НАЦІОНАЛЬНИЙ ТЕХНІЧНИЙ УНІВЕРСИТЕТ УКРАЇНИ
«КИЇВСЬКИЙ ПОЛІТЕХНІЧНИЙ ІНСТИТУТ
імені ІГОРЯ СІКОРСЬКОГО»

Факультет інформатики та обчислювальної техніки

Кафедра автоматики та управління в технічних системах

До захисту допущено:

Завідувач кафедри

_____ Олександр РОЛІК

«___»_____20__ р.

Дипломний проєкт

на здобуття ступеня бакалавра
за освітньо-професійною програмою «Програмне забезпечення інформаційно-комунікаційних систем»
спеціальності 121 «Інженерія програмного забезпечення»
на тему: «Інформаційна система “Електронна медична картка”»

Виконав:
студент IV курсу, групи ІТ-61
Смольков Олександр Олександрович _____

Керівник:
доцент каф. АУТС, к.т.н., доцент
Писаренко Андрій Володимирович _____

Рецензент:
доцент каф. ТК, к.т.н., доцент
Ткач Михайло Мартинович _____

Засвідчую, що у цьому дипломному проєкті немає запозичень з праць інших авторів без відповідних посилань.
Студент (-ка) _____

Київ – 2020 року

Національний технічний університет України
«Київський політехнічний інститут імені Ігоря Сікорського»
Факультет інформатики та обчислювальної техніки
Кафедра автоматики та управління в технічних системах

Рівень вищої освіти – перший (бакалаврський)

Спеціальність – 121 «Інженерія програмного забезпечення»

Освітньо-професійна програма «Програмне забезпечення інформаційно-комунікаційних систем»

ЗАТВЕРДЖУЮ
Завідувач кафедри
_____ Олександр РОЛІК
«___»_____20__ р.

ЗАВДАННЯ

на дипломний проєкт студенту
Смолькову Олександру Олександровичу

1.Тема проєкту «Інформаційна система “Електронна медична картка”», керівник проєкту Писаренко Андрій Володимирович доцент каф. АУТС к.т.н., доцент, затверджені наказом по університету від «07» травня 2020 р. №1081-с.

2. Термін подання студентом проєкту: 09 червня 2020 р.

3. Вихідні дані до проєкту: стандарт МОЗ 025/о, стандарт NFC (ISO14443), користувацький інфтерфейс Android застосунку, протоколи комунікації відкритих платформ, протоколи передачі гіпертексту.

4. Зміст пояснювальної записки: вступ, аналіз предметної області, методи та технології, проєктування програмного продукту, реалізація системи.

5. Перелік графічного матеріалу: 5 креслень, 36 зображень, 7 таблиць.

6. Дата видачі завдання: 24 лютого 2020 р.

Календарний план

№ з/п	Назва етапів виконання дипломного проєкту	Термін виконання етапів проєкту	Примітка
1	Отримати завдання на ДП	24 лютого 2020 р.	
2	Розділ ДП ВСТУП, Літературний огляд.	30 березня 2020 р.	
3	Розділ ДП Основна частина (Теоретична).	15 квітня 2020 р.	
4	Розділ ДП Основна частина (Аналітична).	30 квітня 2020 р.	
5	Розділ ДП Основна частина (Практична). ВИСНОВКИ. ДОДАТКИ	25 травня 2020 р.	
6	Підготовка супровідної документації	1 червня 2020 р.	
7	Подання роботи керівнику ДП	9 червня 2020 р.	
8	Перевірка роботи на плагіат (UNICHEK)	15 червня 2020р.	
9	Подання ДП рецензенту. Отримання рецензії.	15 червня 2020р	
10	Подання в електронному вигляді ДП та анотації до неї	15 червня 2020р	
11	Захист ДП	16 червня 2020 р.	

Студент Олександр СМОЛЬКОВ

Керівник проєкту Андрій ПИСАРЕНКО

АНОТАЦІЯ

Смольков О.О. Інформаційна система «Електронна медична картка». КПІ ім. Ігоря Сікорського, Київ, 2020.

Структура та обсяг проєкту: пояснювальна записка складається із вступу, чотирьох розділів, висновків, списку використаної літератури із 26 джерел та двох додатків. Загальний обсяг дипломного проєкту складає: 72 сторінок, основного тексту (без додатків) – 70, ілюстрацій – 36, таблиць – 7.

Метою бакалаврського проєкту є підвищення ефективності процесів надання медичної допомоги за рахунок розроблення інформаційної системи «Електронна медична картка». Для виконання роботи було використано технології для розроблення Restful веб-сервісів та Android застосунків. В результаті було розроблено систему, яка дозволяє вести облік стану здоров'я пацієнтів незалежно від платформи клієнтського застосунку.

Дипломний проєкт виконаний на замовлення фірми ПрАТ «СК «Альфа Страхування», результати впроваджені в роботу (акт впровадження від «07» червня 2020р.)

Ключові слова: електронні медичні записи, система обліку пацієнтів, Android, REST.API, NFC.

SUMMARY

Smolkov O. O. Information system "Electronic medical card". Igor Sikorsky KPI, Kyiv, 2020.

Structure and scope of project: the explanatory note consists of an introduction, four chapters, conclusions, a list of references from 26 sources and two appendices. The total volume of the thesis is: 72 pages, the main text (without appendices) - 70, illustrations - 36, tables - 7.

The aim of the thesis is to create a digital analogue of the system of medical records of patients, as well as a single online database of patients, to store basic data about their health. Technologies for developing Restful web services and Android applications were used to perform the work. As a result, a system has been developed that allows patients to keep track of their health regardless of the client application platform.

The thesis was commissioned by PJSC "Insurance Company" Alpha Insurance", the results are implemented (act of implementation of" 7 "June 2020).

Keywords: electronic medical records, patient registration system, Android, REST.API, NFC.

Поз.	Формат	Позначення	Найменування	Кількість аркушів	№ прим.	Примітки
1			Документація загальна			
2						
3			Знову зроблена			
4						
5	А4	ІТ61.220БАК.004 ПЗ	Інформаційна система	72		
6			«Електронна медична картка».			
7			Пояснювальна записка			
8	А3	ІТ61.220БАК.004 Д1	Інформаційна система	1		
9			«Електронна медична картка».			
10			Діаграма варіантів використання.			
11	А3	ІТ61.220БАК.004 Д2	Інформаційна система	1		
12			«Електронна медична картка».			
13			Діаграма послідовностей.			
14	А3	ІТ61.220БАК.004 Д3	Інформаційна система	1		
15			«Електронна медична картка».			
16			Діаграма станів.			
17	А3	ІТ61.220БАК.004 Д4	Інформаційна система	1		
18			«Електронна медична картка».			
19			Діаграма діяльності.			
20	А3	ІТ61.220БАК.004 Д5	Інформаційна система	1		
21			«Електронна медична картка».			
22			Опис дерева цілей.			
23						
24						
25						

					ІТ61.220БАК.004 ТП	
		Прізвище	Підпис	Дата		
Розроб.		Смольков О. О.			Інформаційна система «Електронна медична картка». Відомість технічного проєкту.	Літ. / Лист 1 / Листів X
Перевірив.		Писаренко А. В.				
Н. кон.						КПІ ім. Ігоря Сікорського кафедра АУТС гр. ІТ-61
Затв.						

Пояснювальна записка
до дипломного проєкту
на тему: «Інформаційна система «Електронна медична картка»»

Київ – 2020 року

ЗМІСТ

ПЕРЕЛІК СКОРОЧЕНЬ, УМОВНИХ ПОЗНАЧЕНЬ, ТЕРМІНІВ

EHR – (від electronic health record) електронні медичні записи.

ЕМК – електронна медична картка.

HL7 – (від Health Level Seven International) міжнародний стандарт рівня здоров'я сім.

OPC – (від Open Platform Communications) комунікація відкритих платформ.

FHIR – (від Fast Healthcare Interoperability Resources) ресурси швидкої взаємодії в сфері охорони здоров'я.

RFID – (від Radio frequency identification) радіочастотна ідентифікація.

NFC – (від Near-Field Communication) комунікація ближнього поля.

POS – (від point of sale) безконтактна точка продажу.

URL – (від Uniform Resource Locator) єдиний вказівник на ресурс.

CMS – (від Content Management System) система керування вмістом.

LINQ – (від Language Integrated Query) запити, інтегровані в мову.

ООП – об'єктно-орієнтоване програмування.

SDK – (від software development kit) набір інструментів розробки.

UI – (від user interface) користувацький інтерфейс.

ОС – операційна система.

IL – (від Intermediate Language) проміжна мова.

API – (від Application Programming Interface) прикладний програмний інтерфейс.

ART – (від Android Runtime) середовище виконання Android застосунків.

MCW – (від Managed Callable Wrappers) обгортки з керованим викликом.

ACW – (від Android Callable Wrappers) Android обгортки, що викликаються.

XAML – (від eXtensible Application Markup Language) декларативна мова розмітки.

REST – (від Representational State Transfer) передача станів представлення.

JSON – (від JavaScript Object Notation) об’єктна нотація JavaScript.

MVC – (від Model-view-controller) модель-представлення-контролер.

IoT – (від Internet of Things) інтернет речей.

IIS – (від Internet Information Services) інтернет-інформаційні сервіси.

IDEF – (від Integrated DEFinition) інтегровані визначення.

ПК – персональний ком’ютер.

UML – (від Unified Modeling Language) уніфікована мова моделювання.

HTTP – (від HyperText Transfer Protocol) протокол передачі гіпертексту.

БД – база даних.

ПП – програмний продукт.

SQL – (від structured query language) мова структурованих запитів.

СУБД – система управління базами даних.

CLR – (від Common Language Runtime) загальномовне виконуюче середовище.

WYSIWYG – (від What You See Is What You Get) те, що бачиш, те і отримуєш.

ВСТУП

Сфера охорони здоров'я є невід'ємною складовою частиною системи функціонування будь-якої держави. З ефективністю її роботи пов'язані як економічна, так і соціальна ніші існування суспільства. На сьогоднішній день в Україні даний аспект державної діяльності не є досконалим. Порівнявши сферу охорони здоров'я нашої країни та більш розвинених країн Заходу, можна побачити великі відмінності, особливо це відчутно у такій частині роботи лікаря, як збір, зберігання та аналіз даних про здоров'я пацієнтів. Сьогодні цей процес реалізований за допомогою медичних карток пацієнтів – зошитів або записників, де рукописно фіксується історія хвороби особи.

Такий підхід є досить ефективним, адже дозволяє відслідковувати стан здоров'я пацієнта і, як наслідок, надавати ефективне лікування. Проте такий паперовий носій інформації є досить ненадійним, адже його можна загубити або забути. Тому, спираючись на досвід провідних країн світу, запропоновано реалізувати та впровадити у сферу охорони здоров'я країни систему автоматизації обліку медичних даних пацієнтів. Така система спрямована для використання, насамперед, працівниками медичної сфери у лікарнях, а особливо в бригадах швидкої допомоги.

Метою бакалаврського проєкту є підвищення ефективності процесів надання медичної допомоги за рахунок розроблення інформаційної системи «Електронна медична картка». Основною функціональною задачею системи є ведення обліку стану здоров'я пацієнтів, надання зручного і зрозумілого інтерфейсу лікарям, для заповнення даних та доступність інформації про пацієнта у будь який момент. Останній аспект може бути розширено у майбутньому з використанням великої кількості

технологій. Для прикладу у даній роботі використано технологію зв'язку на невеликих відстанях (у подальшому NFC).

Шляхами розвитку розробленої системи є створення застосунку для платформи Windows або веб-застосунку. Така система дозволить отримувати статистику, проводити дослідження і навіть реалізовувати діагностування за допомогою штучного інтелекту.

Зважаючи на розвиток законодавства України, можна побачити все більше процедур, спрямованих на переведення процесів надання державних послуг у цифровий формат, тобто диджиталізацію. Ще 18 вересня 2019 року, Кабінет Міністрів України затвердив постанову щодо діяльності Міністерства цифрової трансформації. Згідно з прийнятим Положенням, новостворене Міністерство відповідатиме за формування та реалізацію державної політики у сфері цифровізації, відкритих даних, національних електронних інформаційних ресурсів та інтероперабельності, впровадження електронних та адміністративних послуг, електронних довірчих послуг тощо [1]. Отже, система автоматизації обліку стану здоров’я пацієнтів, є одним із векторів розвитку країни, тому обрана тема є актуальною.

На практиці така система дозволить зменшити час, який лікар витрачає на пошук інформації у медичній картці, створення записів до неї, діагностування. Також, для членів бригади швидкої допомоги вона дозволить швидко надати медичну допомогу потерпілому і передати попередній діагноз у лікарню. Результати роботи системи можна оцінити провівши якісне тестування та порівнявши її з впровадженим аналогом. З цими результатами можна ознайомитись у третьому розділі.

Враховуючи мету проєкту та практичні цілі, були сформовані та розв’язані наступні завдання:

– аналіз предметної області;

– огляд існуючих рішень та виділення основних аспектів розробки;

– вибір технологій та інструментів реалізації системи;

– розроблення інформаційної системи.

Структура та обсяг проєкту: пояснювальна записка складається із вступу, чотирьох розділів, висновків, списку використаної літератури із 26 джерел та двох додатків. Загальний обсяг дипломного проєкту складає: 72 сторінок, основного тексту (без додатків) – 70, ілюстрацій – 36, таблиць – 7.

1 АНАЛІЗ ПРЕДМЕТНОЇ ОБЛАСТІ

1.1 Актуальність проблеми

Однією з головних сфер життя людини у суспільстві, котрі врегульовуються державою є система охорони здоров’я. Охорона здоров'я – галузь діяльності держави, метою якої є організація та забезпечення доступного медичного обслуговування населення [2]. Ефективною такою систему можна назвати, тоді коли вона відповідає наступним вимогам:

– доступність – будь-яка людина, незалежно від статусу і положення у соціумі повинна мати змогу отримати медичну допомогу;

– якість – послуги, що надаються, повинні відповідати міжнародним стандартам;

– швидкодія – медична допомога повинна надаватись у найкоротший час, адже у багатьох випадках людям необхідна невідкладна медична допомога.

Рисунок 1.1 – Основні складові ефективної системи охорони здоров’я

У даній роботі пропонується використати систему, що значною мірою дозволить вплинути на покращення таких параметрів функціонування

системи охорони здоров’я як якість та швидкодія. Інформаційна система електронних медичних карток, дозволить зробити інформацію більш доступною для лікарів, що підвищить швидкість реагування, а структурована історія захворювань дозволить якнайкраще поставити діагноз та, як наслідок, покращити лікування. Більш того, спираючись на документ Міністерства Охорони Здоров’я України «МОЗ України: Що було, є і буде», у державі уже існують процеси, що направлені на перехід до використання сучасних технологій у медичній сфері: «Трансформація відбувається за рахунок підвищення ефективності, модернізації застарілої совєцької системи та поліпшення доступу до медичного обслуговування кращої якості» [3]. Тому система автоматизації обліку пацієнтів є ще одним кроком у розвитку держави.

1.2 Аналіз досвіду інших країн

Системи цифрових медичних карток уже застосовуються у багатьох провідних країнах світу. У постіндустріальний період розвитку, сучасні технології відіграють значну роль у всіх сферах діяльності людини.

Яскравим прикладом є досвід Сполучених Штатів Америки. В США існує три види медичних закладів:

– державні клініки;

– приватні прибуткові клініки;

– приватні безприбуткові.

Не залежно від способу фінансового забезпечення, кожен із видів медичних закладів намагається покращувати рівень медичних послуг що надаються. Сполучені Штати Америки є країною з найбільшим фінансуванням медичної сфери у світі, і не дивно, що провадження і

розвиток систем електронних медичних карток є одним із напрямів еволюції системи охорони здоров’я даної країни.

Невід’ємним чинником для впровадження систем електронних карток у США є зменшення витрат на медицину. Ринок систем ЕМК розвинувся, як одна з найбільш швидкозростаючих галузей США. Ряд федеральних політик та актів працювали як каталізатори зростання ринку і, як очікується, це також буде сприяти розвитку ринку. Прийняття системи ЕМК відкривають широкі ринкові можливості для системи охорони здоров’я. Ще у 2009 році в США було виділено 27,4 мільярдів доларів для переходу на систему електронних медичних карток. Закон пропонує лікарям до 63 тисяч доларів протягом п'яти років , якщо медики зможуть довести, що вони доцільно використовують такі системи, скануючи записи та подаючи приписи в електронному вигляді.

Такий технологічний прогрес спрощує лікування та діагностику. Крім того, особи, що належать до молодого населення є технічно спрямованими та цінують впровадження IT у системі охорони здоров'я. Цей сегмент населення витрачає більше коштів на електронне здоров'я та додатки для охорони здоров’я, і, як очікується, ця тенденція залишиться і в найближчому майбутньому.

Не менш важливим є збільшена потреба у спеціалістах в галузі розроблення та підтримки програмного забезпечення, що в свою чергу породжує нові робочі місця.

Національні дані Сполучених Штатів Америки за 2013 рік вказують на те, що 80% лікарів, що працюють в офісі, використовували частину технології електронного здоров'я / електронної системи медичного обліку (EMR) / систем електронних медичних записів (EHR).

Повністю функціональні можливості системи EHR включають електронні діаграми, замовлення тестів та управління звітами, описи, консультації та доповіді, підтримку клінічних рішень, підтримку управління захворюванням та звіти про якість.

Для прикладу розглянемо дослідження, що проводилося у 2010-2014 роках ресурсом «Black Book» [4]. Провівши аналіз даних, що наведені у таблицях 1.1 та 1.2 можна зробити висновок про стрімкий розвиток систем електронних медичних карток, та диджиталізації процесів, які відбуваються у медичній сфері Сполучених Штатів Америки.

Таблиця 1.1 – Загальна кількість користувачів систем електронних медичних карток в США.

Рік	Кількість лікарів, що користуються системами ЕМК	Кількість лікарів, що не використовують системи ЕМК
2014	22059	10727
2013	16623	26991
2012	12075	68113
2011	4506	21493
2010	787	3555

Таблиця 1.2 – Відносна кількість користувачів систем ЕМК

Рік	Лікарі, що використовують систему ЕМК, %
2014	81,6
2012	71,3
2009	44,5

Рік	Лікарі, що використовують систему ЕМК, %
2006	29

1.3 Аналіз існуючих програмних продуктів

На даний момент існує широке різноманіття систем електронних медичних записів, значна частина з яких орієнтована на англомовних користувачів. Розглянемо три найпопулярніші системи, такі як: PCC (Physician’s Computer Company) Pediatric EHR Solution, AdvanecMD та DrChrono EHR .

1.3.1 Аналіз системи Physicians Computer Company Pediatric EHR Solution

Дана система орієнтована для використання виключно вузькоспеціалізованими лікарями - педіатрами. Основний акцент і вектор застосування направлений на лікування та діагностування захворювань котрі в своїй більшості притаманні лише дітям та підліткам. У PCC присвячено всі ресурси речам, які найбільш важливі для педіатрів: спеціальні графіки, що відповідають їх робочому процесу, гнучкі педіатричні шаблони, затверджені графіки зростання синдрому Дауна та Фентона, конфіденційність для підлітків, передові інструменти управління клінікою та фінансами, а головне – мобільність рішень. Особлива спрямованість даної системи допомагає їй і надалі отримувати найвищі рейтинги задоволеності клієнтів у IT-галузі охорони здоров'я.

За допомогою PCC лікарі отримують миттєву інформацію про фінансове та клінічне здоров'я своєї практики. Панель практичних досліджень Vitals, що є частиною системи – це щоденний огляд, що забезпечує прості для розуміння графіки та оцінки, а також організацію пріоритетів.

На рисунку 1.2 зображено інтерфейс системи Physicians Computer Company Pediatric EHR Solution.

Рисунок 1.2 – Знімок екрану роботи системи PCC HER Solution [5]

Такий аналіз допомагає спеціалістові бути в курсі того, що відбувається у його практиці та що може вплинути на подальшу роботу.

Отже основною перевагою даної системи є її можливість до структуризації даних та їх аналізу, що допомагає надавати допомогу

лікарям під час їх практичної діяльності. Також слід звернути увагу на вузькоспеціалізованість системи, що можна віднести як до позитивного так і негативного аспекту її функціонування.

1.3.2 Аналіз системи AdvanecMD

Порівняно з Physicians Computer Company Pediatric EHR Solution, система AdvancedMD EHR орієнована на спеціаліста будь-якого профілю медичної сфери. Вона є модульною і включає в себе велику кількість можливостей для розширення базового функціоналу. Це є основною перевагою над рішеннями конкурентів.

На рисунку 1.3 зображено інтерфейс системи AdvanecMD.

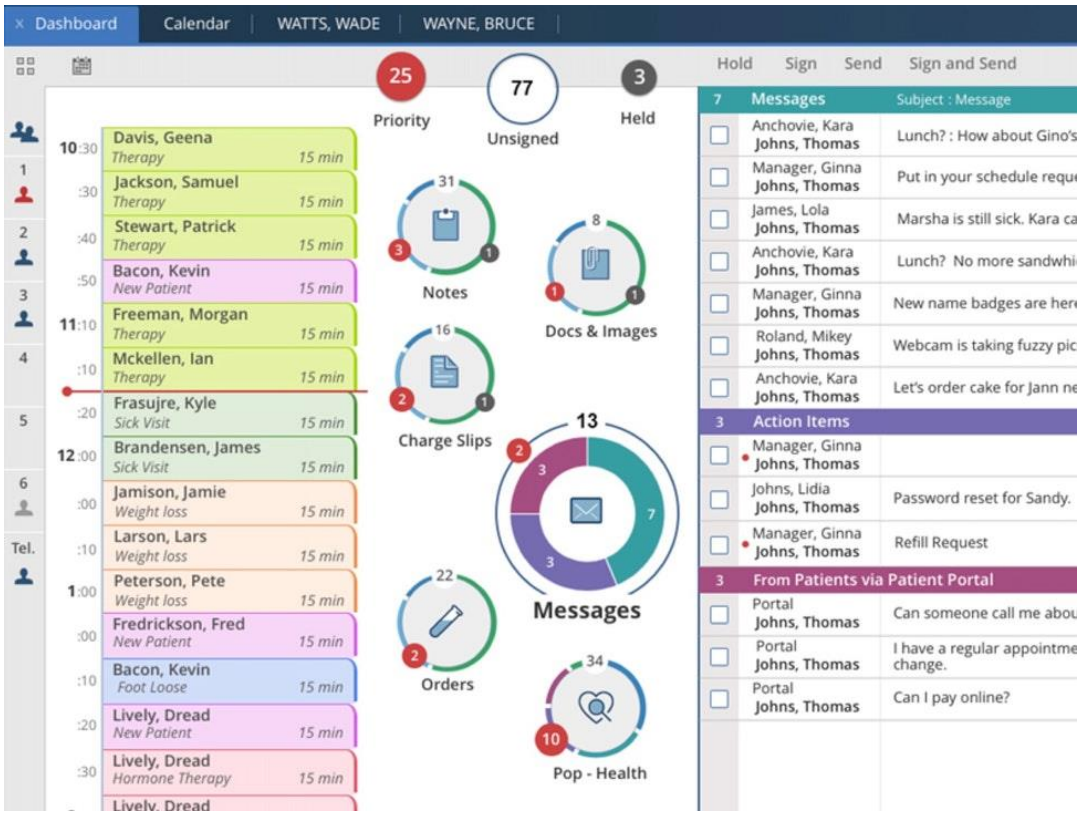


Рисунок 1.3 – Знімок екрану роботи системи AdvanecMD [6]

Система включає в себе такі функціональні складові:

– домашня сторінка, що дозволяє отримати швидкий доступ до довідників, розкладу роботи, пріоритетним задачам, елементам дій та карточок пацієнтів;

– стіл задач, який дозволяє сфокусуватись на об’ємах роботи і переглядати невирішені та важливі завдання, а також розуміти над чим потрібно працювати в першу чергу;

– зберігання даних на хмарних серверах із системами резервного копіювання;

– підсистема карток пацієнтів, що дозволяє у хронологічному порядку зберігати інформацію про пацієнтів, а також виділяти з них тих, у кого найчастіше виникають проблеми зі здоров’ям для більш швидкого реагування і доступу до графіків управління процесами;

– система сповіщень для пацієнтів, для покращення взаємовідносин між лікарем та пацієнтом, а також процесу лікування.

Поєднання таких складових, надає лікарю можливість в значно покращити свою ефективність.

1.3.3 Аналіз системи DrChrono EHR

Платформа EHR та медична платіжна платформа для мобільних телефонів DrChrono дозволяє лікарям керувати процесами прийому пацієнтів, клінічними діаграмами, виставленням рахунків та управлінням доходами. Він включає настроювані медичні форми, засоби електронного призначення, інструменти планування, перевірки відповідності пацієнтів у реальному часі та портал пацієнтів. Каталог додатків DrChrono також

пропонує безліч додатків та медичний API для розробників програм охорони здоров'я.

EHR DrChrono пропонує гнучкі або готові форми. Будь-який рецепт, включаючи електронне приписування контрольованих речовин, може бути надісланий в електронному вигляді на мобільний застосунок. Замовлення в лабораторію можна надсилати в електронному вигляді до понад 40 000 лабораторій по всій території США. Потім результати можуть бути безпосередньо завантажені у картку пацієнта. Спеціальні життєві позиції надають постачальникам можливість створювати базові дані про стан здоров'я та контролювати стан здоров'я кожного пацієнта протягом певного часу.

На рисунку 1.4 зображено інтерфейс роботи системи DrChrono EHR для портативних планшетів з операційною системою IOS.

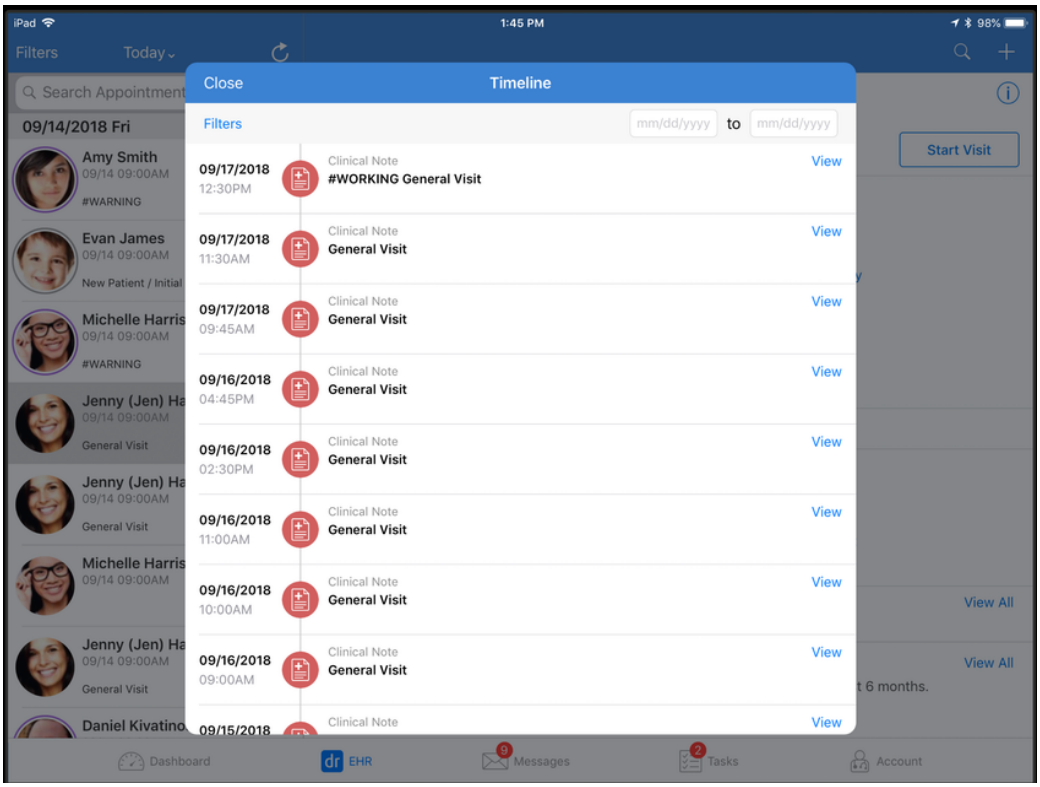


Рисунок 1.4 – Емуляція інтерфейсу роботи системи DrChrono EHR [7]

Постачальники можуть використовувати функцію Free Draw для анотації діаграм, рентгенівських знімків або інших файлів; вони також можуть використовувати вбудовані камери мобільних пристроїв для створення фотографій та вбудовування їх у картку пацієнта. Система оснащена попередньо заповненими кодами виставлення рахунків, та сумісна з системою хмарного зберігання.

1.4 Висновки до розділу

В даному розділі було проведено аналіз предметної області, проведено аналіз та порівняння існуючих рішень та аналогів. В результаті, було виявлено, що в плані розвитку систем медичного обліку, Україна значно відстає від таких країн як США. Дана галузь в нашій країні потребує переходу до використання цифрових даних та систем електронного медичного обліку. Також було визначено основні аспекти, що повинні бути втілені в розроблюваній системі.

2 МЕТОДИ ТА ТЕХНОЛОГІЇ

Одною із переваг розроблюваної системи повинна бути мобільність та швидкість доступу до інформації. Саме тому необхідно реалізувати систему, що здатна зберігати і передавати велику кількість даних, при цьому бути доступної для кожного. У зв’язку з цим було обрано до реалізації клієнт-серверну архітектуру застосунку. Сервер відповідатиме за збереження та обробку даних. Клієнт, в свою чергу, розробляється для мобільних платформ, найпопулярнішими з яких на даний момент є операційні системи Android та IOS.

2.1 Мова програмування C#

Для створення інформаційної системи «Електронна медична картка» було використано мову програмування C #, яка є однією з найбільш прийнятних, організованих та популярних мов програмування у світі. Її визнано однією з найвпливовіших і найпотужніших мов програмування. До переваг C# перед іншими мовами відносять:

– об'єктно-орієнтована мова. C # є об'єктно-орієнтованою мовою, це дозволяє створювати модульні програми та код для багаторазового використання;

– автоматичне збирання сміття. C # має дуже ефективну систему для видалення всього сміття, присутнього в системі;

– не виникає проблем з витоком пам'яті. На відміну від багатьох популярних мов C # має чіткий край у кількості зарезервованої пам’яті під час роботи;

– легкість у використанні. Велика кількість підтримуваних бібліотек, дозволяє розробнику використовувати уже реалізований функціонал без потреб його розроблення;

– кросплатформеність. З допомогою мови C# можна розробляти застосунки для більшості популярних операційних систем [10];

– легка інтеграція. Завдяки тому, що виконуваний код програм написаних на C# є кодом CLR, ці програми можуть використувовувати модулі, написані на інших мовах програмування з підтримкою CLR [11].

2.2 ASP.NET

Для розроблення серверної частини, було обрано технологію ASP.NET. ASP.NET розшифровується як Active Server Pages .NET і розробляється компанією Microsoft. ASP.NET використовується для створення веб-сторінок та інших складових веб-застосунків. Як член рамки .NET, ASP.NET є дуже цінним інструментом для програмістів і розробників, оскільки дозволяє їм створювати динамічні, насичені веб-сайти та веб-додатки.

Багато технологій, що використовуються для розроблення веб-застосунків, базуються на використанні лише скриптових мов програмування. ASP.NET не обмежується мовами скриптів, він дозволяє використовувати такі мови .NET, як C #, J #, Visul Basic тощо. Це дозволяє розробникам створювати більш гнучкі та функціональні програми, використовуючи Visual Studio, інструмент розробки, що надається компанією Microsoft . ASP.NET – це суто серверна технологія. Вона побудований на загальній мові виконання, яку можна використовувати на

будь-якому сервері Windows для розміщення потужних веб-сайтів і технологій ASP.NET [15].

У перші дні Інтернету, тобто до випуску Інтернет-інформаційних служб (IIS) у 1997 році, вміст веб-сторінок був значною мірою статичним. Ці веб-сторінки потрібно постійно та вручну змінювати. Виникла нагальна потреба у створенні динамічних веб-сайтів, які оновлювалися б автоматично.

Для того, щоб задовольнити цю потребу, на ринок було виведено Microsoft Active Server Pages (ASP). ASP, виконаний на стороні сервера, з його результатом надсилається у веб-браузер користувача, таким чином, дозволяючи серверу генерувати динамічні веб-сторінки на основі дій користувача. Даний процес зображено на рисунку 2.1.

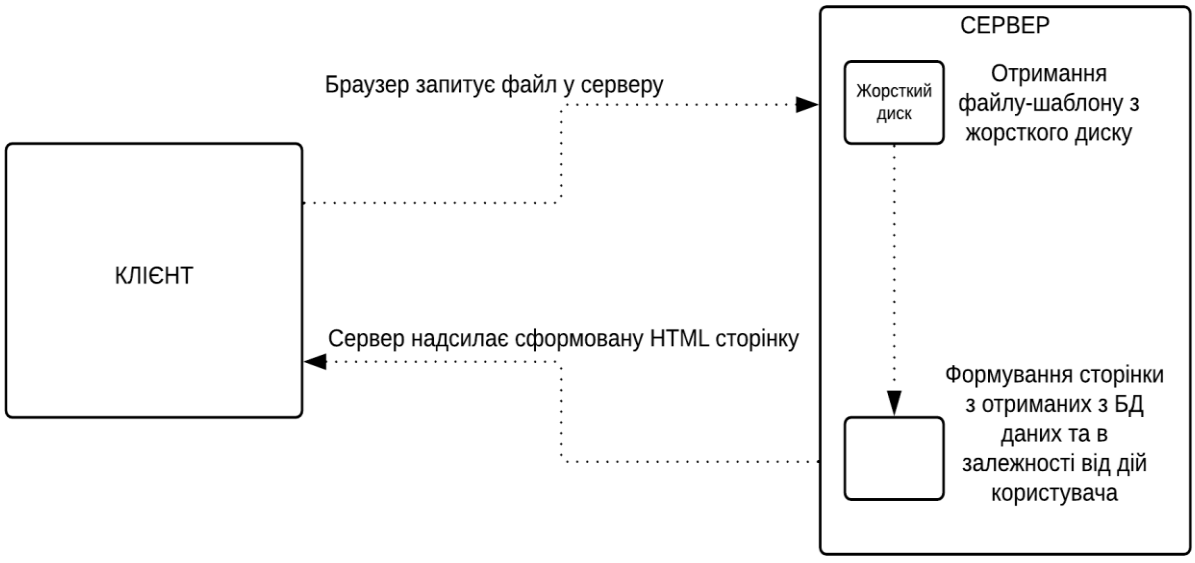


Рисунок 2.1 – Процес роботи системи на базі активних серверних сторінок

Ці серверні технології є важливим внеском у розвиток Інтернету. Amazon.com, eBay.com та багато інших популярних веб-сайтів використовують ASP.NET в якості основи для свого сайту.

ASP.NET має низку переваг перед іншими технологіями:

– ASP.NET зменшує кількість коду, необхідного для створення великих програм;

– завдяки вбудованій аутентифікації Windows та конфігурації програми, процес захисту програм є досить простим та ефективним;

– забезпечення вискої ефективності завдяки своєчасній компіляції, нативній оптимізації та кеш-сервісам;

– технологія ASP.NET доповнюється великим набором зручних інструментів для полегшення розробки у середовищі Visual Studio. WYSIWYG редагування, перетягування серверних елементів управління та автоматичне розгортання – лише деякі функції, які надає цей потужний інструментарій;

– забезпечення простоти реалізації, оскільки ASP.NET дозволяє легко виконувати загальні завдання, від простого подання форми та автентифікації клієнта до розгортання та налаштування сайту;

– зовнішній вигляд сторінок та внутрішня логіка є невід'ємними складовими одного елементу. Це забезпечує велику потужність та гнучкість веб-сторінок;

– усі процеси ретельно контролюються та керуються програмою виконання ASP.NET, так що, якщо процес перестав працювати, на його місці може бути створений новий процес, який допомагає постійно підтримувати програму для обробки запитів;

– це суто серверна технологія, тому ASP.NET-код виконується на сервері до того, як він буде відправлений у браузер;

– незалежність від конкретної мови програмування;

– ASP.NET застосунок легко розгорнути. Не потрібно реєструвати компоненти, оскільки інформація про конфігурацію вбудована в сам застосунок;

– веб-сервер постійно відстежує сторінки, компоненти та програми, що працюють на ньому. Якщо він помітить будь-які витоки пам'яті, нескінченні цикли, інші незаконні дії, він негайно знищує ці дії та перезавантажує себе;

– легко працює з ADO.NET, використовуючи функції прив'язки даних та форматування сторінок. Це програма, яка працює швидше і протидіє великим обсягам користувачів, не маючи проблем із продуктивністю [15].

2.3 REST API

Для передачі даних між сервером та клієнтом використовується технологія REST API. REST – це будь-який інтерфейс між системами, що використовують HTTP для отримання даних та генерації операцій над цими даними у всіх можливих форматах, таких як XML та JSON. Це все більш популярна альтернатива іншим стандартним протоколам обміну даними, таким як SOAP (Simple Object Access Protocol), які мають високу ємність, але є дуже складними. Іноді бажано використовувати більш просте рішення для обробки даних, таке як REST.

Особливості REST:

– протокол клієнта/сервера без громадянства: кожен HTTP запит містить всю необхідну інформацію для запуску REST, а це означає, що ні клієнту, ні серверу не потрібно пам'ятати жоден попередній стан, щоб його задовольнити. Як би там не було, деякі програми HTTP містять кеш-

пам'ять, відому як протокол кеш-сервера без стану клієнта: можна визначити деякі відповіді на конкретні запити HTTP як керовані, тому клієнт може в майбутньому запустити ту саму відповідь на однакові запити;

– у будь-якій системі REST та специфікації HTTP є чотири дуже важливі транзакції даних: POST (створення), GET (читання та консультація), PUT (редагування) та DELETE (видалення);

– об'єкти в REST завжди ідентифікуються за допомогою URI . Саме URI є єдиним ідентифікатором кожного ресурсу в системі REST. URI дозволяє отримувати доступ до інформації, щоб змінити або видалити її, або, наприклад, поділитися її точним розташуванням з третьою стороною;

– уніфікований інтерфейс: для передачі даних система REST застосовує конкретні дії (POST, GET, PUT та DELETE) на ресурси, за умови, що вони ідентифіковані з URI. Це полегшує отримання єдиного інтерфейсу, який поєднує процес з інформацією;

– використання гіпермедіа: гіпермедіа – термін, який є продовженням поняття гіпертексту. Ця концепція, взята на розробку веб-сторінок, – це те, що дозволяє користувачеві переглядати набір об’єктів за допомогою посилань HTML. Що стосується API REST, то поняття гіпермедіа пояснює здатність інтерфейсу розробки додатків для надання клієнту та користувачеві адекватних посилань для виконання конкретних дій над даними [13].

На рисунку 2.2 зображено процес функціонування системи з використанням REST API.

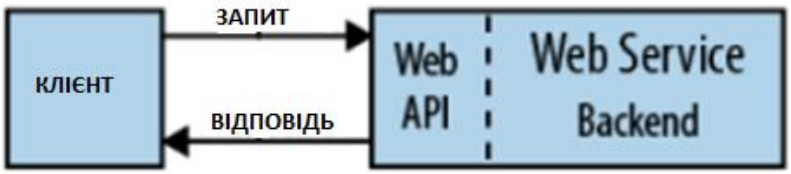


Рисунок 2.2 – Функціонування REST API

Переваги REST:

– розмежування між клієнтом та сервером: протокол REST повністю відокремлює інтерфейс користувача від сервера та зберігання даних. Це має деякі переваги при розробці застосунків. Наприклад, це покращує портативність інтерфейсу для інших типів платформ, збільшує масштабованість проектів і дозволяє самостійно розвивати різні компоненти систем;

– видимість, надійність та масштабованість. Розмежування між клієнтом і сервером має одну очевидну перевагу, і це те, що кожна команда розробників може масштабувати продукт без особливих проблем. Вони можуть мігрувати на інші сервери або вносити всі види змін у базу даних за умови коректного надсилання даних із кожного запиту;

– API REST завжди не залежить від типу платформи або мов: API REST завжди адаптується до типу синтаксису або платформ, що використовуються, що дає значну свободу при зміні або тестуванні нових середовищ в рамках розробки. З API REST ви можете мати сервери PHP, Java, Python або Node.js. Єдине, що обов'язково, щоб відповіді на запити завжди відбувалися мовою, що використовується для обміну інформацією, як правило, XML або JSON.

2.4 Xamarin

Xamarin – платформа з відкритим кодом для створення сучасних та ефективних додатків для iOS, Android та Windows з .NET. Xamarin – рівень абстракції, який управляє зв’язком спільного коду з базовим кодом платформи. Xamarin працює в керованому середовищі, яке забезпечує такі зручності, як розподіл пам’яті та збирання сміття.

Xamarin дозволяє розробникам спільно використовувати 90% своїх додатків на різних платформах. Він надає розробникам модливість писати свою бізнес-логіку однією мовою (або використовувати повторно існуючий код програми), але досягати відповідної продуктивності, інтерфейсу та функціоналу на кожній платформі.

Xamarin.Forms – це структура інтерфейсу з відкритим кодом, що дозволяє розробникам створювати програми Xamarin.Android, Xamarin.iOS та Windows з єдиної спільної бази коду.

Xamarin.Forms дозволяє розробникам створювати користувальницькі інтерфейси в XAML з кодовим доповненням у C#. Ці інтерфейси відображаються як виконавчі елементи управління на кожній платформі.

Xamarin.Forms надає послідовний API для створення елементів інтерфейсу для кожної платформи. Цей API може бути реалізований у XAML або C# і підтримує прив'язку даних для таких моделей, як Model-View-ViewModel (MVVM).

Під час виконання Xamarin.Forms використовує візуалізатори платформи для перетворення елементів міжплатформового інтерфейсу в початкові елементи управління на Xamarin.Android, Xamarin.iOS. Це дозволяє розробникам отримувати нативні вигляд та продуктивність, усвідомлюючи переваги спільного використання коду на всіх платформах.

Програми Xamarin.Forms зазвичай складаються із спільної бібліотеки .NET Standard та окремих проектів платформи. Бібліотека, що поділяється, містить сторінки інфтерфейсу XAML або C # та будь-яку бізнес-логіку. Проекти платформ містять будь-яку логічну платформу або пакети, необхідні додатку.

Xamarin.Forms використовує платформу Xamarin для запуску .NET додатків на власних платформах.

2.5 NFC

NFC – це технологія безконтактного зв’язку, яка дозволяє двом пристроям, розташованим у безпосередній близькості (відстань між двома пристроями не повинна перевищувати 10 см) швидко підключатись та передавати дані, не користуючись Інтернетом. У будь-якому зв'язку NFC є два пристрої, один – активний пристрій, також відомий як зчитувач, а інший – пасивний пристрій. Іноді зв'язок також може відбуватися між двома активними пристроями, наприклад, зв’язок між двома смартфонами з підтримкою NFC. Пасивний пристрій зазвичай зберігає інформацію, яку читають активні пристрої по радіочастотному діапазону Пристрої NFC працюють у трьох режимах:

– режим зчитування тегів / запису тегів (рисунок 2.3). У цьому режимі роботи зв'язок відбувається між активним пристроєм та пасивним пристроєм. Наприклад, читання тегів NFC, вбудованих у смарт-карту за допомогою смартфона. Тут тег NFC – це пасивний пристрій, а смартфон буде діяти як активний пристрій. Тег NFC, вбудований у карту, може містити будь-яку інформацію залежно від того, для якої мети використовується ця карта. Тег NFC може містити інформацію про будь-які спеціальні пропозиції, які надає компанія, або інформацію про новий продукт, який компанія буде запускати найближчим часом, або може просто мати URL-адресу компанії, яка доставить клієнта безпосередньо на веб-сайт компанії. Тег NFC є пасивним пристроєм, тому що він може зберігати дані і не здатний читати жоден інший тег NFC, тоді як смартфон з підтримкою NFC може читати та навіть записувати дані на будь-який тег NFC, таким чином він діє як активний пристрій або також може бути називається читачем;

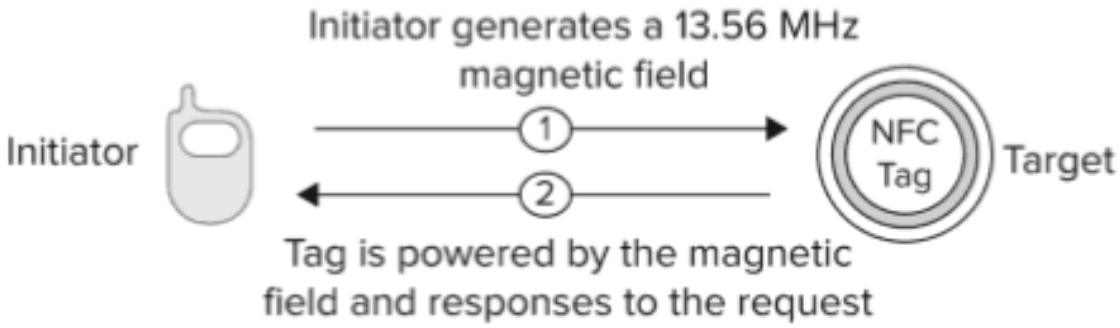


Рисунок 2.3 – Режим зчитування тегів/запису тегів [16]

– в режимі «одноранговий» (рисунок 2.4). В цьому режимі роботи зв'язок відбувається між двома активними пристроями або іншими словами між двома пристроями, що підтримують NFC. Прикладом такого типу зв'язку може бути обмін контактними даними між двома смартфонами з підтримкою NFC, або обмін фотографіями чи будь-якими іншими даними. Це робиться просто прикладанням або піднесенням на близьку відстань двох смартфонів з підтримкою NFC;

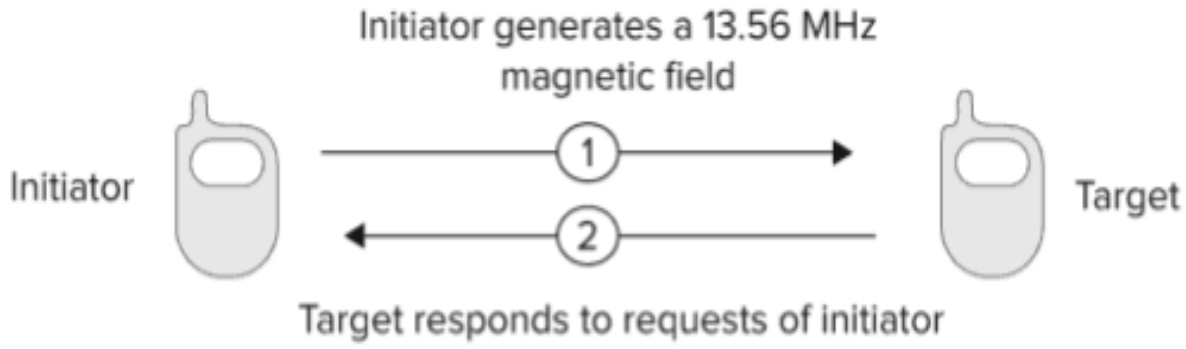


Рисунок 2.4 – Режим «одноранговий» [16]

– режим емуляції картки (рисунок 2.5). Цей режим використовується в основному для фінансових транзакцій, тобто користувач може використовувати цей режим для оплати придбаного товару або для придбання квитка. Пристрій з підтримкою NFC, що використовується в цьому режимі, діє як смарт-карта. Прикладами пристроїв із підтримкою NFC, які діють як смарт-картки, є кредитна картка, смартфони тощо. У цьому режимі спілкування відбувається між пристроєм, що підтримує NFC, та віддаленим зчитувачем карт. Найбільшою перевагою використання цього режиму є те, що навіть якщо звичайна кредитна картка перетворена на пристрій з включенням NFC, тобто в смарт-карту, вбудуючи в неї чіп NFC, вона все одно буде працювати як звичайна. А просто вставивши NFC-чіп у зчитувач карток, який використовується для читання звичайних безконтактних карт, можна використовувати його для читання смарт-карти, таким чином, не змушуючи змінювати існуючу інфраструктуру [17].

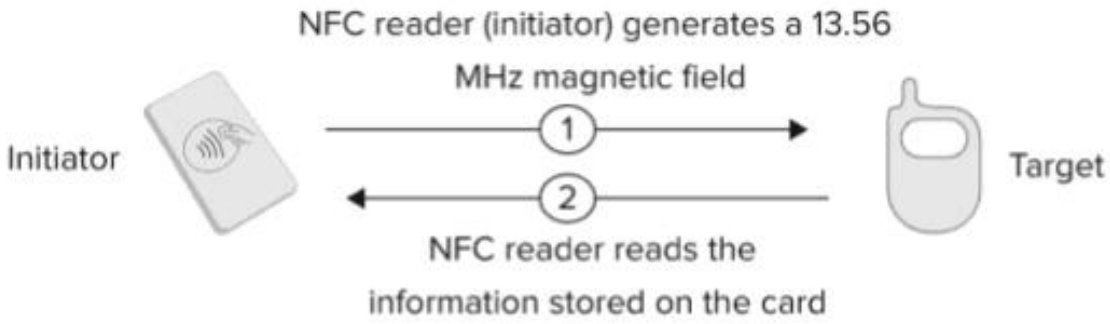


Рисунок 2.5 – Режим емуляції картки [16]

Теги NFC належать до категорії пасивних тегів, тобто вони не здатні читати чи записувати дані на будь-який інший тег. Їх можна використовувати для різних цілей, як, наприклад, їх можна вбудувати в смарт-карту, або вставити в наручний браслет, або вбудувати у візитну картку або в будь-яку програму, де не потрібно передавати чи зберігати

великі дані. Їх можна читати чи записувати на активних пристроях з підтримкою NFC, таких як смартфони. URL-адреси є найбільш бажаною формою даних, що зберігаються в тезі NFC, оскільки вони займають дуже мало місця і можуть містити багато інформації. Наприклад, зберігаючи URL-адресу на тезі NFC, вбудованому в смарт-плакат, компанія може перенаправляти клієнтів на свій веб-сайт, де вони відображали останні пропозиції, а не зберігати всі пропозиції на самому тезі. Беручи до уваги формат і ємність тегу, теги NFC діляться на 4 різні типи [18].

2.6 Висновки до розділу 2

В даному розділі було розглянуто методи та технології, що будуть використані у розробці інформаційної системи «Електронна медична картка». Обґрунтовано використання обраних інструментів та методології розроблення, а саме реалізації клієнт-серверної системи з використанням мови C#, фрейм ворків Xamarin.Forms та ASP.NET та технологій комунікацій ближнього поля.

3 ПРОЄКТУВАННЯ ПРОГРАМНОГО ПРОДУКТУ

3.1 Діаграма IDEF0

IDEF0 (інтегроване визначення для функціонального моделювання) – це функціональне моделювання для опису виробничих функцій, що забезпечує моделювання для аналізу, розробки, перепроектування та інтеграції інформаційних систем, бізнес-процесів або інженерії програмного забезпечення. IDEF0 є частиною сімейства методів IDEF, що є мовою моделювання в галузі інженерії програмного забезпечення та побудований на мові функціонального моделювання SADT Структурована методика аналізу та проектування (SADT) – це методологія інженерії та інженерії програмного забезпечення для опису систем як ієрархії функцій. SADT – це модель структурованого аналізу, яка використовує два типи діаграм: моделі діяльності та моделі даних [19].

На рисунку 3.1 зображено діаграму IDEF0 функціонування системи в цілому. Вона описує матеріали, стандарти та дані, що використовуються та оброблюються системою загалом.

Перша стрілка показує вхідні дані системи, а саме інформацію про стан здоров’я пацієнта. Остання стрілка результат роботи системи – оцифровані дані про стан здоров’я пацієнта.

Верхні стрілки описують стандарти, використані системою:

– OPC (Open Platform Communications) стандарти проектування баз даних;

– стандарт МОЗ 025/о;

– NFC стандарти передачі даних;

– HTTPS стандарт.

Ніжні стрілки описують матеріали, використані системою:

– користувач;

– телефон;

– Чіп NFC;

– Сервер;

– Інтернет.

Рисунок 3.1 – Діаграма IDEF0

3.2 Діаграма декомпозиції IDEF0

Для більш повноцінного розуміння роботи системи, проводиться процес декомпозиції. Декомпозиція – це процес розбиття рішення великої задачі, на менші функціональні блоки. В даному випадку, процес роботи

системи електронних медичних карток, розбито на підпроцеси нижнього рівня.

Кожен блок, так само як і для діаграми першого рівня, описується вхідними даними, використаними стандартами та матеріалами і вихідними даними.

У даному випадку, робота системи на першому рівні декомпозиції описується чотирьома підзадачами (рисунок 3.2): реєстрація пацієнта в системі, зчитування даних з мітки, отримання даних із серверу та вивід даних на екран користувача.

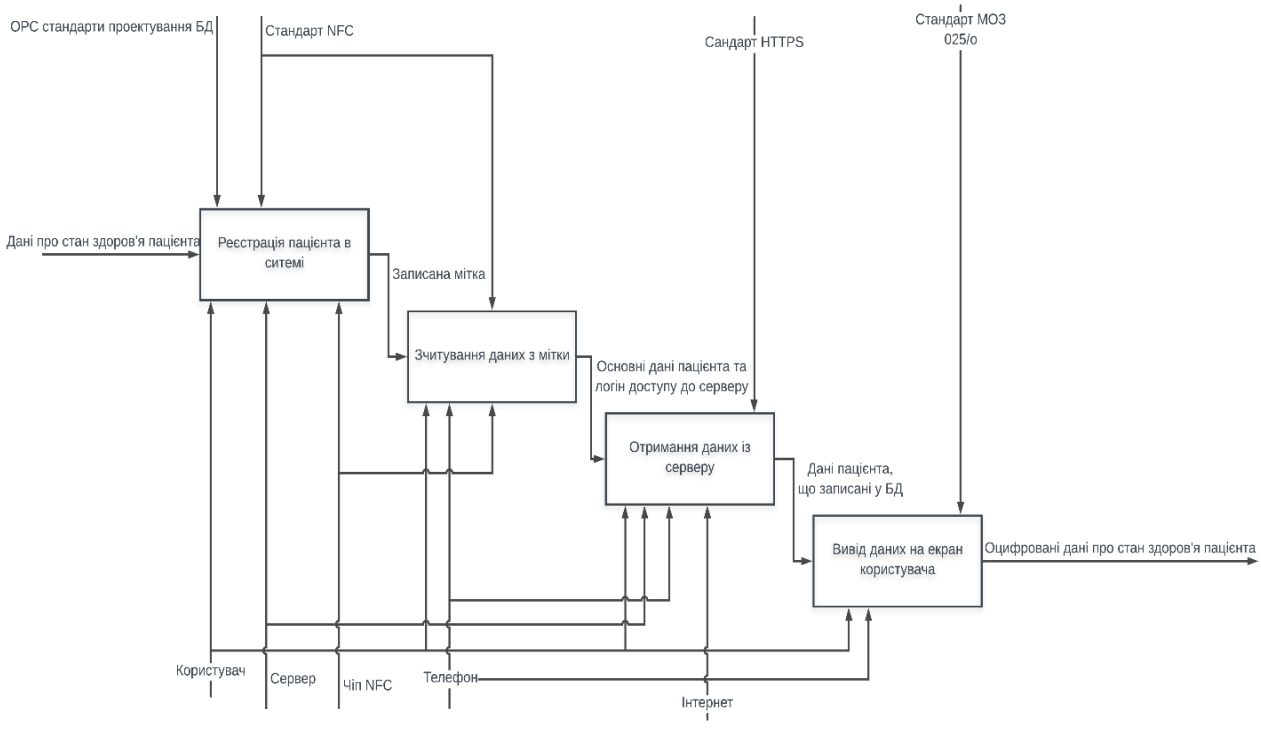


Рисунок 3.2 – Діаграма першого рівня декомпозиції IDEF0

Кожен блок діаграми декомпозиції першого рівня може бути розбитий на підзадачі так само як і діаграма верхнього рівня. Такий процес може проводитись, доки проектувальник буде вважати це необхідним. На

рисунку 3.3 зображено процес декомпозиції модулю зчитування даних з мітки. Даний процес складається з наступних функціональних завдань:

а) включення режиму зчитування міток поля ближньої комунікації;

б) зчитування знайденої мітки;

в) розшифровування отриманих даних;

г) перевірка відповідності даних встановленому формату.

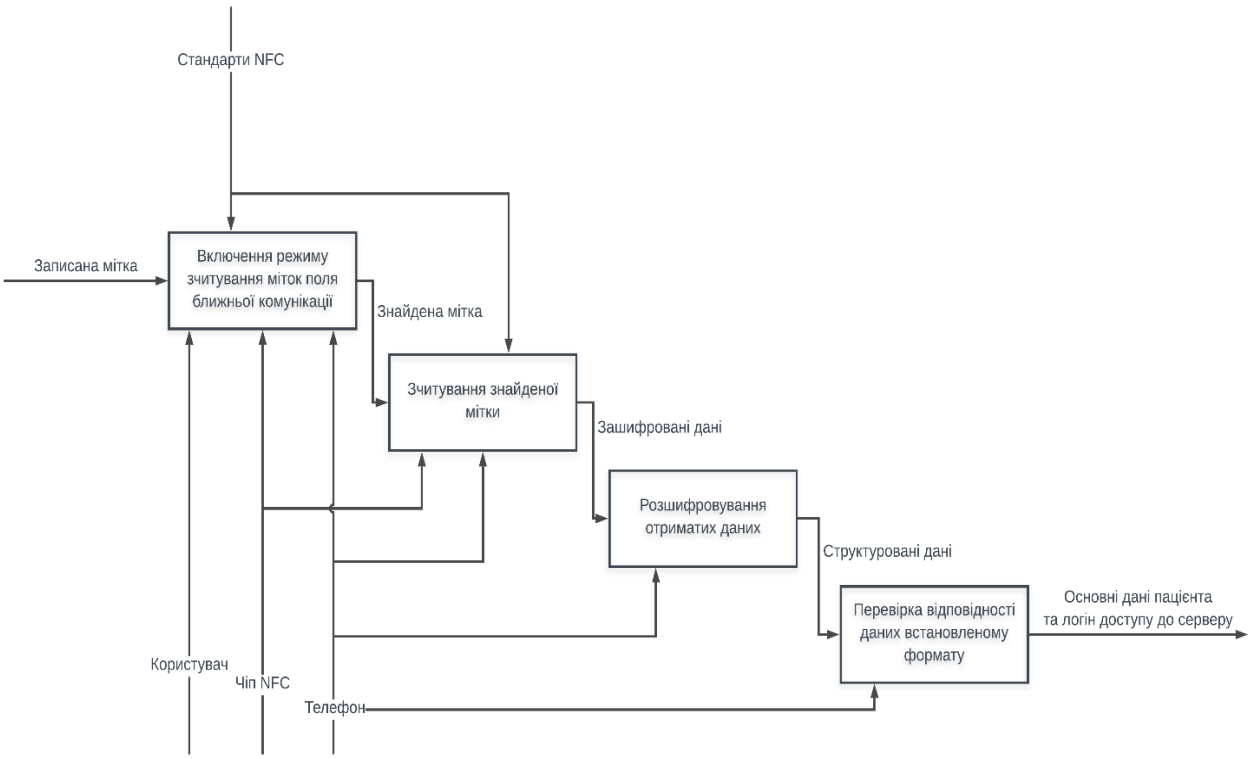


Рисунок 3.3 – Діаграма другого рівня декомпозиції IDEF0.

3.3 Методологія IDEF3

IDEF3 – член сімейства методів моделювання IDEF, який може бути використаний для опису операцій у бізнес-процесі. Діаграма IDEF3 надає опис фактичного потоку процесу в організації чи підприємстві, або змін,

які відбуваються з об'єктом у цій системі. Цей метод засвоєння знань реєструється у двох різних перспективах – користувачі мають змогу створювати як технологічну схему, так і об'єктну схему, використовуючи схематичні символи IDEF3 [20].

На рисунку 3.4 зображена діаграма IDEF3, яка описує роботу блоку обробки інформації.

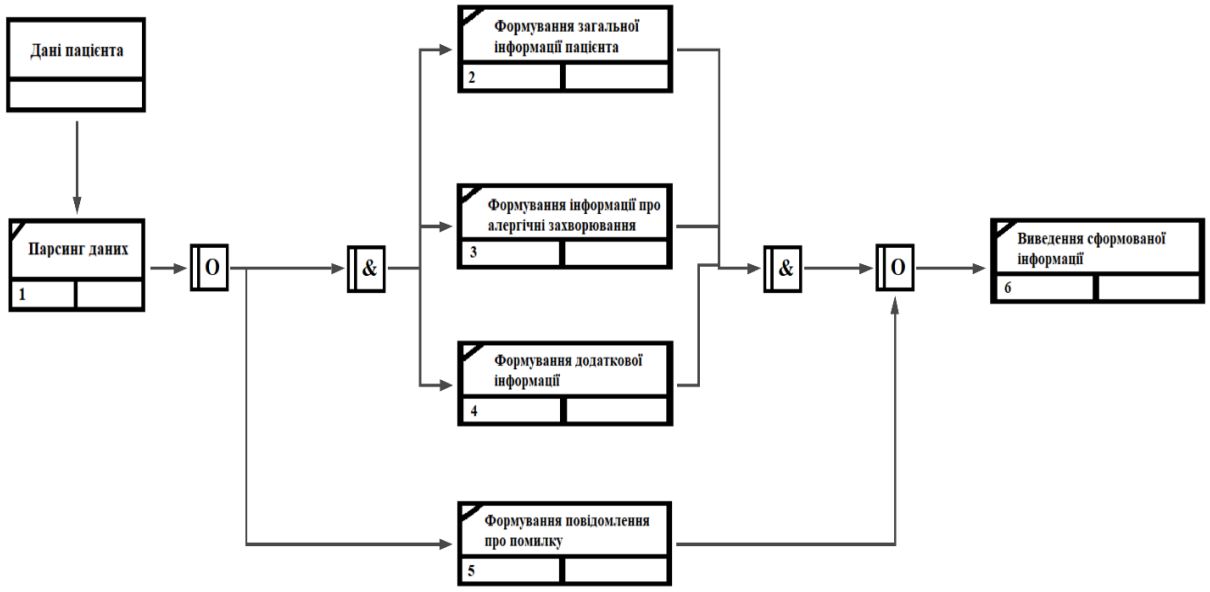


Рисунок 3.4 – Діаграма IDEF3

3.4 Діаграма варіантів використання

Діаграма варіантів використання UML є основною формою системних / програмних вимог для нової програмної програми. Випадки використання вказують на очікувану поведінку, а не точний спосіб її втілення. Зазначені варіанти використання можуть бути позначені як текстовим, так і візуальним поданням. Ключова концепція моделювання випадків

використання полягає в тому, що вона допомагає розробити систему з точки зору кінцевого користувача. Це ефективний прийом передачі поведінки системи в термінах користувача шляхом визначення конкретної зовнішньої поведінки системи [21].

Діаграма варіантів використання зазвичай проста. У ній не відображаються деталі варіантів використання, вона лише підсумовує деякі взаємозв'язки між випадками використання, дійовими особами та системами.

Діаграма варіантів використання не показує порядок виконання кроків для досягнення цілей кожного випадку використання.

На кресленику ІТ61.220БАК.004 Д1 зображена діаграма варіантів використання. В залежності від ролі, користувач може виконувати ті чи інші дії. Користувач із роллю пацієнт, може:

– подати заявку на підключення до системи;

– переглянути інформацію про свій стан здоров’я;

– подати заявку на зміну особистих даних;

– подати заявку на відключення від системи.

Користувач із роллю лікар, в свою чергу, може:

– ввести дані про пацієнта в систему, після його заявки;

– переглянути інформацію про будь-якого свого пацієнта;

– змінити дані пацієнта, після його заявки.

Користувач із роллю адміністратор, може видалити пацієнта з системи, після його заявки.

3.5 Діаграма класів

Діаграма класів надає огляд програмної системи шляхом відображення

класів, атрибутів, операцій та їх взаємозв'язків. Ця діаграма включає в себе назву класу, атрибути та операції в окремих призначених відсіках.

Діаграма класів визначає типи об’єктів у системі та різні типи зв’язків, що існують серед них. Це дозволяє надати оцінку структури програми на високому рівні абстракції. Цей метод моделювання може працювати майже з усіма об'єктно-орієнтованими методами. Клас може посилатися на інший клас. Клас може мати свої об'єкти або може успадковувати інші класи [22].

Діаграма класів допомагає побудувати код для розробки програмного забезпечення.

На рисунку 3.5 зображена діаграма класів.

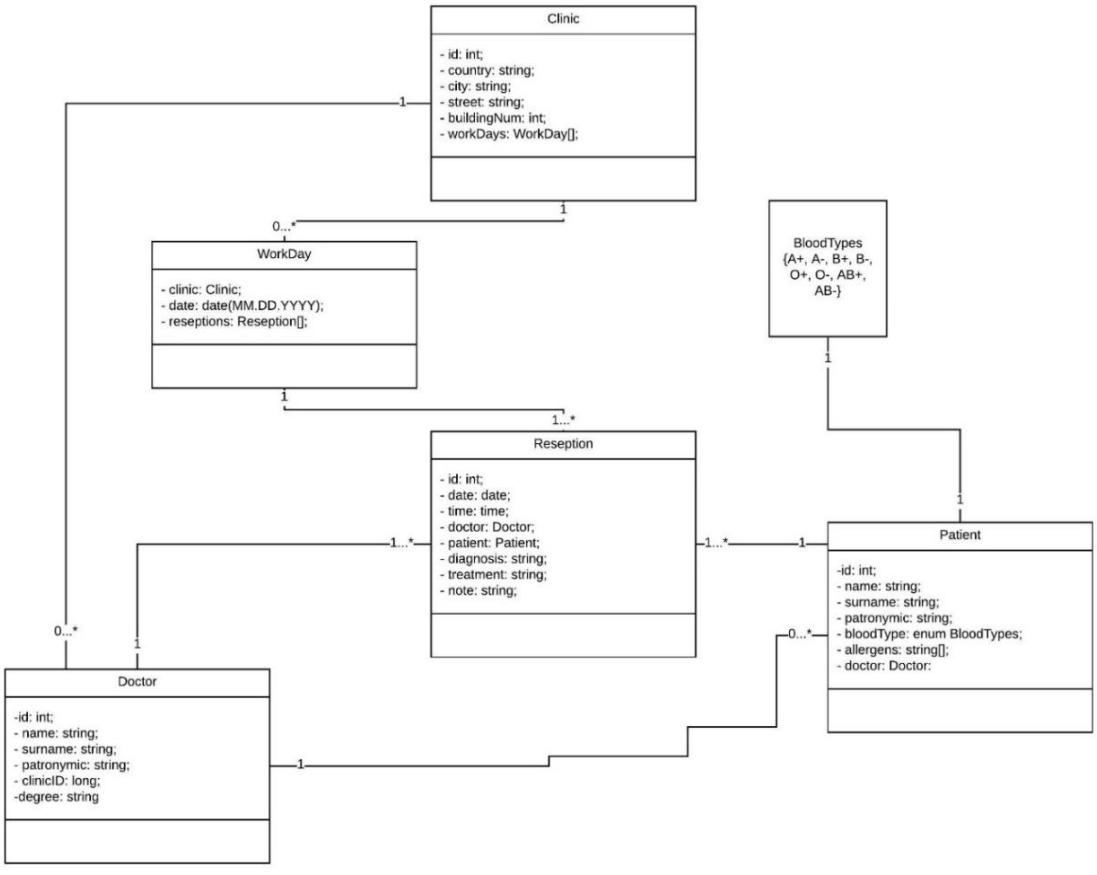


Рисунок 3.5 – Діаграма класів в нотації UML

3.6 Діаграма послідовностей

Діаграми послідовності UML – це діаграми взаємодії, які детально описують спосіб виконання операцій. Вони фіксують взаємодію між об'єктами в контексті співпраці. Діаграми послідовності орієнтовані на час, і вони візуально показують порядок взаємодії, використовуючи вертикальну вісь діаграми, щоб представити час, у який повідомлення надсилаються та коли. На кресленику ІТ61.220БАК.004 Д2 зображено діаграму послідовностей [23].

3.7 Діаграма комунікацій

Діаграми комунікацій UML, як діаграми послідовності – свого роду діаграма взаємодії, показує, як взаємодіють об'єкти. Діаграма комунікацій – це розширення об'єктної діаграми, що показує об'єкти разом із повідомленнями, які надходять від одного до іншого.

На рисунку 3.6 зображена діаграмма комунікацій.

3.7 Діаграма станів

Діаграма станів складається з станів, переходів, подій та діяльності. Діаграми стану використовуються для ілюстрації динамічного перегляду системи. Вони особливо важливі при моделюванні поведінки інтерфейсу, класу чи співпраці. Діаграми стану підкреслюють порядок впорядкованості поведінки об'єкта, що особливо корисно при моделюванні реактивних систем. Діаграма станів зображена на кресленику ІТ61.220БАК.004 Д3.

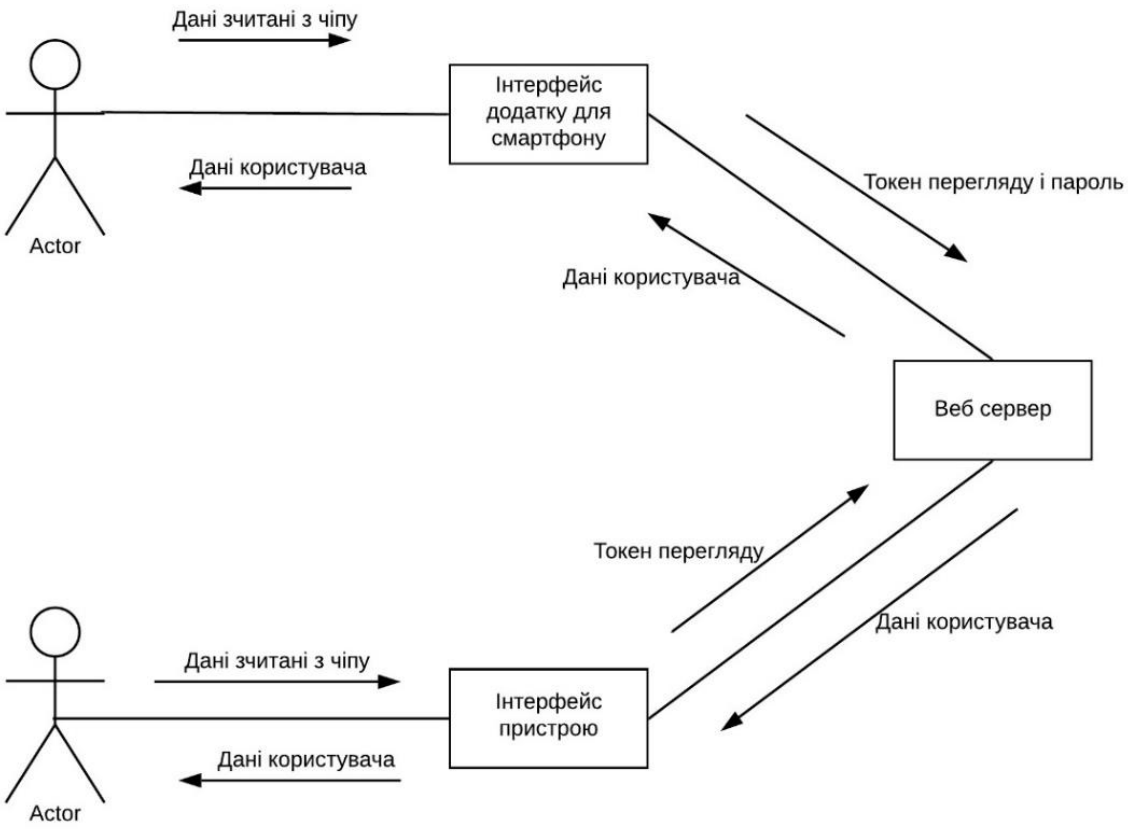


Рисунок 3.6 – Діаграма комунікацій

3.8 Діаграма компонентів

Діаграми компонентів UML використовуються при моделюванні фізичних аспектів об'єктно-орієнтованих систем, які використовуються для візуалізації, уточнення та документування систем на основі компонентів, а також для побудови виконуваних систем за допомогою прямої та зворотної інженерії. Діаграми компонентів – це, по суті, діаграми класу, які фокусуються на компонентах системи, які часто використовуються для моделювання статичного виду реалізації системи [24].

На рисунку 3.7 зображена діаграма компонентів в нотації UML.

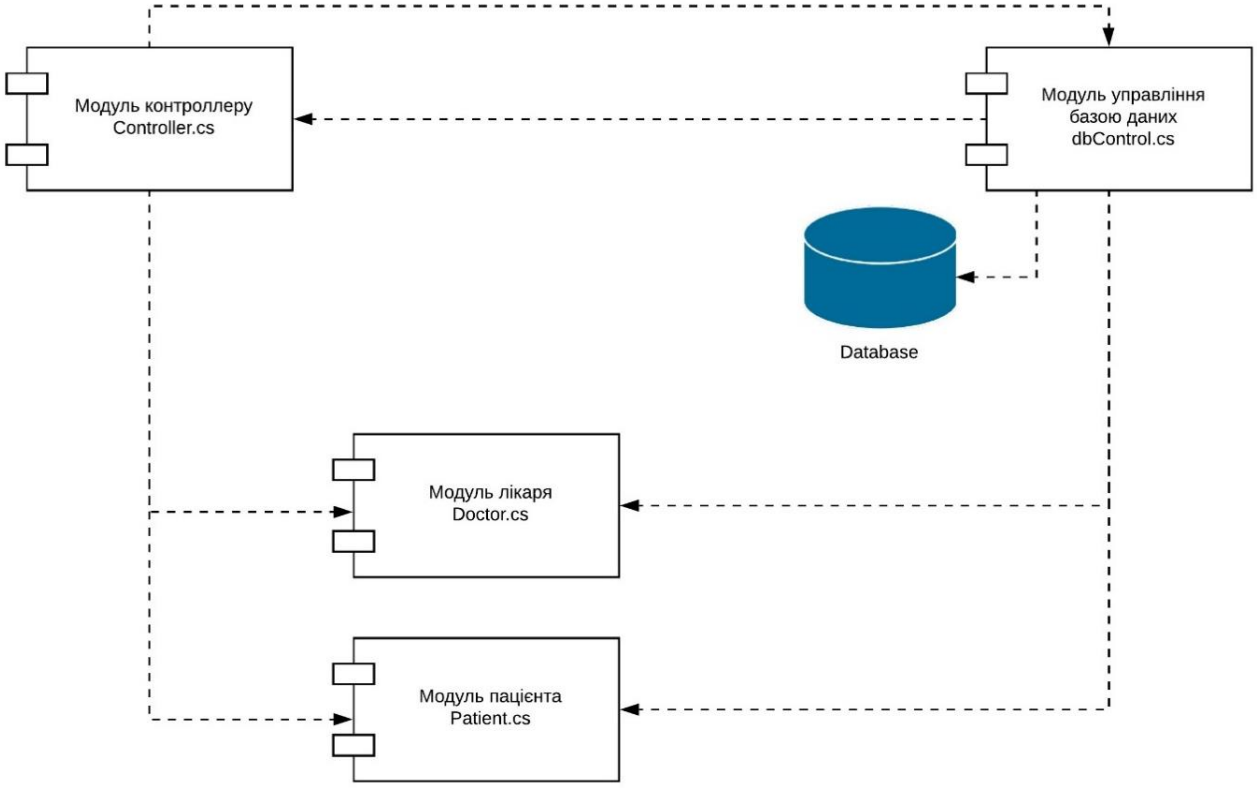


Рисунок 3.7 – Діаграма компонентів

3.9 Діаграма діяльності

Діаграма діяльності – ще одна важлива діаграма поведінки в діаграмах UML для опису динамічних аспектів системи. Діаграма діяльності по суті є вдосконаленою версією діаграми потоків, яка моделює потік від однієї діяльності до іншої.

Діаграми діяльності описують, як діяльності координуються для надання послуг, які можуть мати різний рівень абстракції. Як правило, подія повинна бути досягнута шляхом виконання деяких операцій, особливо якщо операція призначена для досягнення ряду різних цілей, які потребують координації, або як події в одному випадку використання стосуються один одного, зокрема, використовують випадки, коли

діяльність може перекриватися і вимагати координації. Вони також підходить для моделювання того, як колекція випадків використання координується для представлення бізнес-процесів. На кресленику ІТ61.220БАК.004 Д4 зображена діаграма діяльності.

3.10 Діаграма розгортання

Діаграма розгортання UML – це діаграма, яка показує конфігурацію вузлів виконання та компонентів, що розміщуються на них. Діаграми розгортання – це своєрідна структурна діаграма, яка використовується при моделюванні фізичних аспектів об'єктно-орієнтованої системи. Їх часто використовують для моделювання статичного перегляду системи розгортання.

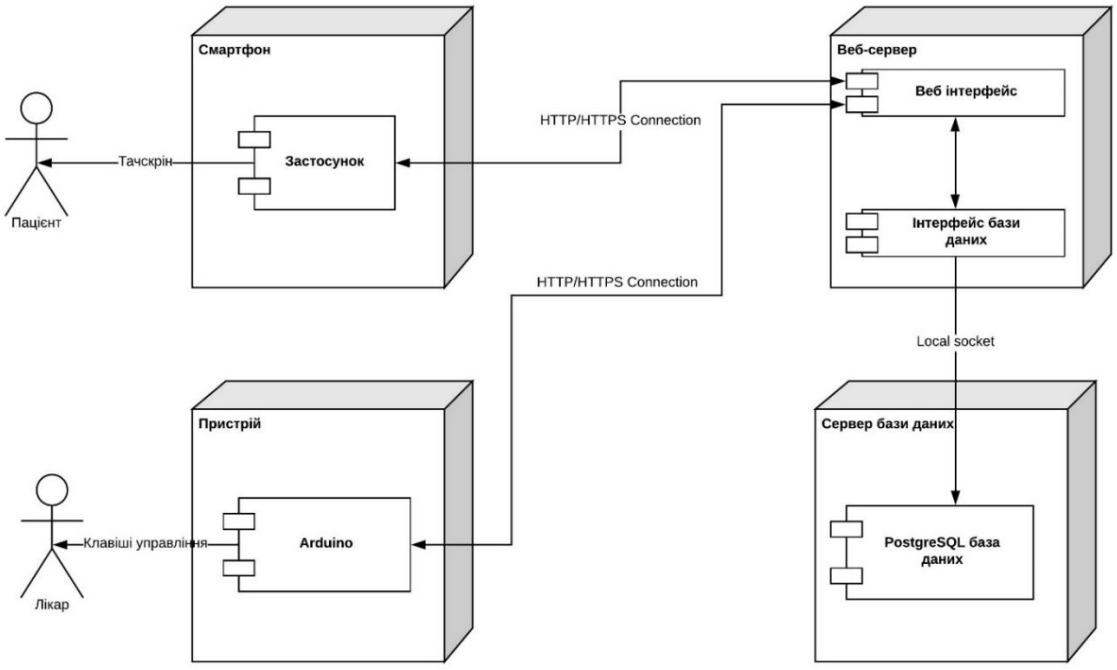


Рисунок 3.8 – Діаграма розгортання

Діаграму розгортання системи можна описати наступним чином. Програма відправляє запити та отримує відповіді від web-сервера. Сервер надає користувацький інтерфейс, а також використовує інтерфейс бази даних для доступу до даних, їх зміни, видалення, модифікації. Діаграму розгортання показано на рисунку 3.8.

3.11 Опис дерева цілей

Проміжна карта цілей або дерево цілей є в першу чергу інструментом раціонального аналізу всіх передумов, тобто необхідних умов для досягнення цілі та їх залежностей.

Дерево цілей є центральним інструментом процесу логічного мислення і доповненням до оригінальних інструментів логіки під назвою «процеси мислення» з теорії обмежень.

У самій верхівці дерева міститься мета або бачення. На наступному рівні три – максимум п’ять критичних факторів успіху – це основні цілі, які обов'язково потрібно досягти для досягнення мети.

Критичні фактори успіху можна розуміти як кінцеві основні етапи перед досягненням мети, а мета – це унікальна фінішна лінія [25].

На кресленику ІТ61.220БАК.004 Д5 зображено опис дерева цілей.

3.12 Висновки до розділу 3

В розділі докладно розглянуто усі етапи проєктування програмного додатку. Було розроблено контекстну діаграму, описано діаграму декомпозиції першого та другого рівня, діаграму дерева вузлів, діаграму послідовності та діаграму станів.

4 РЕАЛІЗАЦІЯ СИСТЕМИ

Для уникнення необхідності синхронізації даних між різними користувачами системи, та неминучого розростання кількості пам’яті, яку займає клієнтський застосунок, для реалізації було обрано клієнт-серверну архітектуру. Модель клієнт-сервер – це розподілена структура додатків, яка розбиває завдання або навантаження між постачальниками ресурсу або послуги, що називаються серверами, і запитувачами послуг, що називаються клієнтами. [26] Часто клієнти та сервери спілкуються через комп'ютерну мережу на окремому апаратному забезпеченні, але і клієнт, і сервер можуть знаходитись в одній системі. Хост сервера запускає одну або декілька серверних програм, які діляться своїми ресурсами з клієнтами. Клієнт не ділиться жодним із своїх ресурсів, але він вимагає вмісту чи послуги від сервера. Таким чином, клієнти ініціюють сеанси зв'язку із серверами, які очікують на вхідні запити. Завдяки такій реалізації, користувач не зберігає дані на своєму пристрої, а потребує тільки застосунок для подавання команд та виводу тієї частини інформації, яку було запитано. Усі ж дії із даними в виконуються на сервері.

Враховуючи результати аналізу, основною технологією для розроблення програмного продукту обрано мову програмування C#. Головним критерієм вибору стала можливість реалізації як серверного застосунку, так і клієнтського за допомогою однієї мови програмування. Механізми серверної частини системи було реалізовано за допомогою .net Core, а клієнтської – Xamarin.Forms. Також, перевагою використання C# є можливість розширення уже існуючого рішення, шляхом реалізації механізмів веб-застосунку з використанням тих самих інструментів та технологій.

4.1 Розроблення застосунку серверної частини системи

Проаналізувавши уже існуючі рішення і варіанти розвитку системи, можна зробити висновок, що головною умовою під час розробки застосунку серверної частити є її гнучкість і незалежність від клієнтського застосунку. Саме тому, для її реалізації було обрано архітектуру REST API. Дана технологія дозволяє створити набір команд, які можуть бути надіслані до серверу з будь якого клієнтського застосунку, та оброблені ним.

Серверний застосунок можна умовно розділити на три складові:

– шар бізнес логіки;

– база даних;

– API.

Процес реалізації доцільно розпочинати із шару бізнес логіки, адже на його основі будується структура бази даних та команди REST API.

4.1.1 Шар бізнес логіки

Дані, що будуть оброблятися та зберігатися мають об’єктну структуру, саме тому для реалізації обрано об’єктно орієнтовану парадигму програмування. Об'єктно-орієнтоване програмування – парадигма програмування, в якій програми визначаються за допомогою об'єктів – елементів, що з'єднують стан (тобто дані, які зазвичай називають полями) та поведінку (тобто процедури, тут: методи).

Найбільша перевага програмування, проектування та об’єктно-орієнтованого аналізу – сумісність цього підходу з реальністю – людський мозок, природно, найкраще адаптується до такого підходу під час обробки інформації.

В результаті, було створено п’ять класів, що описують структуру основних об’єктів що беруть участь в роботі програми.

Клас «Пацієнт» (Patient.cs) описує структуру основних даних пацієнта, що користується розробленою системою електронних карток. В таблиці 4.1 описано поля даного класу.

Таблиця 4.1 – Опис класу «Пацієнт»

Поле	Тип	Пояснення
Ідентифікатор	Рядок	Є унікальним для кожного пацієнта. Дозволяє ідентифікувати його в межах системи
Ім’я	Рядок	Ім’я, що вказано у документі, наданому під час реєстрації
Прізвище	Рядок	Прізвище що вказано у документі, наданому під час реєстрації
По батькові	Рядок	По батькові що вказано у документі, наданому під час реєстрації
Тип крові	Перечислення	Група крові пацієнта та резус фактор. Вибирається із вбудованого перечислення

Поле	Тип	Пояснення
Алергени	Масив рядків	Зберігає перелік алергенів, що викликають негативну реакцію організму у пацієнта
Лікар	Посилання	Вказує на лікаря, що є сімейним для даного пацієнта

Клас «Лікар» (Doctor.cs) описує структуру основних даних лікаря, що користується розробленою системою електронних карток. В таблиці 4.2 описано поля даного класу.

Таблиця 4.2 – Опис класу «Лікар»

Поле	Тип	Пояснення
Ідентифікатор	Рядок	Є унікальним для кожного пацієнта. Дозволяє ідентифікувати його в межах системи
Ім’я	Рядок	Ім’я, що вказано у документі, наданому під час реєстрації
Прізвище	Рядок	Прізвище що вказано у документі, наданому під час реєстрації
По батькові	Рядок	По батькові що вказано у документі, наданому під час реєстрації

Поле	Тип	Пояснення
Лікарня	Посилання	Вказує на лікарню, у якій працює лікар
Спеціальність	Рядок	Вказує на спеціальність

Клас «Лікарня» (Clinic.cs) описує структуру основних даних лікарні, що зареєстрована у розробленій системі електронних карток. В таблиці 4.3 описано поля даного класу.

Таблиця 4.3 – Опис класу «Лікарня»

Поле	Тип	Пояснення
Ідентифікатор	Рядок	Є унікальним для кожного пацієнта. Дозволяє ідентифікувати його в межах системи
Країна	Рядок	Країна місцезнаходження клініки
Місто	Рядок	Місто місцезнаходження клініки
Вулиця	Рядок	Вулиця місцезнаходження клініки
Номер будівлі	Число	Номер будівлі місцезнаходження клініки
Приватна	Булево	Вказує чи є клініка державною чи приватною

Клас «Робочий день» (WorkDay.cs) описує структуру робочого дня у клініці, що зареєстрована у розробленій системі електронних карток. В таблиці 4.4 описано поля даного класу.

Таблиця 4.4 – Опис класу «Робочий день»

Поле	Тип	Пояснення
Ідентифікатор	Рядок	Є унікальним для кожного пацієнта. Дозволяє ідентифікувати його в межах системи
Клініка	Посилання	Вказує відношення до певної клініки
Дата	Дата	Дата у форматі число, місяць та рік

Клас «Прийом» (Reseption.cs) описує структуру окремого прийому пацієнта лікарем. Він є частиною історії хвороби пацієнта у клініці, що зареєстрована у розробленій системі електронних карток. В таблиці 4.5 описано поля даного класу.

Таблиця 4.5 – Опис класу Прийом

Поле	Тип	Пояснення
Ідентифікатор	Рядок	Є унікальним для кожного пацієнта. Дозволяє ідентифікувати його в межах системи

Поле	Тип	Пояснення
Робочий день	Посилання	Вказує відношення до певної клініки
Лікар	Посилання	Вказує на лікаря, що провів прийом
Пацієнт	Посилання	Вказує на пацієнта, що звернувся за допомогою
Діагноз	Рядок	Діагноз, що був поставлений пацієнту
Лікування	Рядок	Описує препарати та способи лікування хвороби
Примітка	Рядок	Додаткові дані, записи, тощо, що зроблені лікарем
Час	Дата	Час у форматі години та хвилини. Вказує момент часу у який було проведено прийом

4.1.2 База даних

База даних (БД) – сукупність даних, збережених за певними правилами. У більш вузькому розумінні він охоплює цифрові дані, зібрані відповідно до правил, прийнятих для даної комп’ютерної програми, спеціалізованої для збору та обробки цих даних. Така програма (часто програмний пакет) називається "системою управління базами даних" (СУБД).

Розроблення бази даних для системи, що реалізується, є одним із головних пунктів. Адже основною функцією системи є зберігання даних по

здоров’я пацієнтів. Класичним методом розробки БД є опис та створення таблиць за допомогою СУБД та мови структурованих запитів. В більшості випадків, таблиці БД проектуються на основі об’єктів, що беруть участь у роботі програми, а стовпці таблиць повністю повторюють поля класів. Враховуючи ці особливості, не дивно, що було розроблено набір інструментів, що дозволяють розробнику абстрагуватись від конкретики створення бази даних та управління нею, а використовувати тільки можливості мови програмування. Для мови C# таким інструментом є Entity Framework. Даний набір технологій дозволяє створити таблиці у базі даних та співпрацювати з нею, за допомогою уже створених об’єктних моделей.

Розглянемо роботу Entity Framework в межах нашої системи.

```
1   using Microsoft.EntityFrameworkCore;
2   using System;
3   using System.Collections.Generic;
4   using System.Linq;
5   using System.Threading.Tasks;
6
7   namespace PatientCard.Models
8   {
        Ссылок: 13
9       public class PatientCardContext: DbContext
10      {
            Ссылок: 0
11          public PatientCardContext(DbContextOptions<PatientCardContext> options)
12              : base(options)
13          {
14          }
15
            Ссылок: 6
16          public DbSet<Clinic> Clinics { get; set; }
            Ссылок: 6
17          public DbSet<Doctor> Doctors { get; set; }
            Ссылок: 6
18          public DbSet<Patient> Patients { get; set; }
            Ссылок: 6
19          public DbSet<WorkDay> WorkDays { get; set; }
            Ссылок: 6
20          public DbSet<Reception> Receptions { get; set; }
21
22
23      }
24  }
25
```

Рисунок 4.1 – Створення таблиць бази даних

На рисунку 4.1 зображено, як за допомогою функцій Entity Framework створюються таблиці у базі даних з використанням класів, що були створені на попередньому етапі. Тепер, усі новостворені об’єкти будуть попадати у базу даних, і всі зміни будуть відображені у ній.

Так само для того, щоб база даних повноцінно функціонувала, необхідно вказати систему управління базами даних, яка буде використовуватися, і вказати назву БД. Цей проект використовує СУБД Microsoft SQL (рисунок 4.2).

```
public void ConfigureServices(IServiceCollection services)
{
    services.AddDbContext<PatientCardContext>(opt =>
       opt.UseInMemoryDatabase("PatientCard"));
    services.AddControllers();
}
```

Рисунок 4.2 – Налаштування СУБД

4.1.3 API

Уже на даному етапі система здатна функціонувати, проте ніяких дій з даними не буде відбуватись і сервер не зможе прийняти ніяких команд від серверу. Для організації доступу до даних та управління взаємодії із даними в цілому існує окремий вид класів, що називають контролерами. Даний вид класів будується на основі класів моделей і кожен з контролерів відповідає за роботу з окремою моделлю. Наприклад, для класу Лікар (Doctor) створюється клас DoctorController.

У класі-контролері DoctorController описуються команди для отримання, редагування, запису та видалення даних типу Doctor. Для

кожного методу, за допомогою анотацій, описується свій шлях доступу та сама процедура або функція обробки команди.

На рисунку 4.3 зображено основні складові команди класу контролеру. Цифра 1 вказує на шлях доступу до команди, цифра 2 – назва процедури обробника, а цифра 3 – результат функціонування методу.

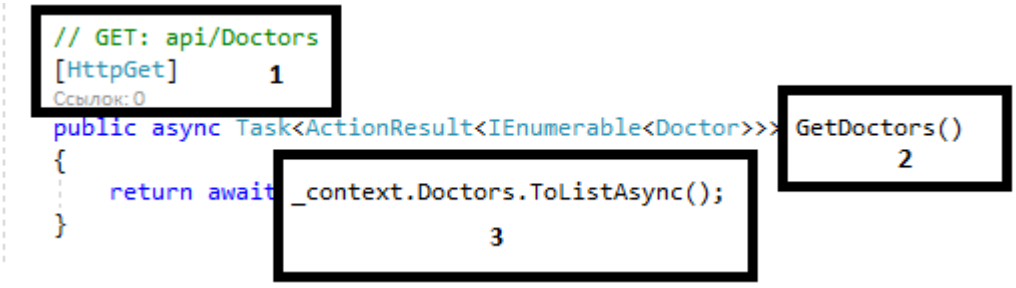


Рисунок 4.3 – Опис складових команди REST API

Для кожної моделі може бути виконаний повний набір операцій CRUD (create, read, update, delete), тобто отримати, створити, змінити та видалити дані в базі даних. Можуть бути використані такі команди:

– GET: api/Doctors – повертає перелік усіх об’єктів, записаних у БД в таблицю Doctors (рисунок 4.4);

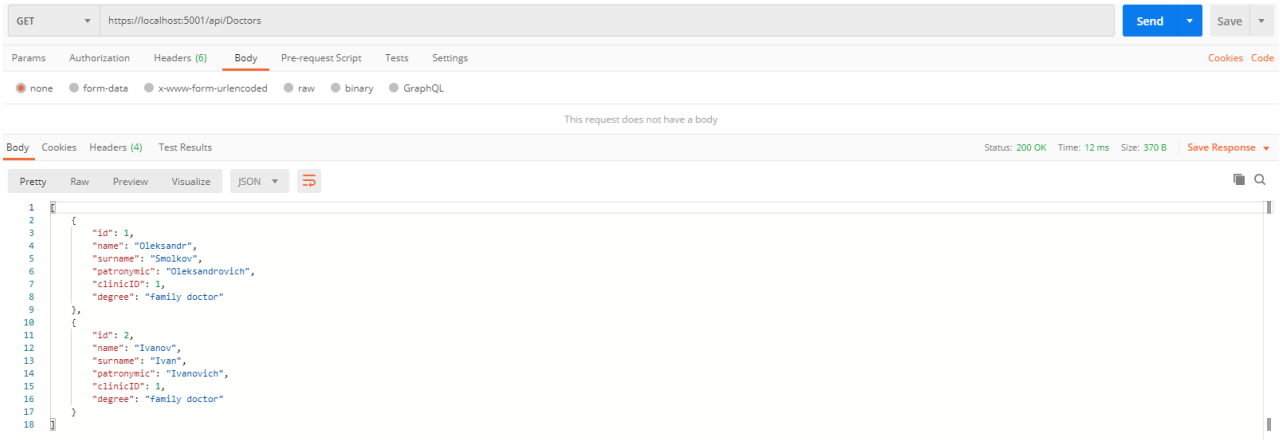


Рисунок 4.4 – Виконання команди // GET: api/Doctors

– GET: api/Doctors/1 – повертає об’єкт, що записано у БД в таблиці Doctors з унікальним ідентифікатором 1 (рисунок 4.5);

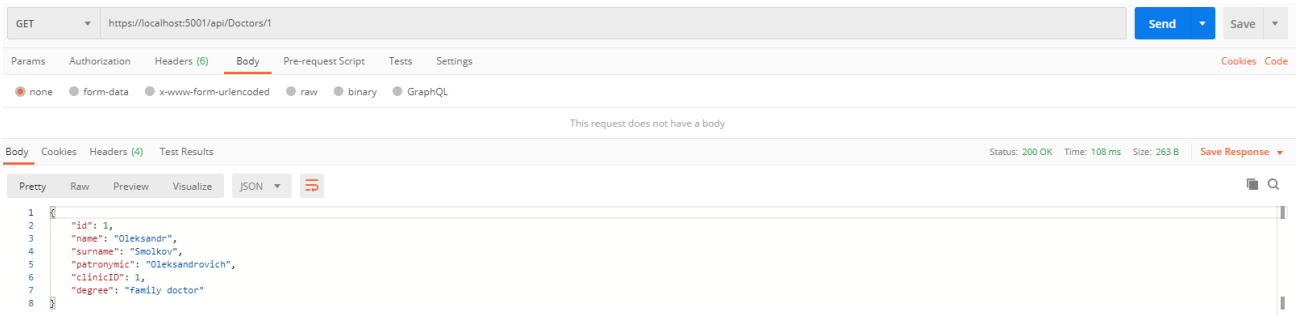


Рисунок 4.5 – Виконання команди // GET: api/Doctors/1

– PUT: api/Doctors/1 – редагує об’єкт що записано у БД в таблиці Doctors з унікальним ідентифікатором 1 (рисунок 4.6);

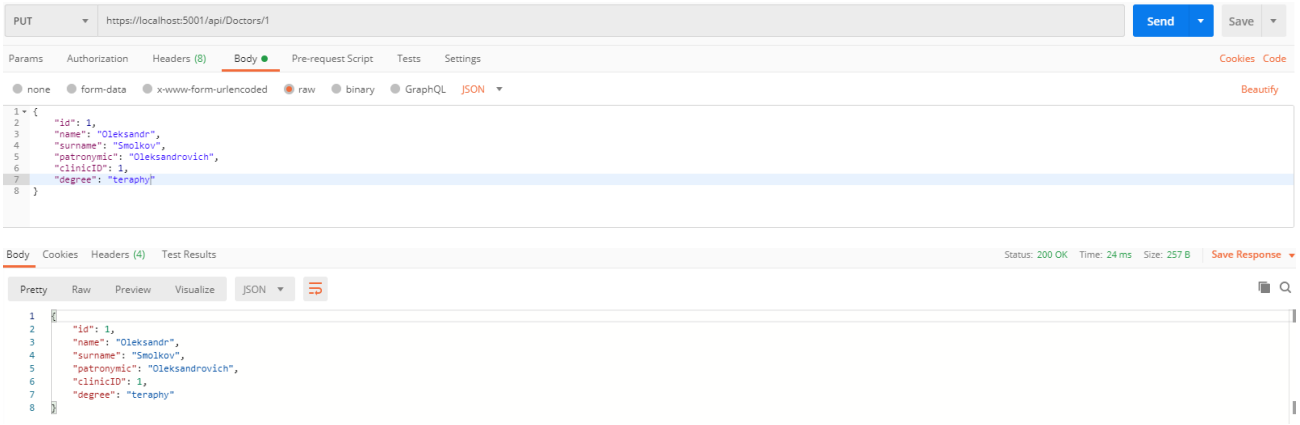


Рисунок 4.6 – Виконання команди // PUT: api/Doctors/1

– POST: api/Doctors – створює новий об’єкт з переданим описом об’єкту в тілі запиту, та записує його до БД (рисунок 4.7);

– DELETE: api/Doctors/1 – видаляє об’єкт, що записано у БД в таблиці Doctors з унікальним ідентифікатором 1 (рисунок 4.8).

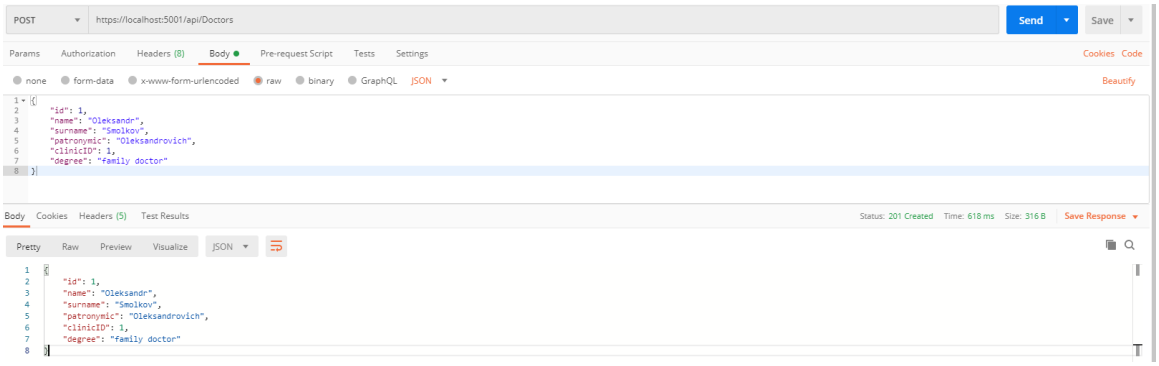


Рисунок 4.7 – Виконання команди // POST: api/Doctors

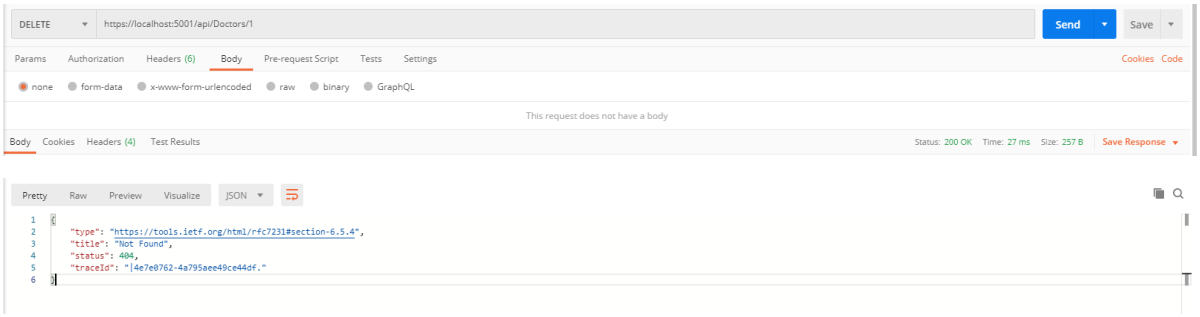


Рисунок 4.8 – Виконання команди // DELETE: api/Doctors/1

На рисунку 4.9 зображено процедури класу контролеру.

```
// GET: api/Doctors
[HttpGet]
Ссылок: 0
public async Task<ActionResult<IEnumerable<Doctor>>> GetDoctors()...

// GET: api/Doctors/5
[HttpGet("{id}")]
Ссылок: 0
public async Task<ActionResult<Doctor>> GetDoctor(long id)...

// PUT: api/Doctors/5
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see https://aka.ms/RazorPagesCRUD.
[HttpPut("{id}")]
Ссылок: 0
public async Task<IActionResult> PutDoctor(long id, Doctor doctor)...

// POST: api/Doctors
// To protect from overposting attacks, please enable the specific properties you want to bind to, for
// more details see https://aka.ms/RazorPagesCRUD.
[HttpPost]
Ссылок: 0
public async Task<ActionResult<Doctor>> PostDoctor(Doctor doctor)...

// DELETE: api/Doctors/5
[HttpDelete("{id}")]
Ссылок: 0
public async Task<ActionResult<Doctor>> DeleteDoctor(long id)...
```

Рисунок 4.9 – Команди класу DoctorController

4.2 Розроблення застосунку клієнтської частини системи

Враховуючи той факт, що система повинна бути у швидкому доступі для користувача, клієнтський застосунок найкраще реалізувати для мобільних платформ, таких як Android та IOS. Процес реалізації клієнтського застосунку розбито на два етапи: розроблення дизайну та реалізація самого застосунку.

4.2.1 Розроблення дизайну клієнтського застосунку

За допомогою застосунку Figma було розроблено дизайн усіх сторінок клієнтського застосунку. Всього застосунок складається з шести сторінок. В залежності від ролі користувача, деякі елементи можуть бути приховані та недоступні для використання. Застосунок Figma дозволяє отримати код розробленої сторінки у форматі XML, що дозволить під час реалізації зекономити час на створення візуальної частини застосунку.

Користувацький інтерфейс складається з п’яти основних сторінок:

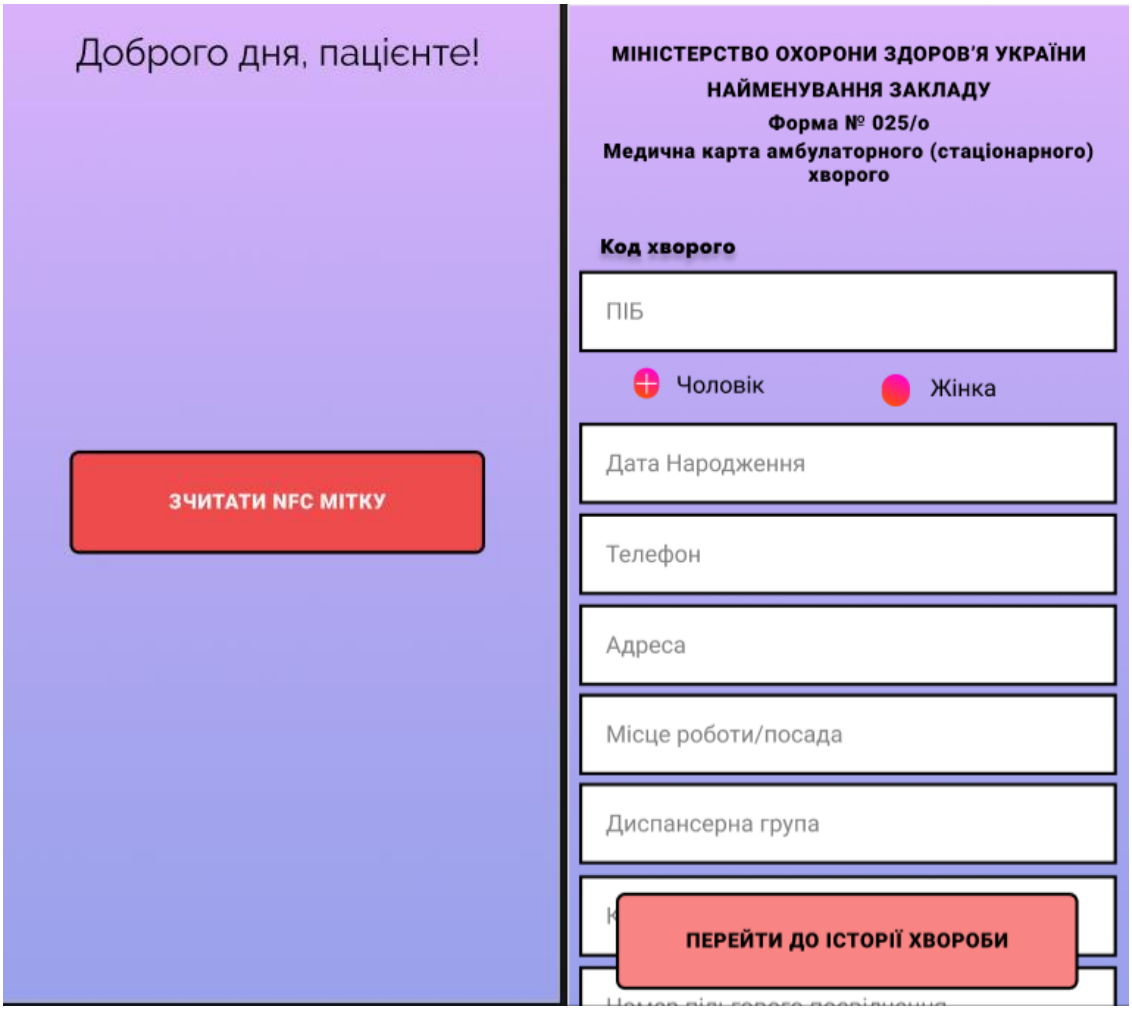
Сторінка входу (рисунок 4.10 (а). На цій сторінці зображено логотип застосунку та доступні два поля вводу: логіну та паролю користувача. За цими даними користувач авторизується в системі і в залежності від його ролі отримує певний рівень доступу.

Головна сторінка застосунку. Дана сторінка містить кнопку зчитування даних з NFC мітки та поле вводу логіну для ручного пошуку пацієнта у інтерфейсі користувача із роллю лікар (рисунок 4.10 (б). У інтерфейсі пацієнта, ручний пошук відсутній, адже пацієнт може зчитати тільки свою мітку (рисунок 4.11 (а).

а) б)

Рисунок 4.10 – Графічний інтерфейс: а) сторінка входу; б) головна сторінка інтерфейсу лікаря.

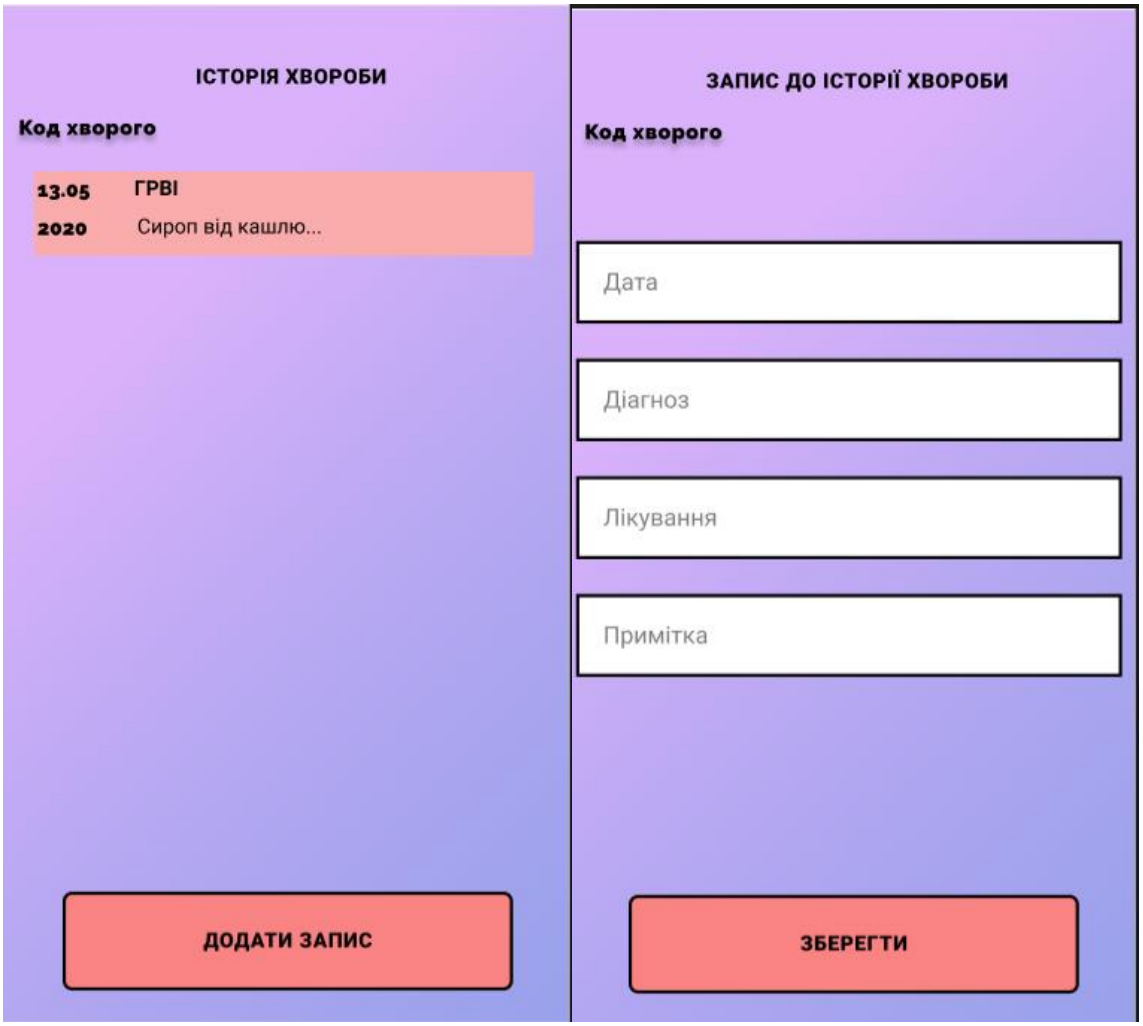
Сторінка основних даних пацієнта (рисунок 4.11 (б). Дана сторінка є цифровим аналогом форми №025/о. Вона містить інформацію про медичний заклад, у якому зареєстрований пацієнт, код хворого, стать, дату народження, номер телефону, адресу, місце роботи та посаду, інформацію про диспансерну групу, контингенти, номер пільгового посвідчення, групу крові, дату взяття на облік, дату зняття з обліку, а також кнопку переходу до історії хвороби.



а) б)

Рисунок 4.11 – Графічний інтерфейс: а) головна сторінка інтерфейсу пацієнта; б) сторінка основних даних.

Сторінка з історією хвороби пацієнта (рисунок 4.12 (а). На даній сторінці в оберненій хронологічній послідовності виводиться історія хвороби пацієнта. В інтерфейсі лікаря доступна кнопка додавання нового запису до історії хвороби, пацієнт же може тільки переглядати історію хвороби. В обох інтерфейсах, після натискання на окремий запис, користувач потрапить на сторінку редагування/перегляду запису історії хвороби.



а) б)

Рисунок 4.12 – Графічний інтерфейс: а) історія хвороби; б) редагування запису історії хвороби.

Сторінка додавання/редагування/перегляду запису історії хвороби (рисунок 4.12 (б). На ній зображена інформація про дати прийому, діагноз, лікування та примітку. Для користувача із роллю пацієнт, елементи даної сторінки недоступні для редагування. Лікар же може вносити зміни в існуючий запис або створити новий.

4.2.2 Реалізація клієнтського застосунку

Враховуючи розроблений раніше серверний застосунок та набір команд, що він надає, а також інтерфейс клієнтського застосунку, реалізуємо програмний додаток, що дозволить зчитувати дані міток NFC та надсилати команди серверу для отримання необхідних даних.

Для розроблення клієнтського застосунку було обрано набір інструментів Xamarin.Forms. Дана технологія дозволила реалізувати одну логіку для обох операційних систем, що є зараз найбільш поширеними для мобільних платформ: Android та IOS. Розглянувши структуру рішення (рисунок 4.13), можна помітити три проєкти. AnyClinic – це основний проєкт в рішенні, у ньому описується логіка, що є спільною для обох платформ. Під час побудови рішення, на основі головного проєкту, створюються виконувані файли для платформ Android – AnyClinic.Android та IOS – AnyClinic.IOS.

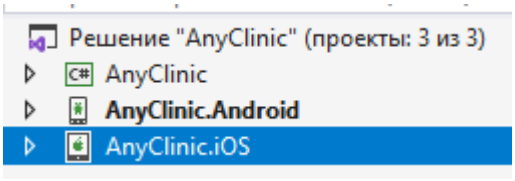


Рисунок 4.13 – Структура рішення клієнтського застосунку

Саме рішення є набором сторінок, кожна з яких складається з двох складових (рисунок 4.14):

– файл розмітки XAML – описує зовнішній вигляд сторінки. Вказує які саме елементи знаходяться на сторінці, їх положення, колір, поведінку тощо;

– файл опису логіки сторінки. Він має розширення cs, у ньому описана логіка подій, що можуть відбуватися на сторінці. Будь то натискання на кнопку чи введення певних даних в поле сторінки.

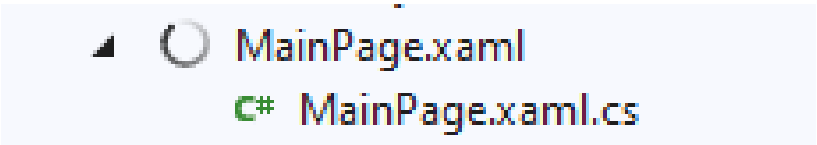


Рисунок 4.14 – Структура окремої сторінки

У файлі розмітки описується дизайн майбутньої сторінки, об’єкти що на ній розташовуються, їх позиції на сторінці, кольори, шрифти тощо. Для прикладу розглянемо головну сторінку для інтерфейсу пацієнта. ЇЇ зовнішній вигляд продемонстровано на рисунку 4.10 (а), а код розмітки на рисунку 4.15.

З точки зору програми, ця сторінка не є головною, адже головною у є сторінка авторизації, саме вона викликається під час першого запуску застосунку. Тому, як наслідок, ця сторінка є сторінкою контенту, відповідно тегом верхнього рівня є ContentPage, а його атрибутами є посилання на схему, що використовуються. Особливої уваги потребує атрибут «x:Class», він вказує шлях у просторі імен до класу-обробника подій, що відбуваються на даній сторінці. Вкладеними елементами є ContentPage.Content – він описує контент сторінки, тобто те що буде відображатись. В середині нього, вкладено елемент StackLayout, який вказує на групу елементів, у даному випадку – Label (надпис) та Button (кнопка).

```
<?xml version="1.0" encoding="utf-8" ?>
<ContentPage xmlns="http://xamarin.com/schemas/2014/forms"
             xmlns:x="http://schemas.microsoft.com/winfx/2009/xaml"
             xmlns:d="http://xamarin.com/schemas/2014/forms/design"
             xmlns:mc="http://schemas.openxmlformats.org/markup-compatibility/2006"
             mc:Ignorable="d"
             x:Class="AnyClinic.PageRead">
    <ContentPage.Content>
        <StackLayout BackgroundColor="#9A58FF">
            <Label Text="Доброго дня, пацієнте!"
              HorizontalOptions="CenterAndExpand"
              VerticalOptions="StartAndExpand"
              FontSize="Large"
              FontAttributes="Bold" />
            <Button Text="NFC"
                HorizontalOptions="CenterAndExpand"
                VerticalOptions="CenterAndExpand"
                BackgroundColor="#ED4C4C"
                BorderColor="#000000"
                BorderWidth="1"
                FontAttributes="Bold"
                FontSize="Large"
                Clicked="OnButtonClicked"/>
        </StackLayout>
    </ContentPage.Content>
</ContentPage>
```

Рисунок 4.15 – Файл PageRead.xaml.

Оскільки, після натискання на кнопку, повинна виконатись якась дія, у елемента Button існує атрибут Clicked, в якому вказується назва процедури обробника у класі обробнику. Приклад реалізації такого механізму показано на рисунку 4.16.

Процедура OnButtonClicked, що приймає до виконання два агрументи: об’єкт відправник та подію що відбулась, створює об’єкт класу кнопка, на основі того об’єкту, що був переданий до процедури. Далі, встановлює текст кнопки на “Reading” та за допомогою основного класу застосунку, включає режим зчитування мітки.

Описувати зовнішній вигляд сторінки можна і динамічно формуючи її за допомогою файлу з розширенням cs, але більш правильним, з точки

зору розроблення вважається розділення візуальної частини та логіки сторінки.

```
using Xamarin.Forms;
using Xamarin.Forms.Xaml;

namespace AnyClinic
{
    [XamlCompilation(XamlCompilationOptions.Compile)]
    Ссылок: 5
    public partial class PageRead : ContentPage
    {
        ссылка: 1
        public PageRead()
        {
            InitializeComponent();
        }
        Ссылок: 0
        private void OnButtonClicked(object sender, System.EventArgs e)
        {
            Button button = (Button)sender;
            button.Text = "Reading";
            App.TurnNFCReadingOn();
        }
    }
}
```

Рисунок 4.16 – Файл PageRead.xaml.cs.

Розглянемо роботу застосунку поетапно.

Після запуску застосунку, на екрані користувача висвітлюється вікно авторизації, що зображено на рисунку 4.10 (а), з двома доступними для вводу даних полями: Логін та Пароль. Після введення даних, в залежності від ролі користувача він перенаправляється на сторінку зчитування (рисунки 4.10 (б), 4.11 (а). Перехід відбувається за допомогою вбудованих механізмів переходу для навігаційних сторінок (рисунок 4.17).

```
MainPage = new NavigationPage(new MainPage());
```

```
Navigation.PushAsync(new PageRead());
```

а) б)

Рисунок 4.17 – Механізми навігації по сторінках. а) Ініціалізація навігаційної сторінки. б) Функція переходу на іншу сторінку

На сторінці зчитування, після натискання на кнопку, вмикається механізм зчитування мітки. Телефон в асинхронному режимі шукає мітку, при знаходженні її, зчитує та передає дані у функцію, яка генерує сторінку перегляду інформації про основні дані пацієнта (рисунок 4.11 (б). Механізм роботи з міткою зображено на рисунку 4.18.

Сторінка з основними даними пацієнта є навігаційною і підтримує перехід на сторінку з історією хвороби (рисунок 4.12 (а), яка в свою чергу дозволяє виконати перехід до редагування окремого запису в історії хвороби(рисунок 4.12 (б)).

```
try
{
    isoDep.Connect();

    var aidLength = (byte)(SAMPLE_LOYALTY_CARD_AID.Length / 2);
    var aidBytes = StringToByteArray(SAMPLE_LOYALTY_CARD_AID);
    var command = SELECT_APDU_HEADER
        .Concat(new byte[] { aidLength })
        .Concat(aidBytes)
        .ToArray();

    var result = isoDep.Transceive(command);
    var resultLength = result.Length;
    byte[] statusWord = { result[resultLength - 2], result[resultLength - 1] };
    var payload = new byte[resultLength - 2];
    Array.Copy(result, payload, resultLength - 2);
    var arrayEquals = SELECT_OK_SW.Length == statusWord.Length;

    if (Enumerable.SequenceEqual(SELECT_OK_SW, statusWord))
    {
        var msg = Encoding.UTF8.GetString(payload);
        await App.DisplayAlertAsync(msg);
    }
```

Рисунок 4.18 – Функція зчитування даних з мітки NFC

Звичайно, дана процедура не може бути включена постійно, адже мобільний телефон може стикатися з мітками у повсякденній роботі, а випадкові спрацювання будуть неприємними для користувача. Саме тому у відповідному класі застосунку було створено набір процедур направлених на управління використання адаптеру NFC (рисунок 4.18).

Процедура EnableReaderMode включає використання адаптеру NFC, DisableReaderMode ж у свою чергу, виключає його використання. Ці процедури використовуються після натискання на кнопку зчитати, на головній сторінці застосунку, а також у процедурах OnPause та OnResume, які відповідають за припинення роботи програми після її згортання, та відновлення роботи програми після її розгортання (рисунок 4.19).

```
public void EnableReaderMode()
{
    var nfc = NfcAdapter.GetDefaultAdapter(this);
    if (nfc != null) nfc.EnableReaderMode(this, cardReader, NfcReaderFlags.NfcA, null);

}

ссылка: 1
public void DisableReaderMode()
{
    var nfc = NfcAdapter.GetDefaultAdapter(this);
    if (nfc != null) nfc.DisableReaderMode(this);
}

Ссылок: 0
protected override void OnPause()
{
    base.OnPause();
    DisableReaderMode();
}

Ссылок: 0
protected override void OnResume()
{
    base.OnResume();
    EnableReaderMode();
}
```

Рисунок 4.19 – Процедури управління адаптером NFC

4.3 Висновки до розділу 4

У даному розділі, на основі проведеного проектування, було розроблено основні складові системи: серверний та клієнтський застосунки. Було реалізовано об’єктну структуру серверного застосунку, розроблено базу даних та інтерфейс доступу до даних у вигляді REST API. Було спроектовано дизайн клієнтського застосунку та реалізовано сам застосунок за допомогою технологій Xamarin.Forms. Реалізовано механізми роботи з модулями комунікацій ближнього поля , вбудованими в сучасні смартфони.

ВИСНОВКИ

На основі проведеного аналізу, порівняння способів ведення медичного обліку, та беручи до уваги досвід інших країн, можна дійти висновку, що процес використання електронних медичних карток, є наступним кроком у розвитку медичної сфери України.

Було спроектовано та розроблено інформаційну систему «Електронна медична картка», що дозволяє вести облік стану здоров'я пацієнтів у будь який час, використовуючи смартфон та доступ до інтернету. А також надає швидкий доступ до основних даних стану здоров'я пацієнта завдяки технологіям комунікацій ближнього поля.

Мета, що була поставлена на початку роботи, була досягнута, а також виконані поставлені задачі, а саме:

– аналіз предметної області;

– огляд існуючих рішень та виділення основних аспектів розробки;

– вибір технологій та інструментів реалізації системи;

– розроблення інформаційної системи;

Система, що була розроблена, може стати повноцінною заміною для медичних карток.

У подальшому, шляхами розвитку можуть стати процеси використання різних носіїв даних, створення інтерфейсів користувачів різного профілю, хмарна система зберігання даних, нейронна мережа діагностування.

ПЕРЕЛІК ВИКОРИСТАНИХ ДЖЕРЕЛ

1. КАБІНЕТ МІНІСТРІВ УКРАЇНИ ПОСТАНОВА від 18 вересня 2019 р. № 856 Київ Питання Міністерства цифрової трансформації [Електронний ресурс] : Режим доступу: https://zakon.rada.gov.ua/laws/show/856-2019-п

2. МОЗ України: Що було, є і буде [Електронний ресурс] : Режим доступу: https://moz.gov.ua/uploads/2/13773-transition_book_healthcare.pdf

3. Опалько В.В. Реформування системи охорони здоров'я в Україні // Вісник Черкаського університету Науковий журнал. – № 207, 2011

4. Top Ambulatory Electronic Health Records Vendors// Black Book™ a division of Brown-Wilson Group. – 2014

5. C# – лучший язык для мобильной разработки [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/touchinstinct/blog/189060/

6. [Електронний ресурс] Режим доступу до ресурсу: https://www.softwareadvice.com/medical/pcc-profile/

7. [Електронний ресурс] Режим доступу до ресурсу: https://www.softwareadvice.com/medical/advancedmd-advancedemr-profile/

8. [Електронний ресурс] Режим доступу до ресурсу: https://www.softwareadvice.com/medical/advancedmd-advancedemr-profile/

9. Достоинства и недостатки Xamarin [Електронний ресурс] – Режим доступу до ресурсу: https://habr.com/ru/company/microsoft/blog/415833/

10. Top 10 advantages of C# [Електронний ресурс] – Режим доступу до ресурсу: http://proprogrammershub.blogspot.com/2016/04/top-10-advantages-of-c.html

11. Прайс Марк. C# 7 и .NET Core. Кросс-платформенная разработка для профессионалов. 3-е издание / Прайс Марк.: "Издательский дом ""Питер""", 2018. ISBN 5446105168, 9785446105168. – 595 -622 с.

12. Xamarin и кросс-платформенная разработка [Електронний ресурс] – Режим доступу до ресурсу: https://metanit.com/sharp/xamarin/1.1.php

13. Mark Masse. REST API Design Rulebook: Designing Consistent RESTful Web Service Interfaces / Mark Masse.: "O'Reilly Media, Inc.", 2011. ISBN 1449319904, 9781449319908. – 5-6 с.

14. Kevin Hoffman. Building Microservices with ASP.NET Core: Develop, Test, and Deploy Cross-Platform Services in the Cloud / Mark Masse.: "O'Reilly Media, Inc.", 2017. ISBN 1491961708, 9781491961704. – 4-11 с.

15. ASP.NET Core Overview [Електронний ресурс] – Режим доступу до ресурсу: https://www.tutorialsteacher.com/core/aspnet-core-introduction

16. Vedat Coskun, Kerem Ok, Busra Ozdenizci. Professional NFC Application Development for Android / Vedat Coskun, Kerem Ok, Busra Ozdenizci. : John Wiley & Sons, 2013. ISBN 1118380568, 9781118380567. – 5-21 с.

17. Vedat Coskun, Kerem Ok, Busra Ozdenizci. Near Field Communication / Vedat Coskun, Kerem Ok, Busra Ozdenizci.: John Wiley & Sons, 2012. ISBN: 978-1119971092, 1119971098.

18. Tom Igoe, Don Coleman, Brian Jepson. Beginning NFC: Near Field Communication with Arduino, Android, and PhoneGap / Tom Igoe, Don

Coleman, Brian Jepson.: "O'Reilly Media, Inc.", 2014. ISBN 1449324126, 9781449324124. – 11-15 с.

19. System engineering fundamentals defence// ACQUISITION UNIVERSITY PRESS FORT BELVOIR, 2001, VIRGINIA 22060-5565 – 47-100 с.

20. IDEF3 diagrams. [Електронний ресурс] – Режим доступу до ресурсу: https://www.conceptdraw.com/examples/idef3-diagram-software

21. What is use-case diagram. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-use-case-diagram/

22. All you need to know about class diagram. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/about-state-diagrams/

23. What is sequency diagram. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-sequence-diagram/

24. What is component diagram. [Електронний ресурс] – Режим доступу до ресурсу: https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-component-diagram/

25. What is goal tree. [Електронний ресурс] – Режим доступу до ресурсу: https://hohmannchris.wordpress.com/2014/03/07/what-is-a-goal-tree/

26. Distributed Application Architecture. [Електронний ресурс] – Режим доступу до ресурсу: https://web.archive.org/web/20110406121920/http:/java.sun.com/developer/Books/jdbc/ch07.pdf

ДОДАТОК А

ПрАТ «СК «АЛЬФА СТРАХУВАННЯ»
01011, Україна, м. Київ, вул. Рибальська, 22
Тел.: 044 499 77 66
Факс: 044 499 77 60
e-mail: office@alfaic.ua
www.alfaic.ua

вих. № 345/02
від 06 / 04 / 2020 р.

ЗАЯВКА

організації на виконання дипломної роботи

Завідуючому кафедрою
автоматики та управління в технічних
системах факультету інформатики та
обчислювальної техніки
Національного технічного університету
України «Київський політехнічний
інститут імені Ігоря Сікорського»
професору
Олександру РОЛІКУ

Адміністрація організації *ПРИВАТНЕ АКЦІОНЕРНЕ ТОВАРИСТВО «СТРАХОВА КОМПАНІЯ «АЛЬФА СТРАХУВАННЯ»*
(назва організації)

просить доручити студенту ***Смолькову Олександру Олександровичу(ІТ-61)***
(Прізвище, Ім'я, По-батькові студента(ки))

Розробити дипломну роботу на тему
Інформаційна система «Електронна медична картка»
(назва теми дипломної роботи (дипломного проекту, магістерської дисертації))

у зв'язку з
необхідністю забезпечення ефективного медичного лікування і слідкування за станом здоров'я працівників

(обґрунтування заявки на виконання ДР (ДП, МД))

Директор Вінницького
регіонального управління
ПрАТ «СК «Альфа
Страхування»
М.П.

(підпис) (*Т. І. Малильо*)

Дата (*06.04.2020р.*)

ДОДАТОК Б

ПрАТ «СК «АЛЬФА СТРАХУВАННЯ»
01011, Україна, м. Київ, вул. Рибальська, 22
Тел.: 044 499 77 66
Факс: 044 499 77 60
e-mail: office@alfaic.ua
www.alfaic.ua

АКТ ВПРОВАДЖЕННЯ
в роботу сторонньої організації

вих. № 945/02
від 07 / 06 / 2020 р.

«ЗАТВЕРДЖУЮ»
Директор
ПрАТ « СК «Альфа Страхування»
Вінницьке регіональне управління
(назва організації)
Малильо Тетяна Іванівна
«7» червня 2020 р.

Результати дослідження студента(ки) ***Смолькова Олександра Олександровича (гр. ІТ-61)***
(Прізвище, Ім'я, По-батькові)

на тему «Інформаційна система Електронна медична картка»
(назва дипломної роботи))

що затверджена наказом по КПІ імені Ігоря Сікорського від «7» *травня* 2020 р. за № 1081-с

та виконана під керівництвом к.т.н., доц. каф. АУТС Писаренко Андрій Володимирович
(посада, науковий ступінь, звання, Прізвище І.П. наукового керівника)

впроваджені в діяльність організації *ПрАТ « СК «Альфа Страхування»*
(назва організації, юридична адреса)

01001, м. Київ, вул. Рибальська, 22, Башта №5

Основні результати, що були впроваджені:
1. Створено серверне програмне забезпечення із базою даних для зберігання та обробки даних щодо здоров'я працівників компанії.
2. Створено Android застосунок для моніторингу стану здоров`я.

Даний акт не є підставою для одержання винагороди.

Директор Вінницького РУ
ПрАТ « СК «Альфа Страхування»
М.П.

(підпис) (*Т.І. Малильо*)

«7» червня 2020р.