

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

ПОЯСНЮВАЛЬНА ЗАПИСКА

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

на тему:

**«Інформаційна система управління та автоматизації
обліку стоматологічної клініки»**

Виконав студент групи ІПЗ-21-4
ТЕРЕЩУК Владислав Олексійович

Керівник роботи:
ЧИЖМОТРЯ Олексій Володимирович

Рецензент:
ТОЛСТОЙ Ігор Анатолійович

Житомир – 2025

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

«ЗАТВЕРДЖУЮ»

Зав. кафедри інженерії програмного
забезпечення

Тетяна ВАКАЛЮК

«14» лютого 2025 р.

ЗАВДАННЯ
на кваліфікаційну роботу

Здобувач вищої освіти: **ТЕРЕЩУК Владислав Олексійович**

Керівник роботи: **ЧИЖМОТРЯ Олексій Володимирович**

Тема роботи: **«Інформаційна система управління та автоматизації обліку стоматологічної клініки»,**

затверджена Наказом закладу вищої освіти від **«14» лютого 2025 р., №61/с**

Вихідні дані для роботи: **процеси документообліку, управління та планування графіків роботи стоматологічної клініки, які використовують сучасні технології розробки настільних та веборієнтованих програмних продуктів.**

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

Розділ	Консультант	Завдання видав	Завдання прийняв
1	Чижмотря О. В.	14.02.2025р.	14.02.2025р.
2	Чижмотря О. В.	14.02.2025р.	14.02.2025р.
3	Чижмотря О. В.	14.02.2025р.	14.02.2025р.

РЕФЕРАТ

Випускна кваліфікаційна робота бакалавра складається з програмного комплексу та пояснювальної записки. Пояснювальна записка до випускної роботи містить 94 сторінки, включає 40 ілюстрацій, 13 таблиць, 3 додатки та 23 використаних літературних джерела.

Метою роботи є розробка алгоритмів функціонування та реалізація програмного комплексу інформаційної системи управління та автоматизації обліку стоматологічної клініки, що дозволяє підвищити ефективність документообігу, планування та обслуговування пацієнтів.

У роботі здійснено аналіз предметної області, сформульовано вимоги до функціоналу системи, розроблено архітектуру з урахуванням клієнт-серверного підходу, реалізовано компоненти на основі сучасних технологій ті фреймворків платформи .NET а також MS SQL. Система складається з серверної частини, вебсайту для пацієнтів і настільного Windows застосунку для співробітників клініки.

Застосовано мікросервісну архітектуру, реалізовано програмний продукт, що включає функціонал авторизації та автентифікації, електронної черги на прийом до лікарів, формування розкладу роботи працівників, перегляду та управління клініками, прейскурантом послуг, управління профілями користувачів, а також адміністрування клінік та налаштування контенту вебсайту.

КЛЮЧОВІ СЛОВА: АВТОМАТИЗАЦІЯ, ВЕБСАЙТ, ЗАСТОСУНОК, ІНФОРМАЦІЙНА СИСТЕМА, МІКРОСЕРВІС, УПРАВЛІННЯ.

					ІПЗ.КР.Б – 121 – 25 – ПЗ				
Змн.	Арк.	№ докум.	Підпис	Дата	Інформаційна система управління та автоматизації обліку стоматологічної клініки Пояснювальна записка	Лім.	Арк.	Аркушів	
Розроб.		Терещук В. О							
Керівник		Чижмотря О. В.					3	94	
						Житомирська політехніка, група ІПЗ-21-4			
Рецензент		Толстой І. А.							
Зав. каф.		Вакалюк Т. А.							

ABSTRACT

The bachelor's qualification thesis consists of a software system and an explanatory report. The explanatory report includes 94 pages, 40 illustrations, 13 tables, 3 appendices, and 23 referenced literary sources.

The objective of the work is to develop the operation algorithms and implement a software system for managing and automating the record-keeping processes of a dental clinic, which helps improve the efficiency of document workflow, scheduling, and patient service.

The thesis presents an analysis of the subject area, defines the system's functional requirements, and develops the architecture based on a client-server approach. The components are implemented using modern .NET platform technologies and frameworks, as well as MS SQL. The system consists of a backend server, a website for patients, and a Windows desktop application for clinic staff.

A microservice architecture is applied, and the system includes functionality for user authorization and authentication, electronic appointment queue management, staff schedule generation, clinic and service catalog management, user profile administration, clinic administration, and website content configuration.

KEYWORDS: AUTOMATION, APPLICATION, INFORMATION SYSTEM, MANAGEMENT, MICROSERVICE, WEBSITE

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						4
Змн.	Арк.	№ докум.	Підпис	Дата		

ЗМІСТ

РЕФЕРАТ	3
ABSTRACT	4
ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ.....	7
ВСТУП.....	8
РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ В СТОМАТОЛОГІЧНИХ КЛІНІКАХ	10
1.1 Постановка задачі	10
1.2 Аналіз аналогів програмного продукту.....	12
1.3 Вибір архітектури системи управління та автоматизації обліку стоматологічної клініки	18
1.4 Обґрунтування вибору інструментальних засобів.	22
Висновки до першого розділу	26
РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ СТОМАТОЛОГІЧНОЇ КЛІНІКИ	28
2.1 Специфікація вимог до програмного продукту.....	28
2.2 Визначення варіантів використання та проєктування об'єктно-орієнтованої структури системи	30
2.3 Розробка бази даних системи	35
2.4 Проєктування та реалізація алгоритмів роботи системи.....	42
Висновки до другого розділу	51
РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ СТОМАТОЛОГІЧНОЇ КЛІНІКИ.....	53
3.1 Порядок встановлення та налаштування параметрів системи.....	53
3.2 Структура інтерфейсу інформаційної системи управління та автоматизації обліку стоматологічної клініки	55
3.3 Тестування роботи програмного комплексу.....	68
Висновки до третього розділу	69

ВИСНОВКИ.....	71
СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ.....	73
ДОДАТКИ.....	75

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						6
Змн.	Арк.	№ докум.	Підпис	Дата		

ПЕРЕЛІК УМОВНИХ ПОЗНАЧЕНЬ

SOA – Service-oriented architecture
JSON – JavaScript Object Notation
JWT – JSON Web Token
HTTP – Hypertext Transfer Protocol
TCP – Transmission Control Protocol
CRUD – Create, Read, Update, Delete
REST – Representational State Transfer
API – Application Programming Interface
DTO – Data Transfer Object
WASM – WebAssembly
WPF – Windows Presentation Foundation
MVVM – Model-View-ViewModel
СКБД – Система керування базами даних
ORM – Object-relational mapping
LINQ – Language Integrated Query
XAML – Extensible Application Markup Language
NSIS – Nullsoft Scriptable Install System

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						7
Змн.	Арк.	№ докум.	Підпис	Дата		

ВСТУП

У багатьох галузях сучасного бізнесу та промисловості інформаційні системи та системи автоматизації процесів використовуються для організації та структурування даних, а також їх зберігання та зручного використання, зокрема пошуку, редагування, тощо. Сфера медичних послуг не є виключенням в цьому випадку, у ній навпаки активно використовується згадане програмне забезпечення. Інформаційні системи допомагають зменшити обсяги паперової документації, оптимізувати процеси зберігання критично важливих даних, збільшити темпи надання послуг пацієнтам та зменшити кількість черг в медичних закладах.

Згідно Закону України «Про державні фінансові гарантії медичного обслуговування населення», а також відповідними наказами Міністерства охорони здоров'я укладення договорів з Національною службою здоров'я України, облік пацієнтів та їхніх декларацій із сімейними лікарями, виписування електронних рецептів та ведення електронних медичних записів здійснюється через Електронну систему охорони здоров'я (ЕСОЗ). Інтеграція ЕСОЗ є обов'язковою умовою для роботи в рамках державної системи фінансування та надання медичних послуг. Водночас приватні медичні заклади, які не впроваджують сучасні інформаційні системи, стикаються з низкою організаційних проблем.

Паперові журнали з графіками роботи лікарів та обліком записів на прийом є потенційно вразливими до численних помилок, плутанини чи накладок в розкладі при ручному заповненні, що негативно впливає в першу чергу на якість обслуговування пацієнтів, а також на загальну ефективність роботи. Додатково, ручний облік особистих даних пацієнтів, таких як медичні картки та історія хвороб та діагнозів може спричинити труднощі під час пошуку необхідної інформації, що в свою чергу є ризиком під час фактичного надання медичних послуг.

Перехід на цифровий тип документообліку позитивно вплине на ефективність роботи, забезпечивши зручність виконання щоденних операцій персоналу та автоматизацію багатьох процесів, наприклад – формування розкладу. Крім того, інформаційні системи виконують обов'язкову валідацію даних, перевірки на конфлікти чи накладки, що зменшує вплив «людського

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						8
Змн.	Арк.	№ докум.	Підпис	Дата		

фактору» в загальній проекції документообліку. Ще однією перевагою цифрових систем є фактори безпеки та надійності: традиційні паперові архіви документів можуть бути пошкоджені чи знищені за різних обставин (пожежа, затоплення, катаклізм, тощо), або ж до них може отримати доступ зловмисник, що в свою спричинить витік персональних даних пацієнтів. Тоді як віддалені або хмарні сховища інформації забезпечують надійне зберігання даних на безпечних пристроях і часто надають послуги резервного копіювання, шифрування та захисту інформації від несанкціонованих втручань, гарантуючи її цілісність.

Метою випускної кваліфікаційної роботи є розробка алгоритмів роботи та реалізація відповідного програмного комплексу інформаційної системи управління та автоматизації обліку стоматологічної клініки, який допоможе вирішити описані проблеми.

Об'єктом дослідження є процеси документообліку, управління та планування графіків роботи стоматологічної клініки.

Предметом дослідження є використання сучасних технологій розробки настільних та веборієнтованих програмних продуктів для розробки визначеної інформаційної системи. Основні методи дослідження – аналіз вимог та системного проєктування, метод об'єктно-орієнтованого програмування, аналіз та візуалізація даних, методи захисту інформації.

Описані складові обумовлюють наступні **завдання випускної роботи**:

- аналіз існуючих інформаційних систем управління медичними закладами, виявлення переваг та недоліків;
- аналіз основних процесів та задач предметної області, що підлягають автоматизації;
- формування списку вимог та критеріїв розробки програмного комплексу;
- проєктування та моделювання архітектури інформаційної системи;
- вибір та обґрунтування засобів розробки програмного комплексу;
- реалізація програмного продукту інформаційної системи управління та автоматизації обліку стоматологічної клініки;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						9
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 1. АНАЛІЗ НАПРЯМКІВ ВИКОРИСТАННЯ ІНФОРМАЦІЙНИХ ТЕХНОЛОГІЙ ДЛЯ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ В СТОМАТОЛОГІЧНИХ КЛІНІКАХ

1.1. Постановка задачі

Інформаційна система управління та автоматизації обліку покликана сприяти вирішенню низки проблем організації, процесів управління, документообігу та формування розкладу роботи медичного закладу, що надає послуги в галузі стоматології. Ключова мета – поступовий повний перехід стоматологічної клініки на ведення документації та обліку в електронному вигляді з використанням майбутнього програмного продукту.

В першу чергу, програмний комплекс повинен надавати можливість безперешкодного доступу до інформації (за умови відповідного рівня доступу) та зручний інтерфейс користувача для виконання всіх необхідних операцій як з боку клієнта, так і з боку працівників клініки. CRUD – це базовий набір операцій, за допомогою якого програмний продукт може покрити всі основні задачі роботи з даними. Такі операції є універсальним інструментом для будь-якої інформаційної системи, адже вони охоплюють всі етапи життєвого циклу даних в системі та дозволяють обробляти інформацію незалежно від її природи.

Важливим аспектом роботи інформаційної системи є реалізація механізмів аутентифікації та авторизації. Оскільки система працюватиме з великою кількістю конфіденційної інформації, такої як контактні дані, історія хвороб, медичні картки та облікові документи фінансових операцій, надвичайно важливо реалізувати надійні механізми захисту та обмежень доступу. Програмний комплекс повинен підтримувати багаторівневу модель авторизації на базі ролей користувачів. Наприклад, для пацієнтів передбачити доступ виключно до особистих даних, а для працівників клінік рівень доступу визначатиметься посадою: лікарі, персонал, адміністрація. В якості аутентифікації використовувати особистий номер телефону або email-адресу користувача з обов’язковим проходженням етапу підтвердження,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						10
Змн.	Арк.	№ докум.	Підпис	Дата		

що включає в себе відправку відповідного повідомлення з використанням месенджера або на електронну пошту користувача.

Неможливо уявити інформаційну систему обліку медичного закладу, що не буде включати в себе функціонал реєстратури. Програмний продукт повинен надавати можливість реєстрації користувачів (як самотійно, так і з допомогою працівника безпосередньо під час візиту до клініки), облік даних медичних карток пацієнтів та наданих їм послуг, заповнення історії хвороб та діагнозів, особистий кабінет користувача, прогнозування профілактичних візитів.

Обов'язковою складової майбутньої системи є реалізація модуля «Електронна черга». Цей сервіс як частина програмного комплексу повинен вирішити проблеми, що виникають під час використання паперових журналів запису, які заповнюються вручну. Механізм електронного запису на прийом зменшить навантаження на персонал клінік, знявши з них відповідальність за контроль розкладу роботи лікарів та записів пацієнтів на прийом, а також вирішить проблему накладок записів та помилок у веденні журналів. Крім того, завдяки реалізації модуля в вигляді онлайн-сервісу для користувачів зникне необхідність особистого відвідування бажаного закладу або проведення телефонних розмов для запису на прийом. Такі класичні способи залишаються як додаткові варіанти запису на прийом для клієнтів, що не мають можливості використовувати онлайн-сервіс, при цьому відповідальним за надання консультації та введення інформації в систему залишається персонал клініки.

Задачі предметної області, що підлягають автоматизації:

- ведення історії візитів пацієнтів та наданих послуг в рамках цих візитів;
- формування статистики, обрахунків сумарних доходів та витрат за певний проміжок часу;
- підтвердження записів на прийом;
- формування графіку роботи лікарів з врахуванням вихідних, інуючих записів на прийом;
- інвентаризація стоматологічних матеріалів, медичних засобів, ліків, тощо.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						11
Змн.	Арк.	№ докум.	Підпис	Дата		

1.2. Аналіз аналогів програмного продукту

В результаті реформування системи охорони здоров'я України медичні інформаційні системи зазнали активного розвитку, тому на сьогоднішній день можна знайти досить багато програмних систем, що надають схожі функціональні можливості. Серед них можна визначити два типи: медичні системи загального користування та вузькоспеціалізовані рішення. Дослідження програмних продуктів загальної спеціалізації допоможе виділити особливості реалізації важливих функцій для медичного закладу незалежно від роду діяльності, таких як облік пацієнтів та послуг, реєстратура та автоматизація розкладу. Тоді як аналіз вузьконаправлених медичних систем стоматологічних клінік допоможе зосередитись на специфічних потребах закладів стоматологічної спеціалізації.

Як приклад системи загального користування можна навести систему Simplex Med – комплексну медичну інформаційну систему, що призначена для автоматизації процесів у медичних закладах [1]. Складається з окремих підсистем, зокрема: електронна черга, реєстри працівників та пацієнтів, облік послуг та інших. Програмний комплекс реалізовано в вигляді настільного застосунку, що працює в парі з віддаленим сервером (рис. 1.1).

Рисунок 1.1 – MIC Simplex-Med (Робоче місце лікаря)

Основними перевагами цієї системи можна вважати:

- автоматизовані процеси планування розкладу лікарів;
- широкий спектр функціональних можливостей;

До недоліків можна віднести:

- складний та застарілий інтерфейс програми з точки зору користувача;
- загально-спеціалізована система;
- доступ лише з пристрою, де встановлено програмний комплекс;

Doctor Eleks – ще одна популярна серед медичних закладів медична інформаційна система, яка складається з більш ніж 40 модулів (рис. 1.2). Систему позиціонують як зручне рішення для комунальних та приватних закладів, а також для медичних кабінетів та аптечних закладів [2]. Вона складається з таких продуктів, як Кабінет лікаря, Кабінет пацієнта, Реєстратура, Статистика, тощо. Перераховані продукти в свою чергу включають в себе функціонал для працівників та пацієнтів, автоматизоване формування довідкових матеріалів та звітної документації різних типів, інструменти для аналітики та звітності.

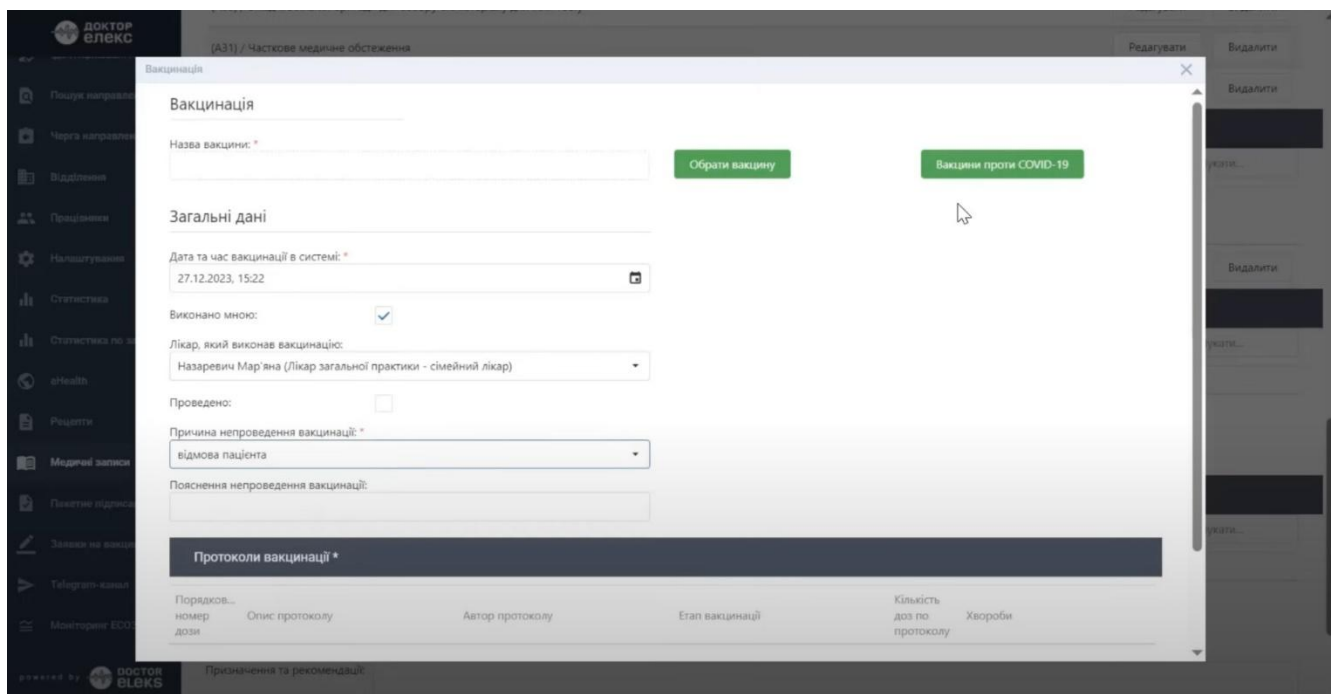


Рисунок 1.2 – МІС Doctor Eleks (Робоче місце лікаря)

Сильні сторони програмного комплексу:

- гнучкість та модульна архітектура;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк. 13
Змн.	Арк.	№ докум.	Підпис	Дата		

- широкий функціонал та велика кількість додаткових рішень;
- підтримка SMS-сервісів та IP-телефонії;
- підтримка роботи з медичним обладнанням;

Недоліки:

- система призначена для загального використання, не надає вузькоспеціалізованого функціоналу;

SkyService POS – універсальна система автоматизації бізнес процесів яка зазвичай використовується в сфері торгівлі та ресторанного бізнесу, але також має заявлену підтримку для стоматологічних клінік (рис. 1.3). Основною спеціалізацією цієї системи є автоматизація процесів обліку доходів. Також програмний комплекс включає в себе функціонал планування прийому, інвентаризації та управління базою користувачів [3].

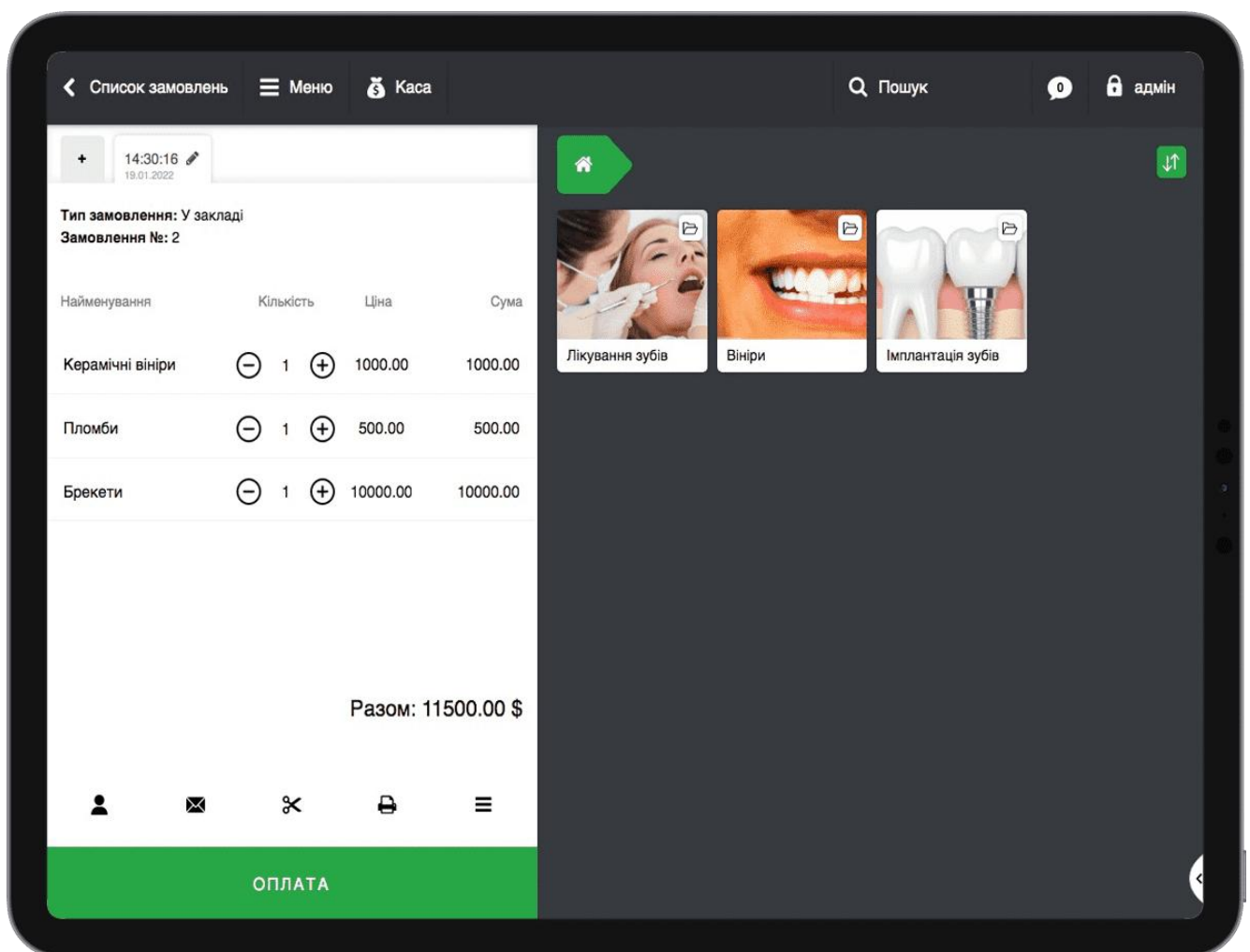


Рисунок 1.3 – SkyService POS

Ключові переваги:

- наявність модуля інвентаризації та обліку складу;
- повністю незалежне хмарне рішення з доступом з будь-якого пристрою;

Недоліки:

- універсальне рішення бізнес-обліку;
- обмежений функціонал медичної спрямованості;
- критична залежність від інтернет-з'єднання;

Open Dental – це спеціалізована інформаційна система для стоматологічних клінік з відкритим кодом. Вона надає клінікам інструменти для ведення медичних карток, управління записами пацієнтів, фінансового обліку та інтеграції з рентген-обладнанням [4]. Завдяки відкритому коду є можливість адаптувати її під потреби конкретної клініки. Та не дивлячись на те, що система є досить потужним спеціалізованим інструментом в галузі, на сьогоднішній день вона має складний та морально застарілий інтерфейс (рис. 1.4). Одночасно з цим, система досі отримує оновлення, що свідчить про її актуальність та підтримку.

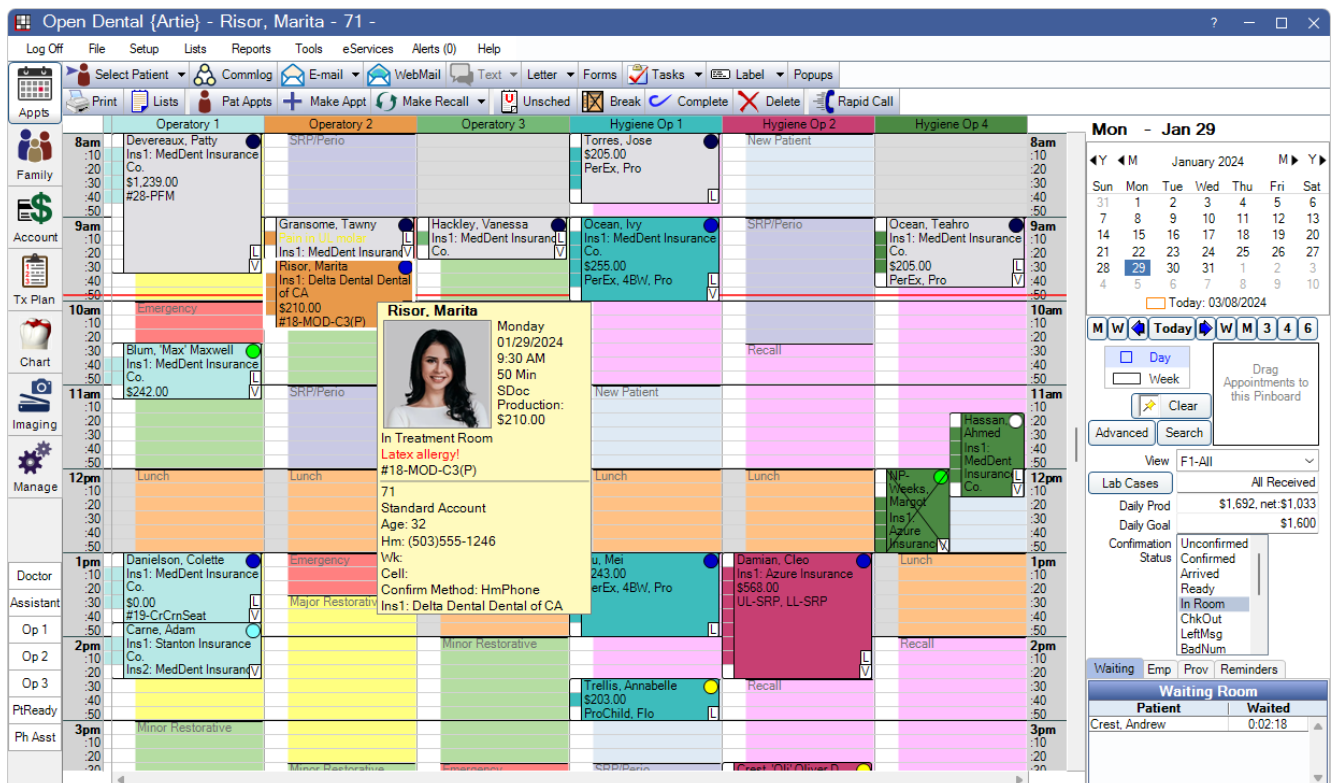


Рисунок 1.4 – Open Dental

Переваги:

- інтеграція з медичним обліднанням;
- підтримка як хмарного так і локального розгортання;
- функціонал обліку фінансових операцій;

Недоліки:

- складність встановлення та налаштування – програмний комплекс потребує досить високого рівня технічних знань та навичок;
- монолітна структура, неможливість інтеграції сучасних сервісів;

В результаті проведених досліджень було складено порівняльну таблицю існуючих аналогів програмного продукту (табл. 1.1). Для повноти отриманих результатів до порівняння включено як МІС специфічного спрямування, так і МІС загального призначення.

Таблиця 1.1

Порівняльна таблиця існуючих програмних продуктів

Характеристика	Simplex Med	Doctor Eleks	SkyService POS	Open Dental
1	2	3	4	5
Призначення	Загальна МІС	Загальна МІС	CRM система для бізнесу	Інформаційна система стоматологічної спеціалізації
Тип програмного комплексу	Настільний застосунок	Модульна система, окремі Web, Android та iOS застосунки	Web застосунок	Настільний застосунок
Доступність з мобільних пристроїв	+	+	+	–
Гнучкість, розширюваність	Обмежена	Висока	Середня	Обмежена
Залежність від інтернет з'єднання	Часткова	Часткова або Відсутня	Висока	Часткова або Відсутня
Електронна черга	+	+	–	–

1	2	3	4	5
Запис на прийом	+	+	+	+
Облік пацієнтів	Розширений	Розширений	Базовий	Розширений
Облік працівників	Розширений	Розширений	Відсутній	Розширений
Інвентаризація	–	+	+	–
Статистика та аналітика	Відсутня	Розширена	Розширена	Базова
Інтеграція медичного обладнання	–	+	–	+
SMS/IP Телефонія/Інше	–	+	+	–

На основі отриманих даних було визначено критично-важливі та другорядні функціональні можливості, які необхідно реалізувати в інформаційній системі управління та автоматизації обліку стоматологічної клініки. Зокрема:

- модуль авторизації та аутентифікації;
- модуль обліку пацієнтів та працівників;
- модуль електронної черги та автоматизованого формування розкладу роботи;
- модуль інвентаризації, фінансів, статистики та аналітики;
- модуль контролю «робочого простору» – адміністрування підключених до системи клінік.

Аналіз слабких сторін кожної з систем допоміг визначити помилки та аспекти, яких варто уникнути в розробці нової системи, зокрема: складний та перевантажений інтерфейс користувача, монолітна структура з поганою підтримкою розширень та інтеграцій, обмежений функціонал обліку, відсутність повноцінного онлайн запису на прийом та можливість використання програмного лише на певній платформі чи певному пристрою.

1.3. Вибір архітектури системи управління та автоматизації обліку стоматологічної клініки

Існує багато різних архітектурних підходів розробки програмних систем. Вони визначають структуру проекту, принципи взаємодії її компонентів та спосіб розгортання готового продукту. Найбільш широко використовуються такі архітектурні шаблони, як: монолітна архітектура, клієнт-серверна архітектура, сервісна (SOA) та мікросервісна архітектура.

Монолітна архітектура – це традиційна модель проектування програмного забезпечення, у якій вся програма функціонує як окремий модуль, автономний від інших програм [5]. Такі програми містять в собі весь функціонал в вигляді єдиної кодової бази яка зазвичай тісно пов'язана між собою, окремі частини програми активно взаємодіють одна з одною та напряму залежать одна від одної, а інтерфейс користувача також є частиною програмного комплексу.

Монолітні системи значно простіше в розробці та тестуванні в порівнянні з іншими архітектурними підходами, проте такий шаблон найкраще підійде для невеликих проектів зі стабільним та чітко сформованим функціоналом, адже масштабувати та розширювати такі проекти дуже непросто через велику кількість залежностей між компонентами. Розгортаються такі системи зазвичай на єдиному сервері в єдиному екземплярі, що часто призводить до перевантаження чи навіть відмов, що в свою чергу може призвести до фінансових збитків, тощо.

В основі **клієнт-серверної архітектури** лежать два компоненти: замовник послуг (клієнт) та постачальник послуг (сервер). Клієнтом прийнято вважати комп'ютер на стороні користувача, що відкрив певний ресурс тим самим відправивши запит про надання інформації чи виконання певних дій на сервер. Сервер – це зазвичай потужний стаціонарний комп'ютер, який призначений для безпосереднього вирішення тих чи інших завдань, виконання програмного коду, сервісних функцій, надання доступу користувачам до певних ресурсів, зберігання інформації, тощо. Алгоритм роботи клієнт-серверних систем полягає в тому, що клієнт відправляє запит на сервер, та очікує, поки він обробляється [6]. Як тільки сервер має готовий результат виконання певної операції – він відправляє цей

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						18
Змн.	Арк.	№ докум.	Підпис	Дата		

результат клієнту. Ключовим недоліком таких систем є чутливість до збоїв в роботі сервера. У випадку, коли з певних причин припиняє виконувати свої функції – усі клієнти також змушені призупиняти виконання тих чи інших дій.

Для запобігання згаданих проблем розробники ПЗ перейшли на кардинально інший підхід в розробці програмних продуктів. Цим підходом є використання **сервісної або мікросервісної архітектури**, яка полягає в поділі системи на окремі незалежні одна від одної частини, що взаємодіють між собою через стандартизовані протоколи HTTP або за допомогою брокерів повідомлень протоколу TCP. Обидва шаблони дозволяють зручно масштабувати систему, додавати новий функціонал або змінювати вже існуючий без прямого впливу на інші модулі системи.

Відмінність сервісної та мікросервісної архітектури полягає в тому, що у випадку мікросервісів кожен компонент відповідає за одну конкретну функцію, а між компонентами практично відсутні залежності, тому відмова одного мікросервіса ніяк не впливає на роботу інших. У сервісній архітектурі кожен компонент може виконувати декілька бізнес-функцій, а між сервісами все-ще залишаються певні зв'язки, зокрема використання спільних модулів, тощо [7].

Завдяки тому, що мікросервісна архітектура забезпечує повну незалежність компонентів один від одного, значно спрощується процес розгортання готового продукту. Крім того, з'являється можливість дуже гнучко керувати ресурсами та навантаженням на сервер за допомогою механізмів балансування навантаження (англ. Load Balancing). Такі механізми передбачають запуск найбільш навантажених мікросервісів у декількох екземплярах та забезпечують рівномірний розподіл запитів між цими екземплярами, що дозволяє уникнути загального зниження швидкодії системи через перевантаження чи затримки певного мікросервіса [8].

З огляду на необхідність реалізації модульної структури майбутньої системи, оптимальним вибором для серверної частини можна вважати мікросервісну архітектуру. Таке рішення прийнято, спираючись на низку важливих переваг мікросервісного підходу, зокрема: гнучкість та масштабованість, стійкість до

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						19
Змн.	Арк.	№ докум.	Підпис	Дата		

відмов, зручність розробки, підтримки та інтеграції зовнішніх API, SMS-шлюзів, тощо.

Програмний комплекс складатиметься з наступних компонентів:

Мікросервіс аутентифікації та авторизації: відповідальний за процес ідентифікації користувача (аутентифікація) та надання чи обмеження йому доступу опираючись на його роль та права в системі (авторизація). Для безпечної авторизації користувачів використовуватиметься механізм JWT – це відкритий стандарт для безпечної передачі інформації між сторонами у вигляді JSON об'єкта [9]. Такий підхід широко застосовується в розробці не тільки мікросервісних систем. Він полягає в збереженні інформації про користувача в токени, який надсилається з кожним запитом. Таким чином підвищується загальний рівень безпеки, виключається необхідність зберігання сесій на сервері, а на сам токен зазвичай накладають цифровий підпис для запобігання його підробці.

Мікросервіс планування та розкладу: відповідальний за автоматизацію складання графіків роботи лікарів та прийому пацієнтів. Включатиме в себе функціонал «Електронна черга», а також автоматично враховуватиме вихідні, відпустки та лікарняні лікарів під час формування розкладу й буде здійснювати контроль відсутності накладок чи конфліктів у розкладі.

Мікросервіс робочого простору (англ. Workspace): відповідальний за організацію та управління робочим простором клінік. Він дозволить здійснювати стандартні операції над клініками, налаштовувати їх ресурси та встановлювати взаємозв'язки між працівниками та клініками. Таким чином буде забезпечено чітку структуру та організацію даних про клініки, їх персонал та клієнтів. Сюди ж входить налаштування прейскуранту послуг тієї чи іншої клініки.

Мікросервіс обліку користувачів: відповідальний за зберігання та управління даних користувачів, як лікарів так і пацієнтів. Для пацієнтів передбачено керування особистою інформацією, історією відвідувань, діагнозом, медичними картками. Для лікарів – управління інформацією

Мікросервіс фінансів та статистики: відповідальний за процеси фінансового обліку клінік включаючи обробку платежів за медичні послуги,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						20
Змн.	Арк.	№ докум.	Підпис	Дата		

автоматичне створення квитанцій та накладних. Дозволить формувати та зберігати фінансову звітність, надаватиме інструменти для збору та перегляду статистичних даних, а також функціонал інвентаризації клінік: залишки розхідних матеріалів, витрати, поповнення, тощо.

Не можна забувати також про клієнтську частину програмного продукту. Спираючись на аналіз існуючих аналогів системи управління та автоматизації обліку стоматологічної клініки, можна визначити таку архітектуру користувацьких інтерфейсів: вебзастосунок для пацієнтів та настільний застосунок для працівників клініки.

Вебзастосунок являтиме собою інтерфейс для пацієнтів, через який будуть здійснюватись усі операції запису на прийом, перегляду преїскуранту послуг, розкладу роботи та особистого кабінету пацієнта, який включатиме історію візитів до клінік, фінансові виписти, тощо. Взаємодія клієнтського вебзастосунку з сервером здійснюватиметься за протоколом HTTP(S) через інтерфейс REST API.

Для персоналу та адміністрації клінік передбачено окремий настільний застосунок, який являтиме собою панель працівника з функціоналом відповідно до посади та рівня доступу. З міркувань безпеки, настільну версію для персоналу буде встановлено тільки на робочих пристроях клінік, щоб уникнути несанкціонованого доступу зловмисників до персональних даних пацієнтів через вебзастосунок. Для контролю факту встановлень передбачено механізм обліку активних сесій користувачів. За допомогою цього механізму адміністрація зможе здійснювати контроль над доступом до даних, а в разі необхідності вручну обмежити доступ завершивши ту чи іншу активну сесію.

Щоб уникнути прямої взаємодії клієнтських застосунків безпосередньо з мікросервісами, передбачено використання додаткового API шлюзу (англ. API Gateway). API Gateway – це архітектурний компонент, що виконує роль централізованого шлюзу для обробки всіх запитів від клієнтів до мікросервісів. [10] Фактично він є посередником між клієнтами та мікросервісами, що надають певні послуги. Головна мета використання API Gateway – маршрутизація клієнтських

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						21
Змн.	Арк.	№ докум.	Підпис	Дата		

запитів до відповідних мікросервісів, закривши прями доступ до мікросервісів ззовні системи. Другорядними варіантами використання API шлюзу є:

- аутентифікація, авторизація та фільтрація запитів до перенаправлення їх на конкретний мікросервіс;
- балансування навантаження при використанні підходу дублікації сервісів;
- кешування часто запитуваних даних;
- моніторинг та логування;
- обмеження швидкості та захист від DDoS атак.

Кожен з перерахованих другорядних варіантів застосування API шлюзу може бути застосований по мірі необхідності під час масштабування системи, завдяки чому буде закладено гнучку основу програмного комплексу.

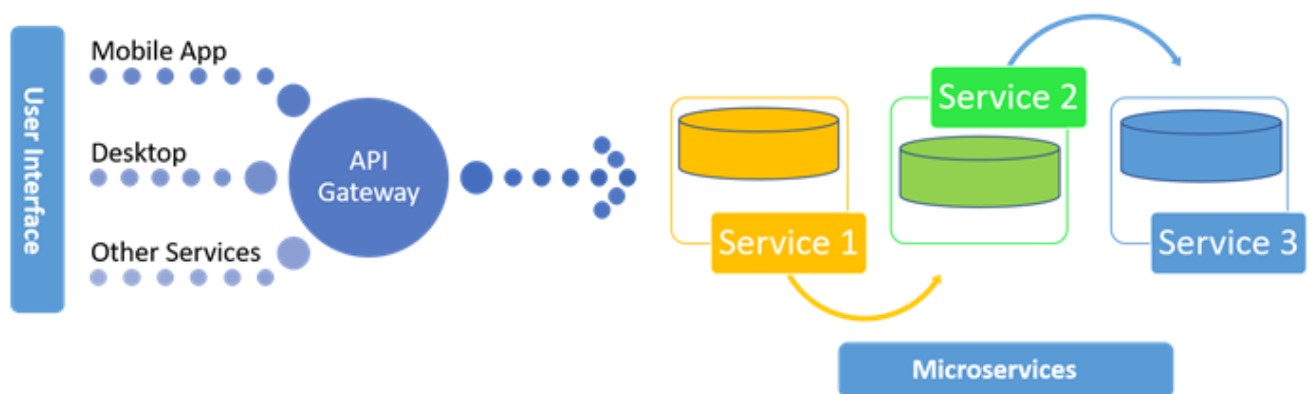


Рисунок 1.5 – Використання API шлюзу в мікросервісній архітектурі

1.4. Обґрунтування вибору інструментальних засобів.

Найголовнішим питанням перед початком роботи над програмним продуктом є вибір інструментальних засобів розробки, а саме мови програмування, платформи та фреймворків, тощо.

Обираючи мову програмування було розглянуто різні варіанти, проте було віддано перевагу платформі .NET для розробки на мові програмування C# (F# для певних компонентів) від компанії Microsoft. Однією з основних причин вибору мови програмування C# для розробки інформаційної системи є її багатофункціональність, надійність, а також велика кількість матеріалів та

прикладів у вільному доступі. Такий вибір надає можливість реалізації усіх частин системи за допомогою різних фреймворків платформи .NET, проте виключно на C# без використання інших мов програмування, в тому числі JavaScript у вебчастині системи.

Використання єдиної мови програмування для усіх компонентів системизначно спрощує процес розробки та підтримки, оскільки коли вся команда повністю розуміє будь яку частину програми, немає необхідності у окремому залученні спеціалістів з певної мови програмування чи фреймворку.

Ключовими перевагами платформи .NET є висока продуктивність, швидкість роботи програмних продуктів, вбудовані засоби для забезпечення захисту даних, а також гнучкість у виборі середовища розгортання. Систему, що розроблена на цій платформі, легко розгортати як у локальній інфраструктурі клініки, так і в хмарних сервісах, таких як Microsoft Azure, що дозволяє використовувати Azure SQL, App Services, Kubernetes Services для швидкого масштабування та управління ресурсами.

Розробку серверної частини буде здійснено на основі фреймворку ASP.NET Core, який дозволяє створювати високопродуктивні вебзастосунки, підтримує кросплатформову розробку, завдяки чому стає можливим розгортання на пристроях як з операційною системою Windows, так і Linux. Крім того, ASP.NET є однією з найшвидших серверних платформ завдяки оптимізованій JIT-компіляції, підтримці асинхронного програмування, ефективній роботі з пам'яттю, тощо. Дослідження показують, що серверні програми на базі ASP.NET працюють швидше за решту альтернатив, таких як Node.js, Java Spring, тощо. [11]

Неможливо уявити розробку системи на основі мікросервісної архітектури без використання засобів контейнеризації. Контейнеризація – це технологія віртуалізації на рівні операційної системи, що дозволяє запускати програмне забезпечення в ізольованих середовищах, які називаються контейнерами. Контейнер містить застосунок разом із усіма залежностями, бібліотеками та конфігураційними файлами, необхідними для виконання, що забезпечує стабільну роботу незалежно від операційної системи або інфраструктури, на якій

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						23
Змн.	Арк.	№ докум.	Підпис	Дата		

розгортається такий застосунок [12]. Docker – найпопулярніший та один з найпотужніших інструментів управління контейнерами. В мережі наявна велика кількість прикладів, а також детальна документація інструменту Docker, що значно допомагає в процесі розробки.

Контейнеризація значною мірою спрощує процеси розгортання, підтримки та масштабування програмних продуктів, забезпечує уніфікацію середовища запуску й виконання програми. Крім того, модуль, що виконується у контейнері є ізольованим від інших модулів, а також від операційної системи фізичного пристрою (комп'ютера, сервера, тощо), що є позитивним фактором безпеки. Контейнеризація також гарно впливає на розподіл навантаження, шляхом запуску додаткових екземплярів того чи іншого мікросервісу при зростанні обсягів трафіку на ньому.

Для розробки клієнтського вебзастосунку буде використано фреймворк Blazor, який дозволяє створювати інтерактивні вебзастосунки за допомогою C#. Такий підхід з'явився після релізу WebAssembly. WebAssembly – це технологія, що дозволяє виконувати код, який написаний на мовах програмування C, C++, Rust або C#, безпосередньо в браузері з високою продуктивністю, близькою до звичайного виконання [13]. WebAssembly дозволяє браузерам запускати компільований код, що значно покращує швидкість і ефективність виконання вебзастосунків, у порівнянні з традиційними JavaScript-застосунками. Розробка на Blazor замість звичайного JavaScript також спрощує процес розробки та забезпечує єдність технологій, оскільки усю кодову базу буде реалізовано на мові C#.

Розробку настільного застосунку для персоналу буде здійснено за допомогою WPF. Не дивлячись на вік, WPF досі є потужним інструментом для створення сучасних графічних інтерфейсів користувача на платформі .NET. Він дозволяє створювати багатофункціональні та візуально привабливі застосунки для ОС Windows, підтримує використання різних анімацій, мультимедійних елементів, складних векторних графічних елементів, а також забезпечує зручні засоби для створення багатоканальних інтерфейсів. Для створення інтерфейсу користувача у WPF використовується мова розмітки XAML.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						24
Змн.	Арк.	№ докум.	Підпис	Дата		

Для розробки WPF застосунків використовується архітектурний паттерн MVVM, який вже став стандартним підходом розробки у цьому фреймворку. Цей шаблон забезпечує чітке розділення інтерфейсу користувача та бізнес логіки застосунку. Основними компонентами є Model (сутність даних, їх обробка, безпосередня взаємодія з базою даних через API), View (представлення, інтерфейс користувача, що не містить бізнес-логіки) та ViewModel (посередник між представленням та моделлю, що містить логіку взаємодії з даними та забезпечує оновлення та відображення актуальних даних за допомогою механізму Data Binding) [22].

Зберігання даних здійснюватиметься в Microsoft SQL Server. SQL Server – це надійна реляційна система керування базами даних, що підтримує високу продуктивність, масштабованість та має гарні показники безпеки даних. Вона дозволяє ефективно працювати з великими обсягами даних та забезпечує можливості для резервного копіювання, відновлення та масштабування. Реляційні бази даних використовують для зберігання та управління даними, які мають чітко визначену структуру та зв'язки між сутностями, що чудово підходить в умовах інформаційної системи.

Однією з найбільших переваг MS SQL є розширена підтримка механізмів безпеки. SQL Server надає потужні інструменти для шифрування даних на рівні бази даних (Transparent Data Encryption, TDE) та контролю доступу. Враховуючи необхідність надійного зберігання персональної та конфіденційної інформації користувачів, фактор безпеки є надзвичайно пріоритетним у виборі СКБД.

Для зручної взаємодії з базою даних було вирішено використати концепцію ORM. Цей підхід до взаємодії з базою даних дозволяє використовувати об'єкти та їхні властивості замість звичайних SQL-запитів. ORM-інструменти автоматично перетворюють об'єкти в записи бази даних та навпаки, що спрощує роботу з даними та дозволяє зменшити кількість коду, необхідного для виконання тих чи інших операцій у базі даних (створення, читання, оновлення, видалення). У C# одним найбільш поширеним способом взаємодії з даними є використання LINQ – інструменту, що дозволяє писати запити до бази даних у зручному та зрозумілому

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						25
Змн.	Арк.	№ докум.	Підпис	Дата		

для розробника синтаксисі, який автоматично буде конвертовано в SQL-запит програмою. Такий підхід дозволяє працювати з базою даних на рівні об'єктів, використовуючи усі переваги статичної типізації C#, що зменшує ризик помилок, а також робить програмний код більш зрозумілим.

В якості ORM було обрано Dapper – легкий і швидкий ORM для .NET, що дозволяє працювати з базою даних без надмірних витрат ресурсів. Навідміну від найбільш відомого ORM на платформі .NET Entity Framework, Dapper є мікро-ORM, основна перевага якого – висока продуктивність. Він працює майже так само швидко, як звичайний ADO.NET та забезпечує швидку і просту інтеграцію з базою даних. На відміну від Entity Framework, який автоматично будує складні запити та може спричиняти зайві витрати ресурсів, Dapper залишає контроль за SQL-запитами у руках розробника, що дозволяє оптимізувати запити відповідно до потреб системи.

Висновки до першого розділу

У першому розділі було проведено дослідження предметної області, в результаті якого визначено завдання випускної роботи, сформовано список основних задач предметної області, які підлягають автоматизації в інформаційній системі та виконано постановку задачі.

Під час аналізу існуючих аналогів програмного продукту було складено порівняльну таблицю, що дозволила виявити переваги та слабкі сторони різних систем. На основі отриманих результатів дослідження було визначено першочергові та другорядні функціональні можливості, які необхідно реалізувати в інформаційній системі для ефективного управління клінікою та автоматизації процесів обліку, а також низку помилок та проблем, яких варто уникнути під час розробки.

Для правильного вибору архітектурного шаблону програми було виконано дослідження та порівняння найбільш поширених патернів розробки програмного забезпечення. В результаті аналізу було обрано мікросервісну архітектуру, як оптимальний підхід для створення гнучкої, масштабованої та безпечної системи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						26
Змн.	Арк.	№ докум.	Підпис	Дата		

Мікросервісна архітектура дозволить ефективно організувати взаємодію між компонентами системи, забезпечуючи можливість масштабування, легку та зручну підтримку.

Обираючи стек технологій для реалізації програмного продукту, прийнято рішення про використання мови програмування C# як потужного та універсального інструменту розробки усіх компонентів інформаційної системи управління та автоматизації обліку стоматологічної клініки, зокрема серверної частини з використанням фреймворку ASP.NET Core, клієнтської web частини з використанням фреймворку Blazor та клієнтського настільного застосунку з використанням фреймворку WPF. В якості СКБД віддано перевагу MS SQL Server, як надійній, безпечній та продуктивній системі управління базами даних.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						27
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 2. ПРОЄКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ СТОМАТОЛОГІЧНОЇ КЛІНІКИ

2.1. Специфікація вимог до програмного продукту

Розпочати проєктування програмного комплексу варто з визначення його функціональних можливостей та опису варіантів взаємодії користувачів із системою. Чітке формулювання функціональних та нефункціональних вимог до майбутньої системи є ключовою складовою успішного проєктування інформаційної системи управління та автоматизації обліку стоматологічної клініки. На цьому етапі буде визначено набір обов'язкових до реалізації функцій, обмеження, характеристики та інші важливі критерії для досягнення зручності у використанні, надійності та ефективності роботи програмного комплексу.

Бізнес вимоги. Метою створення системи є автоматизація процесів управління клієнтами та обліку процедур, виконаних у стоматологічній клініці. Головною ціллю є підвищення ефективності роботи клініки завдяки зменшенню ручної роботи з документацією, мінімізації впливу людського фактору, а також об'єднанню всіх бізнес-процесів у єдину цифрову екосистему.

Проєкт передбачає встановлення настільного застосунку на робочі комп'ютери працівників клініки, за допомогою якого буде здійснюватися облік пацієнтів, історій їхніх відвідувань та виконаних процедур, а також керування розкладом лікарів. Для пацієнтів передбачається вебзастосунок, що дозволяє переглядати доступні години для візиту, власну історію відвідувань, а також здійснювати онлайн-запис на прийом через сторінку електронної черги. Це значно зменшить навантаження на адміністративний персонал.

Вимоги користувачів та персоналу. Користувачі системи (пацієнти клініки) повинні мати можливість записатися до спеціаліста онлайн, редагувати чи скасовувати свої записи, переглядати та редагувати власні персональні дані, а також мати безперешкодний доступ до історії хвороб і наданих медичних послуг. Персонал клініки потребує можливості диференційованого доступу до

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						28
Змн.	Арк.	№ докум.	Підпис	Дата		

функціоналу системи залежно від посади. Працівники повинні мати інструменти для створення й редагування бази медичних послуг із вказанням їх вартості, ведення бази лікарів із розкладом роботи з урахуванням відпусток, лікарняних та вихідних днів, а також бази пацієнтів із усією необхідною інформацією – історією хвороб, діагнозами, протипоказаннями тощо.

Передбачена можливість ручного запису пацієнтів, редагування та видалення записів, автоматизованого розрахунку вартості послуг і формування квитанцій з фіксацією в базі даних. Також необхідне регулярне резервне копіювання даних.

Характеристика об'єкта комп'ютеризації. Основні користувачі системи – персонал та клієнти клініки. Для персоналу система надає можливість зручного ведення обліку наданих послуг та забезпечує автоматизацію процесу запису пацієнтів, керування розкладом роботи лікарів та формування необхідної документації (історія відвідувань, медичні картки, фінансові звіти, інвентаризація, тощо). Для клієнтів система надає можливість зручного онлайн-запису на прийом, перегляду доступних дат прийому, а також забезпечує отримання нагадувань про майбутні візити та збереження історії лікування. Це дозволяє уникнути незручностей, пов'язаних з необхідністю телефонувати або особисто відвідувати клініку, щоб записатися на прийом.

Функціональні вимоги. Система має містити механізми аутентифікації та авторизації, що дозволяють визначити особу користувача та рівень його доступу. Важливим є забезпечення контролю прав доступу до інформації та функціоналу залежно від ролі користувача. Дані повинні зберігатися надійно, із можливістю резервного копіювання. Увесь необхідний функціонал для роботи з даними має бути реалізований у повному обсязі. Система повинна дотримуватись політики конфіденційності даних пацієнтів, зберігати їх у захищеному вигляді та забезпечувати можливість повного видалення інформації про пацієнта. Також важливо, щоб система підтримувала відокремлення робочих просторів для кожної клініки в межах загальної інфраструктури.

Нефункціональні вимоги. З точки зору зручності сприйняття, система повинна бути простою в освоєнні — орієнтовний час навчання користувача не

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						29
Змн.	Арк.	№ докум.	Підпис	Дата		

повинен перевищувати 2–4 години. Час відгуку для типових задач не має перевищувати 10 секунд, а для нетипових — 20 секунд. При слабкому інтернет-з'єднанні час очікування не повинен бути більшим за 30 секунд.

Технічне обслуговування системи не повинно перевищувати 6% від загального часу роботи. Середній час безвідмовної роботи системи має становити не менше 15 робочих днів. При цьому допустимий рівень помилок — не більше однієї на кожні десять тисяч рядків коду. Система має підтримувати не менше 40 активних користувачів, які одночасно працюють із єдиною базою даних.

Система повинна забезпечувати можливість горизонтального масштабування, тобто розширення функціональних можливостей без порушення стабільності роботи. Також передбачається можливість збільшення серверних ресурсів для покращення продуктивності.

Вимоги до апаратного забезпечення. Серверна частина має працювати пристрої, який має процесор з щонайменше чотирма ядрами, не нижче рівня Intel Core i5 або AMD Ryzen 5. Обсяг оперативної пам'яті — не менше 16 ГБ. Накопичувач повинен бути SSD об'ємом від 512 ГБ. Також обов'язковим є стабільне інтернет-з'єднання зі швидкістю не менше 100 Мбіт/с і використання сучасної серверної операційної системи, такої як Windows Server 2019 або Ubuntu Server 22.04 LTS. Сервер має бути підключений до системи безперебійного живлення (UPS).

Робочі комп'ютери персоналу повинні мати процесор не нижче Intel Core i3 або аналогічний, щонайменше 8 ГБ оперативної пам'яті, SSD на 128 ГБ і операційну систему Windows 10 або новішу. Також необхідний доступ до локальної мережі або стабільне інтернет-з'єднання. Для клієнтів достатньо сучасного браузера (Google Chrome, Firefox, Safari або Edge останніх двох версій) та інтернет-з'єднання зі швидкістю не менше 5 Мбіт/с.

2.2. Визначення варіантів використання та проєктування об'єктно-орієнтованої структури системи

Опираючись на сформовані вимоги можна перейти до безпосереднього моделювання варіантів використання, у результаті чого буде побудовано діаграми

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						30
Змн.	Арк.	№ докум.	Підпис	Дата		

варіантів використання (діаграма прецедентів, англ. Use Case Diagram), які відображатимуть всі типові сценарії взаємодії користувачів із системою, а також буде сформовано об'єктно-орієнтовану структуру, що дозволить описати всі сутності, їхні властивості та зв'язки між ними.

Так як передбачається спільне використання системи як пацієнтами, так персоналом та адміністрацією клініки, доцільним є розділення користувачів за ролями та моделювання варіантів використання окремо для кожної ролі. Отримані діаграми прецедентів зображено на рисунках 2.1 – 2.2, детальний опис представлено у таблицях 2.1 та 2.2:

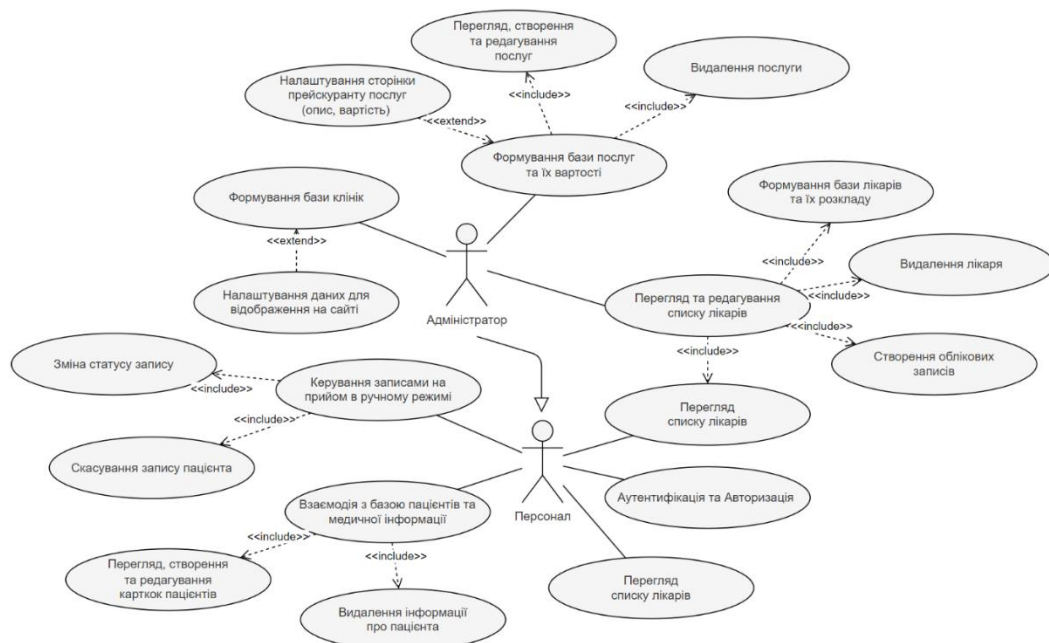


Рисунок 2.1 – Use-case діаграма для персоналу

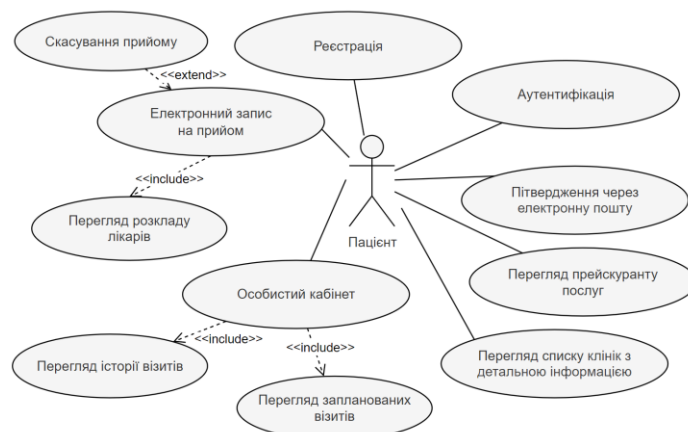


Рисунок 2.2 – Use-case діаграма для пацієнтів

Таблиця 2.1

Короткий опис акторів

Актор	Опис
Пацієнт	Використовує систему для запису на прийом, перегляду розкладу лікарів, преїскуранту послуг та загальнодоступної інформації.
Адміністратор	Керує обліковими записами й розкладом лікарів, обліковими записами пацієнтів та інформацією про них. Займається управлінням інформацією, що відображається на сайті клініки, контролює роботу системи.
	Має повний доступ як до власного набору функціоналу, так і до функціоналу, що доступний усьому персоналу, тобто такий користувач може безперешкодно користуватися тими ж самими функціями системи, що й звичайний працівник.
Працівник (Персонал)	Переглядає свій розклад та розклад колег, отримує інформацію про пацієнтів та їх записи на прийом. Вносить дані про виконані процедури.

Таблиця 2.2

Опис варіантів використання

Актор	Назва	Опис
1	2	3
Пацієнт	Реєстрація	Пацієнт створює свій обліковий запис шляхом заповнення форми для доступу в особистий кабінет.
	Аутентифікація	Пацієнт заповнює форму облікових даних та заходить у свій особистий кабінет.
	Перегляд списку клінік	Пацієнт може переглянути список клінік та детальну інформацію на відповідний вебсторінці.
	Перегляд преїскуранту послуг	Пацієнт може переглянути список послуг, дізнатись їх вартість та детальну інформацію на відповідний вебсторінці.

1	2	3
Пацієнт	Електронний запис на прийом	Пацієнт обирає лікаря, дату та час для запису на прийом через вебінтерфейс, створює відповідний запис шляхом підтвердження форми.
	Особистий кабінет	Особистий кабінет користувача, де відображається історія візитів та найближчі заплановані візити до клінік
	Скасування запису на прийом	Пацієнт скасовує запланований візит через вебінтерфейс.
Адміністратор	Формування бази лікарів	Адміністратор створює та має можливість редагування облікових записів лікарів.
	Формування бази послуг послуг та їх вартості	Адміністратор визначає список послуг, їх вартість. Інформація відображається клієнтам на вебсторінці.
	Налаштування сторінки преїскуранту послуг	Використовуючи інтерфейс програми, Адміністратор налаштовує відображення тих чи інших послуг пацієнтам на вебсторінці клініки.
	Формування бази клінік	Адміністратор налаштовує список клінік та інформацію, яка відображається клієнтам на вебсторінці.
	Налаштування сторінки клінік	Використовуючи інтерфейс програми, Адміністратор налаштовує відображення тих чи інших клінік пацієнтам на вебсторінці.
	Налаштування ролей та рівня доступу	Адміністратор налаштовує рівень доступу та роль користувачів через інтерфейс застосунку.
Персонал та Адміністратор	Перегляд списку лікарів	Персонал має доступ до перегляду списку лікарів.

1	2	3
Персонал та Адміністратор	Перегляд списку лікарів	Персонал має доступ до перегляду списку лікарів.
	Перегляд, створення та редагування облікових записів пацієнтів	Персонал займається веденням бази даних пацієнтів, внесенням та оновленням інформації.
	Автоматизований розрахунок вартості послуг	Система автоматично розраховує вартість послуг та загальну суму до сплати пацієнту за візит.
	Скасування запису пацієнта.	Персонал може скасувати запис на прийом пацієнта
	Формування бази пацієнтів	Персонал має можливість ручного створення та редагування облікових записів пацієнтів.
Система	Автоматична зміна статусу для пропущених записів	Система автоматично встановлює статус «Пропущено» для таких записів, що мають статус «Заплановано» через 10 хвилин після запланованого часу початку.
	Автоматичне скасування непідтверджених записів	Система автоматично скасовує такі записи на прийом, що не були підтверджені через електронну пошту протягом 10 хвилин після створення.

На рисунку 2.3 зображено узагальнений алгоритм взаємодії користувачів з системою, що включає в себе відправку запитів, їх обробку, отримання інформації або внесення змін в базу даних, формування та відправку відповіді на запит. Діаграму можна вважати спільною для більшості прецедентів, адже практично всі вони працюють за схожим алгоритмом.

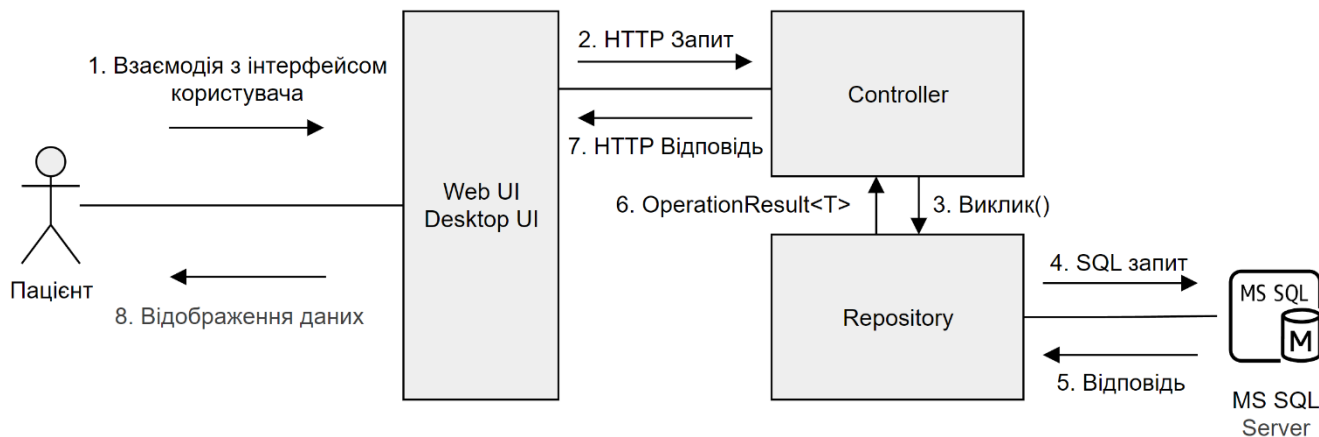


Рисунок 2.3 – Узагальнена діаграма взаємодії системи

2.3. Розробка бази даних системи

В якості сховища для зберігання даних системи використовується реляційна база даних на платформі Microsoft SQL Service. Такий підхід забезпечує ефективну організацію даних для взаємопов'язаних сутностей системи, підтримує складні запити, гарантує цілісність та безпеку даних за допомогою вбудованих механізмів. У базі даних зберігатимуться дані про клініки, пацієнтів, лікарів, активні сесії, розклад роботи та записи на прийом, преїскурант медичних послуг, налаштування контенту вебсайту. Для цього створено 10 відповідних таблиць:

1. AppUsers – зберігає інформацію про всіх користувачів системи (пацієнтів, лікарів, адміністраторів);
2. Clinics – зберігає дані про клініки;
3. UserClinics – реалізує зв'язок багато-до-багатьох між користувачами та клініками;
4. Services – зберігає доступні послуги клініки;
5. TimeSlots – зберігає зайняті часові слоти записів на прийом;
6. Appointments – інформація про прийоми;
7. UserSessions – зберігає інформацію про сесії користувачів для авторизації;
8. ClinicContent – зберігає детальну інформацію про клініки;
9. ServiceContent – зберігає детальну інформацію про послуги;
10. ClinicImages – зберігає зображення клінік, що відображаються на сайті.

Опис кожної з таблиць наведено у таблицях 2.3 – 2.12:

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						35
Змн.	Арк.	№ докум.	Підпис	Дата		

Таблиця 2.3

Опис таблиці AppUsers

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор користувача
Email	NVARCHAR(256)	Адреса електронної пошти
EmailConfirmationToken	NVARCHAR(256)	Токен підтвердження електронної пошти
PhoneNumber	NVARCHAR(256)	Номер телефону
FirstName	NVARCHAR(50)	Ім'я користувача
LastName	NVARCHAR(50)	Прізвище користувача
IsEmailConfirmed	BIT	Прапорець підтвердження електронної пошти
IsRegistered	BIT	Прапорець реєстрації користувача
IsPasswordConfirmed	BIT	Прапорець підтвердження пароля
IsActive	BIT	Прапорець активності
PasswordHash	NVARCHAR(256)	Пароль користувача
BirthDate	DATE	Дата народження користувача
CreationDate	DATETIMEOFFSET	Дата створення облікового запису
WorkspaceRole	INT	Роль користувача в системі

Таблиця 2.4

Опис таблиці Appointments

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор
AppointmentConfirmationToken	NVARCHAR(256)	Токен підтвердження запису
EmployeeId	BIGINT	Зовнішній ключ до AppUsers
PatientId	BIGINT	Зовнішній ключ до AppUsers
TimeSlotId	BIGINT	Зовнішній ключ до TimeSlots
ClinicId	VARCHAR(50)	Зовнішній ключ до Clinics
ServiceId	VARCHAR(50)	Зовнішній ключ до Services
CreationTimestamp	DATETIMEOFFSET	Дата створення запису
Discount	DECIMAL	Розмір знижки
TotalPrice	DECIMAL	Загальна вартість
Status	INT	Статус запису

Таблиця 2.5

Опис таблиці UserSessions

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор
UserId	BIGINT	Зовнішній ключ до AppUsers
AccessToken	NVARCHAR	Токен доступу
RefreshToken	NVARCHAR(256)	Токен оновлення
ExpiryTime	DATETIMEOFFSET	Дата автоматичного завершення сесії
CreationDate	DATETIMEOFFSET	Дата створення сесії

Таблиця 2.6

Опис таблиці TimeSlots

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор
UserId	BIGINT	Зовнішній ключ до AppUsers
ClinicId	VARCHAR(50)	Зовнішній ключ до Clinics
StartTimestamp	DATETIMEOFFSET	Timestamp початку запису
EndTimestamp	DATETIMEOFFSET	Timestamp закінчення запису

Таблиця 2.7

Опис таблиці UserClinics

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор
UserId	BIGINT	Зовнішній ключ до AppUsers
ClinicId	VARCHAR(50)	Зовнішній ключ до Clinics
AccessLevel	INT	Рівень доступу користувача в конкретній клініці

Таблиця 2.8

Опис таблиці Services

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор
Name_en	NVARCHAR(100)	Назва послуги англійською
Name_uk	NVARCHAR(100)	Назва послуги українською
Price	DECIMAL	Вартість послуги
ApproximateTime	DECIMAL	Тривалість процедури
IsAvailable	BIT	Прапорець доступності

Опис таблиці Clinics

Поле	Тип даних	Опис
Id	VARCHAR(50)	Унікальний текстовий ідентифікатор.
Name_en	NVARCHAR(100)	Назва клініки англійською
Name_uk	NVARCHAR(100)	Назва клініки українською
Address_en	NVARCHAR(256)	Адреса клініки англійською
Address_uk	NVARCHAR(256)	Адреса клініки українською
PhoneNumber	NVARCHAR(50)	Контактний номер телефону клініки
IsActive	BIT	Прапорець активності

Опис таблиці ClinicContent

Поле	Тип даних	Опис
Id	BIGINT	Ідентифікатор користувача
ClinicId	NVARCHAR(50)	Зовнішній ключ до Clinics
Description_en	NVARCHAR(MAX)	Опис клініки англійською
Description_uk	NVARCHAR(MAX)	Опис клініки українською
ShortDescription_en	NVARCHAR(MAX)	Короткий опис клініки англійською
ShortDescription_uk	NVARCHAR(MAX)	Короткий опис клініки українською
MetaTitle_en	NVARCHAR(MAX)	Мета-заголовок клініки англійською
MetaTitle_uk	NVARCHAR(MAX)	Мета-заголовок клініки українською
MetaDescription_en	NVARCHAR(MAX)	Мета-опис клініки англійською
MetaDescription_uk	NVARCHAR(MAX)	Мета-опис клініки українською
WorkingHours_en	NVARCHAR(500)	Графік роботи англійською
WorkingHours_uk	NVARCHAR(500)	Графік роботи українською
LastUpdated	DATETIMEOFFSET	Дата та час останнього редагування
IsActive	BIT	Прапорець активності

Таблиця 2.11

Опис таблиці ClinicImages

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор користувача
ClinicId	NVARCHAR(50)	Зовнішній ключ до Clinics
ImageType	INT	Тип зображення
ImageData	VARBINARY(MAX)	Тіло зображення
ContentType	NVARCHAR(100)	Тип зображення
FileSize	BIGINT	Розмір зображення
UploadDate	DATETIMEOFFSET	Дата завантаження
IsActive	BIT	Прапорець активності

Таблиця 2.12

Опис таблиці ServiceContent

Поле	Тип даних	Опис
Id	BIGINT	Унікальний ідентифікатор користувача
ServiceId	NVARCHAR(50)	Зовнішній ключ до Services
Description_en	NVARCHAR(MAX)	Опис клініки англійською
Description_uk	NVARCHAR(MAX)	Опис клініки українською
MetaTitle_en	NVARCHAR(MAX)	Мета-заголовок клініки англійською
MetaTitle_uk	NVARCHAR(MAX)	Мета-заголовок клініки українською
MetaDescription_en	NVARCHAR(MAX)	Мета-опис клініки англійською
MetaDescription_uk	NVARCHAR(MAX)	Мета-опис клініки українською
LastUpdated	DATETIMEOFFSET	Дата та час останнього редагування
IsActive	BIT	Прапорець активності

Основна взаємодія з БД відбувається на рівні системи шляхом відправки raw-SQL запитів, отримання відповідей та мапінгу даних (англ. Data Mapping) – процесу зіставлення даних між двома різними системами або джерелами, в нашому випадку база даних та API, для подальшого використання та відображення користувачу [17]. Проте, для таких операцій, як створення запису на прийом, де необхідно вносити зміни одразу в декілька таблиць, критично важливим є використання транзакцій та

«відкату» всіх змін в даних у випадку неуспішного результату на певному етапі виконання транзакції.

Транзакція – це сукупність однієї або кількох SQL-операцій, як одна логічна одиниця роботи з даними. [14] Використання такого підходу забезпечить консистентність даних та запобігатиме виникненню помилок в базі даних, адже дані зберігаються лише після успішного завершення всіх операцій однієї транзакції. Для зручності під час розробки, також щоб уникнути зайвого ускладнення програмного коду, на рівні бази даних було створено низку збережених процедур:

1. CreateUser

Процедура реєстрації користувача, що включає в себе перевірку наявності створеного раніше акаунту зі значенням 0 в полі IsRegistered, що означає, що користувач записався на прийом без реєстрації облікового запису лише за електронною поштою. Для процедури визначено власні можливі коди результату виконання: 0 – операція успішна; 51009 – акаунт користувача з вказаною електронною поштою вже зареєстрований;

2. CreateTimeSlot

Процедура створення зайнятого часового проміжку. Використовується в процедурах створення запису на прийом, включає в себе перевірку на перетин з іншими часовими проміжками. Можливі коди результату виконання: 0 – операція успішна; 51000 – вказаний часовий слот перетинається з іншими зайнятими слотами.

3. CreateAppointment_Unauthorized

Процедура створення запису на прийом без попередньої реєстрації користувача. Перевіряє усі дані, права доступу, створює запис в таблиці Users зі значенням 0 в полі IsRegistered, виконує перевірку лімітів кількості непідтверджених та активних записів на прийом для користувача. В рамках однієї транзакції створює записи в таблицях TimeSlot та Appointments, у випадку виникнення помилки на будь-якому етапі виконує «відкат» та повертає відповідний числовий код. Передбачає відправку листа підтвердження запису на електронну пошту в разі успішного виконання. Можливі коди результату виконання: 0 – операція успішна;

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						40
Змн.	Арк.	№ докум.	Підпис	Дата		

51001 – недостатньо прав доступу; 51002 – користувач не може записатись на прийом до самого себе; 51003 – перевищено ліміт непідтверджених записів; 51000 – вказаний часовий слот перетинається з іншими зайнятими слотами.

4. CreateAppointment_Authorized

Процедура створення запису на прийом з попередньою реєстрацією користувача. Призначена для використання у випадку, коли користувач увійшов у свій обліковий запис та оформлює новий запис на прийом. Виконує схожі перевірки та операції, як і процедура CreateAppointment_Unauthorized, проте має певні відмінності, зокрема В поле Status встановлюється значення 1 (Scheduled) замість 0 (WaitingForConfirm). Можливі коди результату виконання збігаються з процедурою CreateAppointment_Unauthorized.

5. UpdateAppointmentStatus

Процедура оновлення статусу запису на прийом. Виконує низку перевірок на існування, коректність вхідного параметру, що містить новий статус, а також перевіряє права доступу користувача. Можливі коди результату виконання: 0 – операція успішна; 51001 – недостатньо прав доступу; 51005 – запис на прийом не знайдено.

6. ManageUserClinicAccess

Процедура оновлення прав доступу (посади) користувачів у клініках. Виконує перевірки існування вказаних записів у таблицях бази даних, автоматично видаляє несумісні ролі (наприклад – паралельне призначення користувача лікарем та менеджером рецепції у різних клініках), залишаючи вказану у вхідних параметрах. Можливі коди результату виконання: 0 – операція успішна; 51006 – користувача не знайдено, 51007 – клініку не знайдено; 51008 – некоректне значення вхідного параметру.

Ці процедури покривають собою ті операції, у яких можуть потенційно виникнути помилки. Передбачивши максимальну кількість несприятливих варіантів розвитку подій, було сформовано список кодів помилок, які використовуються. Додатково на рівні бази даних реалізовано ряд обмежень та лімітів, таких як SQL Constraint, SQL Unique, а саме перевірка формату електронної

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						41
Змн.	Арк.	№ докум.	Підпис	Дата		

пошти, обмеження кількості однотипних ролей одного користувача у різних клініках, реалізація зв'язку «один до одного» для таблиць Clinic та ClinicContent, Serice та ServiceContent, TimeSlots та Appointments. Завдяки використанню таких обмежень забезпечується цілісність та консистентність інформації, що додається у базу даних.

2.4. Проєктування та реалізація алгоритмів роботи системи

На основі сформованих варіантів використання та прототипу бази даних розпочато процес проєктування системи та моделювання архітектури, використовуючи IDE Visual Studio 2022. Вбудований інструмент для роботи з діаграмами класів підтримує кодогенерацію, завдяки чому, моделюючи систему, автоматично створюються справжні класи з їх базовими реалізаціями.

В першу чергу було створено проєкт типу «Бібліотека класів», що міститиме усі сутності (рис. 2.4), а також пов'язані з ними допоміжні класи DTO – шаблон проєктування програмних систем, який використовується для передачі даних між підсистемами для спрощення операцій над даними на клієнтському боці, а також формуванні комплексних об'єктів даних для зменшення кількості необхідних API запитів для отримання потрібної інформації [15].

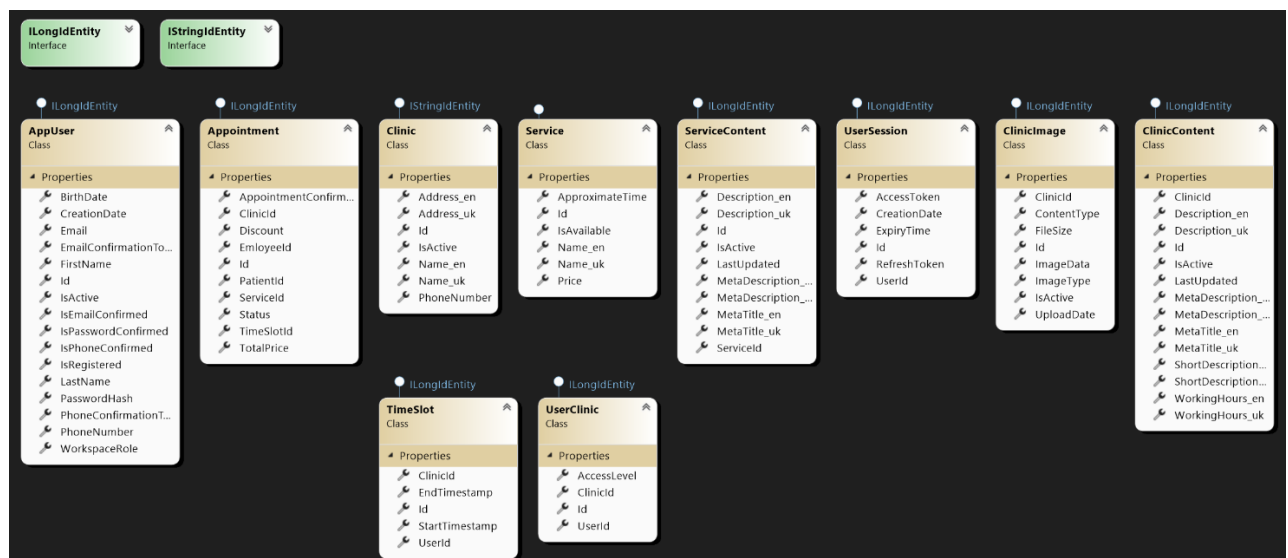


Рисунок 2.4 – Діаграма класів «Сутності системи»

Цей проєкт є самостійною бібліотекою класів, завдяки чому зводиться до нуля утворення зайвих та непотрібних взаємозв'язків між сутностями та бізнес

логікою. Спільні класи буде перевикористано в різних мікросервісах, проте вони будуть ізольовані. Однозначною перевагою використання такого підходу є неможливість внесення змін у спільні класи, адже активне редагування спільних класів може призвести до виникнення проблем у вже реалізованих модулях.

На діаграмі класів мікросервісу «Auth» (рис. 2.5) зображено робочі контролери, які безпосередньо здійснюють обробку HTTP запиту, викликають методи репозиторіїв для взаємодії з базою даних та формують тіло HTTP відповіді. За аналогічним алгоритмом працюють усі мікросервіси системи.

Особливістю мікросервісу «Auth» є наявність класів, що виконують стон-задачі, а також реалізація класу AuthService, який відповідає за генерування JWT токенів, їх валідацію та оновлення, а також авторизацію користувача за введеними логіном та паролем.

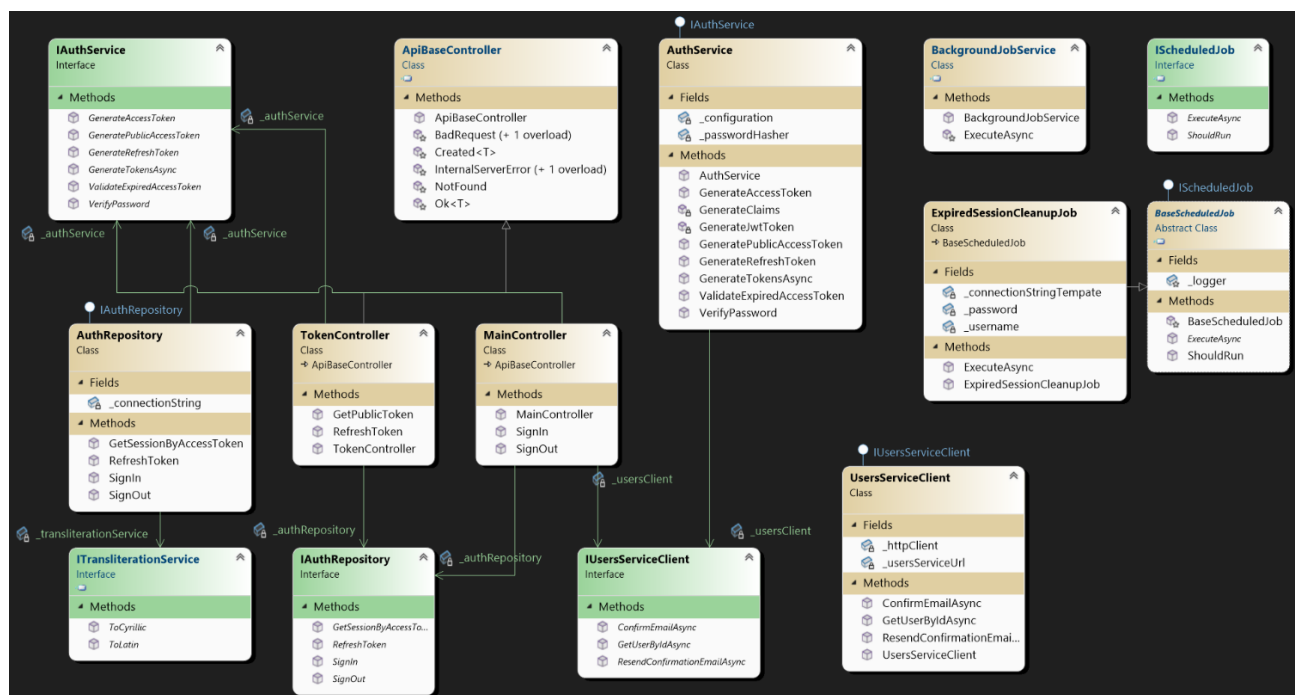


Рисунок 2.5 – Діаграма класів мікросервісу «Auth»

На рисунку 2.6 зображено діаграму класів мікросервісу User, де окрім контролерів можна побачити сервіс OTPService (One-Time-Password Service), який відповідає за генерування, перевірку та відправку одноразових паролів користувачам, чиї облікові записи були створені працівником клініки. Для відправки сервіс перевикористовує вже згаданий раніше EmailService, який

відповідає за розсилку листів на електронну пошту користувачів за допомогою протоколу SMTP.

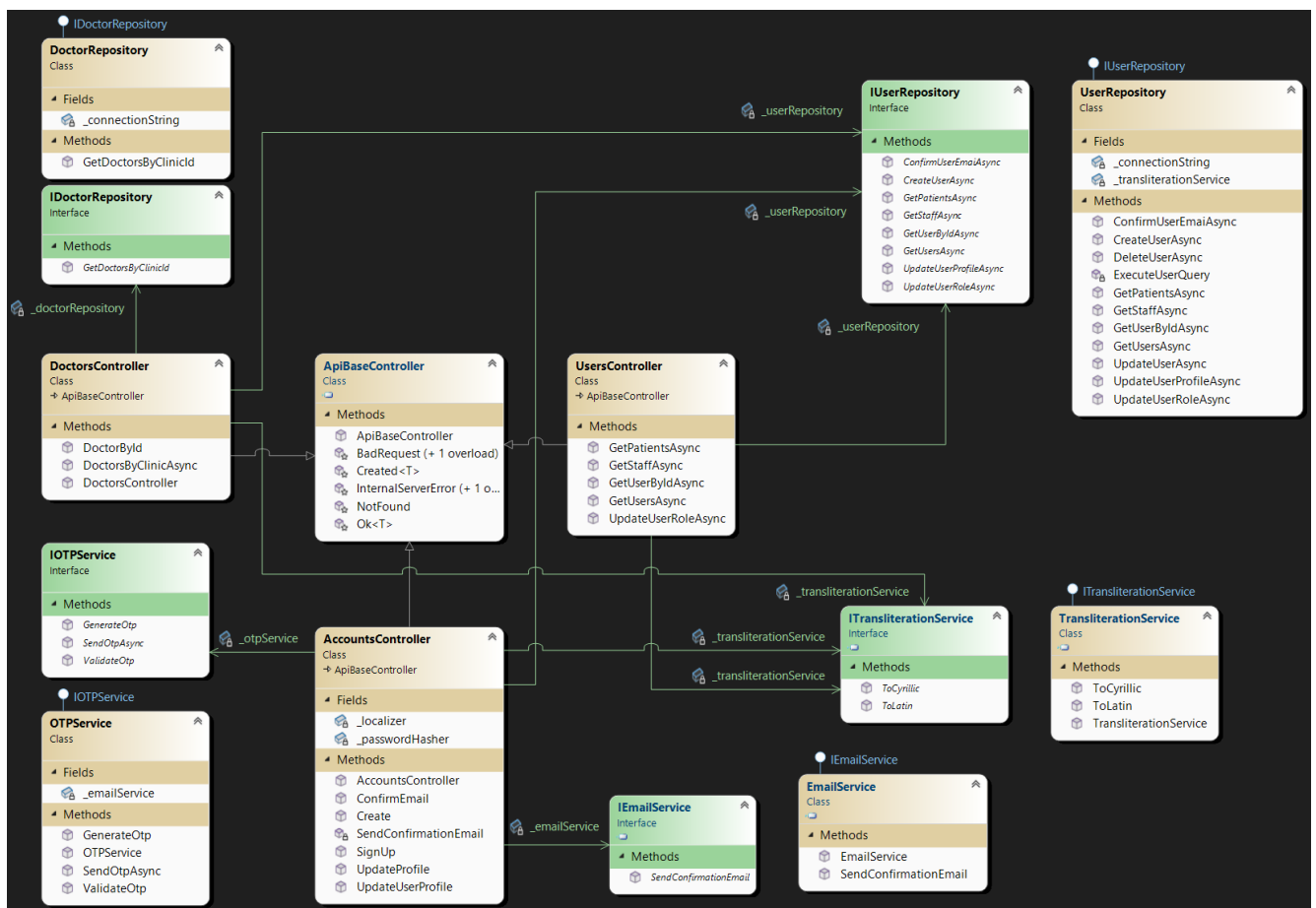


Рисунок 2.6 – Діаграма класів мікросервісу «Users»

На діаграмі класів мікросервісу «Schedule» (рис. 2.7) окрім основних класів контролерів представлено класи cron-завдань, а також класи TransliterationService та EmailService.

TransliterationService відповідає за автоматичну транслітерацію імен та прізвищ користувачів, завдяки чому не важливо яку мову використовує користувач – його ім'я та прізвище буде збережено в коректному вигляді згідно загальноприйнятого алгоритму транслітерації.

EmailService в даному випадку перевикористовується для відправки листів з підтвердженням записів на прийом тих користувачів, що не реєструвались у системі і записуються на прийом за допомогою електронної пошти.

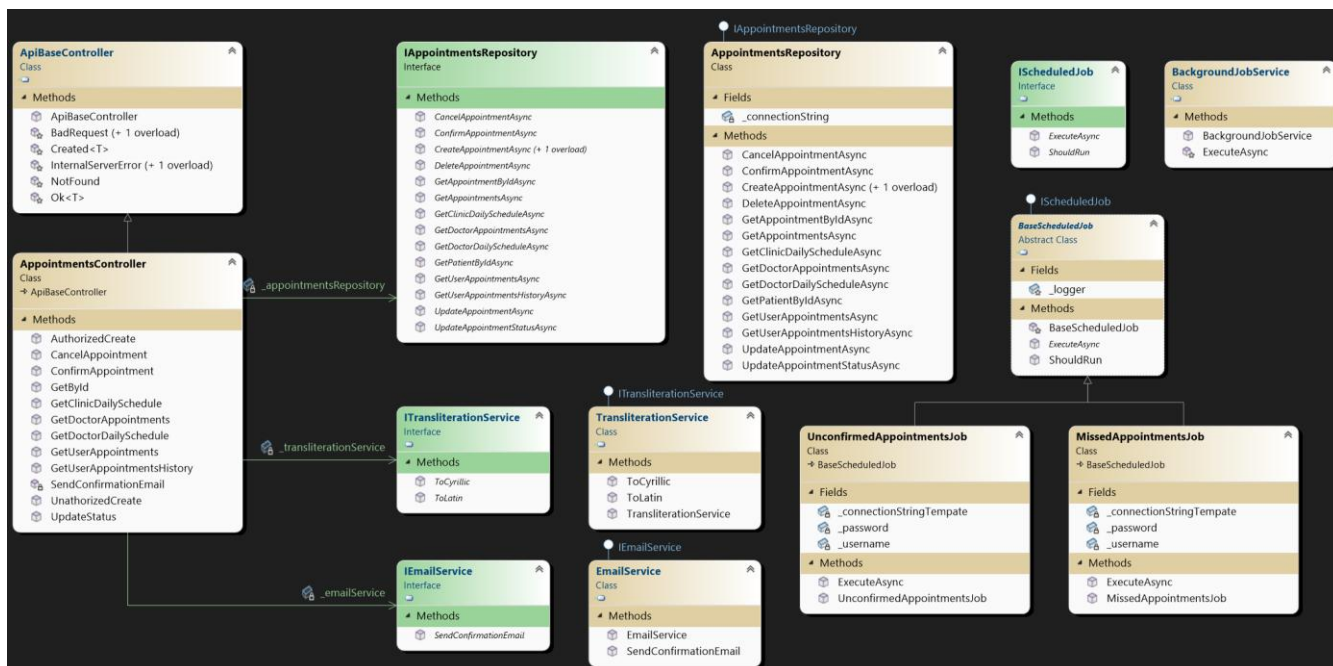


Рисунок 2.7 – Діаграма класів мікросервісу «Schedule»

На рисунку 2.8 представлено діаграму класів мікросервісу «Workspace», зокрема його контроллерів та репозиторіїв.

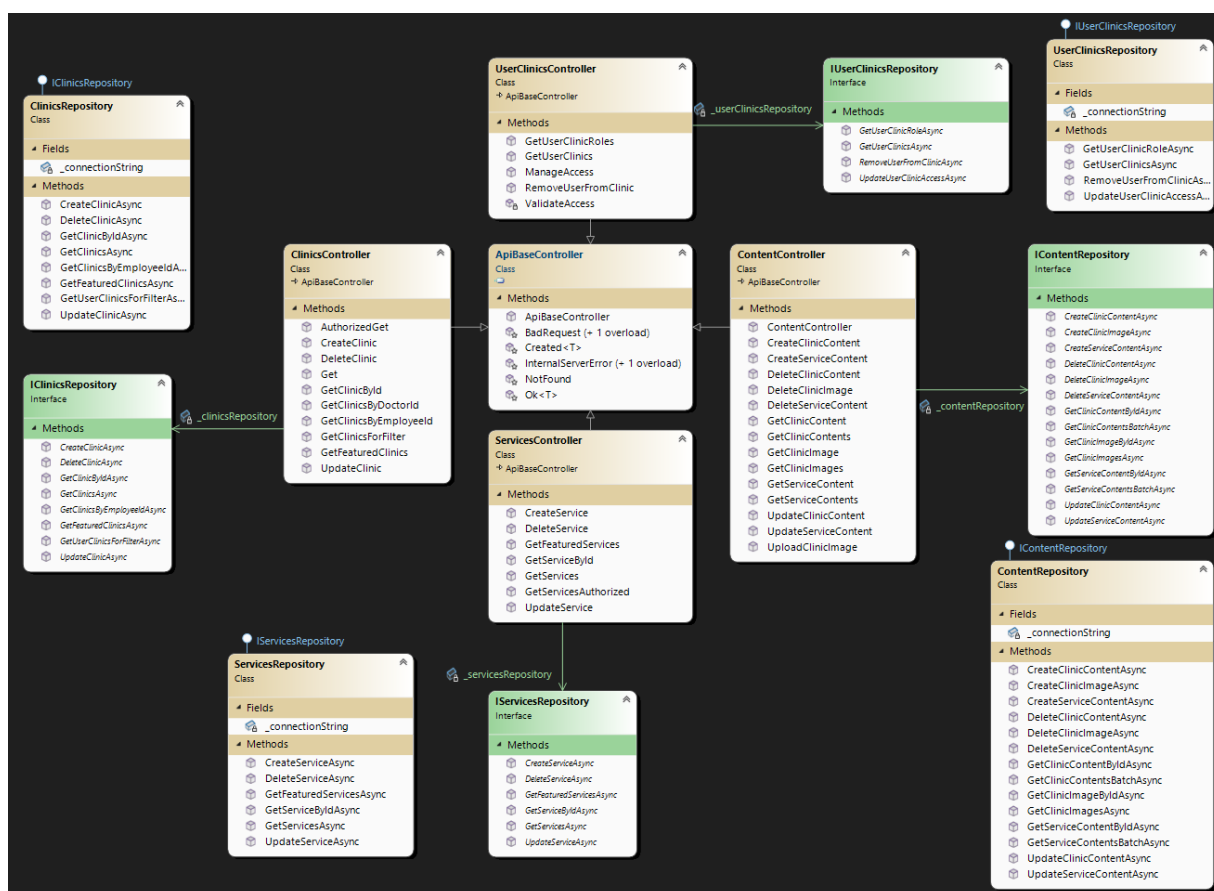


Рисунок 2.8 – Діаграма класів мікросервісу «Workspace»

На діаграмі компонентів (рис. 2.9) визначено такі складові: Ocelot API Gateway, екземпляр довільного мікросервіса та сервер бази даних. API Gateway включає в себе такі елементи, як Reverse Proxy, Route Configuration, Rate Limiting та Auth. Reverse Proxy – це «ретранслятор» запитів клієнтів із зовнішньої мережі на сервери внутрішньої мережі [21], який в поєднанні з механізмом Rate Limiting захищає систему від надмірного навантаженнями запитами та потенційних зловмисних атак.

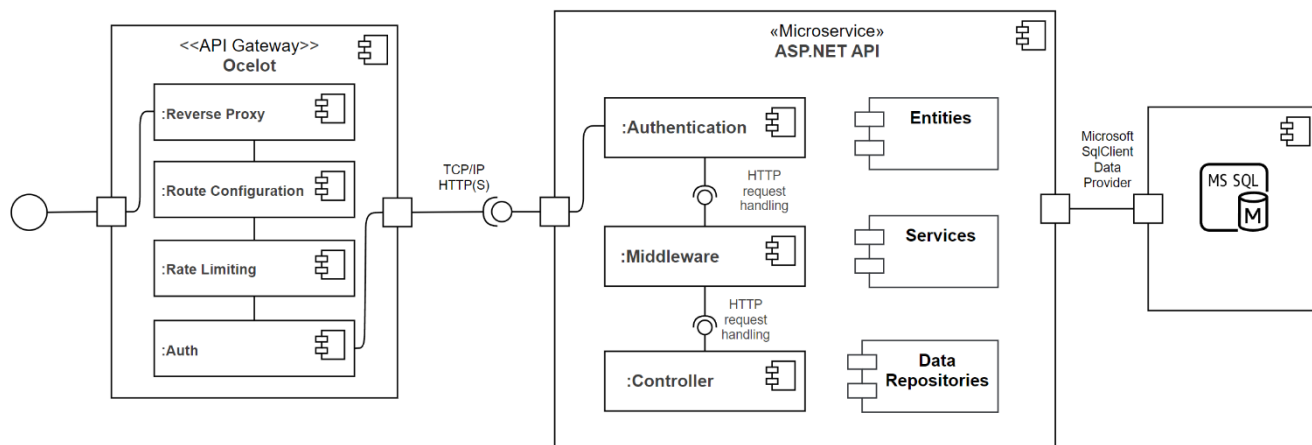


Рисунок 2.9 – Діаграма компонентів системи

Авторизація та аутентифікація користувачів відбувається як на рівні API Gateway, так і на рівні кожного мікросервісу. Усі мікросервіси мають схожу структуру та складаються з конвеєра обробки HTTP запиту, контролерів та допоміжних сервісів, репозиторіїв, тощо. З'єднання з базою даних встановлюється за допомогою Microsoft SqlClient Data Provider.

Після реалізації серверної частини було розпочато проектування клієнтського настільного застосунку. У розробці Windows застосунку активно застосовуються різні патерни проектування, зокрема MVVM для формування чистої архітектури проекту, та дотримання принципів реактивності, тобто здатності автоматичного оновлення інтерфейсу користувача у відповідь на зміни в даних.

Реактивність інтерфейсу користувача технічно досягається за допомогою реалізації інтерфейсу INotifyPropertyChanged класами ViewModel, або будь якими іншими, що виконують роль контексту даних (DataContext), а також Event-driven

архітектурою. Як тільки фіксується зміна даних у контексті даних – інтерфейс за допомогою події PropertyChanged автоматично повідомляє систему, що необхідно оновити той чи інший елемент інтерфейсу.

Основні принципи реалізації реактивного інтерфейсу користувача, а також використання згаданих архітектурних підходів та шаблонів проектування зображено на рисунку 2.10.

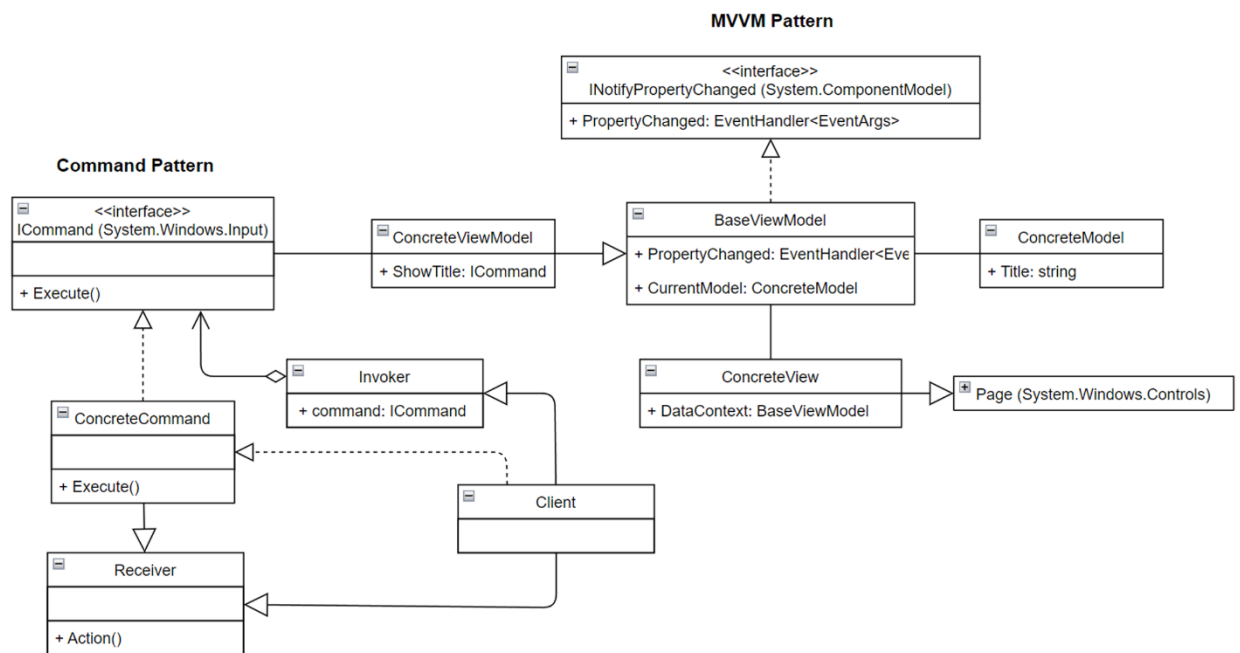


Рисунок 2.10 – Загальний вигляд архітектури WPF застосунку

Ключовою складовою коректної роботи реактивного інтерфейсу користувача є система прив'язки даних (DataBinding) – саме завдяки їй компоненти інтерфейсу користувача, такі як текстові поля, випадаючі списки, кнопки, тощо «підключаються» до контексту даних, що дозволяє їм перехоплювати події PropertyChanged та реагувати на зміну значень, а також змінювати дані безпосередньо в контекст, коли використовується режим «TwoWay». Замість класичної обробки взаємодії користувача з кнопками безпосередньо у code-behind класі вікна чи сторінки застосовується шаблон проектування Command, завдяки чому методи-обробники натискань на кнопки реалізуються також у класах ViewModel.

[illegible]

Важливим етапом розробки системи є реалізація механізму авторизації та аутентифікації. Для цього було підготовано відповідну таблицю бази даних `UserSessions`, яка зберігає записи про активні сесії користувачів та JWT токени, що закріплені в рамках тої чи іншої сесії. Будь-який API запит повинен супроводжуватись передачею відповідного HTTP заголовка авторизації. Якщо такий заголовок відсутній – HTTP запит буде відхилено ще на етапі API Gateway, щоб не навантажувати конкретний мікросервіс додатковими перевітками. У разі успішної аутентифікації користувача, на сервері генерується пара токенів: `Access` токен, який містить в собі зашифровану інформацію про користувача, а також `Refresh` токен – спеціальний JWT з більшим терміном життя для оновлення `Access` токена, коли у першого закінчується термін дії. Обидва токена записуються в базу даних та повертаються на клієнтську сторону незалежно від того, веб чи настільний застосунок є клієнтом. Прикріпивши отриманий `Access` токен в `Authorization` заголовок HTTP запита, користувач отримує доступ до бажаної інформації.

Для неавторизованих користувачів передбачено окремий незахищений API endpoint для отримання Public JWT токена. З його допомогою можна отримати базові дані для вебзастосунку, які відображаються користувачам на публічних сторінках вебсайту. Такий токен не має терміну придатності та не фіксується у базі даних, адже його можна вважати «одноразовим» та з його допомогою неможливо отримати доступ до жодних приватних даних.

Щоб гарантувати автоматичне обмеження доступу на неактуальних пристроях після закінчення терміну дії токенів, на сервері реалізовано Сron-задачу, яка періодично видаляє усі неактуальні сесії та токени з бази даних. Сron-задачі – це один зі способів автоматичного виконання команд чи сценаріїв з певною періодичністю або в певний визначений час [23].

Крім того, на рівні API Gateway реалізовано Middleware-компонент (додаток А), який перевіряє актуальність токена безпосередньо в момент HTTP запиту, коли JWT токен вважається таким, що пройшов усі перевірки. Завдяки цьому є можливість вручну завершувати ті сесії, для яких є підозра викрадення токена або ж ті, які були розпочаті на неперевіреному пристрої та не були коректно завершені.

На стороні клієнтського вебзастосунку, а також windows застосунку реалізовано алгоритми автоматичного оновлення Access токена з декілька хвилин до закінчення його терміну дії, завдяки чому поки користувач проявляє активність у застосунку, проблеми з раптовим завершенням сесії та необхідністю перезайти в обліковий запис не виникають. Вебзастосунок зберігає отримані токени в сховище cookies, а при наступному вході користувача виконується спроба оновлення збережених токенів та автоматичний вхід в обліковий запис у випадку, коли спроба була успішна. У Windows застосунку реалізовано аналогічний алгоритм збереження токенів у локальному сховищі та автоматичного входу при перезапуску застосунку.

Окрім авторизації, значну увагу приділено розподілу користувачів за клініками. Система передбачає роботу в мережі клінік, що мають спільну базу пацієнтів та послуг, адже такий підхід значно облегшує масштабування та розширення системи за потреби. Для його реалізації було створено таблицю

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						49
Змн.	Арк.	№ докум.	Підпис	Дата		

UserClinics, у якій зберігаються посади лікарів у тій чи іншій клініці. Також, закладено певні обмеження паралельного призначення рівня доступу (AccessLevel) користувачам у різних клініках, а саме: один користувач може мати рівень доступу працівника (Doctor або Receptionist) лише в одній клініці. Паралельно з цим, користувач може мати необмежену кількість записів з роллю адміністратора у інших клініках.

Такий розподіл рівня доступу користувачів є внутрішнім для клінік, де кожен користувач займає певну посаду у певній клініці. Крім цього реалізовано «глобальний» розподіл ролей – у таблиці AppUsers присутнє поле WorkspaceRole, яке допускає 3 можливі значення: Patient, Employee та Administrator. Перш ніж налаштовувати посаду користувача на рівні клінік, йому необхідно встановити роль Employee у таблиці AppUsers. Такий механізм допомагає розділити користувачів на працівників та пацієнтів в рамках однієї таблиці, що спрощує велику частину пов'язаних з нею операцій.

Запис на прийом до лікаря є головною функціональною можливістю системи. Для створення запису користувачу необов'язково реєструвати обліковий запис – достатньо вказати свої контактні дані та адресу електронної пошти на відповідному кроці форми, а система автоматично збереже їх до таблиці AppUsers з поміткою IsRegistered = false. На вказану адресу електронної пошти надійде лист підтвердження запису на прийом, який дійсний протягом 10 хвилин з моменту створення. Перейшовши за посиланням з електронного листа користувач підтверджує запис на прийом та відповідний статус встановлюється для запису у базі даних. Усі створені записи фіксуються в базу даних без можливості видалення для контролю кількості оформлених записів на прийом у кожного користувача, адже не можна допускати ситуацій, коли зловмисник анонімно «записався» на усі доступні години прийому без жодних обмежень.

Система допускає 5 активних записів на прийом для одного користувача, при цьому кількість непідтверджених записів на прийом не може перевищувати 2. Окрім того, реалізовано дві cron задачі (UnconfirmedAppointmentsJob та MissedAppointmentsJob).

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						50
Змн.	Арк.	№ докум.	Підпис	Дата		

Задача UnconfirmedAppointmentsJob періодично знаходить ті записи на прийом, з моменту створення яких пройшло більше 10 хвилин та вони все-ще мають статус «WaitingForConfirm». Таким записам автоматично встановлюється статус «Cancelled», а відповідний часовий слот звільняється для інших користувачів;

Задача MissedAppointmentsJob періодично знаходить ті записи на прийом, які мають статус «Scheduled» та час початку яких минув більше ніж 10 хвилин тому. Таким записам автоматично встановлюється статус «Missed», адже лікар чи менеджер рецепції клініки не змінив статус такого запису на «Виконується», що означало б, що пацієнт з'явився вчасно.

Як у вебзастосунку, так і настільному Windows застосунку реалізовано сторінку «Панель керування», на якій користувачі можуть переглянути найбільш актуальну для них інформацію. У випадку вебсайту – пацієнти мають можливість переглянути історію своїх візитів до клінік з такими деталями, як дата та час, яку послугу було надано, який лікар проводив прийом, тощо. Для працівників клінік на цій сторінці відображається додатку відображено розклад клініки або їх особистий розклад залежно від посади користувача.

Для адміністрації передбачено сторінку налаштувань інформації та контенту, який буде відображатися пацієнтам на вебсайті. З її допомогою можна керувати інформацією про клініки та послуги, додавати або прибирати зображення, змінювати статус, або приховувати й показувати той чи інший об'єкт.

Висновки до другого розділу

У другому розділі було здійснено повне проєктування інформаційної системи управління та автоматизації обліку стоматологічної клініки. Сформульовано специфікацію вимог до системи, детально описано варіанти використання системи, які відображають основні сценарії взаємодії акторів з програмним комплексом, наведено низку відповідних діаграм. Детально описано усі прецеденти та акторів системи, особливості розподілу ролей та основні відмінності між ними.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						51
Змн.	Арк.	№ докум.	Підпис	Дата		

Проведено проектування бази даних, створено реляційну модель з використанням таблиць, зовнішніх ключів, обмежень та унікальних індексів, також збережених процедур. Особливу увагу приділено забезпеченню цілісності даних шляхом використання збережених процедур та транзакцій, які дозволяють реалізувати «відкат» внесених змін без додаткових проблем.

Змодельовано архітектуру та основні алгоритми роботи як програмного комплексу в цілому, так і окремих його компонентів та модулів, що потребували більш детального опису, додано відповідні діаграми класів, схеми та діаграму компонентів.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						52
Змн.	Арк.	№ докум.	Підпис	Дата		

РОЗДІЛ 3. ІНТЕРФЕЙС ТА ПОРЯДОК РОБОТИ З ІНФОРМАЦІЙНОЮ СИСТЕМОЮ УПРАВЛІННЯ ТА АВТОМАТИЗАЦІЇ ОБЛІКУ СТОМАТОЛОГІЧНОЇ КЛІНІКИ

3.1. Порядок встановлення та налаштування параметрів системи

Будь-який Windows застосунок, який запускається для розповсюдження серед кінцевих користувачів, потребує зручного та стабільного способу встановлення. Класичний варіант розповсюдження – створення інсталяційного пакету. Інсталятор – це окрема комп’ютерна програма, призначена для встановлення, оновлення та видалення програмного забезпечення. [16] Якісний інсталятор повинен не просто копіювати файли, а повноцінно реєструвати застосунок у системі, створювати ярлики й коректно працювати з оновленнями. В рамках дипломного проєкту було реалізовано професійний інсталятор для настільного застосунку, що підтримує платформи x64 та x86, забезпечує автоматичне оновлення та можливість об’єднаного EXE-файлу з платформо-специфічним розгортанням. Щоб отримати такий інсталятор на виході, було реалізовано додатковий проєкт типу «Консольний застосунок» та низку PowerShell скриптів, за допомогою яких можна без зайвих труднощів згенерувати інсталяційний пакет для програмного продукту.

Для реалізації безпосередньо інсталятора прийнято рішення про використання C# фреймворку WiX#, який в свою чергу «під капотом» генерує інсталятор за допомогою найбільш відомого набору інструментів для розробки інсталяційних пакетів – WiX Toolset, поєднавши його з WiX Toolset (Windows installer XML Toolset) – це набір інструментів, що дозволяє створювати інсталяційні пакети Windows Installer (.MSI) на основі XML описів. [18] WiXSharp дозволяє описати логіку встановлення безпосередньо мовою C#, що значним чином спрощує процес розробки. Крім того, WiX# дозволяє зручно кастомізувати будь який етап встановлення та зовнішній вигляд інсталяційної програми.

На етапі збірки усі C# інструкції буде перетворено в XML, з якого в свою чергу буде згенеровано MSI-інсталятор під визначену архітектуру процесора.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						53
Змн.	Арк.	№ докум.	Підпис	Дата		

Такий варіант розповсюдження не є найзручнішим з точки зору користувачього досвіду, тому вирішено об'єднати окремі інсталяційні пакети в єдиний EXE-інсталятор. Для цього використовується інструмент NSIS.

Фактично весь проект інсталятора побудовано навколо різних скриптів, які дозволяють виконувати збірку інсталятора покроково. Першим етапом є збирання Release-версій самого застосунку, наступний крок – генерація MSI-файлів, останній крок – об'єднання MSI-інсталяторів в єдиний EXE-інсталятор.

Завдяки усім вищезгаданим операціям, кінцевий користувач отримує максимально простий досвід встановлення програми: завантаження інсталяційного пакету «Майстра встановлення», знайомий та інтуїтивно зрозумілий інтерфейс встановлення, автоматичне створення ярлика робочого столу та запуск застосунку після встановлення.

Розгортання серверної частини проекту реалізовано за допомогою контейнеризації за допомогою Docker а також оркестратора контейнерів Docker Compose (рис. 3.1). Контейнеризація забезпечує уніфікацію середовища виконання програми та дозволяє запакувати застосунок разом з усіма залежностями, бібліотеками та налаштуваннями в єдиний образ, що може бути легко перенесений та запущений на будь-якій робочій станції незалежно від платформи чи операційної системи. Єдина вимога в даному випадку – підтримка програмного забезпечення Docker Engine та Docker Compose. Такий підхід максимально спрощує конфігурацію та забезпечує легке масштабування й розгортання системи як у локальному середовищі, так і у хмарній інфраструктурі.

Щоб запобігти викриттю чутливих даних, усі змінні середовища виносяться у .env файл, зокрема секретні ключі шифрування, налаштування SMTP підключення, різноманітні паролі, тощо.

Для спрощення процесу розгортання клієнтського вебзастосунку, його також запаковано у Docker-контейнер разом з Nginx. Nginx – це високопродуктивний вебсервер з відкритим кодом, який виконує багато додаткових важливих функцій в сучасній інфраструктурі. [19] В нашому випадку Nginx використовується для перенаправлення HTTP запитів (Reverse Proxy), налаштовує правила кешування,

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						54
Змн.	Арк.	№ докум.	Підпис	Дата		

обслуговує скомпільовані файли Blazor WebAssembly та забезпечує їх правильну обробку.

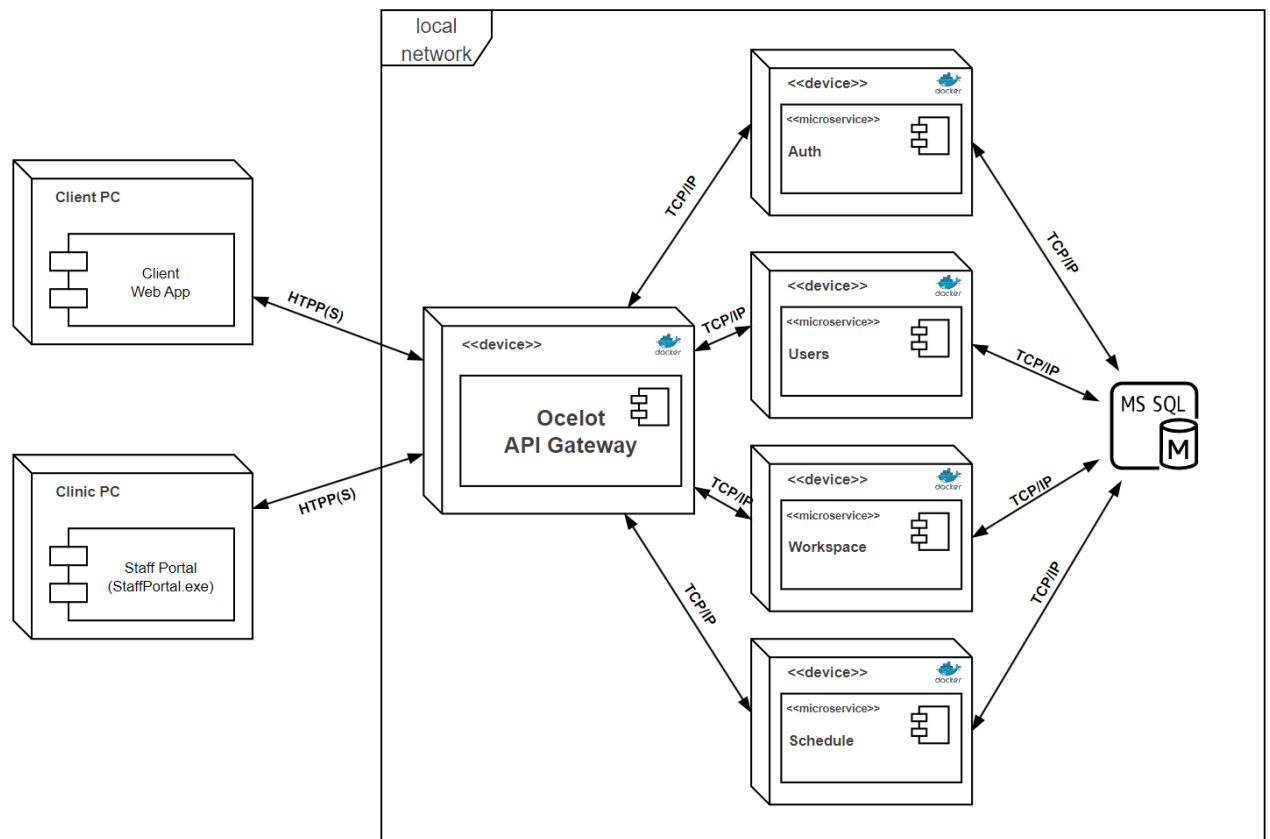


Рисунок 3.1 – Діаграма розгортання системи

3.2. Структура інтерфейсу інформаційної системи управління та автоматизації обліку стоматологічної клініки

Пацієнти та звичайні відвідувачі взаємодіють з системою через вебсайт. При вході користувача на сайт його зустрічає головна сторінка, на якій зображено блоки з найбільш популярними послугами та клініками (рис. 3.2 – 3.3). Користувач може як обрати одну з запропонованих послуг чи клінік, так і просто перейти на сторінку запису на прийом. У верхній частині сайту розташовано header, що містить кнопки для навігації між основними сторінками й перемикач для вибору мови інтерфейсу, а також кнопки для переходу на сторінки реєстрації та авторизації.

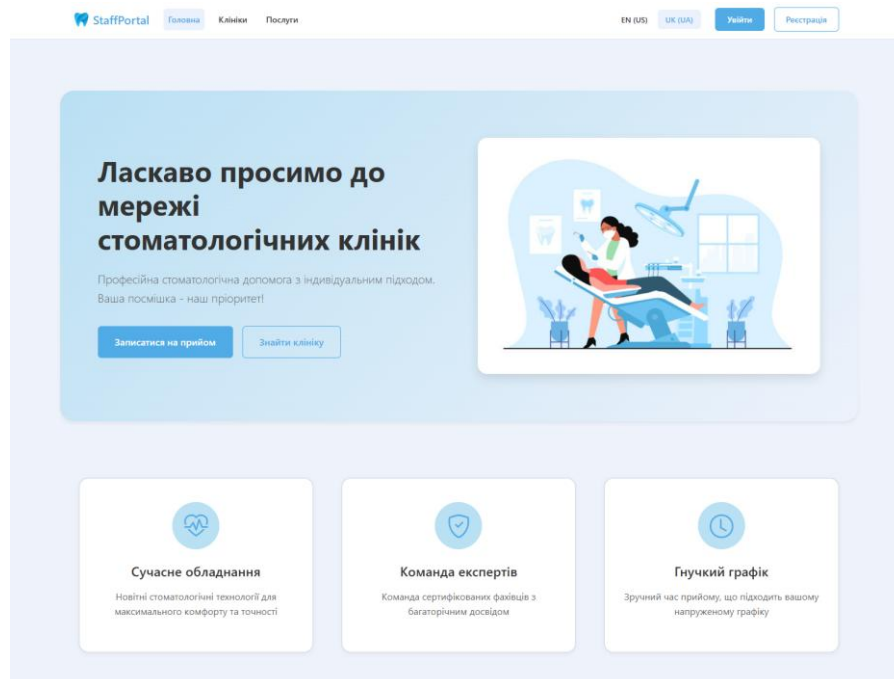


Рисунок 3.2 – Головна сторінка вебсайту (початок)

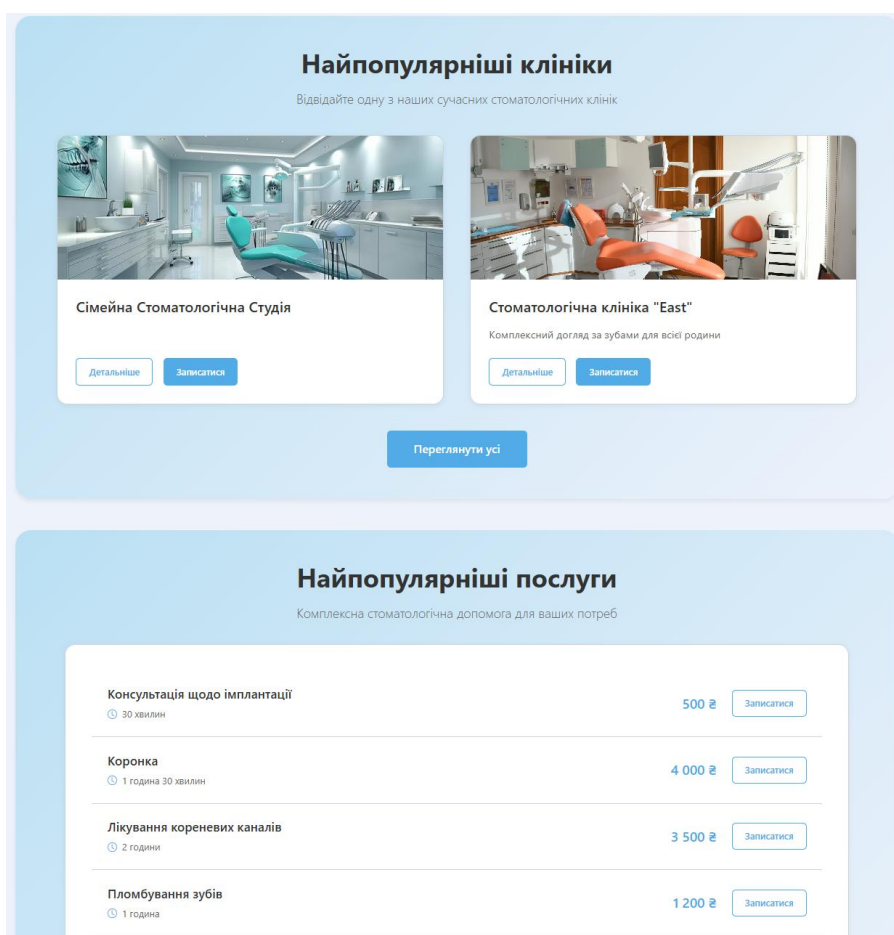


Рисунок 3.3 – Головна сторінка вебсайту (продовження)

Реєстрацію користувача реалізовано у вигляді класичної форми (рис. 3.4 – 3.5). Валідація обов’язкових полів, перевірки формату електронної пошти та надійності паролю присутня як на клієнтській стороні, так і дублюється на сервері.

Рисунок 3.4 – Реєстрація користувача (крок 1)

Рисунок 3.5 – Реєстрація користувача (крок 2)

Після успішної реєстрації облікового запису користувачу необхідно підтвердити адресу електронної пошти. Для цього на вказану при реєстрації електронну пошту надходить лист (рис. 3.6) двома мовами, що містить відповідну кнопку.

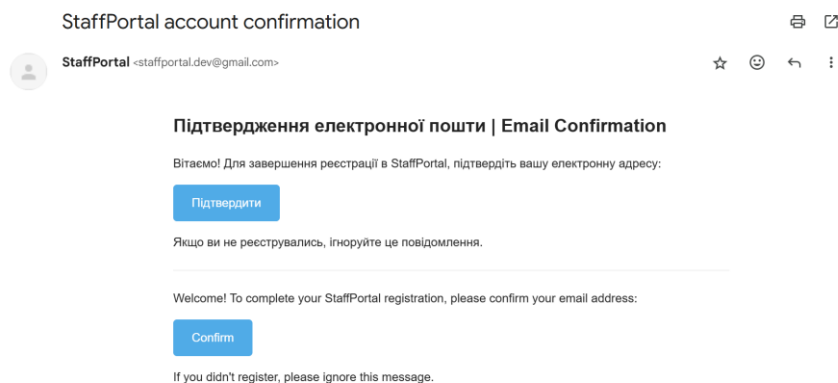


Рисунок 3.6 – Лист підтвердження

Після реєстрації користувач може увійти у свій особистий кабінет або продовжити користуватись сайтом без авторизації. При виборі послуги чи клініки на відповідних сторінках та натисканні кнопки «Записатися» сайт автоматично перенаправляє користувача на сторінку оформлення запису на прийом зі збереженням вибраного елемента через query параметр. На рисунках 3.7 – 3.8 продемонстровано строніки перегляду прейскуранту послуг та списку клінік. При великій кількості елементів передбачено динамічне відображення блоку пагінції в нижній частині сторінки.

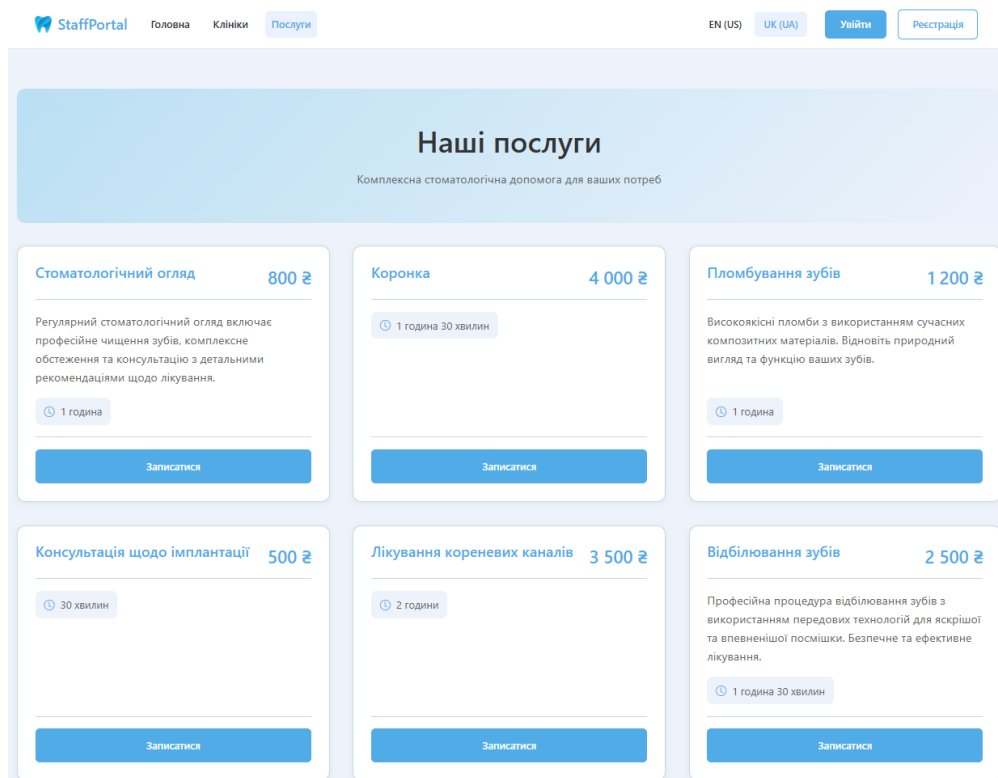


Рисунок 3.7 – Сторінка перегляду прейскуранту послуг

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						58
Змн.	Арк.	№ докум.	Підпис	Дата		

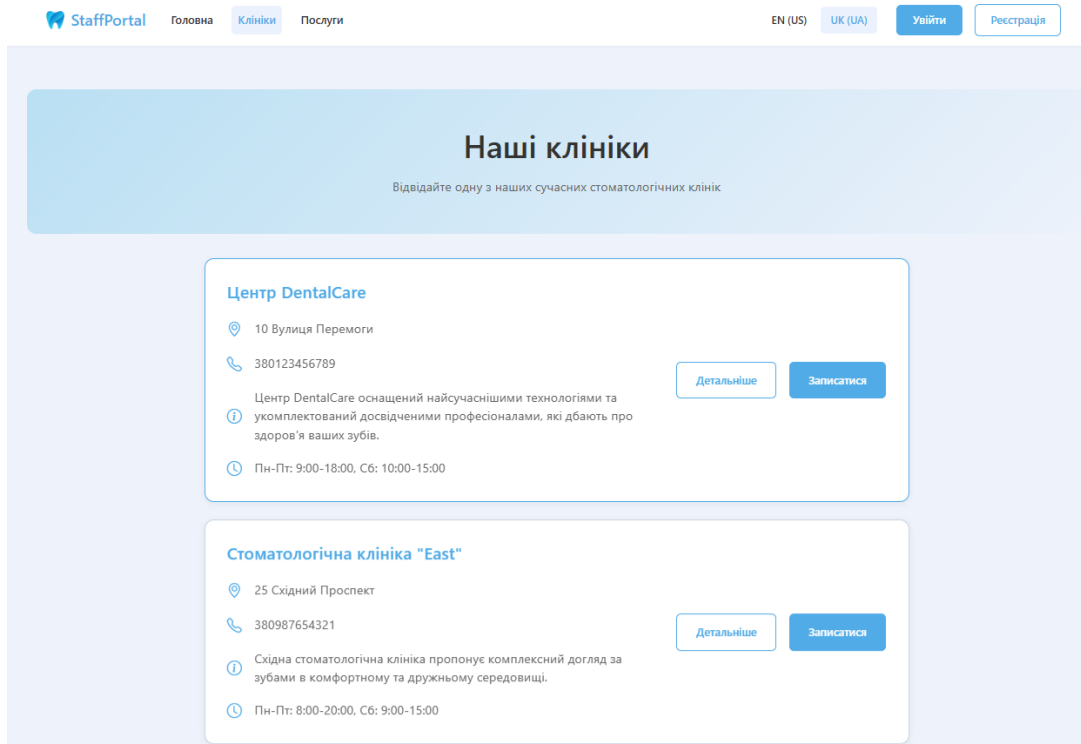


Рисунок 3.8 – Сторінка перегляду клінік

Для оформлення запису на прийом реалізовано зручну та інтуїтивно зрозумілу багатокрокову форму, де користувач покроково обирає клініку, послугу, лікаря та час початку прийому. Першим кроком форми є вибір клініки (рис. 3.9).

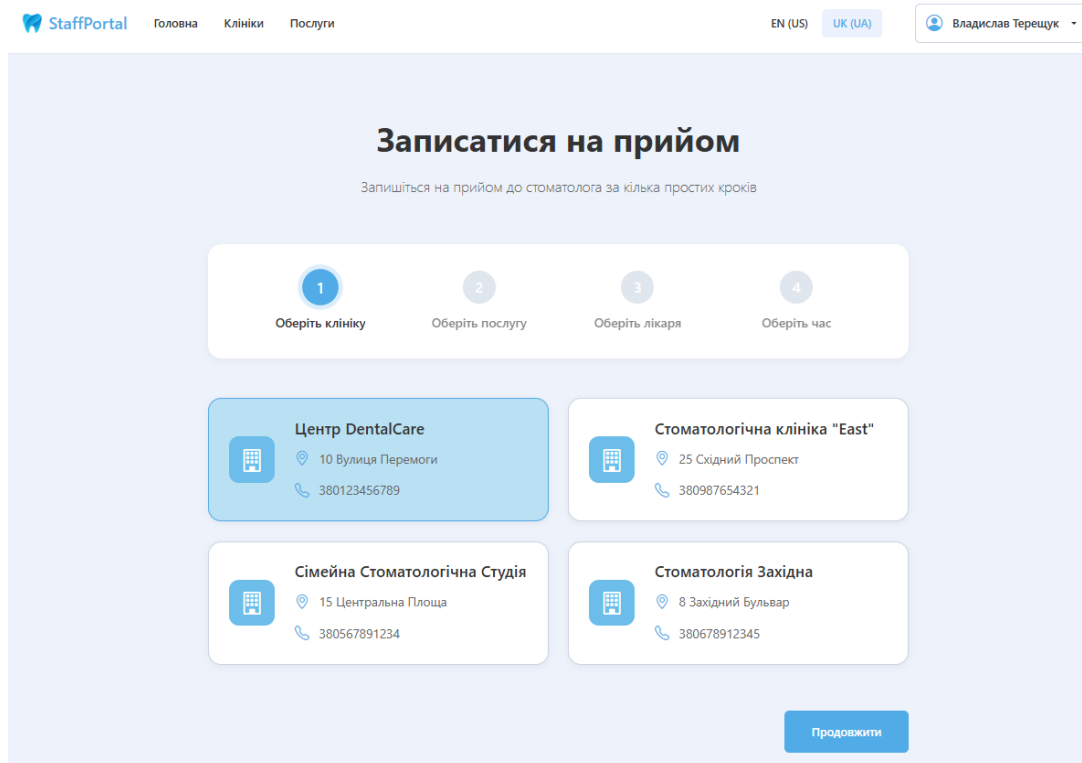


Рисунок 3.9 – Запис на прийом (крок 1)

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						59
Змн.	Арк.	№ докум.	Підпис	Дата		

Після вибору клініки користувачу необхідно обрати послугу (рис. 3.10) на яку він хоче записатися та лікаря (рис. 3.11). Кожна послуга зображена у вигляді картки з назвою, орієнтовною тривалістю процедури та вартістю.

Рисунок 3.10 – Запис на прийом (крок 2)

Рисунок 3.11 – Запис на прийом (крок 3)

Після вибору клініки, послуги та лікаря користувач переходить на крок форми 4 (рис. 3.12), де йому пропонується обрати зручну дату та час початку

послуги. Недоступні для запису дні та години візуально позначені напівпрозорим кольором та відповідною зміною курсора миші, що інтуїтивно сигналізує користувачу про неможливість запису.

The screenshot shows the 'StaffPortal' interface for booking an appointment. At the top, there are navigation links: 'Головна', 'Клініки', and 'Послуги'. The user is logged in as 'Владислав Терещук'. The main heading is 'Записатися на прийом' (Book an appointment), with a subtext 'Запишіться на прийом до стоматолога за кілька простих кроків' (Book an appointment with a dentist in a few simple steps). Below this, there are four numbered steps: 1. 'Оберіть клініку' (Select clinic) - Центр DentalCare; 2. 'Оберіть послугу' (Select service) - Коронка; 3. 'Оберіть лікаря' (Select doctor) - Микола Іванчук; 4. 'Оберіть час' (Select time). The current step is step 4. Below the steps, there is a calendar for June 2025. The date '5' is selected. Below the calendar, it says 'Доступні години на четвер, 5 червня 2025 р.' (Available hours on Thursday, 5 June 2025). A grid of time slots is shown: 08:00, 08:30, 09:00 (selected), 09:30, 10:00, 10:30, 11:00, 11:30, 12:00, 12:30, 14:00, 14:30, 15:00, 15:30, 16:00, 16:30. A 'Обідня перерва' (Lunch break) is indicated between 12:00 and 14:00. At the bottom, there are two buttons: 'Назад' (Back) and 'Підтвердити запис' (Confirm booking).

Рисунок 3.12 – Запис на прийом (крок 4)

Для авторизованого користувача оформлення запису завершується кроком 4, після чого автоматично отримує статус «Заплановано» без необхідності додаткового підтвердження через електронну пошту.

У випадку, коли неавторизований користувач оформлює запис на прийом – сайт відображає додатковий крок форми (рис. 3.13), а також такому користувачу необхідно обов’язково підтвердити свій запис протягом 10 хвилин після створення за допомогою електронної пошти (рис. 3.14).

Рисунок 3.13 – Запис на прийом (крок 5 для неавторизованих користувачів)

Рисунок 3.14 – Успішне оформлення запису на прийом

Після створення запису на прийом авторизований користувач може перейти на сторінку «Панель керування» та переглянути новостворений запис, найважливішу інформацію, а також скасувати запис, відредагувати свої дані та переглянути історію візитів (рис. 3.15).

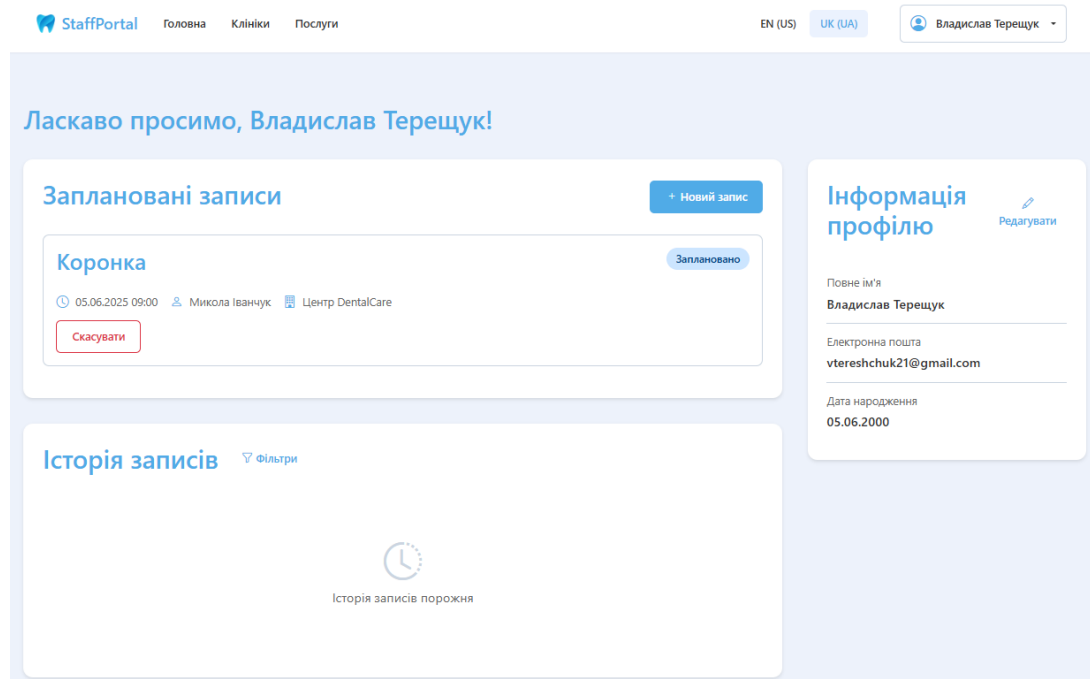


Рисунок 3.15 – Сторінка «Панель керування»

Працівники клініки також можуть взаємодіяти з системою через сайт в ролі пацієнтів, проте для роботи вони використовують окремий Windows застосунок. Увійти в обліковий запис через такий застосунок можуть лише ті користувачі, яким Адміністратор встановив роль «Employee», в іншому випадку вхід в застосунок буде відхилено (рис. 3.16).

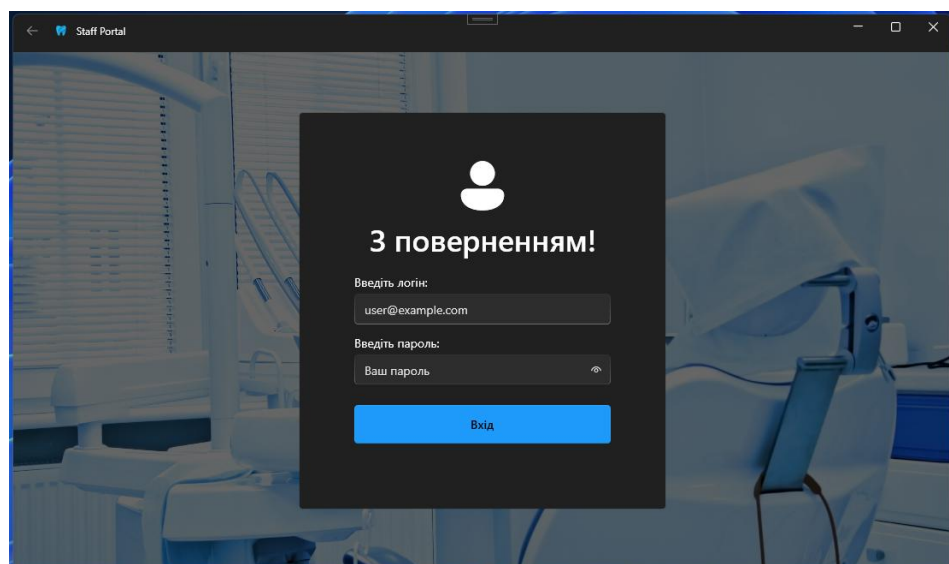


Рисунок 3.16 – Сторінка авторизації

Після авторизації користувачі, які мають роль «Employee» перенаправляються на сторінку вибору клініки (рис. 3.17), а для користувачів з

роллю Administrator цей крок пропускається та одразу відкривається сторінка «Панель керування» (рис. 3.19), де адміністратор може в будь який момент обрати клініку та переглянути специфічну для неї інформацію.

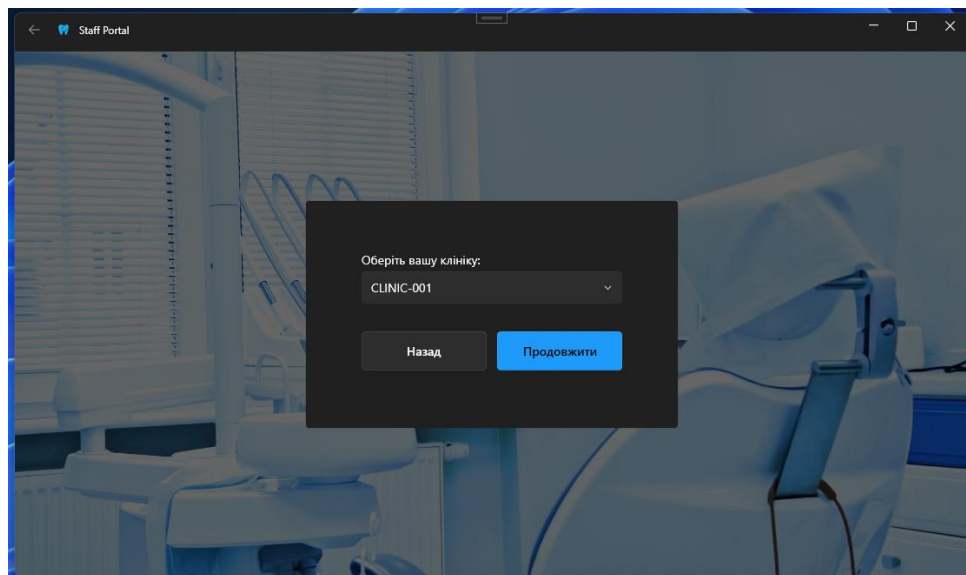


Рисунок 3.17 – Вибір клініки

Після вибору клініки звичайний користувач автоматично перенаправляється на сторінку «Панель керування» де надано можливість редагування облікового запису, перевибору клініки та власний розклад (рис. 3.18).

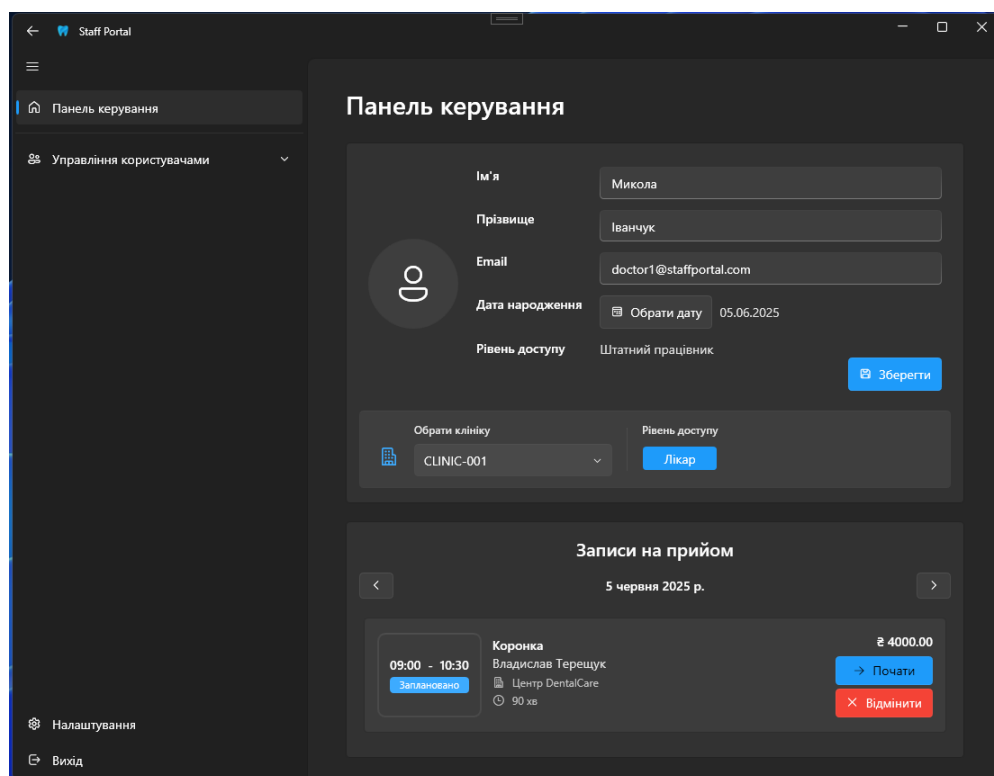


Рисунок 3.18 – Панель керування (роль Employee)

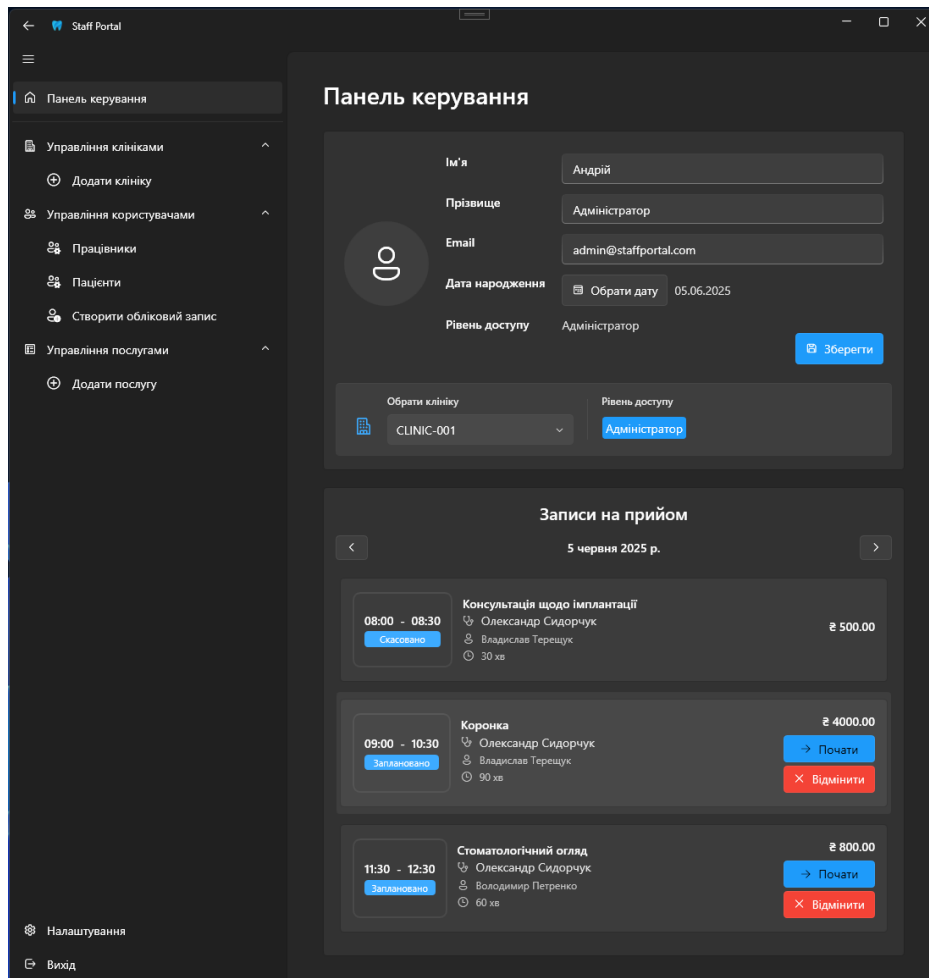


Рисунок 3.19 – Панель керування (роль Administrator)

На панелі керування лікарі можуть переглянути свій персональний розклад та управляти запланованими записами на прийом – змінювати їх статус або скасовувати. Менеджери з рецепції та адміністратори в свою чергу можуть переглянути розклад обраної клініки. Залежно від загальної ролі або від посади у вибраній клініці, для користувача показуються та приховуються ті чи інші сторінки.

Зокрема, штатні працівники можуть тільки переглядати інформацію про інших працівників в рамках обраної клініки, а також переглядати та редагувати інформацію про пацієнтів різних клінік (рис. 3.20). Крім того, користувачі, які мають роль Адміністратора в рамках певної клініки можуть редагувати посади інших користувачів, а саме встановлювати посаду «Лікар», «Менеджер рецепції» або «Адміністратор» (рис. 3.21).

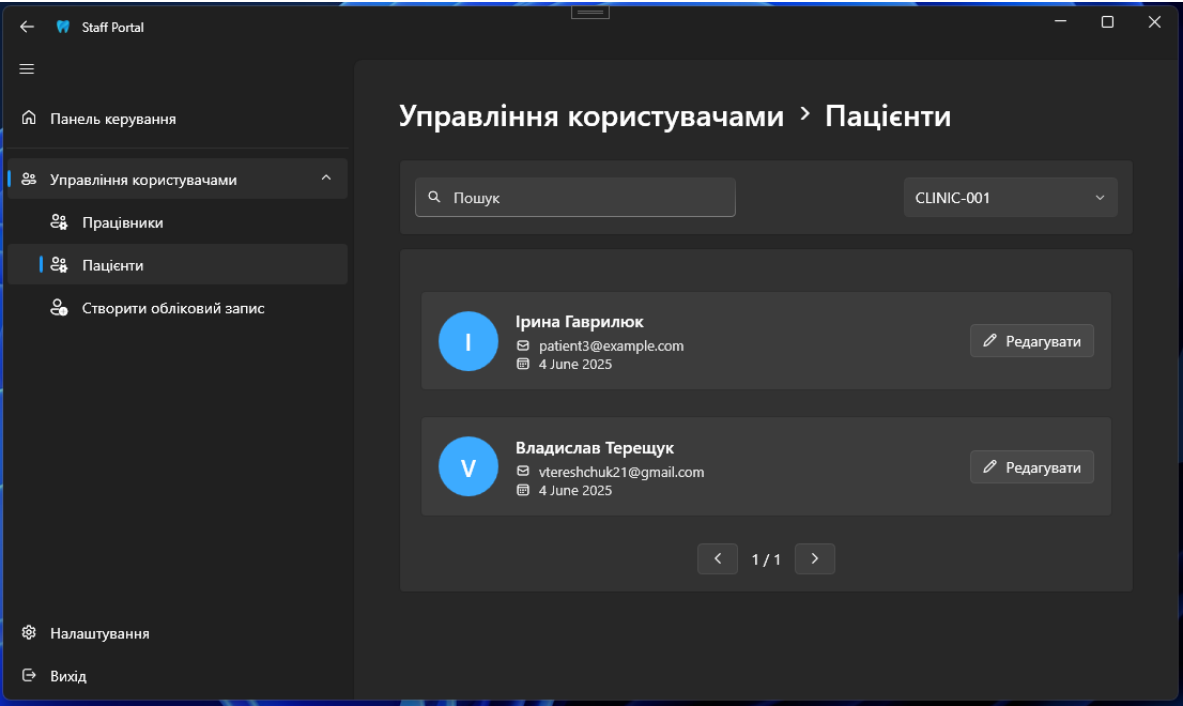


Рисунок 3.20 – Перегляд пацієнтів (Штатний працівник)

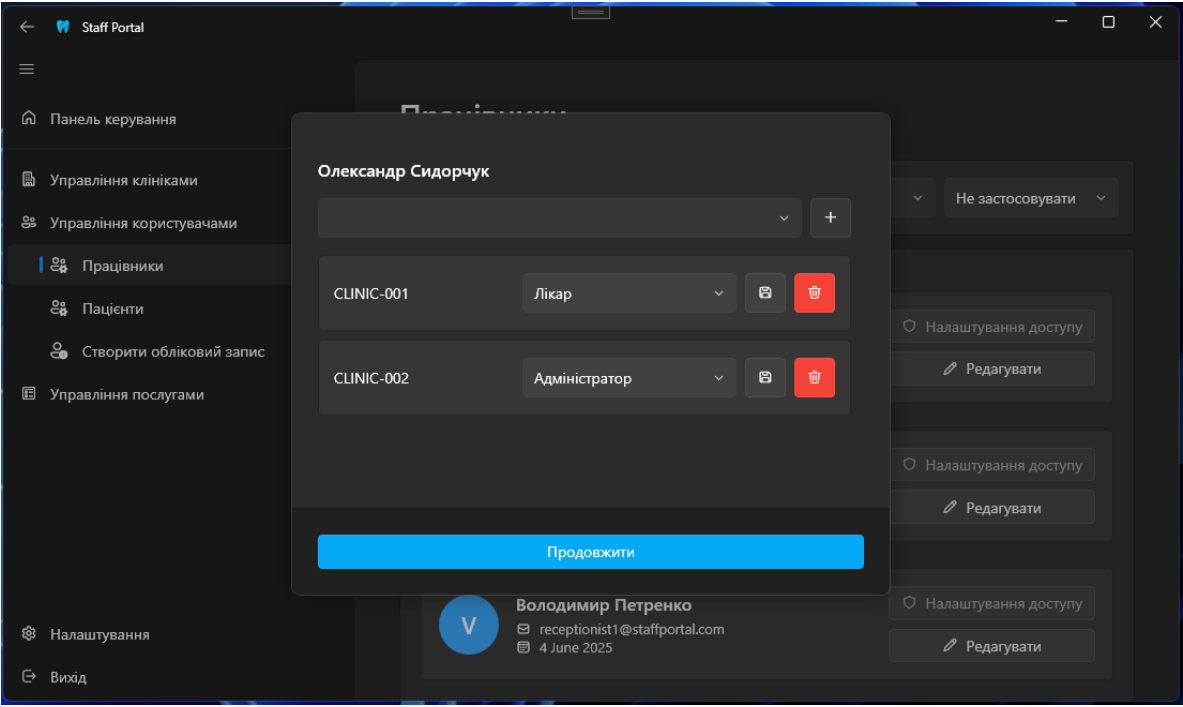


Рисунок 3.21 – Редагування посади працівника у клініках (Адміністратор)

Адміністратор в свою чергу може переглянути загальний список усіх користувачів та налаштувати їх ролі в рамках системи (рис. 3.22). При встановленні ролі «Штатний працівник» такий користувач з’являється на однойменній сторінці, та його може бути призначено на певну посаду у певній клініці.

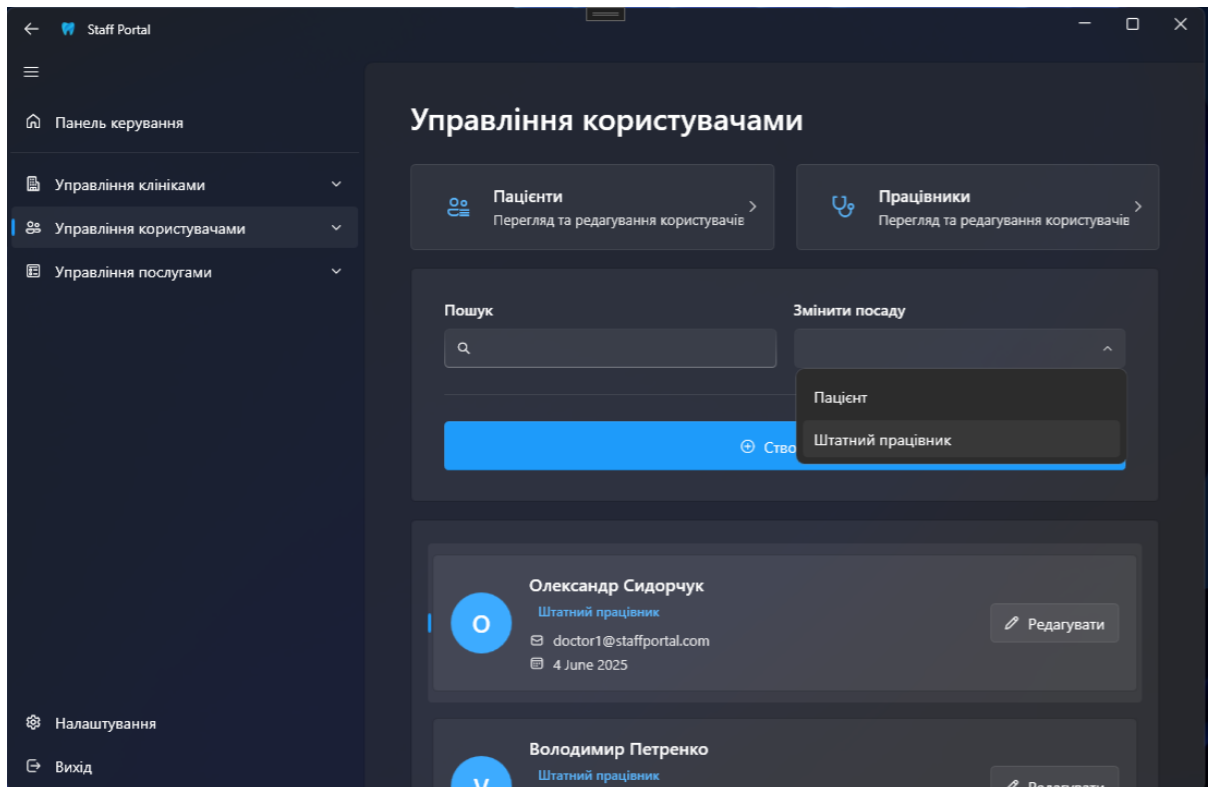


Рисунок 3.22 – Перегляд загального списку користувачів (Адміністратор)

Окрім описаного функціоналу, для Адміністраторів також додано функціонал налаштування контенту вебсайту, а саме додавання й редагування як клінік та послуг (рис. 3.23), так і пов'язаного контенту, описів, зображень, тощо (рис. 3.24).

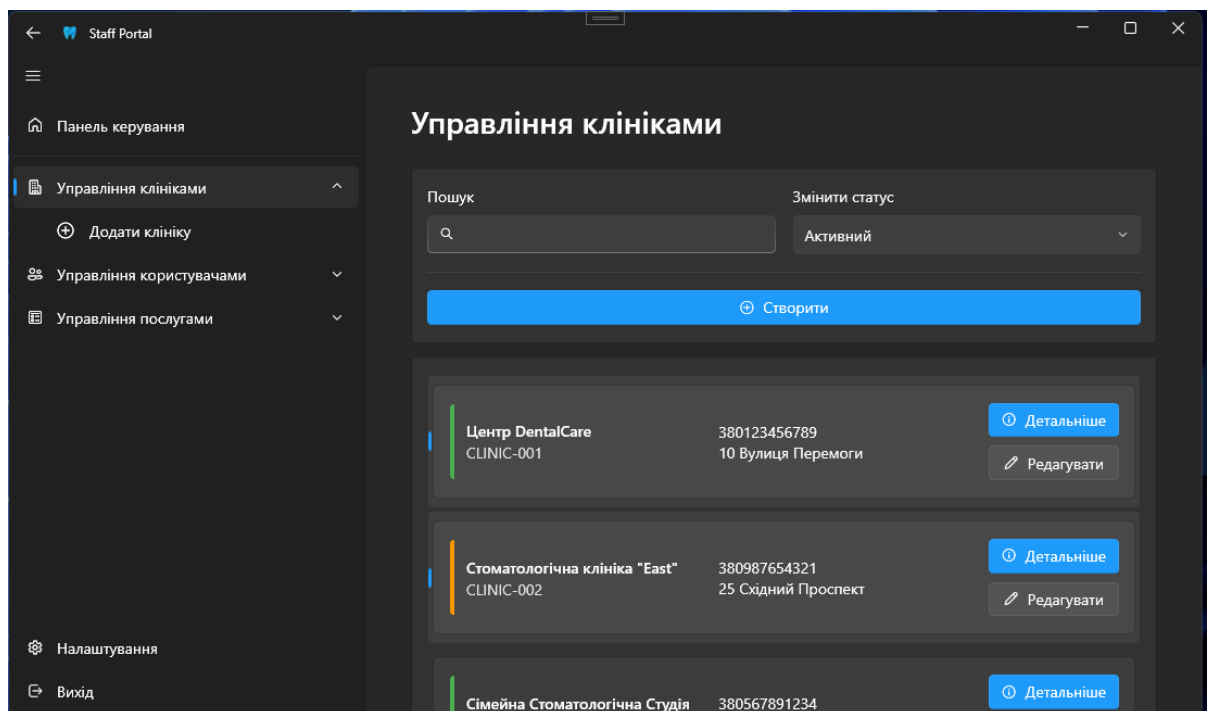


Рисунок 3.23 – Перегляд та редагування статусів клінік

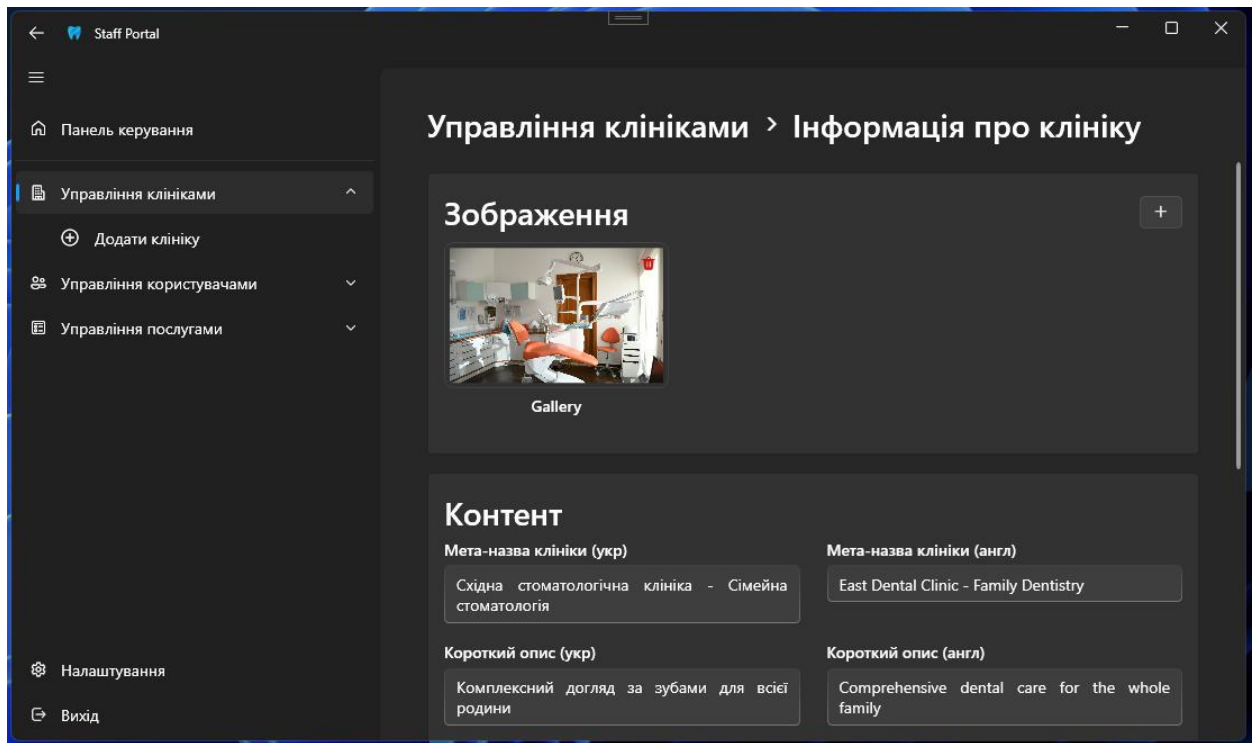


Рисунок 3.24 – Управління контентом

Для усіх сутностей реалізовано класичні сторінки створення та редагування, що являють собою звичні для більшості користувачів форми. При виконанні певних операцій над даними, які потребують відправки API запитів користувачу відображається тост-повідомлення про результат виконання.

Настільний застосунок, аналогічно до вебсайту підтримує дві мови, зокрема на сторінці налаштувань є відповідний перемикач активної мови інтерфейсу, а також реалізовано вибір світлої або темної кольорової палітри інтерфейсу користувача. З точки зору дизайну використано готовий набір компонентів, які візуально виглядають як системні органи управління Windows 11, завдяки чому інтерфейс користувача є досить зручним, інтуїтивно зрозумілим та знайомим для більшості людей.

3.3. Тестування роботи програмного комплексу

Для розробки та тестування програмного продукту використовувалась методологія Agile. Вона передбачає ітеративний підхід до створення програмного забезпечення, завдяки чому вдалось уникнути накопиченню помилок, підвищити загальний рівень якості коду та максимально ефективно розподіляти час на

розробку, тестування та виправлення помилок. [20] У межах ітерацій було визначено етапи Develop – Test – Review, де на кожній ітерації кожен новий блок функціоналу спочатку реалізовувався, після чого проводилось його тестування, аналіз та рефакторинг.

Під час реалізації настільного застосунку активно застосовувався підхід регресивного тестування, адже часто виникали ситуації, які потребували внесення змін у вже реалізований функціонал серверної частини. Регресивні тести допомогли виявити та виправити усі помилки, що виникли в результаті редагування того функціоналу, роботу на яким вже було завершено. Окрім регресивного тестування використовувалось smoke-тестування для оперативної оцінки стабільності поточної версії програмного комплексу, швидкого виявлення критичних недоліків, які можуть перешкоджати повноцінному проведенню основної ітерації тестування.

Для тестування серверної частини, а також зручності розробки клієнтських частин, до кожного мікросервісу додано Swagger – інструмент документування та тестування API. Він дозволяє швидко відправляти необхідні запити, відображає відповіді сервера та всю необхідну інформацію в зручному для сприйняття форматі. Так як інструмент Swagger – це фактично вебсторінка, з міркувань безпеки було підключено Basic HTTP authentication для входу на неї.

Значну увагу приділено тестуванню модулів авторизації та аутентифікації, зокрема проведено ряд модульних тестів та тестових сценаріїв авторизації користувачів. Детально перевірено усі механізми обмеження доступу на основі ролі та посади користувача в системі.

Висновки до третього розділу

У третьому розділі було розглянуто порядок встановлення та розгортання, налаштування, тестування та використання інформаційної системи управління та автоматизації обліку стоматологічної клініки. Детально описано процес та технології для створення інсталяційного пакету Windows застосунку. Також

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						69
Змн.	Арк.	№ докум.	Підпис	Дата		

описано особливості розгортання, контейнеризації та оркестрації серверної частини та вебсайту.

Надано детальний опис основних сторінок вебсайту, а також опис інтерфейсу користувача настільного застосунку, які безпосередньо використовуються користувачами для взаємодії з системою, виконання всіх необхідних операцій з даними та їх перегляду, представлено відповідні рисунки та схеми.

Описано використання методології Agile, наведено опис ітерацій спринтів розробки програмного продукту, а також описано підходи та методи тестування, які використовувались найбільше, зокрема регресивне, модульне, smoke тестування. Кожен тип тестування застосовувався на визначених етапах скриптів, що в результаті призвело до задовільних результатів стабільності роботи системи.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						70
Змн.	Арк.	№ докум.	Підпис	Дата		

ВИСНОВКИ

В ході виконання кваліфікаційної роботи було здійснено проєктування та реалізацію інформаційної системи управління та автоматизації обліку стоматологічної клініки. Метою проведення досліджень та створення програмного продукту стало вирішення наявних організаційних проблем в роботі стоматологічної клініки, підвищення якості обслуговування пацієнтів та забезпечення ведення електронного обліку в єдиному середовищі.

Виконано аналіз існуючих аналогів системи, проведено порівняння їх переваг та недоліків, визначено список основних блоків функціоналу, які буде реалізовано в системі, що розробляється. В результаті аналізу предметної області, формування каталогу вимог до програмного комплексу та визначення варіантів використання було проведено моделювання бази даних. Проведено дослідження найкращих методик налаштування сховища даних, реалізовано скрипт для ініціалізації, створення таблиць, збережених процедур та налаштування обмежень (додаток Б).

Визначено набір технологій та інструментів, які якнайкраще підходять для вирішення поставлених завдань, спроектовано архітектуру та об'єктно-орієнтовану модель програмного комплексу. Зокрема, для розробки було використано платформу .NET та різні її фреймворки, такі як ASP.NET, Blazor, WPF, WiX#. В якості архітектури серверної частини надано перевагу мікросервісному підходу, який забезпечить модульність системи, ізоляцію окремих блоків функціоналу та гнучкість розширення й масштабування системи у майбутньому.

В роботі представлено велику кількість діаграм, схем та зображень, які детальніше відображають описане. Реалізовано програмний продукт, що включає в себе функціонал авторизацій та аутентифікації, електронної черги на прийом до лікарів, формування розкладу роботи працівників, перегляд та управління клініками, прейскурантом послуг, управління профілями користувачів, функціонал адміністрування клінік та налаштування контенту вебсайту.

Функціонал розділено на спільні за призначенням групи, з яких сформовано мікросервіси Auth, Users, Schedule, Workspace. На поточному етапі реалізований

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						71
Змн.	Арк.	№ докум.	Підпис	Дата		

програмний комплекс готовий до поступової інтеграції у невеликий заклад стоматологічних послуг та містить якісний фундамент для масштабування, покращення й інтеграції більшої кількості функціональних можливостей. Зокрема, потенційно важливими блоками функціоналу для майбутньої реалізації можна вважати підключення SMS інформування та телефонії, інтеграція функціоналу статистики та аналітики, підключення платіжних систем та інтеграція медичного обладнання.

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						72
Змн.	Арк.	№ докум.	Підпис	Дата		

СПИСОК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. SimplexMed [Електронний ресурс]. – Режим доступу: <https://simplex-med.com/>
2. Doctor ELEKS – Медична інформаційна система [Електронний ресурс]. – Режим доступу: <https://doctor.eleks.com/>
3. SkyService – Хмарна POS-система для бізнесу [Електронний ресурс]. – Режим доступу: <https://skyservice.pro/uk/main/>
4. Open Dental Software [Електронний ресурс]. – Режим доступу: <https://www.opendental.com/> – Назва з екрана
5. Powell Ph., Smalley I. Monolithic Architecture [Електронний ресурс] / Ph. Powell, I. Smalley. – Режим доступу: <https://www.ibm.com/think/topics/monolithic-architecture>
6. Client-Server Architecture [Електронний ресурс] / Admin. – Режим доступу: <https://training.qatestlab.com/blog/technical-articles/client-server-architecture>
7. What is Microservices Architecture? [Електронний ресурс]. – Режим доступу: <https://cloud.google.com/learn/what-is-microservices-architecture?hl=en>
8. IBM – What is Load Balancing? [Електронний ресурс]. – Режим доступу: <https://www.ibm.com/think/topics/load-balancing>
9. RFC 7519: JSON Web Token (JWT) [Електронний ресурс]. – Режим доступу: <https://datatracker.ietf.org/doc/html/rfc7519>
10. Jackson G., Goodwin M. API Gateway [Електронний ресурс] / Gita Jackson, Michael Goodwin. – Режим доступу: <https://www.ibm.com/think/topics/api-gateway>
11. TechEmpower Framework Benchmarks [Електронний ресурс]. – Режим доступу: <https://www.techempower.com/benchmarks/#hw=ph&test=fortune&ion=data-r22>
12. Моат А. Using Docker: Developing and Deploying Software with Containers. 1-е вид. Бірмінгем: O'Reilly Media, 2015. – 354 с.

13. WebAssembly [Електронний ресурс] // MDN Web Docs. – Режим доступу: <https://developer.mozilla.org/docs/WebAssembly>
14. Google Cloud – Data mapping overview [Електронний ресурс]. – Режим доступу: <https://cloud.google.com/application-integration/docs/data-mapping-overview>
15. Transactions – Transact-SQL [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/sql/t-sql/language-elements/transactions-transact-sql>
16. What is a Data Transfer Object (DTO)? [Електронний ресурс] // Oka. – Режим доступу: <https://www.okta.com/identity-101/dto/>
17. What is a Software Installer? [Електронний ресурс] // Revenera. – Режим доступу: <https://www.revenera.com/install/products/installshield/installshield-tips-tricks/what-is-a-software-installer>
18. Документація WiX Toolset [Електронний ресурс] // FireGiant Docs. – Режим доступу: <https://docs.firegiant.com/wix3/main/>
19. Офіційний сайт NGINX [Електронний ресурс]. – Режим доступу: <https://nginx.org/en/>
20. What is the Agile methodology? [Електронний ресурс] // Atlassian. – Режим доступу: <https://www.atlassian.com/agile>
21. Microsoft Learn – YARP: How a reverse proxy handles HTTP [Електронний ресурс]. – Режим доступу: <https://learn.microsoft.com/en-us/aspnet/core/fundamentals/servers/yarp/yarp-overview?view=aspnetcore-9.0#how-a-reverse-proxy-handles-http>
22. Stonis M. Enterprise Application Patterns Using .NET MAUI / M. Stonis ; Microsoft Developer Division, .NET, and Visual Studio product teams. – Redmond : Microsoft Corporation, 2022. – 144 с.
23. Task-based asynchronous programming [Електронний ресурс] // Microsoft Learn. – Режим доступу: <https://learn.microsoft.com/en-us/dotnet/standard/parallel-programming/task-based-asynchronous-programming>

ДОДАТКИ

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						75
Змн.	Арк.	№ докум.	Підпис	Дата		

Middleware-компонент валідації сесії користувача

```

public class SessionValidationMiddleware
{
    private readonly RequestDelegate _next;
    private readonly IDbSettingsProvider _dbSettingsProvider;

    public SessionValidationMiddleware(RequestDelegate next, IDbSettingsProvider
dbSettingsProvider)
    {
        _next = next;
        _dbSettingsProvider = dbSettingsProvider;
    }

    public async Task InvokeAsync(HttpContext context)
    {
        var accessToken =
context.Request.Headers["Authorization"].FirstOrDefault()?.Split(" ").Last();
        if (!string.IsNullOrEmpty(accessToken))
        {
            using var connection = new
SqlConnection(_dbSettingsProvider.GetConnectionString());
            var session = await connection.QuerySingleOrDefaultAsync<UserSession>(
                "SELECT * FROM UserSessions WHERE AccessToken = @AccessToken AND
ExpiryTime > @Now",
                new { AccessToken = accessToken, Now = DateTimeOffset.UtcNow });

            if (session == null)
            {
                context.Response.StatusCode = StatusCodes.Status401Unauthorized;
                await context.Response.WriteAsync("Session expired or revoked.");
                return;
            }
        }

        await _next(context);
    }
}

```

Скрипт створення та ініціалізації бази даних

```

IF NOT EXISTS (SELECT * FROM sys.databases WHERE name = 'StaffPortal')
BEGIN
    CREATE DATABASE StaffPortal;
END
GO

USE StaffPortal;
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'AppUsers')
BEGIN
    CREATE TABLE AppUsers (
        Id BIGINT IDENTITY PRIMARY KEY,
        Email NVARCHAR(256) NULL,
        EmailConfirmationToken NVARCHAR(256) NULL,
        PhoneNumber NVARCHAR(256) NULL,
        PhoneConfirmationToken NVARCHAR(256) NULL,
        FirstName NVARCHAR(50) NOT NULL,
        LastName NVARCHAR(50) NOT NULL,
        IsEmailConfirmed BIT NOT NULL DEFAULT 0,
        IsPhoneConfirmed BIT NOT NULL DEFAULT 0,
        IsPasswordConfirmed BIT NOT NULL DEFAULT 0,
        IsActive BIT NOT NULL DEFAULT 1,
        PasswordHash NVARCHAR(256) NULL,
        BirthDate DATE NULL,
        CreationDate DATETIMEOFFSET NOT NULL,
        WorkspaceRole INT NOT NULL DEFAULT 2,
        IsRegistered BIT NOT NULL DEFAULT 1,
        CONSTRAINT UserEmailIsValid CHECK (Email IS NULL OR Email LIKE '%@%.__%'),
        CONSTRAINT UserPhoneIsValid CHECK (PhoneNumber IS NULL OR PhoneNumber LIKE
'[0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9][0-9]'),
        CONSTRAINT UserHasContact CHECK (Email IS NOT NULL OR PhoneNumber IS NOT
NULL)
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Clinics')
BEGIN
    CREATE TABLE Clinics (
        Id NVARCHAR(50) PRIMARY KEY,
        Name_en NVARCHAR(100) NOT NULL,
        Name_uk NVARCHAR(100) NOT NULL,
        Address_en NVARCHAR(256) NULL,
        Address_uk NVARCHAR(256) NULL,
        PhoneNumber NVARCHAR(50) NULL,
        IsActive BIT NOT NULL DEFAULT 1
    );
END
GO

```

```

);
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'UserClinics')
BEGIN
    SET QUOTED_IDENTIFIER ON;

    CREATE TABLE UserClinics (
        Id BIGINT IDENTITY PRIMARY KEY,
        UserId BIGINT NOT NULL,
        ClinicId NVARCHAR(50) NOT NULL,
        AccessLevel INT NOT NULL,
        FOREIGN KEY (UserId) REFERENCES AppUsers(Id) ON DELETE CASCADE ON UPDATE NO
ACTION,
        FOREIGN KEY (ClinicId) REFERENCES Clinics(Id) ON DELETE CASCADE ON UPDATE NO
ACTION
    );

    CREATE UNIQUE NONCLUSTERED INDEX IX_UserClinics_OneClinicPerStaff
ON UserClinics(UserId)
WHERE AccessLevel IN (3, 7); -- Doctor or Receptionist access levels

    CREATE UNIQUE NONCLUSTERED INDEX IX_UserClinics_OneStaffRolePerUser
ON UserClinics(UserId, AccessLevel)
WHERE AccessLevel IN (3, 7); -- Doctor or Receptionist access levels
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Services')
BEGIN
    CREATE TABLE Services (
        Id BIGINT IDENTITY PRIMARY KEY,
        Name_en NVARCHAR(100) NOT NULL,
        Name_uk NVARCHAR(100) NOT NULL,
        Price DECIMAL(18, 2) NOT NULL,
        ApproximateTime INT NOT NULL,
        IsAvailable BIT NOT NULL DEFAULT 1
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'TimeSlots')
BEGIN
    CREATE TABLE TimeSlots (
        Id BIGINT IDENTITY PRIMARY KEY,
        UserId BIGINT NOT NULL FOREIGN KEY (UserId) REFERENCES dbo.AppUsers(Id) ON
DELETE NO ACTION ON UPDATE NO ACTION,
        ClinicId NVARCHAR(50) NOT NULL FOREIGN KEY (ClinicId) REFERENCES
dbo.Clinics(Id) ON DELETE NO ACTION ON UPDATE NO ACTION,

```

```

        StartTimestamp DATETIMEOFFSET NOT NULL,
        EndTimestamp DATETIMEOFFSET NOT NULL,
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'Appointments')
BEGIN
    CREATE TABLE Appointments (
        Id BIGINT IDENTITY PRIMARY KEY,
        AppointmentConfirmationToken NVARCHAR(256) NOT NULL,
        EmployeeId BIGINT NOT NULL FOREIGN KEY (EmployeeId) REFERENCES
dbo.AppUsers(Id) ON DELETE NO ACTION ON UPDATE NO ACTION,
        PatientId BIGINT NOT NULL FOREIGN KEY (PatientId) REFERENCES dbo.AppUsers(Id)
ON DELETE NO ACTION ON UPDATE NO ACTION,
        TimeSlotId BIGINT NOT NULL UNIQUE FOREIGN KEY (TimeSlotId) REFERENCES
dbo.TimeSlots(Id) ON DELETE NO ACTION ON UPDATE NO ACTION,
        ClinicId NVARCHAR(50) NOT NULL FOREIGN KEY (ClinicId) REFERENCES
dbo.Clinics(Id) ON DELETE NO ACTION ON UPDATE NO ACTION,
        ServiceId BIGINT NOT NULL FOREIGN KEY (ServiceId) REFERENCES dbo.Services(Id)
ON DELETE NO ACTION ON UPDATE NO ACTION,
        CreationTimestamp DATETIMEOFFSET NOT NULL DEFAULT SYSDATETIMEOFFSET(),
        Discount DECIMAL(18, 2) NOT NULL,
        TotalPrice DECIMAL(18, 2) NOT NULL,
        Status INT NOT NULL
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'UserSessions')
BEGIN
    CREATE TABLE UserSessions (
        Id BIGINT IDENTITY PRIMARY KEY,
        UserId BIGINT NOT NULL,
        AccessToken NVARCHAR(MAX) NOT NULL,
        RefreshToken NVARCHAR(256) NOT NULL,
        ExpiryTime DATETIMEOFFSET NOT NULL,
        CreationDate DATETIMEOFFSET NOT NULL,
        FOREIGN KEY (UserId) REFERENCES AppUsers(Id) ON DELETE NO ACTION ON UPDATE NO
ACTION
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ClinicContent')
BEGIN
    CREATE TABLE ClinicContent (
        Id BIGINT IDENTITY PRIMARY KEY,
        ClinicId NVARCHAR(50) NOT NULL,
        Description_en NVARCHAR(MAX) NULL,

```

```

        Description_uk NVARCHAR(MAX) NULL,
        ShortDescription_en NVARCHAR(500) NULL,
        ShortDescription_uk NVARCHAR(500) NULL,
        MetaTitle_en NVARCHAR(100) NULL,
        MetaTitle_uk NVARCHAR(100) NULL,
        MetaDescription_en NVARCHAR(300) NULL,
        MetaDescription_uk NVARCHAR(300) NULL,
        WorkingHours_en NVARCHAR(500) NULL,
        WorkingHours_uk NVARCHAR(500) NULL,
        LastUpdated DATETIMEOFFSET NOT NULL DEFAULT SYSDATETIMEOFFSET(),
        IsActive BIT NOT NULL DEFAULT 1,
        FOREIGN KEY (ClinicId) REFERENCES Clinics(Id) ON DELETE CASCADE,
        CONSTRAINT UQ_ClinicContent_ClinicId UNIQUE (ClinicId)
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ServiceContent')
BEGIN
    CREATE TABLE ServiceContent (
        Id BIGINT IDENTITY PRIMARY KEY,
        ServiceId BIGINT NOT NULL,
        Description_en NVARCHAR(MAX) NULL,
        Description_uk NVARCHAR(MAX) NULL,
        MetaTitle_en NVARCHAR(100) NULL,
        MetaTitle_uk NVARCHAR(100) NULL,
        MetaDescription_en NVARCHAR(300) NULL,
        MetaDescription_uk NVARCHAR(300) NULL,
        LastUpdated DATETIMEOFFSET NOT NULL DEFAULT SYSDATETIMEOFFSET(),
        IsActive BIT NOT NULL DEFAULT 1,
        FOREIGN KEY (ServiceId) REFERENCES Services(Id) ON DELETE CASCADE,
        CONSTRAINT UQ_ServiceContent_ServiceId UNIQUE (ServiceId)
    );
END
GO

IF NOT EXISTS (SELECT * FROM sys.tables WHERE name = 'ClinicImages')
BEGIN
    CREATE TABLE ClinicImages (
        Id BIGINT IDENTITY PRIMARY KEY,
        ClinicId NVARCHAR(50) NOT NULL,
        ImageType INT NOT NULL, -- 1=Hero, 2=Thumbnail, 3=Gallery
        ImageData VARBINARY(MAX) NOT NULL,
        ContentType NVARCHAR(100) NOT NULL,
        FileSize BIGINT NOT NULL,
        UploadDate DATETIMEOFFSET NOT NULL DEFAULT SYSDATETIMEOFFSET(),
        IsActive BIT NOT NULL DEFAULT 1,
        FOREIGN KEY (ClinicId) REFERENCES Clinics(Id) ON DELETE CASCADE
    );
END

```

GO

```
IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'CreateUser')
    DROP PROCEDURE CreateUser
```

GO

```
CREATE PROCEDURE CreateUser
```

```
    @Email NVARCHAR(256),
    @PhoneNumber NVARCHAR(256),
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @IsEmailConfirmed BIT,
    @IsPhoneConfirmed BIT,
    @IsPasswordConfirmed BIT,
    @IsActive BIT,
    @PasswordHash NVARCHAR(256),
    @BirthDate DATE,
    @CreationDate DATETIMEOFFSET,
    @WorkspaceRole INT,
    @IsRegistered BIT = 1,
    @EmailConfirmationToken NVARCHAR(256) = NULL,
    @UserId BIGINT OUTPUT,
    @ErrorCode INT OUTPUT
```

```
AS
```

```
BEGIN
```

```
    SET NOCOUNT ON;
```

```
    SET @ErrorCode = 0;
```

```
    SET @UserId = 0;
```

```
    -- Check if email already exists (if provided)
```

```
    IF @Email IS NOT NULL
```

```
    BEGIN
```

```
        DECLARE @ExistingUserId BIGINT;
```

```
        DECLARE @IsUserRegistered BIT;
```

```
        SELECT @ExistingUserId = Id, @IsUserRegistered = IsRegistered
```

```
        FROM AppUsers
```

```
        WHERE Email = @Email;
```

```
        IF @ExistingUserId IS NOT NULL
```

```
        BEGIN
```

```
            IF @IsUserRegistered = 1
```

```
            BEGIN
```

```
                SET @ErrorCode = 51009; -- EmailAlreadyExists
```

```
                RETURN;
```

```
            END
```

```
            ELSE
```

```
            BEGIN
```

```
                -- User exists but not registered, update the existing record
```

```
                UPDATE AppUsers
```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						81
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        SET FirstName = @FirstName,
            LastName = @LastName,
            PhoneNumber = @PhoneNumber,
            IsEmailConfirmed = @IsEmailConfirmed,
            IsPhoneConfirmed = @IsPhoneConfirmed,
            IsPasswordConfirmed = @IsPasswordConfirmed,
            IsActive = @IsActive,
            PasswordHash = @PasswordHash,
            BirthDate = @BirthDate,
            WorkspaceRole = @WorkspaceRole,
            IsRegistered = 1,
            EmailConfirmationToken = @EmailConfirmationToken
        WHERE Id = @ExistingUserId;

        SET @UserId = @ExistingUserId;
        RETURN;
    END
END
END

-- Check if phone number already exists (if provided)
IF @PhoneNumber IS NOT NULL AND EXISTS (SELECT 1 FROM AppUsers WHERE PhoneNumber
= @PhoneNumber)
BEGIN
    SET @ErrorCode = 51010; -- PhoneNumberAlreadyExists
    RETURN;
END

BEGIN TRY
    INSERT INTO AppUsers (
        Email, PhoneNumber, FirstName, LastName,
        IsEmailConfirmed, IsPhoneConfirmed, IsActive,
        PasswordHash, BirthDate, CreationDate, WorkspaceRole,
        IsRegistered, IsPasswordConfirmed, EmailConfirmationToken
    )
    VALUES (
        @Email, @PhoneNumber, @FirstName, @LastName,
        @IsEmailConfirmed, @IsPhoneConfirmed, @IsActive,
        @PasswordHash, @BirthDate, @CreationDate, @WorkspaceRole,
        @IsRegistered, @IsPasswordConfirmed, @EmailConfirmationToken
    );

    SET @UserId = SCOPE_IDENTITY();
    RETURN;
END TRY
BEGIN CATCH
    SET @ErrorCode = ERROR_NUMBER();
    RETURN;
END CATCH
END

```

```

GO

IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'CreateTimeSlot')
    DROP PROCEDURE CreateTimeSlot
GO

CREATE PROCEDURE CreateTimeSlot
    @UserId BIGINT,
    @ClinicId NVARCHAR(50),
    @StartTimestamp DATETIMEOFFSET,
    @EndTimestamp DATETIMEOFFSET,
    @TimeSlotId BIGINT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    IF EXISTS (
        SELECT 1
        FROM TimeSlots ts
        INNER JOIN Appointments a ON ts.Id = a.TimeSlotId
        WHERE ts.UserId = @UserId
            AND ts.ClinicId = @ClinicId
            AND (ts.StartTimestamp < @EndTimestamp AND ts.EndTimestamp >
@StartTimestamp)
            AND a.Status IN (0, 1, 2) -- WaitingForConfirm, Scheduled, Completed
        )
    BEGIN
        RETURN 51000; -- TimeSlotOverlap
    END

    INSERT INTO TimeSlots (UserId, ClinicId, StartTimestamp, EndTimestamp)
    VALUES (@UserId, @ClinicId, @StartTimestamp, @EndTimestamp);

    SET @TimeSlotId = SCOPE_IDENTITY();
    RETURN 0;
END
GO

IF EXISTS (SELECT * FROM sys.procedures WHERE name =
'CreateAppointment_Unauthorized')
    DROP PROCEDURE CreateAppointment_Unauthorized
GO

CREATE PROCEDURE CreateAppointment_Unauthorized
    @Email NVARCHAR(256),
    @FirstName NVARCHAR(50),
    @LastName NVARCHAR(50),
    @EmployeeId BIGINT,
    @ClinicId NVARCHAR(50),
    @ServiceId BIGINT,

```

```

@StartTimestamp DATETIMEOFFSET,
@AppointmentConfirmationToken NVARCHAR(256),
@AppointmentId BIGINT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;

    IF NOT EXISTS (
        SELECT 1 FROM UserClinics
        WHERE UserId = @EmployeeId
        AND ClinicId = @ClinicId
        AND AccessLevel = 3
    )
    BEGIN
        RETURN 51001; -- InsufficientAccessRights
    END

    BEGIN TRY
        BEGIN TRANSACTION;

        DECLARE @PatientId BIGINT;
        SELECT @PatientId = Id FROM AppUsers WHERE Email = @Email;

        IF @PatientId IS NULL
        BEGIN
            INSERT INTO AppUsers (Email, FirstName, LastName, CreationDate,
WorkspaceRole)
            VALUES (@Email, @FirstName, @LastName, SYSDATETIMEOFFSET(), 2);
            SET @PatientId = SCOPE_IDENTITY();
        END

        -- Check for unconfirmed appointments limit
        DECLARE @UnconfirmedCount INT;
        SELECT @UnconfirmedCount = COUNT(*)
        FROM Appointments
        WHERE PatientId = @PatientId AND Status = 0;

        IF @UnconfirmedCount >= 2
        BEGIN
            ROLLBACK;
            RETURN 51003;
        END

        -- Check for total active appointments limit
        DECLARE @ActiveCount INT;
        SELECT @ActiveCount = COUNT(*)
        FROM Appointments
        WHERE PatientId = @PatientId AND Status IN (0, 1);

        IF @ActiveCount >= 5

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
Змн.	Арк.	№ докум.	Підпис	Дата		84

```

BEGIN
    ROLLBACK;
    RETURN 51004;
END

IF @PatientId = @EmployeeId
BEGIN
    ROLLBACK;
    RETURN 51002;
END

DECLARE @ApproximateTime INT;
SELECT @ApproximateTime = ApproximateTime FROM Services WHERE Id =
@ServiceId;

DECLARE @EndTimestamp DATETIMEOFFSET = DATEADD(MINUTE, @ApproximateTime,
@StartTimestamp);
DECLARE @TimeSlotId BIGINT;

DECLARE @Result INT;
EXEC @Result = CreateTimeSlot
    @UserId = @EmployeeId,
    @ClinicId = @ClinicId,
    @StartTimestamp = @StartTimestamp,
    @EndTimestamp = @EndTimestamp,
    @TimeSlotId = @TimeSlotId OUTPUT;

IF @Result != 0
BEGIN
    ROLLBACK;
    RETURN @Result;
END

DECLARE @Price DECIMAL(18,2);
SELECT @Price = Price FROM Services WHERE Id = @ServiceId;

INSERT INTO Appointments (AppointmentConfirmationToken, EmployeeId,
PatientId, TimeSlotId, ClinicId, ServiceId, Discount, TotalPrice, Status)
VALUES (@AppointmentConfirmationToken, @EmployeeId, @PatientId, @TimeSlotId,
@ClinicId, @ServiceId, 0, @Price, 0);

SET @AppointmentId = SCOPE_IDENTITY();
COMMIT;
RETURN 0;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK;
    RETURN ERROR_NUMBER();
END CATCH

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						85
Змн.	Арк.	№ докум.	Підпис	Дата		

```

END
GO

IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'CreateAppointment_Authorized')
    DROP PROCEDURE CreateAppointment_Authorized
GO

CREATE PROCEDURE CreateAppointment_Authorized
    @EmployeeId BIGINT,
    @PatientId BIGINT,
    @ClinicId NVARCHAR(50),
    @ServiceId BIGINT,
    @StartTimestamp DATETIMEOFFSET,
    @AppointmentConfirmationToken NVARCHAR(256),
    @AppointmentId BIGINT OUTPUT
AS
BEGIN
    SET NOCOUNT ON;
    BEGIN TRY
        BEGIN TRANSACTION;

        IF NOT EXISTS (
            SELECT 1 FROM UserClinics
            WHERE UserId = @EmployeeId
            AND ClinicId = @ClinicId
            AND AccessLevel = 3
        )
        BEGIN
            RETURN 51001;
        END

        DECLARE @UnconfirmedCount INT;
        SELECT @UnconfirmedCount = COUNT(*)
        FROM Appointments
        WHERE PatientId = @PatientId AND Status = 0;

        IF @UnconfirmedCount >= 2
        BEGIN
            ROLLBACK;
            RETURN 51003;
        END

        DECLARE @ActiveCount INT;
        SELECT @ActiveCount = COUNT(*)
        FROM Appointments
        WHERE PatientId = @PatientId AND Status IN (0, 1);

        IF @ActiveCount >= 5
        BEGIN
            ROLLBACK;
        END
    END TRY
    BEGIN CATCH
        ROLLBACK;
    END CATCH
END

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						86
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        RETURN 51004;
    END

    IF @PatientId = @EmployeeId
    BEGIN
        RETURN 51002;
    END

    DECLARE @ApproximateTime INT;
    SELECT @ApproximateTime = ApproximateTime FROM Services WHERE Id =
@ServiceId;

    DECLARE @EndTimestamp DATETIMEOFFSET = DATEADD(MINUTE, @ApproximateTime,
@StartTimestamp);
    DECLARE @TimeSlotId BIGINT;

    DECLARE @Result INT;
    EXEC @Result = CreateTimeSlot
        @UserId = @EmployeeId,
        @ClinicId = @ClinicId,
        @StartTimestamp = @StartTimestamp,
        @EndTimestamp = @EndTimestamp,
        @TimeSlotId = @TimeSlotId OUTPUT;

    IF @Result != 0
    BEGIN
        ROLLBACK;
        RETURN @Result;
    END

    DECLARE @Price DECIMAL(18,2);
    SELECT @Price = Price FROM Services WHERE Id = @ServiceId;

    INSERT INTO Appointments (AppointmentConfirmationToken, EmployeeId,
PatientId, TimeSlotId, ClinicId, ServiceId, Discount, TotalPrice, Status)
    VALUES (@AppointmentConfirmationToken, @EmployeeId, @PatientId, @TimeSlotId,
@ClinicId, @ServiceId, 0, @Price, 1);

    SET @AppointmentId = SCOPE_IDENTITY();
    COMMIT;
    RETURN 0;
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK;
    RETURN ERROR_NUMBER();
END CATCH
END
GO

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						87
Змн.	Арк.	№ докум.	Підпис	Дата		

```

IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'UpdateAppointmentStatus')
    DROP PROCEDURE UpdateAppointmentStatus
GO

CREATE PROCEDURE UpdateAppointmentStatus
    @AppointmentId BIGINT,
    @UserId BIGINT,
    @NewStatus INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Check if appointment exists and get the clinic
    DECLARE @ClinicId NVARCHAR(50);
    SELECT @ClinicId = ClinicId
    FROM Appointments
    WHERE Id = @AppointmentId;

    IF @ClinicId IS NULL
    BEGIN
        RETURN 51005; -- Appointment not found
    END

    -- Check if user has access to this clinic
    IF NOT EXISTS (
        SELECT 1
        FROM UserClinics
        WHERE UserId = @UserId
        AND ClinicId = @ClinicId
        AND AccessLevel IN (3, 7, 15) -- Doctor, Receptionist, Admin access levels
    )
    BEGIN
        RETURN 51101; -- Insufficient access rights
    END

    -- Update appointment status
    UPDATE Appointments
    SET Status = @NewStatus
    WHERE Id = @AppointmentId;

    RETURN 0; -- Success
END
GO

IF EXISTS (SELECT * FROM sys.procedures WHERE name = 'ManageUserClinicAccess')
    DROP PROCEDURE ManageUserClinicAccess
GO

CREATE PROCEDURE ManageUserClinicAccess
    @UserId BIGINT,
    @ClinicId NVARCHAR(50),

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						88
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        @AccessLevel INT
AS
BEGIN
    SET NOCOUNT ON;

    -- Validate input
    IF NOT EXISTS (SELECT 1 FROM AppUsers WHERE Id = @UserId)
        RETURN 51006; -- User not found

    IF NOT EXISTS (SELECT 1 FROM Clinics WHERE Id = @ClinicId)
        RETURN 51007; -- Clinic not found

    IF @AccessLevel NOT IN (3, 7, 15) -- Doctor, Receptionist, Admin
        RETURN 51008; -- Invalid input value

    BEGIN TRY
        BEGIN TRANSACTION;

        -- For Doctor or Receptionist roles
        IF @AccessLevel IN (3, 7)
        BEGIN
            -- Delete all existing Doctor/Receptionist roles
            DELETE FROM UserClinics
            WHERE UserId = @UserId
            AND AccessLevel IN (3, 7);

            -- Add new role
            INSERT INTO UserClinics (UserId, ClinicId, AccessLevel)
            VALUES (@UserId, @ClinicId, @AccessLevel);
        END

        -- For Administrator role
        ELSE IF @AccessLevel = 15
        BEGIN
            -- Check if any role exists for this user in this clinic
            IF EXISTS (
                SELECT 1
                FROM UserClinics
                WHERE UserId = @UserId
                AND ClinicId = @ClinicId
            )
            BEGIN
                -- Update existing role to Administrator
                UPDATE UserClinics
                SET AccessLevel = @AccessLevel
                WHERE UserId = @UserId
                AND ClinicId = @ClinicId;
            END
        ELSE
        BEGIN
            -- Add new admin role

```

```

        INSERT INTO UserClinics (UserId, ClinicId, AccessLevel)
        VALUES (@UserId, @ClinicId, @AccessLevel);
    END
END

COMMIT;
RETURN 0; -- Success
END TRY
BEGIN CATCH
    IF @@TRANCOUNT > 0
        ROLLBACK;
    RETURN ERROR_NUMBER();
END CATCH
END
GO

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						90
Змн.	Арк.	№ докум.	Підпис	Дата		

Конфігурація маршрутизації ocelot.json для API Gateway

```

{
  "GlobalConfiguration": {
    "BaseUrl": "https://localhost:8082",
    "UseOriginalForwardedHeaders": true,
    "ForwardOriginalHost": true,
    "PreserveHeaderCase": true,
    "DownstreamHeaderTransform": {
      "X-Real-IP": "{X-Real-IP}",
      "X-Forwarded-For": "{X-Forwarded-For}",
      "X-Forwarded-Proto": "{X-Forwarded-Proto}",
      "X-Forwarded-Host": "{X-Forwarded-Host}",
      "Accept-Language": "{Accept-Language}"
    }
  },
  "Routes": [
    {
      "SwaggerKey": "auth",
      "DownstreamPathTemplate": "/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "staffportal.auth",
          "Port": 8090
        }
      ],
      "UpstreamPathTemplate": "/api/v1/auth/{everything}",
      "UpstreamHttpMethod": [
        "POST",
        "GET",
        "PUT",
        "DELETE",
        "OPTIONS"
      ],
      "CorsOptions": {
        "AllowCredentials": true,
        "AllowedOrigins": ["http://localhost:8080"],
        "AllowedMethods": ["GET", "POST", "PUT", "DELETE", "OPTIONS"],
        "AllowedHeaders": ["Content-Type", "Authorization", "Accept-Language"]
      }
    },
    {
      "SwaggerKey": "schedule",
      "DownstreamPathTemplate": "/{everything}",
      "DownstreamScheme": "http",
      "DownstreamHostAndPorts": [
        {
          "Host": "staffportal.schedule",

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						91
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "Port": 8084
    },
    ],
    "UpstreamPathTemplate": "/api/v1/schedule/{everything}",
    "UpstreamHttpMethod": [
        "POST",
        "GET",
        "PUT",
        "DELETE"
    ],
    "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
    },
    "RouteClaimsRequirement": {
        "Role": [ "Public", "Patient", "Employee", "Administrator" ]
    }
},
{
    "SwaggerKey": "users",
    "DownstreamPathTemplate": "/{everything}",
    "DownstreamScheme": "http",
    "DownstreamHostAndPorts": [
        {
            "Host": "staffportal.users",
            "Port": 8086
        }
    ],
    "UpstreamPathTemplate": "/api/v1/users/{everything}",
    "UpstreamHttpMethod": [
        "POST",
        "GET",
        "PUT",
        "DELETE"
    ],
    "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
    },
    "RouteClaimsRequirement": {
        "Role": [ "Public", "Patient", "Employee", "Administrator" ]
    }
},
{
    "SwaggerKey": "workspace",
    "DownstreamPathTemplate": "/{everything}",
    "DownstreamScheme": "http",
    "DownstreamHostAndPorts": [
        {
            "Host": "staffportal.workspace",

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						92
Змн.	Арк.	№ докум.	Підпис	Дата		

```

        "Port": 8088
    },
    ],
    "UpstreamPathTemplate": "/api/v1/workspace/{everything}",
    "UpstreamHttpMethod": [
        "POST",
        "GET",
        "PUT",
        "DELETE"
    ],
    "AuthenticationOptions": {
        "AuthenticationProviderKey": "Bearer",
        "AllowedScopes": []
    },
    "RouteClaimsRequirement": {
        "Role": [ "Public", "Patient", "Employee", "Administrator" ]
    }
}
],
"SwaggerEndPoints": [
{
    "Key": "auth",
    "Config": [
        {
            "Name": "Auth API",
            "Version": "v1",
            "Url": "http://staffportal.auth:8090/swagger/v1/swagger.json"
        }
    ]
},
{
    "Key": "schedule",
    "Config": [
        {
            "Name": "Schedule API",
            "Version": "v1",
            "Url": "http://staffportal.schedule:8084/swagger/v1/swagger.json"
        }
    ]
},
{
    "Key": "users",
    "Config": [
        {
            "Name": "Users API",
            "Version": "v1",
            "Url": "http://staffportal.users:8086/swagger/v1/swagger.json"
        }
    ]
}
],
},

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						93
Змн.	Арк.	№ докум.	Підпис	Дата		

```

{
  "Key": "workspace",
  "Config": [
    {
      "Name": "Workspace API",
      "Version": "v1",
      "Url": "http://staffportal.workspace:8088/swagger/v1/swagger.json"
    }
  ]
}

```

					ІПЗ.КР.Б – 121 – 25 – ПЗ	Арк.
						94
Змн.	Арк.	№ докум.	Підпис	Дата		