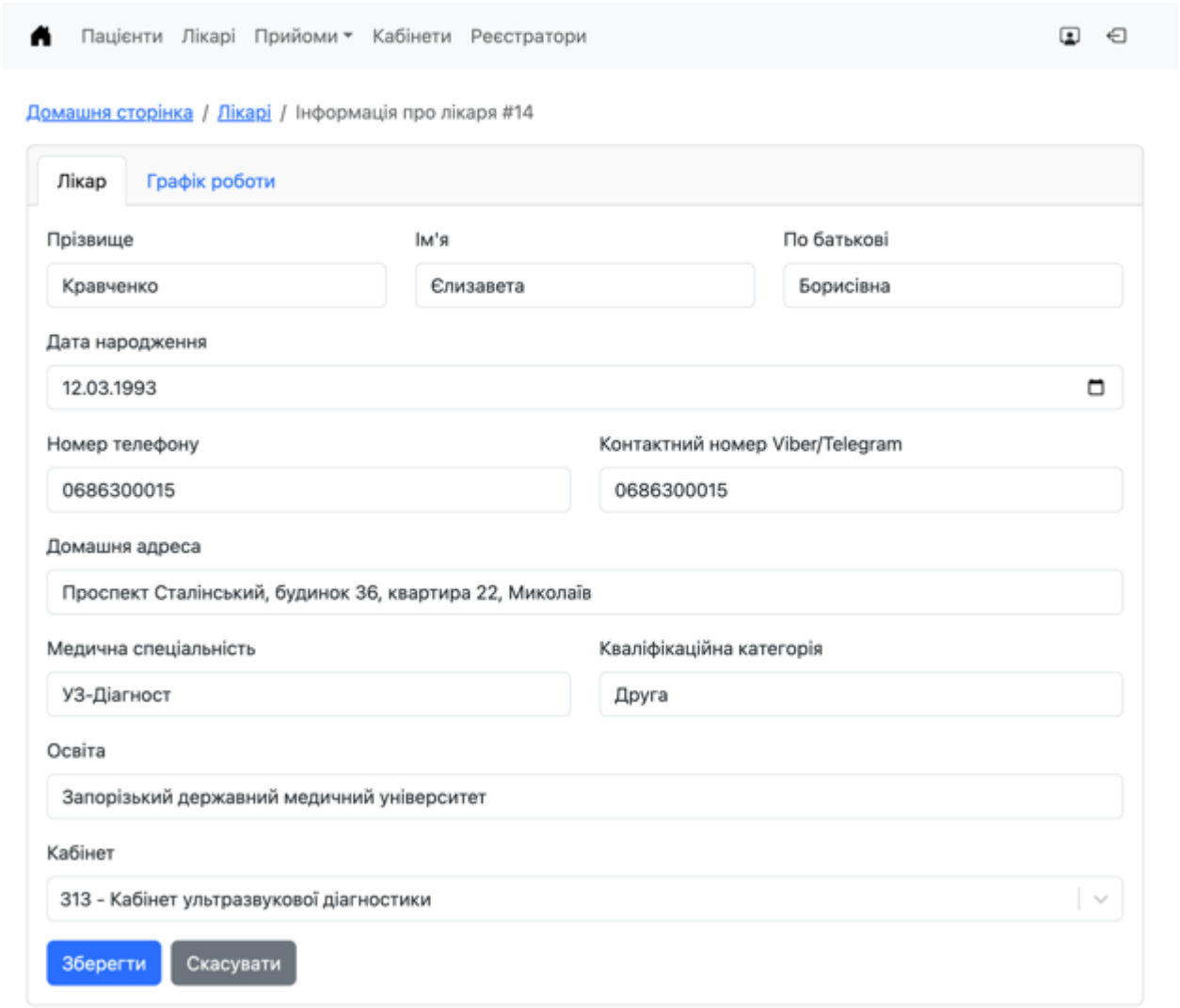


Рисунок Г.10 – Редагування інформації про лікаря

Для видалення певного лікаря з системи слід натиснути на піктограму смітника. Після цього у вікні підтвердження натиснути “Видалити” (рисунок Г.11).

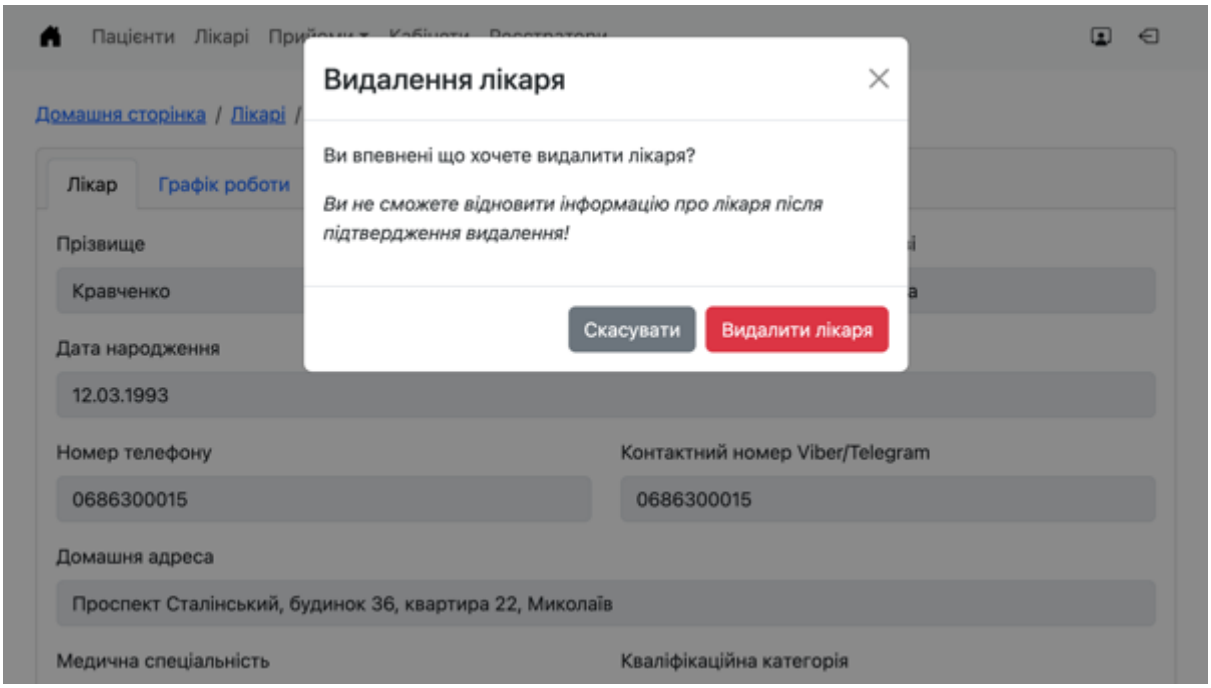


Рисунок Г.11 – Видалення інформації про лікаря

Редагування та видалення інформації про пацієнтів, кабінети, реєстраторів, прийоми відбувається аналогічно до роботи з лікарями.

Г.7 Робота з прийомами пацієнтів до лікарів

Для призначення нового прийому пацієнта до лікаря слід натиснути на піктограму журналу в стовпчику "Дія" у рядку необхідного пацієнта у таблиці на сторінці “Пацієнти”, після цього відкриється сторінка “Призначення нового прийому” (рисунок Г.12).

На сторінці призначення нового прийому необхідно обрати бажаного лікаря з випадаючого списку у стовпчику “Лікар” та бажану дату прийому. Виходячи з графіку роботи лікаря, буде згенерована таблиця розкладу роботи лікаря на обрану дату, можна переглянути вільні та зайняті іншими прийомами проміжки часу. Після вирішення часу прийому слід вибрати бажаний час на лінії часу, затиснувши ліву кнопку миші, або натиснути на кнопку “Призначити прийом” та ввести вручну час прийому, в обох випадках відкривається модальне вікно підтвердження призначення прийому (рисунок Г.13).

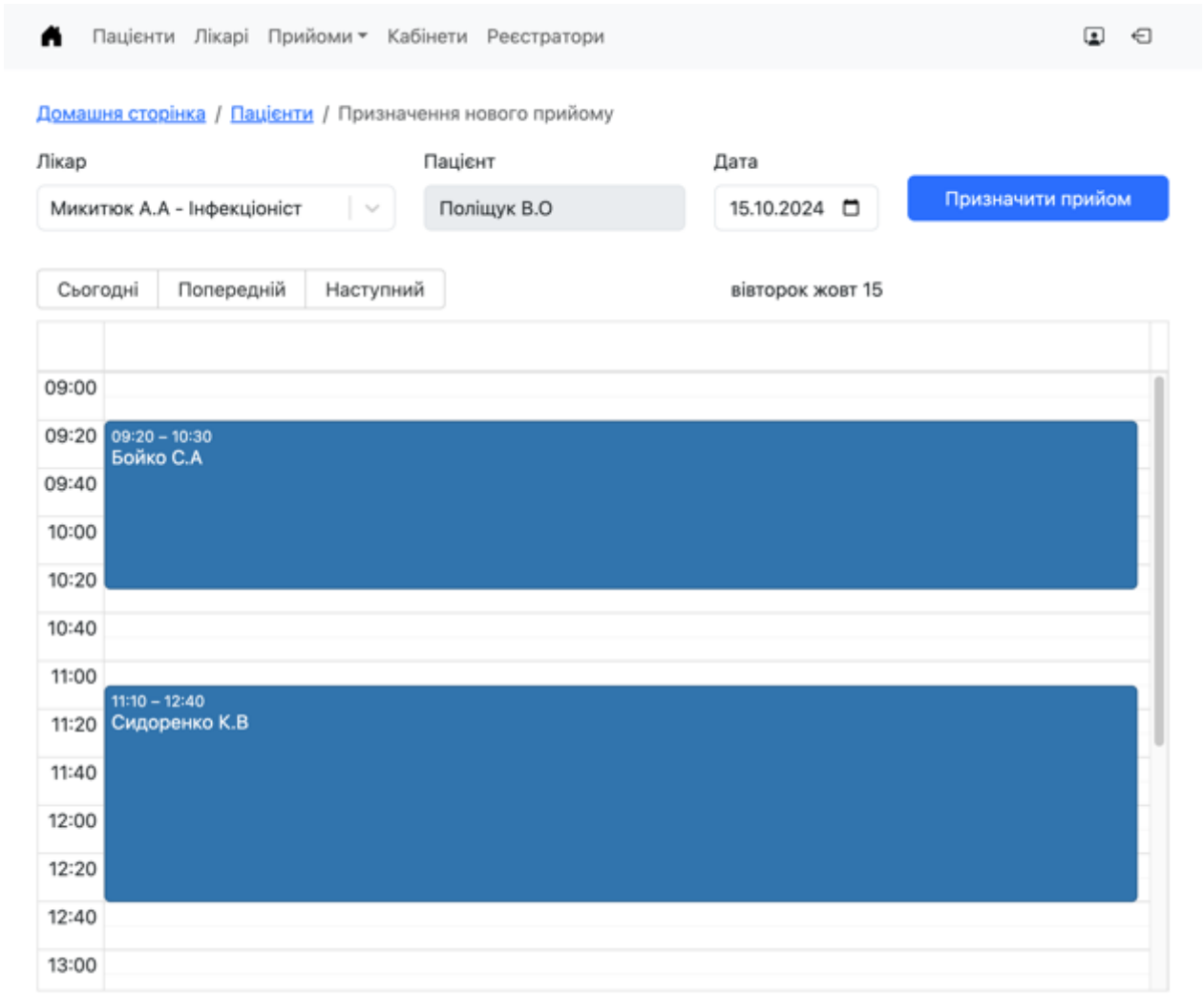


Рисунок Г.12 – Сторінка “Призначення нового прийому”

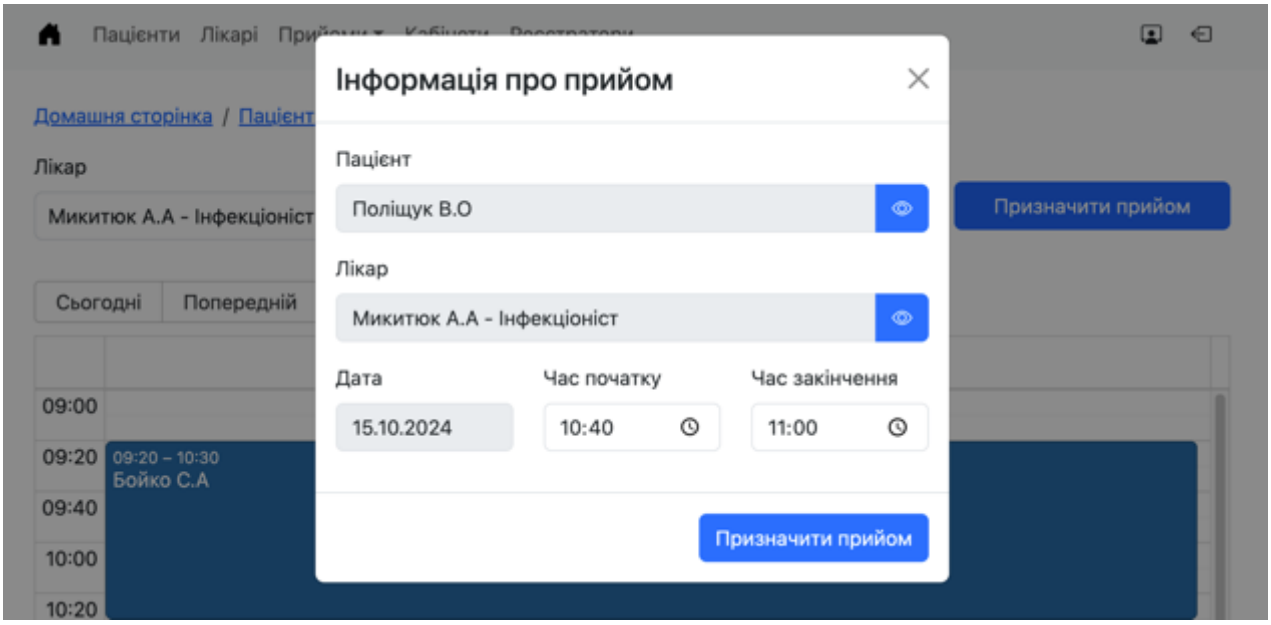


Рисунок Г.13 – Призначення нового прийому

Після призначення нового прийому користувач має змогу перейти на сторінку створеного прийому, натиснувши на прийом в розкладі (рисунок Г.14).

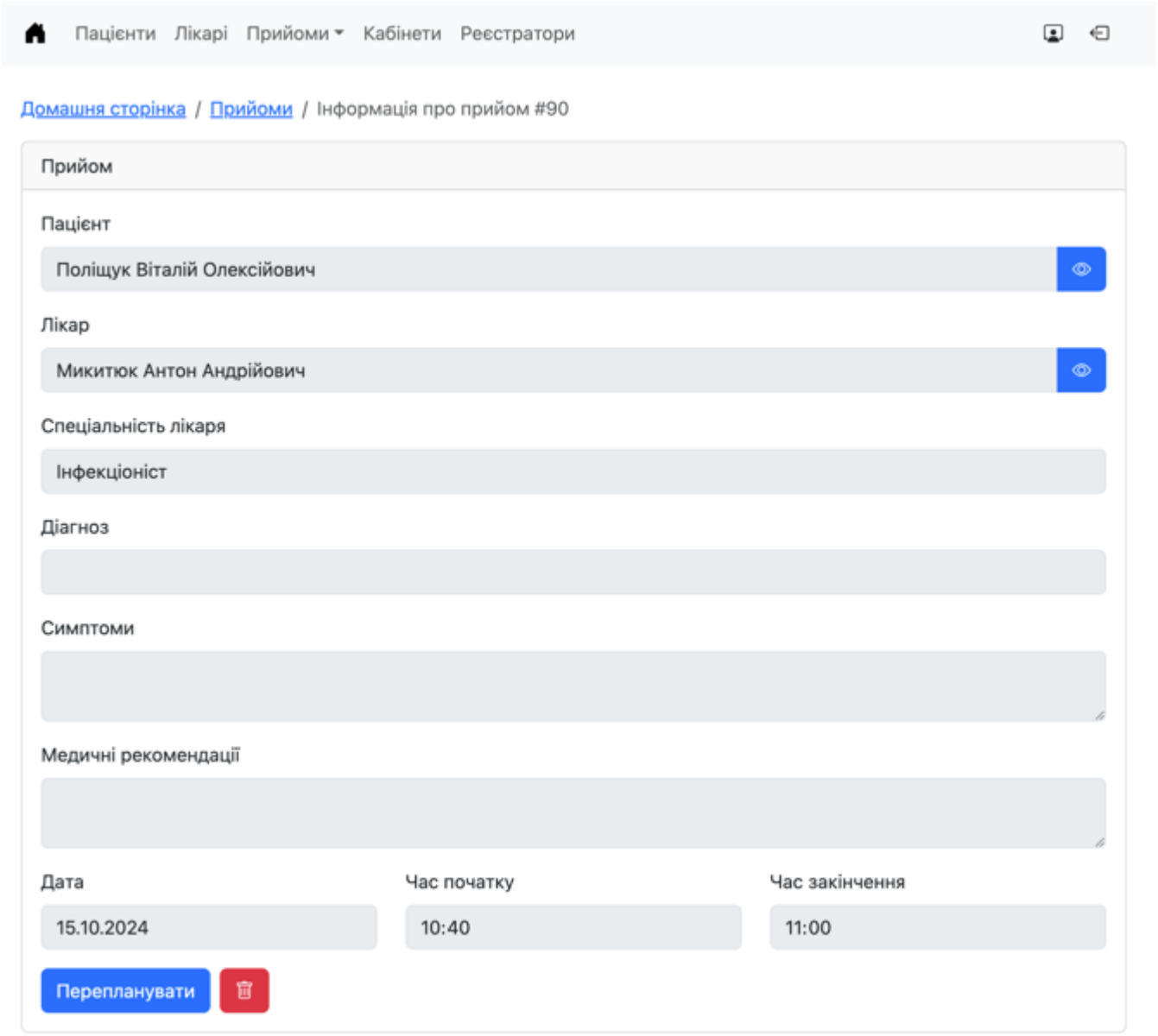


Рисунок Г.14 – Сторінка “Інформація про прийом”

На цю ж сторінку можна потрапити через сторінку “Прийоми”, натиснувши на піктограму ока у рядку необхідного прийому, подібно до того, як було показано при роботі з лікарем.

ДОДАТОК Д – ПРОГРАМА ТА МЕТОДИКА ВИПРОБУВАНЬ

1 Об'єкт випробувань

1.1 Назва об'єкту

Повна назва об'єкту випробувань: вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed".

Позначення: вебзастосунок.

1.2 Область застосування

Вебзастосунок призначений для спрощення роботи реєстратури медичної клініки "CompliMed" та покращення ефективності роботи медичного клініки.

2 Мета випробувань

Метою проведення випробувань є перевірка працездатності вебзастосунку, перевірка відповідності характеристик розробленого вебзастосунку та вимог, викладених в технічному завданні, яке наведено в додатку А.

3 Вимоги до програми

При проведенні випробувань функціональних характеристик, вебзастосунок для автоматизації роботи реєстратури медичної клініки "CompliMed" підлягає перевірці на відповідність вимогам, викладеним у пункті "Вимоги до функціональних характеристик" технічного завдання.

4 Вимоги до програмної документації

До складу програмної документації повинні входити наступні документи:

– технічне завдання;

– програма та методика випробувань;

– керівництво користувача;

– опис програми;

– код програми.

5 Засоби і порядок випробувань

5.1 Склад технічних засобів

Склад технічних засобів що використовувались під час випробувань:

– процесор: 8 ядер, 3.2 GHz;

– оперативна пам'ять: 8 ГБ;

– SSD: 100 ГБ вільного місця;

– інтернет з'єднання зі швидкістю: 50 Мбіт/с.

5.2 Склад програмних засобів

Склад програмних засобів, що використовувались під час випробувань:

– операційна система: macOS Sequoia 15.2;

– браузер Google Chrome 134;

5.3 Порядок проведення випробувань

Проведення випробувань програмного забезпечення має відбуватися у наступному порядку:

– виконуються інструкції до кожної функції;

– проводиться аналіз отриманих результатів та визначається відповідність програмного забезпечення вимогам.

– робиться візуальний перегляд програмного забезпечення для виявлення незначних помилок;

– проводиться перевірка вимог до програмної документації.

У разі виявлення помилок в роботі програмного забезпечення складається їх перелік та обговорюється термін їх виправлення розробником. Після усунення недоліків замовник проводить повторне тестування в повному обсязі.

6 Методи випробувань

6.1 Методика проведення перевірки вимог до функціональних характеристик програмного продукту

Перевірка працездатності та відповідності вимогам вебзастосунку здійснюється згідно з пунктом "Вимоги до функціональних характеристик" технічного завдання.

Випробування вважається успішно завершеним в разі відповідності функціональних можливостей розробленого вебзастосунку вимогам викладеним в технічному завданні.

Тестування виконується за методом чорної скриньки. В цьому методі виконується тестування функціоналу програмного забезпечення без знання принципів та організації внутрішньої роботи програми. Усі функції вебзастосунку проходитимуть випробування. Через велику кількість функцій, які треба випробовувати, буде представлено лише декілька результатів випробувань функцій вебзастосунку.

Складемо програму випробувань:

Випробування функції "Введення інформації про нового лікаря"

Для відтворення необхідно виконати наступні кроки:

1) Перейти на сторінку "Лікарі" з навігаційного меню.

2) Ініціювати створення нового лікаря натиснувши на кнопку "Новий лікар", тим самим відбудеться переведення на сторінку створення нового лікаря з відповідною формою.

3) Заповнити необхідну інформацію про лікаря у формі на сторінці введення інформації про нового лікаря. Для тестового прикладу в якості вхідних даних можна використати наступні:

– Прізвище: Здражевський, Ім'я: Євген, По батькові: Юрійович.

– Лікарська спеціальність: Лікар кардіолог, Кваліфікаційна категорія: Вища.

– Освіта: Київський національний медичний університет ім. О.О. Богомольця.

– Контактний номер Viber/Telegram: +380915448534, Номер телефону: +380965527526.

– Дата народження: 10.03.1990.

– Домашня адреса: вул. Космонавтів, буд. 42, кв. 15, м. Миколаїв, 54058.

– Номер кабінету: 216.

4) Підтвердити створення нового лікаря натиснувши на кнопку керування "Додати нового лікаря".

Інформація про нового лікаря та його обліковий запис повинна бути збережена у системі. Лікар повинен бути закріплений за кабінетом 216.

Користувачу повинні повідомитися автоматично згенеровані дані від облікового запису лікаря (пара логін:пароль).

Випробування функції "Призначити новий прийом пацієнта до лікаря"

Для відтворення необхідно виконати наступні кроки:

1) Перейти на сторінку "Пацієнти" з навігаційного меню.

2) На сторінці "Пацієнти" обрати пацієнта та ініціювати призначення йому прийому. Після цього відбудеться переведення на сторінку "Розклад" з розкладом роботи лікарів та можливістю призначення нового прийому.

3) На сторінці "Розклад" обрати бажаного лікаря із запропонованого списку існуючих в системі лікарів.

4) Ввести дату, на яку необхідно призначити прийом та переглянути розклад роботи лікаря.

4) Обравши бажаний час прийому та дату, натиснути на кнопку керування створення нового прийому та ввести необхідні вхідні дані. Для тестового прикладу в якості вхідних даних можна використати наступні:

– Лікар: 12 - Васильєв Микола Михайлович (обраний зі списку).

– Пацієнт: 7 - Кузьменко Сергій Ігорович (обраний зі списку).

– Дата: 07.04.2025, Час: 10:20 – 10:50.

5) Підтвердити призначення нового прийому.

Інформація про новий прийом пацієнта до лікаря повинна бути збережена у системі, а після успішного призначення прийому повинно відобразитися відповідне повідомлення.

6.2 Методика проведення перевірки вимог до програмної документації

Перевірка відповідності програмної документації на програмний продукт здійснюється візуально представником організації, відповідальної за експлуатацію. У процесі перевірки, склад та повнота програмної документації, наданої розробником, порівнюються зі списком програмної документації, вказаним у пункті "Вимоги до програмної документації" цього документа. Якщо склад та повнота програмної документації, наданої розробником, відповідають списку вимагаємої програмної документації, то перевірка вважається успішно завершеною.