МІНІСТЕРСТВО ОСВІТИ І НАУКИ УКРАЇНИ
ОДЕСЬКИЙ ДЕРЖАВНИЙ ЕКОЛОГІЧНИЙ УНІВЕРСИТЕТ

Факультет комп'ютерних наук,
управління та адміністрування
Кафедра інформаційних технологій

Кваліфікаційна робота бакалавра

на тему: Розроблення інформаційної системи «Приватний кабінет лікаря стоматолога»

Виконав студент групи К-20і
спеціальності 122 «Комп'ютерні науки»
Воробйов Олександр Олександрович

Керівник к.геогр.н., доцент
Кузніченко Світлана Дмитрівна

Консультант

Рецензент к.техн.н., доцент
Гнатовська Ганна Арнольдівна

Одеса 2022

ЗМІСТ

СКОРОЧЕННЯ ТА УМОВНІ ПОЗНАКИ

БД – база даних

ОС – операційна система

ПЗ – програмне забезпечення

ADT – Android Development Tools – Інструменти розробки в ОС Android

API – Application Programming Interface – програмний інтерфейс програми

AR – Augmented Reality – доповнена реальність

IDE – Integrated Development Environment – інтегроване середовище розробки

SDK – Software Development Kit – набір засобів розробки

ВСТУП

У сучасному світі міцно влаштувалися інформаційні технології, повністю змінивши як бізнес-процеси, так і повсякденне життя людей. Незважаючи на швидкий розвиток інформаційних технологій і проникнення їх в усі сфери життя, є галузі, які до сих пір залишаються в минулому столітті, і одна з цих областей - як не дивно, медицина. Щороку розробляються методи діагностики і лікування, а хворий, що приходить сьогодні до медичного закладу, як і 20 років тому, найчастіше стоїть у черзі, в реєстратурі шукають його картку, написану від руки.

Наш час ознаменувався високим темпом автоматизації бізнес процесів та перенесенням їх в інтернет. Будь-яка компанія, велика або мала, намагається впровадити IT технології та продукти, які б збільшили прибуток, кількість клієнтів або спростили деякі процеси. Не становить винятку і стоматологія. Лікарям стоматологам усе частіше необхідні інструменти для спрощення або автоматизації деяких повсякденних завдань. Звичайно, можна скористатися уже існуючими способами: розсилка по електронній пошті, вебінари, віддалені візити «по скайпу», соціальні мережі, форуми. Але здебільшого не всі ці інструменти однаково корисні і результативні, тому розробки в даній сфері – необхідні.

Метою кваліфікаційної роботи є розробка інформаційної системи «Приватний кабінет лікаря стоматолога». Дана система допоможе лікарю стоматологу керувати своєю приватною практикою, а також полегшить взаємодію лікаря з пацієнтом.

Для досягнення мети необхідно вирішити ряд завдань:

− проаналізувати аналогічні системи та виявити функціонал, на який варто звернути увагу;

− вибрати інструменти для проектування;

− спроектувати систему;

− реалізувати систему;

− провести тестування системи.

Структура кваліфікаційної роботи складається з вступу, трьох розділів, висновків, переліку посилань на 15 найменувань. Повний обсяг роботи становить 70 сторінок і містить 24 рисунка та 2 таблиці.

1 ОГЛЯД ІСНУЮЧИХ СИСТЕМ-АНАЛОГІВ ДЛЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ПРИВАТНИЙ КАБІНЕТ ЛІКАРЯ СТОМАТОЛОГА»

1.1 Аналіз предметної області

У роботі будь-якої організації з часом з'являється необхідність в спрощенні або автоматизації деяких процесів. З наростанням виробництва зростає кількість бізнес процесів та інформації, яку необхідно контролювати. І для вилучення більшої вигоди і більш злагодженої їх роботи необхідна автоматизація всього підприємства. Таким чином, автоматизація підприємств здійснює контроль всіх етапів виробництва і вносить в виробничий процес значні корективи.

Відштовхуючись від вище сказаного, можна зробити висновок, що люба сфера потребує автоматизації за допомогою програмного забезпечення, для більш легкої роботи.

Тому даний продукт призначений для стоматологічного кабінету, аби зробити його більш сучасними, що дасть змогу не сидіти довго у чергах та уникнути втрати картки. Користувачами цієї системи будуть лікарі, а також відвідувачі клініки.

Бізнес проблема яку вирішує даний продукт:

− зручна реєстратура, тобто планування зайнятості стоматологів, що дозволяє швидко оцінити навантаження і вибрати час для створення запису;

− облік та ведення пацієнтів (зберігання історії захворювань та відвідувань);

− електрона черга, тобто тепер непотрібна довга присутність у клініці аби попасти до лікаря, це можна зробити онлайн;

− можливість пацієнта завжди мати при собі медичну карту.

Ідея проекту полягає у тому, щоб лікарі могли вести історію захворювань кожного пацієнта онлайн, зберігати діагнози та знімки. А пацієнт в свою

чергу має власний кабінет за допомогою якого може подивитися історію, або ж нагадати лікування.

1.2 Аналіз існуючих рішень та аналогів

Перед розробкою даної системи був проведений аналіз аналогів на ринку а зокрема облікових стоматологічних систем. Було виявлено, що існуючі рішення мають суттєві недоліки або обмежений функціонал.

У процесі пошуку аналогів, були знайдені такі схожі системи:

− QStoma [1];

− IDENT [2];

− 1С:Медицина [3].

Нижче розглянемо ці системи більш детально.

1.2.1 Система QStoma

QStoma – хмарний веб-сервіс, створений для спрощення і автоматизації адміністративної роботи лікарів стоматологів. Профільна система обліку та управління стоматологічною клінікою [1].

Система на перший погляд нагадує простий сайт-візитку. Нижче будуть вказані переваги та недоліки системи.

Переваги:

− зручний інтерфейс;

− простий у використанні;

− необмежена кількість користувачів;

− можливість лікаря планувати за допомогою календаря;

− низька вартість (купівля додатку один раз, за низьку ціну).

Оскільки ця система дуже проста та має дуже низьку вартість, були виявлені декілька недоліків. По-перше немає запису онлайн та ведення лікарем мед картки пацієнта. По-друге, а ні лікар, а ні пацієнт не може про-

дивитися історію захворювань. По-третє, у пацієнта немає доступу до власної мед картки.

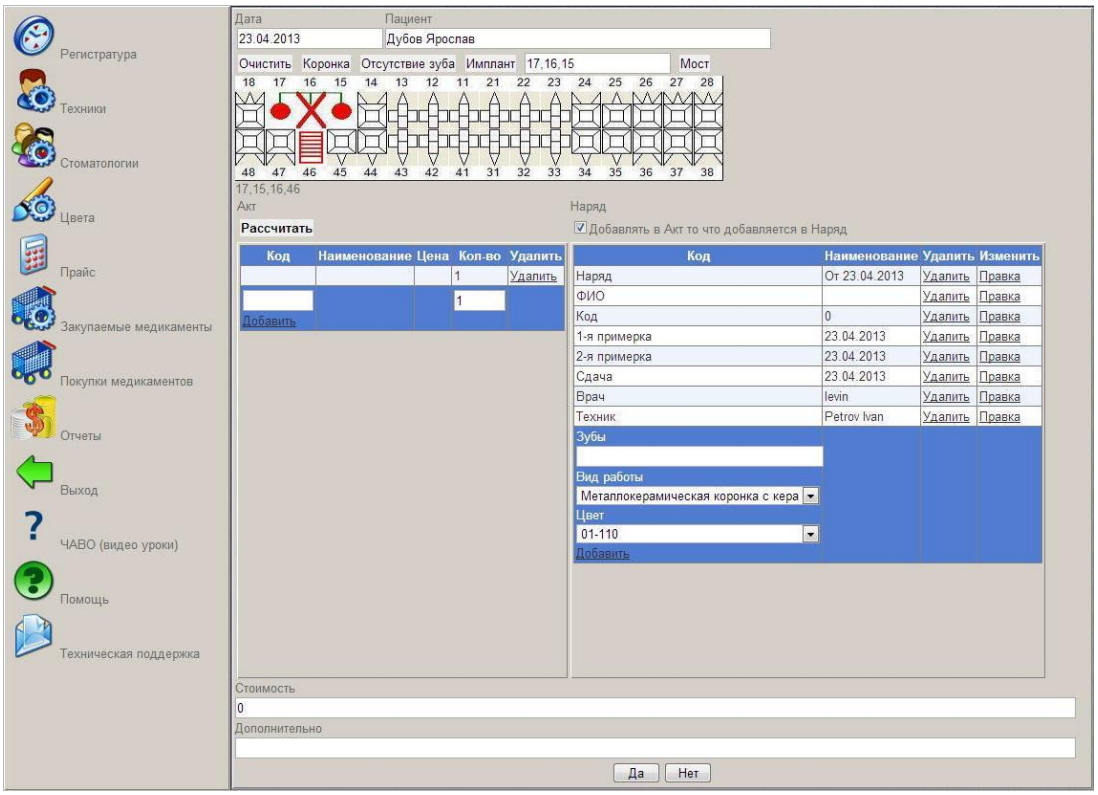

Рисунок 1.1 – Скриншот програми QStoma

1.2.2 Система IDENT

IDENT – програма для керування клінікою. Дана система для стоматології сприяє оптимізації роботи, допомагає скоротити час при обслуговуванні пацієнтів. Функціонал включає: ведення амбулаторних карт, планування, перегляд історії захворювань [2].

У системи є такі переваги:

− зручний інтерфейс;

− необмежена кількість користувачів;

− запис онлайн.

До недоліків даної програми відноситься висока вартість, оскільки для отримання необмеженої кількості користувачів необхідно купляти місце для кожного наступного лікаря, також ця система не має можливості надавати пацієнтам доступу до їх мед карток.

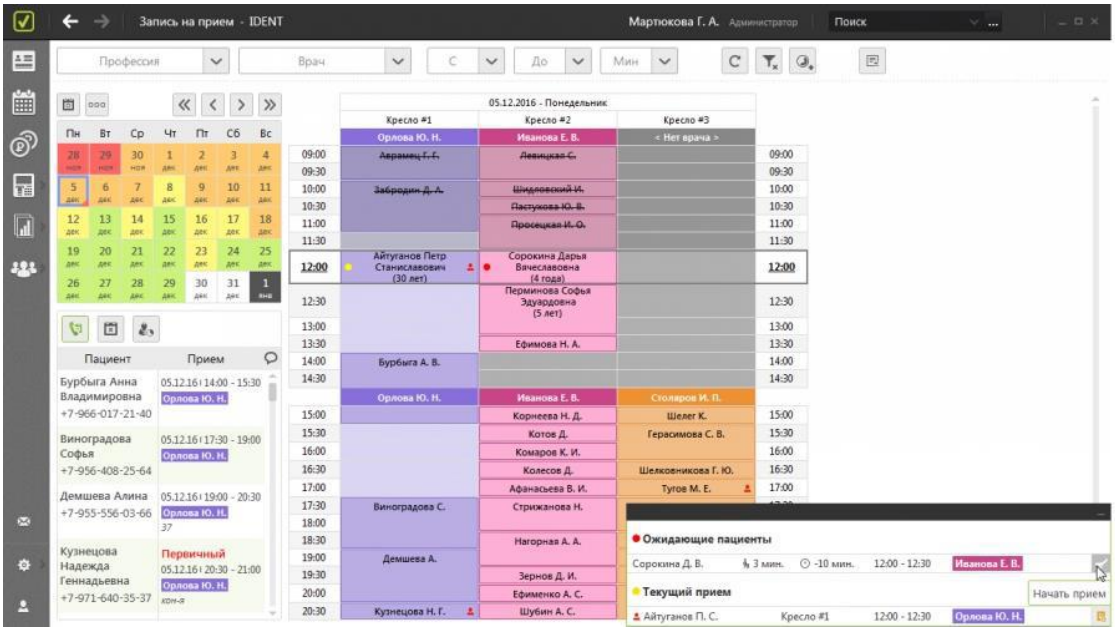

Рисунок 1.2 – Скриншот програми IDENT

1.2.3 Система 1С:Медицина

1С:Медицина – програма для стоматології можна використовувати на одному або декількох комп'ютерах. Кількість робочих місць необмежено і залежить від кількості обраних Вами ліцензій, також з функціоналу: можливість лікаря управляти своєю зайнятістю а також переглядати історію хвороб пацієнта [3].

До переваг цієї системи можна віднести необмежену кількість користувачів, що є дуже зручним для використання додатку цілою клінікою. А також надає можливість лікарям вести медичні картки пацієнтів.

Недоліки:

− висока вартість (необхідно сплачувати кожен місяць);

− немає запису онлайн;

− немає доступу пацієнта до онлайн мед картки;

− складний інтерфейс.

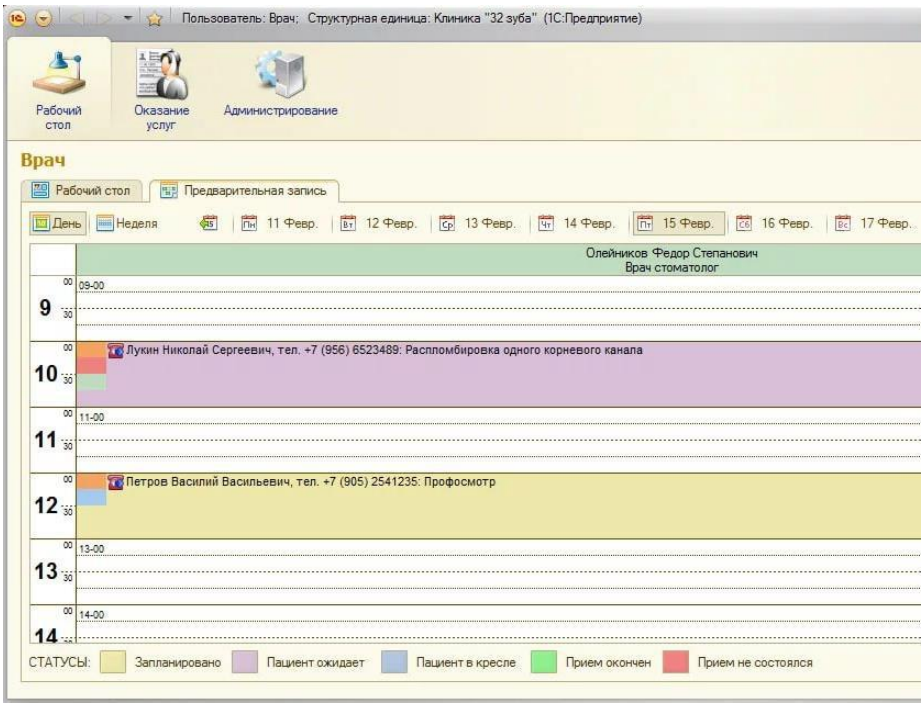

Рисунок 1.3 – Скриншот програми 1С:Медицина

1.2.4 Висновки щодо порівняння аналогів

Зробивши повний аналіз аналогів та побачивши переваги та недоліки інших систем, були сформульовані основні вимоги що до створення даної системи.

Система повинна бути проста у використанні, та допомоги лікарям керувати своєю приватною практикою, а пацієнтам зажди мати свою мед картку під рукою.

У табл. 1.1 були занесені отримані при порівнянні вимоги, систем що вже є на ринку.

Таблиця 1.1 – Порівняння аналогів

Функціонал	QStoma	IDENT	1С:Медицина	Система, що розробляється
Зручний інтерфейс	+	+	–	+
Необмежена кількість користувачів	+	+	+	+
Планування зайнятості лікаря	+	+	+	+
Ведення мед карт	–	+	+	+
Історія захворювань	–	+	+	+
Наявність онлайн медичної карти у пацієнта	–	–	–	+
Запис онлайн	–	+	–	+
Низька вартість	+	–	–	+

Дивлячись на таблицю, можна побачити, що основними недоліками всіх систем, що порівнювались, є те, що пацієнти не мають можливості переглядати власну медичну картку.

Також по таблиці видно, що у таких систем як QStoma та 1С:Медицина, великими недоліками є відсутність онлайн запису. На відмінно від QStoma, системи 1С:Медицина та IDENT, мають високу вартість, від якої залежить який функціонал буде доступний, та скільки лікарів будуть працювати у системі. QStoma коштує набагато дешевше, але неї досить обмежений функціонал.

Система, що розробляється, краща, оскільки вона перекриває всі ті недоліки, які були виявлені у аналогів.

1.3 Вибір архітектурного шаблона та паттернів проектування

Більшість сучасних інформаційних систем побудовані за схожими принципами. З роками ці принципи розвинулись та систематизувались у архітектурні шаблони. Отже, архітектурний шаблон це сукупність належних

практик вирішення архітектурних проблем розробки програмного забезпечення.

Зазвичай програмне забезпечення реалізує декілька архітектурних шаблонів. Тому даний проект матиме гібридну архітектуру, яка складатиметься з три-ланкового архітектурного шаблону та шаблону MVVM.

З рис. 1.4 можна побачити, що першу ланку представляє клієнт. Він реалізує два за стосунки десктоп та мобільний. У свою чергу мобільний застосунок побудований відповідно шаблону MVVM. Це необхідно для швидкого реагування представлення на зміни у моделі.

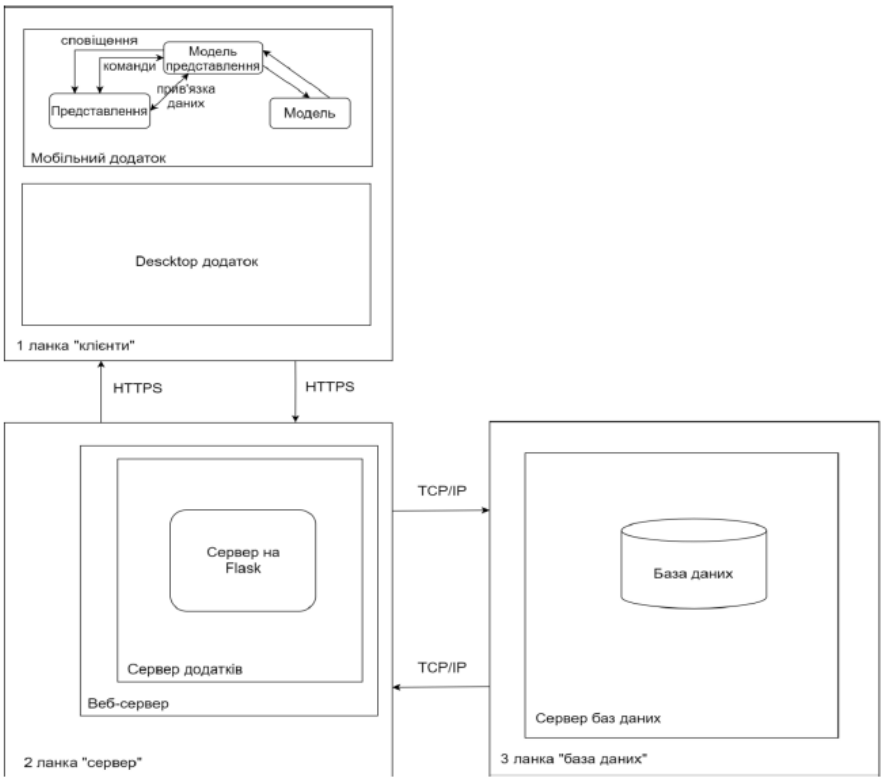

Рисунок 1.4 – Архітектура інформаційної системи

Другу ланку представляє сервер. Серверна частина побудована відповідно принципам REST. Написана серверна частина на Flask виконується за допомогою серверу додатків. Веб-сервер виступає в якості балансувальника навантаження, тобто він приймає запити від клієнтів та перенаправляє всі запити до серверу додатків.

Третя ланка це база даних. На сервері баз даних міститься база даних системи. Взаємодія між ланками відбувається за допомогою протоколу HTTPS. Побудована архітектура має ряд значних переваг: масштабованість, відмово-стійкість, легкість у тестуванні. Одним із значних недоліків даної архітектури виступає затримка запитів між слоями. Дану проблему можна вирішити завдяки використанню кешів.

Нижче подано перелік шаблонів проектування, які допоможуть у створенні даної інформаційної системи.

Шаблон Адаптер – це структурний шаблон проектування, який дозволяє об'єктам з несумісними інтерфейсами працювати разом.

Шаблон Singleton – це породжувальний патерн проектування, який гарантує, що у класу є тільки один екземпляр, і надає до нього глобальну точку доступу.

Шаблон State – це поведінковий патерн проектування, який дозволяє об'єктам змінювати поведінку залежно від свого стану. Ззовні створюється враження, що змінився клас об'єкта.

1.4 Вибір мови та середовища програмування

1.4.1 Вибір мови програмування

Якщо сильно узагальнити, можна сказати, що сучасні мови програмування поділяються на три типи:

− «швидкі», які використовуються для оперативного створення додатків або їхніх прототипів.

− «інфраструктурні», які допомагають оптимізувати або допрацьовувати окремі частини вже написаного додатки для того, щоб підвищити його продуктивність.

− так звані системні мови програмування, використання яких дозволяє отримати в своє розпорядження повноцінний контроль над пам'яттю пристрою.

Звичайно, реальний поділ на типи серед мов програмування менш суворе: є проміжні, гібридні варіанти різних типів.

Оскільки проект складається з трьох основних частин, розглянемо мови програмування для кожної з них. Розпочнемо з десктопної частини проекту. Поглянемо на дві основних мови програмування для розробки додатку для ПК це C++, C#.

C++ – компільований, статично типізована мова програмування загального призначення. Природна для нього область застосування – системне програмування, що розуміється в широкому сенсі цього слова. Підтримує такі парадигми програмування, як процедурне програмування, об'єктно-орієнтоване програмування, узагальнене програмування. Мова має багату стандартну бібліотеку, яка включає в собі поширені контейнери і алгоритми, інструменти для роботи з потоками вводу-виведення, регулярні вирази, підтримку багатопоточності та інші можливості. C++ поєднує властивості як високорівневих, так і низькорівневих мов [4].

C# – це сучасний об'єктно-і компонентно-орієнтована мова програмування. C# надає мовні конструкції для безпосередньої підтримки такої концепції роботи. Завдяки цьому C# підходить для створення і застосування програмних компонентів. Мова C#, розроблений компанією Майкрософт, один з найпопулярніших сучасних мов програмування. Він затребуваний на ринку розробки в різних країнах, C# застосовують при роботі з програмами для ПК, створення складних веб-сервісів або мобільних додатків [5].

Для порівняння цих мов програмування можна використовувати дійсно широке розмаїття критеріїв. Однак, для простоти, були обрані декілька важливих моментів, які будуть використані в цьому порівнянні C# або C++. Ці пункти – швидкість, синтаксис, гнучкість, популярність, простота використання [6].

Спочатку поглянемо на найбільші відмінності C++ та C#, для майбутнього порівняння по критеріям:

− у відмінності від C++, який компілюється у машиний код, C# компілюється у байт код.

− к C# керування пам’ятю набагато простіше ніж C++, а також C# самостійно займається збіркою сміття. C++ не пропонує такої можливості, і потрібно виконувати все управління розподілом пам'яті вручну, але це дає більший контроль над пам’ятю.

− C++ допускає множинне наслідування, а C# – ні. У зв’язку з цим при порівнянні швидкості C# і C++ переможцем виявляється C++. Отже, для програм, що вимагають високої продуктивності, краще вибирати C++. Проте є способи оптимізації коду C#, щоб зробити його швидкість схожою на C++.

− незважаючи на те, що C# працює у всіх популярних операційних системах, він є найбільш популярним в середовищі Windows. C++ не має кращої або більш популярної операційної системи і працює однаково будь де.

− незважаючи на те, що обидві мови програмування досить популярні, цей пункт виграє C++. Ця мова програмування існує набагато довше, ніж C#.

− на питання про синтаксис C++ і C # неважко відповісти. C# має більш лаконічний та більш читабельний синтаксис

− стосовно простоти використання, початківцям може бути складно зрозуміти структуру і згоди C++ оскільки є багато тяжкий речей, таких як: множинне наслідування, вказівники, глобальні функції ті інше чого немає у C#.

Підсумуємо вищесказане і проведемо порівняння за критеріями табл. 1.2.

За критеріями вище обидві мови програмування мають однакову кількість переваг та недоліків. Але поглянемо саме на потрібні переваги для цього проекту це швидкість гнучкість та простота використання. По даним характеристикам більшу перевагу має C++.

Таблиця 1.2 – Порівняння мов C++ та C#

Критерії	C++	C#
Швидкість(за наявності різноманітних інструментів)	+	–
Синтаксис(читабельність)	–	+
Гнучкість (можливість керування пам'яттю в ручну)	+	–
Простота використання	–	+
Популярність	+	+

Для розробки десктопної версії була обрана мова програмування C++, оскільки вона має більшу швидкість та гнучкість, а недолік стосовно простоти використання покривається хорошим володінням цієї мови програмування.

Перейдемо до наступної частини проекту – андроїд додатку, та проведемо порівняння стосовно мов програмування. Основними мовами програмування під андроїд є Java і Kotlin, отож розглянемо детальніше їх переваги та недоліки саме у мобільній розробці.

Java – об'єктно-орієнтована мова програмування. Java дозволяє розробляти та розгортати програми на робочих столах та серверах. Java та компонентні технології пропонують багатий користувальницький інтерфейс, продуктивність, універсальність, портативність та безпеку, що вимагають сучасні програми.

Java являє собою універсальну платформу для створення прикладного програмного забезпечення: серверної логіки; розподілених систем; вебдодатків; десктопних програм; мобільних додатків.

Основні можливості Java:

− простота у Java чіткі синтаксичні правила і зрозуміла семантика.

− об'єктно-орієнтований підхід. У центрі уваги знаходяться дані (об'єкти), тобто все є об’єктом, який має певні дані та поведінку інтерфейси і алгоритми вторинні.

− безпека і надійність. Компілятор здатний виявити помилки ще до виконання коду, тобто на ранніх стадіях [4]. При компіляції програма Java компілюється в байт-код. Цей байт-код не залежить від платформи і може бути запущений на будь-якому комп'ютері, плюс цей формат байт-коду також забезпечує безпеку.

− універсальність. Написання програмного забезпечення на одній платформі і його запуск практично на будь-який інший платформі.

Kotlin – це статично типізований мова. Він підтримує як об'єктно-орієнтоване, так і процедурне програмування [5].

Ось основні можливості Kotlin:

− як і Java компілюється в байткод JVM;

− програми можуть використовувати всі існуючі Java-фреймворки і бібліотеки. Kotlin можна інтегрувати з Maven, Gradle і іншими системами збірки;

− мова дуже проста для вивчення

− мова програмування null-безпечна. Тобто у відмінності від Java, у Kotlin при спробі привласнення або повернення null код не скомпілюється.

Тепер перейдемо до основних відмінностей:

− Null-безпека. Як вже говорилося раніше, Kotlin не допускає виникнення NullPointerException, видаючи помилку компіляції.

− Kotlin дозволяє розширювати функціональність існуючих класів, не вдаючись до спадкоємства. Це робиться за допомогою функцій-розширень

− розумні приведення типів. У більшості випадків не потрібно явно вказувати оператори приведення, оскільки в мові є спеціальний оператор.

− збірка Kotlin-коду займає приблизно на 15-20% більше часу, ніж аналогічний процес на Java.

− розробка додатків для Android давно і тісно пов'язана з Android Studio. Дане середовище спочатку була заточена під роботу з Java, а тому код можна писати буквально по одній букві - IDE самостійно підтягне все необхідне.

− Kotlin – молода мова, і невідомо, що буде далі, в той час як Java характеризується кроссплатформенную: на ній не тільки мобільна розробка тримається, але і бекенд з десктопом.

Відштовхуючись від порівняння вище, для розробки мобільного додатку була обрана мова програмування Java. Оскільки Java займає не так багато часу для збірки, а більшою перевагою є знання цієї мови, не дивлячись на те, що середовище Android Studio спочатку була заточена під роботу з Java.

Тепер поглянемо на серверну частину проекту. Проведемо порівняння серед мов програмування, враховуючи їх переваги та недоліки. Основними мовами для написання серверної частини є Python та Java.

Python – це об'єктно-орієнтована мова програмування високого рівня. Він має вбудовані структури даних в поєднанні з динамічною типізацією і прив'язкою, що робить його ідеальним вибором для швидкої розробки додатків. Python також пропонує підтримку модулів і пакетів, що дозволяє використовувати модульність системи і повторне використання коду.

Це одина з найшвидших мов програмування, так як він вимагає дуже мало рядків коду. Акцент робиться на удобочитаемости і простоті, що робить його відмінним вибором для початківців Ось деякі причини, за якими вам слід вибрати Python:

− у порівнянні з кодом іншої мови код Python легко писати і налагоджувати. Тому його вихідний код відносно простий в обслуговуванні.

− Python може працювати на самих різних операційних системах і платформах.

− Python поставляється з безліччю вбудованих бібліотек, що полегшує завдання розробки.

− Python допомагає спростити складне програмування. Він внутрішньо обробляє адреси пам'яті, прибиральників сміття.

− Python пропонує високорівневі динамічні типи даних, а також підтримує динамічну перевірку типів.

PHP розшифровується як гіпертекстовий препроцесор. Це серверний мова скриптів. Він використовується для розробки динамічного веб-сайту або веб-додатки. PHP може легко інтегруватися з усіма основними веб-серверами в усіх основних операційних системах. Причини обрати PHP:

− працює на різних платформах, таких як Windows, Unix, Linux, Mac OS X і т. д.

− інтеграція баз даних. Підтримує безліч баз даних, таких як Oracle, MySQL і т. Д.

− легко вивчити, він простий у використанні.

− сумісний практично з усіма Apache, IIS серверами.

У порівнянні PHP або Python варто обумовити відразу одне з основних відмінностей. Мова Python – це високорівнева мова програмування загального призначення, а PHP – скриптова мова [9].

Ще одне велика різниця між двома мовами полягає в тому, що Python є об'єктно-орієнтованим, а PHP – тільки частково.

Навіть після вище сказаного вибір однієї з них, безсумнівно, є складним завданням. Але аналізуючи вищесказане, можна зробити висновок, що незважаючи на те, що Python не підтримує підключення до БД так широко, як PHP, у нього є ряд таких переваг, як читабельність, більш швидка розробка у порівняні з PHP, та більш легша мова для засвоєння. Саме тому для написання серверної частини була обрана мова програмування Python.

1.4.2 Вибір середовища програмування

Оберемо середовища програмування для декстоп, мобільної та серверної частин. Спочатку розглянемо універсальний редактор коду Visual Studio Code, а вже потомі порівняємо його з більш спеціалізованими редакторами: Android Studio (для Android), PyCharm (для Python), QtCreator (для C++ та фреймворка Qt, який буде розглянуто в наступному пункті).

Visual Studio Code – це сервіс, який позиціонується як легкий редактор коду для кросплатформенної розробки різних видів додатків та програм. В редактор інтегрована велика кількість функій, таках як налагодження та рефакторинг коду, робота з системою контролю версій Git, автодоповнювання типових конструкцій, підсвітка синтаксису та навігація по коду. Visual Studio Code підтримає велику кількість мов програмування, платформ розробки та фреймворків. Наприклад, якщо необхідно розробити мобільний додаток для Android, достатньо завантажити спеціальне розширення, що додасть редактору підтримку необхідних функцій.

PyCharm – це кросплатформенне інтегроване середовище програмування для Python. Середовище має ефективну навігацію, підтримку різних фреймворків та бібліотек для Python, потужний інструментарій для тестування, профілювання та інспекції коду. У PyCharm інтегрована підтримка різних систем контролю версій, повнофункціональний відладник коду, а також можливість інтеграції з популярними сервісами (AWS, різноманітні багтрекери).

Середовище надає допомогу при написанні коду у вигляді автодоповнювання, автовиправлення, автоформатуванні коду та багато іншого. Присутнє автоматичне імпортування модулів.

PyCharm славиться своїм потужним рефакторингом коду, який надає широкі можливості виконання глобальних виправлень в проекті.

Android Studio – це кросплатформенне інтегроване середовище програмування для роботи з платформою Android. Android Studio заснований на технології компанії JetBrains, тому середовище має однакові можливості в аналізі коду, інструментарії та інтеграції зі сторонніми сервісами як і у PyCharm. Однак, Android Studio має більш спеціалізовані інструменти та можливості необхвдні для розробки під Android. Наприклад, у середовище інтегровано редактор макетів, який дозволяє графічними засобами створювати UI для проекту, непишучи при цьому код. Також середовище має декільна вбудованих збірників проектів та надає генерацію .apk файлів.

Android Studio підтримує та емулює значну частку пристроїв під управлінням ОС Andriod. Також має велику кількість невеличких прикладів, які дозволять на практиці розібрати складні аспекти платформи.

Qt Creator – кросплатформеменне середовище програмування з вікритим кодом для розробки на C, C++ та фрейморці Qt. Як і інші популярні редактори коду, Qt Creator має значні успіхи у синтаксичному аналізі, редагуванні, відладці, профілювананні та рефакторинзі коду. Має інструменти для тестування та інтегровані збірники проектів.

Оскільки Qt Creator розроблявся для спрощення розробки додатків за допомогою фреймворка Qt, тому серед можливостей, притаманних усім середовищам програмування, існують спеціалізовані інструменти та можливості. Наприклад, вбудований редактор/дизайнер інтерфейсів, відладчик додатків на QML та відображення даних з контейнерів Qt.

Отже, вище було розглянуто універсальний редактор коду Visual Studio Code та спеціалізовані редактори Android Studio, PyCharm, Qt Creator. Оскільки, спеціалізовані середовища програмування мають більший функціонал та інструментарій для роботи зі своєю технологією, тому майбутня розробка буде проводитися з використанням вище вказаних спеціальних середовищах програмування.

1.5 Вибір технологій для розробки інформаційної системи

При розробці проекту, особливо актуальним є питання про вибір технологій. Тому при перегляді можливих технологій для даного проекту вибір був зроблений в сторону декількох технологій щодо десктопа та декілька технологій щодо серверу. Серед технологій для десктопу такі як: Qt та SFML. Для серверу такі технології як: Flask та Django.

Тепер перейдемо до порівняння технологій Qt та SFML. Але перед цим, слід сказати, що Qt є фреймворком і схожого аналогу серед фреймворків не

існує, він єдиний у своєму роді, тому для порівняння була взята бібліотека SFML. Розпоченемо з фреймворку Qt.

Qt – багатоплатформовий фреймворк для розробки програмного забезпечення на мові програмування C++. Qt повністю об'єктно-орієнтований, крос-платформний. Дає можливість розробляти платформо-незалежне ПО, написаний код можна компілювати для Linux, Windows, Mac OS X і інших операційних систем. Включає в себе безліч класів для роботи з мережею, базами даних, класи-контейнери, а також для створення графічного інтерфейсу і безліч іншого.

В Qt є величезний набір віджетів (Widget), таких як: кнопки, прогрес бари, перемикачі, checkbox, і інші – вони забезпечують стандартну функціональність GUI (графічний інтерфейс користувача).

Qt має середовище розробки Qt Creator. Вона включає в себе Qt Designer, за допомогою якого можна створювати графічний інтерфейс.

Бібліотека Qt складається з різних модулів, які можна підключити. Туди входять модулі для роботи з мережею, базою даних, графічним інтерфейсом та інші [10].

SFML (Simple and Fast Multimedia Library) – одна з найбільш зручних і швидких графічних бібліотек для C++. Написана на мові C++. Її незаперечна перевага – мінімальні вимоги до рівня знань мови і легкість освоєння.

SFML забезпечує простий інтерфейс для різних компонентів вашого ПК, щоб полегшити розробку додатків. Він складається з п’яти модулів: системного, віконного, графічного, аудіо та мережевого. За допомогою SFML ваш додаток може компілюватись і запускатись із найпоширеніших операційних систем: Windows, Linux, macOS і незабаром Android та iOS.

SFML складається з різних модулів:

− System – векторні і строкові класи Unicode, що переносяться кошти потокової передачі і таймера

− Window – управління вікнами і пристроями введення, включаючи підтримку джойстиків, управління контекстом OpenGL

− Graphics – апаратне прискорення 2D-графіки, включаючи спрайт, багатокутники і рендеринг тексту.

− Audio – просторове відтворення і запис звуку з апаратним прискоренням

− Network – мережеві сокети TCP і UDP, засоби інкапсуляції даних, класи HTTP і FTP.

Виходячи з вище сказаного, SFML більше підходить для створення ігрових меню, а ніж для написання даного додатку. А з огляду на те що Qt має власне середовище розробки, графічний редактор та більше можливостей для розробки додатку (віджети, модулі для роботи з мережею та інше) вибір очевидний. Розробка буде проводитися на Qt.

Перейдемо до порівняння Flask та Django.

Django – вільний фреймворк для веб-додатків на мові Python, що використовує шаблон проектування MVC. Він дозволяє без особливих складнощів створювати динамічні веб-додатки. Фраза «все включено» означає, що більшість інструментів для створення програм – частина фреймворку, а не а поставляються у вигляді окремих бібліотек.

Для даної технології можна виділити наступні особливості:

− швидкість. Був розроблений, щоб допомогти розробникам створити додаток настільки швидко, на скільки це можливо;

− вбудована ORM;

− особливий пріоритет віддається безпеки;

− можливість масштабувати проект.

Django складається з проекту, який ділиться на додатки. Проект – коренева папка і глобальні налаштування. Додатки – це функціонал сайту, розбитий по різних модулів. У кожного проекту вони свої, але їх можно повторно використовувати у нових або інших проектах.

Django вирішує серйозні проблеми, тому він такий популярний серед гігантів YouTube та Instagram. І цей факт говорить про те, що він призначений для величезних проектів завдяки своєму великому вбудованому функці-

оналу. Хоча, наприклад, вбудована ORM не може похвалитися своєю гнучкістю по відношенню до SQLAlchemy. Таким чином, надлишок модулів може виявитися непотрібним задоволенням для невеликих і середніх проектів.

Flask – легкий та гнучкий мікрофреймворк для створення веб-додатків на мові програмування Python. Він не надає ORM і поставляється тільки з базовим набором інструментів для веб-розробки, таких як шаблоні затор Jinja 2 та WSGI сервер під назвою Werkzeug. У зв’язку з цим, підключається бібліотека SQLAlchemy**.**

Одним з проектних рішень під Flask є те, що прості завдання повинні бути простими; вони не повинні займати багато коду, і це не повинно обмежувати.

У Flask немає жорсткої структури. Можна поселити моделі, контролери та ініціалізацію в одному файлі, можна – в різних. Спочатку проект складається з порожньої папки. Структура формується по ходу розробки, у кожного виходить щось своє [11].

Оскільки розміри проекту невеликі, більшість функціоналу django не потрібні, тому розробка серверної частини буде проходити з використанням фреймворку flask та підключенням до нього необхідних бібліотек та розширень.

Нижче подано перелік необхідних бібліотек, технологій та розширень для швидкої та комфортної розробки серверної частини на Flask:

SQLAlchemy – це програмна бібліотека мовою Python для роботи з реляційними СУБД з застосуванням технології ORM. Служить для синхронізації об'єктів Python і записів реляційної бази даних. SQLAlchemy дозволяє описувати структури баз даних і способи взаємодії з ними на мові Python без використання SQL. SQLAlchemy може похвалитися своєю гнучкістю на відмінно від вбудованого ORM у Django.

Платформа android надає усі необхідні можливості для створення мобільних додатків тому не має необхідності у використанні фреймворків. Але є необхідність у використанні бібліотек:

Retrofit – це REST-клієнт для Java і Android. Він дозволяє легко отримати і завантажити JSON (або інші структуровані дані) через веб-сервіс на основі REST. Для HTTP-запитів використовується бібліотека OkHttp.

Room – заснована на SQLite бібліотека ORM. Room є абстракцією над слоем, в якому виконуються конкретні дії над базою даних, такі як вставки, видалення, створення таблиць і так далі. Room використовує анотації для генерації коду під час компіляції.

1.6 Вибір системи управління базами даних

Реляційна модель даних, яка впорядковує дані в таблицях рядків і стовпців, переважає в інструментах управління базами даних. Існують інші моделі даних, включаючи NoSQL і NewSQL, але системи керування базами даних (СУБД) залишаються домінуючими для зберігання і управління даними.

Тому порівняємо следующиие СУБД з відкритим вихідним кодом: SQLite, MySQL і PostgreSQL.

PostgreSQL, також відомий як Postgres, позиціонує себе як «сама просунута в світі реляційна база даних з відкритим кодом». Він був створений з метою забезпечення високої розширюваності і відповідності стандартам.

PostgreSQL – це вільна об'єктно-реляційна система управління базами даних, заснована на мові SQL, а це означає, що хоча це в першу чергу реляційна база даних, вона також включає в себе такі функції, як наслідування таблиць і перевантаження функцій, які частіше пов'язані з об'єктними базами даних [12].

PostgreSQL здатний ефективно обробляти декілька завдань одночасно, що називається паралелізмом. Це досягається без блокувань читання завдяки реалізації Multiversion Concurrency Control (MVCC), яка забезпечує атомарность, узгодженість, ізоляцію і довговічність транзакцій, також відому як ACID.

PostgreSQL не так широко використовується, як MySQL, але все ще існує ряд сторонніх інструментів і бібліотек, призначених для спрощення роботи з PostgreSQL, включаючи pgAdmin і Postbird [13].

MySQL – це реляційна система управління базами даних з відкритим вихідним кодом [14].

MySQL був розроблений для забезпечення швидкості і надійності за рахунок повної відповідності стандарту SQL. На відміну від додатків, що використовують SQLite, додатки, що використовують базу даних MySQL, отримують доступ до неї через окремий процес-демон. Оскільки серверний процес стоїть між базою даних і іншими додатками, він дозволяє краще контролювати, хто має доступ до бази даних.

З переваг СУБД MySQL варто відзначити простоту використання, гнучкість, низьку, а також масштабованість і продуктивність.

Гнучкість СУБД MySQL забезпечується підтримкою великої кількості типів таблиць: користувачі можуть вибрати як таблиці типу MyISAM, що підтримують повнотекстовий пошук, так і таблиці InnoDB, що підтримують транзакції на рівні окремих записів. Є й інші типи таблиць, розроблені спільнотою.

SQLite – це автономна, файлова СУБД з повністю відкритим вихідним кодом, відома своєю портативністю, надійністю і високою продуктивністю навіть в середовищах з низьким обсягом пам'яті [15]. Його транзакції є ACID-сумісними, навіть в тих випадках, коли система виходить з ладу або піддається відключення електроенергії.

SQLite описують як «бессерверну» базу даних. Більшість механізмів реляційних баз даних реалізовані як серверний процес, в якому програми взаємодіють з хост-сервером за допомогою взаємодії між процесами, які передають запити. Однак з SQLite будь-який процес, який звертається до бази даних, читає і записує в файл диска бази даних безпосередньо. Це спрощує процес установки SQLite, оскільки усуває необхідність в налаштуванні серверного процесу. Точно так само немає необхідності в налаштуванні про-

грам, які будуть використовувати базу даних SQLite: все, що їм потрібно - це доступ до диска.

SQLite – це безкоштовне програмне забезпечення з відкритим вихідним кодом, і для його використання не потрібно спеціальної ліцензії.

Для даного проекту обрана база даних PostgreSQL. У PostgreSQL безліч можливостей. Створений з використанням об'єктно-реляційної моделі, він підтримує складні структури і широкий спектр вбудованих і обумовлених користувачем типів даних. Він забезпечує розширену ємність даних і заслужив довіру обережним ставленням до цілісності даних.

1.7 Висновки до першого розділу

Під час виконання першого розділу була сформульована постановка задачі. Завдяки порівнянню аналогів, було виявлено недоліки та переваги систем, та на основі цього був зроблений функціонал даної системи, яка є унікальною та буде нейтралізувати недоліки порівнюваних систем.

Також були розглянуті необхідні технології для реалізації системи.

Середовище розробки desktop версії буде QT Creatoror, який включає в себе Qt Designer, за допомогою якого можна створювати графічний інтерфейс. Для android додатку була обрана Android Studio. Серверна частина буде створена за допомогою фреймворку flask.

Для розробки desktop додатку була обрана мова програмування C++, для android додатку – Java, а для серверної частини була обрана мова програмування Python.

2 ПРОЕКТУВАННЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ПРИВАТНИЙ КАБІНЕТ ЛІКАРЯ СТОМАТОЛОГА»

2.1 Мета та задачі системи

Дана інформаційна система призначена для лікарів стоматологів та пацієнтів. Система складається з десктопної частини для лікарів та андроїд частини для пацієнтів.

Мета ІС: Створити інформаційну систему, яка допомагає лікарям вести облік пацієнтів та планувати свій графік, а пацієнтам надає можливість завжди мати під рукою свою медичну картку, а також без зайвих клопотів записатися на візит до лікаря.

Десктоп додаток допоможе лікарям швидко записати на прийом пацієнта або додати лікування. Андроїд додаток допоможе пацієнтам нагадати лікування та час прийому, переглянути мед карту.

Головними функціями цієї системи є:

− реєстрація та авторизація пацієнтів;

− редагування акантів пацієнтів;

− видалення акаунтів пацієнтів;

− додавання нотатків у лікаря;

− запис на прийом;

− перегляд мед карти та історії захворювань з призначеним лікуванням;

− призначення лікування у мед карті;

− підтвердження або відміна запису;

− оповіщення про підтвердження/відміну запису.

Вхідні дані:

− ПІП, номер телефону, дата народження, вибір дати візиту, вибір лікаря, опис причини відвідування;

− нотатки лікаря, призначене пацієнту лікування, опис захворювання, дата візиту, вибір клієнта.

Вихідні дані: Розпланований розклад лікаря, історія захворювань та лікування, медичні картки.

Інформаційні потоки системи зображені нижче.

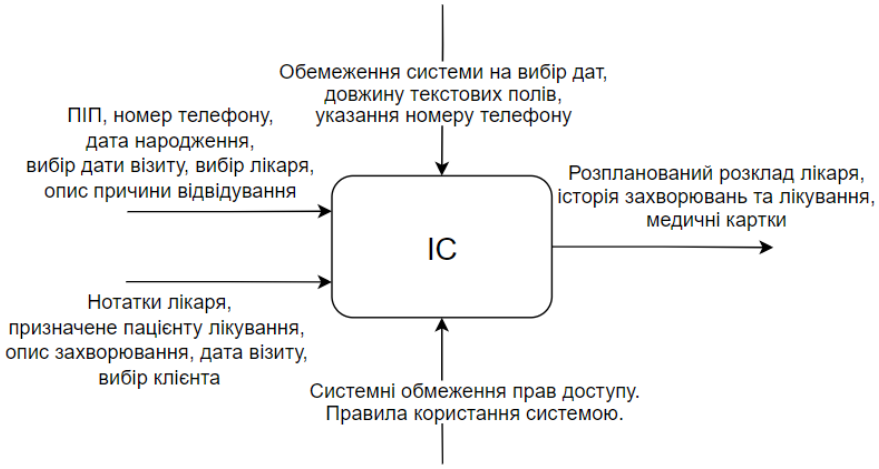

Рисунок 2.1 – Діаграма вхідних і вихідних інформаційних потоків

2.2 Типи користувачів

Гість – це незареєстрований користувач, він буде взаємодіяти з Android додатком. Даний тип користувачів може зареєструватися та переглянути зміст головного екрану.

Пацієнт – це зареєстрований в системі користувач, який користується Android додатком та може редагувати/видалити свій профіль, записатися на візит до лікаря, переглянути свою мед картку в якій є історія хвороби та призначене лікування, а також може подивитися дату найближчого візиту, подивитися адресу клініки та обрати лікаря.

Лікарь – це користувач системи, який користується Desktop додатком, він може записати на прийом пацієнта, підтвердити/відмінити дату прийому, додавати лікування у мед карту пацієнта, робити замітки, подивитися історію хвороби пацієнта.

Адміністратор – це користувач, який буде займатися додаванням нових лікарів у систему, або видаляти лікарів які більше не працюють у клініці.

Діаграма прецедентів зображена на рисунку 2.2. З рисунку видно, що є 4 актори: гість, пацієнт, лікар та адміністратор. Можна зауважити, що пацієнт та наслідує функціональні можливості від актора гість. Гість може зареєструватися у системі та переглянути головний екран мобільного додатку. Пацієнт може упрвялти своїм аккаунтом (видалити або редагувати), записатися на візит до лікаря, авторизуватись переглядати адреси клінік, список лікарів та свою медичну картку. Лікар може переглядати медичну картку пацієнта, призначати йому лікування, авторизуватись, додавати нотатки та підтверджувати запис на візит. Адміністратор має можливість зареєструватися та авторизуватися, а також реєструвати та видаляти з системи лікарів.

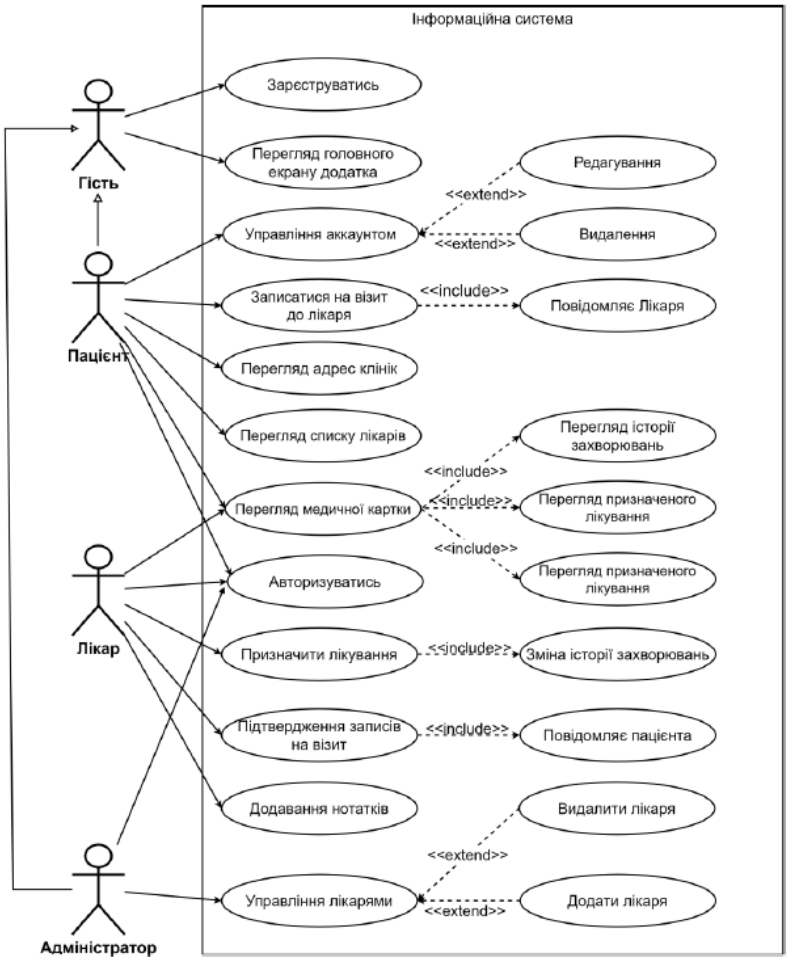

Рисунок 2.2 – Діаграма прецедентів

2.3 Користувацький інтерфейс (UI View)

Дана система складається з 2 додатків: Desktop лікарів та Android для пацієнтів. Перед створенням додатків спершу потрібно розробити макети кожного екрану та визначитись з дизайном.

Для початку розглянемо мокапи для android додатку.

Android частина буде складатися з таких екранів:

− екран авторизації;

− екран реєстрації;

− головна сторінка, яка включає переходи на такі екрани: Запис на консультацію, медична картка, лікарі, адреса клініки.

Розглянемо основні екрани, а на решту будемо посилатися у Додаток А.

При завантажені додатку користувачем він потрапляє на головну сторінку, з якої він може продивитися всі екрани та зареєструватися або авторизуватися, оскільки функціонал не буде доступний. На екрані реєстрації користувач, повинен заповнити форму: ім’я, прізвище, логін, пароль. А потім натиснути кнопку підтвердити (рисунок А.1).

У випадку якщо акаунт існує, користувач переходить на екран авторизації та вводить логін і пароль, щоб увійти в акаунт (рисунок А.2).

Після реєстрації користувача, на головному екрані з’являється можливість користуватися такимим функціями, як: запис на консультацію, перегляд медичної картки, перегляд лікарів, перегляд адресу(и) клініки (рисунок 2.2).

Потрапляючи на екран «Запис на консультацію» користувач має заповнити форму яка складається з наступних полів: вибір лікаря, заповнення дати та опис, де коротко має бути написано причину візиту. При натисканні на стрілку, або на кнопку «Підтвердити», користувач потрапляє на головний екран.

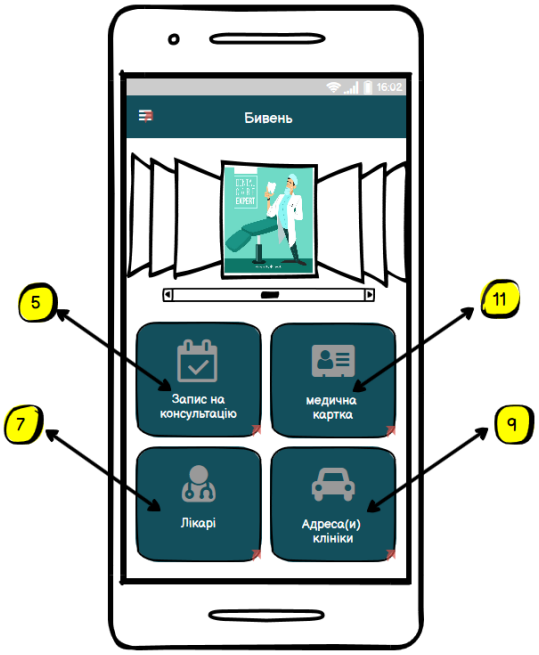

Рисунок 2.3 – Головний екран після авторизації

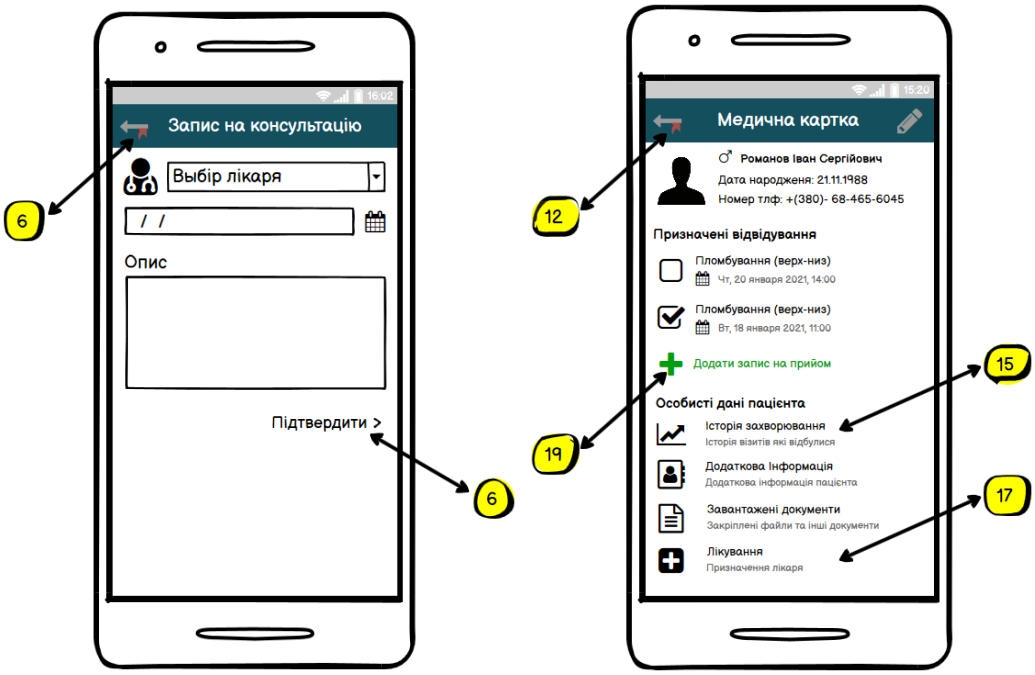

Рисунок 2.4 – Екран «Запис на консультацію» та екран «Медична картка»

При потраплянні на екран «Лікарі», користувач може переглянути всіх можливих лікарів для запису на візит. При натисканні на «стрілку», користувач потрапляє на головний екран.

Далі користувач може зайти на екран «Медична картка», де може подивитися історію захворювань, лікування, дату наступного візиту, записатися на візит, або відредагувати свій профіль .

При натисканні на стрілку, користувач перейде на головний екран. Якщо натиснути на «Додати запис на прийом», користувач перейде до екрану «Запис на консультацію».

При потраплянні на екран «Адреса(и) клініки», користувач може подивитися адресу клініки. Якщо натиснути на стрілку, користувач перейде на головний екран (рисунок А.3).

На діаграмі нижче зображені переходи у android додатку.

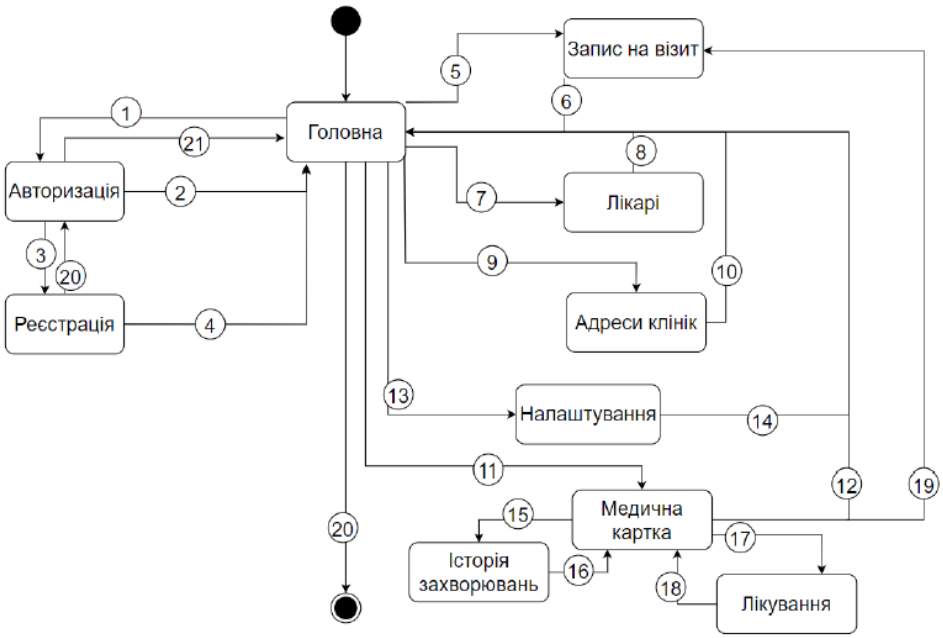

Рисунок 2.5 – Діаграма переходів для Android

Тепер розглянемо мокапи для desktop додатку. Desktop частина буде складатися з таких екранів:

– екран авторизації;

– головний екран;

– медичні карти;

– заявки;

– електронний журнал.

Спочатку лікар потрапляє на екран авторизації, де має авторизуватися. Після успішної авторизації лікар потрапляє на головний екран, де він може подивитися розклад на сьогодні, а також додати якісь необхідні нотатки.

Далі екран «Мед карти», де є пошук по пацієнтам. При виборі необхідного пацієнта, з правої сторони з’являється медична картка цього пацієнта, його дані, а також можливість продивитися історію хвороби.

Рисунок 2.6 – Головний екран лікаря

Натиснувши на кнопку лікування, відкриється віджет де лікар може заповнити форму та додати лікування, заповнивши форму, де необхідно вказати: хворобу, симптоми та призначити лікування.

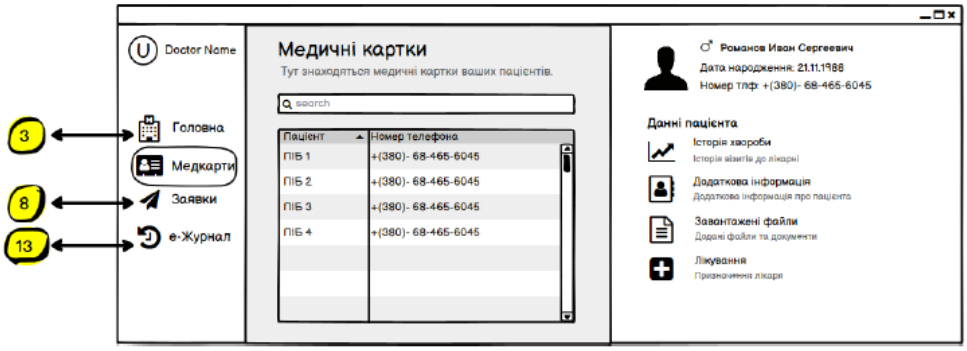

Рисунок 2.7 – Екран «Мед карти»

Натиснувши на кнопку «Історія хвороби», лікар може продивитися минулі захворювання (рисунок А.4)

Екран «Заявки», відповідає за підтвердження запису, тобто коли пацієнта записується онлайн, лікарю приходить приходить заявка на запис, яку він може підтвердити, або відмінити (рисунок А.5).

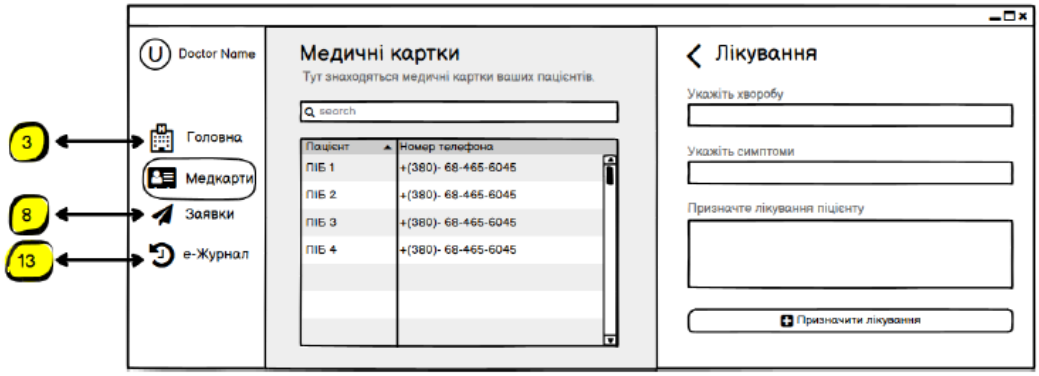

Рисунок 2.8 – Екран «Мед карти», лікування

Останній екран – це «е-Журнал». Тут лікар може планувати свій розклад, та записувати пацієнтів на наступний візит, або ж скасувати його (рисунок А.6).

Нижче наведено діаграму переходів між екранами для desktop додатку.

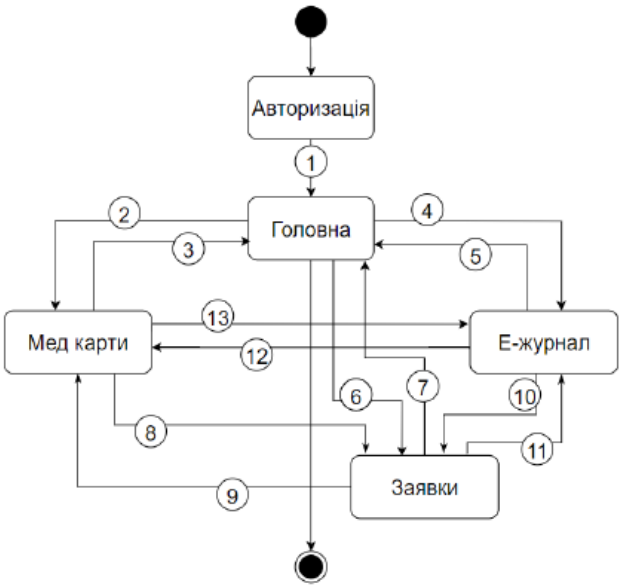

Рисунок 2.10 – Діаграма переходів для Desktop

2.4 Логічне представлення ІС (Logical View)

Побудуємо діаграму логічного представлення системи (рисунок 2.12). З рисунку № видно, що з системою взаємодіють 2 типи користувачів: лікар та пацієнт. Лікар взаємодіє з настільним додатком, який відображає інформацію лікарю. Пацієнт взаємодіє з мобільним додатком на базі Android, який у свою чергу відображає інформацію пацієнту. Настільний та мобільний додатки надсилають запити на сервер, а сервер повертає їм результат у форматі JSON. У свою чергу сервер надсилає SQL-запити до бази даних, яка відповідь повертає дані.

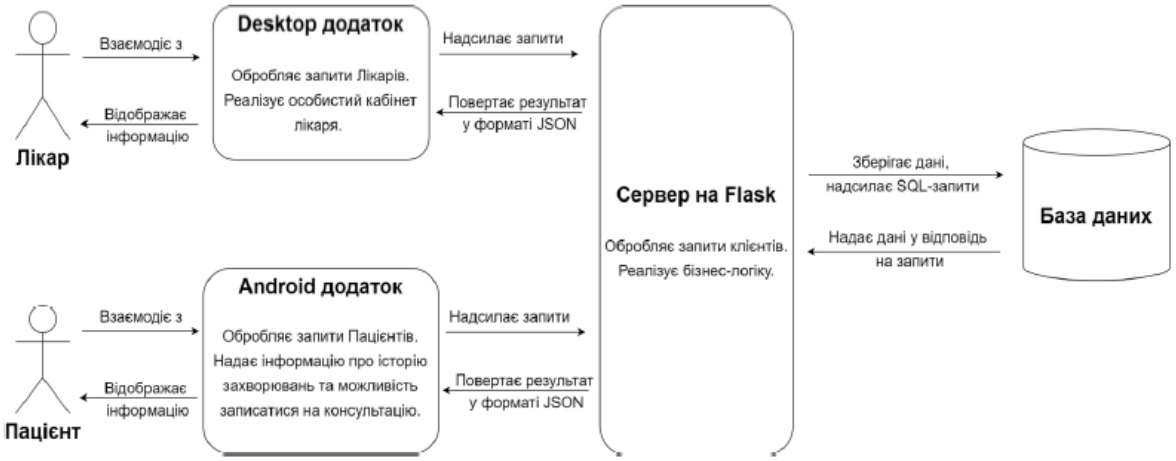

Рисунок 2.11 – Логічне представлення ІС (Logical View)

2.5 Уявлення процесів ІС (Process View)

Система має два основні процеси: запис на візит до лікаря та призначення лікування пацієнту. Розглянемо їх детальніше.

Сценарій 1 – Запис на візит до лікаря.

Для запису на візит пацієнт має натиснути відповідну кнопку. Android додаток відображає користувачу екран запису на візит. Користувач заповнює необхідні поля підтверджує запис. У свою чергу додаток відправляє на сервер дані для їх збереження. Сервер обробляє запит на простить базу даних

зберегти дані. База даних зберігає інформацію, а сервер повідомляє лікаря та android додаток про успішно виконаний запит. Android додаток відображає пацієнту повідомлення про успішний запис на візит до лікаря. На рисунку 2.12 та рисунку Б.1 зображені діаграми діяльності та послідовності для даного сценарію.

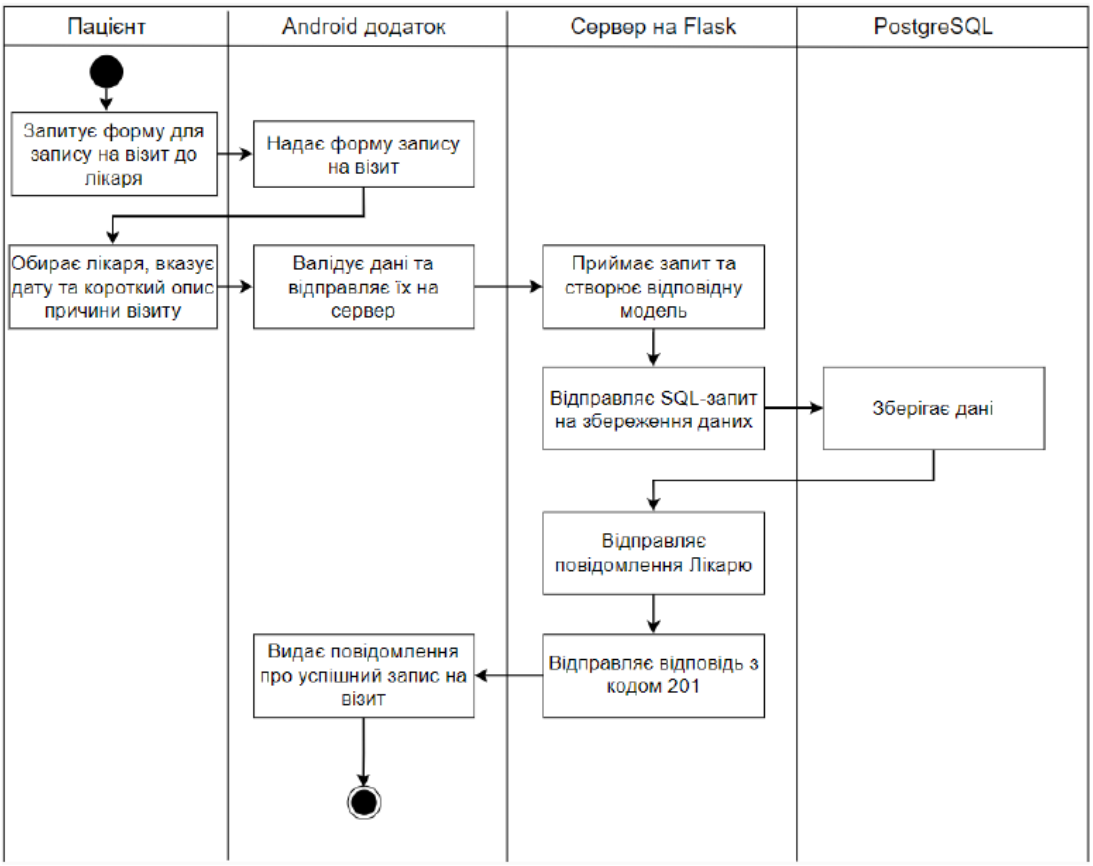

Рисунок 2.12 – Діаграма діяльності для процесу запису на візит до лікаря

Сценарій 2 – Призначення лікування пацієнту

Для призначення лікування пацієнту лікар обирає зі списку медичну картку одного з пацієнтів. Desktop додаток відображає інформацію про його медичну картку. У медичній картці лікар має обрати відповідний розділ для призначення лікування. Додаток надасть екран з формою, яку лікар заповнює. Після підтвердження призначення лікування, додаток повідомляє

сервер про намір зберегти призначення. Сервер просить базу даних зберегти інформацію, а база даних зберігає її. Після успішного збереження даних, сервер повідомляє desktop додаток, що його запит виконано. У свою чергу desktop додаток відображає лікарю повідомлення про успішне призначення лікування пацієнту. На рисунку 2.13 та рисунку Б.2 зображені діаграми діяльності та послідовності для даного сценарію.

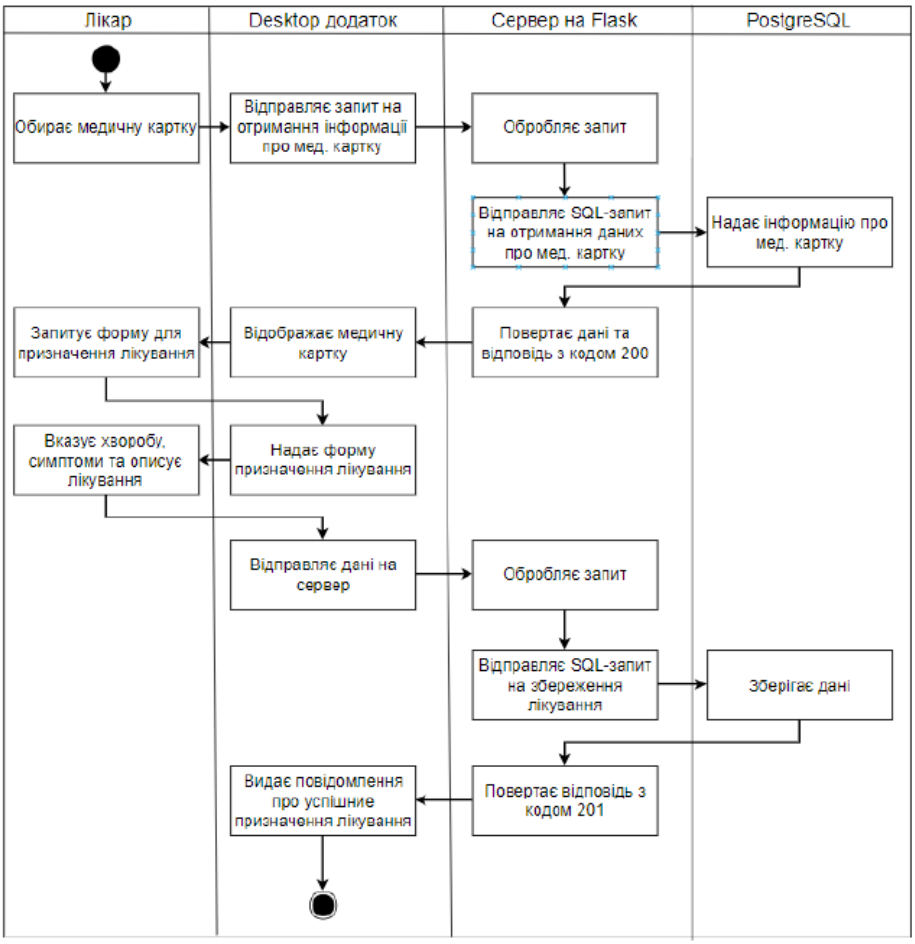

Рисунок 2.13 – Діаграма діяльності для процесу призначення лікування пацієнту

2.6 Уявлення даних ІС (Data View)

У якості сховища даних буде використовуватись реляційна база даних PostgreSQL. База даних взаємодіє з сервером через протокол TCP/IP, з ука-

занням хоста, назви бази даних, юзера та пароля. Для спрощення взаємодії бази даних з сервером використовується SQLAlchemy ORM. PostgreSQL буде використовувати рівень ізоляції транзакцій за замовчуванням, тобто рівень Read Commited. Далі розглянемо сутності бази даних.

База даних складається з таких сутностей (рисунок 2.14):

− Doctor – містить в собі інформацію про лікаря. Має атрибут id, який виступає у якості первинного ключа.

− Note – містить в собі нотатки лікаря. Має атрибут id, який виступає у якості первинного ключа. Дана сутність пов’язана зв’язком один-до-багатьох з Doctor, а у якості зовнішнього ключа виступаю атрибут doctor_id.

− Treatment – містить в собі інформацію про лікування. Має атрибут id, який виступає у якості первинного ключа. Дана сутність пов’язана зв’язком один-до-багатьох з Type_of_work, а у якості зовнішнього ключа виступаю атрибут doctor_id.

− Type_of_work – містить в собі інформацію про тип робіт лікаря. Має атрибут id, який виступає у якості первинного ключа.

− Visit – містить в собі інформацію про вже відвідані візити (історія візитів). Має атрибут id, який виступає у якості первинного ключа. Дана сутність пов’язана зв’язком один-до-багатьох з Patient, а у якості зовнішнього ключа виступаю атрибут patient _id.

− Patient – містить дані про пацієнт. Має атрибут id, який виступає у якості первинного ключа.

− Appointment_for_visit – містить інформацію про плановані візити. Має атрибут id, який виступає у якості первинного ключа. Дана сутність пов’язана зв’язком один-до-багатьох з Status, а у якості зовнішнього ключа виступаю атрибут status_id, також сутність пов’язана зв’язком один-до-багатьох з Doctor, а у якості зовнішнього ключа виступаю атрибут doctor_id та пов’язана зв’язком один-до-багатьох з Patient, а у якості зовнішнього ключа виступаю атрибут patient_id

− Status – містить в собі інформацію про статус візиту. Має атрибут id, який виступає у якості первинного ключа.

− Authentication – містить в собі інформацію про логіни та паролі. Має атрибут id, який виступає у якості первинного ключа.

База даних зберігає у собі дані про лікарів та пацієнтів, історію візитів, лікування, нотатки лікаря, тип робіт та статус візиту.

Нижче схематично представлена база даних системи.

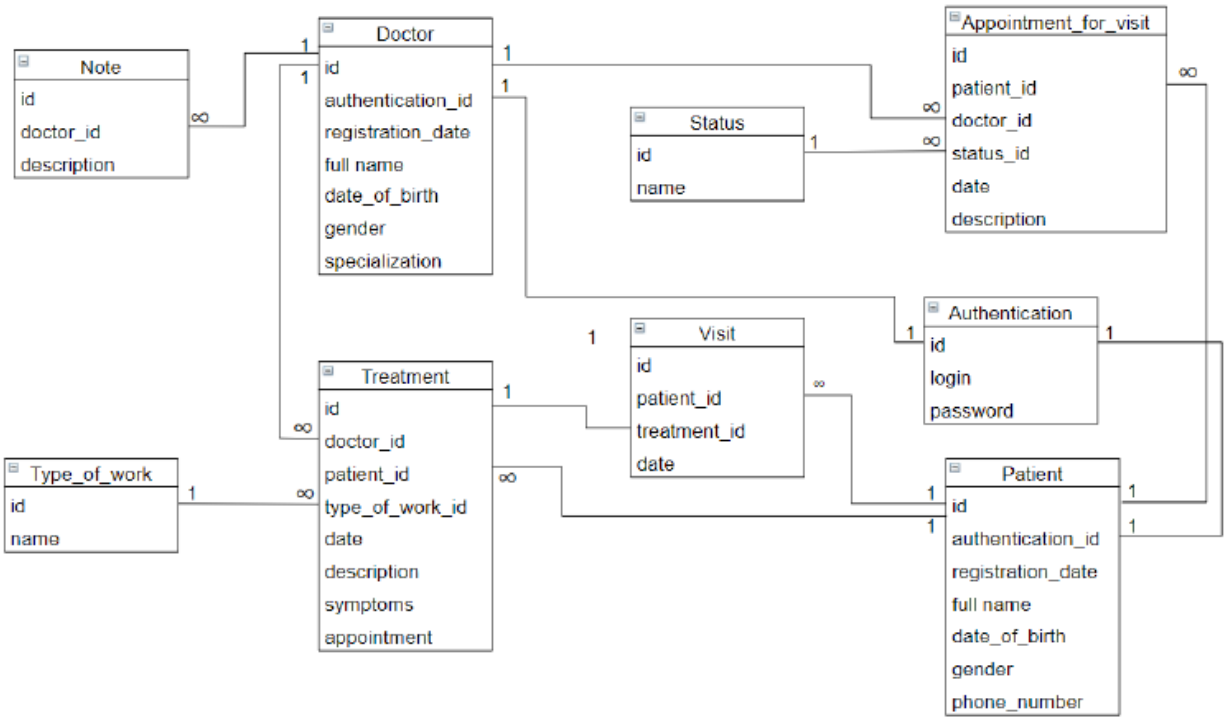

Рисунок 2.18 – Діаграма бази даних

2.7 Висновок до другого розділу

У цьому розділі було проведено проектування системи. Було визначено мету та основні задачі ІС. Отже, мета ІС – створити інформаційну систему, яка допомагає лікарям вести облік пацієнтів та планувати свій графік, а пацієнтам надає можливість завжди мати під рукою свою медичну картку, а також без зайвих клопотів записатися на візит до лікаря.

Для кращого розуміння задач системи, були сформовані функціональні та не функціональні вимоги, а також побудовані діаграми такі, як: діаграма прецедентів, діаграма комунікацій та розгортання системи.

Також були розроблені мокапи графічного інтерфейсу для додатків та побудована діаграма переходів.

Була спроектована та представлена база даних та розглянуті зв’язки між таблицями. Також були описані технології, які будуть використовуватися в системі.

3 РЕАЛІЗАЦІЯ ІНФОРМАЦІЙНОЇ СИСТЕМИ «ПРИВАТНИЙ КАБІНЕТ ЛІКАРЯ СТОМАТОЛОГА»

3.1 Уявлення про структуру проекту ІС

Розглянемо структуру проекту, яка складається з android додатку, desktop додатку та серверу. Для кожної частини розглянемо структуру проекту.

Спочатку розглянемо структуру desktop додатку на рисунку 3.1.

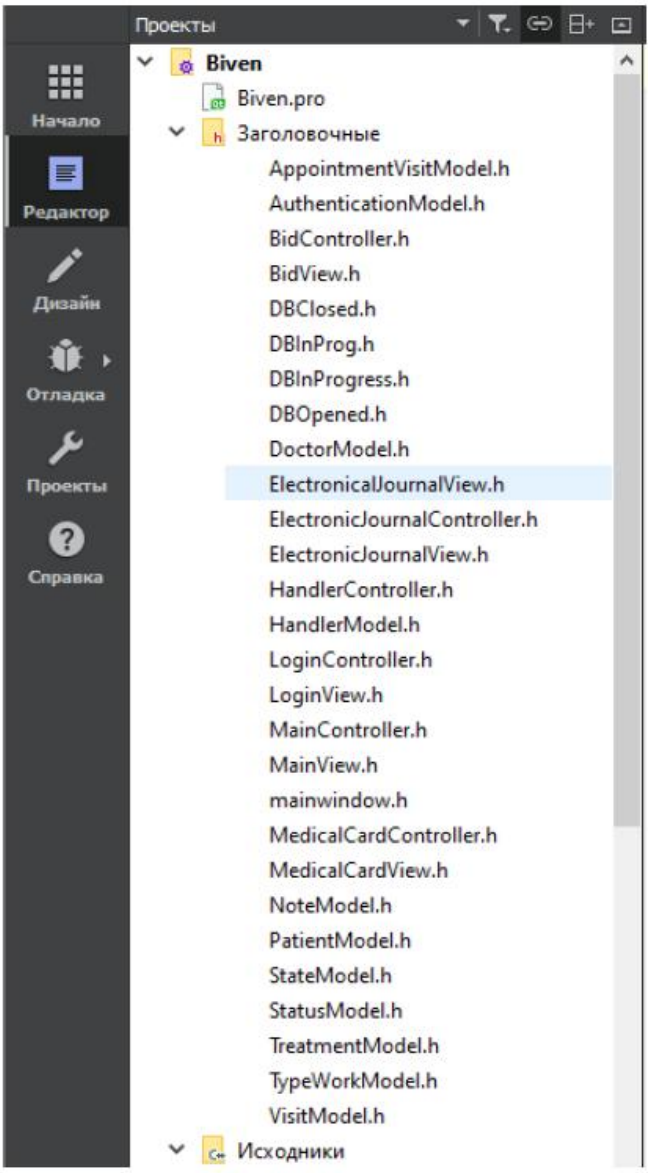

Рисунок 3.1 – Структура desktop частина

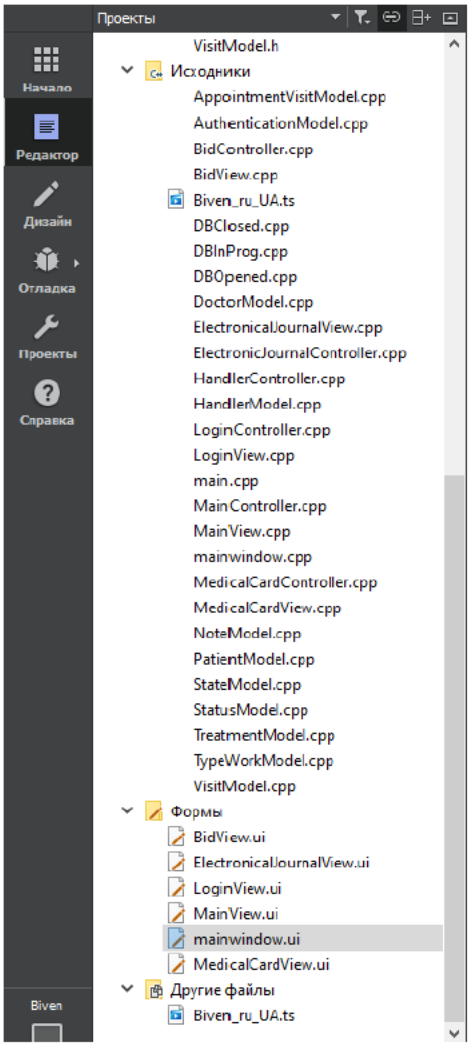

Рисунок 3.2 – Структура desktop частина (продовження)

Desktop частина складається з таких файлів, як:

– заголовні файли. За допомогою цих файлів підключаються до програми типи даних, структури прототипи функції, перелікові типи, які використовуються у другому модулі;

– вихідні файли. Ці файли загалом включають заголовні файли та реалізують прототипи функцій, структури і тд які визначені у заголовних файлах;

– форми. В них знаходяться файли які відповідають за графічний інтерфейс;

– в інший файлах знаходяться всі можливі налаштування, у даному випадку це файл який відповідає за мову проекту.

Тепер розглянемо структуру android частини, яка зображена на рисунку 3.3.

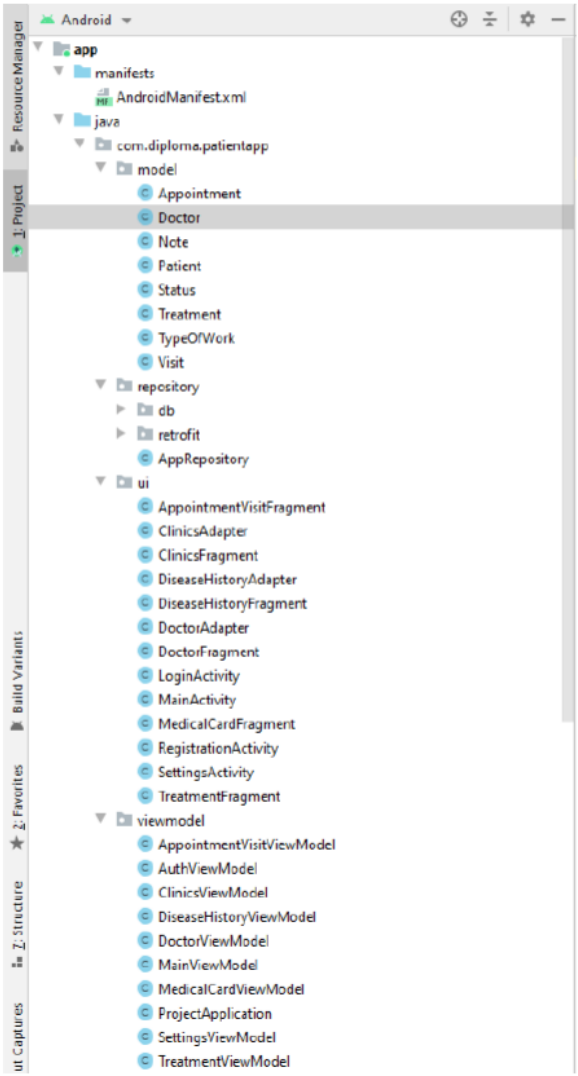

Рисунок 3.3 – Структура android частини

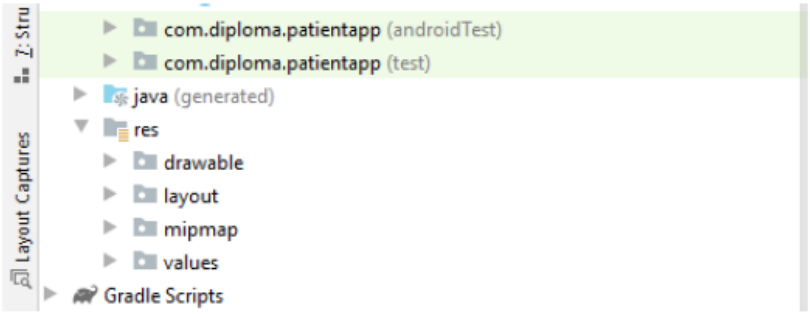

Рисунок 3.4 – Структура android частини (продовження)

В Android частина складається з таких файлів:

− файл manifest це необхідний файл в любому проекті. Він визначає глобальні значення для проекту, в ньому описується, що знаходиться всередині додатку, там визначається як всі елементи взаємодіють з андроїд;

− директорія app містить файли, які відносяться до окремого модулю проекту;

− директорія java містить вихідний код java-класів модуля;

− директорія res містить набір так званих ресурсів (resources), які не є вихідним кодом, але включаються в ваш проект. До ресурсів, наприклад, відносяться зображення, текстові написи, аудіо та відео файли і так далі. Тема ресурсів є однією з ключових, тому ми детально будемо розбирати її пізніше;

− модуль – складова автономна частина проекту, в будь-якому проекті повинен бути, як мінімум, один модуль. При створенні проекту, за замовчуванням, створюється модуль з ім'ям app. Вміст цієї директорії ми розглянемо трохи пізніше;

− набір файлів, які відносяться до збірки проекту за допомогою збирача Gradle. З деякими файлами будемо взаємодіяти походу проекту. Серед всіх зазначених файлів, може цікавити файл build.gradle, який містить інструкції для складання проекту в цілому;

− директорія .idea містить файли, специфічні для середовища розробки Android Studio;

− директорія .gradle містить виконувані файли для збирача Gradle;

− директорія gradle містить скрипт Gradle Wrapper для скачування і установки потрібної версії Gradle.

Перейдемо до розгляду структури серверу на рисунку 3.5.

Нижче приводиться короткий опис файлів:

− manage.py – утиліта, яка дозволяє запускати сервер додатків, робити міграції базі данних, форматувати її і багато іншого;

– appserver.py – модуль, який визначає скрипт для запуску відладочного сервера;

– manage.py – утиліта, яка дозволяє запускати сервер додатків, робити міграції базі данніх, форматувати її і багато іншого;

– migrations – містить міграції бази даних;

– venv – ця директорія представляє віртуальне середовище, у якому встановлені усі необхідні бібліотеки;

– serverapi – python пакет, що складається з модулів, які реалізують сервер;

– __init__.py – перетворює звичайну папку в python пакет, має суто службове застосування;

– api.py – модуль в якому визначені контролери сервера.

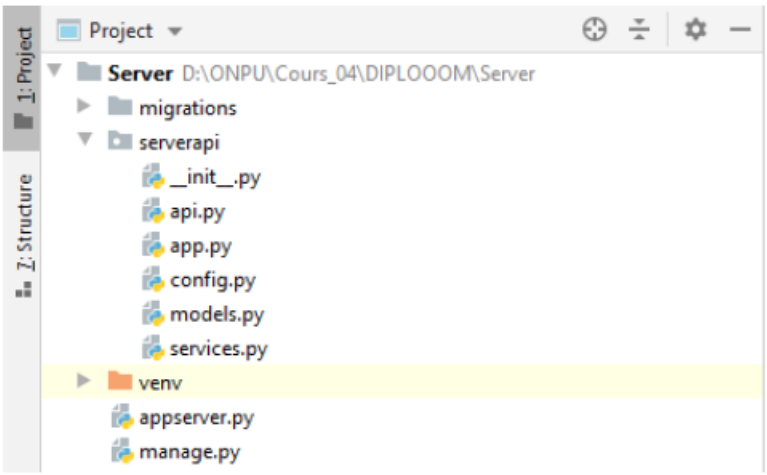

Рисунок 3.5 – Структура серверу на flask

3.2 Уявлення про класи ІС

Розглянемо спроектовані діаграми класів та взаємодію між ними, а також патерни які використовувалися у даній системі. Система складається з 3 частин: серверу, desktop та android додатків.

Для початку розглянемо серверну частину, де використовується такий патерн, як Decorator. Патерн Decorator дозволяє динамічно додавати об’єкту обов’язки.

Перейдемо до діаграми класів серверу на рисунку 3.6.

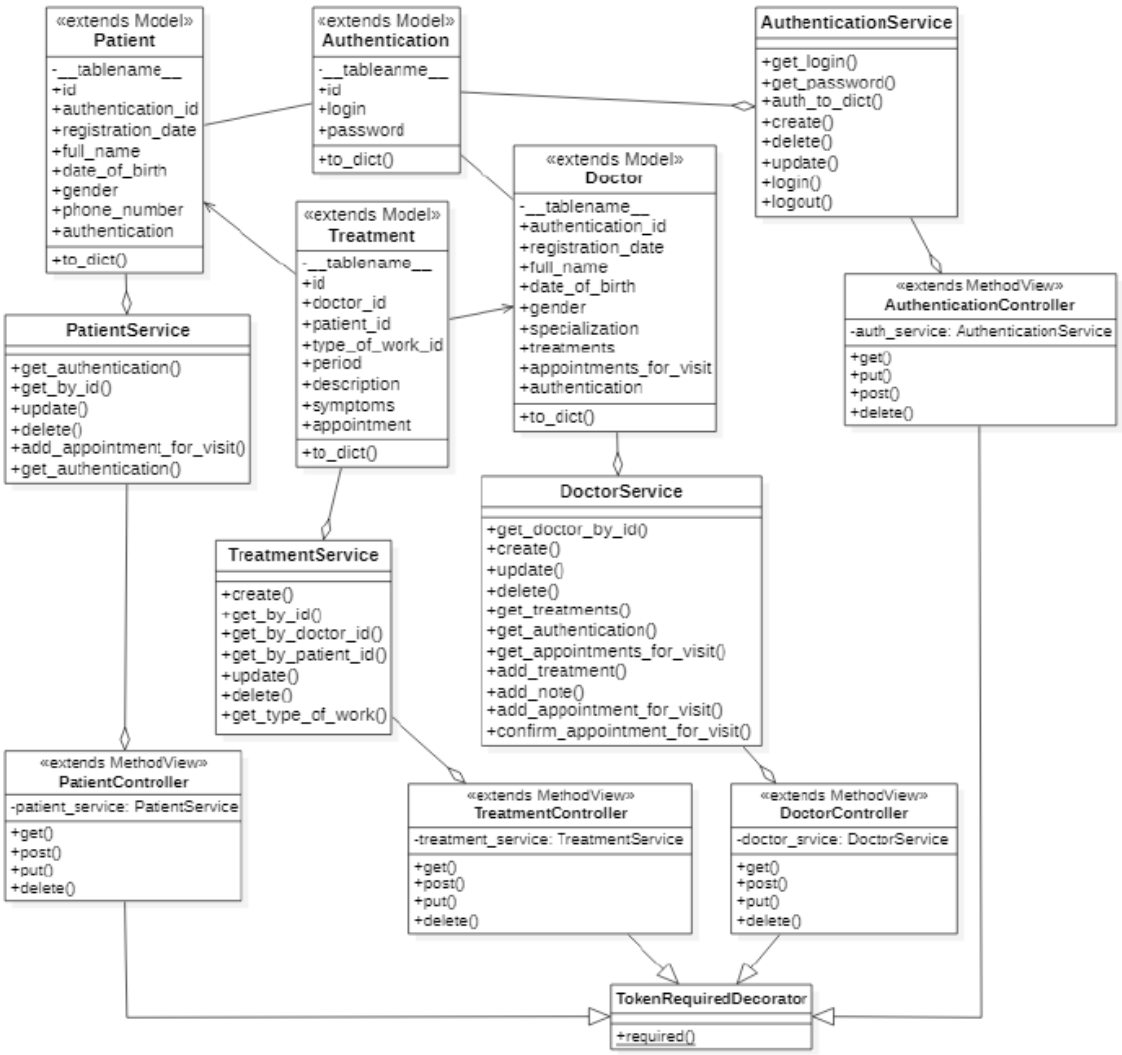

Рисунок 3.6 – Діаграма класів для серверу

Класи PatientService DoctorService AuthenticationService TreatmentService – це класи, відповідають реалізують бізнес логіку.

Класи DoctorController, TreatmentController, AuthenticationController, PatientController це класи які відповідають за обробку HTTPS запитів, займаються валідацією даних, а також повертають дані у форматі JSON.

Клас TokenRequiredDecorator це клас, що реалізує шаблон декоратор, використовуючи особливості Python. Клас реалізує метод required, який перевіряє наявність та правильність токену (іншими словами – перевіряє авторизацію користувача) кожний раз, коли викликаються методи get, post, put та delete.

У desktop частині використовується такі патерни як State та Singleton. Патерн State дозволяє легко однотипно працювати з базою даних, тобто не слідкувати за станом з'єднання з локальною базою даних.

Патерн Singleton гарантує, що у класа є тільки один екземпляр і надає до нього глобальну точку доступу. У випадку даної системи це необхідно для двох класів таких як, HandlerModel та HandlerController.

Тепер перейдемо до діаграми класів у android додатку, на якій будуть зображені базові класи, зображена на рисунку 3.7.

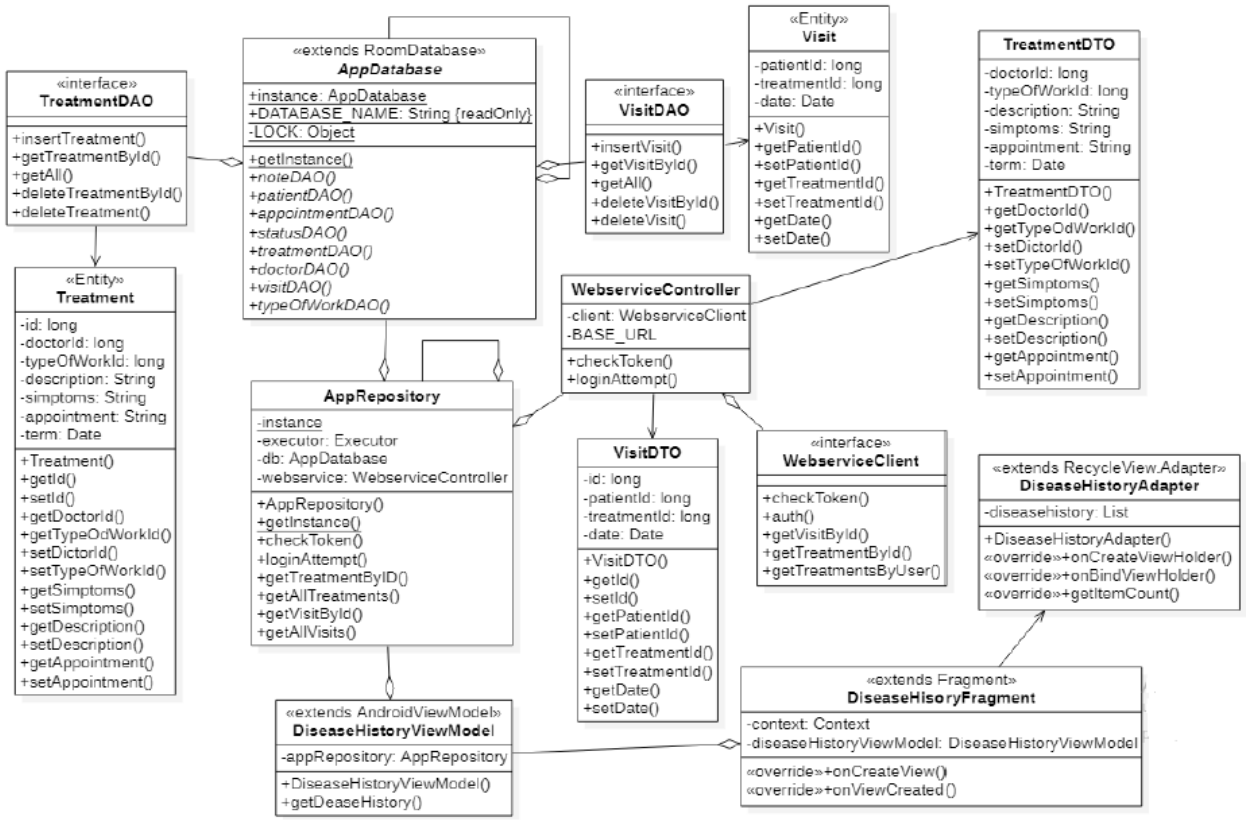

Рисунок 3.7 – Діаграма класів android додатку

На даній діаграмі зображені основні класи. Розглянемо ці класи нижче. Такі класи, як Treatment та Visit містять інформацію про певну сутність.

Класи TreatmentDAO та VisitDAO представляють інтерфейс взаємодії з локальною базою даних. Клас AppDatabase взаємодіє з DAO та встановлює з'єднання з базою даних.

Класи TreatmentDTO та VisitDTO, записують дані, коли приходить відповідь від серверу.

Клас DiseaseHistoryViewModel реалізує бізнес логіку та має зв'язок з AppRepository та запитує у нього дані.

Клас DiseaseHistoryFragment – відображає інтерфейс та реалізує логіку представлення.

Клас DiseaseHistoryAdapter – виступає посередником між даними і віджетом списку, в даному випадку RecyclerView. Тобто, RecyclerView реалізує приховані від користувача технічні аспекти, такі як прокрутка, отрисовка списку і тд., А адаптер відповідає за дані, які потрібно отрисовать, і їх уявлення.

Клас WebserviceController – запитує дані запитує або відправляє дані на сервер.

Клас AppRepository зберігає дані у локальні базі даних, після запиту у WebserviceController даних, та потім відає їх ViewModel.

Клас WebserviceClient – відповідає за маршрутизацію даних до серверу.

ВИСНОВКИ

В процесі написання кваліфікаційної роботи була, розроблена інформаційна система, «Приватний кабінет лікаря стоматолога». Дана система складається з 2 додатків, android та desktop, а також серверної системи. Цей продукт допоможе лікарям керувати своєю приватною практикою, та облегшити облік пацієнтів.

У першому розділі було сформульована постановка задачі, а також було проведене порівняння аналогів, завдяки чому було виявлено недоліки та переваги системи, на основі чого був зроблений функціонал даної системи. Для реалізації даної системи були обрані найбільш доцільні технології. Для desktop версії був обраний QT Creatoror та мова програмування C++. Для android додатку була обрана Android Studio та мова програмування Java. Для серверної частини був обраний фреймворку Flask та мова програмування Python.

У другому розділі було проведено проектування системи. Було визначено мету, основні задачі та головний функціонал. Для кращого розуміння системи, були сформовані функціональні та не функціональні вимоги, а також побудовано безліч діаграм. Потім були розроблені мокапи графічного інтерфейсу для додатків та побудована діаграма переходів. Також була спроектована база даних для даної системи.

У третьому розділі була розглянута структура проекту android додатку, desktop додатку та серверу. Для кращого розуміння були розроблені діаграми основних класів для кожної частини системи.

Таким чином, мета та усі поставлені до даної роботи завдання були успішно виконані.

ПЕРЕЛІК ДЖЕРЕЛ ПОСИЛАННЯ

1. QStoma – Офіційний сайт. URL: https://qstoma.com/ (дата звернення 16.04.2022).
2. IDENT – Офіційний сайт. URL: https://dent-it.ru/ (дата звернення 16.04.2022).
3. 1С:Медицина – Офіційний сайт. URL: https://solutions.1c.ru/catalog/stoma/ (дата звернення 16.04.2022).
4. Реферат з інформаційних технологій «C++»: реферат. URL: https://bit.ly/2SrteNj (дата звернення 18.04.2022).
5. Мова програмування C #: короткий: стаття. URL: https://bit.ly/3ueLUwJ (дата звернення 18.04.2022).
6. C # або C ++: яку мову вивчити?: стаття. URL: https://bit.ly/3wycD94 (дата звернення 18.04.2022).
7. Переваги мови програмування Java: стаття. URL: https://bit.ly/3vjBDkt (дата звернення 19.04.2022).
8. Навчальне керівництво і порівняння нової мови Android-розробки Kotlin з Java : стаття. URL: https://bit.ly/2RMppSP (дата звернення 19.04.2022).
9. Порівняння Python і PHP: в чому різниця?: стаття. URL: https://bit.ly/3fJWtCT (дата звернення 20.04.2022).
10. Короткий огляд кроссплатформенного фреймворка Qt: стаття. URL: https://bit.ly/34gH6N2 (дата звернення 20.04.2022).
11. Чим Джанго краще/гірше Фласк?: стаття. URL: https://bit.ly/3yEVFYq (дата звернення 20.04.2022).
12. PostgreSQL – Офіційний сайт. URL: https://postgrespro.ru/ (дата звернення 21.04.2022).

13. SQLite vs MySQL vs PostgreSQL: A Comparison Of Relational Database Management Systems: стаття. URL: https://do.co/3hR160x (дата звернення 22.04.2022).
14. MySQL – Офіційний сайт. URL: https://www.mysql.com/ (дата звернення 22.04.2022).
15. SQLite – Офіційний сайт. URL: https://www.sqlite.org/index.html (дата звернення 22.04.2022).

ДОДАТОК А
МОКАПИ ГРАФІЧНОГО ІНТЕРФЕЙСУ

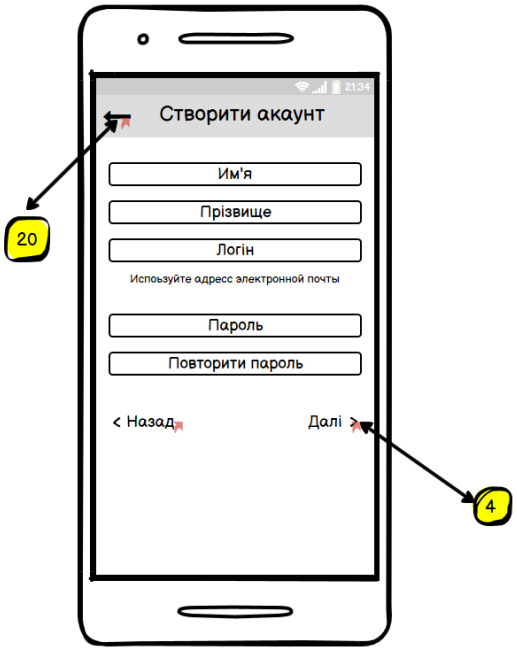

Рисунок А.1 – Екран реєстрації

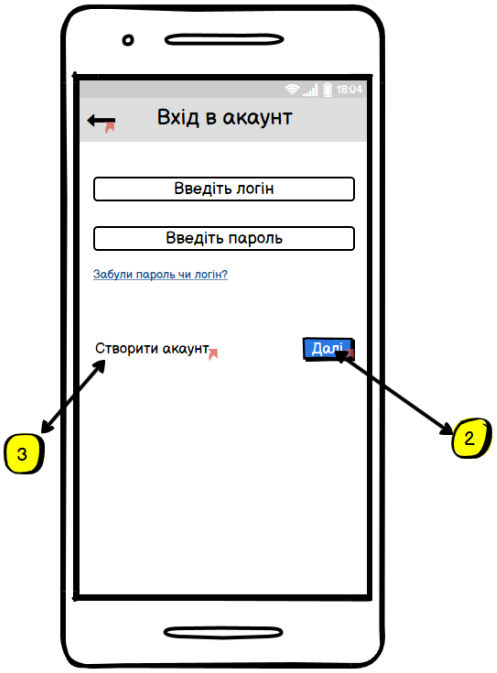

Рисунок А.2 – Екран авторизації

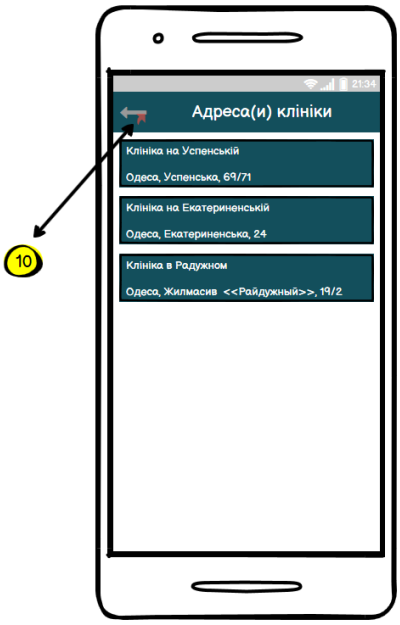

Рисунок А.3 – Екран «Андреса(и) клініки»

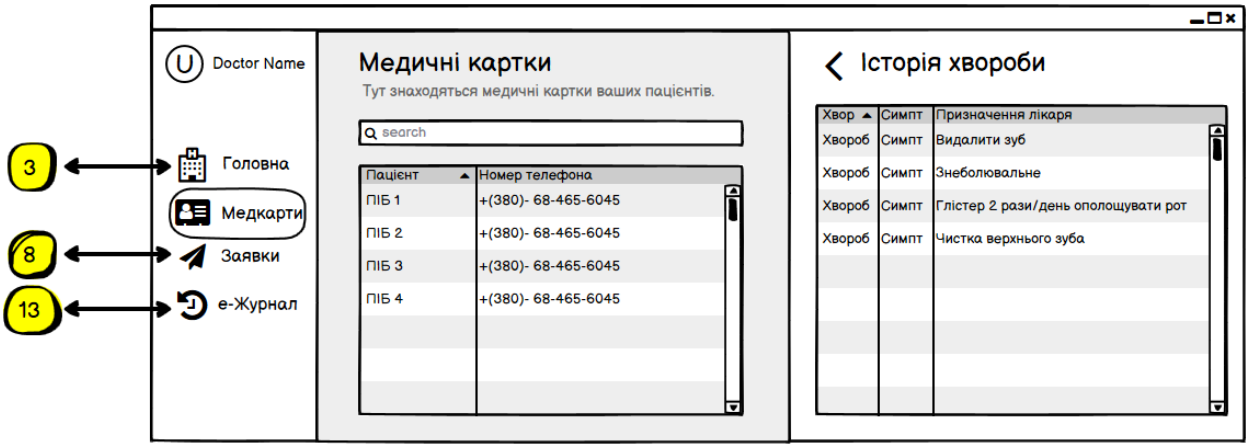

Рисунок А.4 – Екран «Медкарти», історія хвороби

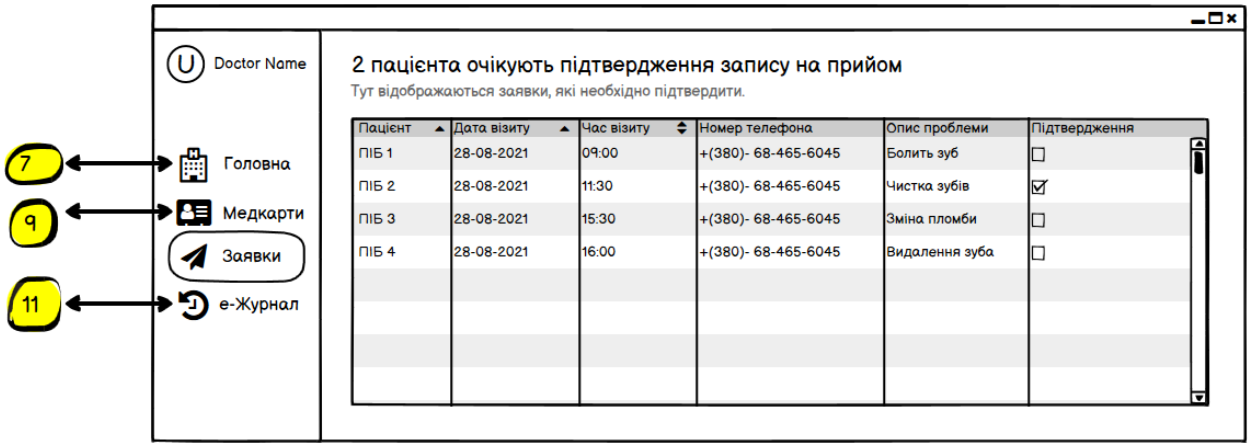

Рисунок А.5 – Екран «Заявки»

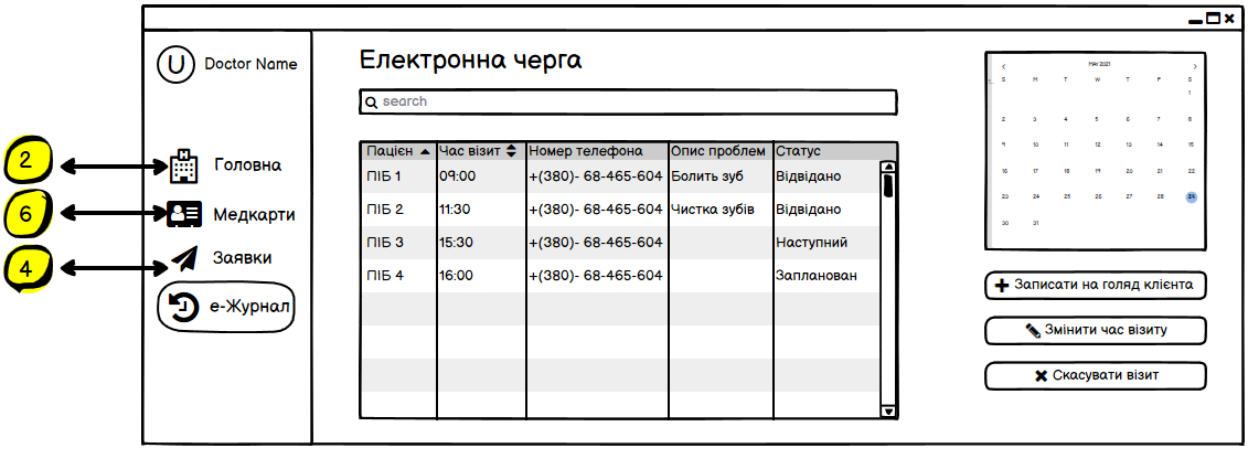

Рисунок А.6 – Екран «е-Журнал»

ДОДАТОК Б

ДІАГРАМА ПОСЛІДОВНОСТІ

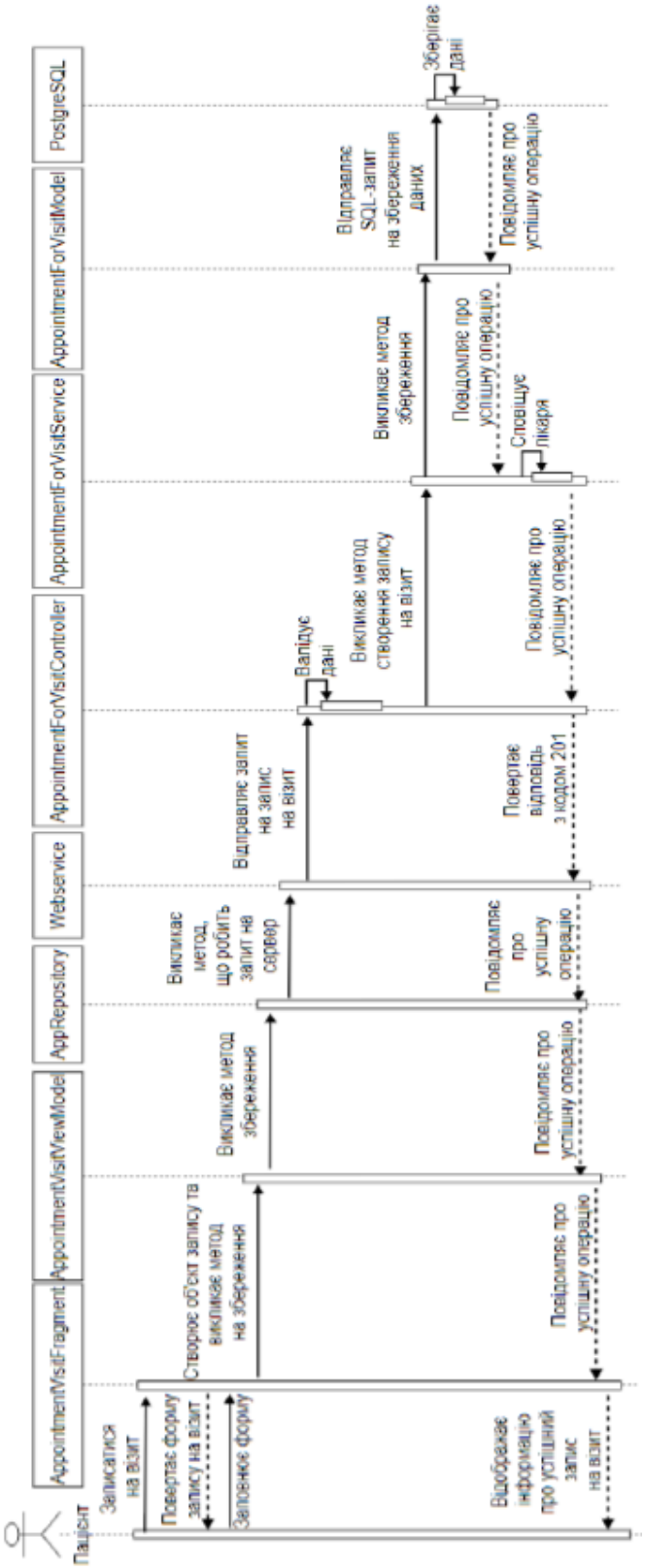

Рисунок Б.1 – Діаграма послідовності для запису на візит до лікаря

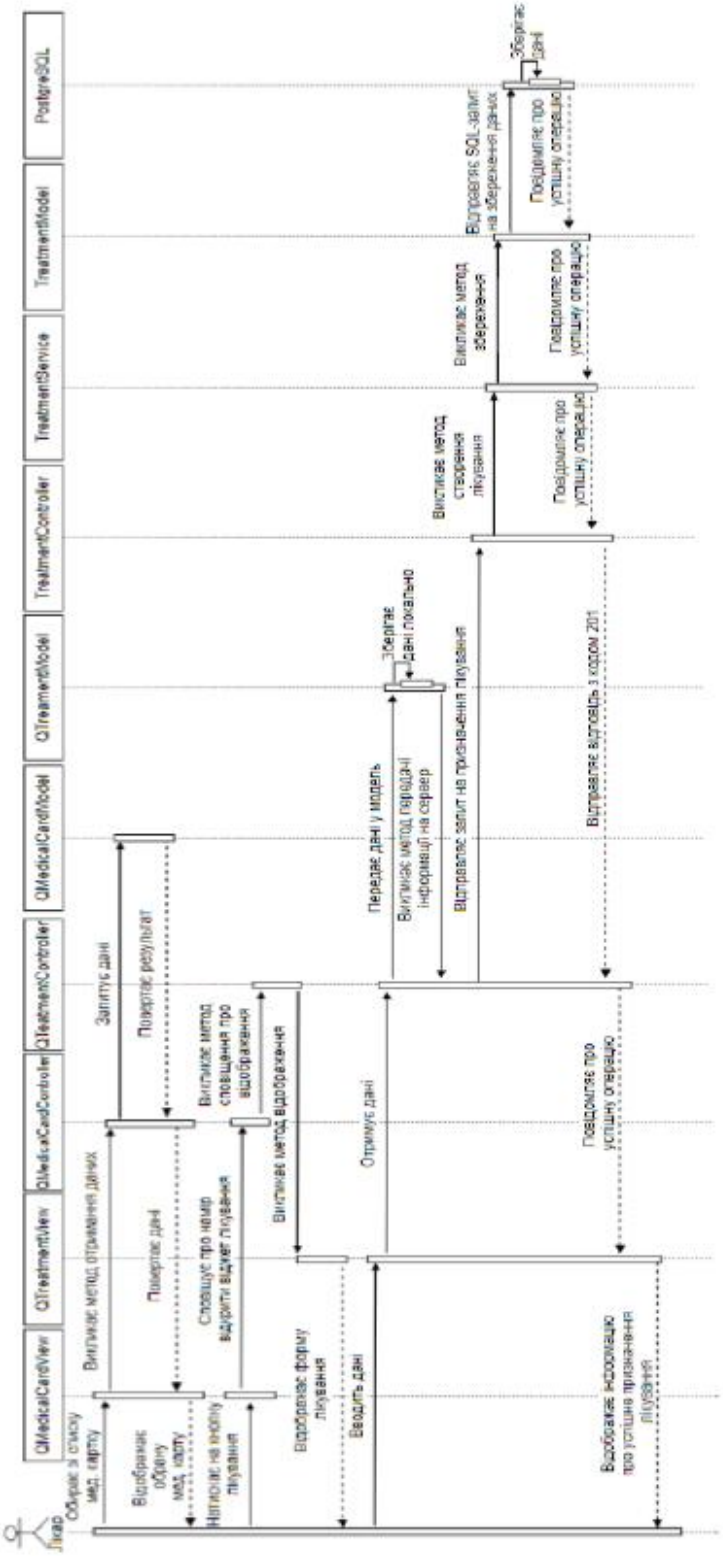

Рисунок Б.2 – Діаграма послідовності для призначення лікування пацієнту

ДОДАТОК В

ДІАГРАМА КЛАСІВ

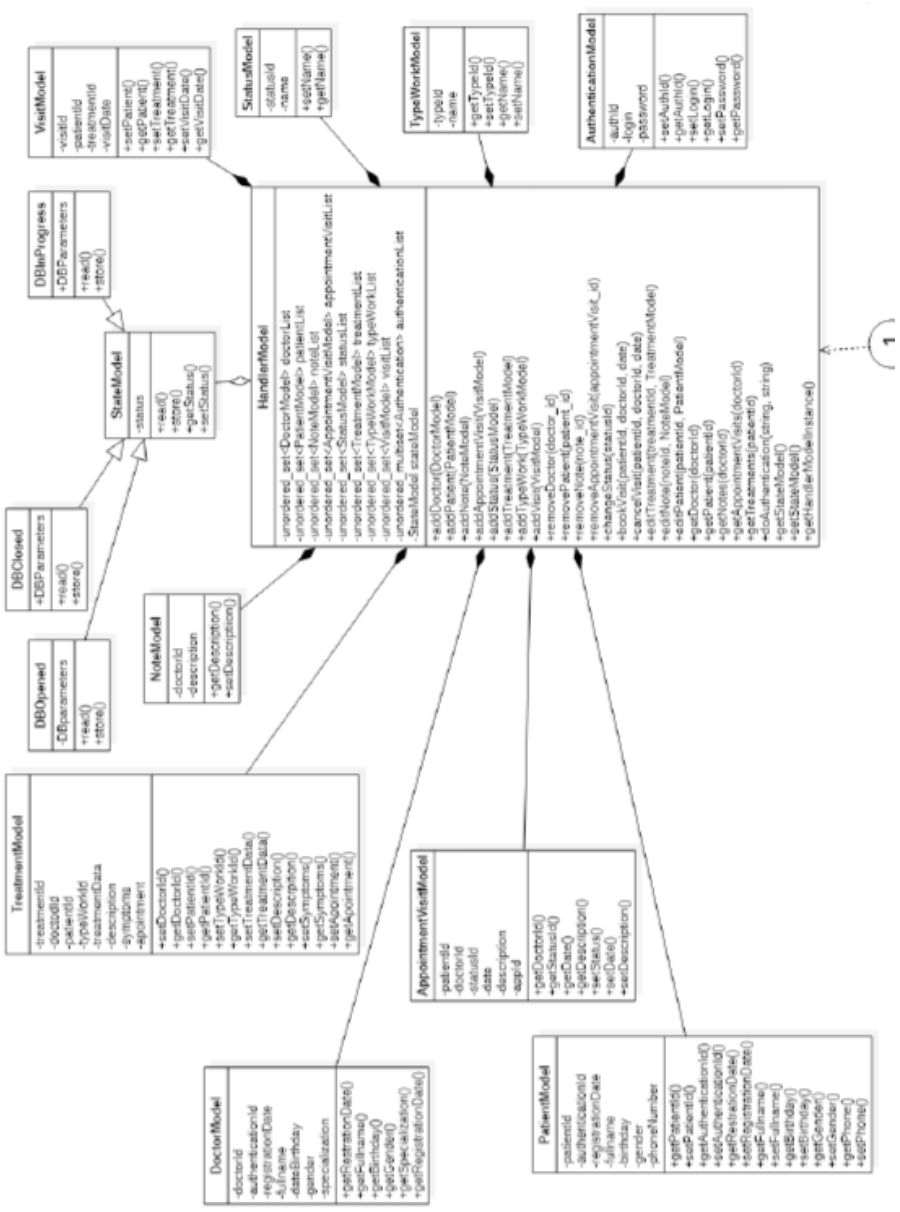

Рисунок В.1 – Діаграма класів для desktop додатку

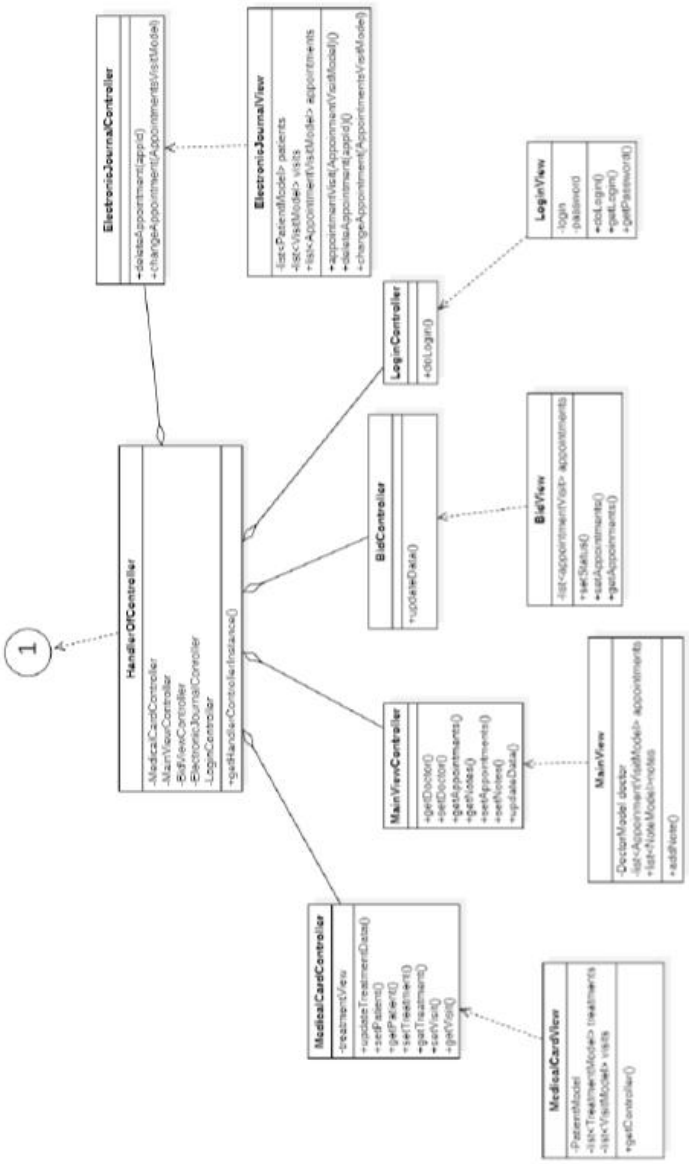

Рисунок В.2 – Діаграма класів для desktop додатку (продовження)

ДОДАТОК Г

СКРИНШОТИ ДОДАТКУ

Рисунок Г.1 – Скриншот екрану «Головний екран»

Рисунок Г.2 – Скриншот екрану «Запис на візит»

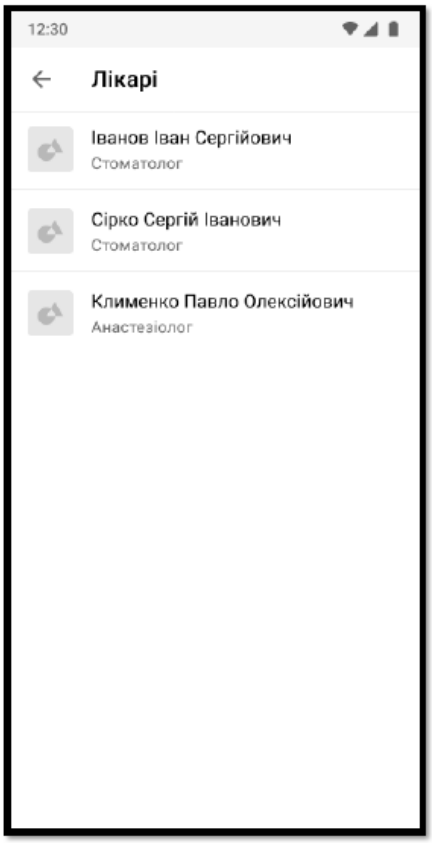

Рисунок Г.3 – Скриншот екрану «Лікарі»

Рисунок Г.4 – Скриншот екрану «Адреси клінік»

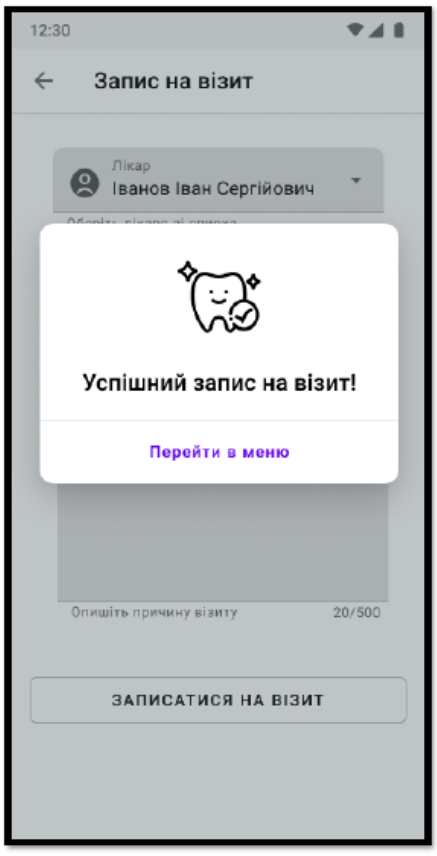

Рисунок Г.5 – Скриншот екрану «Успішний запис на візит»

ДОДАТОК Д

ПРОГРАМНИЙ КОД

Файл «requester.h»

```
#ifndef REQUESTER_H
#define REQUESTER_H
#include <QObject>
#include <QBuffer>
#include <QNetworkReply>
#include <QNetworkRequest>
#include <QJsonObject>
#include <QJsonArray>
#include <QJsonDocument>
#include <functional>
class Requester : public QObject
{
Q_OBJECT
public:
typedef std::function<void(const QJsonObject &)> handleFunc;
typedef std::function<void()> finishFunc;
static const QString KEY_QNETWORK_REPLY_ERROR;
static const QString KEY_CONTENT_NOT_FOUND;
enum class Type {
POST,
GET,
PATCH,
DELET
};
explicit Requester(QObject *parent = 0);
void initRequester(const QString& host, int port, QSslConfiguration
*value); 93

void sendRequest(const QString &apiStr,
const handleFunc &funcSuccess,
const handleFunc &funcError,
Type type = Type::GET,
const QVariantMap &data = QVariantMap());
void sendMulishGetRequest(const QString &apiStr,
const handleFunc &funcSuccess,
const handleFunc &funcError,
const finishFunc &funcFinish);
QString getToken() const;
void setToken(const QString &value);
private:
static const QString httpTemplate;
static const QString httpsTemplate;
QString host;
int port;
QString token;
QSslConfiguration *sslConfig;
QString pathTemplate;
QByteArray variantMapToJson(QVariantMap data);
QNetworkRequest createRequest(const QString &apiStr);
QNetworkReply *sendCustomRequest(QNetworkAccessManager *manager,
QNetworkRequest &request,
const QString &type,
const QVariantMap &data); 94

QJsonObject parseReply(QNetworkReply *reply);
bool onFinishRequest(QNetworkReply *reply);
void handleQtNetworkErrors(QNetworkReply *reply, QJsonObject &obj);
```

```
QNetworkAccessManager *manager;
signals:
void networkError();
public slots:
};
#endif // REQUESTER_H
```

Файл «requester.cpp»

```
#include "requester.h"
const QString Requester::httpTemplate = "http://%1:%2/api/%3";
const QString Requester::httpsTemplate = "https://%1:%2/api/%3";
const QString Requester::KEY_QNETWORK_REPLY_ERROR = "QNetworkReplyError";
const QString Requester::KEY_CONTENT_NOT_FOUND = "ContentNotFoundError";
Requester::Requester(QObject *parent) : QObject(parent)
{
manager = new QNetworkAccessManager(this);
}
void Requester::initRequester(const QString &host, int port,
QSslConfiguration *value)
{
this->host = host;
this->port = port; 95

sslConfig = value;
if (sslConfig != nullptr)
pathTemplate = httpsTemplate;
else
pathTemplate = httpTemplate;
}
void Requester::sendRequest(const QString &apiStr,
const handleFunc &funcSuccess,
const handleFunc &funcError,
Requester::Type type,
const QVariantMap &data)
{
QNetworkRequest request = createRequest(apiStr);
QNetworkReply *reply;
switch (type) {
case Type::POST: {
QByteArray postDataByteArray = variantMapToJson(data);
reply = manager->post(request, postDataByteArray);
break;
} case Type::GET: {
reply = manager->get(request);
break;
} case Type::DELET: {
if (data.isEmpty())
reply = manager->deleteResource(request);
else
reply = sendCustomRequest(manager, request, "DELETE", data);
break;
} case Type::PATCH: {
reply = sendCustomRequest(manager, request, "PATCH", data);
break;
} default:
reply = nullptr; 96

Q_ASSERT(false);
}
connect(reply, &QNetworkReply::finished, this,
[this, funcSuccess, funcError, reply]() {
QJsonObject obj = parseReply(reply);
if (onFinishRequest(reply)) {
if (funcSuccess != nullptr)
```

```
funcSuccess(obj);
} else {
if (funcError != nullptr) {
handleQtNetworkErrors(reply, obj);
funcError(obj);
}
}
reply->close();
reply->deleteLater();
} );
}
void Requester::sendMulishGetRequest(const QString &apiStr, //а ничего
что здесь нигде не проверяется func != nullptr?
const handleFunc &funcSuccess,
const handleFunc &funcError,
const finishFunc &funcFinish)
{
QNetworkRequest request = createRequest(apiStr);
// QNetworkReply *reply;
qInfo() << "GET REQUEST " << request.url().toString() << "\n";
auto reply = manager->get(request);
connect(reply, &QNetworkReply::finished, this,
[this, funcSuccess, funcError, funcFinish, reply]() { 97

QJsonObject obj = parseReply(reply);
if (onFinishRequest(reply)) {
if (funcSuccess != nullptr)
funcSuccess(obj);
QString nextPage = obj.value("next").toString();
if (!nextPage.isEmpty()) {
QStringList apiMethodWithPage = nextPage.split("api/");
sendMulishGetRequest(apiMethodWithPage.value(1),
funcSuccess,
funcError,
funcFinish
);
} else {
if (funcFinish != nullptr)
funcFinish();
}
} else {
handleQtNetworkErrors(reply, obj);
if (funcError != nullptr)
funcError(obj);
}
reply->close();
reply->deleteLater();
});
}
QString Requester::getToken() const
{
return token;
}
void Requester::setToken(const QString &value)
{
token = value; 98

}
QByteArray Requester::variantMapToJson(QVariantMap data)
{
QJsonDocument postDataDoc = QJsonDocument::fromVariant(data);
QByteArray postDataByteArray = postDataDoc.toJson();
return postDataByteArray;
```

```
}
QNetworkRequest Requester::createRequest(const QString &apiStr)
{
QNetworkRequest request;
QString url = pathTemplate.arg(host).arg(port).arg(apiStr);
request.setUrl(QUrl(url));
request.setRawHeader("Content-Type","application/json");
if(!token.isEmpty())
request.setRawHeader("Authorization",QString("token
%1").arg(token).toUtf8());
if (sslConfig != nullptr)
request.setSslConfiguration(*sslConfig);
return request;
}
QNetworkReply* Requester::sendCustomRequest(QNetworkAccessManager* manag-
er,
QNetworkRequest &request,
const QString &type,
const QVariantMap &data)
{
request.setRawHeader("HTTP", type.toUtf8());
QByteArray postDataByteArray = variantMapToJson(data);
QBuffer *buff = new QBuffer;
buff->setData(postDataByteArray);
buff->open(QIODevice::ReadOnly);
QNetworkReply* reply = manager->sendCustomRequest(request, type.toUtf8(),
buff); 99

buff->setParent(reply);
return reply;
}
QJsonObject Requester::parseReply(QNetworkReply *reply)
{
QJsonObject jsonObj;
QJsonDocument jsonDoc;
QJsonParseError parseError;
auto replyText = reply->readAll();
jsonDoc = QJsonDocument::fromJson(replyText, &parseError);
if(parseError.error != QJsonParseError::NoError){
qDebug() << replyText;
qWarning() << "Json parse error: " << parseError.errorString();
}else{
if(jsonDoc.isObject())
jsonObj = jsonDoc.object();
else if (jsonDoc.isArray())
jsonObj["non_field_errors"] = jsonDoc.array();
}
return jsonObj;
}
bool Requester::onFinishRequest(QNetworkReply *reply)
{
auto replyError = reply->error();
if (replyError == QNetworkReply::NoError ) {
int code = reply-
>attribute(QNetworkRequest::HttpStatusCodeAttribute).toInt();
if ((code >=200) && (code < 300)) {
return true;
}
}
return false;
} 100
```

```
void Requester::handleQtNetworkErrors(QNetworkReply *reply, QJsonObject
&obj)
{
auto replyError = reply->error();
if (!(replyError == QNetworkReply::NoError ||
replyError == QNetworkReply::ContentNotFoundError ||
replyError == QNetworkReply::ContentAccessDenied ||
replyError == QNetworkReply::ProtocolInvalidOperationError
) ) {
qDebug() << reply->error();
obj[KEY_QNETWORK_REPLY_ERROR] = reply->errorString();
} else if (replyError == QNetworkReply::ContentNotFoundError)
obj[KEY_CONTENT_NOT_FOUND] = reply->errorString();
}
```